INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE

A new presentation of the intersection type discipline through principal typings of normal forms

Émilie Sayag, Michel Mauny

N° 2998

October 14, 1996

THÈME 2

Rapport de recherche

ISSN 0249-6399

A new presentation of the intersection type discipline through principal typings of normal forms

Émilie Sayag, Michel Mauny*

Thème 2 — Génie logiciel
et calcul symbolique

Projet Cristal

Rapport de recherche n°2998 — October 14, 1996 — 38 pages

Abstract: We introduce an intersection type system which is a restriction of the intersection type discipline. This restriction leads to a principal type property for normal forms in the classical sense, while retaining the expressivity of the classical discipline. We characterize the structure of principal types of normal forms and give an algorithm that reconstructs normal forms from types. Having shown the equivalence between principal types and normal forms, we define an expansion operation on types which allows us to recover all possible types for any normalizable λ-term. The contribution of this work is a new and simpler presentation of the intersection type discipline through a purely syntactic and completely characterized notion of principal types.

(Résumé : tsvp)

*{Emilie.Sayag,Michel.Mauny}@inria.fr

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : (33 1) 39 63 55 11 – Télécopie : (33 1) 39 63 53 30

Une nouvelle présentation des systèmes de types intersections à travers le typage principal des formes normales

Résumé : Nous introduisons un système de types intersections qui est une restriction des systèmes de types intersection classiques. Cette restriction conduit à la propriété de typage principal au sens classique pour les formes normales tout en gardant l'expressivité des systèmes classiques. Nous caractérisons complètement la structure des types principaux des formes normales et nous donnons un algorithme de reconstruction d'une forme normale à partir d'un type principal. Ayant montré l'équivalence entre notre notion de type principal et les formes normales, nous définissons une opération d'expansion qui permet, composée avec l'opération de substitution, de retrouver tous les types possibles pour un λ-terme normalisable. La contribution de ce travail est la nouveauté et la simplicité de la présentation des types intersections à travers une notion de types principaux purement syntaxique et complètement caractérisée.

1 Introduction

In the approach of untyped λ-calculus as a model of programming languages, Curry's type system is the basis of type systems of programming languages like ML [18]. Indeed, Curry's type system has the principal type property *i.e.*, for each typable λ-term there exists a type, the principal type, from which we can find all possible types for this term. This property is the basis of parametric polymorphism. Furthermore, the problem of typability in this system is decidable. However, this type system has some limitations: polymorphic abstractions are not allowed and types are not preserved under β-conversion. For instance, the λ-term $(\lambda x.x\ x)$ is not typable 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 have been proposed, the most studied of which being the intersection type discipline. Using intersection types, terms and term variables can have more than one type. This allows polymorphic abstraction, and types are invariant under β-conversion of terms [2] i.e., two λ-terms which are β-equivalent have the same type. Moreover, intersection types characterize normalizable λ-terms: a term is normalizable if and only if it is typable. Intersection type systems are therefore very expressive: this is why many authors have been interested by their theory or their usage [23, 24, 21, 10, 30].

However, the price of this expressiveness is that type assignment is only semi-decidable. Another drawback is the loss of the 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 just substitutions. In [6, 26, 35] a property which is similar to the principal type property is proved by adding two new operations on types: *expansion* and *rise* in [26] (or *lifting* in [35]). S. Ronchi della Rocca proposed a semi-algorithm for type inference in [25]. These results give important theoretical benefits, but unfortunately, they provide a good understanding neither of the structure of principal types nor of their characteristic properties. Furthermore, the semi-algorithm proposed in [25] is not practical because of its conceptual complexity. Thus, we are convinced that all interesting properties of intersection type systems have not been highlighted yet. In this paper, in order to fill this gap, we propose a new approach to intersection types. The work presented here introduces a restriction of the intersection type system presented in [1]. The type system obtained by this restriction is as expressive as the classical one [1]. Furthermore, with a restricted infernce rule for variables, it gives the existence of principal types for λ-terms in normal form, in the classical sense. Further, we completely characterize the structure of these principal types and show the equivalence between normal forms and principal types. This is, to our knowledge, the first detailed study of the structural properties of principal types for intersection type systems. These properties enable us to give a simpler definition of the expansion operation than the one proposed in [6, 25, 35], and a simpler proof of the existence of a principal type for all normalizable λ-terms.

The primary motivation for this work is to tighten the theories of intersection types and λ-calculus: the coincidence of typability and normalizability let us think that both theories can be made closer than they currently are. This work is a first step in this direction: we define a new notion of principal type, corresponding exactly to the notion of normal form in the λ-calculus. Tightening further intersection types and λ-calculus should provide a greatly simplified presentation of intersection types and therefore, a better understanding of the runtime behaviour of polymorphic programs, which itself could be used as a basis for improving the current technology of debuggers

Normal forms	N	$::=$	x	variable
		$\mid$	$\lambda x.N$	abstraction
		$\mid$	$x\ N_1\ \ldots\ N_n$	application $n \geq 1$
Types	$\rho \in \mathcal{T}$	$::=$	α	type variable
		$\mid$	$[\rho_1, \ldots, \rho_n] \rightarrow \rho$	for $n \geq 0$

Figure 1: Normal forms and types

for polymorphic languages.

The general outline of this paper is as follows: in section 2, we introduce the type system that we study. In section 3, we characterize the sets of normalizable and head normalizable λ-terms through typing. Section 4 presents an inference algorithm for normal forms and proves the correctness and completeness of this algorithm with respect to the inference rules of section 2. In section 5, we state and prove the characteristic properties of principal types for normal forms. Section 6 introduces an algorithm that reconstructs normal forms from types and characterizes the set of principal types of normal forms. In section 7, we define the operation of expansion and we give some of its properties. The main result of section 8 states the principal type property for normalizable terms and section 9 gives an overview of the related works. Finally, section 10 contains a few concluding remarks.

2 Definitions

2.1 Types

In the usual definition of intersection type systems [1], there are two type constructors $\rightarrow$ and $\wedge$ and a type constant ω: the universal type. This type constant has been introduced to deal with the invariance under β-conversion for the λK-terms (without ω, interesting results have been proved only for λI-terms [3]).

In the first definition of intersection types introduced in [3], there is only one type constructor $\rightarrow$. An intersection is written $[\rho_1, \ldots, \rho_n]$ and called a *sequence*[1]. Our definition of types (*cf.* figure 1) is close to this one. The major difference between our system and classical ones, is that we use a n-ary type constructor $[\ldots] \rightarrow \_$, with intersections occurring on the left hand side of arrows. Therefore, in our system intersections are *not* types, and our syntax for types admits $[\,] \rightarrow \rho$ as a perfectly legal type, where $[\,]$ denotes the empty intersection.

Our set of types is clearly a restriction of the classical set of intersection types since intersections occur only on the left hand side of arrows and ω does not belong to $\mathcal{T}$. Nevertheless, the empty intersection allows us to type exactly all normalizable λ-terms. Therefore, our restriction of the set of types does not reduce the class of typable λ-terms.

In any case, we know that in classical intersection type systems, the notion of principal type of a term does not exist in the classical sense (see [2] for more details). In the same way, in our intersection type system, substitutions are not sufficient to deduce all possible types for a term from a unique type. For example, in our system, as in [6], we can type the term $\lambda x.x\ (\lambda y.y)$ with

[1] Here, we prefer to call it *intersection*.

the type $[[[\alpha] \to \alpha] \to [\beta]] \to \beta$, but also with the type $[[[\alpha] \to \alpha, [\gamma] \to \gamma] \to \beta] \to \beta$. Still there is no substitution relating the former (principal type of $(\lambda x.x\ (\lambda y.y))$ in the classical theory) to the latter.

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

In figure 1, we give the grammar which defines the set of normal forms and the definition of the set of types. We assume a countably infite set TV of type variables.

2.2 Occurrences and sub-terms

The notion of *occurrence* has been often used in literature. Different refinements have been defined (see for example [13], page 33 and 34 for *positive, negative* and *final occurrences* and [25] for level of an occurrence of a type) according to the precision of informations that the author wants to obtain with this notion. In our case, we only need to distinguish the sign of occurrences and wether or not a type variable has a final occurrence.

Definition 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 $\rho = [\rho_1, \ldots, \rho_n] \to \rho'$, then the positive (respectively negative) occurrences of α in ρ are the positive (respectively negative) occurrences of α in ρ' and if $n \geq 1$, the negative occurrences of α in ρ_i for $i = 1, \ldots, n$.

Definition Let ρ be a type in $\mathcal{T}$ and α a type variable. We say that α has a *final occurrence* in ρ if one of the following cases is verified:

- $\rho = \alpha$
- $\rho = [\rho_1, \ldots, \rho_n] \to \rho'$ and α has a final occurrence in ρ'.

Remark: If a type variable has a final occurrence in 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 Let $\rho \in \mathcal{T}$, the set $L(\rho)$ of *left sub-terms* of ρ is defined by induction on the structure of ρ, in the following way:

- if $\rho = \alpha$, $L(\rho) = \emptyset$
- if $\rho = [\rho_1, \ldots, \rho_n] \to \rho'$, $L(\rho) = \{\rho_1, \ldots, \rho_n\} \cup L(\rho')$.

We also define a mapping *TypeVar* from types to sets of type variables. This function returns the set of type variables which occur in a type.

2.3 Substitutions

The operation of substitution is defined as usual: 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 modified by S. More formally:

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

We write $[\alpha/\rho]$ the substitution that maps α to ρ and leaves other types variables unchanged.

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

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

2.4 Constraint environments

We can now define *constraint environments* and their associated operations. This notion of constraint environment was introduced by Z. Shao and A. Appel in [29] where it was called *assumption environment*. Informally, a constraint environment links the free term variables of a λ-term with their type constraints. Since we study the principal type property of λ-terms, we prefer this notion of constraint environment to the notion of *basis* used in all papers about intersection type discipline. Bases give some hypothesis about the types of free term variables. Using constraint environments leads us to obtain the minimal type constraints for free term variables. S. van Bakel in [35], must distinguish between basis, *used basis*, and *minimal basis*. Since we are only interested in the equivalent of van Bakel's minimal basis, it is enough to have one definition.

Definition A *constraint environment* A, is a mapping from the set $\mathcal{V}$ of term variables to the multi-sets of types. We use multi-sets rather than simple sets to keep track of the different occurrences of the same type.

The empty multi-set is written $[\,]$ and the multi-set of types $\rho_1, \ldots, \rho_n$ is written $[\rho_1, \ldots, \rho_n]$. These notations allow us to write the type $A(x) \to \rho$ for any constraint environment A, any term variable x and any type ρ, consistently with the syntax of types.

In order to handle easily these constraint environments, we introduce several operations giving informations on constraint environments or transforming them.

Definition Let A be a constraint environment. We define the *domain* of A, written $Dom(A)$ as:

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

Let A be a constraint environment such that $Dom(A) = \{x_1, \ldots, x_n\}$ and $A(x_i) = [\rho_{i1}, \ldots, \rho_{ip_i}]$, for all $i \in \{1, \ldots, n\}$. We use the following notation for A:

$$\{x_1 : [\rho_{11}, \ldots, \rho_{1p_1}], \ldots, x_n : [\rho_{n1}, \ldots, \rho_{np_n}]\}$$

The two following operations allow, respectively, to restrict and extend the domain of a constraint environment.

$$\vdash_{s\omega} x : \rho; \{x : [\rho]\} \qquad (\textsc{Var})$$

$$\frac{\vdash_{s\omega} e_1 : \rho_1; A_1}{\vdash_{s\omega} \lambda x.e_1 : A_1(x) \to \rho_1; A_1 \setminus \{x\}} \qquad (\textsc{Abs})$$

$$\frac{\vdash_{s\omega} e_1 : [\rho_2^1, \ldots, \rho_2^n] \to \rho_1; A_1 \qquad \vdash_{s\omega} e_2 : \rho_2^1; A_2^1 \quad \ldots \quad \vdash_{s\omega} e_2 : \rho_2^n; A_2^n}{\vdash_{s\omega} e_1\ e_2 : \rho_1; A_1 + A_2^1 + \ldots + A_2^n} \quad (n \geq 0) \qquad (\textsc{App})$$

$$\vdash_{s\omega_p} x : [\rho_1] \to \cdots \to [\rho_n] \to \alpha; \{x : [[\rho_1] \to \cdots \to [\rho_n] \to \alpha]\} \quad (n \geq 0) \quad (\textsc{Var}_p)$$

$$\frac{\vdash_{s\omega_p} e_1 : \rho_1; A_1}{\vdash_{s\omega_p} \lambda x.e_1 : A_1(x) \to \rho_1; A_1 \setminus \{x\}} \qquad (\textsc{Abs})$$

$$\frac{\vdash_{s\omega_p} e_1 : [\rho_2^1, \ldots, \rho_2^n] \to \rho_1; A_1 \qquad \vdash_{s\omega_p} e_2 : \rho_2^1; A_2^1 \quad \ldots \quad \vdash_{s\omega_p} e_2 : \rho_2^n; A_2^n}{\vdash_{s\omega_p} e_1\ e_2 : \rho_1; A_1 + A_2^1 + \ldots + A_2^n} \quad (n \geq 0) \qquad (\textsc{App})$$

Figure 2: Inference rules

Definition 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 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 the union of multi-sets.

Remark: We adopt the usual conventions for omitting parentheses in types and terms and some other syntactic conventions: we use metavariables $x, y, \ldots$ to denote term variables, $\alpha, \beta, \gamma, \ldots$ for type variables, and $A, A_1, \ldots$ for constraint environments.

2.5 Type assignments

The type assignment relations $\vdash_{sw}$ and $\vdash_{s\omega p}$, relating λ-terms, types and constraint environments, are defined in figure 2.

In fact, $\vdash_{s\omega_p}$ is a restriction of the type assignment $\vdash_{sw}$ since the only difference between $\vdash_{s\omega_p}$ and $\vdash_{sw}$ is in the typing rules for variables: while the typing rule for variables (Var) allows us to type a term variable without restriction on its structure, our (Var$_p$) rule gives a strong restriction on the structure of the type which occurs in that rule. We shall see in the following section, that this restriction enables us to find all types of normal forms using just substitutions. We also notice that in the rules for applications, if $n = 0$ then there is only one premisse in that inference rule and the argument of the application does not interfere in the derivation.

3 Expressiveness of type system

In this section, we are only interested in the type assignment $\vdash_{s\omega}$. We prove that though we have restricted the set of types in our system, we can characterize normalizable λ-terms and head normalizable λ-terms according to the structure of the assigned type as we can do in the classical intersection type systems. A number of proofs of these results for intersection type discipline have been written, the first one can be found in [1] and we follow the method of [13].

First, we show that two β-equivalent λ-terms have the same type and the same constraint environment. Then we prove that every term in normal form or in head normal form is typable. Finally, we give a translation from our type system to the classical intersection type system which enable us to use the normalization theorems proved for this system [1, 13].

3.1 Stability under β-conversion

To prove this property, we show independantly the stability under β-expansion and under β-reduction.

3.1.1 β-expansion

The following lemma and its proof are technical, but they explain how the typing derivation of a term $e[x/f]$ is transformed to obtain a typing derivation of e.

