

Characterization of principal type of normal forms in intersection type system

Emilie Sayag, Michel Mauny*
INRIA-Rocquencourt

Abstract

We introduce a restriction of the intersection type discipline that leads to a principal type property in the classical sense for normal forms. We characterize completely the structure of principal types of normal forms and we give an algorithm that reconstructs normal forms from types.

1 Introduction

In the untyped λ -calculus approach as a model of programming languages, Curry's type system is the basis of type systems of programming languages like ML [12].

Indeed, Curry's type system has the principal type property, which allows using parametric polymorphism. Furthermore, the problem of typeability in this system is decidable.

However, this type system has some limitations. For example, the λ -term $(\lambda x.x x)$ is not typeable in this system and the λ -terms $(\lambda x.\lambda y.y)$ and $(\lambda x.\lambda y.\lambda z.x z (y z))(\lambda s.\lambda t.s)$ are β -equivalent, but they have two different principal types.

To supply a type system that does not have these drawbacks, several extensions of Curry's type system are proposed, the most studied of which is the intersection type discipline [17, 4]. This extension allows terms and term-variables to have more than one type. The types are formed with the constructors \rightarrow and \cap for system \mathcal{D} [3], a universal type ω for the system \mathcal{D}_ω [6], and a partial order \leq on types for the system $\mathcal{D}_{\omega \leq}$ [1].

Studies of intersection type systems led to the following theoretical results : a term is strongly normalizable if and only if it is typeable in system \mathcal{D} [14, 9], a term is normalizable if and only if it is typeable in system \mathcal{D}_ω with a type in which ω doesn't appear [6, 9, 8] and the system $\mathcal{D}_{\omega \leq}$ gives rise to a filter lambda model and is semantically complete [1]. Moreover, types in \mathcal{D}_ω are invariant under β -conversion of terms [2]. Intersection type systems are therefore very expressive and this expressivity justifies the interest and use of this system by many authors [13, 7].

However the price of this expressivity is the in-decidability of typeability. Another drawback of this system is the loss of principal type property in the classical sense. As a matter of fact, in order to find all possible types of a term from a unique type, we must have more than only substitutions. Nevertheless in [5, 16, 21] a property which is similar to the principal type property is proved by adding two new operations on types : *expansion* and *rise* in [16] or *lifting* in [21]. S. Ronchi della Rocca in [15] proposed a semi-algorithm for type inference in system \mathcal{D}_ω .

Although all these results give important theoretical benefits, they don't give a good understanding of the structure of principal types or their characteristic properties, and the semi-algorithm proposed in [15] is not practical because of its complexity.

*Address: Projet Cristal, B.P. 105, F-78153 Le Chesnay Cedex, France. Email: {Emilie.Sayag,Michel.Mauny}@inria.fr. Fax: +33 (1) 39 63 53 30.

The work presented here introduces a restriction of system \mathcal{D} which allows us to completely characterize the structure of principal types of terms in normal forms and their properties.

The general outline of this paper is as follows : in section 2, we introduce the type system we study. Section 3 presents an inference algorithm for normal forms and proves the correction and completeness of this algorithm with respect to inference rules of section 2. In section 4, we state and prove the characteristic properties of principal types for normal forms. Finally in section 5 we introduce an algorithm that reconstructs normal forms from types and characterize completely principal types of normal forms.

2 Definitions

We know that in system \mathcal{D} , the notion of principal type of a term doesn't exist in the classical sense [2]. In the same way, in our intersection type system, in order to deduce all possible types for a term from a unique type, substitutions are not sufficient.

For example in our system, as in [5], we can type the term $\lambda x.x (\lambda y.y)$ with the type $[[[\alpha] \rightarrow \alpha] \rightarrow [\beta]] \rightarrow \beta$, but also with the type $[[[\alpha] \rightarrow \alpha, [\gamma] \rightarrow \gamma] \rightarrow \beta] \rightarrow \beta$ where $[\tau_1, \dots, \tau_n]$ denotes intersection of τ_1, \dots, τ_n . Still there is no substitution relating the former (principal type of $(\lambda x.x (\lambda y.y))$ in the classical theory) to the latter.

However in this paper, we will study only terms in normal form and we will prove that for this class of terms, classical notion of principal type exists with a restriction of the term variable inference rule. Then we shall characterize the set of principal types.

First we recall that the set of normal forms can be defined by the following grammar :

$$\begin{array}{lll}
 N & ::= & x \quad \text{variable} \\
 & | & \lambda x.N \quad \text{abstraction} \\
 & | & x N_1 \dots N_n \quad \text{application}
 \end{array}$$

We define next the set \mathcal{T} of intersection types in system \vdash_{NF} :

$$\begin{array}{ll}
 \tau & ::= \alpha \quad \text{type variable} \\
 & | [\tau_1, \dots, \tau_n] \rightarrow \tau \quad \text{for } n \geq 0
 \end{array}$$

where we suppose to have a countable set \mathcal{V} of type variables.

A *substitution* is a mapping from type variables to types, which can be extended in a natural way to a mapping from types to types. The *domain* of a substitution S is the set of type variables which are transformed by S . More formally :

$$Dom(S) = \{\alpha \in \mathcal{V} / S(\alpha) \neq \alpha\}$$

We write $[\alpha/\tau]$ the substitution of domain $\{\alpha\}$ which maps α to τ .

We define the substitution $S_1 + S_2$ where the two substitutions S_1 and S_2 have disjoint domains in the following way :

$$S_1 + S_2(\alpha) = \begin{cases} S_i(\alpha) & \text{if } \alpha \in Dom(S_i), i = 1 \text{ or } 2 \\ \alpha & \text{if } \alpha \notin Dom(S_1) \cup Dom(S_2) \end{cases}$$

The domain of $S_1 + S_2$ is the union of the two domains of S_1 and S_2 . This operation $+$ on substitutions is associative. So we write $S_1 + \dots + S_n$ for $(\dots(S_1 + S_2) + \dots) + S_n$.

We also define a mapping *VarType* from types to type variables, which returns the set of type variables of a type.

We can now define *constraint environments* and their associated operations. This notion of constraint environment was introduced by Z. Shao et A. Appel in [19] where it was called *assumption environment*. Informally, a constraint environment links the free term variables of a λ -term with their type constraints.

Definition 1 A *constraint environment* A , is a mapping from the set \mathcal{V} of term variables to the multi-sets of types.

Notation : the multi-set of types τ_1, \dots, τ_n is written $[\tau_1, \dots, \tau_n]$. The empty multi-set is written $[]$.

Definition 2 Let A be a constraint environment. We define the *domain* of A , denoted as $Dom(A)$ in the following way :

$$Dom(A) = \{x \in \mathcal{V} / A(x) \neq []\}$$

This notion of domain enables us to use the following notation for constraint environments :

Notation : let A be a constraint environment, then if $Dom(A) = \{x_1, \dots, x_n\}$ and for all $i \in \{1, \dots, n\}$, $A(x_i) = [\tau_{i1}, \dots, \tau_{ip_i}]$, we will denote A in this way :

$$\{x_1 : [\tau_{11}, \dots, \tau_{1n_1}], \dots, x_n : [\tau_{n1}, \dots, \tau_{np_n}]\}$$

Now, in order to restrict and extend the domain of a constraint environment, we define two operations in the next definitions.

