Type Inference for a Distributed π-Calculus*

Cédric Lhoussaine

University of Sussex, Brighton, UK
clh23@cogs.susx.ac.uk

Abstract. We study the type inference problem for a distributed π-calculus with explicit notions of *locality* and *migration*. Location types involve names that may be bound in terms. This requires some accurate new treatments. We define a notion of *principal typing*. We provide a formal description of sound and complete type inference algorithm.

1 Introduction

In wide area distributed systems, such as the Internet, sensitive administrative domains have to be protected against malicious agents, and tools are required for the analysis of security properties. In this paper we focus on this topic using Hennessy and Riely's distributed π-calculus Dπ [6]. It is based on the polyadic asynchronous π-calculus, involving explicit and simple notions of *locality* and *migration*. The distribution is one dimensional: in contrast with Djoin [5] or Mobile Ambients [4], locations do not contain sub-locations (as in π_ℓ [1]). As in Mobile Ambients, communication is purely local: only co-located processes can communicate. Mobility is *weak* in the sense that we migrate code instead of computations (as it is the case in Mobile Ambients and Djoin). Thus, this calculus provides a simple but powerful framework to model fundamental features of distributed computations.

In [6] a type system is proposed for Dπ. It does not only deal with arity mismatch of communications as sorts do for the polyadic π-calculus. It also investigates an important issue of distributed systems: the *controlled access to system resources*. There, a resource is represented by a communication channel bound to a particular location. Hence, a *location type* is the set of channels available to a process at a location. It is of the form:

$$\{a_1 : \gamma_1, \ldots, a_n : \gamma_n\}$$

where each a_i is a channel name and γ_i a *channel type*. Locations can be sent through channels that have a type $Ch(\psi)$ where ψ is a location type. For instance, a process knowing of a location ℓ with type $\{a : \gamma, b : \gamma'\}$ has permission to use channels a and b at ℓ and only these. In [2], this type system is required to guarantee the *message deliverability* property.

* Research supported by "MyThs: Models and Types for Security in Mobile Distributed Systems", EU FET-GC IST-2001-32617.

P. Degano (Ed.): ESOP 2003, LNCS 2618, pp. 253–268, 2003.

Even though it checks crucial properties, this type system assumes that all agents are well-typed. However, in networks such as the Internet, only few locations may be statically typed, and a dynamic type checking is necessary for agents coming from untyped or unknown locations. A fundamental question arises: does a type checking algorithm exist ?

We design a type inference algorithm *à la* ML for Dπ with undecorated terms (à la Curry). Usually, such an algorithm proceeds in two steps:

- given a term S and its *initial typing context* Γ (associating a type variable to each location name of S), generate type constraints involving type variables;
- produce a substitution μ of type variables for types solving the constraints.

If the algorithm succeeds, then the application of the substitution to the initial context $\mu\Gamma$ is a valid typing context for S. Otherwise, S is not typable. Despite this simple scheme, we have to face two major difficulties described below together with the solutions worked out in this paper.

Principal typing. Basically, a principal typing is a typing context which represents all possible types by ground instantiation of type variables. Unfortunately, this definition is not suitable here because our type system involves *subtyping* on location types.

Indeed, consider the term $\overline{a}\ell \mid a(k).P$ that sends the location name ℓ over the channel a and binds k to ℓ in P, and suppose that a has type $Ch(\{b:\gamma, c:\gamma'\})$. Then P is allowed to use *at most* b and c at the received location, that is k has a type $\psi \subseteq \{b:\gamma, c:\gamma'\}$. And, all location names sent through a, must have a type that declares *at least* b and c, that is ℓ has a type $\psi' \supseteq \{b:\gamma, c:\gamma'\}$. We see that the types of ℓ and k are related to the one of a, and all valid typings for that term have to satisfy this relation.

Therefore, we define a principal typing for a term S as a pair $(\Gamma; \mathcal{A})$ where Γ is a typing context involving type variables and $\mathcal{A}$ is a set of subtyping constraints. And an instance $\mu\Gamma$ is a valid typing for S if and only if μ satisfies $\mathcal{A}$. Thus, our algorithm generates not only equations of types but also inequations.

Dependant types. The second and most important difficulty comes from the fact that location types are so-called *dependant types*: they involve names that may be bound in terms. For instance, consider the term $Q = a(b).(\nu\ell)\, P$ that receives a name b, creates a location ℓ, and triggers P. Here b is bound to $(\nu\ell)\, P$, and ℓ may be given a type $\{b:\gamma\}$. If we create the location ℓ before the reception as in $R = (\nu\ell)\, a(b).P$, then ℓ cannot have a type where b occurs. Otherwise typing would not be preserved by α-equivalence of terms. However, the type inference algorithm cannot find a type for ℓ before the exploration of P where there is not anymore difference between the creation of ℓ before the reception of b and the reverse. Thus, a naive algorithm may assign types to untypable terms!

Our solution introduces a novel notion of *binding relation* that, intuitively, keeps track of the order in which channels are bound. It relates channel names and type variables such that if (a, α), is in the relation, then it is forbidden to substitute type variable α by a type in which name a occurs. In the first step

$$
\begin{array}{rll}
u, v \ldots ::= & a \mid \ell \mid a@\ell & \textit{values} \\
P, Q, R \ldots ::= & \mathbf{0} \mid \overline{a}u \mid a(u).P \mid (P \mid Q) \mid (\nu u)\, P \mid \mathsf{go}\,\ell.P \mid !P & \textit{processes} \\
S, T, \ldots ::= & \mathbf{0} \mid \ell[\![P]\!] \mid (S \mid T) \mid (\nu a@\ell)\, S \mid (\nu \ell)\, S & \textit{networks}
\end{array}
$$

Fig. 1. Syntax of terms

of the algorithm, this relation is updated each time a bound channel is met. For instance, consider R above, and suppose that ℓ is given a type variable α. When the algorithm treats the reception, it updates the binding relation by associating b to each current type variable. In particular, the pair (b, α) is added to the binding relation, thus preventing ℓ to be assigned a type where b occurs. In Q, ℓ is given a fresh type variable α when the creation is met, i.e. after the reception of b. Hence, α is not associated to b and may be substituted by $\{b : \gamma\}$.

We give a sound and complete type inference algorithm that produces a principal typing. We express our algorithm in the form of a rewriting system, which allows for neat formal proofs that can be found in [8].

Related works. Our calculus is actually a variant of the one in [6]: the latter uses a synchronous communication and an explicit typing (terms involve types). However, we could easily treat synchronous and decorated terms. For instance, consider the term $Q = (\nu a : \tau)\, P$. Applied to $(\nu a)\, P$, either our algorithm fails and Q is not typable. Or it succeeds and produces a type σ for a. And Q is typable if and only if τ is an instance of σ. Our location types are very similar to record types of [13,10,11,7]. However, using techniques developed in these papers, the use of a binding relation requires some accurate new treatments. To our knowledge, the type inference algorithm presented in this paper is the first one dealing with these kind of dependant types.

2 A Calculus with Localities

In this section we introduce our distributed calculus, we give the operational semantics and the syntax of types. For the sake of simplicity, this calculus is presented in its monadic version. Apart from that, it is the usual asynchronous π-calculus with some primitives for spatial distribution and migration of processes organised as a two-levels model: the processes (or thread) one and the network (or configuration) one. As in [6] communication is local, that is we cannot directly send a message from a location ℓ to a remote process at location k: we must migrate the message to k, and then we can communicate.

In order to state the syntax, we consider a denumerable set $\mathcal{N}$ of (*simple*) *names* which is assumed to be partitioned into two sets: the channel names $\mathcal{N}_{chan} = \{a, b, \ldots\}$ and the location names $\mathcal{N}_{loc} = \{\ell, k, \ldots\}$. The objects of communications may be *compound* that is a channel name a together with a location name ℓ denoted by $a@\ell$ and meaning "the channel a (to be) used at location ℓ". We can also abstract and restrict location names. The grammar of

terms is given in figure 1. We denote by $U, V \ldots$ a process or a network, $fn(U)$ (resp. $bn(U), nm(U)$) the set free (resp. bound, all) names occurring in U.

