An unification semi-algorithm for intersection type schemes

Simona Ronchi Della Rocca
Dipartimento di Informatica - Universita' di Torino
corso Svizzera 185 - 10149 Torino

1. Introduction.

The intersection type discipline for λ-calculus (ITD), defined in [Coppo et al.,1980 b], is an extension of the classical functionality theory of Curry [Curry et al.,1958] .In Curry type discipline type schemes are built from type variables using the constructor → (arrow). In the ITD, type schemes are built from type variables and the type constant ω (the universal type) using, as constructors, beside the arrow, the intersection (∧). The semantics of a type scheme of the shape $\alpha\rightarrow\beta$ 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 semantic domain. In the ITD every term has at least one type scheme, and type assignment preserves β-convertibility.

In the Curry 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 the Curry type discipline is decidable, and algorithms to solve it have been proposed by Hindley [Hindley,1969] (for terms of Combinatory Logic), and by Milner [Milner, 1978] [Milner et al.,1982] (for terms of λ-calculus). Milner uses this algorithm in the design of the ML type checker. Both these algorithms are based on the classical unification algorithm of Robinson [Robinson,1965].
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 the substitution, the expansion and the rise [Ronchi et al.,1984].

In this paper the unification problem for intersection type scheme is studied. This problem is semi-decidable. The semi-algorithm UNIFY solving it is presented, and it is proved that ,in the case a solution exists, it finds the more general one. While UNIFY uses also operations different from substitution, it is conservative with respect to Robinson's unification algorithm. Moreover a semi-algorithm PP is presented, using UNIFY as essential tool, which, given a term X, if X is strongly normalizing computes its pts .Since there is a one-one correspondence between a term and its pts (if it exists), 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.

2.The intersection type discipline.

The reader is supposed to have some acquaintance with λ-calculus; in any case he can refer to [Barendregt,1984], whose notations we will use.

Definition 1. i) The set T of *intersection type schemes* is inductively defined by:

$\varphi_i,\varphi_j,\ldots\in T \quad (i\geq 0)$ (type variables)

$\omega\in T$ (type constant)

$\sigma,\tau\in T \Rightarrow (\sigma\to\tau)\in T,(\sigma\wedge\tau)\in T.$

ii) A *statement* is of the form σx with $\sigma\in T$ and x is a variable. x is the *subject* and σ the *predicate* of σM. A *basis scheme* is a (possibly infinite) set of statements .

The notion of subtype of a given type scheme is obtained in a straightforward way from Definition 1.i). $\sigma_1\to\sigma_2\to\ldots\to\sigma_n\to\tau$ is an abbreviation for $\sigma_1\to(\sigma_2\to(\sigma_3\to\ldots(\sigma_n\to\tau)\ldots))$ and $\sigma_1\wedge\sigma_2\wedge\ldots\wedge\sigma_n$ is an abbreviation for $(\sigma_1\wedge(\sigma_2\wedge\ldots(\sigma_{n-1}\wedge\sigma_n)\ldots))$.

The *simple semantics* (for the definition of simple semantics see [Hindley, 1983]) for T can be given in the following way:

Definition 2. Let $\mathcal{M}=\langle D,\cdot,[\![]\!]\rangle$ be a λ-model .

i) Let $N\in\Lambda$; if ξ is a valuation of variables in D, then $[\![N]\!]^{\mathcal{M}}_{\xi}\in D$ is the interpretation of N in $\mathcal{M}$ via ξ.

ii)Let $PD=\{X\mid X\subseteq D\}$ and $V:\{\varphi\mid\varphi \text{ is a type variable}\}\to PD$. Then the interpretation of $\sigma\in T$ in $\mathcal{M}$ via V, notation $[\![\sigma]\!]^{\mathcal{M}}_{V}\in PD$, 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 induces naturally a pre-order relation $\leq$ on T, whose intended meaning is : $\sigma\leq\tau \Leftrightarrow \forall\mathcal{M},V.\ [\![\sigma]\!]^{\mathcal{M}}_{V}\subseteq[\![\tau]\!]^{\mathcal{M}}_{V}$.

Definition 3. The relation $\leq$ (and $\sim$) on T is inductively defined by:

i) $\tau\leq\tau,\ \tau\leq\omega,\ \omega\leq\omega\to\omega,\ \tau\leq\tau\wedge\tau,\ \tau\wedge\sigma\leq\sigma,\ \tau\wedge\sigma\leq\tau,\ (\sigma\to\rho)\wedge(\sigma\to\tau)\leq\sigma\to(\rho\wedge\tau),$
$\sigma\leq\tau\leq\rho\Rightarrow\sigma\leq\rho,\ \sigma\leq\sigma'\text{and}\tau\leq\tau'\Rightarrow\sigma\wedge\tau\leq\sigma'\wedge\tau',\ \sigma\geq\sigma'\text{ and }\tau\leq\tau'\Rightarrow\sigma\to\tau\leq\sigma'\to\tau'.$

ii)$\sigma\sim\tau\Leftrightarrow\sigma\leq\tau\leq\sigma.$

Definition 4. *Type scheme assignment rules*

Let B be a basis scheme, and let Λ be the set of type free λ-terms.

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

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

$(\rightarrow I)\ \dfrac{B\cup(\sigma x)^{*}\vdash \tau X}{B\vdash \sigma\rightarrow\tau\lambda x.X}$ $\qquad$ $(\rightarrow E)\ \dfrac{B\vdash \sigma\rightarrow\tau X \quad B\vdash \sigma Y}{B\vdash \tau XY}$

* where σx is the unique statement of B ,whose subject is x,used to deduce τX.

$(\leq)\ \dfrac{B\vdash \sigma x \quad \sigma\leq\tau}{B\vdash \tau X}$

Note that the rule $(\wedge E)$ is redundant, since it can be directly derived from rule $(\leq)$.