Definition 3 Let A be a constraint environment and x be a term variable, we define the constraint environment $A \setminus \{x\}$ by :

$$A \setminus \{x\}(y) = \begin{cases} A(y) & \text{if } y \neq x \\ [] & \text{otherwise} \end{cases}$$

Definition 4 Let A_1 and A_2 be two constraint environments, the constraint environment $A_1 + A_2$ is defined as :

$$A_1 + A_2(x) = A_1(x) \cup A_2(x), \text{ for all } x \in \mathcal{V}$$

where \cup is union of multi-sets.

Since the empty multi-set is written $[]$, $[\tau_1, \dots, \tau_n] \rightarrow \tau$ with $n = 0$ is simply denoted as $[] \rightarrow \tau$. So for any term variable x , constraint environment A and any type τ , we can write $A(x) \rightarrow \tau$.

We adopt the usual conventions that allows us to omit parentheses in types and terms, and some other syntactic conventions : x, y, \dots denotes term variables, $\alpha, \beta, \gamma, \dots$ type variables, A, A', A_1, \dots constraint environments.

The type assignment relation \vdash_{NF} , relating λ -terms, types and constraint environments, is defined by the following inference rules :

$$\vdash_{s\omega} x : [\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha; \{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\} \quad (\text{VAR})$$

$$\frac{\vdash_{s\omega} e_1 : \tau_1; A_1}{\vdash_{s\omega} \lambda x. e_1 : A_1(x) \rightarrow \tau_1; A_1 \setminus \{x\}} \quad (\text{ABS})$$

$$\frac{\vdash_{s\omega} e_1 : [] \rightarrow \tau_1; A_1 \quad \vdash_{s\omega} e_2 : \tau_2; A_2}{\vdash_{s\omega} e_1 e_2 : \tau_1; A_1 + A_2} \quad (\text{APP1})$$

$$\frac{\vdash_{s\omega} e_1 : [\tau_2^1, \dots, \tau_2^m] \rightarrow \tau_1; A_1 \quad \vdash_{s\omega} e_2 : \tau_2^1; A_2^1 \quad \dots \quad \vdash_{s\omega} e_2 : \tau_2^m; A_2^m}{\vdash_{s\omega} e_1 e_2 : \tau_1; A_1 + A_2^1 + \dots + A_2^m} \quad (\text{APP2})$$

From now on, the type assignment relation \vdash_{NF} will be written \vdash , since it the only relation that we consider in this paper.

3 Type assignment

We provide here a type assignment algorithm for normal forms and we prove its soundness and completeness with respect to the inference rules given in the previous section. This algorithm is not really original, since we can find similar definitions in [5, 16, 15, 21, 11] which study the principal type property in the intersection type discipline. Their authors introduce the notion of *approximant* or *λ - Ω -normal forms*, and define principal typing for these extended normal forms before generalizing to λ -terms using an approximation property, i.e. $B \vdash e : \tau$ if and only if there exists an approximant a of e such that $B \vdash a : \tau$. The novelty in the work presented here, is not the type assignment algorithm itself but the study of the structure of pairs inferred by this algorithm.

In this algorithm, we use the notion of *new type variable*. This notion could be formalize without difficulty (see for example [10]), but the presentation would be uselessly more complex. So we choose to just name it informally.

$Infer(N) =$

- **Case** $N = x$
let α be a new type variable
return $(\alpha, \{x : [\alpha]\})$
- **Case** $N = \lambda x. N_1$
let $(\tau_1, A_1) = Infer(N_1)$
return $(A_1(x) \rightarrow \tau_1, A_1 \setminus \{x\})$
- **Case** $N = x N_1 \dots N_n$
let $(\tau_1, A_1) = Infer(N_1)$
 \vdots
 $(\tau_n, A_n) = Infer(N_n)$
 α be a new type variable
return $(\alpha, \{x : [\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha\} + A_1 + \dots + A_n)$

This algorithm is well defined modulo the name of type variables, since we don't fix the choice of the new type variables. This algorithm is sound and complete, as expressed by the two following theorems. Their proofs can be found in [18]¹.

Theorem 1 *Let N be a normal form, if $Infer(N) = (\tau, A)$, then $\vdash N : \tau; A$.*

Theorem 2 *Let N be a normal form such that $\vdash N : \tau; A$, then $Infer(N) = (\tau_p, A_p)$ and there exists a substitution S such that $S(\tau_p) = \tau$ and $S(A_p) = A$.*

In the following, the *principal pair* of a λ -term is the pair of type and constraint environment given by $Infer$. The previous theorem justifies this name.

¹They appear as an annex in this submission version.

4 Characterization of principal pairs

In order to characterize the set of principal pairs of normal forms, we define a set of types \mathcal{T}_{NF} and a set of constraint environments \mathcal{E} and we prove that the algorithm *Infer* produces only types and constraint environments belonging to \mathcal{T}_{NF} and \mathcal{E} respectively. Then we restrict further this two sets to provide a complete characterization of the set of pairs (τ, A) given by *Infer*.

4.1 Principal pairs belong to $\mathcal{T}_{NF} \times \mathcal{E}$

We first define \mathcal{T}_E and \mathcal{T}_{NF} , two sub-sets of \mathcal{T} in a mutually recursive way. In the following, $\sigma, \sigma', \sigma_1, \dots$ will denote an element of \mathcal{T}_E , $\tau, \tau', \tau_1, \dots$ an element of \mathcal{T}_{NF} . \mathcal{T}_E is the set of types of constraint environments and is defined by :

$$\begin{array}{l} \sigma ::= \alpha \\ \quad | [\tau] \rightarrow \sigma \end{array}$$

with $VarType(\tau) \cap VarType(\sigma) = \emptyset$.

Then we define the set \mathcal{T}_{NF} of principal types :

$$\begin{array}{l} \tau ::= \alpha \\ \quad | [\sigma_1, \dots, \sigma_n] \rightarrow \tau \end{array}$$

with $n \geq 0$.

When we will not want to specify whether a type belongs to \mathcal{T}_E or to \mathcal{T}_{NF} , we denote it as $\rho, \rho', \rho_1, \dots$.

Some types belong to both \mathcal{T}_E and \mathcal{T}_{NF} , since each type variable belongs at the same time to \mathcal{T}_E and to \mathcal{T}_{NF} . Moreover, if ρ_1 and ρ belong to $\mathcal{T}_E \cap \mathcal{T}_{NF}$, then $[\rho_1] \rightarrow \rho \in \mathcal{T}_E \cap \mathcal{T}_{NF}$ since $[\rho_1] \rightarrow \rho$ have the shape of $[\tau] \rightarrow \sigma$ but also the shape of $[\sigma_1, \dots, \sigma_n] \rightarrow \tau$ with $n = 1$. Thus, in definitions about types in \mathcal{T}_E and in \mathcal{T}_{NF} , we should verify that the two cases of definition which apply to types in $\mathcal{T}_E \cap \mathcal{T}_{NF}$ are consistent.

