Theoretical Computer Science 59 (1988) 181–209
North-Holland

PRINCIPAL TYPE SCHEME AND UNIFICATION FOR INTERSECTION TYPE DISCIPLINE

Simona RONCHI DELLA ROCCA

Dipartimento di Informatica, Universita' di Torino, 10149 Torino, Italy

Abstract. The intersection type discipline for the λ-calculus (ITD) is an extension of the classical functionality theory of Curry. In the ITD a term satisfying a given property has a principal type scheme in an extended meaning, i.e., there is a type scheme deducible for it from which all and only the type schemes deducible for it are reachable, by means of suitable operations. The problem of finding the principal type scheme for a term, if it exists, is semidecidable. In the paper a procedure is shown, building the principal type scheme of a term through the construction of the most general unifier for intersection type schemes.

1. Introduction

The intersection type discipline for the λ-calculus (ITD), defined in [5] is an extension of the classical functionality theory of Curry [6]. In Curry type discipline, type schemes are built from type variables using the constructor → (arrow), and type schemes are assigned to λ-terms by means of deductions in a natural deduction style. In the ITD, type schemes are built from type variables and the type constant ω (the universal type) using as constructor the intersection ($\wedge$) in addition to arrow.

The semantics of a type scheme of the shape $\alpha \to \beta$ is the classical one, the semantics of a type scheme of the shape $\alpha \wedge \beta$ is the intersection of the sets representing the meanings of α and β, the semantics of ω is the whole domain of values. Type schemes are assigned to λ-terms by means of a deduction system, which is an extension of Curry's system.

In Curry's type discipline, the set of terms having a type scheme is a proper subset of the set of strongly normalizing terms. Moreover, the type schemes are preserved under β-reduction (i.e., if the term X β-reduces to Z, then every type scheme deducible for X can be deduced also for Z), while they are not preserved, in general, under β-expansion. In the ITD, every term has at least one type scheme, and type schemes are preserved under β-convertibility. Moreover, the set of all type schemes deducible for the same term is a filter (i.e., it contains ω and it is closed under intersection and semantic inclusion). So the ITD gives rise to a model of the λ-calculus, where the domain of values is the set of filters built on type schemes, and the interpretation of a term is the set of type schemes deducible for it. In this model an approximation theorem holds, in the sense that the interpretation of a term is the supremum of the set of interpretations of its approximants.

In Curry's type discipline, every term X which can be typed has a principal type scheme (pts), from which all and only the type schemes deducible for X can be

derived, by means of substitutions. The problem of computing the pts, if it exists, of a term in Curry's type discipline is decidable, and algorithms to solve it have been proposed by Hindley [7] (for terms of Combinatory Logic), and by Milner [9, 10] (for terms of λ-calculus). Milner uses an extension of this algorithm in the design of the ML type checker. Both these algorithms are based on the classical unification algorithm of Robinson [11].

In the ITD each term X, which has a finite set of approximants, has a pts in an extended meaning. More precisely, every type scheme deducible for X is derived from its pts by means of a sequence of suitable operations, namely substitution, expansion and lifting [13]. The problem of finding the pts of a term, if it exists, is semidecidable.

The aim of this paper is to build the pts of a term, if it exists, through the construction of the most general unifier for intersection type schemes.

The procedure UNIFY is presented, not only based on substitution, which, when applied to two type schemes, either gives a unifier for them, or it does not stop. In particular, UNIFY is complete for type schemes belonging to a given class, and in this case it gives the most general unifier, if it exists. While UNIFY uses also operations different from substitution, it is conservative with respect to the unification algorithm for Curry's type schemes. More precisely, if σ and τ are two Curry type schemes such that their most general unifier, according to Robinson's algorithm, is the substitution s, then UNIFY, when applied to σ and τ, gives s as output.

Moreover, the procedure PP is presented, using UNIFY as an essential tool, which builds the pts of a term, if it exists. It is proved that the procedure PP stops iff the input term is strongly normalizing. Since there is a one-one correspondence (modulo α-conversion) between a normal form and its pts, PP can be viewed also as a reduction machine, using an innermost reduction strategy. The use of unification between type schemes instead of β-reduction in computing the normal form of a term avoids the necessity of α-conversions.

The procedure PP is being implemented by a student of the present author [3], on a VAX 780, in Pascal.

It can be interesting to introduce other type constants in the ITD, such as "integer" or "boolean" for instance, in order to use it for real programming languages. The definition of principal pair, and both procedures UNIFY and PP can easily be extended in such a way that they can deal with any kind of type constants.

2. The intersection type discipline

The reader is supposed to have some acquaintance with λ-calculus; in any case he can refer to [1] whose notations we will use.

Definition 2.1. (i) The set T of *type schemes* is inductively defined by

- $\varphi_i, \psi_i, \ldots \in T$ $(i \geq 0)$ (type variables),

- $\omega \in T$ (type constant),
- $\sigma, \tau \in T \Rightarrow (\sigma \to \tau) \in T$ (arrow type scheme) and $(\sigma \wedge \tau) \in T$ (intersection type scheme).

(ii) A *statement* is of the form σM with $\sigma \in T$ and M is a term. M is the *subject* and σ the *predicate* of σM. A *basis scheme* is a (possibly infinite) set of statements, where all subjects are variables.

The notion of syntactic subtype of a given type scheme is obtained in a straightforward way from Definition 2.1. $\sigma_1 \to \sigma_2 \to \cdots \to \sigma_n \to \tau$ is an abbreviation for $\sigma_1 \to (\sigma_2 \to (\sigma_3 \to \ldots (\sigma_n \to \tau)\ldots))$ and $\sigma_1 \wedge \sigma_2 \wedge \cdots \wedge \sigma_n$ is an abbreviation for $(\sigma_1 \wedge (\sigma_2 \wedge \ldots (\sigma_{n-1} \wedge \sigma_n)\ldots))$ (this order between intersections is given only for syntactical convenience since, semantically, the intersection is associative and commutative as can be seen in the following).

The *simple semantics* (for the definition of simple semantics, see [8]) for T can be given in the following way.

Definition 2.2. Let $\mathcal{M} = \langle D, \cdot, [\![\ ]\!] \rangle$ be a λ-model, and let Λ be the set of untyped λ-terms. Let $\mathcal{P}(D) = \{X \mid X \subseteq D\}$ and $V: \{\varphi \mid \varphi \text{ is a type variable}\} \to \mathcal{P}(D)$. Then the interpretation of $\sigma \in T$ in $\mathcal{M}$ via V, denoted $[\![\sigma]\!]^{\mathcal{M}}_V \in \mathcal{P}(D)$, is defined as follows:

- $[\![\omega]\!]^{\mathcal{M}}_V = D$,
- $[\![\varphi]\!]^{\mathcal{M}}_V = V(\varphi)$,
- $[\![\sigma \to \tau]\!]^{\mathcal{M}}_V = \{d \in D \mid \forall e \in [\![\sigma]\!]^{\mathcal{M}}_V.\ d \cdot e \in [\![\tau]\!]^{\mathcal{M}}_V\}$,
- $[\![\sigma \wedge \tau]\!]^{\mathcal{M}}_V = [\![\sigma]\!]^{\mathcal{M}}_V \cap [\![\tau]\!]^{\mathcal{M}}_V$.

This semantics naturally induces a pre-order relation $\leqslant$ on T, whose intended meaning is $\sigma \leqslant \tau \Leftrightarrow \forall \mathcal{M}, V.\ [\![\sigma]\!]^{\mathcal{M}}_V \subseteq [\![\tau]\!]^{\mathcal{M}}_V$. This semantics relation is syntactically characterized as follows [2].

Definition 2.3. The relation $\leqslant$ (and $\sim$) on T is inductively defined by

(i) $\tau \leqslant \tau$, $\quad \tau \leqslant \omega$, $\quad \omega \leqslant \omega \to \omega$,

$\tau \leqslant \tau \wedge \tau$, $\quad \tau \wedge \sigma \leqslant \sigma$, $\quad \tau \wedge \sigma \leqslant \tau$,

$(\sigma \to \rho) \wedge (\sigma \to \tau) \leqslant \sigma \to (\rho \wedge \tau)$,

$\sigma \leqslant \tau \leqslant \rho \Rightarrow \sigma \leqslant \rho$,

$\sigma \leqslant \sigma'$ and $\tau \leqslant \tau' \Rightarrow \sigma \wedge \tau \leqslant \sigma' \wedge \tau'$,

$\sigma \geqslant \sigma'$ and $\tau \leqslant \tau' \Rightarrow \sigma \to \tau \leqslant \sigma' \to \tau'$;

(ii) $\sigma \sim \tau \Leftrightarrow \sigma \leqslant \tau \leqslant \sigma$.

Definition 2.4 (*Type scheme assignment rules*). Let B be a basis scheme.

$(\omega)\, B \vdash \omega X$ for all $X \in \Lambda$, $\quad (\mathrm{var})\, B \cup \{\sigma x\} \vdash \sigma x$,

$$(\wedge \mathrm{I})\frac{B \vdash \sigma X \quad B \vdash \tau X}{B \vdash (\sigma \wedge \tau) X} \qquad (\wedge \mathrm{E})\frac{B \vdash (\sigma \wedge \tau) X}{B \vdash \sigma X} \quad \frac{B \vdash (\sigma \wedge \tau) X}{B \vdash \tau X}$$

$$(\to \mathrm{I})\frac{B \cup \{\sigma x\} \vdash \tau X}{B \vdash (\sigma \to \tau)\lambda x.X} \qquad (\to \mathrm{E})\frac{B \vdash (\sigma \to \tau) X \quad B \vdash \sigma Y}{B \vdash \tau XY}$$

(in $(\to \mathrm{I})$ σx is the only statement of $B \cup \{\sigma x\}$, whose subject is x, that is used to deduce τX).

$$(\leqslant)\frac{B \vdash \sigma X \quad \sigma \leqslant \tau}{B \vdash \tau X}$$

Note that the rule $(\wedge \mathrm{E})$ is redundant since it can be directly derived from rule $(\leqslant)$.

Notation. Let Z, W, z be either two type schemes and a type variable, or two terms and a variable. $Z[z/W]$ denotes the type scheme (term) obtained from Z by simultaneous replacing each free occurrence of z with W. If W is a (type) variable not occurring in Z, $Z[z/W]$ is an *instance* of Z.

Let $\to_\beta$ denote the β-reducibility, i.e.,

$$C[(\lambda x.M)N] \to_\beta C[M[x/N]] \quad \text{(where } C[\,] \text{ is any context)}$$

iff N is substitutible for x in M (note that a term N can always become substitutible for a given variable in a term M by means of renaming of bound variables in M). Let $=_\beta$ denote β-convertibility, i.e., the transitive, symmetric and reflexive closure of β-reducibility.

We recall that a term X is in *normal form* iff there is no X' such that $X \to_\beta X'$, and it is in *head normal form* iff $X = \lambda x_1 x_2 \ldots x_n.\zeta X_1 X_2 \ldots X_m$ $(n, m \geqslant 0)$, where ζ is any variable and $X_i \in \Lambda$ $(1 \leqslant i \leqslant m)$. X has a normal form (a head normal form) iff $X =_\beta X'$ and X' is in normal form (head normal form).

The following theorem holds.

Theorem 2.5 (Barendregt et al. [2]). (i) *Let $X =_\beta X'$. Then $B \vdash \tau X \Leftrightarrow B \vdash \tau X'$.*

(ii) *$\exists B, \tau \neq \omega$. $B \vdash \tau X \Leftrightarrow X$ has a head normal form.*

(iii) *$\exists B, \tau$. $B \vdash \tau X$ and ω does not occur neither in B nor in $\tau \Leftrightarrow X$ has a normal form.*

Without loss of generality, we can restrict ourselves to considering only finite basis schemes.

Definition 2.6. (i) A *pair* $\langle B, \tau \rangle$, where B is a basis scheme and τ is a type scheme, is *suitable* for $X \in \Lambda$ iff there exists a deduction D such that $D: B \vdash \tau X$.

(ii) $\equiv$ will denote the syntactical identity between type schemes, basis schemes and pairs.

(iii) The equivalence relation $\sim$ between pairs is defined as follows:

$$\langle B, \tau\rangle \sim \langle B', \tau'\rangle \Leftrightarrow \tau \sim \tau'$$

and if, $\forall x.\sigma_i x$ $(1 \leqslant i \leqslant m)$ and $\rho_j x$ $(1 \leqslant j \leqslant n)$ are all and the only statements whose subject is x belonging respectively to B and B', then

$$\sigma_1 \wedge \cdots \wedge \sigma_m \sim \rho_1 \wedge \cdots \wedge \rho_n.$$

Now we will define three functions on pairs, which preserve suitable pairs, namely substitution, expansion and lifting.

Definition 2.7. (i) Let f be a finite partial function from type variables to type schemes. The *substitution* s_f is the total function from type schemes to type schemes defined as follows:

$$s_f(\tau) \equiv \tau[\varphi_1/\mu_1] \ldots [\varphi_n/\mu_n] \text{ iff } f(\varphi_i) \equiv \mu_i \ (1 \leqslant i \leqslant n);$$

(ii) by an abuse of notation, a substitution s_f can be extended to basis schemes and pairs in the following way:

$$s_f(B) = \{s_f(\sigma)x \mid \sigma x \in B\}, \qquad s_f(\langle B, \tau\rangle) \equiv \langle s_f(B), s_f(\tau)\rangle.$$

Notation. The finite partial function f such that $f(\varphi_i) \equiv \tau_i$ $(1 \leqslant i \leqslant n)$ will be denoted by $[\varphi_1 \Rightarrow \tau_1, \ldots, \varphi_n \Rightarrow \tau_n]$. $\perp$ will denote the everywhere undefined function.