Before describing types, we briefly present the operational semantics in the "chemical style" of BERRY and BOUDOL [3]. To this aim, we define the *structural equivalence* as the least equivalence satisfying the commutative monoid laws for parallel composition, containing the α-equivalence and satisfying the following:

$$(\!(\nu u)\, U \mid V) \equiv (\!(\nu u)\, (U \mid V) \quad \text{if } subj(u) \notin fn(V)$$
$$\ell[\![(\nu u)\, P]\!] \equiv (\!(\nu u @ \ell)\, \ell[\![P]\!] \quad \text{if } subj(u) \neq \ell$$
$$!P \equiv (P \mid !P) \qquad \ell[\![P \mid Q]\!] \equiv \ell[\![P]\!] \mid \ell[\![Q]\!] \qquad U \equiv V \Rightarrow \mathbf{E}[U] \equiv \mathbf{E}[V]$$

where

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

and $\mathbf{E}$ is any *evaluation context*, defined by: $\mathbf{E} ::= \bullet \mid (\mathbf{E} \mid U) \mid (\!(\nu u)\, \mathbf{E} \mid \ell[\![\mathbf{E}]\!]$

As usual, in an evaluation context $\bullet$ stands for a "hole" and $\mathbf{E}[U]$ denotes the substitution of the hole for the term U in $\mathbf{E}$ providing that the resulting term is valid. The reduction is built upon two laws, that is the standard communication one plus a *law of movement* for migration:

$$(\overline{a}v \mid a(u).P) \;\rightarrow\; [v/u]P \qquad\qquad \ell[\![\mathsf{go}\, k.P]\!] \;\rightarrow\; k[\![P]\!]$$

up to structural equivalence and under evaluation context.

3 The Type System

Let $\mathcal{V} = \{\alpha, \beta, \ldots\}$ be a denumerable set of type variables partitioned into three sets: the set of *general type variables* $(t, t', \ldots)$, the set of *channel type variables* $(h, h', \ldots)$, and the set of location type variables (or *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:

$$\begin{array}{ll} \tau, \sigma, \ldots ::= \psi \mid \gamma \mid \gamma@\psi \mid t & \textit{types} \\ \gamma, \delta, \ldots ::= Ch(\tau) \mid h & \textit{channel types} \\ \psi, \phi, \ldots ::= \{a : \gamma, \psi\} \mid \{\} \mid \rho_L & \textit{location types} \end{array}$$

We denote by $var(\tau)$ the set of the type variables occurring in τ. We denote by $\tau, \gamma, \psi, \ldots$ the ground types that is the types τ such that $var(\tau) = \emptyset$. We will often note $\{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \psi\}$ for $\{a_1 : \gamma_1, \{\ldots, \{a_n : \gamma_n, \psi\}\}\ldots\}$. Typing a location name means: "recording the names and types of channels on which a communication is possible inside the location". That is, a location type is a record of channel names together with their types: this a *dependant type* since it contains terms, viz. channel names. Extension of location type is achieved by means of a row variable that may be substituted with a location type. A location type that ends with a row variable is called an *extensible location type*. In the latter, the row variable is obtained by means of the partial function ρvar (e.g. $\rho var(\{a : \gamma, \rho_L\}) = \rho_L$). We also denote the set of names typed in a

$$\frac{}{k : \psi \vdash_\ell k : \psi} \qquad \frac{}{\ell : \{a : \gamma\} \vdash_\ell a : \gamma} \qquad \frac{}{\ell : \{a : \gamma, \psi\} \vdash_k a@\ell : \gamma@\psi} \qquad \frac{\Gamma \vdash_\ell u : \tau}{\Gamma, \Delta \vdash^W_\ell u : \tau}$$

Fig. 2. Type system for names

$$\frac{\Gamma, \Delta \vdash_\ell P \quad , \quad \Delta \vdash_\ell u : \tau}{\ell : \{a : Ch(\tau)\}, \Gamma \vdash_\ell a(u).P} \qquad \frac{\ell : \{a : \gamma\}, \Gamma \vdash_\ell P}{\Gamma \vdash_\ell (\nu a)\, P} \qquad \frac{k : \{a : \gamma\}, \Gamma \vdash_\ell P}{\Gamma \vdash_\ell (\nu a@k)\, P} \qquad \frac{k : \psi, \Gamma \vdash_\ell P}{\Gamma \vdash_\ell (\nu k)\, P}$$

$$\frac{\Gamma \vdash^W_\ell u : \tau}{\ell : \{a : Ch(\tau)\}, \Gamma \vdash_\ell \overline{a}u} \qquad \frac{\Gamma \vdash_\ell P \quad , \quad \Gamma \vdash_\ell Q}{\Gamma \vdash_\ell P \mid Q} \qquad \frac{\Gamma \vdash_k P}{\Gamma \vdash_\ell \mathtt{go}\, k.P} \qquad \frac{\Gamma \vdash_\ell P}{\Gamma \vdash_\ell !P} \qquad \frac{}{\Gamma \vdash_\ell \mathbf{0}}$$

Fig. 3. Type system for processes

$$\frac{}{\Gamma \vdash \mathbf{0}} \qquad \frac{\Gamma \vdash_\ell P}{\Gamma \vdash \ell[\![P]\!]} \qquad \frac{\Gamma \vdash S \quad , \quad \Gamma \vdash T}{\Gamma \vdash S \mid T} \qquad \frac{\ell : \{a : \gamma\}, \Gamma \vdash S}{\Gamma \vdash (\nu a@\ell)\, S} \qquad \frac{\ell : \psi, \Gamma \vdash S}{\Gamma \vdash (\nu \ell)\, S}$$

Fig. 4. Type system for networks

location type ψ by $dom(\psi)$ (e.g. $dom(\{a : \gamma, b : \delta, \psi\}) = \{a, b\}$). As for record types in $\{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_L\}$ the names $a_1, \ldots, a_n$ have to be distinct, an assumption we will take throughout the paper. In order to preserve this property by instantiation, the row variable of a type ψ is equipped with a subscript, that is a set of names which is meant to contain at least $dom(\psi)$. 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 duplicated assignments. However, in the section 4 we show that with dependant types this is not sufficient.

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 with a type *at least* ψ". These types are *existential* because $\gamma@\psi$ should be read as $\exists a.\{a : \gamma, \psi\}$. We assume that in $a(u).P$ we have $u \neq a$, and in $\{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_L\}$ we have $\rho_L \notin var(\gamma_1, \ldots, \gamma_n)$.

The type system is given in figures 2 to 4. We give a formal definition of well-formed types in the next section. 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, and similarly $\Gamma \vdash S$ for systems. We have two kinds of sequents for names: $\Gamma \vdash^W_\ell u : \tau$ to type names *with weakening* of hypotheses and sequents $\Gamma \vdash_\ell u : \tau$ *without weakening*. The latter are used to type formal parameters in input.

A typing context Γ is a mapping from a finite subset $dom(\Gamma)$ of $\mathcal{N}_{loc}$ into the set of ground location types. We use of a partial operation of union Δ, Γ of typing contexts, defined as follows:

$$(\Delta, \Gamma)(x) = \begin{cases} \Delta(\ell) & \text{if } \ell \in dom(\Delta) - dom(\Gamma) \\ \phi \sqcap \psi & \text{if } \Delta(\ell) = \phi \ \&\ \Gamma(\ell) = \psi \\ \Gamma(\ell) & \text{if } \ell \in dom(\Gamma) - dom(\Delta) \end{cases}$$

where $\psi \sqcap \phi$ denotes the union of ψ and ϕ, which is only defined if ψ and ϕ assign the same types to the names they share. As usual, we assume that bound names are renamed such that no collision with other bound or free names arises.