Finally, we define the set \mathcal{E} of principal constraint environments :

$$\begin{array}{l} A ::= \{ \} \\ \quad | \{x : [\sigma_1, \dots, \sigma_n]\} \\ \quad | A_1 + A_2 \end{array}$$

Since \mathcal{T}_E and \mathcal{T}_{NF} are sub-sets of \mathcal{T} , we can apply to elements of \mathcal{T}_E , \mathcal{T}_{NF} or \mathcal{E} , all functions defined for types in \mathcal{T} or for constraint environments.

We notice here that the only difference with a general constraint environment is that the type constraints only belong to \mathcal{T}_E . The following lemma proves that principal pairs are elements of $\mathcal{T}_{NF} \times \mathcal{E}$.

Lemma 4.1 *Let N be a normal form, if $Infer(N) = (\tau, A)$, then $\tau \in \mathcal{T}_{NF}$ and $A \in \mathcal{E}$.*

Proof by induction on the structure of N .

- Case $N = x$.

$Infer(x) = (\alpha, \{x : [\alpha]\})$ and we have $\alpha \in \mathcal{T}_{NF}$ and $\alpha \in \mathcal{T}_E$, so $\{x : [\alpha]\} \in \mathcal{E}$.

- Case $N = \lambda x. N_1$.

$Infer(N_1) = (\tau_1, A_1)$ and by induction hypothesis, $\tau_1 \in \mathcal{T}_{NF}$ and $A_1 \in \mathcal{E}$.

If we denote $A_1(x) = [\sigma_1, \dots, \sigma_n]$ with $n \geq 0$, then

$$Infer(\lambda x. N_1) = ([\sigma^1, \dots, \sigma^n] \rightarrow \tau_1, A_1 \setminus \{x\})$$

Since $A_1 \in \mathcal{E}$, if $n \geq 1$, $\sigma^i \in \mathcal{T}_E$ for all $i = 1, \dots, n$, and since $\tau_1 \in \mathcal{T}_{NF}$, we have $[\sigma^1, \dots, \sigma^n] \rightarrow \tau_1 \in \mathcal{T}_{NF}$ for $n \geq 0$ and $A_1 \setminus \{x\} \in \mathcal{E}$.

- Case $N = x N_1 \dots N_n$.

Let α be a new type variable and for all $i = 1, \dots, n$, $(\tau_i, A_i) = \text{Infer}(N_i)$. Then

$$\text{Infer}(x N_1 \dots N_n) = (\alpha, \{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\} + A_1 + \dots + A_n)$$

Now $\alpha \in \mathcal{T}_E$ and $\alpha \in \mathcal{T}_{NF}$, moreover by induction hypothesis, for all $i = 1, \dots, n$, $\tau_i \in \mathcal{T}_{NF}$ and $A_i \in \mathcal{E}$. Therefore we must prove that $[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha \in \mathcal{T}_E$, which is immediate.

□

If we regard Infer as a function from the set of normal forms to the set of pairs (τ, A) , we can define $\text{Image}(\text{Infer})$ by the set of pairs (τ, A) such that there exists a normal form N which verifies $\text{Infer}(N) = (\tau, A)$. So we can give the previous lemma in the following way :

$$\text{Image}(\text{Infer}) \subset \mathcal{T}_{NF} \times \mathcal{E}$$

4.2 Restriction of $\mathcal{T}_{NF} \times \mathcal{E}$

Now we want be more precise about the structure of principal pairs, so we define several easy notions which restrict the set of pairs $\mathcal{T}_{NF} \times \mathcal{E}$ in order to obtain the set of principal pairs.

4.2.1 Occurrences

The notion of *occurrence* has been often used in literature. Different refinements have been defined (see for example [8], page 33 and 34 for *positive*, *negative* and *final occurrences* and [15] for level of an occurrence of a type) according to the precision of informations that the author wants. In our case, we do not need to be very precise, we only need to distinguish between three kinds of occurrences.

Definition 5 We define the *positive* and *negative occurrences* of a type variable α in a type ρ by induction on the structure of ρ in the following way :

- if ρ is a type variable, then the possible occurrence of α in ρ is positive
- if $\rho = [\tau] \rightarrow \sigma$, then the positive (respectively negative) occurrences of α in ρ are the positive (respectively negative) occurrence of α in σ , and the negative (respectively positive) occurrences of α in τ
- if $\rho = [\sigma^1, \dots, \sigma^n] \rightarrow \tau$, then the positive (respectively negative) occurrences of α in ρ are the positive (respectively negative) occurrences of α in τ and if $n \geq 1$, the negative occurrences of α in σ^i for $i = 1, \dots, n$.

Remark: It is easy to prove that for types in $\mathcal{T}_E \cap \mathcal{T}_{NF}$, cases 2 and 3 of this definition, define the same occurrences.

Definition 6 Let A be a constraint environment and α be a type variable, the positive (respectively negative) occurrences of α in A are the negative (respectively positive) occurrences of α in each type of A .

We notice here that a type constraint has a negative sign like types in left hand side of an arrow type.

Definition 7 Let ρ be a type in $\mathcal{T}_{NF} \cup \mathcal{T}_E$. We say that α is the *final occurrence* of τ if we are in one of the following three cases :

- $\rho = \alpha$
- $\rho = [\tau] \rightarrow \sigma$ and α is the final occurrence of σ
- $\rho = [\sigma^1, \dots, \sigma^n] \rightarrow \tau$ and α is the final occurrence of τ .

Remark: If a type variable is the final occurrence of some type then this occurrence is positive.

Afterwards, we will need to distinguish sub-terms of a type which occur in left hand side arrows.

Definition 8 Let $\rho \in \mathcal{T}_{NF} \cup \mathcal{T}_E$, the set $G(\rho)$ of *left sub-terms* of ρ is defined by induction on the structure of ρ , in the following way :

- if $\rho = \alpha$, $G(\rho) = \emptyset$
- if $\rho = [\tau] \rightarrow \sigma$, $G(\rho) = \{\tau\} \cup G(\sigma)$
- if $\rho = [\sigma_1, \dots, \sigma_n] \rightarrow \tau$, $G(\rho) = \{\sigma_1, \dots, \sigma_n\} \cup G(\tau)$.

According to the previous remark, we verify that for any type $[\rho_1] \rightarrow \rho$ in $\mathcal{T}_E \cap \mathcal{T}_{NF}$, the two cases of the definition of G which can apply to $[\rho_1] \rightarrow \rho$, define the same set. We have $G([\rho_1] \rightarrow \rho) = \{\rho_1\} \cup G(\rho)$ if we regard $[\rho_1] \rightarrow \rho$ as an element of \mathcal{T}_E , but also in case we regard it as an element of \mathcal{T}_{NF} .