Lemma 1 *Let x be a term variable, e and f two λ-terms such that x has at least one free occurrence in e. If $\vdash_{s\omega} e[x/f] : \rho; A$ then there exist an interger $n \geq 1$, n types: $\rho_1, \ldots, \rho_n$ and n constraint environments: $A_1, \ldots, A_n$, such that:*

- $\forall i \in \{1, \ldots, n\}$, $\vdash_{s\omega} f : \rho_i; A_i$
- $A = A' + A_1 + \cdots + A_n$ *for one* A' *such that* $A'(x) = [\,]$
- $\vdash_{s\omega} e : \rho; A' + \{x : [\rho_1, \ldots, \rho_n]\}$

Proof by recurrence on the length of the typing derivation of e.

• If $e = x$ then $e[x/f] = f$ and we have the derivation $\vdash_{s\omega} f : \rho; A$, since by hypothesis $\vdash_{s\omega} e[x/f] : \rho; A$. Moreover, according to the inference rule (VAR), we have $\vdash_{s\omega} x : \rho; \{x : [\rho]\}$.

Thus, the type ρ and the constraint environment A are such that:

- $\vdash_{s\omega} f : \rho; A$
- $A = A' + A$ with $A' = \{\,\}$ and $A'(x) = [\,]$
- $\vdash_{s\omega} e : \rho; \{\,\} + \{x : [\rho]\}$

- If $e = \lambda y.e_c$ then $e[x/f] = \lambda y.e_c[x/f]$ and we have:

$$\frac{\vdash_{s\omega} e_c[x/f] : \rho_c; A_c}{\vdash_{s\omega} \lambda y.e_c[x/f] : A_c(y) \to \rho_c; A_c \setminus \{y\}}$$

with $\rho = A_c(y) \to \rho_c$ and $A = A_c \setminus \{y\}$.

Since x has at least one free occurrence in e, x has at least one free occurrence in e_c and we have $y \neq x$. So by the recurrence hypothesis on the typing derivation of e_c, there exist n types $\rho_1, \ldots, \rho_n$ and n constraint environments $A_1, \ldots, A_n$ such that:

  - $\forall i \in \{1, \ldots, n\}$, $\vdash_{s\omega} f : \rho_i; A_i$
  - $A_c = A'_c + A_1 + \cdots + A_n$ for one A'_c such that $A'_c(x) = [\,]$
  - $\vdash_{s\omega} e_c : \rho_c; A'_c + \{x : [\rho_1, \ldots, \rho_n]\}$

By the inference rule (VAR), $\vdash_{s\omega} \lambda y.e_c : A'_c(y) \to \rho_c; A'_c \setminus \{y\} + \{x : [\rho_1, \ldots, \rho_n]\}$

We must show that:

  - $\forall i \in \{1, \ldots, n\}$, $\vdash_{s\omega} f : \rho_i; A_i$
  - $A = A' + A_1 + \cdots + A_n$ with $A' = A'_c \setminus \{y\}$ and $(A'_c \setminus \{y\})(x) = [\,]$
  - $\vdash_{s\omega} \lambda y.e_c : \rho; A' + \{x : [\rho_1, \ldots, \rho_n]\}$.

The first case is exactly the recurrence hypothesis given previously.

For the second case, since $A = A_c \setminus \{y\}$ and $A_c = A'_c + A_1 + \cdots + A_n$, we must show that $A_c \setminus \{y\} = A'_c \setminus \{y\} + A_1 + \cdots + A_n$ *i.e.*, $A_i \setminus \{y\} = A_i$. But, we can suppose without lost of generality, that y has no free occurrence in f since by definition of the substitution of a variable in a λ-term, the free variables of f are always free in $e[x/f]$. Thus the free occurrences of y in e_c are the same as in $e_c[x/f]$ and for all $i \in \{1, \ldots, n\}$, $A_i(y) = [\,]$. So $A = A'_c \setminus \{y\} + A_1 + \cdots + A_n$. We still have to show that $(A'_c \setminus \{y\})(x) = [\,]$, but it is clear since $(A'_c \setminus \{y\})(x) \subset A'_c(x) = [\,]$.

For the last case, we have $\vdash_{s\omega} \lambda y.e_c : A'_c(y) \to \rho_c; A'_c \setminus \{y\} + \{x : [\rho_1, \ldots, \rho_n]\}$. Since $\rho = A_c(y) \to \rho_c$, we must show that $A_c(y) = A'_c(y)$. But we have seen that for all $i \in \{1, \ldots, n\}$, $A_i(y) = [\,]$ and thus $A_c(y) = A'_c(y) + A_1(y) + \cdots + A_n(y) = A'_c(y)$.

- If $e = e_f\ e_a$ then $e[x/f] = e_f[x/f]\ e_a[x/f]$ and we have derived:

$$\frac{\vdash_{s\omega} e_f[x/f] : [\rho_a^1, \ldots, \rho_a^p] \to \rho; A_f \qquad \vdash_{s\omega} e_a[x/f] : \rho_a^1; A_a^1 \quad \ldots \quad \vdash_{s\omega} e_a[x/f] : \rho_a^p; A_a^p}{\vdash_{s\omega} (e_f\ e_a)[x/f] : \rho; A_f + A_a^1 + \cdots + A_a^p}$$

with $A = A_f + A_a^1 + \cdots + A_a^p$.

Several cases arise according to the repartition of the free occurrences of x in $e_f\ e_a$:

  - Let suppose that x has at least one free occurrence in e_f *and* in e_a.

    By the recurrence hypothesis on the typing derivation of e_f, there exist n types $\rho_1, \ldots \rho_n$ and n constraint environments $A_1, \ldots, A_n$ such that:

- $\forall i \in \{1,\ldots,n\}$, $\vdash_{s\omega} f : \rho_i; A_i$
- $A_f = A'_f + A_1 + \cdots + A_n$ for one A'_f such that $A'_f(x) = [\,]$
- $\vdash_{s\omega} e_f : [\rho_a^1,\ldots,\rho_a^p] \to \rho; A'_f + \{x : [\rho_1,\ldots,\rho_n]\}$

In the same way, we can apply the recurrence hypothesis to each typing derivation $\vdash_{s\omega} e_a : \rho_a^i; A_a^i$. Thus for all $i \in \{1,\ldots,p\}$, there exist n_i types $\rho_i^1,\ldots,\rho_i^{n_i}$ and n_i constraint environments $A_i^1,\ldots,A_i^{n_i}$ such that:

- $\forall j \in \{1,\ldots,n_i\}$, $\vdash_{s\omega} f : \rho_i^j; A_i^j$
- $A_a^i = B_2^i + A_i^1 + \cdots + A_i^{n_i}$ for one B_2^i such that $B_2^i(x) = [\,]$
- $\vdash_{s\omega} e_a : \rho_a^i; B_2^i + \{x : [\rho_i^1,\ldots,\rho_i^{n_i}]\}$

We deduce the following derivation:

$$\frac{\begin{array}{c}\vdash_{s\omega} e_f : [\rho_a^1,\ldots,\rho_a^p] \to \rho; A'_f + \{x : [\rho_1,\ldots,\rho_n]\} \\ \vdash_{s\omega} e_a : \rho_a^1; B_2^1 + \{x : [\rho_1^1,\ldots,\rho_1^{n_1}]\} \quad \ldots \quad \vdash_{s\omega} e_a : \rho_a^p; B_2^p + \{x : [\rho_p^1,\ldots,\rho_p^{n_p}]\}\end{array}}{\vdash_{s\omega} e_f\ e_a : \rho; A'_f + B_2^1 + \cdots + B_2^p + \{x : [\rho_1,\ldots,\rho_n \rho_1^1,\ldots,\rho_1^{n_1},\ldots,\rho_p^1,\ldots,\rho_p^{n_p}]\}}$$

Since $A'_f(x) = [\,]$ and $B_2^i(x) = [\,]$ for all $i \in \{1,\ldots,p\}$, $(A'_f + B_2^1 + \cdots + B_2^p)(x) = [\,]$

Moreover, we have $A = A_f + A_a^1 + \cdots + A_a^p$, $A_f = A'_f + A_1 + \cdots + A_n$, and for all $i \in \{1,\ldots,p\}$, $A_a^i = B_2^i + A_i^1 + \cdots + A_i^{n_i}$, so

$$A = A'_f + A_1 + \cdots + A_n + B_2^1 + \cdots + B_2^p + A_1^1 + \cdots + A_1^{n_1} + \cdots + A_p^1 + \cdots + A_p^{n_p}$$

Let $N = n + n_1 + \cdots + n_p$.

The N types: $\rho_1,\ldots,\rho_n,\rho_1^1,\ldots,\rho_1^{n_1},\ldots,\rho_p^1,\ldots,\rho_p^{n_p}$, and the N constraint environments: $A_1,\ldots,A_n,A_1^1,\ldots,A_1^{n_1},\ldots,A_p^1,\ldots,A_p^{n_p}$, are such that:

- $\vdash_{s\omega} f : \rho_j; A_j$ for $j = 1,\ldots,n$ and $\vdash_{s\omega} f : \rho_i^j; A_i^j$ for $i = 1,\ldots,p$ and $j = 1,\ldots,n_i$
- $A = A'_f + B_2^1 + \cdots + B_2^p + A_1 + \cdots + A_n + A_1^1 + \cdots + A_1^{n_1} + \cdots + A_p^1 + \cdots + A_p^{n_p}$ avec $(A'_f + B_2^1 + \cdots + B_2^p)(x) = [\,]$
- $\vdash_{s\omega} e : \rho; A'_f + B_2^1 + \cdots + B_2^p + \{x : [\rho_1,\ldots,\rho_n,\rho_1^1,\ldots,\rho_1^{n_1},\ldots,\rho_p^1,\ldots,\rho_p^{n_p}]\}$

- Let suppose that x has no free occurrence in e_f, then x has at least one free occurrence in e_a. Like in the previous case we can apply the recurrence hypothesis to each typing derivation of e_a. We notice that if x has no free occurrence in e_f then $e_f[x/f] = e_f$ and $\vdash_{s\omega} e_f : [\rho_a^1,\ldots,\rho_a^p] \to \rho; A_f$.

  The result can be deduced from the recurrence hypothesis and from the equality $A_f(x) = [\,]$.

- The last case where x has no free occurrence in e_a is similar to the previous case. It is enough to apply the recurrence hypothesis to e_f and notice that $e_a[x/f] = e_a$ and $A_a^i(x) = [\,]$, for all $i = 1,\ldots,p$.

$\square$

Now, we prove that a redex has the same type and the same constraint environment as the corresponding β-reduced term.

Lemma 2 *Let x a term variable, e and f two λ-terms. If $\vdash_{s\omega} e[x/f] : \rho; A$ then $\vdash_{s\omega} (\lambda x.e)f : \rho; A$.*

Proof by case on the number of occurrences of x in e.

• If x has no free occurrence in e then $e[x/f] = e$ and by hypothesis, $\vdash_{s\omega} e : \rho; A$. Since in our type system the domain of the constraint environment A only holds the free variables of e, we have $A(x) = [\,]$ and the following derivation:

$$\cfrac{\cfrac{\vdash_{s\omega} e : \rho; A}{\vdash_{s\omega} \lambda x.e : [\,] \to \rho; A}}{\vdash_{s\omega} (\lambda x.e)f : \rho; A}$$

It proves that $(\lambda x.e)f$ has the same type and the same constraint environment as $e[x/f]$.

• Otherwise x has at least one free occurrence in e. By hypothesis, $\vdash_{s\omega} e[x/f] : \rho; A$. So by the lemma 1, there exist n types: $\rho_1, \dots, \rho_n$ and n constraint environments: $A_1, \dots, A_n$, such that:

- $\forall i \in \{1, \dots, n\}$, $\vdash_{s\omega} f : \rho_i; A_i$
- $A = A' + A_1 + \cdots + A_n$ for one A' such that $A'(x) = [\,]$
- $\vdash_{s\omega} e : \rho; A' + \{x : [\rho_1, \dots, \rho_n]\}$

But $(A' + \{x : [\rho_1, \dots, \rho_n]\})(x) = [\rho_1, \dots, \rho_n]$ and $(A' + \{x : [\rho_1, \dots, \rho_n]\}) \setminus \{x\} = A'$ since $A'(x) = [\,]$. We deduce the following derivation:

$$\cfrac{\cfrac{\vdash_{s\omega} e : \rho; A' + \{x : [\rho_1, \dots, \rho_n]\}}{\vdash_{s\omega} \lambda x.e : [\rho_1, \dots, \rho_n] \to \rho; A'} \qquad \vdash_{s\omega} f : \rho_1; A_1 \quad \dots \quad \vdash_{s\omega} f : \rho_n; A_n}{\vdash_{s\omega} (\lambda x.e)f : \rho; A' + A_1 + \cdots + A_n}$$

Thus $(\lambda x.e)f$ and $e[x/f]$ has the same type and the same constraint environment.$\square$

The previous lemma states a result for a redex and its contraction, we generalize this result for two β-equivalent terms in the following theorem.

Theorem 1 *Let e and e' two λ-terms such that e can be β-reduced in e'. If $\vdash_{s\omega} e' : \rho; A$ then $\vdash_{s\omega} e : \rho; A$.*

Proof by structural induction on e.// We can suppose, without lost of generality, that e' differs of e only by the contraction of one redex.

• If $e = x$ then this case is impossible since we can not β-reduce e in e'.

• If $e = \lambda x.e_1$ then $e' = \lambda x.e_1'$ with e_1 which is reduced, by the contraction of one redex, in e_1'. Thus if we have:

$$\cfrac{\vdash_{s\omega} e_1' : \rho'; A'}{\vdash_{s\omega} \lambda x.e_1' : A'(x) \to \rho'; A' \setminus \{x\}}$$

with $\rho = A'(x) \to \rho'$ and $A = A' \setminus \{x\}$ then by the induction hypothesis we have $\vdash_{s\omega} e_1 : \rho'; A'$. So by the inference rule for abstraction, we have $\vdash_{s\omega} \lambda x.e_1 : \rho; A$.

- If $e = e_1\ e_2$ then three cases are possible:
  - $e' = e'_1\ e_2$ with e_1 which is reduced to e'_1 by the contraction of one redex and we have:

$$\frac{\vdash_{s\omega} e'_1 : [\rho_2^1, \ldots, \rho_2^n] \to \rho; A_1 \qquad \vdash_{s\omega} e_2 : \rho_2^i; A_2^i}{\vdash_{s\omega} e'_1\ e_2 : \rho; A_1 + A_2^1 + \cdots + A_2^n}$$

    with $A = A_1 + A_2^1 + \cdots + A_2^n$.

    Then by induction, $\vdash_{s\omega} e_1 : [\rho_2^1, \ldots, \rho_2^n] \to \rho; A_1$, and by application of the inference rule (App), $\vdash_{s\omega} e_1\ e_2 : \rho; A$.
  - or $e' = e_1\ e'_2$ with e_2 which is reduced to e'_2 and we have:

$$\frac{\vdash_{s\omega} e_1 : [\rho_2^1, \ldots, \rho_2^n] \to \rho; A_1 \qquad \vdash_{s\omega} e'_2 : \rho_2^1; A_2^1 \ldots \vdash_{s\omega} e'_2 : \rho_2^n; A_2^n}{\vdash_{s\omega} e_1\ e'_2 : \rho; A_1 + A_2^1 + \cdots + A_2^n}$$

    with $A = A_1 + A_2^1 + \cdots + A_2^n$.

    Then by induction, for each derivation $\vdash_{s\omega} e'_2 : \rho_2^i; A_2^i$, we have $\vdash_{s\omega} e_2 : \rho_2^i; A_2^i$ and by the inference rule (App) $\vdash_{s\omega} e_1\ e_2 : \rho; A$.
  - otherwise $e_1 = \lambda x.f$ and $e' = f[x/e_2]$ with $\vdash_{s\omega} e' : \rho; A$. By the lemma 2, $\vdash_{s\omega} (\lambda x.f)e_2 : \rho; A$.

