Characterization of typings in polymorphic type discipline

Paola Giannini, Simona Ronchi Della Rocca

Dipartimento di Informatica-Universita' di Torino
Corso Svizzera 185, 10139 TORINO, ITALY

Abstract

The polymorphic type discipline for λ-calculus is an extension of the classical Curry's functionality theory, in which types can be universally quantified. In this paper we present an algorithm that, given a term M, build a "set of constraints" such that M can be typed if and only if this set of constraints is satisfied. Moreover all the typings for M (if any) are built from the set of constraints by means of substitutions. Using the set of constraints some properties of the polymorphic type discipline are proved.

1.Introduction

The polymorphic type discipline for λ-calculus is an extension of classical Curry's functionality theory [Curry 69], in which types can be universally quantified. In this paper, we study the type assignment system for deriving quantified types for λ-terms: a typing for a λ-term M is a statement $B \vdash_{\forall} M:\varphi$, where B is a type assignment (a partial function from λ-variables to types) and φ is a type. In [Leivant 83] an isomorphism is proved between typings in this system and terms of the second order λ-calculus (see [Girard 72] and [Reynolds 74]). Our aim is to characterize, given a term M, all the typings for M (if any). In Curry's type discipline all the typings for a term M are instances of a particular typing for M, the principal one, and there is a decision procedure building the principal typing for M, if it exists (see [Hindley 69] and [Damas and Milner 82]). In the polymorphic type discipline, it can be shown that a principal typing in this sense does not exist. In fact consider the term $\lambda x.xx$; both $(\forall t.t) \rightarrow \forall t.t$ and $(\forall t.t \rightarrow t) \rightarrow \forall t.t \rightarrow t$ are types derivable for it, but it can be easily proved that there is no type φ derivable for $\lambda x.xx$ such that both the considered types are instances of φ [Coppo 85]. Moreover the problem of whether the set of terms having a type in this discipline is recursive is still open.

Starting from an idea of Mitchell [Mitchell 87], we first define a new type assignment system, which is equivalent to the given one, and has the important feature that all the type derivations for a term M have a fixed structure, depending only from the structure of M. Then, we introduce an algorithm, using this property as an essential tool, which, for any term M, build a "set of constraints" such that M can have a typing if and only if the set of constraints can be satisfied. Moreover, if the set of constraints for M can be satisfied, it is possible to build from it, by means of substitutions, all the typings for M. So the problem of whether a term can be typed is reduced to the problem of whether some particular "set of constraints" can be satisfied. This characterization of typings allows also to prove some properties of the polymorphic type discipline. For example, we prove that there exists a strongly normalizing term having no type.

2.Polymorphic Types and Containement

Let x, y, z and w range over *λ-variables*, and M, N, P, Q and R range over *λ-terms*. The terms of untyped λ-calculus are defined by the following grammar.

$$M ::= x \mid MM \mid \lambda x.M$$

Λ is the set of untyped λ-terms, $FV(M)$ denotes the set of the free variables of M.

Let a, b, c and d range over *type variables*, and φ, χ, ψ and ω range over *type expressions* (*types*). The set of type expressions is defined by the following grammar.

$$\varphi ::= a \mid \varphi \rightarrow \varphi \mid \forall a.\varphi$$

$\equiv$ denotes syntactic equality of types modulo α-conversion (renaming of quantified type variables). T is the set of types and $T^- = \{\varphi \in T \mid$ for no a and $\psi \in T$ $\varphi \equiv \forall a.\psi\}$ is the set of *non externally quantified types*. $\varphi_1 \rightarrow \varphi_2 \rightarrow \cdots \varphi_{n-1} \rightarrow \varphi_n$ is an abbreviation for $(\varphi_1 \rightarrow (\varphi_2 \rightarrow \cdots (\varphi_{n-1} \rightarrow \varphi_n)$ and $\forall a_1 a_2 \cdots a_n.\varphi$ is an abbreviation for $\forall a_1.\forall a_2.\ldots.\forall a_n.\varphi$.

CH2608-8/88/0000/0061$01.00 © 1988 IEEE

A *type assignment* A is a partial function from λ-variables to types with finite domain.

Notation. (i) Let $\bar{a}$ denote a (possibly empty) string of variables and $\bar{\varphi}$ denote a (possibly empty) sequence of types. $a \in \bar{a}$ denotes that a type variable a occurs in the string $\bar{a}$.
(ii) Let $\bar{a}$ be the string $a_1 a_2 \cdots a_n$, where $n \geq 0$, and let φ be a type. $\forall \bar{a}.\varphi$ is an abbreviation for $\forall a_1.\forall a_2.\ldots.\forall a_n.\varphi$ (if $n=0$ $\forall \bar{a}.\varphi$ is equal to φ).
Let φ and ψ be types, and a be a type variable. $[\varphi/a]\psi$ denotes the type obtained from ψ replacing each occurrence of a with φ. In the replacement bound variables of φ and ψ can be renamed to avoid collision with free variables. Let $\bar{\varphi}$ be the sequence

$$\varphi_1, \varphi_2, \ldots, \varphi_n$$

and let $\bar{a}$ be $a_1 a_2 \cdots a_n$: both

$$[\bar{\varphi}/\bar{a}]\psi \text{ and } [\varphi_1/a_1, \varphi_2/a_2, \ldots, \varphi_n/a_n]\psi$$

denote the type obtained by the simultaneous replacement of the variable a_i by φ_i $(0 \leq i \leq n)$. Let A be a type assignment; $[\bar{\varphi}/\bar{a}]A$ is the function mapping any x to $[\bar{\varphi}/\bar{a}]A(x)$.
(iii) Let g be a (partial) function from E to F, $domain(g)=\{e \in E \mid g(e) \text{ is defined}\}$. If $domain(g)=\{e_1, e_2, \ldots, e_n\}$ and $g(e_i)=f_i$, g can be denoted by either

$$[e_1 \Rightarrow f_1, e_2 \Rightarrow f_2, \ldots, e_n \Rightarrow f_n] \text{ or } \Sigma_{1 \leq i \leq n} [e_i \Rightarrow f_i].$$