4.2.2 A-types

To handle consistently types and type constraints, we introduce A-types, using a double arrow in order to write negatively type constraints. Formally, we define the set of A-types by the following grammar :

$$T ::= [\sigma_1, \dots, \sigma_n] \Rightarrow \tau \quad \text{with } n \geq 0 \\ | [\sigma_1, \dots, \sigma_n] \Rightarrow \quad \text{with } n \geq 1$$

where $[\sigma_1, \dots, \sigma_n]$ is a multi-set.

The function *VarType*, returning the set of types variables of a type, is naturally extended to A-types.

Since the algorithm *Infer* returns a pair (type,constraint environment), in the following we often want to pass from this pair to the correspondent A-type. So we define an operation which simply consists of erasing all term variables in a constraint environment in order to obtain a single multi-set of type constraints.

Definition 9 Let A be a constraint environment, we define \overline{A} by induction on the structure of A :

- if $A = \{ \}$, then $\overline{A} = []$
- if $A = \{x : [\sigma_1, \dots, \sigma_n]\}$, then $\overline{A} = [\sigma_1, \dots, \sigma_n]$
- if $A = A_1 + A_2$, then $\overline{A} = \overline{A_1} \cup \overline{A_2}$

So for any type τ and any constraint environment A , $\overline{A} \Rightarrow \tau$ is a A-type.

Definition 10 We say that T' is *held* in T , if $T = [\sigma_1, \dots, \sigma_n] \Rightarrow \tau$ and T' is one of the following forms : $[\sigma_{i_1}, \dots, \sigma_{i_p}] \Rightarrow \tau$ or $[\sigma_{i_1}, \dots, \sigma_{i_p}] \Rightarrow$ with $[\sigma_{i_1}, \dots, \sigma_{i_p}]$ sub-multi-set (possibly empty in the first case) of $[\sigma_1, \dots, \sigma_n]$.

Moreover, if T' is held in T and distinct of T , we say that T' is *strictly held* in T .

Extending the notion of left sub-terms of a type, we define for all A-type T , the set $G(T)$ of left sub-terms of T by induction on the structure of T :

- if $T = [\sigma_1, \dots, \sigma_n] \Rightarrow \tau$, $G(T) = \{\sigma_1, \dots, \sigma_n\} \cup G(\tau)$
- if $T = [\sigma_1, \dots, \sigma_n] \Rightarrow$, $G(T) = \{\sigma_1, \dots, \sigma_n\}$.

In the same way, we extend to A-types the notion of sign of occurrences. Let T be a A-type and α a type variable. The positive (respectively negative) occurrences of α in T are defined by induction on the structure of T :

- if $T = [\sigma_1, \dots, \sigma_n] \Rightarrow \tau$, then the positive (respectively negative) occurrences of α in T are the positive (respectively negative) occurrences of α in τ and the negative (respectively positive) of α in σ_i for $i = 1, \dots, n$
- if $T = [\sigma_1, \dots, \sigma_n] \Rightarrow$, then the positive (respectively negative) occurrences of α in T are the negative (respectively positive) of α in σ_i pour $i = 1, \dots, n$.

4.2.3 Closed A-types

Definition 11 A A-type T is *closed* if each type variable α of $VarType(T)$ has exactly one positive occurrence and one negative occurrence in T .

Definition 12 Let $T = [\sigma_1, \dots, \sigma_n] \Rightarrow \tau$ be a A-type. T is *finally closed* if the final occurrence of τ is also the final occurrence of a $G(T)$ type.

Definition 13 Let T be a A-type. T is *minimally closed* if there is no closed A-type strictly held in T .

Example : $T = [[\alpha, [[\alpha] \rightarrow \beta] \rightarrow \beta] \rightarrow \gamma, [\delta] \rightarrow \delta] \Rightarrow \gamma$ is closed, finally closed but not minimally closed because the A-type $[[\alpha, [[\alpha] \rightarrow \beta] \rightarrow \beta] \gamma] \Rightarrow \gamma$ is closed and strictly held in T

4.2.4 Properties of principal typing

In this section, we define properties of A-types which characterize principal pairs. The main result is the last theorem of this section.

The following definition gives a short way to talk about the three previous properties in the same time. We prove in lemma 4.2 that a principal pair always satisfies these properties.

Definition 14 Let $T = [\sigma_1, \dots, \sigma_n] \Rightarrow \tau$ be a A-type. We say that T is *complete* if

- T is closed
- T is finally closed
- T is minimally closed.

Lemma 4.2 Let N be a normal form, if $Infer(N) = (\tau, A)$, then $\overline{A} \Rightarrow \tau$ is complete.

Proof by structural induction on N .

- Case $N = x$.

Then $Infer(x) = (\alpha, \{x : [\alpha]\})$ and $A \Rightarrow \tau = \{\alpha\} \Rightarrow \alpha$. Thus we have :

- α is the only type variable of $A \Rightarrow \tau$ and it has a positive occurrence and a negative occurrence in $A \Rightarrow \tau$. So $[\alpha] \Rightarrow \alpha$ is closed.

- α is the final occurrence of α . Furthermore, $G(A \Rightarrow \tau) = \{\alpha\}$. Therefore $[\alpha] \Rightarrow \alpha$ is finally closed.

- The only A-types strictly held in $A \Rightarrow \tau$ are $[\alpha] \Rightarrow$ and $[\] \Rightarrow \alpha$ which are not closed. So $[\alpha] \Rightarrow \alpha$ is minimally closed.

- Case $N = \lambda x. N_1$.

Let $(\tau_1, A_1) = Infer(N_1)$. By induction hypothesis $\overline{A_1} \Rightarrow \tau_1$ is complete.

If we denote $A_1(x) = [\sigma^1, \dots, \sigma^n]$ with $n \geq 0$, then $Infer(\lambda x. N_1) = ([\sigma^1, \dots, \sigma^n] \rightarrow \tau_1, A_1 \setminus \{x\})$.

- $VarType(\overline{A_1 \setminus \{x\}} \Rightarrow [\sigma^1, \dots, \sigma^n] \rightarrow \tau_1) = VarType(\overline{A_1} \Rightarrow \tau_1)$. Furthermore, occurrences of the σ^i 's type variables keep the same sign in $\overline{A_1 \setminus \{x\}} \Rightarrow [\sigma^1, \dots, \sigma^n] \rightarrow \tau_1$ and in $\overline{A_1} \Rightarrow \tau_1$. Hence $\overline{A_1 \setminus \{x\}} \Rightarrow [\sigma^1, \dots, \sigma^n] \rightarrow \tau_1$ is closed since $\overline{A_1} \Rightarrow \tau_1$ is closed by induction hypothesis.