We comment the rules and note some elementary properties. In the rules for names without weakening the conclusion determines the context. Namely, if $\Gamma \vdash_\ell u : \tau$ and $\Delta \vdash_\ell u : \tau$ then $\Gamma = \Delta$. The rule for output involves a form of subtyping for localities: for instance, the judgement

$$\ell : \{b : \gamma, c : \delta\}\,,\ k : \{a : Ch(\{b : \gamma\})\} \vdash_k \overline{a}\ell$$

is valid, even though the type of ℓ given by the context is more generous than the one carried by the channel a. As usual, to type the body of an input the context is enriched with the information necessary for the typing the formal parameters. There are three cases for typing a name generation $(\nu u)\, P$ and the rules follow the same pattern as these for typing names. To type a migrating process $\mathsf{go}\,\ell.P$, one must type P at locality ℓ, while the resulting current locality is immaterial.

The main result concerning this type system is the subject reduction property: if $\Gamma \vdash S$ and $S \to T$, then $\Gamma \vdash T$. The proof can be found in [8].

4 Managing the Dependant Types

In this section, we define and motivate the notions of *principal typing* and *binding relations*. Our type inference algorithm consists, as usual, of two steps:

1. from a term and an initial typing context, generate type schemes constraints,
2. then we search the most general solution to the constraints such that its application to the initial typing context provides a principal typing. This is so-called *constraint unification*.

Let us begin with substitutions. A substitution (μ, λ, ...) is a finite mapping from type variables to types. For $\mu = [\tau_1/\alpha_1, \ldots, \tau_n/\alpha_n]$ we denote by $dom(\mu)$ the set $\{\alpha_1, \ldots, \alpha_n\}$ and by $vrang(\mu)$ the set $\bigcup_{i \in \{1 \ldots n\}} var(\tau_i)$. We consider idempotent substitutions, in particular we have $dom(\mu) \cap vrang(\mu) = \emptyset$. Outside its domain a substitution is intended to be the identity. They are trivially extended to homomorphisms on types and other objects (as contexts, subtyping assertions, etc.). We note $\lambda\mu$ the composition of λ and μ, and $\emptyset$ the empty substitution.

Principal Typing. Usually, by principal typing for a term U, we mean a context involving type schemes (denoted by $\mathbf{\Gamma}, \mathbf{\Delta}, \ldots$ and called *context schemes*) and whose all ground instantiations are valid typing contexts for U. However, this definition is too permissive because such a principal typing could involve invalid instantiations as the following example emphasises.

Example 4.1. Consider $S = \ell[\![\overline{a}\ell \mid \overline{b}c]\!]$ where c has some type γ. Then ℓ has type

$$\{a : Ch(\psi), b : Ch(\gamma), c : \gamma\}$$

where ψ can be either $\{b : Ch(\gamma), c : \gamma\}$, $\{b : Ch(\gamma)\}$, $\{c : \gamma\}$ or $\{\}$. One might expect the context scheme $\mathbf{\Gamma} = \ell\mathbf{:}\{a : Ch(\rho'_\emptyset), b : Ch(\gamma), c : \gamma, \rho_{\{a,b,c\}}\}$ to be a most general typing for ℓ. Obviously this is not case since, for instance, the application of the substitution $[\{b : Ch(Ch(\gamma))\}/\rho'_\emptyset]$ does not give a valid type for ℓ. This happens because b is used as a channel of type $Ch(Ch(\gamma))$ whereas a actually sends ℓ where b has the type $Ch(\gamma)$. □

We introduce a relation $\psi <\mathbf{:} \phi$ on location types (similar to the one of [6]) that can be understood as $\phi \subseteq \psi$, seeing location types as sets. More formally,

Definition 4.1. *We define the relation $<\mathbf{:}$ on location types as follows:*

$$\psi <\mathbf{:} \{\} \qquad \psi <\mathbf{:} \rho_L \qquad \psi <\mathbf{:} \phi \Rightarrow \{a : \tau, \psi\} <\mathbf{:} \{a : \tau, \phi\}$$

This relation defines subtyping assertions*. The subtyping assertions of the form $\psi <\mathbf{:} \rho_L$ are called* atomic*. Moreover, we write $\psi \equiv \phi$ if $\psi <\mathbf{:} \phi$ and $\phi <\mathbf{:} \psi$.* [1]

We can observe subtyping in receptions and emissions of location names. For instance, if P in $a(\ell).P$ uses channels $a_1, \ldots, a_n$ at location ℓ then a must have a type $Ch(\psi)$ where ψ assigns *at least* a type to each a_i. Therefore, in P, ℓ has a type ϕ such that $\psi <\mathbf{:} \phi$. Symmetrically, when emitting k on a, k must have a type ϕ' that assigns to each a_i the same type as ψ does; that is $\phi' <\mathbf{:} \psi$.

Example 4.2. Let us consider the term $S = \ell[\![\overline{a}k \mid a(\ell').\mathsf{go}\,\ell'.\overline{b}d]\!] \mid k[\![\overline{c}d]\!]$ where d has some type γ. After the input of k on a, S migrates to k a message on b. Then any location transmitted on a has to define at least a channel b with type $Ch(\gamma)$. Then, S can be associated with the following context scheme:

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

a may also have the type $Ch(\{b\mathbf{:}\, Ch(\gamma), c\mathbf{:}\, Ch(\gamma)\})$ that can be obtained by the ground substitution $[\{c\mathbf{:}\, Ch(\gamma)\}/\rho_{\{b\}}]$. Not all substitutions of $\rho_{\{b\}}$ give valid typings for a (as for instance $[\{c : Ch(Ch(\gamma))\}/\rho_{\{b\}}]$). However, whether or not a substitution gives valid types for S is easily decidable: any substitution μ that preserves the atomic subtyping assertion $\{c : Ch(\gamma), \rho''_{\{b,c\}}\} <\mathbf{:} \rho_{\{b\}}$ (that is such that $\mu\{c : Ch(\gamma), \rho''_{\{b,c\}}\} <\mathbf{:} \mu\rho_{\{b\}}$ is still valid), preserves the typing of S. This leads to the following definition of principle typing. □

Definition 4.2. *Let $\mathcal{A}$ be a set of atomic subtyping assertions, we say that $\mathbf{\Gamma}; \mathcal{A}$ is a* principal typing *for S if 1) for all ground substitution λ that preserves $\mathcal{A}$, we have $\lambda\mathbf{\Gamma} \vdash S$, 2) for all Δ such that $\Delta \vdash S$, there exists a ground substitution λ that preserves $\mathcal{A}$ such that $\Delta =_{dom(\mathbf{\Gamma})} \lambda\mathbf{\Gamma}$ and $\lambda\mathbf{\Gamma} \vdash S$.*

Binding relations. In order to keep the names of a location type distinct, as in [11], a row variable of a type ψ is equipped with a subscript. It is a set of names which is meant to contain at least $dom(\psi)$. In this case ψ is said *well-formed*. Substitutions have to satisfy the following requirements: (i) if $\mu\rho_L = \psi$, then

[1] We assume that this should not be confused with structural equivalence. Intuitively, $\psi \equiv \phi$ means that ψ and ϕ are identical modulo their row variables.