Clearly, if $D: B \vdash \tau X$ and s is any substitution, $\exists D': s(B) \vdash s(\tau)$, where D' is obtained from D by simply applying s to every type scheme occurring in D.

Definition 2.8. Let A be a finite set of type schemes, and let $!A = \{\tau \mid \exists \delta \in A$ and τ is a subtype of $\delta\}$ (*completion* of A), and let $\mu \in !A$. The *expansion* $e_{A,\mu}$ is the partial function, whose domain is $!A$, from type schemes to type schemes defined as follows:

(i) Let $L(\mu, !A)$ be a set of type schemes, built as follows:

(1) if ρ is a subtype of μ, and ρ is not an intersection type scheme, then $\rho \in L(\mu, !A)$;

(2) if $\sigma \in L(\mu, !A)$ and δ is a proper subtype of σ and δ is not an intersection, then $\delta \in L(\mu, !A)$;

(3) for each subtype σ of an element of $!A$, if either $\sigma = \nu \to \delta$ and $\delta \in L(\mu, !A)$, or $\sigma = \nu \to (\delta \wedge \alpha)$ and either δ or α belongs to $L(\mu, !A)$, then $\sigma \in L(\mu, !A)$.

Let $L'(\mu, !A)$ be the list obtained by ordering the elements of $L(\mu, !A)$ in decreasing ordering according to the number of symbols (if two type schemes have the same number of symbols, their mutual order is unimportant).

(ii) Let $I = \{\varphi_i \mid \varphi_i$ is a type variable occurring in $L(\mu, !A)\}$. Let $f = [\varphi_i \Rightarrow \psi_i \mid \varphi_i \in I$ and ψ_i is a new type variable, not occurring in $!A]$, and let $g = [\varphi_i \Rightarrow \chi_i \mid \varphi_i \in I$ and χ_i is a new type variable, not occurring in $!A]$.

(iii) Let $\tau \in !A$. $e_{A,\mu}(\tau)$ is obtained from τ by examining in order each element of $L'(\mu, !A)$ and, each time an element ν is found which is a subtype of τ, by replacing ν in τ by $s_f(\nu) \wedge s_g(\nu)$.

(iv) With an abuse of notation, an expansion $e_{A,\mu}$ can be extended to pairs in the following way:

- $e_{A,\mu}$ is defined on $\langle B, \tau\rangle$ iff $\tau \in !A$ and $\forall\sigma.(\sigma x \in B$ implies $\sigma \in !A)$,
- $e_{A,\mu}(\langle B, \tau\rangle) \equiv \langle e_{A,\mu}(B), e_{A,\mu}(\tau)\rangle$, where $e_{A,\mu}(B) = \{e_{A,\mu}(\sigma)x \mid \sigma x \in B\}$.

Note that the expansion is defined modulo the names of new type variables.

Example 2.9. Let $\alpha, \beta, \gamma, \delta$ be type variables, $\sigma \equiv ((\alpha \wedge \beta) \to \gamma) \to \delta$, and $A = \{\sigma\}$.

(1) $L(\gamma, !A) = \{(\alpha \wedge \beta) \to \gamma, \alpha, \beta, \gamma\}$. Then

$$e_{A,\gamma}(\sigma) \equiv (((\alpha_1 \wedge \beta_1) \to \gamma_1) \wedge ((\alpha_2 \wedge \beta_2) \to \gamma_2)) \to \delta,$$

where $\alpha_i, \beta_i, \gamma_i$ are instances of α, β, γ $(1 \leq i \leq 2)$.

(2) $L(\delta, !A) = \{\sigma, (\alpha \wedge \beta) \to \gamma, \alpha, \beta, \gamma\}$. Then $e_{A,\delta}(\sigma) \equiv \sigma_1 \wedge \sigma_2$, where σ_i is an instance of σ $(1 \leq i \leq 2)$.

Let $D: B \vdash \tau X$. Then, if e is an expansion, there exists a deduction $D': e(B) \vdash e(\tau)X$, and D' is obtained from D by duplicating some subdeductions of D and by adjoining some applications of the rule $(\wedge I)$ (see the proof in [13]).

Example 2.10. Let $B = \{(\varphi \to \varphi) \to \alpha y\}$. Then $\langle B, \alpha\rangle$ is a suitable pair for the term $y(\lambda x.x)$. In fact, we can show the deduction D:

$$(\to E)\ \frac{B \vdash (\varphi \to \varphi) \to \alpha y \qquad (\to I)\ \dfrac{B \cup \{\varphi x\} \vdash \varphi x}{B \vdash \varphi \to \varphi \lambda x.x}}{B \vdash \alpha y(\lambda x.x)}$$

Consider the expansion $e_{A,\varphi}$, where

$$A = \{(\varphi \to \varphi) \to \alpha\}, \qquad e_{A,\varphi}(\langle B, \alpha\rangle) = \langle\{((\varphi_1 \to \varphi_1) \wedge (\varphi_2 \to \varphi_2)) \to \alpha y\}, \alpha\rangle$$

is a suitable pair for $y(\lambda x.x)$. In fact, if $B' = e_{A,\varphi}(B)$, there exists a deduction D':

$$\{\to E\}\ \frac{B' \vdash ((\varphi_1 \to \varphi_1) \wedge (\varphi_2 \to \varphi_2)) \to \alpha y \qquad \begin{array}{c}(\to I)\ \dfrac{B' \cup \{\varphi_1 x\} \vdash \varphi_1 x}{B' \vdash \varphi_1 \to \varphi_1 \lambda x.x} \quad (\to I)\ \dfrac{B' \cup \{\varphi_2 x\} \vdash \varphi_2 x}{B' \vdash \varphi_2 \to \varphi_2 \lambda x.x} \\ (\wedge I)\ \dfrac{\qquad\qquad\qquad\qquad\qquad\qquad}{B' \vdash (\varphi_1 \to \varphi_1) \wedge (\varphi_2 \to \varphi_2) \lambda x.x}\end{array}}{B' \vdash \alpha y(\lambda x.x).}$$

Note that if e is any expansion and s is any substitution, $e(\omega) \equiv \omega$ and $s(\omega) \equiv \omega$.

Definition 2.11. A *lifting* r_a, where $a \equiv \langle B', \sigma\rangle$ is a total function from pairs to pairs defined as follows:

$$r_a(\langle B, \tau\rangle) \equiv \textbf{if } \tau \leq \sigma,$$
$$\textbf{and } \forall x.(\sigma_i x \in B\ (1 \leq i \leq n) \textbf{ and } \tau_j x \in B'\ (1 \leq j \leq m)$$
$$\Rightarrow \sigma_1 \wedge \cdots \wedge \sigma_n \geq \tau_1 \wedge \cdots \wedge \tau_m),$$
$$\textbf{then } \langle B', \sigma\rangle \textbf{ else } \langle B, \tau\rangle.$$

In [13] it is proved that the lifting operation preserves suitable pairs: namely, if $\langle B, \tau\rangle$ is such that there exists a $D: B \vdash \tau X$, and r is any lifting such that $r(\langle B, \tau\rangle) \equiv \langle B', \tau'\rangle$, then there exists a deduction $D': B' \vdash \tau' X$, and D' is obtained from D by adjoining to D some applications of rule ($\leq$).

Definition 2.12. (i) A *chain* c is a finite composition of substitutions and expansions.
(ii) A *lifting chain* is a finite composition of substitution, expansions and lifting.

Let op_i be an expansion or a substitution ($1 \leq i \leq n$). $\mathrm{op}_1.\mathrm{op}_2 \ldots \mathrm{op}_n$ will denote the chain c such that $c(\sigma) \equiv \mathrm{op}_n(\mathrm{op}_{n-1}(\ldots.(\mathrm{op}_1(\sigma))\ldots))$ and $c_1.c_2$ will denote the chain which is the composition of the two chains c_1 and c_2, i.e., $c(\sigma) \equiv c_2(c_1(\sigma))$. The domain of $\mathrm{op}_1 \ldots \mathrm{op}_n$ is the set of type schemes σ such that $\mathrm{op}_1 \ldots \mathrm{op}_i(\sigma)$ belongs to the domain of op_{i+1}. Since in the definition of expansion the names of new type variables are not fixed (in fact, they are not essential), it would in general become impossible to decide whether a chain is defined or not on a given type scheme. To avoid this difficulty, we can use indexed type variables, and we can assume to have a generator of new indexes: then, if n is the maximum index occurring in σ, an expansion on σ will introduce new type variables indexed by $n+1, n+2, \ldots$.

$=$ will denote the extensional equality between partial functions: if c_1, c_2 are two chains, $c_1 = c_2$ iff $\mathrm{dom}(c_1) = \mathrm{dom}(c_2) = A$ and, $\forall \sigma \in A$, $c_1(\sigma) \equiv c_2(\sigma)$. When it is important to remember the domain of the equality, we can write $c_1 =_A c_2$.

Definition 2.13. The measure of the expansion $e_{A,\mu}$ is the integer $E(e_{A,\mu}) = |L(\mu, !A)|$ ($|B|$ denotes the cardinality of the set B). Then the measure of a chain c is $E(c) = \sum_{e \in Z} E(e)$, where Z is the set of all and the only expansions occurring in c.

Theorem 2.14 (Ronchi et al. [13]). (i) *Let c be a (lifting) chain. If $\langle B, \tau\rangle$ is a suitable pair for X, then $c(\langle B, \tau\rangle)$ is also a suitable pair for X.*

(ii) *Let σ be an arrow type scheme, and let c be a chain such that $c(\sigma)$ is an intersection type scheme, and let $A \subseteq \mathrm{dom}(c)$ such that $\sigma \in A$ and ρ, $\tau \in A$ implies ρ and τ are disjoint. Then $c =_A e_{A,\sigma}.c'$, and $E(c') < E(c)$.*

3. Principal pair

In [2] it is proved that the intersection type discipline gives rise to a model for the λ-calculus. More precisely, if a filter is any subset of T containing ω and closed under $\wedge$ and $\leq$, the domain of values of this model is the set of filters built on T, and the interpretation of a term is the set of type schemes deducible for it, which is a filter. Moreover, in [12] it is proved that an approximation theorem holds, in the sense that the interpretation of a term is the supremum of the set of interpretations of its (syntactical) approximants. The approximants of a term can be inductively defined as follows.

Definition 3.1. (i) The set $\mathcal{N}$ of *approximate normal forms* is defined from the set of variables plus a new constant symbol Ω in the following way:

- $\Omega \in \mathcal{N}$, $x \in \mathcal{N}$ for all variables x,
- if x is a variable and $A \in \mathcal{N} (A \neq \Omega)$, then $\lambda x.A \in \mathcal{N}$,
- if x is a variable and $A_1, \ldots, A_p \in \mathcal{N}$ $(p \geq 0)$, then $xA_1 \ldots A_p \in \mathcal{N}$

(ii) Let X be a term and $A \in \mathcal{N}$. A is an *approximant* of X $(A \sqsubseteq X)$ iff $\exists X' =_\beta X$ such that A matches X' except at occurrences of Ω in A.

(iii) $\mathcal{A}(X) = \{A \mid A \sqsubseteq X\}$.

(iv) The type assignment rules of Definition 2.4 are generalized to elements of $\mathcal{N}$ simply by extending in an obvious way the definition of statement and by adjoining the following rule:

(ω') $B \vdash \omega A$ for all $A \in \mathcal{N}$.

The following theorem holds.

Theorem 3.2 (Ronchi et al. [13]). *$\langle B, \tau \rangle$ is a suitable pair for $M \in \Lambda$ iff $\langle B, \tau \rangle$ is a suitable pair for some $A \in \mathcal{A}(M)$*

We can define, for an approximate normal form A, a unique *principal pair* (pp(A)) (modulo the relation $\sim$). The principal pair of A is a pair $\langle B, \pi \rangle$ such that there exists a deduction $D: B \vdash \pi A$, and D is one of the shortest deductions of a type scheme for A (D is called a *principal deduction* of A). In particular, D does not contain applications of the rule $(\leq)$, every subdeduction of D of the shape $B' \vdash \omega M$ is a single application of the rule (ω), and there is no a subdeduction of D of the shape $B' \vdash \sigma M$ where $\sigma \sim \omega$ and $\sigma \neq \omega$. The main property of pp(A) is that starting from pp(A) it is possible to generate all the pairs suitable for A itself. More precisely, for any $\langle B, \sigma \rangle$ suitable for A there is a lifting chain c such that $\langle B, \sigma \rangle \sim c(\text{pp}(A))$ [13].

Definition 3.3. Let $A \in \mathcal{N}$;

(i) if $A \equiv \Omega$, then pp(A) $\sim \langle \Phi, \omega \rangle$ (Φ is the empty set);

(ii) if $A \equiv x$, then pp(A) $\sim \langle \{\varphi x\}, \varphi \rangle$, where φ is a type variable;

(iii) if $A \equiv \lambda x.A'$, and pp(A') $\sim \langle B', \pi' \rangle$, then

(1) if x occurs in A', pp(A) $\sim \langle B' - \{\sigma x\}, \sigma \to \pi' \rangle$, where σ is the intersection of the predicates of B' whose subject is x,

(2) otherwise, pp(A) $\sim \langle B', \omega \to \pi' \rangle$;

(iv) if $A \equiv xA_1 \ldots A_n$ and pp(A_i) $\sim \langle B_i, \pi_i \rangle$ $(1 \leq i \leq n)$ (we choose a trivial variant of them such that they are pairwise disjoint), then pp(A) $\sim \langle \bigcup_{1 \leq i \leq n} B_i \cup \{\pi_1 \to \cdots \to \pi_n \to \varphi\}, \varphi \rangle$, where φ is a type variable which does not occur in B_i, π_i $(1 \leq i \leq n)$.

The components of pp(A) are called respectively the *principal basis scheme* and the *principal type scheme* of A.