- On the other hand, the final occurrence of $[\sigma^1, \dots, \sigma^n] \rightarrow \tau_1$ and the one of τ_1 are the same and $G(\overline{A_1 \setminus \{x\}} \Rightarrow [\sigma^1, \dots, \sigma^n] \rightarrow \tau_1) = G(\overline{A_1} \Rightarrow \tau_1)$. Therefore, by induction hypothesis, $\overline{A_1 \setminus \{x\}} \Rightarrow [\sigma^1, \dots, \sigma^n] \rightarrow \tau_1$ is finally closed.

– Let T be a A-type strictly held in $\overline{A_1 \setminus \{x\}} \Rightarrow [\sigma^1, \dots, \sigma^n] \rightarrow \tau_1$, then there exists T' strictly held in $\overline{A_1} \Rightarrow \tau_1$ such that $\text{VarType}(T) = \text{VarType}(T')$. Furthermore, occurrences of type variables in $\overline{A_1} \Rightarrow \tau_1$ keep the same sign in $\overline{A_1 \setminus \{x\}} \Rightarrow [\sigma^1, \dots, \sigma^n] \rightarrow \tau_1$. So for all closed T strictly held in $\overline{A_1 \setminus \{x\}} \Rightarrow [\sigma^1, \dots, \sigma^n] \rightarrow \tau_1$, there exists a closed T' held in $\overline{A_1} \Rightarrow \tau_1$. Hence by induction hypothesis, there doesn't exist any A-type strictly held in $\overline{A_1 \setminus \{x\}} \Rightarrow [\sigma^1, \dots, \sigma^n] \rightarrow \tau_1$ which is closed. So we deduce that $\overline{A_1 \setminus \{x\}} \Rightarrow [\sigma^1, \dots, \sigma^n] \rightarrow \tau_1$ is minimally closed.

• Case $N = x N_1 \dots N_n$.

Let α be a new type variable and for $i = 1, \dots, n$, $(\tau_i, A_i) = \text{Infer}(N_i)$. Then

$$\text{Infer}(x N_1 \dots N_n) = (\alpha, \{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\} + A_1 + \dots + A_n)$$

We denote $A = \{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\} + A_1 + \dots + A_n$.

By induction hypothesis for all $i = 1, \dots, n$, $\overline{A_i} \Rightarrow \tau_i$ is complete.

– $\text{VarType}(\overline{A} \Rightarrow \alpha) = \bigcup_{i=1}^n \overline{A_i} \Rightarrow \tau_i \cup \{\alpha\}$. Now for all $i = 1, \dots, n$, if $\beta \in \bigcup_{i=1}^n \overline{A_i} \Rightarrow \tau_i$, then the occurrences of β have the same sign in $\overline{A_i} \Rightarrow \tau_i$ and in $\overline{A} \Rightarrow \alpha$. Furthermore, α has a positive occurrence and a negative occurrence in $\overline{A} \Rightarrow \alpha$. So $\overline{A} \Rightarrow \alpha$ is closed.

– The final occurrence of α is also the final occurrence of $[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha$ which belongs to $G(\overline{A} \Rightarrow \alpha)$. So $\overline{A} \Rightarrow \alpha$ is finally closed.

– Let T be a A-type held in $\overline{A} \Rightarrow \alpha$. The sets $\text{VarType}(\overline{A_i} \Rightarrow \tau_i)$ are disjoint, since they result from disjoint calls to *Infer*. Furthermore, for all $i = 1, \dots, n$, $\overline{A_i} \Rightarrow \tau_i$ is minimally closed.

Since T is closed, every type of A_i and every τ_i must occur in T . Now the only way for T to have an occurrence of each τ_i is to have an occurrence of $[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha$. But if $[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha$ occurs in T and if T is closed, then α must also occur in T since α is a new type variable which occurs neither in A_i nor in τ_i .

We conclude that T is $\overline{A} \Rightarrow \alpha$ and therefore that $\overline{A} \Rightarrow \alpha$ is minimally closed.

□

The next definition specifies the structure of principal pairs. Intuitively, we want to say that for any given normal form, we can find the principal pair of these sub-terms from its principal pair. Since it is easier to study a A-type than a pair (type, constraint environment), we formalize our intuition on A-types.

Definition 15 Let T be a A-type. We say that T is *principal* if it is complete and if it is in one of the following cases :

- $T = [\alpha] \Rightarrow \alpha$.
- $T = [\sigma_1, \dots, \sigma_n] \Rightarrow \alpha$ and there exists a partition $(E_j)_{j=1, \dots, p}$ of the multi-set $[\sigma_1, \dots, \sigma_{i-1}, \sigma_{i+1}, \dots, \sigma_n]$ such that each $E_j \Rightarrow \tau_j$ is principal where i and the τ_j , $j = 1, \dots, p$ are defined by $i \in \{1, \dots, n\}$ such that σ_i has the shape of $[\tau_1] \rightarrow \dots \rightarrow [\tau_p] \rightarrow \alpha$, $p > 0$.
- $T = [\sigma_1, \dots, \sigma_n] \Rightarrow [\sigma^1, \dots, \sigma^p] \rightarrow \tau'$ and $[\sigma_1, \dots, \sigma_n, \sigma^1, \dots, \sigma^p] \Rightarrow \tau'$ is principal.

Lemma 4.3 Let N be a normal form. If $\text{Infer}(N) = (\tau, A)$, then $\overline{A} \Rightarrow \tau$ is principal.

Proof by structural induction on N .

• Case $N = x$.

Then $\text{Infer}(N) = (\alpha, \{x : [\alpha]\})$ and $[\alpha] \Rightarrow \alpha$ is principal by definition.

• Case $N = \lambda x. N_1$.

Let $(\tau_1, A_1) = \text{Infer}(N_1)$. By induction hypothesis, $\overline{A_1} \Rightarrow \tau_1$ is principal.

If we denote $A_1(x) = [\sigma^1, \dots, \sigma^n]$, then we have :

$$\text{Infer}(N) = ([\sigma^1, \dots, \sigma^n] \rightarrow \tau_1, A_1 \setminus \{x\})$$

By definition of a principal A-type, in order to prove that $\overline{A_1 \setminus \{x\}} \Rightarrow [\sigma^1, \dots, \sigma^n] \rightarrow \tau_1$ is principal, it is enough to prove that $\overline{A_1 \setminus \{x\}} \cup \{\sigma^1, \dots, \sigma^n\} \Rightarrow \tau_1$ is principal. Now $\overline{A_1 \setminus \{x\}} \cup \{\sigma^1, \dots, \sigma^n\} = \overline{A_1}$ and we conclude with the induction hypothesis.

- Case $N = x N_1 \dots N_n$.

Let α be a new type variable and for $i = 1, \dots, n$, $(\tau_i, A_i) = \text{Infer}(N_i)$. By induction hypothesis each $\overline{A_i} \Rightarrow \tau_i$ is principal and in particular complete. Furthermore, we have

$$\text{Infer}(N) = (\alpha, \{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\}) + A_1 + \dots + A_n$$

It is the second case of principal A-type definition and we must prove that there exists a partition of $\overline{A_1 + \dots + A_n}$ in n sub-multi-sets E_1, \dots, E_n such that each $E_i \Rightarrow \tau_i$ is principal.