An *extension* of g is a partial function that coincides with g on $domain(g)$. Let $e_1, e_2, \ldots, e_n$ be distinct elements of E, and $f_1, f_2, \ldots, f_n$ be elements of F; both

$$g+[e_1 \Rightarrow f_1, e_2 \Rightarrow f_2, \ldots, e_n \Rightarrow f_n] \text{ and } g+\Sigma_{1 \leq i \leq n} [e_i \Rightarrow f_i]$$

denote the function that coincides with g on all the elements of $E-\{e_1, e_2, \ldots, e_n\}$ and that for all i, $1 \leq i \leq n$, maps e_i into f_i. If C is a subset of E, g-C is a function that coincides with g on all the elements of E that do not belong to C, and it is undefined on the elements of C. The everywhere undefined function is denoted by $\varnothing$. ◊

Let A be a type assignment, φ be a type, a be a type variable, and M be a λ-term; a is *bindable* in M with respect to A if a is not free in any type $A(x)$ for any free variable x of M. We can derive from A a type φ for M, if $A \vdash_\forall M:\varphi$ is derivable in the following system.

Polymorphic type assignment system

(var) $A \vdash_\forall x: A(x)$

$$(\rightarrow\text{I}) \quad \frac{A+[x \Rightarrow \varphi] \vdash_\forall M:\psi}{A \vdash_\forall \lambda x.M:\varphi \rightarrow \psi}$$

$$(\rightarrow\text{E}) \quad \frac{A \vdash_\forall M:\varphi \rightarrow \psi \qquad A \vdash_\forall N:\varphi}{A \vdash_\forall MN:\psi}$$

$$(\forall\text{I}) \quad \frac{A \vdash_\forall M:\varphi}{A \vdash_\forall M:\forall a.\varphi} \; (*) \qquad (\forall\text{E}) \quad \frac{A \vdash_\forall M:\forall a.\varphi}{A \vdash_\forall M: [\psi/a]\varphi}$$

(*) a is bindable in M with respect to A.

$\vdash_\forall M:\varphi$ stands for $\varnothing \vdash_\forall M:\varphi$.

For Curry's system, the property that every term having a type has a principal type derives directly from the property that every derivation of a type for a given term M has a structure depending only on the structure of M, i.e., it is "syntax directed". The type assignment system described above is not syntax directed, since applications of the rules (∀E) and (∀I) can occur at any point of the derivation.
We now present a new type assignment system, which is syntax directed and will be proved to be equivalent to the previous one. First let us define a containement relation between types, which is the relation introduced in [Damas and Milner 82].

Definition 1. $\forall \bar{a}.\varphi \leq [\bar{\chi}/\bar{a}]\varphi$. ◊

Remark 2. Clearly the relation $\leq$ is decidable. Moreover $\leq$ is reflexive and transitive, and, if

$$\forall \bar{a}.\ \varphi \rightarrow \psi \ \leq \varphi' \rightarrow \psi',$$

then $\forall \bar{a}.\varphi \leq \varphi'$ and $\forall \bar{a}.\psi \leq \psi'$. In particular there is a sequence of types $\bar{\chi}$ such that $[\bar{\chi}/\bar{a}]\varphi \equiv \varphi'$ and $[\bar{\chi}/\bar{a}]\psi \equiv \psi'$. However, the converse is not true. That is, $\forall \bar{a}.\varphi \leq \varphi'$ and $\forall \bar{a}.\psi \leq \psi'$ do not imply $\forall \bar{a}.\ \varphi \rightarrow \psi \ \leq \varphi' \rightarrow \psi'$. ◊

Containement type assignment system

$$(\text{var} \leq) \quad \frac{A(x) \leq \chi}{A \vdash x:\forall \bar{a}.\chi \quad (*)}$$

$$(\rightarrow\text{I}) \quad \frac{A+[x \Rightarrow \varphi] \vdash M:\psi}{A \vdash \lambda x.M:\forall \bar{a}.\ \varphi \rightarrow \psi \quad (*)}$$

$$(\rightarrow E)\ \frac{A \vdash M:\varphi\rightarrow\psi \quad A \vdash N:\varphi \quad \psi\leq\chi}{A \vdash MN:\forall\bar{a}.\chi \quad (*)}$$

(*) the type variables in $\bar{a}$ are bindable in the term w r.t. A.

Theorem 3. $A \vdash_{\forall} M:\varphi$ if and only if $A \vdash M:\varphi$.

Proof. As we can see from the definition of $\leq$, an application of the rule $(\rightarrow E)$ in $\vdash$ is equivalent to an application of the rule $(\rightarrow E)$ in $\vdash_{\forall}$ followed by a sequence of applications of the rule $(\forall E)$ followed by a sequence of applications of $(\forall I)$. (Similarly for (var $\leq$) .) Moreover an application of the rule $(\rightarrow I)$ in $\vdash$ is equivalent to an application of the rule $(\rightarrow I)$ in $\vdash_{\forall}$ followed by a sequence of applications of the rule $(\forall I)$. So the proof of the implication $(\Leftarrow)$ is immediate. The proof of the implication $(\Rightarrow)$ follows directly from the normalization property for second order deductions (w.r.t. the connective $\forall$) given in [Prawitz 65], which states that a derivation can always be transformed into an equivalent one, in which applications of the rule $(\forall E)$ never follow applications of the rule $(\forall I)$. ◊

The following property clarifies that the containement system is syntax directed.