Note that the principal pair is defined modulo names of type variables.
Let

$$\Pi(X) = \{\langle B, \pi\rangle | \exists A \in \mathscr{A}(X). \langle B, \pi\rangle \sim \mathrm{pp}(A)\} \quad \text{and}$$

$$\mathscr{P} = \{\langle B, \pi\rangle | \exists A \in \mathscr{N}. \langle B, \pi\rangle \sim \mathrm{pp}(A)\}.$$

On $\mathscr{P}$ it is possible to define the following preorder relation:

$$\langle B, \pi\rangle \sqsubseteq_\omega \langle B', \pi'\rangle \Leftrightarrow \exists \varphi_1 \ldots \varphi_n. \langle B, \pi\rangle = \langle B'[\varphi_1/\omega, \ldots, \varphi_n/\omega], \pi'[\varphi_1/\omega, \ldots, \varphi_n/\omega]\rangle.$$

Property 3.4. *$\mathscr{P}, \sqsubseteq_\omega$ is a meet semilattice isomorphic to $\mathscr{N}, \sqsubseteq$.*

Then $\Pi(X)$ is an ideal in $\mathscr{P}$ and therefore if $\Pi(X)$ is finite, there exists a pair $\langle B, \pi\rangle \sim \bigsqcup \Pi(X)$, where $\langle B, \pi\rangle \in \mathscr{P}$: then $\langle B, \pi\rangle$ is the pp of X. Otherwise, $\bigsqcup \Pi(X)$ does not exist in $\mathscr{P}$, and then X has an infinite set of pp's, as shown in the following:

Theorem 3.5. (i) *Suppose $\mathscr{A}(X)$ is finite. $\langle B, \pi\rangle \sim \bigsqcup \Pi(X)$ is such that if $\langle B', \tau\rangle$ is suitable for X, then there exists a (lifting) chain c such that $\langle B', \tau\rangle \sim c(\langle B, \pi\rangle)$.*

(ii) *Suppose $\mathscr{A}(X)$ is infinite. For every $\langle B', \tau\rangle$ suitable for X there exists $\langle B, \pi\rangle \in \Pi(X)$ such that $\langle B', \tau\rangle \sim c(\langle B, \pi\rangle)$, for some (lifting) chain c.*

Then, if a term X has a finite set of approximants, it has a principal pair, which is $\bigsqcup \Pi(X)$. The problem of finding, given a term X, $\bigsqcup \Pi(X)$, if it exists, is semidecidable since the construction of the set $\mathscr{A}(X)$, for a given term X, is semidecidable (see [1]).

The following property gives a syntactical characterization of the elements of $\mathscr{P}$.

Property 3.6 (Ronchi [12]).

(i) $\langle B, \pi\rangle \in \mathscr{P}, \pi \neq \omega \Leftrightarrow \langle B, \pi\rangle \sim \langle \{\tau_1 x_1, \ldots, \tau_n x_n\}, \tau_{n+1} \to \ldots \to \tau_{n+m} \to \varphi\rangle$

($n \geq 0, m \geq 0$), *where φ is a type variable, and*

$$\langle B', \varphi\rangle \sim \langle B \cup \{\tau_{n+1} x_{n+1}, \ldots, \tau_{n+m} x_{n+m}\}, \varphi\rangle \in \mathscr{P},$$

$$\exists h (1 \leq h \leq m+n), \exists k (k \geq 0). \quad B' \sim \bigcup_{1 \leq j \leq k} B_j \cup \{\pi_1 \to \cdots \to \pi_k \to \varphi x_h\},$$

where $\langle B_j, \pi_j\rangle \in \mathscr{P}$ $(1 \leq j \leq k)$, and they are pairwise disjoint.

(ii) *Let $\langle B, \pi\rangle$ be as in* (i). *Then $\langle B, \pi\rangle \sim \mathrm{pp}(A)$, where $A \equiv \lambda x_1 \ldots x_m. x_h A'_1 \ldots A'_k$ and $\mathrm{pp}(A'_j) \sim \langle B_j, \pi_j\rangle$.*

Definition 3.7. Let $\langle B_1, \pi_1\rangle$, and $\langle B_2, \pi_2\rangle$ be elements of $\mathscr{P}$, and let them be disjoint. A *closure* of $\langle B_1, \pi_1\rangle$ and $\langle B_2, \pi_2\rangle$ is a pair of type schemes $\langle \sigma, \tau\rangle$ such that:

- $\sigma \equiv \sigma_1 \to \cdots \to \sigma_n \to \pi_1$,
- $\tau \equiv \tau_1 \to \cdots \to \tau_n \to \pi_2 \to \varphi$,

where φ is a fresh type variable and if $x_1, \ldots, x_n$ is the sequence of subjects belonging either to B_1 or to B_2, in any order, and if α_i (β_i) is the predicate of x_i in B_1 (B_2), if it exists and ω otherwise, then $\sigma_i \equiv \alpha_i \wedge \psi_i$ and $\tau_i \equiv \eta_i \wedge \beta_i$, and ψ_i and η_i are fresh type variables.

4. The unification procedure

In the Curry type discipline, every term X possessing a type scheme has a principal pair for which all and only the pairs suitable for X are reached by means of substitutions. It is decidable if a term possesses a principal pair, and the algorithm computing it uses the unification algorithm of Robinson, which, given two objects, finds the most general substitution making them syntactically identical.

The aim of this paper is to solve the semidecidable problem of finding (if it exists), given a term X, its principal pair $\bigsqcup \Pi(X)$ in a similar way, i.e., through unification of type schemes. Clearly, unification, in the intersection type discipline, must be defined taking into account all the operations preserving suitable pairs.

A first formulation of the unification problem is the following: *Given* $\sigma, \tau \in T$, *find, if it exists, a lifting chain* c *such that* $c(\sigma) \equiv c(\tau)$. But in this formulation of the problem, the particular role of the universal type scheme ω is not taken into account. In fact, the semantics of ω naturally induces the property that ω can be unified with any type scheme. In order to realize this, a new equivalence relation between type schemes is needed:

Definition 4.1. (i) An ω-type scheme is a type scheme in which only the symbols ω, $\to$ and $\wedge$ occur.

(ii) $\approx$ is inductively defined as follows:

$$\alpha, \beta \text{ are } \omega \text{ type scheme} \Rightarrow \alpha \approx \beta,$$

$$\alpha \equiv \beta \Rightarrow \alpha \approx \beta,$$

$$\alpha \approx \alpha', \beta \approx \beta' \Rightarrow \alpha \to \beta \approx \alpha' \to \beta', \alpha \wedge \beta \approx \alpha' \wedge \beta'.$$

Example 4.2. $\omega \approx \omega \to \omega$, $\omega \approx \omega \wedge \omega$, $(\omega \to \omega) \to \alpha \approx \omega \to \alpha$.

Note that if α is an ω-type scheme, then $\alpha \sim \omega$ (but the inverse it is not true). Moreover, the fact that, in a principal deduction, applications of the rule ($\leqslant$) do not occur suggests that lifting is not necessary in unification.

So a new formulation of the problem is the following: *Given* $\sigma, \tau \in T$, *find, if it exists, a chain* c *such that* $c(\sigma) \approx c(\tau)$. But now the problem always has a solution, the trivial one $c(\sigma) \approx c(\tau) \approx \omega$. The correct formulation of the unification problem must contain the condition that the trivial solution can be chosen only in the case no other solution exists. To impose this constraint, a further definition is needed.

Definition 4.3. (i) Let $\sigma, \tau \in T$, and let σ', τ' be subtypes respectively of σ and τ. Two occurrences of σ' and τ' in σ and τ are *corresponding* iff

- either $\sigma \equiv \sigma'$ and $\tau \equiv \tau'$,
- or $\sigma \equiv \alpha \to \beta$ and $\tau \equiv \alpha' \to \beta'$ ($\sigma \equiv \alpha \wedge \beta$ and $\tau \equiv \alpha' \wedge \beta'$) and the occurrences of σ' and τ' are corresponding either in α and α' or in β and β'.

(ii) Let $c \equiv \mathrm{op}_1 \dots \mathrm{op}_n$ be a chain such that $c(\sigma) \approx c(\tau)$. c is a *proper chain* unifying σ and τ iff $\forall i (1 \leq i \leq n)$ there exist no two corresponding occurrences of subtypes of $\mathrm{op}_1 \dots \mathrm{op}_i(\sigma)$ and $\mathrm{op}_1 \dots \mathrm{op}_i(\tau)$ (say σ_i and τ_i) such that $\sigma_i, \tau_i \not\approx \omega$ and $\exists j > i$ s.t. an occurrence of $\mathrm{op}_{i+1} \dots \mathrm{op}_j(\sigma_i)$ is corresponding to an occurrence of $\mathrm{op}_{i+1} \dots \mathrm{op}_j(\tau_i)$ and

$$\mathrm{op}_{i+1} \dots \mathrm{op}_j(\sigma_i) \approx \mathrm{op}_{i+1} \dots \mathrm{op}_j(\tau_i) \approx \omega.$$

Roughly speaking, a proper chain unifying two given type schemes is a chain in which a substitution of a type variable with the constant ω is used only in order to unify two subtypes one of which is ω. Then the final formulation of the problem is: *Given* $\sigma, \tau \in T$, *find, if it exists, a proper chain* c *such that* $c(\sigma) \approx c(\tau)$. This problem is semidecidable. In fact, in the following section it will be possible to see that it is equivalent to the semidecidable problem of finding, given a term X, its principal pair if X is strongly normalizing.

Now the procedure UNIFY solving this problem will be shown.

Procedure UNIFY

UNIFY$(\sigma, \tau) = c$, where $\langle c, n \rangle = U(\sigma, \sigma, \tau, \tau, 0)$, and $U(\sigma, \sigma', \tau, \tau', m) = \langle c', p \rangle$ (if defined) (c' is a chain and n, m, p are integers belonging to $\{0, 1\}$), where:

(1) **if** σ is an intersection **and** τ is not an intersection **then let** $A = \{\sigma', \tau'\}$;

if $\langle c_1, p_1 \rangle = U(e_{A,\tau}(\sigma'), e_{A,\tau}(\sigma'), e_{A,\tau}(\tau'), e_{A,\tau}(\tau'), m)$

then $\langle c', p \rangle = \langle e_{A,\tau}.c_1, 1 \rangle$.

(2) **if** τ is an intersection **and** σ is not an intersection **then let** $A = \{\sigma', \tau'\}$;

if $\langle c_1, p_1 \rangle = U(e_{A,\sigma}(\sigma'), e_{A,\sigma}(\sigma'), e_{A,\sigma}(\tau'), e_{A,\sigma}(\tau'), m)$

then $\langle c', p \rangle = \langle e_{A,\sigma}.c_1, 1 \rangle$.

(3) **if** σ is a type variable **then** $\langle c', p \rangle = \langle s_f, m \rangle$, **where**

if $\sigma \equiv \tau$ **then** $f = \bot$

else if σ occurs in τ **then** $f = [\varphi \Rightarrow \omega \mid \varphi$ occurs in $\tau]$ **else** $f = [\sigma \Rightarrow \tau]$.

(4) **if** $\sigma \equiv \omega$ **then** $\langle c', p \rangle = \langle s_f, m \rangle$, **where** $f = [\varphi \Rightarrow \omega \mid \varphi$ occurs in $\tau]$.

(5) **if** $\sigma \equiv \alpha \to \beta$ **then**

(5.1) **if** τ **is a type variable then** $\langle c', p\rangle = \langle s_f, m\rangle$, **where**

if τ **occurs in** σ **then** $f = [\varphi \Rightarrow \omega \mid \varphi$ **occurs in** $\sigma]$

else $f = [\tau \Rightarrow \sigma]$.

(5.2) **if** $\tau \equiv \omega$ **then** $\langle c', p\rangle = \langle s_f, m\rangle$, **where** $f = [\varphi \Rightarrow \omega \mid \varphi$ **occurs in** $\sigma]$.

(5.3) **if** $\tau \equiv \gamma \to \delta$

then if $\langle c_1, p_1\rangle = U(\alpha, \sigma', \gamma, \tau', 0)$

then (**if** $p_1 = 0$

then (**if** $\langle c_2, p_2\rangle = U(c_1(\beta), c_1(\sigma'), c_1(\delta), c_1(\tau'), m)$

then $\langle c', p\rangle = \langle c_1.c_2, p_2\rangle$)

else $\langle c', p\rangle = \langle c_1, m\rangle$).

(6) **if** $\sigma \equiv \alpha \wedge \beta$ **and** $\tau \equiv \gamma \wedge \delta$

then if $\langle c_1, p_1\rangle = U(\alpha, \sigma', \gamma, \tau', m)$ **and** $\langle c_2, p_2\rangle$

$= U(c_1(\beta), c_1(\sigma'), c_1(\delta), c_1(\tau'), p_1)$

then $\langle c', p\rangle = \langle c_1.c_2, \max(p_1, p_2)\rangle$.

Example 4.4. (i) Let α, β, γ, δ be type variables, and let $\sigma \equiv \alpha \to \beta$ and $\tau \equiv \gamma \wedge \delta$. Then UNIFY$(\sigma, \tau) = U(\sigma, \sigma, \tau, \tau, 0)$ which performs

$$U(e_{A,\sigma}(\sigma), e_{A,\sigma}(\sigma), e_{A,\sigma}(\tau), e_{A,\sigma}(\tau), 0),$$

where $A = \{\sigma, \tau\}$. $e_{A,\sigma}(\sigma) = (\alpha' \to \beta') \wedge (\alpha'' \to \beta'')$, where $\alpha', \alpha'', \beta', \beta''$ are type variables, and $e_{A,\sigma}(\tau) = \tau$. Then the semi-algorithm performs

$$U(\alpha' \to \beta', e_{A,\sigma}(\sigma), \gamma, \tau, 0) = \langle s_f, 0\rangle,$$

where $f = [\gamma \Rightarrow \alpha' \to \beta']$ and

$$U(\alpha'' \to \beta'', e_{A,\sigma}(\sigma), \delta, \tau, 0) = \langle s_g, 0\rangle,$$

where $g = [\delta \Rightarrow \alpha'' \to \beta'']$. Then

$$U(e_{A,\sigma}(\sigma), e_{A,\sigma}(\sigma), e_{A,\sigma}(\tau), e_{A,\sigma}(\tau), 0) = \langle c, 1\rangle,$$

where $c = e_{A,\sigma}.s_f.s_g$ and UNIFY$(\sigma, \tau) = c$. $c(\sigma) \approx c(\tau) \approx (\alpha' \to \beta') \wedge (\alpha'' \to \beta'')$.