Now $(\overline{A_i})_{i=1, \dots, n}$ is a partition of $\overline{A_1 + \dots + A_n}$ such that each $\overline{A_i} \Rightarrow \tau_i$ is principal by induction hypothesis.

Thus $\{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\} + A_1 + \dots + A_n \Rightarrow \alpha$ is principal.

□

If we denote $\mathcal{P} = \{(\tau, A)/\tau \in \mathcal{T}_{NF}, A \in \mathcal{E} \text{ and } \overline{A} \Rightarrow \tau \text{ is principal}\}$, the previous lemma proves the following inclusion :

$$\text{Image}(\text{Infer}) \subset \mathcal{P}$$

5 Reconstruction of normal forms

We have proved several properties of principal pairs and deduced the inclusion of $\text{Image}(\text{Infer})$ in a particular set of pairs composed of types of \mathcal{T}_{NF} and constraint environments of \mathcal{E} . In order to characterize the principal pairs of normal forms, we want to prove the opposite inclusion. We will give an algorithm \mathcal{R} which allows us to reconstruct from a type and a constraint environment, a normal form typeable with this type and this constraint environment.

5.1 Reconstruction algorithm

Notation : Let α be a type variable and A be a constraints environment. We denote $F(\alpha, A)$ the set of types which belong to A and have α as final occurrence. More precisely :

$$F(\alpha, A) = \{(\tau, x)/\tau \in A(x) \text{ and } \alpha \text{ is the final occurrence of } \tau\}$$

$\mathcal{R}(\tau, A) =$

- Case $(\alpha, \{\})$
fail

- Case (α, A)

let $\{(\tau^1, x_1), \dots, (\tau^m, x_m)\} = F(\alpha, A)$

if $m = 1$

then let $\tau^1 = [\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha$

if for $i = 1, \dots, n$, there exist $A_i \subset A$ such that $\overline{A_i} \Rightarrow \tau_i$ is principal

then let $(N_1, A'_1) = \mathcal{R}(\tau_1, A_1)$

⋮

$(N_n, A'_n) = \mathcal{R}(\tau_n, A_n)$

$A' = \{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\} + A_1 + \dots + A_n$

return $(x N_1 \dots N_n, A \setminus A' + A'_1 + \dots + A'_n)$

else fail
else fail

- **Case** $([\sigma^1, \dots, \sigma^n] \rightarrow \tau', A)$
 let x be a new term variable
 $A' = A + \{x : [\sigma^1, \dots, \sigma^n]\}$
 $(N, A'') = \mathcal{R}(\tau', A')$
 if $A''(x) = []$
 then return $(\lambda x.N, A'')$
 else fail

Remarks:

- $\{\} \Rightarrow \alpha$ is not closed and so the first case of \mathcal{R} is impossible with a pair (τ, A) belonging to \mathcal{P} .
- In the case (α, A) with $F(\alpha, A) = \{(\tau^1, x_1), \dots, (\tau^m, x_m)\}$ and $m \neq 1$, $\overline{A} \Rightarrow \alpha$ is not finally closed. Therefore this case is impossible with a pair (τ, A) of \mathcal{P} .
 Moreover if $F(\alpha, A) = ([\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha, x)$ with $n = 0$, then we have $\mathcal{R}(\alpha, A) = (x, A \setminus \{x : [\alpha]\})$.
 Now for an element of \mathcal{P} , $\overline{A} \Rightarrow \alpha$ is minimally closed and $A = \{x : [\alpha]\}$. So $\mathcal{R}(\alpha, A) = (x, \{\})$.
- If $\mathcal{R}(\tau, A) = (N, B)$, then $B \subseteq A$. The proof is immediate by recurrence on the number of calls to \mathcal{R} .

5.2 Properties

Lemma 5.4 *If $(\tau, A) \in \mathcal{P}$, then $\mathcal{R}(\tau, A) = (N, A')$ is well defined and $A' = \{\}$.*

Proof by recurrence on the number of calls to \mathcal{R} .

- Case (α, A) .

Let $([\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha, x) = F(\alpha, A)$. So we have $A = \{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\} + B$ for some constraint environment B .

From the previous remark, if $n = 0$, then $\mathcal{R}(\alpha, A) = (x, \{\})$ is well defined. So we can suppose $n \geq 1$.

By hypothesis we know that $\overline{A} \Rightarrow \alpha$ is principal. Thus, by definition of a principal A-type, there exists a partition $(E_i)_{i=1, \dots, n}$ of \overline{B} such that each $E_i \Rightarrow \tau_i$ is principal. Now we can construct from B and for each E_i , a constraint environment A_i such that $\overline{A_i} = E_i$.

Therefore there exist n sub-environments A_1, \dots, A_n of B such that $\overline{A_i} \Rightarrow \tau_i$ is principal for all $i \in \{1, \dots, n\}$. Since each A_i is a sub-environment of B , it is also a sub-environment of A . So in order to prove that $\mathcal{R}(\tau, A)$ is well defined, it is enough to prove that $\mathcal{R}(\tau_i, A_i)$ is well defined for all $i \in \{1, \dots, n\}$, by definition of \mathcal{R} . Since each $\overline{A_i} \Rightarrow \tau_i$ is principal, we can apply the recurrence hypothesis to $\mathcal{R}(\tau_i, A_i)$ for $i = 1, \dots, n$. Thus for $i = 1, \dots, n$, $\mathcal{R}(\tau_i, A_i) = (N_i, A'_i)$ is well defined and $A'_i = \{\}$.

From this we deduce that $\mathcal{R}(\alpha, A) = (x N_1 \dots N_n, A')$ is well defined with $A' = A \setminus (\{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\} + A_1 + \dots + A_n) + \{\}$

Now if we denote $A'' = \{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\} + A_1 + \dots + A_n$, we have $A'' = A$.

By definition of A_i and since $([\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha, x) = F(\alpha, A)$ we have $\overline{A''} \subset \overline{A}$. Moreover, $\overline{A''} \Rightarrow \alpha$ is closed, because each $\overline{A_i} \Rightarrow \tau_i$ is closed by definition of a principal A-type, because the sign of type variables occurrences in $\overline{A_i} \Rightarrow \tau_i$ doesn't change in $\overline{A''} \Rightarrow \alpha$ and because α have exactly one positive occurrence and one negative occurrence in $\overline{A''} \Rightarrow \alpha$. But $\overline{A''} \Rightarrow \alpha$ is hold in $\overline{A} \Rightarrow \alpha$. Now by hypothesis, $\overline{A} \Rightarrow \alpha$ is minimally closed, so $\overline{A''} \Rightarrow \alpha$ can't be strictly hold in \overline{A} and we have

$$A = \{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\} + A_1 + \dots + A_n$$

Therefore $A' = A \setminus A'' = \{\}$

- Case $([\sigma^1, \dots, \sigma^n] \rightarrow \tau', A)$.