Notation. Let $\langle Z,W,z\rangle$ be a triple of either two type schemes and a type variable, or two terms and a variable. $Z[z/W]$ denote the type scheme (term) obtained from Z by simultaneous replacing each free occurrence of z with W. If z' is a (type) variable , $Z[z/z']$ is an *instance* of Z.

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

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

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 a renaming of bound variables in M).Let $=_\beta$ denote β-convertibility, i.e., the transitive and reflexive closure of β-reducibility.
We recall that a term X is in *normal form* iff $\nexists X'.\ X\rightarrow_\beta X'$, and it is in *head normal form* iff $X=\lambda x_1x_2\ldots x_n.\zeta X_1X_2\ldots X_m$ $(n,m\geq 0)$, where ζ is any variable and $X_i\in\Lambda$ $(1\leq i\leq m)$. X has a normal form (a head normal form) iff $X\rightarrow^*_\beta X'$ and X'is in normal form (head normal form).
The following theorem holds:

Theorem 1 ([Barendregt et al,1981].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 less of generality, we can restrict ourselves to consider only finite basis schemes.

Definition 5.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) = will denote the synctactical identity between type schemes, basis schemes and pairs.
iii) The equvalence relation ~ between pairs is so defined:
$\langle B,\tau\rangle\sim\langle B',\tau'\rangle\Leftrightarrow\tau\sim\tau'$ and, if, $\forall x.\ \sigma_i x$ $(1\leq i\leq m)$ and $\rho_j x$ $(1\leq j\leq n)$ are the all and only statements whose subject is x belonging respectively to B and B', $\sigma_1\wedge\ldots\wedge\sigma_m\sim\rho_1\wedge\ldots\wedge\rho_n$.

Now, we will define two operations on pairs, which preserve suitable pairs, namely the substitution and the expansion.

Definition 6. A *substitution* s is a finite set of pairs $\langle\varphi_i,\mu_i\rangle$ $(1\leq i\leq n)$, where φ_i are

distinct type variables and μ_i are type schemes. Then, for every pair $\langle B,\tau\rangle$:

i) $s(\tau)=\tau[\varphi_i/\mu_i]$, $s(B)=\{s(\sigma)x|\sigma x\in B\}$

ii)$s(\langle B,\tau\rangle)=\langle s(B),s(\tau)\rangle$.

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

Notation. Let L be an ordered list. Then $L\ll a$ will denote the ordered insertion of a into L, and $L\gg a$ will denote the extraction of the maximum element of L, whose name is a.

Definition 7. An *expansion* e is a monuple $\langle\mu\rangle$, where μ is a type scheme.

i) Let $L^e(B,\tau)$ a list of type scheme,ordered by number of symbols (when two type schemes have the same number of symbols their mutual order is unimportant). $L^e(B,\tau)$ is built in the following way:

-$L^e(B,\tau)\ll\mu$

-if $\sigma\in L^e(B,\tau)$, and δ is a proper subtype of σ, $L^e(B,\tau)\ll\delta$.

-for each type scheme σ, such that σ is a subtype of either τ or a predicate in B:

-if either $\sigma=\nu\to\delta$ or $\sigma=\nu\to(\delta\wedge\alpha)$ and $\delta\in L^e(B,\tau)$,then $L^e(B,\tau)\ll\sigma$.

ii)Let $I=\{\varphi_i|\varphi_i$ is a type variable occurring in $L^e(B,\tau)\}$.

Let $s_i=\{\langle\varphi_i,\psi_i\rangle|\ \varphi_i\in I,\psi_i$ is a fresh variable$\}$ $(1\le i\le 2)$.

iii)$\forall\sigma\in T$, $e(\sigma)$ is obtained from σ by means of the following procedure:

<u>while</u> $L^e(B,\tau)\neq\epsilon$ (ϵ is the empty list) <u>do begin</u>

$L^e(B,\tau)\gg\gamma$

<u>if</u> γ occurs in σ <u>then</u> replace γ in σ by $s_1(\gamma)\wedge s_2(\gamma)$

<u>end</u>

iv)$e(B)=\{e(\sigma)x|\sigma x\in B\}$.

v)$e(\langle B,\tau\rangle)=\langle e(B),e(\tau)\rangle$.

An expansion e will be called *total* with respect to a type scheme σ iff $e(\sigma)=e'(\sigma)$, where $e'=\langle\sigma\rangle$.

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 $(\wedge I)$, as can be seen in the following:

Example 1. 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:

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

Let $e=\langle\varphi\rangle$. $e(\langle B,\alpha\rangle)=\langle\{((\varphi_1\to\varphi_1)\wedge(\varphi_2\to\varphi_2))\to\alpha y\},\alpha\rangle$ is a suitable pair for $y(\lambda x.x)$. In fact there exists a deduction D':

$$\dfrac{\dfrac{\dfrac{}{e(B)\cup\{(\varphi_1 x)\}\vdash\varphi_1 x}(\to I)\quad \dfrac{}{e(B)\cup\{(\varphi_2 x)\}\vdash\varphi_2 x}(\to I)}{e(B)\vdash\varphi_1\to\varphi_1\lambda x.x \qquad e(B)\vdash\varphi_2\to\varphi_2\lambda x.x}(\wedge I)}{e(B)\vdash(\varphi_1\to\varphi_1)\wedge(\varphi_2\to\varphi_2)\lambda x.x}$$

$$\dfrac{e(B)\vdash((\varphi_1\to\varphi_1)\wedge(\varphi_2\to\varphi_2))\to\alpha y \qquad e(B)\vdash(\varphi_1\to\varphi_1)\wedge(\varphi_2\to\varphi_2)\lambda x.x}{e(B)\vdash\alpha\ y(\lambda x.x).}(\to E)$$

The notion of instance can be naturally extended to substitutions and expansions; we will say that a substitution $s=\{\langle\varphi_i,\sigma_i\rangle\}$ is an instance of $s'=\{\langle\varphi'_i,\sigma'_i\rangle\}$ iff σ_i is an instance of σ'_i, and an expansion $e=\langle\mu\rangle$ is an instance of $e'=\langle\mu'\rangle$ iff μ is an instance of μ'.