(ii) Let $\sigma \equiv \omega$ and $\tau \equiv \omega \to \omega$ $(\sigma \approx \tau)$. It is easy to verify that UNIFY$(\sigma, \tau) = \perp$, where $\perp$ is the everywhere undefined substitution.

(iii) Let α, β, γ, δ be type variables, and let

$$\sigma \equiv \alpha \wedge (\alpha \to \beta) \quad \text{and} \quad \tau \equiv (\gamma \wedge (\gamma \to \delta)) \to \delta.$$

Then UNIFY(σ, τ) does not stop.

Remark 4.5. Let σ and τ have corresponding occurrences respectively in σ' and τ'. For any substitution s, $s(\sigma)$ and $s(\tau)$ have corresponding occurrences respectively in $s(\sigma')$ and $s(\tau')$. On the contrary, if e is any expansion, it is possible that $e(\sigma)$ is not a subtype of $e(\sigma')$ ($e(\tau)$ is not a subtype of $e(\tau')$) (see the definition of expansion).

Theorem 4.6 (Correctness). *If* UNIFY$(\sigma, \tau) = c$, *then* $c(\sigma) \approx c(\tau)$, *and either* c *is proper or there is no proper chain unifying* σ *and* τ.

Proof. It is necessary to prove the stronger assumption: *if* σ *and* τ *are subtypes respectively of* σ' *and* τ', *and two of their occurrences are corresponding* (*see Definition* 4.3), *and* $U(\sigma, \sigma', \tau, \tau', 0) = \langle c, p \rangle$, *then*:

- *if* $p = 0$, *then* $c(\sigma)$ *and* $c(\tau)$ *have corresponding occurrences respectively in* $c(\sigma')$ *and* $c(\tau')$, *and* $c(\sigma) \approx c(\tau)$;
- *if* $p = 1$, *then* $c(\sigma') \approx c(\tau')$.

Then the proof of the theorem follows immediately since σ and τ are corresponding in σ and τ.

The proof will be given by induction on the triple $\langle E(c), n(\sigma, \tau), s(\sigma, \tau) \rangle$, where $E(c)$ is defined in Definition 2.13 and $n(\sigma, \tau)$ and $s(\sigma, \tau)$ are respectively the total number of type variables and the total number of symbols occurring in σ and τ.

Let $E(c) = 0$, and let σ be a type variable. Then, if $\sigma \equiv \tau$, then $U(\sigma, \sigma', \tau, \tau', 0) = \langle s_\perp, 0 \rangle$ and the proof is trivial.

Else, in the case σ does not occur in τ, $U(\sigma, \sigma', \tau, \tau', 0) = \langle s_f, 0 \rangle$, where $f = [\sigma \Rightarrow \tau]$, and so $s_f(\sigma) \equiv \tau \equiv s_f(\tau)$. In the case σ occurs in τ, obviously, there is no proper chain unifying σ and τ, and so the unique way to unify them is to collapse both of them in type schemes $\approx \omega$; in fact, $U(\sigma, \sigma', \tau, \tau', 0) = \langle s_f, 0 \rangle$, where $f = [\varphi \Rightarrow \omega]$, and $s_f(\sigma) \approx s_f(\tau) \approx \omega$. Moreover, by Remark 4.5, $s(\sigma)$ and $s(\tau)$ are corresponding in $s(\sigma')$ and $s(\tau')$. The case where either σ or τ equals ω is obvious.

The case of either σ or τ being an intersection type scheme is not possible since in this case $E(c) > 0$.

Let $\sigma \equiv \alpha \to \beta$ and $\tau \equiv \gamma \to \delta$. Then, to compute $U(\sigma, \sigma', \tau, \tau', 0)$, the procedure computes $U(\alpha, \sigma', \gamma, \tau', 0)$. Let $U(\alpha, \sigma', \gamma, \tau', 0) = \langle c_1, p \rangle$, and let $p = 0$; since the integer parameter of U is changed to 1 only when an expansion is applied, this is the only possible case: $E(c_1) = E(c) = 0$. Since either $n(\alpha, \gamma) < n(\sigma, \tau)$ or $s(\alpha, \gamma) < s(\sigma, \tau)$, by induction, $c_1(\alpha)$ and $c_1(\gamma)$ have corresponding occurrences in $c_1(\sigma')$, $c_1(\tau')$ and $c_1(\alpha) \approx c_1(\gamma)$. Moreover, since c_1 is a substitution, either the total number of type variables occurring in $c_1(\beta)$ and $c_1(\delta)$ is less than $n(\sigma, \tau)$, or $c_1(\beta) \equiv \beta$ and $c_1(\delta) \equiv \delta$ and so $s(\beta, \delta) < s(\sigma, \tau)$; in both cases it is possible to apply the induction hypothesis to

$$U(c_1(\beta), c_1(\sigma'), c_1(\delta), c_1(\tau'), 0) = \langle c_2, p' \rangle.$$

If $p' = 0$, then $c_1.c_2(\beta) \approx c_1.c_2(\delta)$, and $c_1.c_2(\beta)$ and $c_1.c_2(\delta)$ are corresponding in $c_1.c_2(\sigma')$ and $c_1.c_2(\tau')$, so $c_1.c_2(\alpha \to \beta) \approx c_1.c_2(\gamma \to \delta)$. The case $p' = 1$ is not possible.

The case $\sigma \equiv \alpha \wedge \beta$ and $\tau \equiv \gamma \wedge \delta$ directly follows from the induction.

Let $E(c) > 0$. Let τ be an intersection and σ not so Then $U(\sigma, \sigma', \tau, \tau', 0) = \langle c, 1 \rangle$, and $c = e_{A,\sigma}.c'$, where

$$\langle c', p \rangle = U(e_{A,\sigma}(\sigma'), e_{A,\sigma}(\sigma'), e_{A,\sigma}(\tau'), e_{A,\sigma}(\tau'), 0).$$

In the case $E(c') = 0$, by the first step, $c'(e_{A,\sigma}(\sigma')) \approx c'(e_{A,\sigma}(\tau'))$, and the proof is completed. If $E(c') < E(c)$, the proof follows by induction.

The case $\sigma \equiv \alpha \wedge \beta$ and $\tau \equiv \gamma \wedge \delta$ directly follows from the induction since, in this case, for every expansion e, $e(\sigma)$ and $e(\tau)$ are corresponding in $e(\sigma')$ and $e(\tau')$.

Let $\sigma \equiv \alpha \to \beta$ and $\tau \equiv \gamma \to \delta$. Let $U(\alpha, \sigma', \gamma, \tau') = \langle c_1, p \rangle$. In the case $p = 1$, this means c_1 contains at least an expansion. If $E(c_1) = E(c)$, since either $n(\alpha, \gamma) < n(\beta, \delta)$ or $s(\alpha, \gamma) < s(\beta, \delta)$, by induction,

$$c_1(\sigma') \approx c_1(\tau') \approx c(\sigma') \approx c(\tau').$$

If $E(c_1) < E(c)$, by induction, $c_1(\sigma') \approx c_1(\tau')$; then it follows that this case is not possible since $c_1 = c$. In the case $p = 0$, the procedure computes $U(c_1(\beta), c_1(\sigma'), c_1(\delta), c_1(\tau'), 0)$; let the result be $\langle c_2, p' \rangle$ with $E(c) = E(c_2)$. By induction, $c_1.c_2(\sigma') \approx c_1.c_2(\tau')$. □

Moreover, the procedure UNIFY is conservative with respect to Robinson's unification algorithm R. More precisely, we have the following property.

Property 4.7. *Let σ, τ be type schemes without occurrences of the symbols $\wedge$ and ω. If $R(\sigma, \tau) = s$, where s is some substitution, then* UNIFY$(\sigma, \tau) = s$; *if $R(\sigma, \tau)$ fails, then* UNIFY$(\sigma, \tau) = s'$, *where s' is a substitution such that $s'(\sigma) \approx s'(\tau) \approx \omega$.*

Proof. Easy. □

The following theorem (whose proof, very technical, is in the Appendix) proves that the procedure UNIFY is complete when it is applied to two type schemes which are the component of a closure of two disjoint pp's, and that in this case UNIFY finds the most general unifying chain. This property permits to use UNIFY in the design of a procedure for finding the principal term of a term, if it exists, as will be seen in the following section.

Theorem 4.8 (Completeness). *Let $\langle \pi, \sigma \rangle$ be a closure of two disjoint* pp*'s. If there exists a proper chain c such that $c(\sigma) \approx c(\pi)$, then* UNIFY$(\pi, \sigma) = c'$, *c' is proper and $c = c'.c''$ for some c''.*

Proof. See Appendix. □

5. The procedure building the principal pair for a term

Now it is possible to use the procedure UNIFY in order to construct the principal pair of a term X, if it exists. The procedure PP solves this problem, as it will be proved in the sequel. In this procedure, the operation ∇ between basis schemes, with at least one statement on every subject, is used. ∇ is defined as follows:

$$B \nabla B' = \{\sigma \wedge \sigma' x \mid \sigma x \in B \text{ and } \sigma' x \in B'\}$$
$$\cup \{\sigma x \mid (\sigma x \in B \text{ and } B' \text{ has no statement on } x) \text{ or } (\sigma x \in B' \text{ and } B \text{ has no statement on } x)\}.$$

Procedure PP

$\mathrm{PP}(X) = \langle B, \pi\rangle$ (if defined), where

(1) if X is a variable then $\langle B, \pi\rangle = \langle\{\varphi X\}, \varphi\rangle$ where φ is a fresh type variable.

(2) if $X = \lambda x.X'$

then if $\mathrm{PP}(X') = \langle B', \pi'\rangle$

then if B' contains a predicate whose subject is x, say σx,

then $\langle B, \pi\rangle = \langle B' - \{\sigma x\}, \sigma \to \pi'\rangle$ else $\langle B', \omega \to \pi'\rangle$.

(3) if $X = X_1 X_2$

then if $\mathrm{PP}(X_1) = \langle B_1, \pi_1\rangle$ and $\mathrm{PP}(X_2) = \langle B_2, \pi_2\rangle$

then let $\langle \sigma \equiv \sigma_1 \to \cdots \to \sigma_n \to \pi_1,\ \tau \equiv \tau_1 \to \cdots \to \tau_n \to \pi_2 \to \varphi\rangle$

be a closure of $\langle B_1, \pi_1\rangle$ and $\langle B_2, \pi_2\rangle$;

if $\mathrm{UNIFY}(\sigma, \tau) = c$

then $\langle B, \pi\rangle = \langle H(s_k(c(B_1) \nabla c(B_2))), H(s_k(c(\varphi)))\rangle$, where

$$k = [\varphi \Rightarrow \omega \mid \exists \sigma.\sigma \text{ occurs in } \langle c(B_1) \nabla c(B_2), c(\varphi)\rangle \text{ and } H(\sigma) \equiv \omega \text{ and } \varphi \text{ occurs in } \sigma]$$

and H is the function defined thus: $H(\sigma) = \sigma'$, where

(1) if σ is either a type variable or ω then $\sigma' \equiv \sigma$.

(2) if $\sigma \equiv \sigma_1 \to \sigma_2$ then if $H(\sigma_2) \equiv \omega$ then $\sigma' \equiv \omega$ else $\sigma' \equiv H(\sigma_1) \to H(\sigma_2)$.

(3) if $\sigma \equiv \sigma_1 \wedge \sigma_2$ then if $H(\sigma_1) \equiv H(\sigma_2) \equiv \omega$ then $\sigma' \equiv \omega$

else if $H(\sigma_1) \equiv \omega$ then $\sigma' \equiv H(\sigma_2)$

else if $H(\sigma_2) \equiv \omega$ then $\sigma' \equiv H(\sigma_1)$

else $\sigma' \equiv H(\sigma_1) \wedge H(\sigma_2)$.

Procedure PP needs some comments. Consider $\mathrm{PP}(X)$, in the case X is an application $X_1 X_2$. If both X_1 and X_2 are closed terms and if $\mathrm{PP}(X_1) = \langle \Phi, \pi_1\rangle$ and $\mathrm{PP}(X_2) = \langle \Phi, \pi_2\rangle$, then $\mathrm{PP}(X_1 X_2)$ computes $\mathrm{UNIFY}(\pi_1, \pi_2 \to \varphi)$, where φ is fresh, and if $\mathrm{UNIFY}(\pi_1, \pi_2 \to \varphi) = c$, then $\mathrm{PP}(X_1 X_2) = \langle \Phi, H(s_k(c(\varphi)))\rangle$. So, in this case, PP can be obtained from the procedure computing the principal type scheme for Curry's type discipline simply by replacing Robinson's unification procedure with UNIFY and then by applying s_k and H to the resulting type scheme in order to transform the deduction of $c(\varphi)$ into a new deduction by replacing every subdeduction of a type scheme $\sim\omega$ with a single application of the rule (ω).