□

3.1.2 β-reduction

To prove the stability under β-reduction, we follow the same method as for β-expansion. We show successively how the typing derivation of a term e is transformed to obtain a typing derivation of $e[x/f]$, that the typing assignments of a redex and its contraction are the same, and that the typing assignments of two β-equivalent terms are the same.

Lemma 3 *Let x be a term variable, e and f λ-terms such that x has at least one free occurrence in e. If $\vdash_{s\omega} e : \rho; A + \{x : [\rho_1, \ldots, \rho_n]\}$ and $\vdash_{s\omega} f : \rho_i; A_i$ for $i = 1, \ldots, n$ then $\vdash_{s\omega} e[x/f] : \rho; A + A_1 + \cdots + A_n$.*

Proof by structural induction on e.

- If $e = x$ then $e[x/f] = f$ and $\vdash_{s\omega} x : \rho; \{x : [\rho]\}$ with $A = [\,]$. Since by hypothesis, $\vdash_{s\omega} f : \rho; A'$ for one A', we have $\vdash_{s\omega} e[x/f] : \rho; A + A''$.
- If $e = \lambda y.e_1$ then $e[x/f] = \lambda y.e_1[x/f]$ and we have:

$$\frac{\vdash_{s\omega} e_1 : \rho'; A + \{y : [\rho_1, \ldots, \rho_m]\} + \{x : [\rho_1, \ldots, \rho_n]\}}{\vdash_{s\omega} \lambda y.e_1 : [\rho_1, \ldots, \rho_m] \to \rho'; A + \{x : [\rho_1, \ldots, \rho_n]\}}$$

with $\rho = [\rho_1, \ldots, \rho_m] \rightarrow \rho'$.

Moreover x and y are distinct variables since x has at least one free occurrence in e. We can suppose, without lost of generality, that y has no free occurrence in f. So by the induction hypothesis, $\vdash_{s\omega} e_1[x/f] : \rho'; A + \{y : [\rho_1, \ldots, \rho_m]\} + A_1 + \cdots + A_n$ and from the inference rule (ABS), we deduce that $\vdash_{s\omega} \lambda y.e_1[x/f] : \rho; A + A_1 + \cdots + A_n$.

• If $e = e_1\ e_2$ then as in the lemma 1, we must distinguish several cases according to the repartition of free occurrences of x in e_1 and e_2. In the three cases, it is enough to use the induction hypothesis and the inference rule (APP) to show that $\vdash_{s\omega} e_1\ e_2[x/f] : \rho; A + A_1 + \cdots + A_n$.□

Lemma 4 *Let x be a term variable, e and f two λ-terms. If $\vdash_{s\omega} (\lambda x.e)f : \rho; A$ then $\vdash_{s\omega} e[x/f] : \rho; A$.*

Proof

• If x has no free occurrence in e then $e[x/f] = e$ and we have the following derivation:

$$\dfrac{\dfrac{\vdash_{s\omega} e : \rho; A \quad A(x) = [\,]}{\vdash_{s\omega} \lambda x.e : [\,] \rightarrow \rho; A}}{\vdash_{s\omega} (\lambda x.e)f : \rho; A}$$

Thus we have $\vdash_{s\omega} e[x/f] : \rho; A$.

• Otherwise x has at least one free occurrence in e. Since we have supposed that $\vdash_{s\omega} (\lambda x.e)f : \rho; A$, we have the following derivation:

$$\dfrac{\dfrac{\vdash_{s\omega} e : \rho; A' \quad A'(x) = [\rho_1, \ldots, \rho_n]}{\vdash_{s\omega} \lambda x.e : [\rho_1, \ldots, \rho_n] \rightarrow \rho; A' \setminus \{x\}} \quad \vdash_{s\omega} f : \rho_i; A_i}{\vdash_{s\omega} (\lambda x.e)f : \rho; A}$$

with $A = A' \setminus \{x\} + A_1 + \cdots + A_n$.

According to the lemma 3, we have $\vdash_{s\omega} e[x/f] : \rho; A' \setminus \{x\} + A_1 + \cdots + A_n$.□

Theorem 2 *Let e and e' be two λ-terms such that e' is obtained by β-reduction from e. If $\vdash_{s\omega} e : \rho; A$ then $\vdash_{s\omega} e' : \rho; A$.*

Proof by structural induction on e.
We can suppose that e and e' only differs in the contraction of one redex.

• If $e = x$ then this case is impossible since e can not be reduced in e'.

• If $e = \lambda x.e_1$ then $e' = \lambda x.e_1'$ where e_1' is obtained from e_1 by β-reduction and we have derived:

$$\dfrac{\vdash_{s\omega} e_1 : \rho_1; A_1}{\vdash_{s\omega} \lambda x.e_1 : A_1(x) \rightarrow \rho_1; A_1 \setminus \{x\}}$$

with $\rho = A_1(x) \rightarrow \rho_1$ et $A = A_1 \setminus \{x\}$. Thus by the induction hypothesis we have $\vdash_{s\omega} e_1' : \rho_1; A_1$. We deduce $\vdash_{s\omega} \lambda x.e_1' : \rho; A$.

• If $e = e_1\ e_2$ then as for the theorem 1 three cases are possible:

- ∘ $e' = e_1\ e_2'$ or $e' = e_1'\ e_2$ then we use the induction hypothesis and the inference rule (APP).
- ∘ otherwise $e = (\lambda x.f)e_2$ then $e' = f[x/e_2]$ and we use the previous lemma.

□

3.1.3 β-conversion

Theorem 3 *If $e =_\beta e'$ then $\vdash_{s\omega} e : \rho; A$ if and only if $\vdash_{s\omega} e' : \rho; A$.*

Proof by the theorems 1 and 2.□

3.2 Characterization of normalizable λ-terms

To prove the normalization theorem and the head normalization theorem, we use the corresponding results proveed for the classical intersection type system. But we do not recall the definitions of the set of types and of the typing assignment in this system. The reader can see [2] for more details.

3.2.1 Translation between $\mathcal{T}$ and $\mathcal{T}_{\wedge\omega}$

The sets of types in our type system and in the classical intersection type system do not use the same syntax, so to make the link between the both type systems, we need to define a translation function which defines a type of $\mathcal{T}_{\wedge\omega}$ for all type of $\mathcal{T}$.

Definition We define a function $(\_)_{\wedge\omega}$ translating a type ρ of $\mathcal{T}$ in a type $(\rho)_{\wedge\omega}$ of $\mathcal{T}_{\wedge\omega}$ defined by induction on the structure of ρ:

- if $\rho = \alpha$ then $(\rho)_{\wedge\omega} = \alpha$
- if $\rho = [\,] \rightarrow \rho'$ then $(\rho)_{\wedge\omega} = \omega \rightarrow (\rho)_{\wedge\omega}$
- if $\rho = [\rho_1, \ldots, \rho_n] \rightarrow \rho'$ with $n \neq 0$ then $(\rho)_{\wedge\omega} = (\rho_1)_{\wedge\omega} \wedge \ldots \wedge (\rho_n)_{\wedge\omega} \rightarrow (\rho')_{\wedge\omega}$

This function is an injection but not a bijection: there exist no type ρ of $\mathcal{T}$ such that $(\rho)_{\wedge\omega} = \omega$.

We extend this tranlsation to constraint environments in the following way:

- if $a - \{\,\}$ then $(A)_{\wedge\omega} = \emptyset$
- if $A = \{x : [\rho]\}$ then $(A)_{\wedge\omega} = \{x : (\rho)_{\wedge\omega}\}$
- if $A = A_1 + A_2$ then $(A)_{\wedge\omega} = (A_1)_{\wedge\omega} + (A_2)_{\wedge\omega}$

where $+$ is the union of two bases such that if $x : \tau_1 \in (A_1)_{\wedge\omega}$ and if $x : \tau_2 \in (A_2)_{\wedge\omega}$ then $x : \tau_1 \wedge \tau_2 \in (A_1)_{\wedge\omega} + (A_2)_{\wedge\omega}$.

Theorem 4 *Let e be a λ-term. If $\vdash_{s\omega} e : \rho; A$ then $e \vdash_{\wedge\omega} (\rho)_{\wedge\omega} : (A)_{\wedge\omega}$.*

Proof by structural induction on e.□

3.2.2 Normalizable λ-terms

The characterization of normalizable λ-terms depends on the structure of assigned types. Thus an important notion is the notion of *proper* types. Before defining this notion, we need to give the definition of the level of an occurrence in a type.

Definition We define the *level* of an occurrence of $[\,]$ in a type ρ by induction on the structure of ρ:

- if $\rho = [\,] \to \rho'$ then the level of $[\,]$ in ρ is 1
- if $\rho = [\rho_1, \ldots, \rho_n] \to \rho'$ then the level of an occurrence of $[\,]$ in ρ is the level of the corresponding occurrence of $[\,]$ in ρ' or the level of the corresponding occurrence of $[\,]$ in one of the ρ_i $+1$.

Definition We say that ρ is a proper type if $[\,]$ has no occurrence in ρ or if $[\,]$ has only occurrences at odd levels.

We can extend this definition to constraint environments. A constraint environment A is a proper constraint environment if $[\,]$ has no occurrence in A or if $[\,]$ only occurs in a type of A at even levels.

Lemma 5 *Let ρ be a proper type and A a proper constraint environment. Their translation in $\mathcal{T}_{\wedge\omega}$, $(\rho)_{\wedge\omega}$ and $(A)_{\wedge\omega}$ are respectively a proper type and a proper basis.*

Proof

It is enough to notice that the translation does not change the structure of types and that each occurrence of $[\,]$ becomes an occurrence of ω. Thus the levels of occurrences of ω in $(\rho)_{\wedge\omega}$ and $(A)_{\wedge\omega}$ are the one of $[\,]$ in ρ and A, so they have the same parity.$\square$

Theorem 5 *Let e be a normalizable λ-term such that $\vdash_{s\omega} e : \rho; A$ where ρ and A are proper. Then e is normalizable.*

Proof

According to the theorem 4, we have $(A)_{\wedge\omega} \vdash_{\omega} e : (\rho)_{\wedge\omega}$ and by the lemma 5, $(A)_{\wedge\omega}$ and $(\rho)_{\wedge\omega}$ are respectively a proper basis and a proper type. By the normalization theorem of the system $\mathcal{D}_{\omega}$ (cf. [7]), e is normalizable.$\square$

Lemma 6 *Let e be a λ-term in normal form. There exist a proper type ρ and a proper constraint environment A such that $\vdash_{s\omega} e : \rho; A$.*

Proof by induction on the structure of a normal form.

• If $e = x$ then let ρ be a type which has no occurrence of $[\,]$. We can derive $\vdash_{s\omega} x : \rho; A$ with $A = \{x : [\rho]\}$. Since $[\,]$ has no occurrence neither in ρ nor in A, they are both proper.

• If $e = \lambda x.e_1$ where e_1 is in normal form then by the induction hypothesis, there exist A_1 and ρ_1 proper such that $\vdash_{s\omega} e_1 : \rho_1; A_1$. Thus there exist $A = A_1 \setminus \{x\}$ and $\rho = A_1(x) \to \rho_1$ such that $\vdash_{s\omega} e : \rho; A$.

We must prove that ρ and A are proper. The occurrences of $[\,]$ in $A_1(x)$ are at even level since A_1 is proper. This occurrences are at odd level in $A_1(x) \to \rho_1$. Moreover the level of the occurrences of $[\,]$ in ρ_1 do not change in $A_1(x) \to \rho_1$. Thus $A_1(x) \to \rho_1$ and $A_1 \setminus \{x\}$ are proper.

• If $e = x\ e_1\ \ldots\ e_n$ where for all $i \in \{1, \ldots, n\}$, e_i is in normal form, then by the induction hypothesis, for each e_i there exist proper A_i and ρ_i such that $\vdash_{s\omega} e_i : \rho_i; A_i$. We have

$\vdash_{s\omega} x\ e_1\ \dots\ e_n : \alpha; \{x : [[\rho_1] \to \cdots \to [\rho_n] \to \alpha]\} + A_1 + \cdots + A_n$ where α is a type variable. The occurrences of $[\,]$ in ρ_i are at odd level since ρ_i is proper. These occurrences are at even level in $[\rho_1] \to \cdots \to [\rho_n] \to \alpha$. Thus $\{x : [[\rho_1] \to \cdots \to [\rho_n] \to \alpha]\} + A_1 + \cdots + A_n$ is proper.□

Theorem 6 *Let e be a normalizable λ-term. There exist a proper type ρ and a proper constraint environment A such that $\vdash_{s\omega} e : \rho; A$.*

Proof

Since e is a normalizable λ-term, there exists a λ-term e' in normal form such that $e =_\beta e'$. By the lemma 6, there exist a proper constraint environment A and a proper type ρ such that $\vdash_{s\omega} e' : \rho; A$. And by the theorem 3, we have $\vdash_{s\omega} e : \rho; A$.□

Corollary *A λ-term is normalizable if and only if it is typable with a proper type and a proper constraint environment.*

Proof by the theorems 5 and 6.□

3.2.3 Head normalizable λ-terms

Theorem 7 *Let e be a λ-term such that $\vdash_{s\omega} e : \rho; A$. Then e has a head normal form.*

Proof

By the theorem 4, every term typable in our system is typable in $\mathcal{D}_\omega$ by a type distinct of ω sincce $[\,]$ is not a type. But in $\mathcal{D}_\omega$, every term typable with a type distinct of ω has a head normal form.□

Lemma 7 *Let e be a λ-term in head normal form. There exist a type ρ and a constraint environment A such that $\vdash_{s\omega} e : \rho; A$.*

Proof

A term in head normal form can be written as $\lambda x_1.\dots.\lambda x_n.x\ e_1\ \dots\ e_m$ where m and n are two integers, x a term variable and $e_1, \dots, e_m$ some λ-terms. It is enough to show that $x\ e_1\ \dots\ e_m$ is typable. Let ρ be a type, we have $\vdash_{s\omega} x\ e_1\ \dots\ e_m : \rho; \{x : [\underbrace{[\,] \to \cdots \to [\,]}_{m} \to \rho]\}$□

Theorem 8 *Let e be a λ-term which has a head normal form. Then there exist a type ρ and a constraint environment A such that $\vdash_{s\omega} e : \rho; A$.*

Proof

There exist a term e' in head normal form such that $e =_\beta$. By the lemma 7, there exist a constraint environment A and a type ρ such that $\vdash_{s\omega} e' : \rho; A$. Thus by the theorem 3, we have $\vdash_{s\omega} e : \rho; A$.□

Corollary *A λ-term has a head normal form if and only if it is typable.*

Proof by the theorems 7 and 8.□

$Infer(N) =$

- **Case** $N = x$

  let α be a new type variable

  return $(\alpha, \{x : [\alpha]\})$

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

  let $(\rho_1, A_1) = Infer(N_1)$

  return $(A_1(x) \to \rho_1, A_1 \setminus \{x\})$

- **Case** $N = x\ N_1\ \ldots\ N_n$

  let $(\rho_1, A_1) = Infer(N_1)$

  $\vdots$

  $(\rho_n, A_n) = Infer(N_n)$

  α be a new type variable

  return $(\alpha, \{x : [[\rho_1] \to \cdots \to [\rho_n] \to \alpha]\} + A_1 + \cdots + A_n)$

Figure 3: Type inference algorithm

4 Type inference

In this section we present a type assignment algorithm for normal forms and we prove its soundness and completeness with respect to the inference rules defining the $\vdash_{s\omega_p}$ relation (cf. figure 2). Our inference algorithm is not really original, since we can find similar definitions in [6, 26, 25, 35, 17] where the principal type property in the intersection type discipline is studied.