Property 4. (i) $A \vdash \lambda x.M:\varphi$ if and only if $\varphi\equiv\forall\bar{a}.\ \psi\rightarrow\chi$ for some $\bar{a}$, ψ and χ,and $A+[x\Rightarrow\psi] \vdash M:\chi$, where the type variables in $\bar{a}$ are bindable for $\lambda x.M$ in A.
(ii) $A \vdash MN:\forall\bar{a}.\chi$ if and only if,for some ψ and φ,$A \vdash N:\psi$, $A \vdash M:\psi\rightarrow\varphi$ and $\varphi\leq\chi$.
Proof. By induction on derivations. ◊

Note. Mitchell in [Mitchell 87] defines a containement relation similar to the one given in definition 1. More precisely he defines

$$\forall\bar{a}.\varphi\leq\forall\bar{b}.[\bar{\chi}/\bar{a}]\varphi$$

where $\bar{b}$ are not free in $\forall\bar{a}.\varphi$. Moreover Mitchell introduces the following type assignment system:

$$(\text{var})\quad A \vdash_M x:A(x)$$

$$(\rightarrow I)\ \frac{A+[x\Rightarrow\varphi] \vdash_M M:\psi}{A \vdash_M \lambda x.M:\forall\bar{a}.\ \varphi\rightarrow\psi \quad (*)}$$

$$(\rightarrow E)\ \frac{A \vdash_M M:\forall\bar{a}.\ \varphi\rightarrow\psi \quad A \vdash_M N:\forall\bar{a}.\varphi}{A \vdash_M MN:\forall\bar{a}.\psi}$$

$$(\leq)\ \frac{A \vdash_M M:\varphi \text{ and } \varphi\leq\psi}{A \vdash_M M:\psi}$$

(*) the type variables in $\bar{a}$ are bindable in $\lambda x.M$ w r.t. A

and proves that this system and the system $\vdash_{\forall}$ are equivalent. It can be easily seen that Mitchell containement relation synthesizes a sequence of applications of the rule $(\forall E)$ followed by a sequence of applications of the rule $(\forall I)$. We could use this containement relation instead of the one introduced in definition 1, but the latter simplifies the construction of the principal type scheme system of a term defined in the following section. ◊

Remark 5. It is immediate to verify that, in the containement type assignment system (as previously defined), we can always assume, without loss of generality, that $\chi\in T^{-}$. In fact, for any φ and ψ

$$\forall\bar{a}\bar{b}.\varphi\leq\forall\bar{b}.[\bar{\psi}/\bar{a}]\varphi$$

implies

$$\forall\bar{a}\bar{b}.\varphi\leq[\bar{\psi}/\bar{a},\bar{b}/\bar{b}]\varphi.$$

Therefore if $A \vdash M:\forall\bar{b}.[\bar{\psi}/\bar{a}]\varphi$ then $A \vdash M:\ [\bar{\psi}/\bar{a},\bar{b}/\bar{b}]\varphi$. ◊

We can represent a derivation in $\vdash$ by a sequence of containements as follows.

Definition 6. Let D be a derivation of $A \vdash M:\varphi$. The *characteristic* of D, $C(D)$, is the sequence

$$\varphi_1\leq\psi_1,\varphi_2\leq\psi_2,\ldots,\varphi_n\leq\psi_n$$

such that $\varphi_i\leq\psi_i$ occurs in an application of either the rule $(\rightarrow E)$ or (var $\leq$) and precedes $\varphi_{i+1}\leq\psi_{i+1}$ in the *right postorder* visit of the tree D. (The right postorder visit of a tree is the sequence obtained by first right postorder visiting its subtrees from right to left and then the root.) ◊

Property 4 insures that there is a one-to-one correspondence between the derivation $D: A \vdash M:\varphi$ and the quadruple $\langle A,M,\varphi,C(D)\rangle$. Note that the number of components of $C(D)$ depends only on M. More precisely it is the number of occurrences of variables in M plus the number of the subterms of M that are applications.

Example. Let $A\equiv[x\Rightarrow\forall a.a\rightarrow\forall b.b,y\Rightarrow c]$ and let D be the following derivation.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A+[z\Rightarrow\forall d.d]\vdash_\forall z:\forall d.d}(\text{var})}{A\vdash_\forall x:\forall a.a\to\forall b.b}(\forall E)\quad A+[z\Rightarrow\forall d.d]\vdash_\forall z:a'}{A\vdash_\forall x:((\forall d.d)\to a')\to\forall b.b}(\forall E)\quad \dfrac{}{A\vdash_\forall \lambda z.z:(\forall d.d)\to a'}(\to I)}{A\vdash_\forall x(\lambda z.z):\forall b.b}(\to E)}{A\vdash_\forall x(\lambda z.z):c\to b'}(\forall E)\quad \dfrac{}{A\vdash_\forall y:c}(\text{var})}{A\vdash_\forall (x(\lambda z.z))y:b'}(\to E)}{A\vdash_\forall (x(\lambda z.z))y:\forall b'.b'}(\forall I)$$