Consider now the case X_1 and X_2 are not both closed, and let $\mathrm{PP}(X_1) = \langle B_1, \pi_1\rangle$ and $\mathrm{PP}(X_2) = \langle B_2, \pi_2\rangle$. For any type variable x, if $\mu_1 x \in B_1$ and $\mu_2 x \in B_2$, it is not necessary to unify μ_1 and μ_2 since in the principal basis of X the predicate of x is the intersection of the predicates on x used for the subterms of X. On the other side, it is necessary to unify π_1 and $\pi_2 \to \varphi$ regarding them as subtypes of two type schemes containing all the statements respectively of B_1 and B_2 in order to define correctly the operations of expansions, which are context-dependent. Then closures are introduced (see Definition 3.7).

By definition, if $\langle \sigma_1 \to \cdots \to \sigma_n \to \pi_1,\ \tau_1 \to \cdots \to \tau_n \to \pi_2 \to \varphi\rangle$ is a closure of $\langle B_1, \pi_1\rangle$ and $\langle B_2, \pi_2\rangle$, then $\sigma_i \equiv \sigma_i' \wedge \varphi_i$ and $\tau_i \equiv \psi_i \wedge \tau_i'$, where $\sigma_i' x \in B_1$ and $\tau_i' x \in B_2$ for some

x, and φ_i and ψ_i are fresh ($1 \leq i \leq n$). So σ_i and τ_i can always be unified and if $\text{UNIFY}(\sigma_i, \tau_i) = c_i$, then $c_i(\sigma_i) \equiv c_i(\tau_i) \equiv \sigma_i' \wedge \tau_i'$. So the chain c built in PP (if any) is such that $c \equiv c_1 \ldots c_n.c'$, and $c(B_1) \nabla c(B_2) \equiv c'(B_1) \nabla c'(B_2)$ and $c(\varphi) \equiv c'(\varphi)$ since the fresh variables introduced by the closure do not occur in B_1, B_2, φ.

Example 5.1. Let $X_1 = \lambda x.xx$ and $X_2 = xy$. So $\langle B_1, \pi_1 \rangle \equiv \langle \Phi, (\mu \wedge (\mu \to \nu)) \to \nu \rangle$ and $\langle B_2, \pi_2 \rangle \equiv \langle \{\alpha \to \beta x, \alpha y\}, \beta \rangle$, where μ, ν, α, β are type variables.

$$\langle \sigma, \tau \rangle \equiv \langle (\omega \wedge \varphi_1) \to (\omega \wedge \varphi_2) \to (\mu \wedge (\mu \to \nu)) \to \nu, (\psi_1 \wedge (\alpha \to \beta)) \to (\psi_2 \wedge \alpha) \to \beta \to \varphi \rangle$$

is a closure of $\langle B_1, \pi_1 \rangle$ and $\langle B_2, \pi_2 \rangle$. $\text{UNIFY}(\sigma, \tau) = c_1.e.c_2$, where e is the expansion necessary in order to unify $c_1(\beta)$ and $c_1(\mu \wedge (\mu \to \nu))$. So $e = e_{A, c_1(\beta)}$, where $A = \{c_1(\sigma), c_1(\tau)\}$ and $c_1(\alpha \to \beta) \in !A$. So the result is

$$\text{PP}(X) = \langle \{(\alpha_1 \to \mu) \wedge (\alpha_2 \to \mu \to \nu)x, \alpha_1 \wedge \alpha_2 y\}, \nu \rangle$$

where α_1, α_2 are type variables, and $\text{PP}(X) \sim \text{pp}((\lambda x.xx)(xy)) \sim \text{pp}(xy(xy))$.

However, $\text{UNIFY}((\mu \wedge (\mu \to \nu)) \to \nu, \beta \to \varphi) = c_3.e.c_4$, where $e = e_{A', c_3(\beta)}$, where $A' = \{c_3(\mu \wedge (\mu \to \nu)), c_3(\beta \to \varphi)\}$ and $c_3(\alpha \to \beta) \notin !A'$. So without using the closures the result would be incorrect.

Remember that a term X is called *strongly normalizing* iff X, and every subterm of it, possess a normal form.

Theorem 5.2. $\text{PP}(X) = \langle B, \pi \rangle \Leftrightarrow X$ *is strongly normalizing and* $\text{pp}(X) \sim \langle B, \pi \rangle$.

The proof will be given at the end of this section.

Remark 5.3. $\text{PP}(X)$ is one of the shortest type schemes between all the type schemes $\sim\text{pp}(X)$. This is obtained by means of the function H. It would be possible not to use functions H and s_k in procedure PP by defining that every type scheme $\sim\omega$ is an ω-type scheme, and by modifying consequently the procedure UNIFY. But in this case, both UNIFY and PP (and especially procedure PP' defined in the sequel) would become less efficient.

It is possible to define a set of unification algorithms UNIFY_i ($i \geq 0$), each one unifying at every step, with ω, all the subtypes occurring at depth $\geq i$, where the depth of an occurrence of a subtype in a type scheme is defined as follows.

Definition 5.4. Let $\sigma \in T$. The *depth* $d(o(\tau), \sigma)$ of an occurrence $o(\tau)$ of τ in σ is:

(i) if τ does not occur in σ, then $d(o(\tau), \sigma)$ is undefined;

(ii) if $\sigma = \tau$, then $d(o(\tau), \sigma) = 0$;

(iii) if either $\sigma = \sigma_1 \to \sigma_2$ or $\sigma = \sigma_1 \wedge \sigma_2$, then
- if $d(o(\tau), \sigma_1) = i$, then $d(o(\tau), \sigma) = i + 1$,
- if $d(o(\tau), \sigma_2) = i$, then $d(o(\tau), \sigma) = i + 1$.

Algorithms UNIFY$_i$

UNIFY$_i(\sigma, \tau) = U_i(\sigma, \sigma, \tau, \tau, 0, 0)$ where

$U_i(\sigma, \sigma', \tau, \tau', m, j) = \langle c, p\rangle$ where

if $j \geq i$ then $\langle c, p\rangle = U(\omega, \sigma', \omega, \tau', m)$ else

(1) if σ is an intersection and τ is not an intersection
then let $A = \{\sigma', \tau'\}$ and let

$$\langle c_1, p_1\rangle = U_i(e_{A,\tau}(\sigma'), e_{A,\tau}(\sigma'), e_{A,\tau}(\tau'), e_{A,\tau}(\tau'), m, 0)$$

then $\langle c, p\rangle = \langle e_{A,\tau}.c_1, 1\rangle$

(2) if τ is an intersection and σ is not an intersection
then let $A = \{\sigma', \tau'\}$ and let

$$\langle c_1, p_1\rangle = U_i(e_{A,\sigma}(\sigma'), e_{A,\sigma}(\sigma'), e_{A,\sigma}(\tau'), e_{A,\sigma}(\tau'), m, 0)$$

then $\langle c, p\rangle = \langle e_{A,\sigma}.c_1, 1\rangle$

(3) if σ is either a type variable or ω then $\langle c, p\rangle = U(\sigma, \sigma', \tau, \tau', m)$

(4) if $\sigma = \alpha \to \beta$ then

(4.1) if τ is either a type variable or ω then $\langle c, p\rangle = U(\sigma, \sigma', \tau, \tau', m)$

(4.2) if $\tau = \gamma \to \delta$ then let $\langle c_1, p_1\rangle = U_{i-1}(\alpha, \sigma', \gamma, \tau', m, j+1)$;
if $p_1 = 0$ then let

$$\langle c_2, p_2\rangle = U_{i-1}(c_1(\beta), c_1(\sigma'), c_1(\delta), c_1(\tau'), m, j+1); \qquad \langle c, p\rangle = \langle c_1.c_2, p_2\rangle$$

else $\langle c, p\rangle = \langle c_1, p_1\rangle$

(5) if $\sigma = \alpha \wedge \beta$ then if $\tau = \gamma \wedge \delta$ then let

$$\langle c_1, p_1\rangle = U_{i-1}(\alpha, \sigma', \gamma, \tau', m, j+1) \quad \text{and}$$

$$c_2 = U_{i-1}(c_1(\beta), c_1(\sigma'), c_1(\delta), c_1(\tau'), p_1, j+1)$$

then $\langle c, p\rangle = \langle c_1.c_2, \max(p_1, p_2)\rangle$.

Let PP$_i$ be the algorithm obtained from PP by replacing UNIFY with UNIFY$_i$ ($i \geq 0$). The following theorem holds.

Theorem 5.5. (i) $\mathrm{PP}_i(X) = \langle B, \pi\rangle \Rightarrow \langle B, \pi\rangle \in \Pi(X)$.
(ii) $\langle B, \pi\rangle \in \Pi(X) \Rightarrow \exists i.\mathrm{PP}_i(X) = \langle B_i, \pi_i\rangle$ *and* $\langle B, \pi\rangle \sqsubseteq_\omega \langle B_i, \pi_i\rangle$.

Proof. Immediate from Theorem 5.2 and from the definition of the approximants of a term. □

Proof of Theorem 5.2. ($\Leftarrow$): By induction on the structure of X. For X a variable, it is obvious. For $X = \lambda x.X'$, the proof directly follows from the induction hypothesis.

Let $X = YZ$. By induction, $\mathrm{PP}(Y) \equiv \langle B_1, \pi_1\rangle \sim \mathrm{pp}(Y)$ and $\mathrm{PP}(Z) \equiv \langle B_2, \pi_2\rangle \sim \mathrm{pp}(Z)$. Then $B_1 \vdash \pi_1 Y$ and $B_2 \vdash \pi_2 Z$. YZ being strongly normalizing implies $\exists B, \sigma$.

$B \vdash \sigma YZ$, and in B, σ there are no occurrences of ω (see Theorem 2.5). It is easy to see that if A is any normal form, then every deduction of a type scheme for A can be transformed into an equivalent deduction in which the rule $(\leq)$ is applied only in the last step: so $\exists D.B' \vdash \rho YZ$, $\rho \leq \sigma$ and D does not contain an application of the rule $(\leq)$. Moreover, by [2, Lemma 2.8], this implies $B' \vdash \tau \to \rho Y$ and $B' \vdash \tau Z$, and these deductions do not use the rule $(\leq)$. By Theorem 3.5, this implies the existence of two chains c and c' such that

$$c(\langle B_1, \pi_1 \rangle) \approx \langle B'/Y, \tau \to \rho \rangle \quad \text{and} \quad c'(\langle B_2, \pi_2 \rangle) \approx \langle B'/Z, \tau \rangle,$$

where B/Q denote the basis B restricted to the free variables of Q. This means that for any closure of $\langle B_1, \pi_1 \rangle$ and $\langle B_2, \pi_2 \rangle$, $c.c'$ is a proper chain unifying it. Let $\langle \sigma, \tau \rangle$ be a closure of $\langle B_1, \pi_1 \rangle$ and $\langle B_2, \pi_2 \rangle$. So by Theorem 4.8, $\text{UNIFY}(\sigma, \tau) = c''$ and $c.c' = c''.\underline{c}$ for some $\underline{c}$. By the fact that c'' is the minimal unifying chain, it follows that $c''(\langle B_1 \nabla B_2, \varphi \rangle)$ is a pair such that $\exists D.c''(B_1 \nabla B_2) \vdash c''(\varphi) YZ$, and D differs from a principal deduction only in subdeductions of type schemes $\sim\omega$ for subterms of YZ (since UNIFY makes the unification with respect to the equivalence $\approx$). Then it is easy to see that the application of the substitution s_k transforms into ω every type scheme used in a deduction of a type scheme $\sim\omega$ for a subterm of YZ, while H transforms every such subdeduction into a single application of the rule (ω).

$(\Rightarrow)$: Let B be a basis scheme, α a type scheme $\not\sim\omega$, and M a λ-term. Let us define the predicate: $P(B, \alpha, M) \Leftrightarrow \text{PP}(M)$ is defined and $\exists c.\langle B, \alpha \rangle \sim c(\text{pp}(M))$ and $\text{pp}(M) \sim \text{PP}(M)$ and M is strongly normalizing. Let xM denote $xM_1 \dots M_n$ for $n \geq 0$, and let $\text{FV}(M)$ denote the set of variables occurring free in M.

Property 5.6. (i) *$P(B, \alpha \to \beta, xM)$ and $P(B', \alpha, N)$ implies $P(B \cup B', \beta, xMN)$.*

(ii) *$P(B \cup \{\alpha x\}, \beta, Mx)$ and $x \in \text{FV}(M)$ and B does not contain premises on x implies $P(B, \alpha \to \beta, M)$.*

(iii) *$P(B, \sigma_1 \wedge \sigma_2, M)$ implies $P(B, \sigma_1, M)$ and $P(B, \sigma_2, M)$.*

(iv) *$P(B, \sigma, M)$ and $\sigma \leq \tau$ implies $P(B, \tau, M)$.*

Proof. (i): Let $xM = xM_1 \dots M_m$. $P(B', \alpha, N) \Rightarrow \langle \underline{B}', \pi' \rangle = \text{PP}(N) \sim \text{pp}(N)$ and $\exists c.\langle B', \alpha \rangle \sim c(\text{pp}(N))$, and N is strongly normalizing.

$$P(B, \alpha \to \beta, xM) \Rightarrow \text{PP}(xM) \sim \text{pp}(XM)$$

$$\sim \langle \{\pi_1 \to \cdots \to \pi_m \to \varphi x\} \cup B_1 \cup \cdots \cup B_m, \varphi \rangle,$$

where φ is a fresh variable and $\text{pp}(M_i) \sim \langle B_i, \pi_i \rangle \sim \text{PP}(M_i)$ and $\exists c'.\langle B, \alpha \to \beta \rangle \sim c'(\text{pp}(xM))$ and xM is strongly normalizing. If

$$\langle \sigma \equiv \sigma_1 \to \cdots \to \sigma_n \to \varphi, \tau \equiv \tau_1 \to \cdots \to \tau_n \to \pi' \to \psi \rangle$$

is a closure of $\text{pp}(xM)$ and $\text{pp}(N)$, $\text{UNIFY}(\sigma, \tau) = c'' = c_1 \dots c_n.s_f$, where $f = [\varphi \Rightarrow \pi' \to \psi]$ and c_i is the least unifying chain for σ_i and τ_i. So

$$\text{PP}(xMN) \sim \langle H(s_k(c''(B) \nabla c''(B'))), H(s_k(c''(\psi)))) \rangle \sim \langle s_f(B) \nabla s_f(B'), s_f(\psi) \rangle$$

$$\sim \langle \{\pi_1 \to \cdots \to \pi_m \to \pi' \to \psi x\} \cup B_1 \cup B_m \cup \underline{B}', \psi \rangle \sim \text{pp}(xMN).$$

Obviously, xMN is strongly normalizing. Moreover, let $c''' = c.c'.s_g$, where $g = [\psi \Rightarrow \beta]$. Further $\langle B \cup B', \beta \rangle \sim c'''(\mathrm{pp}(xMN))$ since $\mathrm{pp}(N)$ and $\mathrm{pp}(xM)$ are disjoint; then $P(B \cup B', \beta, xMN)$.