$dom(\psi) \cap L = \emptyset$, (ii) and if $\rho var(\psi) = \rho'_{L'}$ then $L \subseteq L'$. Condition (i) forbids the substitution of a row variable with a type that contains a label belonging to the subscript. However, in presence of dependant types, those subscripts are not anymore sufficient. Indeed, the names occuring in location types belongs to the same syntactic category of the names that may be bound in the terms. Moreover, the latter are not supposed to occur in the principal typing produced for it. However, the algorithm "deconstructs" the term, and names that were initially bound appear later free. For instance, associating the context scheme $\ell : \rho_\emptyset, k : \rho'_\emptyset$ to the network $S = \ell[\![a(b).\mathsf{go}\, k.\bar{b}c]\!]$, after two deconstructions the term $\mathsf{go}\, k.\bar{b}c$ appears in which b is free and supposed by known at k. So, we may substitute $\rho'_\emptyset$ with a location type that assigns a type to b. However, S is not typable since it tries to use the name b local to ℓ at a remote location k. To amend this, we could declare that if the generated substitution has bound names in its range, then it is not a valid one, and conclude that the term is not typable. However, this would be too strong. Indeed, consider the term

$$T = \ell[\![d(b).(\nu a)\, Q]\!] \quad \text{where} \quad Q = (\bar{a}\ell \mid a(k).\mathsf{go}\, k.\bar{b}c)$$

that waits at location ℓ for a channel b, then it creates a channel a carrying the location ℓ. Since b and c are used by a process spawned at a location received along a, a must have *at least* the type $Ch(\{b : Ch(\gamma), c : \gamma\})$. A bound name (b) appears in the type of a. However, this is not at variance with the fact that bound names do not appear in the typing context of a term because T is actually typable with the context $\ell : \{d : Ch(Ch(\gamma))\}$ in which the type of a does not appear. But in the type inference we have to compute all the types and produce a substitution that may contains bound names in its range. Therefore, we generalise the notion of subscripts of row variables to all type variables in a stronger form. Intuitively, we associate to each variable α, a set L representing the names not compatibles with α. That is, substitution are not allowed to map α into a type containing elements from L. Contrary to the row variables, those generalised subscripts have to be dynamically updated along the type inference. We formalised this notion by means of finite relations of $\mathcal{N}_{chan} \times \mathcal{V}$.

Definition 4.3. *A* binding relation $\mathcal{B}$ *is a finite subset of* $\mathcal{N}_{chan} \times \mathcal{V}$. *We note* $\mathcal{B}(\alpha)$ *the set* $\{a \mid (a, \alpha) \in \mathcal{B}\}$ *and* $im(\mathcal{B})$ *the set* $\{a \mid \exists \alpha.(a, \alpha) \in \mathcal{B}\}$.

Binding relations appears as subscripts of row variables: $\{a : \gamma, b : \delta, \rho_{\{b\}}\}$ can be considered well-formed with respect to the binding relation $\mathcal{B} = \{(a, \rho_{\{b\}})\}$. Indeed, a substitution that respects $\mathcal{B}$ never substitutes $\rho_{\{b\}}$ by a type containing a and therefore never duplicates its typing. However, we must keep the subscripts of the row variables because the binding relations are stronger constraints. For instance, we want to be able to extend $\{b : \gamma, \alpha\}$ with $\{a : Ch(\{b : \gamma\})\}$. This would be allowed by subscripts (that is if $\alpha = \rho_{\{b\}}$), but not by binding relations (that is if $\alpha = \rho_\emptyset$ and $\mathcal{B}(\rho_\emptyset) = \{b\}$) because subscripts are only concerned with the domains of location types, while binding relations are concerned with the types (in this sense binding relations are stronger). The consequence is that the well-formedness of types depends on binding relations. We say that ψ is well-formed with respect to $\mathcal{B}$ if the names of $dom(\psi)$ are all distinct, and, if ψ is

$$\frac{}{t :: \mathcal{B}} \qquad \frac{}{h :: \mathcal{B}} \qquad \frac{\tau :: \mathcal{B}}{Ch(\tau) :: \mathcal{B}} \qquad \frac{\gamma :: \mathcal{B} \quad , \quad \psi :: (L, \mathcal{B})}{\gamma @ \psi :: \mathcal{B}}$$

$$\frac{\psi :: (L, \mathcal{B})}{\psi :: \mathcal{B}} \qquad \frac{}{\{\} :: (L, \mathcal{B})} \qquad \frac{\gamma :: \mathcal{B} \quad , \quad \psi :: (L \uplus \{a\}, \mathcal{B})}{\{a : \gamma, \psi\} :: (L, \mathcal{B})} \qquad \frac{L' = L \cup \mathcal{B}(\rho_L)}{\rho_L :: (L', \mathcal{B})}$$

Fig. 5. Well-formedness of types with respect to a binding relation

extensible with row variable ρ_L, then $dom(\psi) \subseteq L \cup \mathcal{B}(\rho_L)$. More formally, a type τ is well-formed with respect to $\mathcal{B}$ if there exists a proof of the judgement $\tau :: \mathcal{B}$ in the inference system given in figure 5 where L is finite set of names.

Definition 4.4. *We say that a substitution μ* respects *a binding relation $\mathcal{B}$ if*

1. $\forall \alpha \in dom(\mu), nm(\mu\alpha) \cap \mathcal{B}(\alpha) = \emptyset$ *and* $\mu\alpha :: \mathcal{B}$,
2. $\forall \rho_L \in dom(\mu), dom(\mu\rho_L) \cap L = \emptyset$, *and if* $\rho \mathrm{var}(\mu\rho_L) = \rho'_{L'}$, *then* $\mathcal{B}(\rho_L) \subseteq \mathcal{B}(\rho'_{L'})$ *and* $L \subseteq L' \cup \mathcal{B}(\rho'_{L'})$.

The first point of this definition simply says that μ assigns types to variables according to what the binding relation allows, and that those types are still well-formed. The second point is a generalisation of the requirements (i) and (ii) given above to guarantee that substitution preserves the well-formedness.

Lemma 4.1. *1. if $\tau :: \mathcal{B}$ and μ respects $\mathcal{B}$, then $\mu\tau :: \mathcal{B}$,*
2. if $\mathcal{B} \subseteq \mathcal{B}'$ and $\tau :: \mathcal{B}$ then $\tau :: \mathcal{B}'$.

Unfortunately, the composition of two substitutions that respect $\mathcal{B}$ does not necessarily respect $\mathcal{B}$. For instance, $\mu = [\rho_\emptyset / t]$ and $\lambda = [\{a : \gamma, \rho'_{\{a\}}\} / \rho_\emptyset]$ respect $\mathcal{B} = \{(a, t)\}$. However, the composition $\lambda\mu = [\{a : \gamma, \rho'_{\{a\}}\} / \rho_\emptyset, \{a : \gamma, \rho'_{\{a\}}\} / t]$ does not respect $\mathcal{B}$, because $a \in nm(\lambda\mu t)$ whereas $a \in \mathcal{B}(t)$.

Definition 4.5. *$\mathcal{B}$ is* μ-closed *if* $\forall \alpha \in dom(\mu), \forall \beta \in dom(\mu\alpha). \mathcal{B}(\alpha) \subseteq \mathcal{B}(\beta)$.

Lemma 4.2. *Let μ be a substitution that respects $\mathcal{B}$ and $\mathcal{B}$ be μ-closed, then for all substitution λ that respects $\mathcal{B}$, $\lambda\mu$ respects $\mathcal{B}$.*

It is easy, from a binding relation to construct an another one that is μ-closed.

Definition 4.6. *We define the* μ-closure of $\mathcal{B}$ *as the following binding relation:*

$$\mathcal{B} \cup \bigcup_{\alpha \in dom(\mu)} \bigcup_{\beta \in \mathrm{var}(\mu\alpha)} \mathcal{B}(\alpha) \times \{\beta\}$$

The reader can easily check that if μ respects $\mathcal{B}$, then μ still respects its μ-closure.

5 Solving Type Constraints

In this section we give an algorithm solving type constraints in terms of a rewriting system. The usual unification problem is: given equations between types, does there exist a substitution for type variables that equates types ? As for unification of record types, all type equality is supposed to be modulo the equation E:

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

That is, 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.

Definition 5.1. *A* typing constraint *is a set $\mathcal{E}$ of equations between type schemes $\{\tau_1 \doteq \sigma_1, \ldots, \tau_n \doteq \sigma_n\}$. A* subtyping constraint *is a set $\mathcal{I}$ of inequations between location types $\{\psi_1 \lessdot \phi_1, \ldots, \psi_n \lessdot \phi_n\}$. We say that μ is a* solution *of $\mathcal{E}$ (resp. $\mathcal{I}$) if $\mu\tau_i = \mu\sigma_i$ (resp. $\mu\psi_i <: \mu\phi_i$) for all $1 \leq i \leq n$. We note $\mathcal{E} :: \mathcal{B}$ if $\tau_i :: \mathcal{B}$ and $\sigma_i :: \mathcal{B}$ for all $1 \leq i \leq n$ and similarly for $\mathcal{I} :: \mathcal{B}$. We write $\psi \doteqdot \phi$ for $\psi \lessdot \phi \wedge \phi \lessdot \psi$.*

We write $\mu =_{\mathcal{X}} \lambda$ where $\mathcal{X}$ is a set of type variables if $\mu\alpha = \lambda\alpha$ for all $\alpha \in \mathcal{X}$.