The corresponding derivation D' in the containement system is:

$$\dfrac{\dfrac{\dfrac{}{\forall a.a\to\forall b.b\le((\forall d.d)\to a')\to\forall b.b}(\text{var}\le)}{A\vdash x:((\forall d.d)\to a')\to\forall b.b}\quad D_1\quad \forall b.b\le c\to b'}{A\vdash x(\lambda z.z):c\to b'}(\to E)\quad \dfrac{\dfrac{}{c\le c}\quad \dfrac{}{A\vdash y:c}(\text{var}\le)\quad b'\le b'}{A\vdash (x(\lambda z.z))y:\forall b'.b'}(\to E)$$

where D_1 is the following derivation:

$$\dfrac{\dfrac{}{\forall d.d\le a'}(\text{var}\le)}{A+[z\Rightarrow\forall d.d]\vdash_\forall z:a'}$$
$$\dfrac{}{A\vdash\lambda z.z:(\forall d.d)\to a'}(\to I)$$

and the characteristic of D', $C(D')$ is the sequence:

$$\forall a.a\to\forall b.b\le\varphi,\forall d.d\le a',\forall b.b\le c\to b',c\le c,b'\le b'$$

where φ is the type $((\forall d.d)\to a')\to\forall b.b$. ◊

3. The Principal Type Scheme System Associated with a Term

Since we want to characterize the set of all types derivable for a term, we need to represent sets of quantified types having some syntactical feature in common. For this purpose we introduce type schemes. Type schemes are defined by a grammar similar to the one of types, but variables are just names for free places that can be filled by either types or (sequences of) type variables, i.e., type schemes are representations of type contexts.

Then we define an algorithm Π, which, for any term M, build a type scheme σ, a scheme assignment B (a partial function from λ-variables to type schemes) and a "type scheme system", representing the set of constraints that B and σ must satisfy in order to represent the class of all typings for M. More precisely, we prove that M has a typing if and only if its "type scheme system" has a solution and the set of typings for M is completely determined by the set of solutions of the "type scheme system" associated with M.

Let α, β, γ and δ denote the *scheme variables* and r,t,v and u denote *sequence variables*. The set Σ of *type schemes*, ranged over by ρ, σ, τ and υ is defined by the following grammar.

$$\sigma ::= \alpha \mid \forall t.\alpha \mid \sigma\to\sigma \mid \forall t.\sigma\to\sigma$$

Let $\equiv$ denote identity between type schemes. The set of free variables of a type scheme σ is $FV(\sigma)=\{\alpha,t \mid \alpha,t$ occur in $\sigma\}$, i.e., the symbol $\forall$ does not introduce bound variables. Let Σ^- be $\{\sigma\in\Sigma \mid$ for no t and $\tau\in\Sigma$ $\sigma\equiv\forall t.\tau\}$.

For type schemes we introduce two definitions of substitutions: (simple) substitutions, and scheme substitutions. Simple substitutions map type schemes into types, whereas scheme substitutions map type schemes into type schemes.

A *(simple) substitution*, s, is a partial function with finite domain from the set of scheme variables and sequence variables to the set of types and sequences of type variables, mapping a scheme variable α in a non externally quantified type $\varphi\in T^-$, and a type sequence variable t in a string of type variables $\bar{a}$. Given a type scheme $\sigma\in\Sigma$ and a simple substitution s, defined on all the scheme and sequence variables of σ, $s(\sigma)$ is a type defined by structural induction on σ as follows.

(i) If $\sigma\equiv\alpha$ ($\sigma\equiv\tau\to\tau'$), then $s(\sigma)\equiv s(\alpha)$ ($s(\sigma)\equiv s(\tau)\to s(\tau')$).

(ii) If $\sigma\equiv\forall t.\tau$ ($\sigma\equiv\forall t.\tau\to\tau'$), and $s(t)\equiv\bar{a}$, then $s(\sigma)\equiv\forall\bar{a}.s(\tau)$ ($s(\sigma)\equiv\forall\bar{a}.s(\tau\to\tau')$).

A type scheme σ can be viewed as a representation of the set of types that can be obtained from σ by applying to it a simple substitution whose domain is $FV(\sigma)$. For example:

$\forall t.\alpha$ represents the whole set T;

α represents the set T^-;

$(\forall t.\alpha)\to\forall t.\beta$ represents the set $\{\varphi\to\psi \mid \psi\in T$ and $\varphi\in T\}$.

A *scheme substitution*, S, is a total function from the set of scheme variables to Σ^- almost everywhere equal to the identity function. S is extended to type schemes as follows.

(i) If $\sigma\equiv\alpha$ ($\sigma\equiv\tau\to\tau'$), then $S(\sigma)\equiv S(\alpha)$ ($S(\sigma)\equiv S(\tau)\to S(\tau')$).

(ii) If $\sigma \equiv \forall t.\tau$ ($\sigma \equiv \forall t.\tau \rightarrow \tau'$), then $S(\sigma) \equiv \forall t.S(\tau)$ ($S(\sigma) \equiv \forall t.S(\tau \rightarrow \tau')$).

Let s and s' be simple substitutions and V be a set of scheme and sequence variables. $s =_V s'$ (read *s is equal to s' on V*) means that for all sequence variables $t \in V$ $s(t) \equiv s'(t)$ and for all scheme variables $\alpha \in V$ $s(\alpha) \equiv s'(\alpha)$.

A *scheme assignment* B is a partial function from λ–variables to type schemes with finite domain. For a scheme assignment B we define $FV(B)$ to be the set $\{FV(B(x)) \mid x$ is in the domain of $B\}$. Given a simple substitution s and a scheme assignment B, $s(B)$ is the type assignment with the same domain of B, and such that $s(B)(x) = s(B(x))$. A scheme substitution S transforms a scheme assignment B into the scheme assignment SB that maps any x into $S(B(x))$.

Definition 7. (i) A *type scheme system* is a pair $\langle G,F \rangle$ where G is a sequence of inequalities between type schemes

$$\sigma_1 \leq \tau_1, \sigma_2 \leq \tau_2, \ldots, \sigma_n \leq \tau_n$$

and F is a partial function from type sequence variables to $\wp_{fin}(\Sigma)$ (finite sets of type schemes) with finite domain.