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, which is developed in the further sections.

The type inference algorithm is presented in figure 3. For clarity, we do not formalize the notion of *new type variable* (see [16] for a precise definition). This algorithm is defined modulo the name of type variables, since we do not fix the choice of the new type variables. Moreover, one can notice that there is no notion of bound type variables.

Remark: 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.

The inference algorithm is sound and complete in the sense of [19], as expressed by theorems 9 and 10.

Theorem 9 *Let N be a normal form, if $Infer(N) = (\rho, A)$ then $\vdash_{s\omega_p} N : \rho; A$.*

Proof by structural induction on N.

- Case $N = x$.

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

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

Let $(\rho_1, A_1) = Infer(N_1)$ then by induction, $\vdash_{s\omega_p} N_1 : \rho_1; A_1$ and $Infer(\lambda x.N_1) = (A_1(x) \to$

$\rho_1, A_1 \setminus \{x\})$. We can therefore prove:

$$\frac{\vdash_{s\omega_p} N_1 : \rho_1; A_1}{\vdash_{s\omega_p} \lambda x.N_1 : A_1(x) \to \rho_1; A_1 \setminus \{x\}}$$

• Case $N = x\ N_1\ \ldots\ N_n$.

Let α be a new type variable and for every $i = 1, \ldots, n$, $(\rho_i, A_i) = \mathit{Infer}(N_i)$. $\mathit{Infer}(x\ N_1\ \ldots\ N_n) = (\alpha, \{x : [[\rho_1] \to \cdots \to [\rho_n] \to \alpha]\} + A_1 + \cdots + A_n)$
By induction, we have $\vdash_{s\omega_p} N_i : \rho_i; A_i$ for all $i = 1, \ldots, n$. If we write $\rho = [\rho_1] \to \ldots \to [\rho_n] \to \alpha$, we can derive $\vdash_{s\omega_p} x : \rho; \{x : [\rho]\}$ according to the typing rule ($\text{V}\small{\text{AR}}_p$).

Then we have the following derivation:

$$\cfrac{\cfrac{\vdash_{s\omega_p} x : \rho; A \qquad \vdash_{s\omega_p} N_1 : \rho_1; A_1}{\vdash_{s\omega_p} x\ N_1 : \rho'; A + A_1}}{\begin{array}{c}\ddots \\ \ddots \\ \cfrac{\vdash_{s\omega_p} x\ N_1 \ldots N_{n-1} : [\rho_n] \to \alpha; A' \qquad \vdash_{s\omega_p} N_n : \rho_n; A_n}{\vdash_{s\omega_p} x\ N_1 \ldots N_n : \alpha; A + A_1 + \cdots + A_n}\end{array}}$$

where $\rho' = [\rho_2] \to \ldots \to [\rho_n] \to \alpha$, $A = \{x : [\rho]\}$ and $A' = A + A_1 + \cdots + A_{n-1}$.$\square$

Theorem 10 *Let N be a normal form such that $\vdash_{s\omega_p} N : \rho; A$ then $\mathit{Infer}(N) = (\rho_p, A_p)$ and there exists a substitution S such that $S(\rho_p) = \rho$ and $S(A_p) = A$.*

Proof by structural induction on N.

• Case $N = x$.

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

• Case $N = \lambda x.N_1$.

We have derived:

$$\frac{\vdash_{s\omega_p} N_1 : \rho_1; A_1}{\vdash_{s\omega_p} \lambda x.N_1 : A_1(x) \to \rho_1; A_1 \setminus \{x\}}$$

and by induction, $\mathit{Infer}(N_1) = (\rho'_1, A'_1)$ and there exists a substitution S such that $S(\rho'_1) = \rho_1$ and $S(A'_1) = A_1$. From the latter equality, we deduce that $S(A'_1(x)) = A_1(x)$ and $S(A'_1 \setminus \{x\}) = A_1 \setminus \{x\}$. We have therefore:

$$\mathit{Infer}(\lambda x.N_1) = (A'_1(x) \to \rho'_1, A'_1 \setminus \{x\})$$

with $S(A'_1(x) \to \rho'_1) = A_1(x) \to \rho_1$ and $S(A'_1 \setminus \{x\}) = A_1 \setminus \{x\}$.

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

Because of the ($\textsc{Var}_p$) rule, the only possible derivations have the following form:

$$\dfrac{\dfrac{\dfrac{\vdash_{s\omega_p} x : \rho; A \qquad \vdash_{s\omega_p} N_1 : \rho_1; A_1}{\vdash_{s\omega_p} x\ N_1 : \rho'; A + A_1}}{\ddots}\qquad\ddots}{}$$

$$\dfrac{\vdash_{s\omega_p} x\ N_1 \ldots N_{n-1} : [\rho_n] \to \alpha; A' \qquad \vdash_{s\omega_p} N_n : \rho_n; A_n}{\vdash_{s\omega_p} x\ N_1 \ldots N_n : \alpha; A + A_1 + \cdots + A_n}$$

where $\rho = [\rho_1] \to \ldots \to [\rho_n] \to \alpha$, $\rho' = [\rho_2] \to \ldots \to [\rho_n] \to \alpha$, $A = \{x : [\rho]\}$ and $A' = A + A_1 + \cdots + A_{n-1}$. For all $i \in \{1, \ldots, n\}$, we deduce from the induction hypothesis, that $\mathit{Infer}(N_i) = (\rho'_i, A'_i)$ and there exist n substitutions $S_1, \ldots, S_n$ such that for all $i \in \{1, \ldots, n\}$, $S_i(\rho'_i) = \rho_i$ and $S_i(A'_i) = A_i$.

Let β be a new type variable. $\mathit{Infer}(x\ N_1\ \ldots\ N_n) = (\beta, \{x : [[\rho'_1] \to \cdots \to [\rho'_n] \to \beta]\} + A'_1 + \cdots + A'_n)$

Moreover, according to the previous remark about non overlapped calls to *Infer*, the domains of substitutions S_i are disjoint from each other. We can therefore consider the substitution $S' = S_1 + \cdots + S_n + [\beta/\alpha]$. We have $S'(\beta) = \alpha$ and $S'(\{x : [[\rho'_1] \to \cdots \to [\rho'_n] \to \beta]\} + A'_1 + \cdots + A'_n) = \{x : [[\rho_1] \to \cdots \to [\rho_n] \to \alpha]\} + A_1 + \cdots + A_n$, which implies the result.□

Theorem 10 states that our system has the principal type property for normal forms. Since for any given normal form, the inference algorithm gives a type and a constraint environment from which we can derive any possible type of this normal form by substitution.

As an example, type inference of $\lambda x.\lambda y.x(y\ x)$ produces the type $([\alpha, [\beta] \to \gamma] \to [[\alpha] \to \beta] \to \gamma)$. This type, as well as any other type returned by the algorithm *Infer*, possesses a few peculiarities, the most obvious being that *each type variable has exactly two occurrences: one positive and one negative.* In section 4, we characterize pairs computed by the algorithm. In section 5, we will see that, given such a type (e.g. $[\alpha, [\beta] \to \gamma] \to [[\alpha] \to \beta] \to \gamma$), one can reconstruct a λ-term in normal form, by (roughly speaking):

- following the arrows from left to right: that gives the outermost abstractions of the result ($\lambda u.\lambda v.\_$ in our example)

- following type variables, starting from the rightmost one, in order to built the body of the normal form ($\lambda u.\lambda v.u(v\ u)$ in this case).

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

5 Characterization of principal pairs

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

$$
\begin{array}{rcll}
\sigma \in \mathcal{T}_{E_p} & ::= & \alpha & \\
 & | & [\tau] \to \sigma & \text{with } \mathit{TypeVar}(\tau) \cap \mathit{TypeVar}(\sigma) = \emptyset \\
 & & & \\
\tau \in \mathcal{T}_p & ::= & \alpha & \\
 & | & [\sigma_1, \ldots, \sigma_n] \to \tau & \text{with } n \geq 0 \\
 & & & \\
A \in \mathcal{E}_p & ::= & \{\,\} & \\
 & | & \{x : [\sigma]\} & \\
 & | & A_1 + A_2 &
\end{array}
$$

Figure 4: Principal types and constraint environments

5.1 Principal pairs belong to $\mathcal{T}_p \times \mathcal{E}_p$

In figure 4, we define $\mathcal{T}_{E_p}$ and $\mathcal{T}_p$ two sub-sets of $\mathcal{T}$ and the set $\mathcal{E}_p$ of principal constraint environments. The intersection between $\mathcal{T}_{E_p}$ and $\mathcal{T}_p$ is not empty, since each type variable belongs at the same time to $\mathcal{T}_{E_p}$ and to $\mathcal{T}_p$. Moreover, if ρ_1 and ρ belong to $\mathcal{T}_{E_p} \cap \mathcal{T}_p$, then $[\rho_1] \to \rho \in \mathcal{T}_{E_p} \cap \mathcal{T}_p$ since $[\rho_1] \to \rho$ has the shape of $[\tau] \to \sigma$ with $\tau = \rho_1$ and $\sigma = \rho$ but also the shape of $[\sigma_1, \ldots, \sigma_n] \to \tau$ with $n = 1$, $\sigma_1 = \rho_1$ and $\tau = \rho$.

In the following, we write $\sigma, \sigma', \sigma_1, \ldots$ for elements of $\mathcal{T}_{E_p}$, and $\tau, \tau', \tau_1, \ldots$ for elements of $\mathcal{T}_p$. When we do not want to specify whether a type belongs to $\mathcal{T}_{E_p}$ or to $\mathcal{T}_p$, we write it ρ, ρ', ρ_1, ... as for types in $\mathcal{T}_\wedge$.

The following lemma states that principal pairs belong to $\mathcal{T}_p \times \mathcal{E}_p$.

Lemma 8 *Let N be a normal form, if $\mathit{Infer}(N) = (\tau, A)$ then $\tau \in \mathcal{T}_p$ and $A \in \mathcal{E}_p$.*

Proof by induction on the structure of N.

- Case $N = x$.

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

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

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

If we write $A_1(x) = [\sigma_1, \ldots, \sigma_n]$ with $n \geq 0$ then $\mathit{Infer}(\lambda x.N_1) = ([\sigma_1, \ldots, \sigma_n] \to \tau_1, A_1 \setminus \{x\})$. Since $A_1 \in \mathcal{E}_p$, if $n \geq 1$, $\sigma_i \in \mathcal{T}_{E_p}$ for all $i \in \{1, \ldots, n\}$, and since $\tau_1 \in \mathcal{T}_p$, we have $[\sigma_1, \ldots, \sigma_n] \to \tau_1 \in \mathcal{T}_p$ for $n \geq 0$ and $A_1 \setminus \{x\} \in \mathcal{E}_p$.

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

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

$$\mathit{Infer}(x\ N_1\ \ldots\ N_n) = (\alpha, \{x : [[\tau_1] \to \cdots \to [\tau_n] \to \alpha]\} + A_1 + \cdots + A_n)$$

Now $\alpha \in \mathcal{T}_{E_p}$ and $\alpha \in \mathcal{T}_p$, moreover by induction, for all $i \in \{1, \ldots, n\}$, $\tau_i \in \mathcal{T}_p$ and $A_i \in \mathcal{E}_p$. Therefore we must prove that $[\tau_1] \to \cdots \to [\tau_n] \to \alpha \in \mathcal{T}_{E_p}$, which is immediate.$\square$

If we regard *Infer* as a function from the set of normal forms to the set of pairs (τ, A), the set of pairs (τ, A) such that there exists a normal form N which verifies $\mathit{Infer}(N) = (\tau, A)$ is written

$Range(Infer)$. So we can restate the previous lemma in the following way:

$$Range(Infer) \subset \mathcal{T}_p \times \mathcal{E}_p$$

5.2 A-types

In the following, we always consider pairs consisting of a type and a constraint environment. To handle types and type constraints consistently, we introduce *A-types*, using a double arrow in order to write type constraints negatively. Formally, we define the set of A-types by the following grammar:

$$\begin{array}{rcll} T & ::= & [\sigma_1, \ldots, \sigma_n] \Rightarrow \tau & \text{with } n \geq 0 \\ & | & [\sigma_1, \ldots, \sigma_n] \Rightarrow & \text{with } n \geq 1 \end{array}$$

where $[\sigma_1, \ldots, \sigma_n]$ is a multi-set of elements of $\mathcal{T}_{E_p}$. The function *TypeVar*, returning the set of type variables of a type, is naturally extended to A-types.

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

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

- if $A = \{\,\}$, then $\overline{A} = [\,]$
- if $A = \{x : [\sigma]\}$, then $\overline{A} = [\sigma]$
- 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 an A-type.

Definition We say that T' is *held* in T, if

- $T = [\sigma_1, \ldots, \sigma_n] \Rightarrow \tau$
- and T' has one of the following forms:
  - $[\sigma_{i_1}, \ldots, \sigma_{i_p}] \Rightarrow \tau$
  - $[\sigma_{i_1}, \ldots, \sigma_{i_p}] \Rightarrow$

where $[\sigma_{i_1}, \ldots, \sigma_{i_p}]$ is a sub-multi-set (non empty in the second case) of $[\sigma_1, \ldots, \sigma_n]$. Moreover, if T' is held in T and distinct from T, we say that T' is *strictly held* in T.

Extending the notion of left sub-terms of a type we define for each A-type T, the set $L_0(T)$ of left sub-terms of T by induction on the structure of T:

- if $T = [\sigma_1, \ldots, \sigma_n] \Rightarrow \tau$ then $L_0(T) = \{\sigma_1, \ldots, \sigma_n\} \cup L_0(\tau)$
- if $T = [\sigma_1, \ldots, \sigma_n] \Rightarrow$ then $L_0(T) = \{\sigma_1, \ldots, \sigma_n\}$.

We also extend to A-types the notion of sign of occurrences. Let T be an 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 $T = [\sigma_1, \ldots, \sigma_n] \Rightarrow \tau$, 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) occurrences of α in σ_i for $i = 1, \ldots, n$

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

In order to characterize the structure of principal pairs, we define three structural properties on pairs. The first of them expresses that the inference algorithm always introduces two occurrences of a new type variable with different signs. The second one is less significant but is used in the reconstruction algorithm. The last one is used to express that the inference algorithm *Infer* produces the minimal constraint environment.

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

Definition Let $T = [\sigma_1, \ldots, \sigma_n] \Rightarrow \tau$ be an A-type. T is *finally closed* if the variable α in the final occurrence of τ is also in the final occurrence of a type which is element of $L_0(T)$.

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

5.3 Properties of principal typing

The following definition gives a short way to talk about the three previous properties simultaneously.

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

- T is closed

- T is finally closed

- T is minimally closed.

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

Lemma 9 *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 $\overline{A} \Rightarrow \tau = [\alpha] \Rightarrow \alpha$. Thus we have:

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

◦ α is the final occurrence of α and $L_0(\overline{A} \Rightarrow \tau) = \{\alpha\}$. Therefore $[\alpha] \Rightarrow \alpha$ is finally closed.

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

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

Let $(\tau_1, A_1) = \mathit{Infer}(N_1)$. We deduce from the induction hypothesis, that $\overline{A_1}{\Rightarrow}\tau_1$ is complete.