Definition 5.2. *μ is* more general on $\mathcal{X}$ than *λ if and only if there exists a substitution μ' such that $\lambda =_{\mathcal{X}} \mu'\mu$. In this case we write $\mu \leq_{\mathcal{X}} \lambda$.*

Definition 5.3. *A* constraint *is a tuple $(\mathcal{E}, \mathcal{I})_{\mathcal{B}}$ such that $\mathcal{E} :: \mathcal{B}$ and $\mathcal{I} :: \mathcal{B}$. A substitution μ is a* principal solution of $(\mathcal{E}, \mathcal{I})_{\mathcal{B}}$ on $\mathcal{X}$*, if for all ground solution λ of $\mathcal{E}$ and $\mathcal{I}$ that respects $\mathcal{B}$, we have $\mu \leq_{\mathcal{X}} \lambda$.*

In figure 6 we define a reduction relation $\rightsquigarrow$ on tuples $(\mathcal{E}, \mathcal{I}, \mu)^{\mathcal{X}}_{\mathcal{B}}$. We make use of the abbreviation $\psi \triangleleft \rho_L$ for $[\rho_L/\rho var(\psi)]\psi$. The idea is that starting from $(\mathcal{E}, \mathcal{I}, \emptyset)^{\mathcal{X}}_{\mathcal{B}}$ – that is a constraint $(\mathcal{E}, \mathcal{I})_{\mathcal{B}}$, a set of type variables $\mathcal{X}$ containing those occuring in $\mathcal{E}$ and $\mathcal{I}$ and the empty substitution – we apply the reduction relation until we reach either a configuration $(\emptyset\, \mathcal{A}, \mu)^{\mathcal{Y}}_{\mathcal{B}}$ where μ is a principal solution of $(\mathcal{E}, \mathcal{I})_{\mathcal{B}}$ on $\mathcal{X}$, or the failure configuration $\perp$ if $\mathcal{E}$ and $\mathcal{I}$ have no common solution. This relation almost consists of the decompositions of pairs of types until one of these is a type variable. Then (rule (*elim*)), the current substitution μ is composed with the substitution of the type variable for the other type (the latter being also applied to the remaining constraint) provided that it respects the current binding relation. The condition $\alpha \notin var(\tau)$ ensures that there is no remaining occurrence of the eliminated variable in the resulting constraint, thus avoiding infinite reductions. If this condition fails, the occurs check rule (*oc*) leads to the failure configuration. The rule (*triv*) removes trivial equations. The rules (*chan*) and (*at*) simply decompose types. Rule (loc_2) unifies location types with disjoint domains: it extends each location by means of an appropriate substitution of their row variables.

Rules (st_1) and (st_2) are used to solve subtyping inequations. The first one asserts that $\{a : \gamma, \psi\} \lessdot \{a : \delta, \phi\}$ has a solution if we can unify γ with δ and if $\psi \lessdot \phi$ has a solution. Rule (st_2) applies on $\psi \lessdot \phi$ when the channels defined in ϕ are not in ψ. We then extend ψ with the channels typed in ϕ with respect to the current binding relation.

Definition 5.4. *We say that $(\mathcal{E}, \mathcal{I}, \mu)^{\mathcal{X}}_{\mathcal{B}}$ is a* well-formed configuration *if $dom(\mu) \cap var(\mathcal{E}, \mathcal{I}) = \emptyset$, $var(\mathcal{E}, \mathcal{I}, \mu) \subseteq \mathcal{X}$, μ respects $\mathcal{B}$ that is μ-closed, $\mathcal{E} :: \mathcal{B}$ and $\mathcal{I} :: \mathcal{B}$.*

This definition gives some invariants for the reduction relation of unification.

Lemma 5.1. *The property of well-formed configuration is preserved by $\rightsquigarrow$.*

The following lemma states the *preservation of solutions* by the reductions.

$$(triv) \quad (\{\alpha \doteq \alpha\} \cup \mathcal{E}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow (\mathcal{E}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{B}}$$

$$(elim) \quad (\{\alpha \doteq \tau\} \cup \mathcal{E}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow ([\tau/\alpha]\mathcal{E}\,,\,[\tau/\alpha]\mathcal{I}\,,\,[\tau/\alpha]\mu)^{\mathcal{X}}_{\mathcal{B}'}$$

$$(chan) \quad (\{Ch(\tau) \doteq Ch(\sigma)\} \cup \mathcal{E}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow (\{\tau \doteq \sigma\} \cup \mathcal{E}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{B}}$$

$$(at) \quad (\{\gamma@\psi \doteq \delta@\phi\} \cup \mathcal{E}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow (\{\gamma \doteq \delta, \psi \doteq \phi\} \cup \mathcal{E}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{B}}$$

$$(loc_1) \quad (\left\{\{a : \gamma, \psi\} \doteq \{a : \delta, \phi\}\right\} \cup \mathcal{E}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow (\{\gamma \doteq \delta\} \cup \{\psi \doteq \phi\} \cup \mathcal{E}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{B}}$$

$$(loc_2) \quad (\{\psi \doteq \phi\} \cup \mathcal{E}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow (\lambda\mathcal{E},\,\lambda\mathcal{I}\,,\,\lambda\mu)^{\mathcal{Y}}_{\mathcal{B}'}$$

$$(clash) \quad (\{\sigma \doteq \tau\} \cup \mathcal{E}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow \bot$$

$$(oc) \quad (\{\alpha \doteq \tau\} \cup \mathcal{E}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow \bot$$

$$(st_1) \quad (\mathcal{E}\,,\,\left\{\{a : \gamma, \psi\} \ll \{a : \delta, \phi\}\right\} \cup \mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow (\{\gamma \doteq \delta\} \cup \mathcal{E}\,,\,\{\psi \ll \phi\} \cup \mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{B}}$$

$$(st_2) \quad (\mathcal{E}\,,\,\{\psi \ll \phi\} \cup \mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow (\lambda\mathcal{E}\,,\,\{\psi \lhd \rho''_{L''} \ll \rho'_{L'}\} \cup \lambda\mathcal{I}\,,\,\lambda\mu)^{\mathcal{Y}}_{\mathcal{B}'}$$

where in

($elim$) $\alpha \notin var(\tau) \cup \mathcal{V}_{loc},, \alpha \notin dom(\mu), nm(\tau) \cap \mathcal{B}(\alpha) = \emptyset, \mathcal{B}'$ is the $[\tau/\alpha]$-closure of $\mathcal{B}$.

(loc_2) $\mathcal{Y} = \mathcal{X} \uplus \{\rho''_{L''}\}, \psi' = \psi \lhd \rho''_{L''},\ \phi' = \phi \lhd \rho''_{L''}$ and $\lambda = [\psi'/\rho'_{L'}, \phi'/\rho_L]$ with $\rho_L = \rho var(\psi), \rho'_{L'} = \rho var(\phi)$, and $L'' = (L - \mathcal{B}(\rho'_{L'})) \cup (L' - \mathcal{B}(\rho_L))$ and if

- $dom(\psi) \cap dom(\phi) = \emptyset, L' \cap dom(\psi) = \emptyset$ and $L \cap dom(\phi) = \emptyset$
- $nm(\psi) \cap \mathcal{B}(\rho'_{L'}) = \emptyset$ and $nm(\phi) \cap \mathcal{B}(\rho_L) = \emptyset$
- $\rho_L \notin var(\phi) - \rho'_{L'}$ and $\rho'_{L'} \notin var(\psi) - \rho_L$
- $\mathcal{B}'$ is the λ-closure of $\mathcal{B}$.