(ii) Let F and F' be partial functions from type sequence variables to $\wp_{fin}(\Sigma)$, $F \cup F'$ is defined as follows:

$$F \cup F'(t) = \begin{cases} F(t) \cup F'(t) & \text{if } F \text{ and } F' \text{ are defined on } t, \\ \text{undefined} & \text{if } F \text{ and } F' \text{ are both undefined on } t, \\ F(t)\ (F'(t)) & \text{if only } F\ (F') \text{ is defined on } t. \end{cases}$$

Let S be a scheme substitution, $S(F)$ is the partial function with the same domain of F and such that $S(F)(t)$ is equal to $\{S(\sigma) \mid \sigma \in F(t)\}$.

(iii) Let G be the sequence $\sigma_1 \leq \tau_1, \sigma_2 \leq \tau_2, \ldots, \sigma_n \leq \tau_n$; *leftside*$(G)$ is the set $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ and *rightside*(G) is the set $\{\tau_1, \tau_2, \ldots, \tau_n\}$. ◊

Let $\langle G,F \rangle$ be a type scheme system. $FV(G)$ is the set $\{FV(\sigma) \cup FV(\sigma') \mid \sigma \leq \sigma' \in G\}$ and $FV(F)$ is the set $\{FV(F(t)) \mid t$ is in the domain of $F\} \cup domain(F)$.

Definition 8. Let s be a simple substitution, the type scheme system $\langle G,F \rangle$ *is satisfied by* s if the following conditions are satisfied:

(i) s is defined on $FV(G) \cup FV(F)$,

(ii) for all t such that $F(t)$ is defined, if $s(t) = a_1 a_2 \cdots a_n$ and $n \geq 1$, then for all i, $1 \leq i \leq n$, a_i does not belong to the free variables of any type in $\{s(\sigma) \mid \sigma \in F(t)\}$,

(iii) if G is $\sigma_1 \leq \tau_1, \sigma_2 \leq \tau_2, \ldots, \sigma_n \leq \tau_n$ then for all i, $1 \leq i \leq n$, $s(\sigma_i) \leq s(\tau_i)$ (where $\leq$ is the containement relation between types introduced in definition 1). ◊

Example. Let $\langle G,F \rangle$ be the type scheme system defined by

$$G = \forall t.\beta \leq \forall r.\alpha, \forall t.\beta \leq \forall t'.(\forall u.\alpha) \rightarrow \forall v.\gamma$$

and

$$F = [t \Rightarrow \{\forall r.\alpha,\ \forall t'.(\forall u.\alpha) \rightarrow \forall v.\gamma\}].$$

Let s be the type substitution:

$[t \Rightarrow a, \beta \Rightarrow a, \alpha \Rightarrow b, \gamma \Rightarrow \forall d.d \rightarrow d, r \Rightarrow \varnothing, u \Rightarrow \varnothing, t' \Rightarrow \varnothing, v \Rightarrow \varnothing]$

$\langle G,F \rangle$ is satisfied by s. In fact s is defined on all the variables occurring in G, it clearly satisfies F and the containements in $s(G)$, $\forall a.a \leq b$ and $\forall a.a \leq b \rightarrow \forall d.d \rightarrow d$, are true. ◊

We now introduce an algorithm E, that, given a λ-term M, a scheme assignment B and a type scheme σ returns the conditions that B and σ must satisfy in order to represent a class of typings for M.

Definition 9. Let M be a λ-term, B be a scheme assignment defined on all the free λ-variables of M and σ be a type scheme. $E(M,B,\sigma)$ is a pair $\langle S, \langle G,F \rangle \rangle$, where S is a scheme substitution and $\langle G,F \rangle$ is a type scheme system. E is defined by structural induction on the λ-term M as follows.

(i) If $M \equiv x$, then

if $\sigma \equiv \forall t.\tau$ then $E(M,B,\sigma)$ is $\langle \varnothing, \langle B(x) \leq \tau, [t \Rightarrow \{B(x)\}] \rangle \rangle$;

if $\sigma \in \Sigma^-$ then $E(M,B,\sigma)$ is $\langle \varnothing, \langle B(x) \leq \sigma, \varnothing \rangle \rangle$.

(ii) If $M \equiv \lambda x.M'$, then

(a) if $\sigma \equiv \forall t.\alpha$ ($\sigma \equiv \alpha$) then let S' be the scheme substitution

$$I + [\alpha \Rightarrow (\forall t.\beta) \rightarrow \forall t'.\beta']$$

where I is the identity function and t, t', β and β' are fresh variables, and let

$$E(M', S'B + [x \Rightarrow (\forall t.\beta)], \forall t'.\beta')$$

be $\langle S, \langle G,F \rangle \rangle$. $E(\lambda x.M',B,\forall t.\alpha)$ is $\langle SS', \langle G, F + [t \Rightarrow \Im] \rangle \rangle$ ($E(\lambda x.M',B,\alpha)$ is $\langle SS', \langle G,F \rangle \rangle$ where $\Im$ is $\{SS'B(y) \mid$ y is a free variable of $M\}$;

(b) if $\sigma \equiv \forall t.\tau_1 \rightarrow \tau_2$ ($\sigma \equiv \tau_1 \rightarrow \tau_2$) then let

$$E(M', B + [x \Rightarrow \tau_1], \tau_2)$$

be $\langle S, \langle G,F \rangle \rangle$. $E(\lambda x.M',B,\forall t.\tau_1 \rightarrow \tau_2)$ is

$\langle S, \langle G, F + [t \Rightarrow \Im], E \rangle \rangle$ ($E(\lambda x.M', B, \tau_1 \rightarrow \tau_2)$ is

$\langle S, \langle G,F,E \rangle \rangle$) where $\Im$ is $\{SB(y) \mid y$ is a free λ-variable of $M\}$.