Definition 8.i) A *chain* c is a finite sequence of operations of substitutions and expansions.

ii) Two chains c_1 and c_2 are *equivalent* (notation $c_1==c_2$) iff:

- if $\langle\mu_1,\dots,\mu_n\rangle$ and $\langle\mu'_1,\dots,\langle\mu'_m\rangle$ are all and only the intersections occurring respectively in c_1 and c_2, $\mu_1\wedge\dots\wedge\mu_n$ is an instance of $\mu'_1\wedge\dots\wedge\mu'_m$.
- if $s_1,\dots,s_n$ and $s'_1,\dots,s'_m$ are all and only the substitutions occurring respectively in c_1 and c_2, $U_{1\le i\le n}\ s_i$ is an instance of $U_{1\le i\le m}\ s'_i$.

Note that $c_1==c_2$ does not imply $c_1(\langle B,\tau\rangle)=c_2(\langle B,\tau\rangle)$.

Notation. Let op_i be an operation of expansion or substitution ($1\le i\le n$). We will denote with $op_1.op_2....op_n$ the chain c such that $c(\sigma)=op_n(op_{n-1}(\dots(op_1(\sigma))\dots))$ and with $c_1.c_2$ the chain which is the concatenation of the two chains c_1 and c_2, i.e., $c(\sigma)=c_2(c_1(\sigma))$.

Theorem 2.([Ronchi et al.,1984]).i) Let c be any 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 c be a chain, such that $c(\langle B,\sigma\rangle)=\langle B',\sigma'\rangle$, where $\sigma'=\mu_{\wedge v}$. Then $\exists c'$ such that: $c'=e.c''$, where $e=\langle\sigma\rangle$, $c'(\langle B,\sigma\rangle)=\langle B',\sigma'\rangle$ and $c==c'$.

□

It is possible to see the operation of expansion as operation only on type schemes, not necessary on pairs. The expansion $e=\langle\mu\rangle$, applied on the type scheme τ, is defined as in Definition 7.iii), with $L^e(B,\tau)$ replaced by $L^e(\Phi,\tau)$, where Φ is the empty set. In what follows we will use 'expansion' to denote indifferently the two operations, since the meaning will be clear from the context.

3.The unification semi-algorithm.

The unification problem for type schemes belonging to T could be stated in the following "syntactic" way:

- Given $\sigma,\tau\in T$, find, if it exists, a chain c (of expansions and substitutions), such that $c(\sigma)=c(\tau)$.

But in this formulation of the problem the particular role of the universal type scheme ω is not taken into account. In fact, it is natural to impose that ω can be unified with any type scheme.Then, we can give a "semantic" version of the problem:

- Given $\sigma,\tau\in T$, find, if it exists, a chain c such that $c(\sigma)\sim c(\tau)$.

But this formulation is too general, and it exels our aims. So, we will define a new equivalence relation between type schemes:

Definition 9.i) A ω-type scheme is a type scheme in which only the symbol ω occurs.

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

α,β ω-type schemes $\Rightarrow \alpha\approx\beta$

$\alpha=\beta \quad \Rightarrow \alpha\approx\beta$

$\alpha\approx\alpha',\beta\approx\beta' \quad \Rightarrow \alpha\rightarrow\beta\approx\alpha'\rightarrow\beta',\alpha\wedge\beta\approx\alpha'\wedge\beta'$.

(Note that $\alpha\wedge\beta\neq\beta\wedge\alpha$).

We can now try to give a third formulation of the unification problem:

-Given $\sigma,\tau\in T$, find, if it exists, a chain c such that $c(\sigma)\approx c(\tau)$.

But now the problem has always a solution,the trivial one $c(\sigma)\approx c(\tau)\approx\omega$. The correct formulation of the unification problem must impose that the trivial solution can be choosen only in the case no other solution exists.

Then the final formulation of the problem is:

-Given $\sigma,\tau\in T$,find, if it exists, a chain c such that $c(\sigma)\approx c(\tau)\neq\omega$.

This problem is semi-decidable. In fact, in the following section it will be possible to see that it is equivalent to the problem:

$-\exists\langle B,\tau\rangle.B\vdash\tau X$?

which is clearly semi-decidable (by Theorem 1.i).

The semi-algorithm UNIFY we will show solves the problem in the most general way, i.e., it finds the most general unifying chain, if it exists, otherwise it does not stop, as will be proved in Theorem 3.

Semi-algorithm UNIFY

UNIFY(σ,τ)=U$(\sigma,\sigma,\tau,\tau)$, where U$(\sigma,\sigma',\tau,\tau')=c'$ (if defined) ,where:

1. if σ is a type variable then if $\tau=\sigma$ then $s=\Phi$ else
   if σ occurs in τ then c'=s, where $s=\{\langle\varphi,\omega\rangle,\langle\sigma,\omega\rangle|\varphi$ occurs in $\tau\}$
   else c'=s,where $s=\{\langle\sigma,\tau\rangle\}$;
2. if $\sigma=\omega$ then c'=s, where $s=\{\langle\varphi,\omega\rangle|\varphi$ occurs in$\tau\}$;
3. if $\sigma=\sigma1\rightarrow\sigma2$ then
   3.1. if τ is a variable then
      if τ occurs in σ then c'=s, where $s=\{\langle\varphi,\omega\rangle,\langle\tau,\omega\rangle|\varphi$ occurs in $\sigma\}$
      else c'=s, where $s=\{\langle\tau,\sigma\rangle\}$
   3.2. if $\tau=\omega$ then c'=s, where $s=\{\langle\varphi,\omega\rangle|\varphi$ occurs in $\sigma\}$
   3.3. if $\tau=\tau_1\rightarrow\tau_2$ then
      if c_1=U$(\sigma_1,\sigma',\tau_1,\tau')$ and c_2=U$(c_1(\sigma_2),c_1(\sigma'),c_1(\tau_2),c_1(\tau'))$
      then $c'=c_1.c_2$
   3.4.if$\tau=\tau_1\wedge\tau_2$ then
      let $e=\langle\sigma\rangle$, then if c_1=U$(e(\sigma'),e(\sigma'),e(\tau'),e(\tau'))$
      then $c'=e.c_1$