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

  - $\mathit{TypeVar}(\overline{A_1 \setminus \{x\}}{\Rightarrow}[\sigma^1, \ldots, \sigma^n] \to \tau_1) = \mathit{TypeVar}(\overline{A_1}{\Rightarrow}\tau_1)$. Furthermore, for $i = 1, \ldots, n$, the occurrences of type variables in σ^i keep the same sign in $\overline{A_1 \setminus \{x\}}{\Rightarrow}[\sigma^1, \ldots, \sigma^n] \to \tau_1$ as in $\overline{A_1}{\Rightarrow}\tau_1$. Hence the A-type $\overline{A_1 \setminus \{x\}}{\Rightarrow}[\sigma^1, \ldots, \sigma^n]$ $\to \tau_1$ is closed since $\overline{A_1}{\Rightarrow}\tau_1$ is closed by induction.

  - On the other hand, the final occurrences of the two types $[\sigma^1, \ldots, \sigma^n] \to \tau_1$ and τ_1 are the same and $L_0(\overline{A_1 \setminus \{x\}}{\Rightarrow}[\sigma^1, \ldots, \sigma^n] \to \tau_1) = L_0(\overline{A_1}{\Rightarrow}\tau_1)$. Therefore, by induction, the A-type $\overline{A_1 \setminus \{x\}}{\Rightarrow}[\sigma^1, \ldots, \sigma^n] \to \tau_1$ is finally closed.

  - Let T be an A-type strictly held in the A-type $\overline{A_1 \setminus \{x\}}{\Rightarrow}[\sigma^1, \ldots, \sigma^n] \to \tau_1$, then there exists T' strictly held in $\overline{A_1}{\Rightarrow}\tau_1$ such that $\mathit{TypeVar}(T) = \mathit{TypeVar}(T')$. Furthermore, since the occurrences of type variables in $\overline{A_1}{\Rightarrow}\tau_1$ keep the same sign as in $\overline{A_1 \setminus \{x\}}{\Rightarrow}[\sigma^1, \ldots, \sigma^n] \to \tau_1$, for all closed T strictly held in $\overline{A_1 \setminus \{x\}}{\Rightarrow}[\sigma^1, \ldots, \sigma^n] \to \tau_1$, there exists a closed T' strictly held in $\overline{A_1}{\Rightarrow}\tau_1$. Hence by induction, there does not exist any closed A-type strictly held in $\overline{A_1 \setminus \{x\}}{\Rightarrow}[\sigma^1, \ldots, \sigma^n] \to \tau_1$. Therefore, $\overline{A_1 \setminus \{x\}}{\Rightarrow}[\sigma^1, \ldots, \sigma^n] \to \tau_1$ is minimally closed.

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

Let α be a new type variable and for $i = 1, \ldots, n$, $(\tau_i, A_i) = \mathit{Infer}(N_i)$. Then $\mathit{Infer}(x\ N_1\ \ldots\ N_n) = (\alpha, \{x : [[\tau_1] \to \cdots \to [\tau_n] \to \alpha]\} + A_1 + \cdots + A_n)$. We write $A = \{x : [[\tau_1] \to \cdots \to [\tau_n] \to \alpha]\} + A_1 + \cdots + A_n$.

By induction for all $i \in \{1, \ldots, n\}$, $\overline{A_i}{\Rightarrow}\tau_i$ is complete.

  - $\mathit{TypeVar}(\overline{A}{\Rightarrow}\alpha) = \bigcup_{i=1}^n \mathit{TypeVar}(\overline{A_i}{\Rightarrow}\tau_i \cup \{\alpha\})$. Now, if $\beta \in \bigcup_{i=1}^n \mathit{TypeVar}(\overline{A_i}{\Rightarrow}\tau_i)$ for all $i \in \{1, \ldots, n\}$, 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 $[\tau_1] \to \cdots \to [\tau_n] \to \alpha$ which belongs to $L_0(\overline{A}{\Rightarrow}\alpha)$. So $\overline{A}{\Rightarrow}\alpha$ is finally closed.

  - Let T be an A-type held in $\overline{A}{\Rightarrow}\alpha$. For $i = 1, \ldots, n$, the sets $\mathit{TypeVar}(\overline{A_i}{\Rightarrow}\tau_i)$ have no common type variables, since they result from disjoint calls to *Infer*. Furthermore, for all $i \in \{1, \ldots, n\}$, $\overline{A_i}{\Rightarrow}\tau_i$ 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 $[\tau_1] \to \cdots \to [\tau_n] \to \alpha$. But if $[\tau_1] \to \cdots \to [\tau_n] \to \alpha$ 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 pairs of its sub-terms from its principal pair. Since it is easier to study an A-type than a pair (τ, A), we formalize our intuition on A-types.

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

- $T = [\alpha] \Rightarrow \alpha$.
- $T = [\sigma_1, \dots, \sigma_n] \Rightarrow \alpha$ and $\exists i \in \{1, \dots, n\}$ such that σ_i has the shape $[\tau_1] \to \cdots \to [\tau_p] \to \alpha$ with $p > 0$ 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.
- $T = [\sigma_1, \dots, \sigma_n] \Rightarrow [\sigma_{n+1}, \dots, \sigma_{n+p}] \to \tau'$ and $[\sigma_1, \dots, \sigma_n, \sigma_{n+1}, \dots, \sigma_{n+p}] \Rightarrow \tau'$ is principal.

One may notice a similarity between the three cases above and the three cases defining normal forms. This will be made formal in section 5.

Remark: A complete A-type is not necessarily principal. For example, the A-type defined by $T = [[[\gamma] \to \beta] \to \alpha; [\gamma] \to [[\beta] \to \delta] \to \delta] \Rightarrow \alpha$ is complete, but not principal, since $[[\gamma] \to [[\beta] \to \delta] \to \delta] \Rightarrow [\gamma] \to \beta$ is not finally closed, therefore not principal.

Lemma 10 *Let N be a normal form. If $Infer(N) = (\tau, A)$ then $\overline{A} \Rightarrow \tau$ is principal.*

Proof by structural induction on N.
Thanks to the previous lemma, in any case $\overline{A} \Rightarrow \tau$ is complete.

- Case $N = x$.

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

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

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

If we write $A_1(x) = [\sigma_1, \dots, \sigma_n]$ then we have $Infer(N) = ([\sigma_1, \dots, \sigma_n] \to \tau_1, A_1 \backslash \{x\})$. By definition of a principal A-type, in order to prove that the A-type $\overline{A_1 \setminus \{x\}} \Rightarrow [\sigma_1, \dots, \sigma_n] \to \tau_1$ 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 $i = 1, \dots, n$, $(\tau_i, A_i) = Infer(N_i)$. By induction each $\overline{A_i} \Rightarrow \tau_i$ is principal and in particular complete. Furthermore, we have $Infer(N) = (\alpha, \{x : [[\tau_1] \to \cdots \to [\tau_n] \to \alpha]\} + A_1 + \cdots + A_n)$. This is an instance of the second case defining principal A-types and we must prove that there exists a partition of the multi-set $\overline{A_1 + \cdots + A_n}$ into n sub-multi-sets $E_1, \dots, E_n$ such that each $E_i \Rightarrow \tau_i$ is principal. $(\overline{A_i})_{i=1,\dots,n}$ is a partition of $\overline{A_1 + \cdots + A_n}$ such that each $\overline{A_i} \Rightarrow \tau_i$ is principal by the induction hypothesis.
Thus $\overline{\{x : [[\tau_1] \to \cdots \to [\tau_n] \to \alpha]\} + A_1 + \cdots + A_n} \Rightarrow \alpha$ is principal.□

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

$$Range(Infer) \subset \mathcal{P}$$

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

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

  fail

- **Case** (α, A)

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

  if $m = 1$ and $\tau^1 = [\tau_1] \to \cdots \to [\tau_n] \to \alpha$

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

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

  $\vdots$

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

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

  return $(x\ N_1\ \ldots\ N_n, A \setminus A' + A_1' + \cdots + A_n')$

  else fail

  else fail

- **Case** $([\sigma^1, \ldots, \sigma^n] \to \tau', A)$

  let x be a new term variable

  $A' = A + \{x : [\sigma^1, \ldots, \sigma^n]\}$

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

  if $A''(x) = [\,]$

  then return $(\lambda x.N, A'')$

  else fail

Figure 5: Reconstruction algorithm

6 Reconstruction of normal forms

In order to characterize the principal pairs of normal forms, we give an algorithm $\mathcal{R}$ which, given a type and a constraint environment, constructs a normal form typable with this type and this constraint environment.

Let α be a type variable and A be a constraint environment. We write $F(\alpha, A)$ the set of types which belong to A and have α as final occurrence. More precisely: $F(\alpha, A) = \{(\tau, x)/\tau \in A(x)$ and α is the final occurrence of $\tau\}$

We define the reconstruction algorithm $\mathcal{R}$ in figure 5. Since we make no hypotheses on τ or A in the definition of $\mathcal{R}$, we must justify that $\mathcal{R}$ is always defined for a principal pair. So we prove in the following remarks that the application of $\mathcal{R}$ to a principal pair never leads to a case of failure.

Remarks:

• $\{\,\}\Rightarrow\alpha$ is not closed and so the first case of $\mathcal{R}$ is impossible with a pair (τ, A) belonging to $\mathcal{P}$.

• If $F(\alpha, A) = \{(\tau^1, x_1), \ldots, (\tau^m, x_m)\}$ with $m \neq 1$, in the case (α, A) then $\overline{A}\Rightarrow\alpha$ is not finally closed. Therefore this case is impossible with a pair $(\tau, A) \in \mathcal{P}$.

• If $\mathcal{R}(\tau, A) = (N, B)$ then $B \subseteq A$. The proof is immediate by induction on the number of calls to $\mathcal{R}$.

Lemma 11 *If $(\tau, A) \in \mathcal{P}$ then $\mathcal{R}(\tau, A) = (N, A')$ is well defined and $A' = \{\,\}$.*

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

• Case (α, A).

Let $\{([\tau_1] \to \cdots \to [\tau_n] \to \alpha, x)\} = F(\alpha, A)$. We have therefore $A = \{x : [[\tau_1] \to \cdots \to [\tau_n] \to \alpha]\} + B$ for some constraint environment B.

Moreover, if $F(\alpha, A) = \{([\tau_1] \to \cdots \to [\tau_n] \to \alpha, x)\}$ with $n = 0$, then we have $\mathcal{R}(\alpha, A) = (x, A \backslash \{x : [\alpha]\})$. Now for an element of $\mathcal{P}$, $\overline{A}\Rightarrow\alpha$ is minimally closed and $A = \{x : [\alpha]\}$. So $\mathcal{R}(\alpha, A) = (x, \{\,\})$.

We remark that if $n = 0$ then $\mathcal{R}(\alpha, A) = (x, A \setminus \{x : [\alpha]\})$. Now for an element of $\mathcal{P}$, $\overline{A}\Rightarrow\alpha$ is minimally closed and $A = \{x : [\alpha]\}$. So $\mathcal{R}(\alpha, A) = (x, \{\,\})$ is well defined.

Now we 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 $(E_i)_{i=1,\ldots,n}$ 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 $A_1, \ldots, A_n$ of B such that $\overline{A_i}\Rightarrow\tau_i$ is principal for all $i \in \{1, \ldots, n\}$. 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 $i \in \{1, \ldots, n\}$, by definition of $\mathcal{R}$.

Since each $\overline{A_i}\Rightarrow\tau_i$ is principal, we can apply the induction hypothesis to $\mathcal{R}(\tau_i, A_i)$ for $i = 1, \ldots, n$. Thus for $i = 1, \ldots, n$, $\mathcal{R}(\tau_i, A_i) = (N_i, A'_i)$ is well defined and $A'_i = \{\,\}$.

From this, we deduce that $\mathcal{R}(\alpha, A) = (x\ N_1\ \ldots\ N_n, A')$ is well defined with $A' = A \setminus (\{x : [[\tau_1] \to$

$\cdots \to [\tau_n] \to \alpha]\} + A_1 + \cdots + A_n) + \{\,\}$

Now if we write $A'' = \{x : [[\tau_1] \to \cdots \to [\tau_n] \to \alpha]\} + A_1 + \cdots + A_n$, we have $A'' = A$.

By definition of A_i and since $\{([\tau_1] \to \cdots \to [\tau_n] \to \alpha, x)\} = F(\alpha, A)$ 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 variable occurrences in $\overline{A_i}{\Rightarrow}\tau_i$ does not change in $\overline{A''}{\Rightarrow}\alpha$ and because α has exactly one positive occurrence and one negative occurrence in $\overline{A''}{\Rightarrow}\alpha$. But $\overline{A''}{\Rightarrow}\alpha$ is held in $\overline{A}{\Rightarrow}\alpha$. Now by hypothesis, $\overline{A}{\Rightarrow}\alpha$ is minimally closed, so $\overline{A''}{\Rightarrow}\alpha$ can't be strictly held in $\overline{A}{\Rightarrow}\alpha$ and we have: $A = \{x : [[\tau_1] \to \cdots \to [\tau_n] \to \alpha]\} + A_1 + \cdots + A_n$. Therefore $A' = A \setminus A'' = \{\,\}$.

- Case $([\sigma_1, \ldots, \sigma_n] \to \tau', A)$.

Let $A' = \{x : [\sigma_1, \ldots, \sigma_n]\} + A$. Since $\overline{A}{\Rightarrow}[\sigma_1, \ldots, \sigma_n] \to \tau'$ is principal, $\overline{A'}{\Rightarrow}\tau'$ is also principal. Thus by induction, $\mathcal{R}(\tau', A') = (N, A'')$ is well defined and $A'' = \{\,\}$. Now $\{\,\}(x) = [\,]$, so $\mathcal{R}([\sigma_1, \ldots, \sigma_n] \to \tau', A) = (\lambda x.N, A'')$ is well defined by definition of $\mathcal{R}$, and $A'' = \{\,\}$.$\Box$

From now on, for each $(\tau, A) \in \mathcal{P}$, we can consider the result of $\mathcal{R}(\tau, A)$ without verifying its existence. Moreover, in the following, if the pair (τ, A) belongs to $\mathcal{P}$, we write $\mathcal{R}(\tau, A) = N$ instead of $\mathcal{R}(\tau, A) = (N, \{\,\})$.

Lemma 12 *Let $(\tau, A) \in \mathcal{P}$ and $N = \mathcal{R}(\tau, A)$ then Infer$(N) = (\tau, A)$.*

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

- Case (α, A).

Let $\{([\tau_1] \to \cdots \to [\tau_n] \to \alpha, x)\} = F(\alpha, A)$. By definition of $\mathcal{R}$ and by lemma 11, we have $\mathcal{R}(\alpha, A) = x\ N_1 \ldots N_n$. Moreover, as we did in the proof of lemma 11, one can show that there exist $(A_i)_{i=1,\ldots,n}$ n sub-environments of A such that $\overline{A_i}{\Rightarrow}\tau_i$ is principal for $i = 1, \ldots, n$ and $A = \{x : [[\tau_1] \to \cdots \to [\tau_n] \to \alpha]\} + A_1 + \cdots + A_n$. So we can apply the induction hypothesis to each call of $\mathcal{R}$ with (τ_i, A_i). It follows that if we write $\mathcal{R}(\tau_i, A_i) = N_i$ then *Infer*$(N_i) = (\tau_i, A_i)$ for all $i \in \{1, \ldots, n\}$.