(iii) If $M \equiv PQ$, then

let $E(P, B, (\forall u.\beta) \rightarrow \forall u'.\beta')$ be $\langle S, \langle G,F \rangle \rangle$ where u, u', β and β' are fresh variables and let $E(Q, SB, S(\forall u.\beta))$ be $\langle S', \langle G', F' \rangle \rangle$.

If $\sigma \equiv \forall t.\tau$ ($\sigma \equiv \tau \in \Sigma^-$) then $E(M,B,\forall t.\tau)$ is
$\langle S'S, \langle G'', (S'F \cup F') + [t \Rightarrow \Im] \rangle \rangle$
($E(M,B,\tau)$ is $\langle S'S, \langle G'', (S'F \cup F') \rangle \rangle$) where $\Im$ is $\{S'SB(y) \mid y$ is a free variable of $M\}$ and G'' is the sequence $S'G, G', (S'S(\forall u'.\beta' \leq \tau))$. ◊

The following theorem asserts that $E(M,B,\sigma)$ returns the "minimum" set of constraints that must be satisfied for deriving an instance of the type scheme σ from an instance of B for M.

Theorem 10. Let M be a λ-term, B be a scheme assignment defined on all the free λ-variables of M and σ be a type scheme. Let $E(M,B,\sigma)$ be $\langle S, \langle G, F \rangle \rangle$. There is a simple substitution s such that $sB \vdash M:s\sigma$ if and only if for some extension s'' of s and some simple substitution s'
$s'' =_V s's$, where $V = FV(B) \cup FV(\sigma) \cup FV(G) \cup FV(F)$ and
s' satisfies $\langle G, F \rangle$.
Moreover $s''(G) = C(D)$ for some derivation D of $sB \vdash M:s\sigma$.
Proof. The proof is by induction on M, using the fact that derivations are syntax directed (Property 4). ◊

Given a term M, if B represents the class of all type assignments whose domain is the set of free variables of M, and σ represents the class of all types, then $E(M,B,\sigma)$ gives the most general characterization of a typing for M.

Definition 11. Let M be a λ-term. Let B be $\Sigma_{1 \leq i \leq n}[x_i \Rightarrow \forall t_i.\alpha_i]$, where $\{x_i \mid 1 \leq i \leq n\} = FV(M)$ and $\langle S, \langle G, F \rangle \rangle$ be $E(M,B,\forall t.\beta)$ (where t, β, the t_i's and the α_i's are fresh variables).
$$\Pi(M) = \langle SB, S(\forall t.\beta), \langle G, F \rangle \rangle \quad \Diamond$$

Main Theorem. Let $M \in \Lambda$ and $\Pi(M)$ be $\langle B, \sigma, \langle G, F \rangle \rangle$. There is a type assignment A and a type $\varphi \in T$ such that $A \vdash M:\varphi$ if and only if for some simple substitution s, $A = sB$ (on the free variables of M), $\varphi = s\sigma$ and s satisfies the scheme system $\langle G, F \rangle$. Moreover $s(G) = C(D)$ for some derivation D of $A \vdash M:\varphi$.
Proof. Directly from theorem 10. ◊

4. A Strongly Normalizing Lambda-Term Having No Polymorphic Type

As proved in [Girard 72] any term of the polymorphic λ-calculus is strongly normalizable. From the isomorphism between second order polymorphic λ-calculus and type inference in the polymorphic type discipline, see [Leivant 83], we derive that any λ-term M which can be typed in this system is strongly normalizable. Here we show that the converse is not true by proving that there is a strongly normalizing (closed) λ-term R such that, for no type φ, $\vdash R:\varphi$. A proof of this fact was already given in [Giannini et al 87]. Using the existence of the principal type scheme system, here, we get an easier and clearer proof.

We first need some notation and definition.

Notation. (i) Let $\psi \in T$. Then
$\psi \equiv \forall \bar{a}_1.(\forall \bar{b}_1.\omega_1) \rightarrow \cdots \rightarrow \forall \bar{a}_n.(\forall \bar{b}_n.\omega_n) \rightarrow \forall \bar{a}_{n+1}.\omega_{n+1}$
for some $\bar{a}_1,\ldots,\bar{a}_{n+1},\bar{b}_1,\ldots,\bar{b}_n,\omega_1,\ldots,\omega_{n+1}$, where $n \geq 0$, ω_{n+1} is a type variable and $\omega_i \in T^-$ ($1 \leq i \leq n$). Let $\psi(i)$ denote ω_i and $\psi[i]$ denote $\forall \bar{b}_i.\omega_i$. The *rightlength* of ψ is $n+1$.
(ii) Let $\psi \in T$. Then ψ can be written as
$\forall \bar{c}_1.(\forall \bar{c}_2.\cdots(\forall \bar{c}_{m+1}.\chi_{m+1}) \rightarrow \forall \bar{d}_m.\chi_m) \rightarrow \cdots \rightarrow \forall \bar{d}_1.\chi_1$
for some $\bar{c}_1,\ldots,\bar{c}_{m+1},\bar{d}_1,\ldots,\bar{d}_m,\chi_1,\ldots,\chi_{m+1}$, where $m \geq 0$, χ_{m+1} is a type variable and $\chi_i \in T^-$ ($1 \leq i \leq m$). χ_{m+1} is the *leftmost variable of* ψ and $m+1$ is the *leftlength* of ψ. ◊

Definition 12. (i) The subset of T, T_a, is the least set such that:
$a \in T_a$,
if $\psi \in T_a$ and $\varphi \in T$ then $\psi \rightarrow \varphi \in T_a$,
if $\psi \in T_a$ and $a \notin \bar{b}$ then $\forall \bar{b}.\psi \in T_a$.
(ii) $\psi \in Q_j^{i,k}$ ($i,j,k \geq 1$) if and only if $\psi(i) \in T_c$, where c is the j-th variable of the string $\bar{a}_k$ and $k \leq i$. ◊

Note that if $\psi \in Q_j^{i,k}$ then for any substitution $[\bar{\chi} / \bar{c}]$ of the free variables of ψ, $[\bar{\chi} / \bar{c}]\psi \in Q_j^{i,k}$.