4. if $\sigma=\sigma_1 \wedge \sigma_2$ then:

4.1. [identical to point 3.1]

4.2. [identical to point 3.2]

4.3. if $\tau=\tau_1 \rightarrow \tau_2$ then

let $e=\langle\tau\rangle$ then if $c_1=U(e(\sigma'),e(\sigma'),e(\tau'),e(\tau'))$

then $c'=e.c_1$

4.4. if $\tau=\tau_1 \wedge \tau_2$ then

if $c_1=U(\sigma_1,\sigma',\tau_1,\tau')$ and if $c_2=U(c_1(\sigma_2),c_1(\sigma'),c_1(\tau_2),c_1(\tau'))$

then $c'=c_1.c_2$.

Exemple 2. Let $\alpha,\beta,\gamma,\delta,\mu,\nu$ be type variables.
i) If $\sigma=\alpha\rightarrow\omega\rightarrow\beta$ and $\tau=(\alpha\rightarrow\delta)\rightarrow(\mu\rightarrow\nu)\rightarrow\gamma$, $UNIFY(\sigma,\tau)=c=s_1.s_2.s_3$, where:
$s_1=\{\langle\alpha,\omega\rangle,\langle\delta,\omega\rangle\}$, $s_2=\{\langle\mu,\omega\rangle,\langle\nu,\omega\rangle\}$, $s_3=\{\langle\beta,\gamma\rangle\}$, and $c(\sigma)\approx c(\tau)\approx\omega\rightarrow\omega\rightarrow\gamma$.
ii) If $\sigma=\alpha\wedge(\alpha\rightarrow\beta)$ and $\tau=(\mu\wedge(\mu\rightarrow\nu))\rightarrow\nu$, $UNIFY(\sigma,\tau)$ does not stop. In fact:
$UNIFY(\sigma,\tau)=UNIFY(e(\sigma),e(\tau))$ (where $e=\langle\tau\rangle$)$=UNIFY(\sigma,\tau'\wedge\tau'')$ (where τ' and τ'' are instances of τ)$=UNIFY(s(\alpha\rightarrow\beta),s(\tau''))$ (where $s=\{\langle\alpha,\tau'\rangle\}$)$=$
$=UNIFY(\tau'\rightarrow\beta,\tau'')=UNIFY(\tau'\rightarrow\beta,\sigma'\rightarrow\nu')$ (where σ' is an instance of σ and ν' is a new type variable) = if $c=UNIFY(\tau',\sigma')$ then $UNIFY(c(\beta),c(\nu'))$
and this function is undefined since τ' and σ' are instances of σ and τ.

To prove that this semi-algorithm is correct and (in some sense) complete, we need a further definition:
Definition 10.i) Let $\sigma,\tau\in T$,and let σ' and τ' be subtypes respectively of σ and τ. Two occurrences of σ' and τ' in σ and τ are *corresponding* iff:
-$\sigma=\sigma'$ and $\tau=\tau'$
-$\sigma=\alpha\rightarrow\beta$ and $\tau=\alpha'\rightarrow\beta'$ ($\sigma=\alpha\wedge\beta$ and $\tau=\alpha'\wedge\beta'$) and the occurrences of σ' and τ' are corresponding either in α and α' or in β and β'.
ii) Let $c=op_1....op_n$ 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 $op_1....op_i(\sigma)$ and $op_1...op_i(\tau)$ (say σ_i and τ_i) such that:
$\sigma_i,\tau_i\approx\omega$ and $\exists j>i$. $op_1..op_j(\sigma_i)\approx op_1...op_j(\tau_i)\approx\omega$.

Roughtly 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 we are able to prove:
Theorem 3.i) (Correctness) If $UNIFY(\sigma,\tau)=c$, then $c(\sigma)\approx c(\tau)$.
ii) (Completeness) Let $\sigma,\tau\in T$ be such that there exists a proper chain c unifying σ and τ. Then $UNIFY(\sigma,\tau)=c'$, where c' is a proper chain unifying σ and τ , and $c=c'.c''$ for some c''(i.e., c' is the *minimal* chain unifying σ and τ,in the sense that every other

some c"(i.e., c' is the *minimal* chain unifying σ and τ,in the sense that every other proper chain unifuing σ and τ must contain (an instance of) every operation occurring in c').

Proof. i) Easy, by induction on the lenght of c.

ii) By induction on the pair $\langle l(c),n(\sigma,\tau)\rangle$ (we assume the lexicographical order between pairs), where:

- $l(c)$ is the lenght of c, i.e., the sum of the number of expansions occurring in c and the cardinality of the union of all the substitutions occurring in c.
- $n(\sigma,\tau)$ is the total number of symbols occurring in σ and τ.

The case $l(c)=0$ and $l(c)=1$ are obvious.

Let $l(c)>1$.In the case σ is a type variable, we must distinguish two cases, according to τ contains or not occurrences of σ. In the first case obviously there is no a proper chain unifying σ and τ. Otherwise, UNIFY makes the substitution $s=\{\langle\sigma,\tau\rangle\}$. Obviously this is the minimal between all the proper unifying chains composed only of substitutions (see [Robinson, 1965]). It easy to see that every proper chain unifying σ and τ in which some operations of expansion occur is at least of length 2 (it must contain at least one substitution, since the expansion generates new type variables) and it is always equivalent to a chain composed by a single substitution.