($clash$) σ and τ are not type variables and have distinct top symbols, or are locality types with disjoint domains and conditions of (loc_2) fail.

(oc) $\alpha \in var(\tau)$.

(st_2) $\mathcal{Y} = \mathcal{X} \uplus \{\rho''_{L''}\}, \lambda = [\phi \lhd \rho''_{L''}/\rho_L]$ with $\rho_L = \rho var(\psi), \rho'_{L'} = \rho var(\phi)$, and if $\phi \notin \mathcal{V}_{loc}, dom(\psi) \cap dom(\phi) = \emptyset, L \cap dom(\phi) = \emptyset, \rho_L \notin var(\phi) - \rho'_{L'}$, $nm(\phi) \cap \mathcal{B}(\rho_L) = \emptyset, L'' = (L - \mathcal{B}(\rho'_{L'})) \cup (L' - \mathcal{B}(\rho_L))$, $\mathcal{B}' = \mathcal{B}'' \cup \mathcal{B}(\rho'_{L'}) \times \{\rho''_{L''}\}$ where $\mathcal{B}''$ is the λ-closure of $\mathcal{B}$.

Fig. 6. Reduction relation for unification

Lemma 5.2. *If $(\mathcal{E}, \mathcal{I}, \mu)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow^* (\mathcal{E}', \mathcal{I}', \mu')^{\mathcal{Y}}_{\mathcal{B}'}$ and $(\mathcal{E}, \mathcal{I}, \mu)^{\mathcal{X}}_{\mathcal{B}}$ is a well-formed configuration, then $\mu' = \mu\mu''$ and,*

1. *for all λ ground solution of $\mathcal{E}$ and $\mathcal{I}$ that respects $\mathcal{B}$, then there exists λ' such that $\lambda =_{\mathcal{X}} \lambda'\mu''$ and λ' is a ground solution of $\mathcal{E}'$ and $\mathcal{I}'$ that respects $\mathcal{B}'$.*
2. *For all solution λ of $\mathcal{E}'$ and $\mathcal{I}'$ that respects $\mathcal{B}'$, $\lambda\mu''$ is a solution of $\mathcal{E}$ and $\mathcal{I}$ that respects $\mathcal{B}'$.*

Lemma 5.3 (Termination). *All sequence of reductions* $(\mathcal{E},\mathcal{I},\emptyset)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow (\mathcal{E}',\mathcal{I}',\mu')^{\mathcal{Y}}_{\mathcal{B}'} \rightsquigarrow \ldots$ *terminates either with* $\bot$ *or with* $(\emptyset,\mathcal{A},\mu)^{\mathcal{Y}}_{\mathcal{B}''}$ *where* $\mathcal{A}$ *is a set of atomic subtyping assertions.*

We can finally state the soundness and completeness of our unification algorithm.

Proposition 5.1 (Soundness). *If* $(\mathcal{E},\mathcal{I},\emptyset)^{\mathcal{X}}_{\mathcal{B}}$ *is a well-formed and* $(\mathcal{E},\mathcal{I},\emptyset)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow^* (\emptyset,\mathcal{A},\mu)^{\mathcal{Y}}_{\mathcal{B}'} \not\rightsquigarrow$, *then* μ *is a principal solution of* $(\mathcal{E},\mathcal{I})_{\mathcal{B}}$ *on* $\mathcal{X}$ *and* μ *respects* $\mathcal{B}'$.

Lemma 5.4. $(\mathcal{E},\mathcal{I},\emptyset)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow^* \bot$ *iff* $\mathcal{E}$ *and* $\mathcal{I}$ *have no solution that respects* $\mathcal{B}$.

Proposition 5.2 (Completeness). *If* $\mathcal{E}$ *and* $\mathcal{I}$ *have a solution that respects* $\mathcal{B}$, *if* $\mathcal{E} :: \mathcal{B}$ *and* $\mathcal{I} :: \mathcal{B}$, *then* $(\mathcal{E},\mathcal{I},\emptyset)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow^* (\emptyset,\mathcal{A},\mu)^{\mathcal{Y}}_{\mathcal{B}'} \not\rightsquigarrow$ *where* $\mathcal{X} = \mathrm{var}(\mathcal{E},\mathcal{I})$.

6 Constraint Generation

In this section we describe the inference of types, that is starting from a network term and a minimal typing context, we generate a constraint whose principal solution applied to the initial context gives a principal typing. By initial context, for a term S, we mean the set of location names occurring free in S, associated with a row variable as type. The idea of the algorithm is to build incrementally the inference tree of the typing of a term, *i.e.* 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 $(\mathcal{J},\mathcal{E},\mathcal{I})^{\mathcal{X}}_{\mathcal{B}}$ where $\mathcal{J}$ is a set of sequents involving context schemes and $(\mathcal{E},\mathcal{I})_{\mathcal{B}}$ is the constraint being generated. The reduction is very close to the inference system. Indeed, given a sequent in the tuple, the reduction mostly consists in replacing it by the sequents that are premises of the corresponding rule in the inference system. Possibly, constraints are also generated accordingly.

The rules are collected in figures 7 where an underscore (_) denotes an irrelevant component. We just comment the rules for the binding constructs, the others being relatively straightforward. For an input process the type system uses the auxiliary type system without weakening for names. Since it is very simple and completely deterministic, given a name u, a type τ and a location ℓ we can easily determine Γ such that $\Gamma \vdash_\ell u : \tau$. Actually, τ only needs to be a type variable and u a location or a compound name. We use $gen(u : \alpha, \ell)$ to generate adequate context schemes and typing constraints for the typing of u.

Definition 6.1. *We define the function* $gen(u : t, \ell) = (\mathbf{\Gamma}, \mathcal{E}, \mathcal{X})$ *as follows:*

$$gen(k : t, \ell) = (k : \rho_\emptyset, \{t \doteq \rho_\emptyset\}, \{\rho_\emptyset\})$$
$$gen(a@k : t, \ell) = (k : \{a : h, \rho_\emptyset\}, \{t \doteq h@\rho_\emptyset\}, \{h, \rho_\emptyset\})$$

where $\ell \neq k$, $\rho_\emptyset$ *and* h *are fresh type variables.*

In the rule (p_{2b}), the body of an input (of a location or a compound name) is typed in the initial context extended with the context provided by *gen*. This extension is allowed 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.