(ii): $P(B \cup \{\alpha x\}, \beta, Mx) \Rightarrow \mathrm{PP}(M) = \langle B_1, \pi_1 \rangle \sim \mathrm{pp}(M)$ and $\mathrm{PP}(x) = \langle \{\varphi x\}, \varphi \rangle \sim \mathrm{pp}(x)$ and if

$$\langle \sigma \equiv \sigma_1 \to \cdots \to \sigma_n \to \pi_1,\ \tau \equiv \tau_1 \to \cdots \to \tau_n \to \varphi \to \psi \rangle$$

is a closure of $\langle B_1, \pi_1 \rangle$ and $\langle B_2, \pi_2 \rangle$, then $\mathrm{UNIFY}(\sigma, \tau) = c'$ and

$$\mathrm{PP}(Mx) = \langle H(s_k(c'(B_1) \nabla \{c'(\varphi)x\})),\ H(s_k(c'(\psi))) \rangle$$

$$\sim \langle c'(B_1) \nabla \{c'(\varphi)x\},\ c'(\psi) \rangle \sim \mathrm{pp}(Mx),$$

(by the structure of $\langle B_2, \pi_2 \rangle$) where B_1, and consequently $c'(B_1)$, do not contain predicates whose subjects are x, and $c'(\varphi)$ is a type variable, say φ', and $c'(\psi)$ does not contain φ'. Moreover,

$$\exists c. \langle B \cup \{\alpha x\}, \beta \rangle \sim c(\mathrm{pp}(Mx))$$

and Mx is strongly normalizing.

Then

$$\langle B, \alpha \to \beta \rangle \sim c(\langle c'(B_1), c'(\varphi) \to c'(\psi) \rangle) \sim c(\langle c'(B_1), c'(\pi_1) \rangle)$$

(since c' unifies π_1 and $\varphi \to \psi$, and since they are two principal pairs, $c'(\pi_1) \sim c'(\varphi \to \psi)$ is an arrow type scheme, as proved in the Appendix in the proof of Theorem A.8)

$$\sim c.c'(\langle B_1, \pi_1 \rangle),$$

and then $P(B, \alpha \to \beta, M)$.

(iii) and (iv) are immediate. □

Proof of Theorem 5.2 (*continued*). Then define, by induction on the structure of type schemes $\neq \omega$, the following computability predicate:

$$\mathrm{Comp}(B, \varphi, M) \Leftrightarrow P(B, \varphi, M),$$

$$\mathrm{Comp}(B, \sigma \to \tau, M) \Leftrightarrow (\mathrm{Comp}(B', \sigma, N) \Rightarrow \mathrm{Comp}(B \cup B', \tau, MN)),$$

$$\mathrm{Comp}(B, \sigma_1 \wedge \sigma_2, M) \Leftrightarrow \mathrm{Comp}(B, \sigma_1, M) \text{ and } \mathrm{Comp}(B, \sigma_2, M).$$

It is easy to prove, by induction on the structure of M, that Comp is invariant under β-convertibility.

Lemma 5.7. (i) $P(B, \sigma, xM) \Rightarrow \mathrm{Comp}(B, \sigma, xM)$.

(ii) $\mathrm{Comp}(B, \sigma, M) \Rightarrow P(B, \sigma, M)$.

Proof. (i) and (ii) are proven by simultaneous induction on σ.

- σ is a type variable: (i) and (ii) follow from the definition of Comp.
- $\sigma = \alpha \to \beta$.

(i): $\mathrm{Comp}(B', \alpha, N) \Rightarrow P(B', \alpha, N)$ (by induction hypothesis). $P(B, \alpha \to \beta, xM)$ and $P(B', \alpha, N)$ imply $P(B \cup B', \beta, xMN)$ (by Property 5.6); hence, $\mathrm{Comp}(B \cup B', \beta, xMN)$ (by induction hypothesis). Then $\mathrm{Comp}(B', \alpha, N)$ and $\mathrm{Comp}(B \cup B', \beta, xMN)$ imply $\mathrm{Comp}(B, \alpha \to \beta, xM)$ (by definition of Comp).

(ii): Let $x \in \mathrm{FV}(M) \cup \mathrm{FV}(B)$ and let $\alpha x \in B'$. $P(B', \alpha, x)$ implies $\mathrm{Comp}(B', \alpha, x)$ (by induction). $\mathrm{Comp}(B, \alpha \to \beta, M)$ and $\mathrm{Comp}(B', \alpha, x)$ imply $\mathrm{Comp}(B \cup B', \beta, Mx)$ (by definition); hence $P(B \cup B', \beta, Mx)$ (by induction) and thus $P(B, \alpha \to \beta, M)$ (by Property 5.6).

- $\sigma = \sigma_1 \wedge \sigma_2$.

(i): $P(B, \sigma_1 \wedge \sigma_2, xM)$ implies $P(B, \sigma_1, xM)$ and $P(B, \sigma_2, xM)$ (by Property 5.6); hence, by induction, $\mathrm{Comp}(B, \sigma_1, xM)$ and $\mathrm{Comp}(B, \sigma_2, xM)$ and thus $\mathrm{Comp}(B, \sigma_1 \wedge \sigma_2, xM)$ (by definition).

(ii): By definition of Comp and by the induction hypothesis. □

Lemma 5.8. *Let* $\mathrm{FV}(M) = \{x_1, \ldots, x_m\}$, *and let* $B = \{\sigma_i x_i \mid 1 \leq i \leq m\}$. $\mathrm{Comp}(B', \sigma_i, N_i)$ $(1 \leq i \leq m)$ *and* $\mathrm{PP}(M) = \langle \underline{B}, \pi \rangle$ *and* $\exists c. \langle B, \tau \rangle \sim c(\langle \underline{B}, \pi \rangle) \Rightarrow \mathrm{Comp}(B \cup B', \tau, M[x_i/N_i])$.

Proof. By induction on M. The only nontrivial case is $M = \lambda x.M'$. Then $\mathrm{PP}(M) = \langle \underline{B}, \alpha \to \beta \rangle$ implies $\mathrm{PP}(M') \sim \langle \underline{B} \cup \{\alpha x\}, \beta \rangle$, where $\underline{B}$ does not contain predicates whose subject is x.

$$\mathrm{PP}(M') \sim \langle \underline{B} \cup \{\alpha x\}, \beta \rangle \quad \text{and} \quad \exists c. \langle B \cup \{\sigma x\}, \tau \rangle \sim c(\langle \underline{B} \cup \{\alpha x\}, \beta \rangle),$$

and for $1 \leq i \leq m$

$$\mathrm{Comp}(B'', \sigma, N) \text{ and } \mathrm{Comp}(B', \sigma_i, N_i)$$

$$\Rightarrow \mathrm{Comp}(B \cup \{\sigma x\} \cup B' \cup B'', \tau, M'[x/N, x_i/N_i])$$

(by induction); hence,

$$\Rightarrow \mathrm{Comp}(B \cup \{\sigma x\} \cup B' \cup B'', \tau, (\lambda x.M'[x_i/N_i])N)$$

(since Comp is invariant under β-convertibility) and hence,

$$\Rightarrow \mathrm{Comp}(B \cup B', \sigma \to \tau, \lambda x.M'[x_i/N_i])$$

(by definition). □

Proof of Theorem 5.2 (*conclusion*). Then let $\mathrm{PP}(M)$ be defined and let $\{x_1, \ldots, x_m\} = \mathrm{FV}(M)$, and let $\langle B, \tau \rangle \sim c(\mathrm{PP}(M))$ and let $B = \{\sigma_i x_i \mid 1 \leq i \leq m\}$. Then $\mathrm{Comp}(B, \sigma_i, x_i)$. By Lemma 5.8, this implies $\mathrm{Comp}(B, \tau, M)$, which implies $P(B, \tau, M)$, by Lemma 5.7(ii). □

6. The intersection type discipline without ω

In [4] for the first time an intersection type discipline was introduced, built from a set of type variables, without any constants. More precisely, the type schemes are defined as in Definition 2.1, without the constant ω, and the assignment rules are as in Definition 2.4, without the rule (ω). The definition of pairs, equivalence relation $\sim$, and operations of pairs remain unchanged. It is possible to define a principal pair in this discipline, by means of a procedure PP', which is very similar to PP. PP' is defined in the following way.

Algorithm PP'

$\mathrm{PP}'(X)=\langle B, \pi\rangle$ (if defined) **where**

(1) **if** X is a variable **then** $\langle B, \pi\rangle=\langle\{\varphi X\}, \varphi\rangle$ **where** φ is a fresh type variable.

(2) **if** $X=\lambda x.X'$

then if $\mathrm{PP}'(X')=\langle B', \pi'\rangle$

then if B' contains a premise on x, let σx,

then $\mathrm{PP}'(X)=\langle B'-\{\sigma x\}, \sigma\to\pi'\rangle$

else $\mathrm{PP}'(X)=\langle B', \varphi\to\pi'\rangle$, where φ is a fresh type variable.

(3) **if** $X=X_1X_2$

then if $\mathrm{PP}'(X_1)=\langle B_1, \pi_1\rangle$ **and** $\mathrm{PP}'(X_2)=\langle B_2, \pi_2\rangle$

then let $\langle\sigma\equiv\sigma_1\to\cdots\to\sigma_n\to\pi_1,\ \tau\equiv\tau_1\to\cdots\to\tau_n\to\pi_2\to\varphi\rangle$ be a closure of $\langle B_1, \pi_1\rangle$ and $\langle B_2, \pi_2\rangle$;

if $\mathrm{UNIFY}(\sigma, \tau)=c$ **then** $\langle B, \pi\rangle=\langle c(B_1)\nabla c(B_2), c(\varphi)\rangle$.

The following theorem proves that $\mathrm{PP}'(X)$, if defined, is really the principal pair of X.

Theorem 6.1. (i) $\mathrm{PP}'(X)=\langle B, \pi\rangle$ *implies that, for every (lifting) chain* c, $c(\langle B, \pi\rangle)$ *is a suitable pair for* X.

(ii) $\langle B', \tau\rangle$ *is a suitable pair for* X *implies* $\mathrm{PP}'(X)=\langle B, \pi\rangle$ *and there exists a (lifting) chain* c *such that* $\langle B', \tau\rangle=c(\langle B, \pi\rangle)$.

Proof. (i): By induction on the length of c and, in the first step, by induction on X.

(ii): By induction on the structure of X. The proof is similar to the proof of Theorem 5.2 ($\Leftarrow$). □

Note that in this type discipline every term has at most one principal pair. Now, the following result holds.

Theorem 6.2. $\mathrm{PP}'(X)=\langle B, \pi\rangle\Leftrightarrow X$ *is strongly normalizing.*

The proof is contained in the proof of Theorem 5.2. Then Theorem 6.1 has the following corollary.

Corollary 6.3. *In the intersection type discipline without the constant ω, there exists a pair suitable for X iff X is strongly normalizing.*

Appendix

In order to give the proof of the completeness theorem, first two particular classes of type schemes will be defined, the PTs and the PBs, and it will be proved that UNIFY is complete, when applied to a PT and a PB which are disjoint, and that it finds the minimal proper unifying chain. Moreover, it will be proved that if $\langle \sigma, \tau \rangle$ is a closure of two principal pairs, then σ and τ belong respectively to these classes.

Definition A.1. A *level* of an occurrence of type scheme σ in a type scheme τ is an integer $\geqslant 0$ defined as follows:

- if $\tau \equiv \sigma$, σ occurs at level 0 in τ;
- if $\tau \equiv \mu \wedge \nu$, an occurrence of σ at level n in $\mu(\nu)$ remains at level n in τ;
- if $\tau \equiv \mu \to \nu$, an occurrence of σ at level n in μ comes at level $n+1$ in τ, and an occurrence of σ at level n in ν remains at level n in τ.

If $\tau \equiv \tau_1 \to \cdots \to \tau_n \to \varphi$, then we will say that σ is the head of τ, and if ψ is the head of τ_i, we will say that φ follows ψ.

Definition A.2. The sets of pt's and pb's are inductively defined as follows:

- ω is a pt,
- ω is a pb,
- a type variable φ is a pt,
- a type variable φ is a pb,
- if $\sigma_1, \ldots, \sigma_n$ are pb's, $\sigma_1 \wedge \cdots \wedge \sigma_n$ is a pb iff (*) holds,
- if σ is a pb and τ is a pt, $\pi = \sigma \to \tau$ is a pt iff (*) holds,
- if τ is a pt, and σ is a pb disjoint from τ, then $\tau \to \sigma$ is a pb.