By definition of *Infer*, we have *Infer*$(x\ N_1\ \ldots\ N_n) = (\alpha, \{x : [[\tau_1] \to \cdots \to [\tau_n] \to \alpha]\} + A_1 + \cdots + A_n)$. Since $A = \{x : [[\tau_1] \to \cdots \to [\tau_n] \to \alpha]\} + A_1 + \cdots + A_n$, we finally have *Infer*$(x\ N_1\ \ldots\ N_n) = (\alpha, A)$.

- Case $([\sigma_1, \ldots, \sigma_n] \to \tau', A)$.

Let $A' = \{x : [\sigma_1, \ldots, \sigma_n]\} + A$ where x is a new term variable.

Since $\overline{A}{\Rightarrow}[\sigma_1, \ldots, \sigma_n] \to \tau'$ is principal, $\overline{A'}{\Rightarrow}\tau'$ is also principal. Thus, we can apply the induction hypothesis, if $\mathcal{R}(\tau', A') = N$ then *Infer*$(N) = (\tau', A')$. Now $A'(x) = [\sigma_1, \ldots, \sigma_n]$ because x is a term variable which does not belong to the domain of A. We deduce from this that *Infer*$(\lambda x.N) = ([\sigma_1, \ldots, \sigma_n] \to \tau', A' \setminus \{x\})$. Since $A' \setminus \{x\} = A$, we can conclude.$\Box$

The previous lemma gives the opposite inclusion of lemma 10. We can now completely characterize the set of principal pairs.

Theorem 11 *Range*(*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

$$
\begin{array}{rcl}
\nu \in \mathcal{T}_{E_g} & ::= & \alpha \\
& | & [\mu_1, \ldots, \mu_n] \to \nu
\end{array}
$$

with $n > 0$, $\forall i \in \{1, \ldots, n\}$, $\mu_i \in \mathcal{T}_g$, $\mathit{TypeVar}(\mu_i) \cap \mathit{TypeVar}(\nu) = \emptyset$ and $\forall j \in \{1, \ldots, n\}$ such that $j \neq i$, $\mathit{TypeVar}(\mu_i) \cap \mathit{TypeVar}(\mu_j) = \emptyset$

$$
\begin{array}{rcl}
\mu \in \mathcal{T}_g & ::= & \alpha \\
& | & [\nu_1, \ldots, \nu_n] \to \mu
\end{array}
$$

with $n \geq 0$ and $\forall i \in \{1, \ldots, n\}$, $\nu_i \in \mathcal{T}_{E_g}$

$$
\begin{array}{rcl}
A \in \mathcal{E}_g & ::= & \{\} \\
& | & \{x : [\nu]\} \\
& | & A_1 + A_2
\end{array}
$$

Figure 6: Ground types and constraint environments

The lemma 10 states $\mathit{Range}(\mathit{Infer}) \subset \mathcal{P}$. Thus, we must just prove that for any pair $(\tau, A) \in \mathcal{P}$, there exists a normal form N such that $\mathit{Infer}(N) = (\tau, A)$.

Let (τ, A) be an element of $\mathcal{P}$. By lemma 11, $\mathcal{R}(\tau, A)$ exists and is a normal form written N and by lemma 12, $\mathit{Infer}(N) = (\tau, A)$. So $\mathcal{P} \subset \mathit{Range}(\mathit{Infer})$ and we can conclude.□

The theorem 11 states the equality $\mathit{Range}(\mathit{Infer}) = \mathcal{P}$

7 Ground pairs

We now consider a type assignment relation with a more general inference rule for variables. Our work rejoins here S. van Bakel's article [35]. In his paper, S. van Bakel defines an intersection type system close to the one introduced in [1] with the same partial order relation on types. But before considering intersections as types, he defines a sub-set of *strict types* where intersections occur only on the left hand side of arrows. This set of strict types is equivalent to the set of types presented here. He studies the existence of principal types for this system. He was induced to define several sub-sets of the set of pairs of a type and a basis, ordered by inclusion. The smallest is the *set of principal pairs* which corresponds to set $\mathcal{P}$ in our work. Van Bakel's *set of ground pairs* is equivalent to the set of ground pairs that we define in this section. It is the sub-set of pairs closed under expansion. Because of the partial order relation, van Bakel uses an extra sub-set: the *set of primitive pairs*, closed under *lifting*. (This operation is necessary to take into account the inference rule that deals with the partial order relation.) Finally, his set of pairs is closed under substitution.

Here we only distinguish the set of principal pairs and ground pairs. We obtain the same final result as S. van Bakel: *all normalizable λ-terms have a principal type*, but our definition of expansion and our proofs are much simpler. We also describe in detail the structure of ground pairs.

7.1 B-types

In figure 6 we give mutually recursive definitions of $\mathcal{T}_{E_g}$, $\mathcal{T}_g$ and $\mathcal{E}_g$, extending $\mathcal{T}_p$, $\mathcal{T}_p$ and $\mathcal{E}_p$ respectively. $\mathcal{T}_g$ is the set of ground types and $\mathcal{E}_g$ is the set of ground constraint environments.

From now on, metavariables ν and μ denote elements of $\mathcal{T}_{E_g}$ and $\mathcal{T}_g$ respectively.

In section 4, we defined the sign of an occurrence, the final occurrence and left sub-terms for principal types. We can easily extend these notions to ground types. Since these definitions do not present any difficulty, we do not give them formally.

In order to link types and constraint environments and to write negatively type constraints, we defined A-types from principal types and principal constraint environments. With the same motivations, we define B-types from ground types and ground constraint environments, in the following way:

$$\begin{array}{rcll} U & ::= & [\nu_1,\dots,\nu_n]\Rightarrow\mu & \text{with } n\geq 0 \\ & | & [\nu_1,\dots,\nu_n]\Rightarrow & \text{with } n\geq 1 \end{array}$$

Since the definition of B-types is an obvious extension of the definition of A-types, we deduce without difficulty, the definitions of *closed*, *minimally closed*, *finally closed* and *complete* B-types, from the corresponding definitions on A-types.

Definition We say that U is a *ground B-type* if U is complete and if it is one of the following forms:

- $U = [\rho]\Rightarrow\rho$ with $\rho \in \mathcal{T}_g \cap \mathcal{T}_{E_g}$.

- $U = [\nu_1,\dots,\nu_n]\Rightarrow\alpha$ and $\exists i \in \{1,\dots,n\}$ such that ν_i has the following shape

  $$[\mu_1^1,\dots,\mu_1^{n_1}] \to \cdots \to [\mu_p^1,\dots,\mu_p^{n_p}] \to \alpha$$

  with $p > 0$, and $\exists (E_j^k)_{j=1\dots p, k=1\dots n_j}$, a partition of the multi-set $[\nu_1,\dots,\nu_{i-1},\nu_{i+1},\dots,\nu_n]$ such that each $E_j^k\Rightarrow\mu_j^k$ is a ground B-type.

- $U = [\nu_1,\dots,\nu_n]\Rightarrow[\nu_{n+1},\dots,\nu_{n+p}] \to \mu'$ with $[\nu_1,\dots,\nu_n,\nu_{n+1},\dots,\nu_{n+p}]\Rightarrow\mu'$ a ground B-type.

Remark: The partition $(E_j^k)_{j=1\dots p,k=1\dots n_j}$ is unique. Since U is closed, each type variable has only two occurrences in U and we have not the choice of the definition of each E_j^k.

In the same way as we have defined the set $\mathcal{P}$ of principal pairs, we define the set $\mathcal{G}$ of *ground pairs*, by $\mathcal{G} = \{(\mu, A)/\mu \in \mathcal{T}_g,\ A \in \mathcal{E}_g \text{ and } \overline{A}\Rightarrow\mu \text{ is a ground B-type}\}$.

We can prove, by induction on the structure of A-types, the following inclusion:

$$\mathcal{P} \subset \mathcal{G}$$

7.2 Expansions

In order to simplify the definition of expansion, we need to describe several further notions and prove some properties about the structure of ground B-types. In fact, expansion is a complex operation on pairs. As S. van Bakel explains in [35], the expansion of a sub-term ρ of a type ρ' replaces the occurrences of ρ in ρ' by a number of copies of that sub-term. To be applied an expansion must therefore specify the type to be expanded and the number of necessary copies.

Intuitively, expansion corresponds to the duplication of the sub-derivation of the argument e_2 in the use of the inference rule (App) for a term $e_1\ e_2$. So it is not enough to duplicate one type: we must also copy all the types of this sub-derivation. Until now, this point was the source of the complexity of the definitions of expansion [6, 26, 25, 35]. Even if the need of duplicating more than one type is well understood, the definition of the set of types to be copied, is still a difficult problem. So for, no convincing justification has been given.

The contribution of this section is precisely the definition of this problematic set of types. The justification of this definition is obvious according to the previous results about the structure of principal and ground types.

The notion of left sub-terms does not take into account the full recursive structure of a type. We now define a notion of *generalized left sub-terms*, following the recursive structure of types to consider all possible sub-terms which are to the left of an arrow at any level in the recursive structure of a type.

Definition Let U be a B-type, we define the set $\mathcal{L}(U)$ of *generalized left sub-terms* of U in the following way:

- $L_0(U)$ is defined as for A-types
- $\forall n > 0$, $L_n(U) = \bigcup_{\rho \in L_{n-1}(U)} L_0(\rho)$
- $\mathcal{L}(U) == \bigcup_{n \geq 0} L_n(U)$

The next lemma precises the structure of ρ in a B-type $U = [\rho] \Rightarrow \rho$.

Lemma 13 *Let $U = [\rho] \Rightarrow \rho$ be a ground B-type and ρ' a sub-term of U then ρ' has two occurrences in U, one such that $\rho' \in \mathcal{T}_g$ and one such that $\rho' \in \mathcal{T}_{E_g}$.*

Proof

• If $\rho' = \rho$ then ρ verifies the property by definition of a B-type.

• Otherwise $\rho = [\rho_1, \ldots, \rho_n] \rightarrow \rho''$ where by definition of a ground B-type, ρ_i has no common type variable neither with the other ρ_j nor with ρ''. ρ' is a sub-term of $\rho_1, \ldots, \rho_n$ or ρ. Moreover, since $\rho \in \mathcal{T}_g \cap \mathcal{T}_{E_g}$, we have for $i = 1, \ldots, n$, $\rho_i \in \mathcal{T}_g \cap \mathcal{T}_{E_g}$ and $\rho'' \in \mathcal{T}_g \cap \mathcal{T}_{E_g}$ and each of them has two occurrences in U. If we consider ρ as an element of $\mathcal{T}_{E_g}$ (respectively $\mathcal{T}_g$), $\rho_1, \ldots, \rho_n$ are elements of $\mathcal{T}_g$ (respectively $\mathcal{T}_{E_g}$) and ρ'' of $\mathcal{T}_{E_g}$ (respectively $\mathcal{T}_g$). So ρ' has two occurrences in U such that one is element of $\mathcal{T}_{E_g}$ and the other of $\mathcal{T}_g$.$\Box$

We define in figure 7 an algorithm constructing the multi-set of types that we must duplicate when we expand a type.

Lemma 14 *Let U be a ground B-type and $\mu \in \mathcal{L}(U) \cap \mathcal{T}_g$. $Clos(\mu, U)$ is well-defined and verifies the following conditions:*

- *$Clos(\mu, U) \subset \mathcal{L}(U) \cap \mathcal{T}_{E_g}$*
- *$Clos(\mu, U) \Rightarrow \mu$ is a ground B-type*
- *$Clos(\mu, U)$ is the unique sub-multi-set of $\mathcal{L}(U) \cap \mathcal{T}_{E_g}$ which verifies the previous condition.*

Proof by induction on the structure of U.

• If $U = [\rho] \Rightarrow \rho$ then we notice that $\mu \neq \rho$, otherwise $\mu \notin \mathcal{L}(U) \cap \mathcal{T}_g$. By the lemma 13, we know that every sub-term of ρ has two occurrences, one belonging to $\mathcal{T}_{E_g}$ and the other belonging to

$Clos(\mu, U) =$

- **Case** $U = [\rho] \Rightarrow \rho$

  return $[\rho]$
- **Case** $U = [\nu_1, \dots, \nu_n] \Rightarrow \alpha$

  let $i \in \{1, \dots, n\}$ such that $\nu_i = [\mu_1^1, \dots, \mu_1^{n_1}] \to \dots \to [\mu_p^1, \dots, \mu_p^1, \dots, \mu_p^{n_p}] \Rightarrow \alpha$

  $(E_j^k)_{j=1,\dots,p,k=1,\dots,n_j}$ the partition of $[\nu_1, \dots, \nu_{i-1}, \nu_{i+1}, \dots, \nu_n]$

  such that $\forall j \in \{1, \dots, p\}, \forall k \in \{1, \dots, n_j\}, E_j^k \Rightarrow \mu_j^k$ is a ground B-type

  if $\exists j, k$ such that $\mu = \mu_j^k$

  then return E_j^k

  else if $\exists j, k$ such that$\mu \in \mathcal{L}(E_j^k \Rightarrow \mu_j^k) \cap \mathcal{T}_{E_g}$

  then return $Clos(\mu, E_j^k \Rightarrow \mu_j^k)$

  else fail
- **Case** $U = [\nu_1, \dots, \nu_n] \Rightarrow [\nu_{n+1}, \dots, \nu_{n+m}] \to \mu'$

  return $Clos(\mu, [\nu_1, \dots, \nu_{n+m}] \Rightarrow \mu')$

Figure 7: Closure algorithm

$\mathcal{T}_g$. Thus, there exists an occurrence of μ in U which belongs to $\mathcal{L}(U) \cap \mathcal{T}_{E_g}$. Moreover $[\mu] \Rightarrow \mu$ is a ground B-type since $\mu \in \mathcal{T}_{E_g} \cap \mathcal{T}_g$ and $[\mu]$ is the unique sub-multi-set of $\mathcal{L}(U) \cap \mathcal{T}_{E_g}$ since μ as only two occurrences in U.

- If $U = [\nu_1, \dots, nu_n] \Rightarrow \alpha$ then we distinguish several cases:
  - if $\exists (j,k) / \mu = \mu_j^k$, then by definition of a ground B-type E_j^k is well defined and unique. Moreover, $E_j^k \subset \mathcal{L}(U) \cap \mathcal{T}_{E_g}$ and $E_j^k \Rightarrow \mu$ is a ground B-type.
  - if $\exists (j,k) / \mu \in \mathcal{L}(E_j^k \Rightarrow \mu_j^k) \cap \mathcal{T}_{E_g}$ then by the induction hypothesis on $E_j^k \Rightarrow \mu_j^k$, $Clos(\mu, E_j^k \Rightarrow \mu_j^k)$ is well defined and verifies the conditions of the lemma. Since $\mathcal{L}(E_j^k \Rightarrow \mu_j^k) \subset \mathcal{L}(U)$, we only must show that $Clos(\mu, E_j^k \Rightarrow \mu_j^k)$ is the unique sub-multi-set of $\mathcal{L}(U) \cap \mathcal{T}_{E_g}$. Since each $E_j^k \Rightarrow \mu_j^k$ is a ground B-type, therefore closed, they are disjoint from each other and we deduce the unicity.
  - otherwise it is impossible since we suppose that $\mu \in \mathcal{L}(U) \cap \mathcal{T}_g$.
- If $U = [\nu_1, \dots, \nu_n] \Rightarrow [\nu_{n+1}, \dots, \nu_{n+m}] \to \mu'$ then the induction hypothesis gives the result.□

The next definition is just a syntactic facility.