$$
\begin{array}{lll}
(n_1) & (\{\Gamma \vdash^W_\ell k : t\} \cup \mathcal{J}, \mathcal{E}, \mathcal{I})^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\mathcal{J}, \mathcal{E} \cup \{t \doteq \rho_\emptyset\}, \mathcal{I} \cup \{\Gamma(k) \lessdot \rho_\emptyset\})^{\mathcal{X} \uplus \{\rho_\emptyset\}}_{\mathcal{B}} \\
(n_2) & (\{\Gamma \vdash^W_\ell a : t\} \cup \mathcal{J}, \mathcal{E}, \mathcal{I})^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\mathcal{J}, \mathcal{E} \cup \{t \doteq h\}, \mathcal{I} \cup \left\{\Gamma(\ell) \lessdot \{a : h, \rho_{\{a\}}\}\right\})^{\mathcal{X} \uplus \{\rho_{\{a\}}, h\}}_{\mathcal{B}} \\
(n_3) & (\{\Gamma \vdash^W_\ell a@k : t\} \cup \mathcal{J}, \mathcal{E}, \mathcal{I})^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\mathcal{J}, \mathcal{E} \cup \{t \doteq h@\rho'_\emptyset\}, \\
 & & \quad \mathcal{I} \cup \left\{\rho_{\{a\}} \doteqdot \rho'_\emptyset, \Gamma(k) \lessdot \{a : h, \rho_{\{a\}}\}\right\})^{\mathcal{X} \uplus \{h, \rho_{\{a\}}, \rho'_\emptyset\}}_{\mathcal{B}} \\
(p_0) & (\{\Gamma \vdash_\ell \mathbf{0}\} \cup \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{B}} \\
(p_1) & (\{\Gamma \vdash_\ell \overline{a}u\} \cup \mathcal{J}, \mathcal{E}, \_)^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\mathcal{J} \cup \{\Gamma \vdash^W_\ell u : t\}, \\
 & & \quad \mathcal{E} \cup \left\{\Gamma(\ell) \doteq \{a : Ch(t), \rho_{\{a\}}\}\right\}, \_)^{\mathcal{X} \uplus \{t, \rho_{\{a\}}\}}_{\mathcal{B}} \\
(p_{2a}) & (\{\Gamma \vdash_\ell a(b).P\} \cup \mathcal{J}, \mathcal{E}, \mathcal{I})^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\{\Delta \vdash_\ell P\} \cup \mathcal{J}, \left\{\Gamma(\ell) \doteq \{a : Ch(h), \rho''_{\{a\}}\}\right\} \cup \mathcal{E}, \\
 & & \quad \{\rho_\emptyset \doteqdot \rho'_\emptyset\} \cup \mathcal{I})^{\mathcal{Y}}_{\mathcal{B}'} \\
(p_{2b}) & (\{\Gamma \vdash_\ell a(u).P\} \cup \mathcal{J}, \mathcal{E}, \mathcal{I})^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\{\Gamma, \Delta \vdash_\ell P\} \cup \mathcal{J}, \\
 & & \quad \left\{\Gamma(\ell) \doteq \{a : Ch(t), \rho_{\{a\}}\}\right\} \cup \mathcal{E} \cup \mathcal{E}', \mathcal{I})^{\mathcal{Y}}_{\mathcal{B}'} \\
(p_3) & (\{\Gamma \vdash_\ell P \mid Q\} \cup \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\{\Gamma \vdash_\ell P, \Gamma \vdash_\ell Q\} \cup \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{B}} \\
(p_4) & (\{\Gamma \vdash_\ell (\nu a)\, P\} \cup \mathcal{J}, \_, \mathcal{I})^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\{\Delta \vdash_\ell P\} \cup \mathcal{J}, \_, \{\rho_\emptyset \doteqdot \rho'_\emptyset\} \cup \mathcal{I})^{\mathcal{X} \uplus \{\rho'_\emptyset, h\}}_{\mathcal{B}'} \\
(p_5) & (\{\Gamma \vdash_k (\nu a@\ell)\, P\} \cup \mathcal{J}, \_, \mathcal{I})^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\{\Delta \vdash_k P\} \cup \mathcal{J}, \_, \{\rho_\emptyset \doteqdot \rho'_\emptyset\} \cup \mathcal{I})^{\mathcal{X} \uplus \{\rho'_\emptyset, h\}}_{\mathcal{B}'} \\
(p_6) & (\{\Gamma \vdash_\ell (\nu k)\, P\} \cup \mathcal{J}, \_, \_)^{\mathcal{Y}}_{\mathcal{B}} & \rightsquigarrow (\{k : \rho_\emptyset, \Gamma \vdash_\ell P\} \cup \mathcal{J}, \_, \_)^{\mathcal{X} \uplus \{\rho_\emptyset\}}_{\mathcal{B}} \\
(p_7) & (\{\Gamma \vdash_\ell \mathsf{go}\, k.P\} \cup \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\{\Gamma \vdash_k P\} \cup \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{B}} \\
(p_8) & (\{\Gamma \vdash_\ell !P\} \cup \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\{\Gamma \vdash_\ell P\} \cup \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{B}} \\
(s_0) & (\{\Gamma \vdash \mathbf{0}\} \cup \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{B}} \\
(s_1) & (\{\Gamma \vdash \ell[\![P]\!]\} \cup \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\{\Gamma \vdash_\ell P\} \cup \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{B}} \\
(s_2) & (\{\Gamma \vdash S \mid S'\} \cup \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\{\Gamma \vdash S, \Gamma \vdash S'\} \cup \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{B}} \\
(s_3) & (\{\Gamma \vdash (\nu a@\ell)\, S\} \cup \mathcal{J}, \_, \mathcal{I})^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\{\Delta \vdash S\} \cup \mathcal{J}, \_, \{\rho_\emptyset \doteqdot \rho'_\emptyset\} \cup \mathcal{I})^{\mathcal{X} \cup \{t, \rho'_\emptyset, h\}}_{\mathcal{B}'} \\
(s_4) & (\{\Gamma \vdash (\nu \ell)\, S\} \cup \mathcal{J}, \mathcal{E}, \_)^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow (\{\ell : \rho_\emptyset, \Gamma \vdash S\} \cup \mathcal{J}, \mathcal{E}, \_)^{\mathcal{X} \uplus \{\rho_\emptyset\}}_{\mathcal{B}} \\
(f) & (\mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{B}} & \rightsquigarrow \bot
\end{array}
$$

where in

(p_{2a}) $\Delta =_{dom(\Gamma) - \ell} \Gamma, \Delta(\ell) = [\{b : h, \rho'_\emptyset\}/\rho_\emptyset]\Gamma(\ell)$ with $\rho_\emptyset = \rho var(\Gamma(\ell))$, $\mathcal{Y} = \mathcal{X} \uplus \{h, \rho'_\emptyset, \rho''_{\{a\}}\}$, and $\mathcal{B}' = \mathcal{B} \cup (\mathcal{B}(\rho_\emptyset) \times \{\rho'_\emptyset\}) \cup (\{b\} \times \mathcal{Y})$

(p_{2b}) $(\Delta, \mathcal{E}', \mathcal{X}') = gen(u : t, \ell), \mathcal{Y} = \mathcal{X} \uplus \mathcal{X}' \uplus \{t, \rho_{\{a\}}\}, \mathcal{B}' = \mathcal{B} \cup (chan(u) \times \mathcal{Y})$,

$(p_{4,5})$ (s_3) $\Delta =_{dom(\Gamma) - \ell} \Gamma$ and $\Delta(\ell) = [\{a : h, \rho'_\emptyset\}/\rho_\emptyset]\Gamma(\ell)$ with $\rho_\emptyset = \rho var(\Gamma(\ell))$, and $\mathcal{B}' = \mathcal{B} \cup (\mathcal{B}(\rho_\emptyset) \times \{\rho'_\emptyset\}) \cup (\{a\} \times \mathcal{X} \uplus \{\rho'_\emptyset, h\})$

(f) $\exists \Gamma \in \mathcal{J}$ that is not legal or if conditions of the previous rules fail.

Fig. 7. Constraints generation for processes and networks

Moreover, the binding relation is updated in order to forbid the substitution of any current type variables with a type in which the bound channel occurs. When the name received is simple (say b, in rule (p_{2a})), the type ψ assigned to the current location ℓ in Γ have to be extended with a type assignment for b. This is performed by the substitution of the row variable $\rho_\emptyset$ of ψ with a location type assigning a type (variable) to b. We have to maintain the coherence between the fresh row variable of the type assigned to ℓ in the new context (Δ) and the one in

$\mathbf{\Gamma}$ (that may still occurs in the remains tuple). This is achieved by equating those two row variables. As in (p_{2b}), we update the binding relation but so as to keep $\mathbf{\Delta}$ well-formed with respect to the new binding relation. Rules for restrictions $(p_4, p_5$ and $s_3)$ are very similar.

We give some invariants and states the termination of the reduction. We denote by $bn(\mathcal{J})$ the set of bound names of terms in $\mathcal{J}$.

Definition 6.2. *We say that* $(\mathcal{J},\mathcal{E},\mathcal{I})^{\mathcal{X}}_{\mathcal{B}}$ *is a* well-formed configuration *if:*

1. $var(\mathcal{J},\mathcal{E},\mathcal{I}) \subseteq \mathcal{X}$,
2. $\mathcal{E} :: \mathcal{B}$, $\mathcal{I} :: \mathcal{B}$ *and all types in* $\mathcal{J}$ *are well-formed with respect to* $\mathcal{B}$.
3. $bn(\mathcal{J}) \cap im(\mathcal{B}) = \emptyset$,
4. *for all* $\mathbf{\Gamma} \in \mathcal{J}$ *we have* $dom(\mathbf{\Gamma}) \cap bn(\mathcal{J}) = \emptyset$ *and for all* $\ell \in dom(\mathbf{\Gamma})$ *we have* $\mathbf{\Gamma}(\ell)$ *has the form* $\{a_1 : h_1, \ldots, a_n : h_n, \rho_\emptyset\}$ *with* $\mathcal{B}(\rho_\emptyset) = im(\mathcal{B})$.