Here a type scheme π satisfies condition (*) iff

(*) every type variable occurs in π at most twice and if φ occurs twice in π, one occurrence is at odd level and one at even level.

From Definition A.2, it follows easily that the following property holds.

Property A.3. (1) *Let τ be a* pt (pb); *φ occurs once in τ, and let σ be a* pt (pb). *Then $\tau[\varphi/\sigma]$ is a* pt (pb) *if φ occurs at even (odd) level and condition (*) holds.*

(ii) *Let τ be a* pt (pb), *and let $e_{A,\sigma}$ be an expansion such that $\rho \in L(\sigma, !A)$ implies that either ρ is a* pb *in τ or it is a subtype of a type scheme already occurring in $L(\sigma, !A)$. Then $e_{A,\sigma}$ is a* pt (pb).

Definition A.4. A pt (pb) π is a PT (PB) iff for every type variable φ occurring twice in it the following condition (**) holds: EITHER

(a) φ occurs as head of a pb and of a pt, and the pb is a subtype of the pt, OR

(b) φ occurs in two subtypes of π (no subtypes of each other) and there exists a sequence of type variables $\varphi_1, \ldots, \varphi_n$ $(n \geqslant 0)$ such that φ_1 and φ_n follow φ, and $\exists j$ $(1 \leqslant j \leqslant n)$ such that, for $i < j$, φ_{i+1} follows φ_i, φ_j follows φ_{j-1} and φ_{j+1}, and, for $i > j$, φ_i follows φ_{i+1}.

$\varphi_1, \ldots, \varphi_n$ will be called the *sequence* of φ.

Definition A.5. A *path* is a finite sequence (possibly empty) of 0 and 1. Let τ be a type scheme. An occurrence of a subtype σ in τ can be identified by a path in the following way:
- $\sigma \equiv \tau$, then σ occurs in τ at the empty path $(\emptyset)$;
- if $\tau \equiv \alpha \to \beta$ $(\alpha \wedge \beta)$, and there is an occurrence of σ in α (β) at path γ, then the same occurrence is at path $\gamma.0$ $(\gamma.1)$ in τ.

Let $\langle \tau, \gamma \rangle$ denote the subtype occurring in τ at path γ. Let $\leqslant$ be the lexicographic ordering between paths. If γ and γ' are paths, $\gamma \ll \gamma'$ will denote $\gamma < \gamma'$ and γ not a prefix of γ'.

Note that if two subtypes α and β are corresponding in, say, σ and τ, then there exists a path γ such that $\alpha \equiv \langle \sigma, \gamma \rangle$ and $\beta \equiv \langle \tau, \gamma \rangle$ (corresponding subtypes are defined in Definition 4.3). Moreover, it is necessary to characterize a property of the semi-algorithm UNIFY, which is the key property for proving the completeness.

Definition A.6. (i) The maximum path (mp) of UNIFY(τ, σ), at the nth step, is defined as follows:
- if $n = 0$, mp is the empty path;
- if $n > 0$, let UNIFY(τ, σ) perform at the nth step $U(\alpha, c(\tau), \beta, c(\sigma), m)$, and let its mp at step $n-1$ be γ; if α and β are corresponding in τ and σ at path $\gamma' \geqslant \gamma$, then the mp is γ', otherwise it is γ.

(ii) Let UNIFY(τ, σ) have mp $= \gamma$ at step n, and suppose it has performed a chain c. It is *local* at γ iff either at step n it performs no expansion, or, if it performs an expansion, say $e_{A,\alpha}$, for any $\rho \in L(\alpha, !A)$, the ρ does not occur either in $c(\tau)$ nor in $c(\sigma)$ at a path γ', where γ' is a proper prefix of γ.

Note that if UNIFY (σ, τ) is local at γ, then, in the first n steps, it preserves corresponding subtypes at path $\leqslant \gamma$.

Lemma A.7. *Let π and σ be respectively a* PT *and a* PB, *and let them be disjoint. Let* UNIFY(π, σ) *perform at the n-th step $U(\alpha, c(\tau), \beta, c(\sigma), m)$, and let its* mp *be γ; then*:

(1) UNIFY(π, σ) *is local at γ;*

(2) *all corresponding subtypes at path $\gamma' \geqslant \gamma$ are a* pt *and a* pb *and they are disjoint;*

(3) *if a type variable occurs twice at a path $\gamma' \geqslant \gamma$, it occurs once at even level and once at odd level;*

(4) *if α and β are one an arrow and one an intersection, then if a type variable occurs twice at path $\gamma' \geqslant \gamma$, then it satisfies condition* (**);

(5) *if μ and ν are corresponding in $c(\pi)$ and $c(\sigma)$ at a path $\geqslant\gamma$, then if δ is a subtype of $c(\pi)$ ($c(\sigma)$) occurring at a path $\geqslant\gamma$ and it contains some type variables occurring in μ (ν), then its corresponding subtype is disjoint from ν (μ).*

Proof. By induction on γ. The case $\gamma=\emptyset$ is obvious.

Let $\gamma\neq\emptyset$. Consider the least number n of steps such that UNIFY(π,σ) performs, at the nth step, $U(\alpha, c(\pi), \beta, c(\sigma), m)$, and γ is the mp. Then α and β are corresponding at path γ in $c(\pi)$ and $c(\sigma)$, they are a pt and a pb and they are disjoint, by induction. The proof is given by induction on α and β.

- The case of either α or β being ω is obvious.
- Let α be a type variable and β be a type scheme which is not an intersection. Then $U(\alpha, c(\pi), \beta, c(\sigma), m)$ performs the substitution s_f where $f=[\alpha\Rightarrow\beta]$. Statement (1) is obvious by the definition of locality. We will prove that, for every $\gamma'\geqslant\gamma$, $\langle c.s_f(\pi), \gamma'\rangle$ and $\langle c.s_f(\sigma), \gamma'\rangle$ are a pt and a pb.

If $\langle c(\pi), \gamma\rangle$ is the unique occurrence of α in $c(\pi)$ and $c(\sigma)$ at a path $\geqslant\gamma$, then the proof follows by induction since

$$\langle s_f(\pi), \gamma'\rangle\equiv\langle c(\pi), \gamma'\rangle \quad\text{and}\quad \langle c.s_f(\sigma), \gamma'\rangle\equiv\langle c(\sigma), \gamma'\rangle.$$

Otherwise, note that at a path $\geqslant\gamma$, after the substitution, there are no occurrences of α, and at most two occurrences of every type variable φ occurring in β. Then $\langle c.s_f(\pi), \gamma'\rangle$ and $\langle c.s_f(\sigma), \gamma'\rangle$ are a pt and a pb, by Property A.4, iff (*) holds. We only need to check it for type variables occurring in β and occurring also at a path $\gamma''\geqslant\gamma$. Let φ be any of these type variables. Let

- $\langle c(\pi), \gamma\rangle\equiv\alpha$ be at level p of $c(\pi)$,
- the other occurrence of α be at level p',
- $\langle c(\sigma), \gamma.\gamma'''\rangle\equiv\varphi$ be at level $w=p+r$,
- the other occurrence of φ be at level w'.

By induction, $p(w)$ even (odd) implies $p'(w')$ odd (even). The two occurrences of σ at path $\geqslant\gamma$, after the substitution, are respectively at level $p'+r$ and w'. But there are three cases:

(i) p even and r even$\Rightarrow w$ even and p' odd and $p'+r$ odd,

(ii) p even and r odd$\Rightarrow w$ odd and p' odd and $p'+r$ even,

(iii) p odd and r odd$\Rightarrow w$ even and p' even and $p'+r$ odd.

Then $p'+r$ is even (odd) according to w' being odd (even); since this holds for any type variable, every pair of corresponding subtypes at path $\geqslant\gamma$ is composed by a pt and a pb.

(4) and (5) easily follow by induction.

Now we will prove that if $\gamma=\gamma''.0$ ($\gamma''.1$), then condition (**) holds for type variables occurring twice at path $\geqslant\gamma''$. So (3) is proved since $U(\alpha, c(\tau), c(\sigma), m)$ performs, at the next step, $U(\alpha', c.s_f(\tau), \beta', c.s_f(\sigma), m)$, whose mp is γ''. As before, it is necessary to prove it only for type variables occurring in β, and only if there is an occurrence of α at path $\geqslant\gamma''$. Let φ occur in β, and also at a path $\gamma'\geqslant\gamma''$.

Let $\varphi\equiv\langle c(\sigma), \gamma^*=\gamma.\gamma'''\rangle$. If either φ is not the head of β or $\gamma=\gamma''.1$, it is immediate to verify that

(A) The two occurrences of φ at path $\gamma.\gamma'''$ and at path γ' satisfying condition (**)(a) implies that the two occurrences of φ are both inside β; so after the substitution, the two occurrences of φ at path $\geqslant\gamma''$, if existing, satisfy condition (**)(a).

(B) The two occurrences of φ at path $\gamma.\gamma'''$ and at path γ' satisfying condition (**)(b), and $\varphi_1, \ldots, \varphi_m$ being the sequence of φ imply that the two occurrences of φ at path $\geqslant\gamma''$, if existing, satisfy condition (**)(b), and their sequence is $\varphi_1, \ldots, \varphi_m$.

Let φ occur as head of β, and let $\gamma = \gamma''.0$. If the two occurrences of φ at path γ and γ' satisfy (**)(a), there are not two occurrences of φ at path $\geqslant\gamma'$. Let them satisfy (**)(b), and let $\varphi_1, \ldots, \varphi_m$ be their sequence. Let $\rho \equiv \alpha \to \rho_1 \to \cdots \to \rho_n \to \psi$ be the least subtype $\equiv\alpha$ containing α, and let $\mu \equiv \beta \to \mu_1 \to \cdots \to \mu_p \to \psi'$ be its corresponding subtype. Since φ, and every type variable of the sequence of φ, do not occur in $\rho_1 \to \cdots \to \rho_n \to \psi$ nor in $\mu_1 \to \cdots \to \mu_p \to \psi'$, UNIFY$(\tau, \sigma)$ is local at every path $\gamma'.\delta$ for every path δ. So if there exists an n' such that, after n' steps, UNIFY(π, σ) has mp $\gamma''.1$, the chain c performed by UNIFY from step n to step n' is such that $c(\rho) \approx c(\mu)$ is an arrow type scheme. Then, if ξ is the head of $c(\tau)$, the two occurrences of φ at path $\geqslant\gamma''$ satisfies (**)(b) with sequence $\xi, \varphi_1, \ldots, \varphi_m$.

- Let α be a type variable and β be an intersection. Note that, from induction, this implies $\alpha(\beta)$ is a pt (pb). Then $U(\alpha, c(\pi), \beta, c(\sigma), m)$ performs $U(c.e(\pi), c.e(\pi), c.e(\sigma), c.e(\sigma), m)$, whose mp is γ, where e is the expansion $e_{A,\alpha}$ and $A = \{c(\pi), c(\sigma)\}$. It will be proved that every type scheme η occurring in $L(\alpha, !A\}$ occurs at path $\geqslant\gamma$ and it is either a pb or a subtype of a type scheme already occurring in $L(\alpha, !A)$. Then the proof follows from Property A.3, and from the fact that expansion preserves disjointness between type schemes.

By induction on the construction of $L(\alpha, !A)$, first pose $L(\alpha, !A) = \{\alpha\}$ and look for a subtype η, if it exists, of the shape $\eta_1 \to \cdots \to \eta_p \to \alpha$; by induction, it is a pb and this implies η_h disjoint from η_k $(h \neq k)$. So η cannot occur at a path which is a prefix of γ. Then suppose $L(\alpha, !A) = !\{\alpha, \eta\}$. Now it is necessary to insert into $L(\alpha, !A)$ every subtype of the shape $\nu \equiv \nu_1 \to \cdots \to \nu_q \to \psi$ such that ψ occurs in η. If ψ is a pt in η, then ν is a pb. Otherwise, if the two occurrences of ψ satisfy condition (**)(a), ν is inside η. If they satisfy condition (**)(b) and if $\psi_1, \ldots, \psi_m$ is the sequence of η, then there is a subtype of the shape $\mu \equiv \mu_1 \to \cdots \to \mu_n \to \varphi$, where $\eta \equiv \mu_h$ for some h and φ is either ψ_1 or ψ_m, which is a pb and belongs to $L(\alpha, !A)$. Moreover, condition (**)(b) ensures us that μ occurs at a path $\geqslant\gamma$.

- The case α and β being both arrow type schemes follows directly from induction.
- The case of α and β being both intersection type schemes is not possible by induction.
- In the case of α being an arrow and β an intersection, the proof is similar to the case of α being a type variable and β an intersection. □

Theorem A.8 (Completeness for PTs and PBs). *Let π and σ be respectively a* PT *and a* PB, *and let them be disjoint. If there exists a proper chain c such that $c(\pi) \approx c(\sigma)$, then* UNIFY$(\pi, \sigma) = c'$, *c' is proper and $c = c'.c''$, for some c''.*

Proof. We must prove the following stronger assumption: *Let π and σ be respectively a* PT *and a* PB, *and let them be disjoint. Let* UNIFY(π, σ) *perform* $U(\alpha, \pi', \beta, \sigma', m)$, *where* $\alpha \equiv \langle \pi', \gamma \rangle$, $\beta \equiv \langle \sigma', \gamma \rangle$ *and let* mp $= \gamma$. *If there exists a proper chain c such that* $c(\alpha) \approx c(\beta)$ *and n is the least number of steps such that* $U(\alpha, c(\pi), \beta, c(\sigma), m)$ *performs* $U(\alpha', c'(\pi'), \beta', c'(\sigma'), m')$ *whose* mp *is* $\geqslant \gamma$, *then c' is proper and* $c = c'.c''$ *for some* c''.