In the case $\sigma\approx\omega$, $UNIFY(\sigma,\tau)=s$, where $s=\{\langle\varphi,\omega\rangle|\varphi$ occurs in $\tau\}$. Obviously s is the minimal chain, since every chain unifying σ and τ is such that $c(\sigma)\approx c(\tau)\approx\omega$.

In the case $\sigma=\sigma_1\wedge\sigma_2$ and $\tau=\tau_1\wedge\tau_2$, the proof follows directly from the induction hypothesis.

In the case $\sigma=\sigma_1\rightarrow\sigma_2$ and $\tau=\tau_1\rightarrow\tau_2$,if c is such that $c(\sigma)=c(\sigma_1)\rightarrow c(\sigma_2)$ and $c(\tau)=c(\tau_1)\rightarrow c(\tau_2)$, the proof follows directly from the induction hypothesis. In the case $c(\sigma)\approx c(\tau)\approx\mu\wedge\rho$, by Theorem 2.ii), there exists $c_1=e.c_2$,where $e=\langle\sigma\wedge\tau\rangle$, such that $c_1(\sigma)\approx c_1(\tau)$ and $c==c_1$. So c_2 is a proper unifying chain for $e(\sigma)$ and $e(\tau)$, and, by induction (since $l(c_2)<l(c_1)$), $UNIFY(e(\sigma),e(\tau))=c'$,and $c_2==c'.c''$, for some c". So $c_1==e.c'.c''==c$, and the proof is given, since $UNIFY(\sigma,\tau)=e.c'$.

Consider now the case $\sigma=\sigma_1\wedge\sigma_2$ and $\tau=\tau_1\rightarrow\tau_2$.

If there exists a proper chain c unifying σ and τ, c must contain an operation of total expansion with respect to τ. Let $e=\langle\tau\rangle$; by Theorem2.ii) there exists c_1 such that $c_1=e.c_2$ and $c_1(\sigma)\approx c_1(\tau)$ and $c_1==c$. Then c_2 is a proper chain unifying $e(\sigma)$ and $e(\tau)$, and $l(c_2)<l(c)$. So, by induction hypothesis, $UNIFY(e(\sigma),e(\tau))=c'$, where $c_2==c'.c''$,for some c" and $c==c_1==e.c'.c''$. Then the proof is given, since $UNIFY(\sigma,\tau)=e.c'$.

□

Moreover, the semi-algorithm UNIFY is conservative with respect to Robinson's unification algorithm R. More precisely:

Property 1. 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 contains only pairsof the shape:$\langle\varphi_i,\omega\rangle$, so $s(\sigma)\approx s(\tau)\approx\omega$.

Proof. Easy.

□

4.Principal pairs.

Let us introduce the notion of *principal pair*,as defined in [Ronchi et. al,1984]. First of all, the notion of approximant of a term must be introduced.

Definition 11.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 variable 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 $A_1,\dots,A_p\in\mathcal{N}(p\geq 0)$, then $xA_1\dots A_p\in\mathcal{N}$

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

iv) the type assignment rules of Definition 4 are generalized to elements of $\mathcal{N}$ simply by adjoining the following rule:

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

The following theorem holds:

Theorem 4.[Ronchi et al.,1984]. $\langle B,\tau\rangle$ 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 ~) as follows:

Definition 12. Let $A\in\mathcal{N}$

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

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

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

1) if x occurs in A', $pp(A)\sim\langle B'-\{\delta x\},\delta\to\pi'\rangle$,where δ 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 $A=xA_1\dots A_n$ 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 $pp(A)\sim\langle U_{1\leq i\leq n}B_i\cup\{\pi_1\to\dots\to\pi_n\to\varphi\},\varphi\rangle$, where φ is a type variable which does not occurs in $B_i,\pi_i(1\leq i\leq n)$.

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.

The principal pair of an approximate normal form A has the property that if $\langle B,\delta\rangle$ is suitable for A ,then there exists a chain c of operations of substitution, expansion and rise, such that $\langle B,\delta\rangle\sim c(pp(A))$, where the operation of rise is defined as follows:

Definition 13.A *riser* is a pair of pairs $\langle\langle B_1,B_2\rangle,\langle\varrho_1,\varrho_2\rangle\rangle$, where $\varrho_1\leq\varrho_2$ and B_2 is such

that, for every $\sigma x \in B_1$, there exists $\sigma' x \in B_2$ with $\sigma' \leq \sigma$. Then:

i) $r(\tau)$= if $\tau \sim \varrho_1$ then ϱ_2 else τ

$r(B)$= if $\forall x.\sigma_i x \in B$ $(1 \leq i \leq n)$ and $\tau_j x \in B_1$ $(1 \leq j \leq m) \Rightarrow \sigma_1 \wedge \ldots \wedge \sigma_n \sim \tau_1 \wedge \ldots \tau_m$, then

B_2 else B

ii) $r(\langle B,\tau\rangle)=\langle r(B),r(\tau)\rangle$.

In [Ronchi et al.,1984] it is proved that the operation of rise preserves suitable pairs: namely, if $\langle B,\tau\rangle$ is such that there exists $D:B \vdash \tau X$, and r is any rise, there exists a deduction $D':r(B) \vdash r(\tau)$, and D' is obtained from D by adjoining to D some applications of rule $(\leq)$.

Let $\Pi(X)=\{\langle B,\pi\rangle | \exists A \in \mathcal{A}(X).\langle B,\pi\rangle \sim pp(A)\}$, and $\mathcal{P}=\{\langle B,\pi\rangle | \exists A \in \mathcal{N}\langle B,\pi\rangle \sim pp(A)\}$.