Let $A' = \{x : [\sigma^1, \dots, \sigma^n]\} + A$. Since $\overline{A} \Rightarrow [\sigma^1, \dots, \sigma^n] \rightarrow \tau'$ is principal, $\overline{A'} \Rightarrow \tau'$ is also principal. Thus by recurrence hypothesis, $\mathcal{R}(\tau', A') = (N, A'')$ is well defined and $A'' = \{\}$. Now $\{\}(x) = []$, so $\mathcal{R}([\sigma^1, \dots, \sigma^n] \rightarrow \tau', A) = (\lambda x.N, A'')$ is well defined by definition of \mathcal{R} , and $A'' = \{\}$.

□

Remark: In the following we will write $\mathcal{R}(\tau, A) = N$ instead of $\mathcal{R}(\tau, A) = (N, \{\})$ if the pair (τ, A) belongs to \mathcal{P} .

Lemma 5.5 *Let $(\tau, A) \in \mathcal{P}$ and $N = \mathcal{R}(\tau, A)$, then $Infer(N) = (\tau, A)$.*

Proof by recurrence on the number of calls to \mathcal{R} .

- Case (α, A) .

Let $([\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha, x) = F(\alpha, A)$. According to the previous demonstration there exist $(A_i)_{i=1, \dots, n}$ n sub-environments of A such that $\overline{A_i} \Rightarrow \tau_i$ is principal for $i = 1, \dots, n$. So we can apply the recurrence hypothesis to each call of \mathcal{R} with (τ_i, A_i) .

Thus for $i = 1, \dots, n$, if $\mathcal{R}(\tau_i, A_i) = N_i$, then $Infer(N_i) = (\tau_i, A_i)$.

Since $A = \{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\} + A_1 + \dots + A_n$, according to the demonstration of lemma 4.4, and since $\mathcal{R}(\alpha, A) = x N_1 \dots N_n$, $Infer(x N_1 \dots N_n) = (\alpha, A)$ by definition of $Infer$.

- Case $([\sigma^1, \dots, \sigma^n] \rightarrow \tau', A)$.

Let $A' = \{x : [\sigma^1, \dots, \sigma^n]\} + A$ where x is a new term variable.

Since $\overline{A} \Rightarrow [\sigma^1, \dots, \sigma^n] \rightarrow \tau'$ is principal, $\overline{A'} \Rightarrow \tau'$ is also principal. Thus, by recurrence hypothesis, if $\mathcal{R}(\tau', A') = N$, then $Infer(N) = (\tau', A')$. Now $A'(x) = [\sigma^1, \dots, \sigma^n]$ because x is a term variable which doesn't belong to the domain of A . We deduce from this that $Infer(\lambda x.N) = ([\sigma^1, \dots, \sigma^n] \rightarrow \tau', A' \setminus \{x\})$.

Since $A' \setminus \{x\} = A$, we can conclude.

□

Theorem3

$$Image(Infer) = \mathcal{P}$$

in other words : *The types and the constraint environments inferred for normal forms are exactly the pairs (τ, A) such that $\overline{A} \Rightarrow \tau$ is principal.*

Proof

The lemma 3.3 give $Image(Infer) \subset \mathcal{P}$. Moreover according to lemma 5.5, for any $(\tau, A) \in \mathcal{P}$, there exists a normal form N such that $Infer(N) = (\tau, A)$. So $\mathcal{P} \subset Image(Infer)$ and we can conclude. □

6 Conclusion

In this paper we have introduced a restriction of system \mathcal{D} in the sense that intersections only occur in the left hand side of an arrow and are not types. This restriction corresponds to strict types defined by S. van Bakel in [20, 21]. But in the strict types system, S. van Bakel defines a partial order on types. His system is a restriction of the system defined in [1]. He proves that despite the restriction, his system verifies the same properties as the original system. Moreover his approach to the construction of principal pairs uses the approximate normal forms of a λ -term and he is only interested in the existence of the principal type property, not in the structure of principal pairs.

Furthermore, our system further restricts van Bakel's system because of the specified structure of types which are assigned to term variables. This restriction leads to a principal type property in the classical sense for normal forms.

We then have completely characterized the structure and the properties of principal types inferred for normal forms and we have given an algorithm for reconstruction of normal forms from principal types.

Extension of this work for general λ -terms would give us a better comprehension of links between typing and β -reduction.

References

- [1] H. P. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *Journal of Symbolic Logic*, 48(4):931–940, 1983.
- [2] F. Cardone and M. Coppo. Two extensions of curry’s type inference system. In P. Odifreddi, editor, *Logic in Computer Science*, volume 31, pages 19–75. APIC Studies in Data Processing, Academic Press, 1990.
- [3] M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ -calculus. *Notre Dame Journal of Formal Logic*, 21(4):685–693, 1980.
- [4] M. Coppo, M. Dezani-Ciancaglini, and P. Sallé. Functional characterization of some semantic equalities inside λ -calculus. In E. Maurer, editor, *Automata, Languages and Programming*, volume 71 of *Lecture Notes in Computer Science*, pages 133–146. Springer-Verlag, Berlin, 1979.
- [5] M. Coppo, M. Dezani-Ciancaglini, and B. Venneri. Principal type schemes and λ -calculus semantics. In J. P. Seldin and J. R. Hindley, editors, *To H.B. Curry : Essays in Combinatory Logic, Lambda-calculus and Formalism*, pages 536–560. Academic Press, Londres, 1980.
- [6] M. Coppo, M. Dezani-Ciancaglini, and B. Venneri. Functional characters of solvable terms. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 27:45–58, 1981.
- [7] P. Gardner. Discovering needed reductions using type theory. In M. Hagiya and J. C. Mitchell, editors, *Proceedings of TACS’94*, volume 789 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.
- [8] J.L. Krivine. *Lambda-calcul, types et modèles*. Etudes et Recherches en Informatique. Masson, 1990.
- [9] D. Leivant. Typing and computational properties of lambda-expressions. *Theoretical Computer Science*, 44:51–68, 1986.
- [10] X. Leroy. *Typage polymorphe d’un langage algorithmique*. PhD thesis, Université Paris 7, Juin 1992.
- [11] I. Margaria and Maddalena Zacchi. Principal typing in a $\forall\lambda$ -discipline. *Journal of Logic Computation*, 5(3):367–381, 1995.
- [12] R. Milner. A theory of types polymorphism in programming. *Computer and System Sciences*, 17(3):348–375, 1978.
- [13] B. Pierce. A decision procedure for the subtype relation on intersection types with bounded variables. Technical Report CMU-CS-89-169, School of Computer Science, Carnegie Mellon University, Septembre 1989.
- [14] G. Pottinger. A type assignment for the strongly normalizable λ -terms. In J. R. Hindley and J. P. Seldin, editors, *To H.B. Curry : Essays in Combinatory Logic, λ -Calculus and Formalism*, pages 561–577. Academic Press, 1980.
- [15] S. Ronchi Della Rocca. Principal type scheme and unification for intersection type discipline. *Theoretical Computer Science*, 59:181–209, 1988.
- [16] S. Ronchi Della Rocca and B. Venneri. Principal type schemes for an extended type theory. *Theoretical Computer Science*, 28:151–169, 1984.
- [17] P. Sallé. Une extension de la théorie des types en λ -calcul. In G. Ausiello and C. Böhm, editors, *Automata, Languages and Programming*, volume 62 of *Lecture Notes in Computer Science*, pages 398–410. Springer-Verlag, Berlin, 1978.
- [18] Émilie Sayag and Michel Mauny. Characterization of principal type of normal forms in intersection type system. Technical report, INRIA, 1996. To appear.
- [19] Z. Shao and A. W. Appel. Smartest recompilation. Technical Report CS-TR-395-92, Princeton University, Octobre 1992.
- [20] Steffen van Bakel. Complete restrictions of the intersection type discipline. *Theoretical Computer Science*, 102:135–163, 1992.
- [21] Steffen van Bakel. Principal type schemes for strict type assignment system. *Logic and Computation*, 3(6):643–670, 1993.