This proof will be given by induction on the triples $\langle E(c), n(\alpha, \beta), s(\alpha, \beta) \rangle$, where $n(\alpha, \beta)$ and $s(\alpha, \beta)$ are defined in the proof of Theorem 4.6, ordered in lexicographic order.

• Let $E(c) = 0$. Let α be a type variable. The case of β being an intersection is not possible. Since, by Lemma A.7, α does not occur in β, $U(\alpha, \pi', \beta, \sigma', m)$ performs s_f, where $f = [\alpha \Rightarrow \beta]$. Then the proof is obvious.

Let $\alpha \equiv \omega$. Then every unifying chain is such that $c(\alpha) \approx c(\beta) \approx \omega$, and $U(\alpha, \pi', \beta, \sigma', m)$ performs s_f, where $s_f(\alpha) \approx s_f(\beta) \approx \omega$.

Let $\alpha \equiv \mu \to \nu$ and $\sigma \equiv \rho \to \tau$. Since $E(c) = 0$, $c(\alpha) \approx c(\beta)$ is an arrow type scheme. Then $U(\alpha, \pi', \beta, \sigma', m)$ performs $U(\mu, \pi', \rho, \sigma', 0)$. By induction, after a finite number of steps $U(\mu, \pi', \rho, \sigma', 0)$ performs $U(\mu', c_1(\pi'), \rho', c_1(\sigma'), m')$ (whose mp is $\gamma.1$), and $c = c_1.c'$, for some c' (since $n(\mu, \rho) \leqslant n(\alpha, \beta)$ and $s(\mu, \rho) < s(\alpha, \beta)$). But since c_1 preserves corresponding occurrences, $\mu' \equiv c_1(\nu)$ and $\rho' \equiv c_1(\tau)$, where, by Lemma A.7, $c_1(\nu)$ and $c_1(\tau)$ are a pt and a pb which are disjoint, and c' is a proper unifying chain for them. So, by induction, after a finite number of steps, $U(\mu', c_1(\pi'), \rho', c_1(\sigma'), m')$ performs $U(\alpha', c_1.c_2(\pi'), \beta', c_1.c_2(\sigma'), m'')$, whose path is the least path $\geqslant \gamma.1$, and $c' = c_2.c''$, for some c''. Then $c = c_1.c_2.c''$, and the proof is complete

The case of α and β being both intersection type schemes is similar.

• Let $E(c) > 0$. Let $\alpha \equiv \mu \to \nu$ and $\beta \equiv \rho \wedge \tau$. Then, by Theorem 2.14, since σ and π are disjoint, $c = e_{A,\alpha}.c'$ with $A = \{\alpha, \beta\}$ and $E(c') < E(c)$. It is immediate to verify that since condition (**) holds,

$$e_{A,\alpha}(\alpha) \approx e_{A',\alpha}(\alpha) \quad \text{and} \quad e_{A,\alpha}(\beta) \approx e_{A',\alpha}(\beta) \approx \beta,$$

where $A' = \{\pi', \sigma'\}$. Then the proof follows by induction since $E(c') < E(c)$ and $U(\alpha, \pi', \beta, \sigma', m)$ performs, at the next step,

$$U(e_{A',\alpha}(\pi'), e_{A',\alpha}(\pi'), e_{A',\alpha}(\sigma'), e_{A',\alpha}(\sigma'), 1).$$

Let $\alpha \equiv \mu \to \nu$ and $\beta \equiv \rho \to \tau$. In the case of $c(\alpha) \approx c(\beta)$ being an arrow type scheme, the proof follows by induction, using the fact that UNIFY is local. If $c(\alpha) \approx c(\beta) \approx \rho_1 \wedge \rho_2$, assume ρ_1 and ρ_2 are arrow type schemes. Since α and β are disjoint, by two applications of Theorem 2.14, $c = e.e'.\underline{c}$, where $e = e_{A,\alpha}$, $e' = e_{e(\alpha),e(\beta)}$, $A = \{\pi', \sigma'\}$, $E(\underline{c}) < E(c)$. But $e(\beta) \equiv \beta$, $e(\alpha) \equiv \alpha_1 \wedge \alpha_2$ (where α_1 and α_2 are disjoint instances of α), $e.e'(\beta) = \beta_1 \wedge \beta_2$ (where β_1 and β_2 are disjoint instances of β) and $e.e'(\alpha) = \alpha_1 \wedge \alpha_2$. So, since α_1 is disjoint from β_i $(1 \leqslant i \leqslant 2)$, $c = e.e'.c''.c'''$, where c'' is a proper unifying chain for α_1 and β_1, c''' is a proper unifying chain for α_2 and β_2 and, moreover, c'' and c''' are disjoint. Then, by induction, $U(\alpha, \pi', \beta, \sigma', m)$

performs, after a finite number of steps, a chain c_3 such that if c_1 and c_2 are disjoint suitable instances of c_3, $c'' = c_1.c_1'$ and $c''' = c_2.c_2'$. Then $c = e.e'.c_1.c_1'.c_2.c_2'$. But now it is immediate to verify that $c = c_3.e''.s_f.c_1'.c_2'$, where e'' is the expansion $e_{A,\eta}$, where $\eta \equiv c'(\beta)$ and $A = \{\eta\}$ and s_f is suitable renaming of type variables.

The case of either ρ_1 or ρ_2 being an intersection type scheme can be treated in a similar way, simply repeating the preceding reasoning.

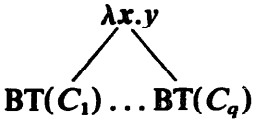
Then the proof of the theorem follows immediately, since π and σ are corresponding in π and σ at path $\emptyset$. □

Now Theorem 4.8 follows from the following property.

Property A.9. *Let $\langle \sigma, \tau \rangle$ be a closure of two principal pairs. Then σ is a* PT *and τ is a* PB.

Proof. It is sufficient to prove that in a principal type scheme of a closed approximant (so every type variable occurs twice in it) condition (**) holds for every type variable. Then the proof follows from the definition of a closure. Let τ be the principal type scheme of the closed approximant $A = \lambda x_1 \ldots x_n.x_i A_1 \ldots A_m$. By definition, $\tau \equiv \tau_1 \to \cdots \to \tau_p \to \psi$, where τ_j is the intersection of the predicates whose subject is x_j, used to derive the principal type scheme of $A_1, \ldots, A_m$. It follows immediately from Property 5.6 that τ is a pt.

We recall the definition of the Böhm tree of C (BT(C)), where C is an approximant (in the following we will use $\lambda x.C$ as abbreviation for $\lambda x_1 \ldots x_p.C$, $p \geq 0$):
- if $C \equiv \lambda x.y$, then BT(C) is the tree consisting of one node, whose name is $\lambda x.y$
- if $C \equiv \lambda x.yC_1 \ldots C_q$, then $BT(C)$ is the treee



Consider BT(A). Given a node $N \equiv \lambda y.t$ of BT(A), the *way* of N in A is the sequence of nodes WN defined as follows
- if $t \in x$, then $WN \equiv N$;
- otherwise, $WN \equiv N,N',WN''$, where N' is the node $\lambda z.r$ such that $t \in z$, and N'' is the father of N' ("," is the concatenation between sequences). Note that WN consists only of nodes which belong to the path from N to the root.

Now, let φ be a type variable occurring in τ. φ is the head of the principal type scheme of a subterm A' of A, let $A' \equiv \lambda z.tB_1 \ldots B_p$. Let N_φ denote the node $\lambda z.t$ in BT(A). It is immediate to verify that if $A' \equiv A$, $t \equiv x_i$ and φ occurs as head of τ and of τ_i (case (**)(a)).

Let N'_φ be the father of N_φ and let $s(WN_\varphi)$ be the sequence of type variables defined as follows: if $WN_\varphi \equiv N_\varphi,N_1,\ldots,N_r$, then $s(WN_\varphi) \equiv \varphi,\psi_1,\ldots,\psi_r$, where ψ_i is the head of the principal type scheme of the approximant whose BT has as root N_i. So it is easy to verify that if $WN_\varphi \equiv N_\varphi,N_1,\ldots,\lambda z.x_j$ and $s(WN_\varphi) \equiv$

$\varphi, \psi_1, \ldots, \psi_r$, then φ occurs in τ_1 followed by ψ_1, and ψ_{i+1} follows ψ_i $(1 \leq i \leq r-1)$. Moreover, if $\mathrm{W}N'_\varphi \equiv N'_\varphi, N'_1, \ldots, \lambda y.x_h$, and $s(\mathrm{W}N'_\varphi) \equiv \xi_1, \ldots, \xi_s$, then φ occurs in τ_h followed by ξ_1, and ξ_{i+1} follows ξ_i $(1 \leq i \leq s)$.

Now, let W_φ (W'_φ) be ways defined as follows:

$$W_\varphi \equiv \mathrm{W}N_\varphi, \mathrm{W}M_1, \ldots, \mathrm{W}M_u \qquad (W'_\varphi \equiv \mathrm{W}M'_\varphi, \mathrm{W}M'_1, \ldots, \mathrm{W}M'_w),$$

where M_1 (M'_1) is the father of the last node of $\mathrm{W}N_\varphi$ $(\mathrm{W}N'_\varphi)$ and M_j (M'_v) is the father of the last node of $\mathrm{W}M_{j-1}$ $(\mathrm{W}M'_{w-1})$ $(2 \leq j \leq u, 2 \leq v \leq w)$. So $\mathrm{W}M_1 \equiv \mathrm{W}M_{\psi_u}$ and $\mathrm{W}N'_1 \equiv \mathrm{W}M_{\xi_w}$. Let $s(W_\varphi) \equiv \psi_1, \ldots, \psi_p$, and $s(W'_\varphi) \equiv \xi_1, \ldots, \xi_q$. Note that, by construction, W_φ and W'_φ have at least a node in common: the node which is the root of A. Let v and u be the least integer such that $\psi_v \equiv \xi_u$. The sequence of φ is $\psi_1, \ldots, \psi_v \equiv \xi_u, \xi_{u-1}, \ldots, \xi_1$. So condition (**)(b) holds for φ. □

Note that the other implication of Property A.8 does not hold; in fact,

$$(\alpha \to \alpha) \to \beta \to \beta \quad \text{and} \quad (\gamma \to \gamma) \to \varphi$$

are respectively a PT and a PB and they are disjoint, but they are not the components of a closure.

Remark A.10. The fact that UNIFY, when applied to a PT and a PB, is local has as a consequence that, for this class of input, the defined semi-algorithm performs some nonnecessary steps. In fact, UNIFY, after an expansion, examines again all the corresponding subtypes, starting from subtypes occurring at the empty path since the expansion does in general not preserve corresponding pairs. But, since UNIFY, when applied to a PT and a PB, is local and so preserves corresponding pairs, it can be modified in the following way, in order to increment its efficiency. UNIFY'(σ, τ) is obtained from UNIFY(σ, τ) by replacing points (1) and (2) with:

(1') **if** σ is an intersection **and** τ is not an intersection **then let** $A = \{\sigma', \tau'\}$;

if $\langle c_1, p_1 \rangle = U(e_{A,\tau}(\sigma), e_{A,\tau}(\sigma), e_{A,\tau}(\tau), e_{A,\tau}(\tau), m)$

then $\langle c', p \rangle = \langle e_{A,\tau}.c_1, 1 \rangle$.

(2') **if** τ is an intersection **and** σ is not an intersection **then let** $A = \{\sigma', \tau'\}$;

if $\langle c_1, p_1 \rangle = U(e_{A,\sigma}(\sigma), e_{A,\sigma}(\sigma), e_{A,\sigma}(\tau), e_{A,\sigma}(\tau), m)$

then $\langle c', p \rangle = \langle e_{A,\sigma}.c_1, 1 \rangle$.

Note that UNIFY' is correct only when applied to a PT and a PB. In the implementation of PP, UNIFY' instead of UNIFY is used.

References

[1] H. Barendregt, *The Lambda Calculus: its Syntax and Semantics* (North-Holland, Amsterdam, 1984).

[2] H. Barendregt, M. Coppo and M. Dezani, A filter λ-model and the completeness of type assignment, *J. Symbolic Logic* 84 (1983) 931-940.

[3] M. Berta, Schema di tipo principale per la disciplina dei tipi con intersezione, Dissertazione di Laurea, Dipartimento di Informatica, Torino (1987).

[4] M. Coppo and M. Dezani, An extension of the basic functionality theory for the λ-calculus, *Notre Dame J. Formal Logic* **21** (1980) 685-693.

[5] M. Coppo, M. Dezani and B. Venneri, Principal type scheme and λ-calculus semantics, in: J.P. Seldin and J.R. Hindley eds., *H.B. Curry. Essays on Combinatory Logic, λ-Calculus and Formalism* (Academic Press, London, 1980) 535-560.

[6] H.B. Curry and R. Feys, *Combinatory Logic, Vol.* 1 (North-Holland, Amsterdam, 1958).

[7] R. Hindley, The principal type scheme as an object in combinatory logic, *Trans. Amer. Math. Soc.* **146** (1969) 29-60.

[8] R. Hindley, The completeness theorem for typing λ-terms, *Theoret. Comput. Sci.* **22** (1983) 1-17.

[9] R. Milner, A theory of type polymorphism in programming, *J. Comput. System Sci.* **17** (1978) 348-375.

[10] R. Milner and L. Damas, Principal type schemes for functional programs, in: *Proc 9th Symp. on Principles of Programming Languages* (1982) 207-212.

[11] J.A. Robinson, A machine oriented logic based on the resolution principle, *J. ACM* **12** (1965) 23-41.

[12] S. Ronchi Della Rocca, Characterization theorems for a filter lambda model, *Inform. and Control* **54** (1982) 201-216.

[13] S. Ronchi Della Rocca and B. Venneri, Principal type scheme for an extended type theory, *Theoret. Comput. Sci.* **28** (1984) 151-169.