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

$\langle B,\pi\rangle \leq_\omega \langle B',\pi'\rangle \Leftrightarrow \exists \psi_1 \ldots \psi_n \,.\, \langle B,\pi\rangle = \langle B'[\psi_1/\omega,\ldots,\psi_n/\omega],\pi'[\psi_1/\omega,\ldots,\psi_n/\omega]$.

Property 2. $\mathcal{P},\leq_\omega$ is a meet semilattice isomorphic to $\mathcal{N},\leq$.

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

Theorem 5. i) $\mathcal{A}(X)$ is finite. $\langle B,\pi\rangle = \sqcup\Pi(X)$ is such that, if $\langle B',\tau\rangle$ is suitable for X, then there exists a chain c such that $\langle B',\tau\rangle \sim c(\langle B,\pi\rangle)$.

ii) $\mathcal{A}(X)$ 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 chain c.

□

In order to compute the principal pair of a term X, if it exists, the semi-algorithm PP will be shown. PP uses the unification semi-algorithm UNIFY, defined in the preceding section. In this semi-algorithm, the operation ∇ between basis schemes, with at least one statement on every subject, is used. ∇ is so defined:

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

Semi-algorithm PP.

$PP(X)=\langle B,\pi\rangle$ (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 $PP(X')=\langle B',\pi'\rangle$ then

if B' contains a premise on x, let σx, then $\langle B,\pi\rangle = \langle B'-\{\sigma x\},\sigma \to \pi'\rangle$ else

$\langle B',\omega \to \pi'\rangle$.

3) if $X=X_1X_2$ then

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

if $UNIFY(\pi_1,\pi_2 \to \varphi)=c$ (φ is a fresh variable) then

$\langle B,\pi\rangle = \langle c(B_1) \nabla c(B_2), c(\varphi)\rangle$.

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

Theorem 6. $PP(X)=\langle B,\pi\rangle \Leftrightarrow$ X is strongly normalizing and $pp(X)\sim\langle B,\pi\rangle$.

The proof will be given in the following section.

It is possible to define a set of unification algorithms $UNIFY_i$ ($i\geq 0$), each one unifying with ω, at every step, 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 14. 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 occurs in σ then $d(o(\tau),\sigma)$ is undefined

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

iii) if either $\sigma=\sigma_1\rightarrow\sigma_2$ 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 $UNIFY_i$.

$UNIFY_i(\sigma,\tau)=U_i(\sigma,\sigma,\tau,\tau,0)$ where

$U_i(\sigma,\sigma',\tau,\tau',j)=c$ where

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

1. if either σ or τ are either a type variable or ω then $c=U(\sigma,\sigma',\tau,\tau')$

2. if $\sigma=\sigma_1\rightarrow\sigma_2$ then

2.1. if $\tau=\tau_1\rightarrow\tau_2$ then let $c_1=U_{i-1}(\sigma_1,\sigma',\tau_1,\tau',j+1)$ and

$c_2=U_{i-1}(c_1(\sigma_2),c_1(\sigma'),c_1(\tau_2),c_1(\tau'),j+1)$ then $c=c_1.c_2$

2.2. if $\tau=\tau_1\wedge\tau_2$ then

let $e=\langle\sigma\rangle$, then let $c_1=U_i(e(\sigma'),e(\sigma'),e(\tau'),e(\tau'),0)$

then $c=e.c_1$

3. if $\sigma=\sigma_1\wedge\sigma_2$ then

3.1. if $\tau=\tau_1\rightarrow\tau_2$ then

let $e=\langle\tau\rangle$, then let $c_1=U_i(e(\sigma'),e(\sigma'),e(\tau'),e(\tau'),0)$

then $c=e.c_1$

3.2. if $\tau=\tau_1\wedge\tau_2$ then

let $c_1=U_{i-1}(\sigma_1,\sigma',\tau_1,\tau',j+1)$ and $c_2=U_{i-1}(c_1(\sigma_2),c_1(\sigma'),c_1(\tau_2),c_1(\tau'),j+1)$

then $c=c_1.c_2$.

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

Theorem 7.i) $PP_i(X)=\langle B,\pi\rangle \Rightarrow \langle B,\pi\rangle\in\Pi(X)$.

ii) $\langle B,\pi\rangle\in\Pi(X) \Rightarrow \exists i.PP_i(X)=\langle B_i,\pi_i\rangle$ and $\langle B,\pi\rangle\subseteq_\omega\langle B_i,\pi_i\rangle$.

Proof. Immediate from Theorem 5 and from the definition of the approximants of a term.

□

4.Proof of Theorem 6.