Lemma 6.1. *The property of well-formed configuration is preserved by* $\rightsquigarrow$.

Lemma 6.2. *All sequence of reductions* $(\mathcal{J},\mathcal{E},\mathcal{I})^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow (\mathcal{J}',\mathcal{E}',\mathcal{I}')^{\mathcal{X}'}_{\mathcal{B}'} \rightsquigarrow \ldots$ *terminates either with* $\bot$ *or with* $(\emptyset,\mathcal{E}'',\mathcal{I}'')^{\mathcal{Y}}_{\mathcal{B}''}$.

We say that a substitution λ is solution of $(\mathcal{J},\mathcal{E},\mathcal{I})$ if it is a solution of $\mathcal{E}$ and $\mathcal{I}$, and if it validates the sequents in $\mathcal{J}$.

Proposition 6.1. *Let* $(\mathcal{J},\mathcal{E},\mathcal{I})^{\mathcal{X}}_{\mathcal{B}}$ *be well-formed, and* $(\mathcal{J},\mathcal{E},\mathcal{I})^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow^* (\mathcal{J}',\mathcal{E}',\mathcal{I}')^{\mathcal{Y}}_{\mathcal{B}'}$

1. *if* λ *is a ground solution of* $(\mathcal{J},\mathcal{E},\mathcal{I})$ *that respects* $\mathcal{B}$ *with* $\mathrm{nm}(\mathrm{im}(\lambda)) \cap bn(\mathcal{J}) = \emptyset$, *then there exists* $\mu =_{\mathcal{X}} \lambda$ *with* $\mathrm{nm}(\mathrm{im}(\mu)) \cap bn(\mathcal{J}') = \emptyset$, *and* μ *is a solution of* $(\mathcal{J}',\mathcal{E}',\mathcal{I}')$ *that respects* $\mathcal{B}'$.
2. *if* λ *is a ground solution of* $(\mathcal{J}',\mathcal{E}',\mathcal{I}')$ *that respects* $\mathcal{B}'$, *then* λ *is also a solution of* $(\mathcal{J},\mathcal{E},\mathcal{I})$.

The unification combined with this reduction provide a sound and complete type inference algorithm.

Theorem 6.1 (Soundness). *Let* $\mathbf{\Gamma}$ *be an initial context for* S, *and* $\mathcal{X} = var(\mathbf{\Gamma})$, *if* $(\{\mathbf{\Gamma} \vdash S\},\emptyset,\emptyset)^{\mathcal{X}}_{\emptyset} \rightsquigarrow^* (\emptyset,\mathcal{E},\mathcal{I})^{\mathcal{Y}}_{\mathcal{B}}$ *and* $(\mathcal{E},\mathcal{I},\emptyset)^{\mathcal{Y}}_{\mathcal{B}} \rightsquigarrow^* (\emptyset,\mathcal{A},\mu)^{\mathcal{Z}}_{\mathcal{B}'} \not\rightsquigarrow$ *then* $\mu\mathbf{\Gamma};\mathcal{A}$ *is a principal typing for* S.

Lemma 6.3. *Let* $\mathbf{\Gamma}$ *be an initial context for* S, *if* $(\{\mathbf{\Gamma} \vdash S\},\emptyset,\emptyset)^{var(\mathbf{\Gamma})}_{\emptyset} \rightsquigarrow^* \bot$ *then* S *is not typable.*

Theorem 6.2 (Completeness). *Let* $\mathbf{\Gamma}$ *be a initial context for* S, *if* S *is typable then* $(\{\mathbf{\Gamma} \vdash S\},\emptyset,\emptyset)^{var(\mathbf{\Gamma})}_{\emptyset} \rightsquigarrow^* (\emptyset,\mathcal{E},\mathcal{I})^{\mathcal{X}}_{\mathcal{B}}$ *and* $(\mathcal{E},\mathcal{I},\emptyset)^{\mathcal{X}}_{\mathcal{B}} \rightsquigarrow^* (\emptyset,\mathcal{A},\mu)^{\mathcal{Y}}_{\mathcal{B}'} \not\rightsquigarrow$.

This completeness theorem combined with the soundness one allows us to say that, whenever a term is typable our algorithms of constraint generation and unification compute a principal typing for it.

7 Conclusion

In this paper we studied the problem of type inference for a distributed π-calculus with code migration and local communication. Using an explicit subtyping relation on location types we define a notion of *principal typing* leading to a *practical* type inference problem *à la* ML.

Technically, we proposed a unification algorithm that computes the principal solution of a constraint. We gave a sound and complete algorithm that, given a system S, generates a constraint whose solution yields a principal typing for S. Since we considered *dependant types*, we showed how to manage substitutions carefully with respect to bound names. To this aim we introduced the novel notion of *binding relation*. For the sake of simplicity, in this paper we considered a monadic calculus, however we could easily extend our results to the full polyadic version. In [8], we also deal with (mis)matching of values and recursion. We have not yet addressed the simplification of the atomic subtyping assertions generated by the algorithm ([9]).

We believe that this work could be easily adapted to the type system of [6]. Moreover, the presentation of algorithms by means of reduction relations, and the fact that we compute a principal type, should be useful for a formal definition of "dynamic" typing and its integration in process reduction. For instance, in [12], Hennessy and Reily, study a partial typing for open systems where only some sites may be typed. However, they informally assume the existence of a type checker, and their terms are explicitly typed. We think that their work could be extended to allow dynamic computation of type information.

References

[1] R. Amadio. On modeling mobility. *Journal of Theoretical Computer Science*, 240(1):147–176, 2000.
[2] R. Amadio, G. Boudol, and C. Lhoussaine. The receptive distributive π-calculus. Technical Report 4080, INRIA, Sophia Antipolis, 2000.
[3] G. Berry and G. Boudol. The Chemical Abstract Machine. In *POPL'90*, pages 81–94, San Francisco, California, January 17–19, 1990. ACM Press, New York.
[4] L. Cardelli and A.D. Gordon. Mobile ambients. In M. Nivat, editor, *FoSSaCS*, volume 1378, pages 140–155. Springer-Verlag, Berlin, Germany, 1998.
[5] C. Fournet, G. Gonthier, JJ. Lévy, L. Maranget, and D. Rémy. A calculus of mobile agents. In *CONCUR'96*, pages 406–421, Pisa, Italy, 1996. Springer-Verlag.
[6] M. Hennessy and J. Riely. Resource access control in systems of mobile agents. In *Information and Computation* 174(2): 143–179 (2002).
[7] L. Jategaonkar and J.C. Mitchell. Type inference with extended pattern matching and subtypes. *Fundamenta Informaticae*, 19(1/2):127–165, 1993.
[8] C. Lhoussaine. *Réceptivité, mobilité et π-calcul*. PhD thesis, Université de Provence, 2002.
[9] F. Pottier. Simplifying subtyping constraints: A theory. *INFCTRL: Information and Computation*, 170, 2001.
[10] Didier Rémy. Syntactic theories and the algebra of record terms. Research Report 1869, INRIA, Rocquencourt, 1993.

[11] Didier Rémy. Type inference for records in a natural extension of ML. In *Theoretical Aspects Of Object-Oriented Programming. Types, Semantics and Language Design*. MIT Press, 1993.
[12] James Riely and Matthew Hennessy. Trust and Partial Typing in Open Systems of Mobile Agents. In *POPL'99*, pages 93–104, 1999.
[13] M. Wand. Complete type inference for simple objects. In *Symposium on Logic in Computer Science, Ithaca, NY*, pages 37–44, 1987.