Definition Let U be a ground B-type and $\mu \in \mathcal{L}(U) \cap \mathcal{T}_g$ a type, the ground B-type $Clos(\mu, U) \Rightarrow \mu$ is called the *closure* of μ in U.

An expansion makes a number of copies of several types. We want each copy of a type to be disjoint from all others, *i.e.* two copies of the same type have no common type variables. In order to be precise, we define specific substitutions which will make the copies of types exactly as we need.

Definition Let S be a substitution, we say that S is a *renaming substitution* if for all $\alpha \in Dom(S)$, $S(\alpha) = \beta$ where β is a type variable and S is injective. Furthermore, if $Range(S)$ is a set of new type variables, we say that S is a *fresh renaming substitution.* Thus any renaming substitution S has an inverse, written S^{-1} which is also a renaming substitution, but even if S is a fresh renaming substitution, S^{-1} is not a fresh renaming substitution since type variables in $Range(S^{-1}) = Dom(S)$ are not fresh.

We can now give the definition of expansions.

Definition Let p be an integer. For all types μ in $\mathcal{T}_g$ we define an operation of *expansion* of μ, on the ground B-type U, written $E_{(p,\mu)}$ by:

$$E_{(p,\mu)}(U) = \begin{cases} U & \text{if } \mu \notin \mathcal{L}(U) \\ U' & \text{otherwise} \end{cases}$$

where if $R_1, \ldots, R_p$ are p fresh renaming substitutions of domain $TypeVar(Clos(\mu, U))$, U' is obtained from U by replacing each occurrence of an element ν of $Clos(\mu, U)$ by $R_1(\nu), \ldots, R_p(\nu)$ and μ by $R_1(\mu), \ldots, R_p(\mu)$. We remark that since the renaming substitutions $R_1, \ldots, R_p$ are not unique, the expansion of μ in U is defined up to a renaming.

Since our work is essentially based on the structure of types, we want to prove that expansions do not change this structure. So we prove that the set of ground pairs is closed under expansion.

Definition We say that two types ρ_1 et ρ_2 *have the same structure* in one of the following cases:

- if $\rho_1 = \alpha$ and $\rho_2 = \beta$
- if $\rho_1 = [\mu_1, \ldots, \mu_n] \to \nu$, $\rho_2 = [\mu'_1, \ldots, \mu'_m] \to \nu'$ and for all $j \in \{1, \ldots, m\}$, there exists $i_j \in \{1, \ldots, n\}$ such that μ_{i_j} and μ_j have the same structure, and if ν and ν' have the same structure.
- if ρ_1 and ρ_2 belong to $\mathcal{T}_g$ and are such that $\rho_1 = [\nu_1, \ldots, \nu_n] \to \mu$ and $\rho_2 = [\nu'_1, \ldots, \nu'_n] \to \mu'$ for all $i \in \{1, \ldots, n\}$, ν_i and ν'_i have the same structure and μ and μ' have the same structure.

Remark: Let ρ be a type, $\mu \in \mathcal{T}_g \cap \mathcal{L}(\rho)$ and p an integer. If we replace in ρ the occurrence of μ by $R_1(\mu), \ldots, R_p(\mu)$ to obtain ρ', where $R_1, \ldots, R_p$ are fresh renaming substitutions then ρ' have the same structure as ρ. The proof by induction on the structure of ρ is immediate.

As usual, we extend the notion of having the same structure to B-types without difficulty. We say that two B-types U_1 and U_2 have the same structure if one of the following cases is verified:

- $U_1 = [\rho_1] \Rightarrow \rho_1$, $U_2 = [\rho_2] \Rightarrow \rho_2$ and ρ_1 and ρ_2 have the same structure.
- $U_1 = [\nu_1, \ldots, \nu_n] \Rightarrow \alpha$, $U_2 = [\nu'_1, \ldots, \nu'_m] \Rightarrow \beta$ and for all $j \in \{1, \ldots, m\}$, there exists $i_j \in \{1, \ldots, n\}$ such that ν_{i_j} and ν'_j have the same structure.
- $U_1 = [\nu_1, \ldots, \nu_n] \Rightarrow [\nu_{n+1}, \ldots, \nu_{n+p}] \to \mu$,
  $U_2 = [\nu'_1, \ldots, \nu'_m] \Rightarrow [\nu'_{m+1}, \ldots, \nu'_{m+q}] \to \mu'$,
  μ and μ' have the same structure and for all $j \in \{1, \ldots, m\}$ (respectively $\{m+1, \ldots, m+q\}$), there exists $i_j \in \{1, \ldots, n\}$ (respectively $\{n+1, \ldots, n+p\}$) such that ν_{i_j} and ν'_j have the same structure.

Lemma 15 *Let U be a ground B-type, $\mu' \in \mathcal{T}_g$, and p an integer. Then $E_{(p,\mu')}(U)$ is a ground B-type which has the same structure as U.*

Proof first by cases on μ' then by induction on the structure of U.

- If $\mu' \notin \mathcal{L}(U)$ then $E_{(p,\mu')}(U) = U$ is a ground B-type which has the same structure as U.
- Otherwise we reason by induction on the structure of U and three cases can arise:
  - if $U = [\rho]{\Rightarrow}\rho$ then μ' is a sub-term of ρ and $Clos(\mu', U) = [\mu']$. So we have $E_{(p,\mu')}(U) = [\rho']{\Rightarrow}\rho'$ where we have replaced μ' in ρ by the p copies of μ': $R_1(\mu'), \dots R_p(\mu')$, to obtain ρ'. Thus according to the previous remark, μ and μ' have the same structure. We can deduce that U and $E_{(p,\mu')}(U)$ have also the same structure.
  - if $U = [\nu_1, \dots, \nu_n]{\Rightarrow}\alpha$ then since U is a ground B-type, there exists $i \in \{1, \dots, n\}$ such that $\nu_i = [\mu_1^1, \dots, \mu_1^{n_1}] \rightarrow \dots \rightarrow [\mu_m^1, \dots, \mu_m^{n_m}] \rightarrow \alpha$ and there exists a partition $(E_j^k)_{j=1\dots m, k=1,\dots n_j}$ of $[\nu_1, \dots, \nu_{i-1}, \nu_{i+1}, \dots, \nu_n]$ such that for all $j \in \{1, \dots, m\}$ and all $k \in \{1, \dots, n_j\}$, $E_j^k{\Rightarrow}\mu_j^k$ is a ground B-type.
    - if there exist j and k such that $\mu' = \mu_j^k$ then $Clos(\mu', U) = E_j^k$ and so by definition of an expansion, we have:
      $E_{(p,\mu')}(U) =$
      $E_1^1 \cup \dots \cup E_1^{n_1} \cup \dots \cup E_j^1 \cup \dots \cup E_j^{k-1} \cup E_j^{k_1} \cup \dots \cup E_j^{k_p} \cup$
      $E_j^{k+1} \cup \dots \cup E_j^{n_j} \cup \dots \cup E_p^1 \cup \dots \cup E_p^{n_p} \cup$
      $[[\mu_1^1, \dots, \mu_1^{n_1}] \rightarrow \dots \rightarrow [\mu_j^1, \dots, \mu_j^{k-1}, \mu_j^{k_1}, \dots, \mu_j^{k_p}, \mu_j^{k+1} \dots, \mu_j^{n_j}]$
      $\rightarrow \dots \rightarrow [\mu_m^1, \dots, \mu_m^{n_m}] \rightarrow \alpha]{\Rightarrow}\alpha$
      where for all $l \in \{1, \dots, p\}$, $E_j^{k_l} = R_l(E_j^k)$ and $\mu_j^{k_l} = R_l(\mu_j^k)$. Then $R_l(E_j^k{\Rightarrow}\mu_j^k)$ is a ground B-type, since $E_j^k{\Rightarrow}\mu_j^k$ is a ground B-type and a renaming substitution does not change the structure of a B-type. We deduce that $E_{(p,\mu')}(U)$ is a ground B-type which has the same structure as U.
    - otherwise there exist j and k such that $\mu' \in \mathcal{L}(E_j^k{\Rightarrow}\mu_j^k) \cap \mathcal{T}_g$. We write $U_j^k = E_j^k{\Rightarrow}\mu_j^k$. Since U_j^k is a ground B-type, we can apply the expansion $E_{(p,\mu')}$. Let $U_j^{k'} = E_j^{k'}{\Rightarrow}\mu_j^{k'}$ be the result of this application. By induction, $U_j^{k'}$ is a ground B-type which has the same structure as U_j^k. So by definition of an expansion, we have:
      $E_{(p,\mu')}(U) =$
      $E_1^1 \cup \dots \cup E_1^{n_1} \cup \dots \cup E_j^1 \cup \dots \cup E_j^{k-1} \cup E_j^{k'} \cup$
      $E_j^{k+1} \cup \dots \cup E_j^{n_j} \cup \dots \cup E_p^1 \cup \dots \cup E_p^{n_p} \cup$
      $[[\mu_1^1, \dots, \mu_1^{n_1}] \rightarrow \dots \rightarrow [\mu_j^1, \dots, \mu_j^{k-1}, \mu_j^{k'}, \mu_j^{k+1}, \dots, \mu_j^{n_j}] \rightarrow \dots \rightarrow [\mu_m^1, \dots, \mu_m^{n_m}]]{\Rightarrow}\alpha$
      which is a ground B-type with the same structure as U.
  - if $U = [\nu_1, \dots, \nu_n]{\Rightarrow}[\nu_{n+1}, \dots, \nu_{n+m}] \rightarrow \mu$ then we write $U' = [\nu_1, \dots, \nu_{n+m}]{\Rightarrow}\mu$.

    By definition of a ground B-type, U' is immediately a ground B-type. So we can apply the expansion $E_{(p,\mu')}$. By induction, the result of this application is a ground B-type with the same structure as U'.

    Thus $E_{(p,\mu')}(U') = [\nu_1', \dots, \nu_q']{\Rightarrow}\mu'$ where μ and μ' have the same structure and for all $j \in \{1, \dots, q\}$, there exists $i_j \in \{1, \dots, n+m\}$ such that ν_{i_j} and ν_j have the same structure.

Let $[\nu'_{j_1}, \ldots, \nu'_{j_{q-r}}]$ and $[\nu'_{j_{q-r+1}}, \ldots, \nu'_{j_q}]$ be the two sub-multi-sets of $[\nu'_1, \ldots, \nu'_q]$ constituted by the types with the same structure as the types of the multi-set $[\nu_1, \ldots, \nu_n]$ and $[\nu_{n+1}, \ldots, \nu_{n+m}]$ respectively. Then $E_{(p,\mu')}(U) = [\nu'_{j_1}, \ldots, \nu'_{j_{q-r}}] \Rightarrow$
$[\nu'_{j_{q-r+1}}, \ldots, \nu'_{j_q}] \rightarrow \mu'$ and $E_{(p,\mu')}(U)$ is a ground B-type with the same structure as U.

□

The next lemma specifies some results about expansions' behavior according to the structure of B-types to which they are applied.

Lemma 16 *Let E be an expansion.*

- *If $E(\overline{A} \Rightarrow \mu) = \overline{A'} \Rightarrow \mu'$, then $E(\overline{A \setminus \{x\}} \Rightarrow A(x) \rightarrow \mu) = \overline{A' \setminus \{x\}} \Rightarrow A'(x) \rightarrow \mu'$*
- *If for $i = 1, \ldots, n$, $E(E_i \Rightarrow \mu_i) = E'_i \Rightarrow \mu'_i$, then $E(E_1 + \cdots + E_n + [[\mu_1] \rightarrow \cdots \rightarrow [\mu_n] \rightarrow \alpha] \Rightarrow \alpha) = E'_1 + \cdots + E'_n + [[\mu'_1] \rightarrow \cdots \rightarrow [\mu'_n] \rightarrow \alpha] \Rightarrow \alpha$*

Proof By definition of expansion and application of lemma 15.□

8 Principal typing of normalizable λ-terms

This section states the existence of principal types for all normalizable λ-terms. This result is not new, since it can be found in [6, 26, 35]. The authors of these papers introduce a notion of *approximants*, also named *λ-Ω-normal forms*, and define principal typing for these extended normal forms before generalizing to λ-terms using an approximation property, i.e. $B \vdash e : \mu$ if and only if there exists an approximant a of e such that $B \vdash a : \mu$. Here we are only interested in normalizable λ-terms. Thus, thanks to theorem 3, it is enough to use normal forms. We do not need to introduce approximants which significantly simplifies the proofs.

The substitution and expansion operations are both necessary to find a possible pair for a normal form from its principal pair. However these operations must be applied in an appropriate order.

Definition We name *chain* a composition of substitutions, expansions and renaming substitutions, of the form $S_n \circ \ldots \circ S_1 \circ O_m \circ \ldots \circ O_1$ where S_i is a substitution for $i = 1, \ldots, n$ and O_j is either a renaming substitution or an expansion for $j = 1, \ldots, m$.

From now on, we will not need to distinguish the different S_i (respectively O_j) from each other. So it will be enough to write a chain C simply by $\mathcal{C} \circ \mathcal{O}$ where $\mathcal{C}$ will be a composition of substitutions and $\mathcal{O}$ a composition of renaming substitutions and expansions.

Theorem 12 *Let N be a term in normal form such that $\vdash_{s\omega} N : \mu; A$. If $Infer(N) = (\mu_p, A_p)$ then there exists a chain C such that $C(\overline{A_p} \Rightarrow \mu_p) = \overline{A} \Rightarrow \mu$.*

Proof by induction on the structure of N.

• If $N = x$ then we have derived $\vdash_{s\omega} x : \mu; \{x : [\mu]\}$ and $Infer(N) = (\alpha, \{x : [\alpha]\})$ where α is fresh type variable. If we define C by $C = [\alpha/\mu]$, we have $C([\alpha] \Rightarrow \alpha) = [\mu] \Rightarrow \mu$.

• If $N = \lambda x.N_1$ then we have derived:

$$\frac{\vdash_{s\omega} N_1 : \mu_1; A_1}{\vdash_{s\omega} \lambda x.N_1 : A_1(x) \rightarrow \mu_1; A_1 \setminus \{x\}}$$

with $\mu = A_1(x) \to \mu_1$ and $A = A_1 \setminus \{x\}$.

On the other hand, if $\mathit{Infer}(N_1) = (\mu_{1p}, A_{1p})$ then $\mathit{Infer}(\lambda x.N_1) = (A_{1p}(x) \to \mu_{1p}, A_{1p} \setminus \{x\})$ with $\mu_p = A_{1p}(x) \to \mu_{1p}$ and $A_p = A_{1p} \setminus \{x\}$.

By the induction hypothesis, there exists a chain C such that $C(\overline{A_{1p}} \Rightarrow \mu_{1p}) = \overline{A_1} \Rightarrow \mu_1$. Moreover, $\overline{A_{1p}} = \overline{A_{1p} \setminus \{x\}} \cup A_{1p}(x)$ and a chain respects the structure of B-types. So we have:

$$C(\overline{A_{1p} \setminus \{x\}} \Rightarrow A_{1p}(x) \to \mu_{1p}) = \overline{A_1 \setminus \{x\}} \Rightarrow A_1(x) \to \mu_1$$

i.e., $C(\overline{A_p} \Rightarrow \to \mu_p) = \overline{A} \Rightarrow \mu$.