(⇐)By induction on the structure of X.
For X variable, obvious. For $X=\lambda x.X'$ or $X=YZ$, where Y does not reduce to $\lambda x.Y'$,for some Y', the proof follows directly from the induction hypothesis.
For $X=(\lambda x.Y)Z$, $PP(X) =\langle B,\pi\rangle\Rightarrow PP(\lambda x.Y)=\langle B_1,\sigma\to\tau\rangle$ and $PP(Z)=\langle B_2,\pi_2\rangle$ and $\langle B,\pi\rangle=\langle c(B_1)\nabla c(B_2),c(\varphi)\rangle$, where $c=UNIFY(\sigma\to\tau,\pi_2\to\varphi)$. It easy too see , by examining the semi-algorithm PP, that, if $\rho\sim\sigma\to\tau$ and $\delta\sim\pi_2\to\varphi$ and $UNIFY(\rho,\delta)=c'$, $\langle B,\pi\rangle\sim\langle c'(B_1)Uc'(B_2),c'(\varphi)\rangle$. Then the proof is given by induction on the normal forms of $\lambda x.Y$ and Z, which exist since X is strongly normalizing by hypothesis.
(⇒) Let B be a basis scheme, α be a type scheme $\neq\omega$,and M be a λ- term.
Let us define the predicate:
$P(B,\alpha,M)\Leftrightarrow$PP(M) is defined and $\exists c.\langle B,\alpha\rangle\sim c(PP(M))$ and M is strongly normalizing.
Let $x\vec{M}$ denote $xM_1...M_n$, for $n\geq 0$, and let FV(M) denote the set of variables occurring free in M.
Property 3.i) $P(B,\alpha\to\beta,x\vec{M})$ and $P(B',\alpha,N) \Rightarrow P(BUB',\beta,x\vec{M}N)$.
ii) $P(BU\{\alpha x\},\beta,Mx)$ and $x\notin FV(M)$ and B does not contain premises on x ⇒
$P(B,\alpha\to\beta,M)$.
iii) $P(B,\sigma_1\wedge\sigma_2,M) \Rightarrow P(B,\sigma_1,M)$ and $P(B,\sigma_2,M)$.
iv) $P(B,\sigma,M)$ and $\sigma\leq\tau \Rightarrow P(B,\tau,M)$.
Proof. i) Let $x\vec{M}=xM_1...M_m$. $P(B',\alpha,N)\Rightarrow PP(N)=\langle\underline{B'},\pi'\rangle$ and $\exists c.\langle B',\alpha\rangle\sim c(PP(N))$. $P(B,\alpha\to\beta,x\vec{M})\Rightarrow PP(x\vec{M})=\langle\{\pi_1\to...\to\pi_m\to\varphi x\}\nabla B_1\nabla...\nabla B_m,\varphi\rangle$, where φ is a fresh variable and $PP(M_i)=\langle B_i,\pi_i\rangle$, and $\exists c'.\langle B,\alpha\to\beta\rangle\sim c'(PP(x\vec{M}))$. So $UNIFY(\varphi,\pi'\to\psi)=s$, where $s=\{\langle\varphi,\pi'\to\psi\rangle\}$ (ψ is fresh),which implies $PP(x\vec{M}N)=\langle\{\pi_1\to...\to\pi_m\to\pi'\to\psi x\}\nabla B_1\nabla...\nabla B_m\nabla\underline{B'},\psi\rangle$.Let $c''=c.c'.s'$, where $s'=\{\langle\psi,\beta\rangle\}$. $\langle BUB',\beta\rangle\sim c''(PP(x\vec{M}N))$,since PP(N) and $PP(x\vec{M})$ are disjoint, then $P(BUB',\beta,x\vec{M}N)$.
ii) $P(BU\{\alpha x\},\beta,Mx)\Rightarrow PP(M)=\langle B_1,\pi_1\rangle$ and $PP(x)=\langle\{\varphi x\},\varphi\rangle$ and $UNIFY(\pi_1,\varphi\to\psi)=c'$ and $PP(Mx)=\langle c'(B_1)\nabla\{c'(\varphi)x\},c'(\psi)\rangle$ where ψ is fresh and B_1 does not contain premises on x, and moreover $\exists c.\langle BU\{\alpha x\},\beta\rangle\sim c(\langle c'(B_1)\nabla\{c'(\varphi)x\},c'(\psi)\rangle$.Note that c' does not contain any expansion involving the type variable ψ.
Then $\langle B,\alpha\to\beta\rangle=c(\langle c'(B_1),c'(\varphi)\to c'(\psi)\rangle)$ (since B_1 does not contain premises on

$x)=c(<c'(B_1),c'(\pi_1)>)=c'.c(<B_1,\pi_1>)$, and then $P(B,\alpha\to\beta,M)$.

iii) and iv) are immediate. □

Then define, by induction on the structure of type schemes $\neq\omega$, the following computability predicate:

$Comp(B,\varphi,M) \Leftrightarrow P(B,\varphi,M)$
$Comp(B,\sigma\to\tau,M) \Leftrightarrow (Comp(B',\sigma,N)\Rightarrow Comp(BUB',\tau,MN))$
$Comp(B,\sigma_1\wedge\sigma_2,M) \Leftrightarrow Comp(B,\sigma_1,M)$ and $Comp(B,\sigma_2,M)$.

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

Lemma 1. i) $P(B,\sigma,x\vec{M}) \Rightarrow Comp(B,\sigma,x\vec{M})$.
ii) $Comp(B,\sigma,M) \Rightarrow P(B,\sigma,M)$.

Proof. i) and ii) by simultaneous induction on σ.
σ is a type variable. i) and ii) follow from the definition of Comp.
$\sigma=\alpha\to\beta$.
i) $Comp(B',\alpha,N)\Rightarrow P(B',\alpha,N)$ (by induction hypothesis).
$P(B,\alpha\to\beta,x\vec{M})$ and $P(B',\alpha,N) \Rightarrow P(BUB',\beta,x\vec{M}N)$ (by Property 3) $\Rightarrow Comp(BUB',\beta,x\vec{M}N)$ (by induction hypothesis). Then $Comp(B',\alpha,N)$ and $Comp(BUB',\beta,x\vec{M}N) \Rightarrow Comp(B,\alpha\to\beta,x\vec{M})$ (by def. of Comp).
ii) Let $x\in FV(M)$ and let $\{\alpha x\}\in B'$. $P(B',\alpha,x) \Rightarrow Comp(B',\alpha,x)$ (by induction).
$Comp(B,\alpha\to\beta,M)$ and $Comp(B',\alpha,x) \Rightarrow Comp(BUB',\beta,Mx)$ (by definition) $\Rightarrow P(BUB',\beta,Mx)$ (by induction) $\Rightarrow P(B,\alpha\to\beta,M)$ (by Property 3).
$\sigma=\sigma_1\wedge\sigma_2$.
ii) by definition of Comp and by induction hypothesis.
i) $P(B,\sigma_1\wedge\sigma_2,x\vec{M}) \Rightarrow P(B,\sigma_1,x\vec{M})$ and $P(B,\sigma_2,x\vec{M})$ (by Property 3)$\Rightarrow Comp(B,\sigma_1,x\vec{M})$ and $Comp(B,\sigma_2,x\vec{M})$ (by induction) $\Rightarrow Comp(B,\sigma_1\wedge\sigma_2,x\vec{M})$ (by definition). □

Lemma 2. Let $\{x_1,\dots,x_m\}\supset FV(M)$, and let B be such that $\{\sigma_i x_i\}\in B$ $(1\le i\le m)$.$Comp(B',\sigma_i,N_i)$ $(1\le i\le m)$ and $PP(M)=<\underline{B},\pi>$ and $\exists c.<B,\tau>\sim c(<\underline{B},\pi>) \Rightarrow Comp(BUB',\tau,M[x_i/N_i])$.