Let I be $\lambda x.x$, K be $\lambda xy.x$ and Δ be $\lambda x.xx$. Let R be $(\lambda xy.y(xI)(xK))\Delta$

Theorem 13. There is no type φ such that $\vdash R:\varphi$.
Proof. $\Pi(R)$ is $\langle \emptyset, \forall t.\beta, \langle G, F \rangle \rangle$, where G is the sequence of inequalities:
1) $\forall v_1.\beta_1 \leq (\forall t_1.\delta_1) \rightarrow \forall t_2.\delta_2$,
2) $\forall u_1.(\forall u_3.\alpha_3) \rightarrow \forall u_4.\alpha_4 \leq (\forall t_3.(\forall t_5.\delta_5) \rightarrow \forall t_6.\delta_6) \rightarrow \forall t_4.\delta_4$
3) $\forall t_5.\delta_5 \leq \delta_6$,
4) $\forall t_4.\delta_4 \leq \delta_1$,
5) $\forall t_2.\delta_2 \leq (\forall v_3.\beta_3) \rightarrow \forall v_4.\beta_4$,
6) $\forall u_1.(\forall u_3.\alpha_3) \rightarrow \forall u_4.\alpha_4 \leq \sigma$,
7) $\forall r_5.\gamma_2 \leq \gamma_4$,

8) $\forall r_4.\gamma_1 \leq \beta_3$,

9) $\forall v_4.\beta_4 \leq \beta_2$,

10) $\forall u_3.\alpha_3 \leq (\forall u_5.\alpha_5) \rightarrow \forall u_6.\alpha_6$,

11) $\forall u_3.\alpha_3 \leq \alpha_5$,

12) $\forall u_6.\alpha_6 \leq \alpha_4$,

13) $\forall u_2.(\forall v_1.\beta_1) \rightarrow \forall v_2.\beta_2 \leq \beta$

where σ is $(\forall r_3.(\forall r_5.\gamma_2) \rightarrow \forall r_6.(\forall r_1.\gamma_3) \rightarrow \forall r_2.\gamma_4) \rightarrow \forall r_4.\gamma_1$ and F is

$[u_5 \Rightarrow \{\forall u_3.\alpha_3\}, u_4 \Rightarrow \{\forall u_3.\alpha_3\}, u_2 \Rightarrow \{\forall u_1.\alpha_1\},$

$r_6 \Rightarrow \{\forall r_5.\gamma_2\}, t_6 \Rightarrow \{\forall t_5.\delta_5\}, t_1 \Rightarrow \{\forall u_1.(\forall u_3.\alpha_3) \rightarrow \forall u_4.\alpha_4\},$

$v_2 \Rightarrow \{\forall u_1.(\forall u_3.\alpha_3) \rightarrow \forall u_4.\alpha_4, \forall v_1.\beta_1\}, r_2 \Rightarrow \{\forall r_5.\gamma_2\}]$.

It is easy to verify that every inequality corresponds to an occurrence of either a λ-variable or an application subterm of

$$(\lambda xy.y(x(\lambda w.w))(x(\lambda x_1 y_1.x_1)))(\lambda z.zz) =_\alpha R$$

in the following way:

1 corresponds to the (only) occurrence of y,
2 " " to the first occurrence of x,
3 " " to the occurrence of w in $\lambda w.w$,
4 " " to the application $x(\lambda w.w)$,
5 " " to the application $y(x(\lambda w.w))$,
6 " " to the second occurrence of x,
7 " " to the occurrence of x_1 in $(\lambda x_1 y_1.x_1)$,
8 " " to the application $x(\lambda x_1 y_1.x_1)$,
9 " " to the application $y(x(\lambda w.w))(x(\lambda x_1 y_1.x_1))$,
10 " " to the first occurrence of z in $\lambda z.zz$,
11 " " to the second occurrence of z in $\lambda z.zz$,
12 " " to the application zz,
13 " " to the whole term (which is an application).

By the main theorem, there exists a type such that $\vdash R:\varphi$ if and only if for some substitution s, s satisfies $\langle G,F \rangle$. Assume that such an s exists, we will prove that this implies a contraddiction. More precisely the contraddiction arises from inequalities 2 and 3 (the constraints for typing xI), inequalities 6 and 7 (the constraints for typing xK) and the inequalities 10 and 11 (the constraints for typing Δ).

Let $s(\forall u_3.\alpha_3)$ be $\forall \bar{a}.\psi$, for some $\psi \in T^-$.

(A) Inequalities 10 and 11 imply that $\forall \bar{a}.\psi \in Q_j^{1,1}$ for some j. In fact

10') $\forall \bar{a}.\psi \leq s(\forall u_5.\alpha_5 \rightarrow \forall u_6.\alpha_6)$ and

11') $\forall \bar{a}.\psi \leq s(\alpha_5)$.

If $\psi \notin T_a$ for some $a \in \bar{a}$ then

$leftlength(\forall \bar{a}.\psi) = leftlength(s(\forall u_5.\alpha_5 \rightarrow \forall u_6.\alpha_6))$ (by 10')

$= leftlength(s(\alpha_5)) + 1$ (by def. of $leftlength$)

$= leftlength(\forall \bar{a}.\psi) + 1$ (by 11').

This is absurd. Therefore $\psi \in T_a$ for some $a \in \bar{a}$, i.e., $\forall \bar{a}.\psi \in Q_j^{1,1}$ for some j.