Annex

We prove here the soundness and completeness of our inference algorithm. We first notice that two non overlapped calls to *Infer* give disjoint types and disjoint constraint environments, since type variables used in each call to *Infer* are new type variables. This fact will be used in the proof of completeness of the algorithm.

Soundness

We want to prove that our type assignment algorithm is sound with respect to inference rules, that is for any normal form, there exist a typing proof relating this normal form to the type and constraints environment given by the algorithm.

Theorem 1 *Let N be a normal form, if $\text{Infer}(N) = (\tau, A)$, then $\vdash N : \tau; A$.*

Proof by structural induction on N .

- Case $N = x$.

$\text{Infer}(x) = (\alpha, \{x : [\alpha]\})$ and we can derive $\vdash x : \alpha; \{x : [\alpha]\}$.

- Case $N = \lambda x.N_1$.

Let $(\tau_1, A_1) = \text{Infer}(N_1)$, then by induction hypothesis, $\vdash N_1 : \tau_1; A_1$ and moreover,

$$\text{Infer}(\lambda x.N_1) = (A_1(x) \rightarrow \tau_1, A_1 \setminus \{x\})$$

Therefore we can prove :

$$\frac{\vdash N_1 : \tau_1; A_1}{\vdash \lambda x.N_1 : A_1(x) \rightarrow \tau_1; A_1 \setminus \{x\}}$$

- Case $N = x N_1 \dots N_n$.

Let α be a new type variable and for every $i = 1, \dots, n$, $(\tau_i, A_i) = \text{Infer}(N_i)$. Then

$$\text{Infer}(x N_1 \dots N_n) = (\alpha, \{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\} + A_1 + \dots + A_n)$$

By induction hypothesis, we have for all $i = 1, \dots, n$, $\vdash N_i : \tau_i; A_i$. Moreover, we can derive $\vdash x : [\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha; \{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\} + A_1 + \dots + A_n$.

Then if we denote $A = \{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\}$, we have the following derivation :

$$\frac{\vdash x : [\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha; A \quad \vdash N_1 : \tau_1; A_1}{\vdash x N_1 : [\tau_2] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha; A + A_1}$$

$$\vdots$$

$$\frac{\vdash x N_1 \dots N_{n-1} : [\tau_n] \rightarrow \alpha; A + A_1 + \dots + A_{n-1} \quad \vdash N_n : \tau_n; A_n}{\vdash x N_1 \dots N_n : \alpha; A + A_1 + \dots + A_n}$$

that is the result.

□

Completeness

This property establishes that our system has the principal type property for normal forms. Since for any given normal form, it shows that the inference algorithm gives a type and a constraint environment from which we can derive any possible type of this normal form by substitution.

Theorem 2 *Let N be a normal form such that $\vdash N : \tau; A$, then $\text{Infer}(N) = (\tau_p, A_p)$ and there exists a substitution S such that $S(\tau_p) = \tau$ and $S(A_p) = A$.*

Proof by structural induction on N .

- Case $N = x$.

We derive $\vdash x : \sigma; \{x : [\sigma]\}$ for some σ and we have $\text{Infer}(x) = (\alpha, \{x : [\alpha]\})$. Then in order to have the expected result, we simply define the substitution S as $[\alpha/\sigma]$.

- Case $N = \lambda x.N_1$.

We proved :

$$\frac{\vdash N_1 : \tau_1; A_1}{\vdash \lambda x.N_1 : A_1(x) \rightarrow \tau_1; A_1 \setminus \{x\}}$$

and by induction hypothesis, $\text{Infer}(N_1) = (\tau'_1, A'_1)$ and there exists a substitution S such as $S(\tau'_1) = \tau_1$ and $S(A'_1) = A_1$. From the latter equality, we deduce $S(A'_1(x)) = A_1(x)$ and $S(A'_1 \setminus \{x\}) = A_1 \setminus \{x\}$. So we have $\text{Infer}(\lambda x.N_1) = (A'_1(x) \rightarrow \tau'_1, A'_1 \setminus \{x\})$ with $S(A'_1(x) \rightarrow \tau'_1) = A_1(x) \rightarrow \tau_1$ and $S(A'_1 \setminus \{x\}) = A_1 \setminus \{x\}$.

- Case $N = x N_1 \dots N_n$.

Because of the VAR rule, the only possible derivations have the following form :

$$\frac{\vdash x : [\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha; A \quad \vdash N_1 : \tau_1; A_1}{\vdash x N_1 : [\tau_2] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha; A' + A_1}$$

$$\vdots$$

$$\frac{\vdash x N_1 \dots N_{n-1} : [\tau_n] \rightarrow \alpha; A + A_1 + \dots + A_{n-1} \quad \vdash N_n : \tau_n; A_n}{\vdash x N_1 \dots N_n : \alpha; A + A_1 + \dots + A_n}$$

By induction hypothesis, for all $i = 1, \dots, n$, $\text{Infer}(N_i) = (\tau'_i, A'_i)$ and there exist n substitutions S_1, \dots, S_n such that for all $i = 1, \dots, n$, $S_i(\tau'_i) = \tau_i$ and $S_i(A'_i) = A_i$.

Let β be a new type variable, then

$$\text{Infer}(x N_1 \dots N_n) = (\beta, \{x : [[\tau'_1] \rightarrow \dots \rightarrow [\tau'_n] \rightarrow \beta]\} + A'_1 + \dots + A'_n)$$

Moreover the domains of substitutions S_i are disjoint from each other, since they work on types and constraint environments which result from independent calls to *Infer*.

We can therefore consider the substitution $S' = S_1 + \dots + S_n + [\beta/\alpha]$. We have $S'(\beta) = \alpha$ and $S'(\{x : [[\tau'_1] \rightarrow \dots \rightarrow [\tau'_n] \rightarrow \beta]\} + A'_1 + \dots + A'_n) = \{x : [[\tau_1] \rightarrow \dots \rightarrow [\tau_n] \rightarrow \alpha]\} + A_1 + \dots + A_n$, which implies the result.

□