Proof. By induction on M. The only not trivial case is $M=\lambda x.M'$.
Then $PP(M)=<\underline{B},\alpha\to\beta> \Rightarrow PP(M')\sim<\underline{B}U\{\alpha x\},\beta>$, where $\underline{B}$ does not contain premises on x.
$PP(M')\sim<\underline{B}U\{\alpha x\},\beta>$ and $\exists c.<BU\{\sigma x\},\tau>\sim c(<\underline{B}U\{\alpha x\},\beta>)$ and $Comp(B'',\sigma,N)$ and $Comp(B',\sigma_i,Ni)$ $(1\le i\le m) \Rightarrow Comp(BU\{\sigma x\}UB'UB'',\tau,M'[x/N,x_i/N_i])$ $(1\le i\le m)$ (by induction) $\Rightarrow Comp(BU\{\sigma x\}UB'UB'',\tau,(\lambda x.M'[x_i/N_i])N)$ (since Comp is invariant under β-convertibility) $\Rightarrow Comp(BUB',\sigma\to\tau,\lambda x.M'[x_i/N_i])$ (by definition).

□

Then let PP(M) be defined, and let $\{x_1,\dots,x_m\}=FV(M)$, and let $<B,\tau>\sim c(PP(M))$ and let $\{\sigma_i x\}\in B$ $(1\le i\le m)$. Then $Comp(B,\sigma_i,x_i)$. By Lemma 2, this implies $Comp(B,\tau,M)$, which implies $P(B,\tau,M)$, by Lemma 1.ii). □

5. The intersection type discipline without ω.

In [Coppo et al.,1980 a], for the first time an intersection type discipline was introduced, built from a set of type variables, without any constant. More precisely, the type schemes are defined as in Definition 1 ,without the constant ω, and the assignment rules are as in Definition 4, without the rule (ω). The definition of pairs, equivalence relation ~, and operations of pairs remain unchanged. It is possible to define a principal pair in this discipline, in the following way:

Definition 15. Let X be a normal form. pp'(X) is so defined:

i)if X=x then pp'(x)~<{φx},φ> where φ is a fresh type variable

ii) if X=λx.X' and pp'(X')~<B',π'>, then:

1) if x occurs in X' then pp'(X)~<B'-{σx},σ→π'>,where σ is the intersection of the predicates in B' whose subject is x

2) otherwise pp'(X)~<B',φ→π'>, where φ is fresh.

iii) if $X=xX_1...X_n$ and $pp'(X_i)\sim\langle B_i,\pi_i\rangle$ then $pp'(X)\sim\langle U_{1\le i\le n}B_i\cup\{\pi_1\to...\to\pi_n\to\varphi x\},\varphi\rangle$, where φ is fresh.

The proof that pp'(X) is really the principal pair of X, in the sense that all and only the type scheme deducible for X are obtained from pp'(X) by means of chains of substitutions, expansions and rise, is a particular case of the proof that, for $A\in\mathcal{N}$, pp(A) is the principal pair of A, given in [Ronchi et al.,1984]. Moreover an algorithm PP' can be define, which differs from PP only in the point 2), which must be replaced by:

2) if X=λx.X' then

if PP'(X')=<B',π'> then

if B' contains a premise on x, let σx, then PP'(X)=<B'-{σx},σ→π'>

else PP'(X)=<B',φ→π'>, where φ is a fresh type variable.

Then we obtain, as corollary of Theorem 6, the following:

Theorem 8. In the intersection type discipline without the constant ω, there exists a pair suitable for X iff X is strongly normalizing.

□

This result is stated, but not proved, in [Coppo et al., 1980 a].

Aknoledgments. The author is very grateful to Paolo Bussa and Mauro Berta, who gave an essential contribution in designing and implementing the semi-algorithms UNIFY and PP.

References.

[Barendregt,1984] **Barendregt H.**, The Lambda Calculus: its syntax and semantics, North Holland, (Amsterdam).

[Barendregt et al.,1981] **Barendregt H, Coppo M.,Dezani M.**, A filter λ-model and the completeness of type assignment,Journal of Symbolic Logic,84,4.

[Coppo et al.,1980 a] **Coppo M., Dezani M.**, An extension of the basic Functionality Theory for the L-calculus, Notre Dame Journal of Formal Logic, 21,4.

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

[Curry et al.,1958] **Curry H.B., Feys R.**, Combinatory Logic, vol.1, Nort Holland (Amsterdam).

[Hindley,1969] **Hindley R.**,The principal type scheme as an object in combinatory logic, Trans. Amer. Math. Soc.,146.

[Hindley,1983] **Hindley R.**, The completeness theorem for typing λ-terms, Theoretical Computer Science, 22.

[Milner,1978] **Milner R.**, A theory of type polimorphism in programming, J. Comput. System Sci.,17.

[Milner et al.,1982] **Milner R., Damas L.**,Principal type schemes for functional programs, 9-th Symp. on Principle of programming languages.

[Robinson,1965] **Robinson J.A.**, A machine oriented logic based on the resolution principle, Journal of ACM, 12.

[Ronchi et al.,1984] **Ronchi Delle Rocca S., Venneri B.**, Principal type scheme for an extended type theory, Theoretical Computer Science, 28.