- If $N = x\ N_1 \dots N_n$ then we have derived:

$$\cfrac{\cfrac{\cfrac{\vdash_{s\omega} x : \mu_1; A \qquad \vdash_{s\omega} N_1 : \mu_1^j; A_1^j}{\vdash_{s\omega} x\ N_1 : \mu_2; B_1}}{\ddots}}{\ddots} \qquad \cfrac{\vdash_{s\omega} x\ N_1 \dots N_{n-1} : \mu_n; B_{n-1} \qquad \vdash_{s\omega} N_n : \mu_n^j; A_n^j}{\vdash_{s\omega} x\ N_1 \dots N_n : \mu; B_n}$$

where $\mu_k = [\mu_k^1, \dots, \mu_k^{m_k}] \to \dots \to [\mu_n^1, \dots, \mu_n^{m_n}] \to \mu$, $A = \{x : [[\mu_1^1, \dots, \mu_1^{m_1}] \to \dots \to [\mu_n^1, \dots, \mu_n^{m_n}] \to \mu]\}$ and for $k = 1, \dots, n$, $B_k = A + A_1^1 + \dots + A_1^{m_1} + \dots + A_k^1 + \dots + A_k^{m_k}$.

On the other hand, if α is a fresh type variable and if for $i = 1, \dots, n$, $\mathit{Infer}(N_i) = (\mu_{ip}, A_{ip})$, then

$$\mathit{Infer}(x\ N_1 \dots N_n) = (\alpha, A_p + A_{1p} + \dots + A_{np})$$

where $A_p = \{x : [[\mu_{1p}] \to \dots \to [\mu_{np}] \to \alpha]\}$.

Moreover, from the induction hypothesis, we deduce that for all $i \in \{1, \dots, n\}$, there exist m_i chains $C_i^1, \dots, C_i^{m_i}$ such that $\forall j \in \{1, \dots, m_i\}$, $C_i^j(\overline{A_{ip}} \Rightarrow \mu_{ip}) = \overline{A_i^j} \Rightarrow \mu_i^j$.

We write for all $i \in \{1, \dots, n\}$ and all $j \in \{1, \dots, m_i\}$, $C_i^j = \mathcal{S}_i^j \circ \mathcal{O}_i^j$ where $\mathcal{S}_i^j$ names a composition of substitutions and $\mathcal{O}_i^j$ a composition of expansions and renaming substitutions and $\overline{A_i^{j'}} \Rightarrow \mu_i^{j'} = \mathcal{O}_i^j(\overline{A_{ip}} \Rightarrow \mu_{ip})$.

Let $E_{(m_i, \mu_{ip})}$ be an expansion for all $i \in \{1, \dots, n\}$ and $R_i^1, \dots, R_i^{m_i}$ the associated renaming substitutions. If we define the chain C by:

$$\begin{aligned} C = [\alpha/\mu] o \mathcal{S}_n^{n_m} \circ \dots \circ \mathcal{S}_n^1 \circ \dots \circ \mathcal{S}_1^{m_1} \circ \dots \circ \mathcal{S}_1^1 \circ \\ \mathcal{O}_n^{m_n} \circ (R_n^{m_n})^{-1} \circ \dots \circ \mathcal{O}_n^1 \circ (R_n^1)^{-1} \circ \dots \circ \mathcal{O}_1^{m_1} \circ (R_1^{m_1})^{-1} \circ \dots \circ \mathcal{O}_1^1 \circ (R_1^1)^{-1} \circ \\ E_{(m_n, \mu_{pn})} \circ \dots \circ E_{(m_1, \mu_{p1})} \end{aligned}$$

then $C(\overline{A_p + A_{1p} + \dots + A_{np}} \Rightarrow \alpha) = \overline{B_n} \Rightarrow \mu$.

In fact, we have:

$$\begin{aligned} U' = E_{(m_n, \mu_{np})} \circ \dots \circ E_{(m_1, \mu_{1p})}(\overline{A_p + A_{1p} + \dots + A_{np}} \Rightarrow \alpha) = \\ \underline{[[R_1^1(\mu_{1p}), \dots, R_1^{m_1}(\mu_{1p})] \to \dots \to [R_n^1(\mu_{np}), \dots, R_n^{m_n}(\mu_{np})] \to \alpha]} + \\ \overline{R_1^1(A_{1p})} + \dots + \overline{R_1^{m_1}(A_{1p})} + \dots + \overline{R_n^1(A_{np})} + \dots + \overline{R_n^{m_n}(A_{np})} \Rightarrow \alpha \end{aligned}$$

Then for all $i \in \{1, \dots, n\}$ and all $j \in \{1, \dots, m_i\}$, $\mathcal{O}_i^j \circ (R_i^j)^{-1}(\overline{R_i^j(A_{ip})} \Rightarrow R_i^j(\mu_{ip})) = \overline{A_i^{j'}} \Rightarrow \mu_i^{j'}$, so we have:

$$
\begin{aligned}
U'' = {} & \mathcal{O}_n^{m_n} \circ (R_n^{m_n})^{-1} \circ \dots \circ \mathcal{O}_n^1 \circ (R_n^1)^{-1} \circ \dots \circ \mathcal{O}_1^{m_1} \circ (R_1^{m_1})^{-1} \circ \dots \circ \mathcal{O}_1^1 \circ (R_1^1)^{-1}(U') = \\
& \overline{[[\mu_1^{1'}, \dots, \mu_1^{m'_1}] \to \dots \to [\mu_n^{1'}, \dots, \mu_n^{m'_n}] \to \alpha] + A_1^{1'} + \dots + A_1^{m'_1} + \dots + A_n^{1'} + \dots + A_n^{m'_n}} \Rightarrow \alpha
\end{aligned}
$$

and since for all $i \in \{1, \dots, n\}$ and all $j \in \{1, \dots, m_i\}$, $S_i^j(\mu_i^{j'}) = \mu_i^j$ and $S_i^j(\overline{A_i^{j'}}) = \overline{A_i^j}$, we have:

$$
\begin{aligned}
U''' = {} & \mathcal{S}_n^{n_m} \circ \dots \circ \mathcal{S}_n^1 \circ \dots \circ \mathcal{S}_1^{m_1} \circ \dots \circ \mathcal{S}_1^1(U'') = \\
& \overline{[[\mu_1^1, \dots, \mu_1^{m_1}] \to \dots \to [\mu_n^1, \dots, \mu_n^{m_n}] \to \alpha] + A_1^1 + \dots + A_1^{m_1} + \dots + A_n^1 + \dots + A_n^{m_n}} \Rightarrow \alpha
\end{aligned}
$$

and finally

$$
\begin{aligned}
& [\alpha/\mu](U''') = \\
& \quad \overline{[[\mu_1^1, \dots, \mu_1^{m_1}] \to \dots \to [\mu_n^1, \dots, \mu_n^{m_n}] \to \mu] + A_1^1 + \dots + A_1^{m_1} + \dots + A_n^1 + \dots + A_n^{m_n}} \Rightarrow \mu
\end{aligned}
$$

So $[\alpha/\mu](U''') = \overline{B_n} \Rightarrow \mu$.□

Corollary *Let e be a normalizable term such that $\vdash_{s\omega} e : \mu; A$, N its normal form and $(\mu_p, A_p) = Infer(N)$. Then there exists a chain C such that $C(\overline{A_p} \Rightarrow \mu_p) = \overline{A} \Rightarrow \mu$.*

Proof

Since $e =_\beta N$ by theorem 3 we have $\vdash_{s\omega} N : \mu; A$ and we deduce the result directly from theorem 12.□

In other words, there exists a principal type for all normalizable λ-terms.

9 Related work

The first work on intersection type discipline has more than fifteen years. Since then many authors have been interested in intersection types or used them. But the presentation of this discipline does not really change since the beginning.

In [27, 5], M. Coppo, M. Dezani-Ciancaglini and P. Sallé introduce first a notion of intersection types which allows terms and term variables to have more than one type. In [3], M. Coppo and M. Dezani-Ciancaglini give the bases of an intersection type system and provide the first study of properties of a such system, but the system presented in [1] is considered as the reference in intersection type discipline and is the most often used and studied. According to J. L. Krivine in [13] (we use here his notations), there are essentially three intersection type systems. Types are formed with the constructors $\to$ and $\wedge$ for system $\mathcal{D}$ [3], a universal type ω for the system $\mathcal{D}_\omega$ [7], and a partial order $\leq$ on types for the system $\mathcal{D}_{\omega\leq}$ [1].

The theoretical study of system $\mathcal{D}$ led to results about λI-calculus and to the following characterization: a term is strongly normalizable if and only if it is typable in system $\mathcal{D}$ [22, 15]. In system $\mathcal{D}_\omega$ a term is normalizable if and only if it is typable in system $\mathcal{D}_\omega$ with a type in which ω does not occur [7, 15, 13]. The system $\mathcal{D}_{\omega\leq}$ gives rise to a filter lambda model that has been used as a starting point for much other work (see for example [4, 8, 9, 12]) and which is semantically complete [1]. Some papers about the principal type property complete these theoretical studies [6, 26, 35]. There are also many works which use intersection types in several domains (programming languages, partial evaluation, term rewriting system). [20, 10, 23, 11, 31, 34, 14] is a non exhaustive list of these works.

In [32, 33], S. van Bakel determines the essential characteristics of the intersection type systems and proposes two restrictions of system $\mathcal{D}_{\omega\leq}$ which have the same properties as the original system. As far as we know, his work is the first real advance in the simplication of the presentation of the intersection type discipline since the initial presentations.

Our result about the equivalence between normal forms and principal types has been reported in [28] as a preliminary report.

10 Conclusion

In this paper we have introduced a restriction of system $\mathcal{D}$ in the sense that intersections only occur on the left hand side of an arrow and are not types. This restriction corresponds to strict types defined by S. van Bakel in [32, 35]. But in the strict type system, S. van Bakel defines a partial order on types and his approach of the principal type property is close to the one proposed in the other works. In this article, we study principal types through their structural properties.

First, we have proved that though we have restricted the set of types in our intersection type system, this expressivenes is the same as the classical intersection type systems. We characterize by typing the same sets of types *i.e.*, normalizable and head normalizable λ-terms.

Then, with a restriction on the structure of types that occur in the inference rule for variables we prove the principal type property for normal forms in the classical sense.

We then have completely characterized the structure and the properties of principal types inferred for normal forms thanks to an algorithm for reconstructing normal forms from principal pairs of types and constraint environments.

According to the structure of principal types, we define ground types which correspond to S. van Bakel's ground types. But, as we did for principal types, we have studied in detail the structure of these ground types. These structural properties lead us to a simple definition of the expansion operation. Then we prove that in our intersection type system, expansions and substitutions are enough to find all possible types of a normalizable λ-term from a unique type called its principal type.

These results shed a new light on principal typing and we expect a better understanding of links between polymorphic typing and β-reduction, problem that we intend to study in future work.

References

[1] Henk P. Barendregt, Mario Coppo, and Mariangiola Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *Journal of Symbolic Logic*, 48(4):931–940, 1983.

[2] Felice Cardone and Mario 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] Mario Coppo and Mangiola 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] Mario Coppo, Mariangiola Dezani-Ciancaglini, and Furio Honsell. Applicative information system and D_∞-models. Technical report, Turino University, 1988.

[5] Mario Coppo, Mariangiola Dezani-Ciancaglini, and Patrick 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.

[6] Mario Coppo, Mariangiola Dezani-Ciancaglini, and Betti 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, London, 1980.

[7] Mario Coppo, Mariangiola Dezani-Ciancaglini, and Betti Venneri. Functional characters of solvable terms. *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik*, 27:45–58, 1981.

[8] Mario Coppo, Mariangiola Dezani-Ciancaglini, and Maddalena Zacchi. Type theories, normal forms and D_∞-lambda models. *Information and Computation*, 72(2):85–116, February 1987.

[9] Mariangiola Dezani-Ciancaglini and Ines Margaria. A characterization of F-complete type assignments. *Theoretical Computer Science*, 45:121–157, 1986.

[10] Philippa Gardner. Discovering needed reductions using type theory. In M. Hagiya and John C. Mitchell, editors, *Proceedings of TACS'94*, volume 789 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[11] C. Hankin and D. Le Metayer. Deriving algorithms from type inference systems: Applications to strictness analysis. In *Proceedings of the 21st Conference on Principles of Programming Languages*, pages 202–212. ACM Press, New York, 1994.

[12] Furio Honsell and Simona Ronchi Della Rocca. Models for theories of functions strictly depending on all their arguments. Technical report, Turino University, 1984.

[13] Jean-Louis Krivine. *Lambda-calcul, types et modèles*. Etudes et Recherches en Informatique. Masson, 1990.

[14] T.-M. Kuo and P. Mishra. Strictness analysis: a new perspective based on type inference. In *Proceedings of the Fourth International Conference on Functional Programming and Computer Architecture*. ACM Press, 1989.

[15] Daniel Leivant. Typing and computational properties of lambda-expressions. *Theoretical Computer Science*, 44:51–68, 1986.

[16] Xavier Leroy. *Typage polymorphe d'un langage algorithmique*. PhD thesis, University of Paris 7, June 1992.

[17] Ines Margaria and Maddalena Zacchi. Principal typing in a $\forall\wedge$-discipline. *Journal of Logic Computation*, 5(3):367–381, 1995.

[18] Robin Milner. A theory of type polymorphism in programming. *Computer and System Sciences*, 17(3):348–375, 1978.

[19] Robin Milner and Luis Damas. Principal type schemes for functional programs. In *Proceedings of Annual ACM Symposium on Principles of Programming Languages*, pages 207–212, 1982.

[20] Benjamin 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, September 1989.

[21] Benjamin Pierce. *Programming with Intersection Types and Bounded Quantification.* PhD thesis, Carnegie Mellon University, 1991.

[22] Garel 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.

[23] John C. Reynolds. Preliminary design of the programming language Forsythe. Technical Report CMU-CS-88-159, Carnegie Mellon University, June 1988.

[24] John C. Reynolds. Syntactic control of interference, part 2. Technical Report CMU-CS-89-130, Carnegie Mellon University, April 1989.

[25] Simona Ronchi Della Rocca. Principal type scheme and unification for intersection type discipline. *Theoretical Computer Science*, 59:181–209, 1988.

[26] Simona Ronchi Della Rocca and Betti Venneri. Principal type schemes for an extended type theory. *Theoretical Computer Science*, 28:151–169, 1984.

[27] Patrick 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.

[28] Émilie Sayag and Michel Mauny. Characterization of principal types of normal forms in intersection type system. In *Proceedings of Foundations of Software Technology and Theoretical Computer Science*, 1996. To appear.

[29] Zhong Shao and Andrew W. Appel. Smartest recompilation. Technical Report CS-TR-395-92, Princeton University, October 1992.

[30] Jim Trevor. Rank 2 type systems and recursive definitions. Technical Report MIT/LCS/TM-531, Laboratory for Computer Science, Massachussetts Institute of Technology, August 1995.

[31] Jim Trevor. What are principal typings and what are they good for? Technical Report MIT/LCS/TM-532, Laboratory for Computer Science, Massachussetts Institute of Technology, August 1995.

[32] Steffen van Bakel. Complete restrictions of the intersection type discipline. *Theoretical Computer Science*, 102:135–163, 1992.

[33] Steffen van Bakel. Essential intersection type assignment. In R. K. Shyamasundar, editor, *Proceedings of Thirteenth Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 761 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[34] Steffen van Bakel. Partial intersection type assignment in applicative term rewriting systems. In M. Bezem and J.-F. Groote, editors, *Proceedings of International Conference on Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science*, pages 29–44. Springer-Verlag, 1993.

[35] Steffen van Bakel. Principal type schemes for strict type assignment system. *Logic and Computation*, 3(6):643–670, 1993.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105,
78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur

INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)

ISSN 0249-6399