(B) Inequalities 2 and 3 and the fact that $F(t_6) = \{\forall t_5.\delta_5\}$ imply that $\forall \bar{a}.\psi \in Q_j^{2,1}$.

First from inequality 2 and $\forall \bar{a}.\psi \in Q_j^{1,1}$ we derive that $rightlength(\forall \bar{a}.\psi)$ is at least 2. (In fact if $a \equiv \psi$, $s(\forall u_5.\alpha_5 \rightarrow \forall u_6.\alpha_6) \equiv \forall \bar{a}.a$, that is clearly absurd.) Then, again from inequality 2, $s(t_3) = \bar{a}$ and $\forall \bar{a}.\psi \in Q_j^{1,1}$ we get $\forall \bar{a}.s(\forall t_5.\delta_5) \rightarrow s(\forall t_6.\delta_6) \in Q_j^{1,1}$. Therefore $s(\forall t_5.\delta_5) \in T_a$ for some $a \in \bar{a}$. By 3, $s(\delta_6) \in T_a$ (a is free in δ_6). Since a is free in $s(\forall t_5.\delta_5)$ the clause $F(t_6) = \{\forall t_5.\delta_5\}$ insures that a is free in $s(\forall t_6.\delta_6)$. Therefore

$$\forall a.s(\forall t_5.\delta_5) \rightarrow s(\forall t_6.\delta_6) \in Q_j^{2,1}.$$

Moreover that fact that no substitution for the variables in $s(u_1)$ can introduce an occurrence of a bound variable in $\forall \bar{a}.\psi$ and again inequality 2 imply that $\forall \bar{a}.\psi \in Q_j^{2,1}$.

(C) Inequality 6, $\forall \bar{a}.\psi \in Q_j^{1,1}$ and $\forall \bar{a}.\psi \in Q_j^{2,1}$ imply that $rightlength(\forall \bar{a}.\psi)$ is at least 3.

(Similar to the proof of (B).) Therefore let

$$\psi \equiv (\forall \bar{b}.\psi_1) \rightarrow \forall \bar{c}.\psi_2 \rightarrow \psi_3$$

where $\psi_1 \in T^-$ and let

$p = \max_i$(for all $k \leq i$, $\forall \bar{a}.\psi_1 \in Q_j^{k,1}$) and

$q = \max_i$(for all $k \leq i$, $\forall \bar{a}.\forall \bar{c}.\ \psi_2 \rightarrow \psi_3 \in Q_j^{k,1}$).

(D) Inequalities 2, 3 and $F(t_6) = \{\forall t_5.\delta_5\}$ imply that $p \leq q$.

For all i such that $\forall \bar{a}.\psi_1 \in Q_j^{i,1}$, by 2, $s(\forall t_5.\delta_5)(i) \in T_a$ where a is the j-th variable of $\bar{a}$. Hence, by 3, $s(\delta_6)(i) \in T_a$. Since a is free in $s(\forall t_5.\delta_5)$, the clause $F(t_6) = \{\forall t_5.\delta_5\}$ implies $s(\forall t_6.\delta_6)(i) \in T_a$. Therefore, the fact that no substitution for the variables in $s(u_1)$ can introduce an occurrence of a bound variable in $\forall \bar{a}.\forall \bar{c}.\psi_2 \rightarrow \psi_3$ and again inequality 2 imply that $\forall c.\ \psi_2 \rightarrow \psi_3\ (i) \in T_a$ and then $\forall \bar{a}.\forall \bar{c}.\psi_2 \rightarrow \psi_3 \in Q_j^{i,1}$.

(E) Inequalities 6, 7, $F(r_2) = \{\forall r_5.\gamma_2\}$ and $F(r_6) = \{\forall r_5.\gamma_2\}$ imply that $p < q$. (Similar to the proof of (D).)

(F) By $p < q$ and inequalities 2 and 3 we also derive that $rightlength(\forall \bar{b}.\psi_1) > p$ and $\forall \bar{b}.\psi_1 \in Q_r^{p+1,1}$ for some r.

(G) Finally we show that (F) and inequalities 10 and 11

together yield a contraddiction. This concludes the proof that for no substitution s , s satisfies $\langle G,F\rangle$.

Since $\forall \bar{a}.\psi \equiv s(\forall u_3.\alpha_3)$ is an arrow type and, by 11, $\forall u_3.\alpha_3 \leq \alpha_5$, $s(\alpha_5)$ is equal to $\varphi_1 \rightarrow \varphi_2$ for some $\varphi_1, \varphi_2 \in T$.

Inequalities 10 and 11 then imply that for some $\bar{\chi}$ and $\bar{\omega}$:

10") $[\bar{\chi} / \bar{a}]\forall \bar{b}.\psi_1 \equiv s(\forall u_5.\alpha_5) \equiv \forall s(u_5).\varphi_1 \rightarrow \varphi_2$

11") $[\bar{\omega} / \bar{a}](\forall \bar{b}.\psi_1) \rightarrow \forall \bar{c}.\psi_2 \rightarrow \psi_3 \equiv s(\alpha_5) \equiv \varphi_1 \rightarrow \varphi_2$.

Hence $s(u_5)=\bar{b}$ and by (F) $\forall \bar{b}.\varphi_1 \rightarrow \varphi_2 \in Q_r^{p+1,1}$. Let χ and ω be the type scheme replacing a in $[\bar{\chi} / \bar{a}]$ and $[\bar{\omega} / \bar{a}]$ respectively. Let us consider the possible cases:

(a) $\chi \in T_d$ for some d free in χ,

(b) the leftmost variable of χ is bound,

(i) $\omega \in T_{d'}$ for some d' free in ω,

(ii) the leftmost variable of ω is bound.

By 10" and 11", (a) and (ii) and (b) and (i) are not possible. Let us consider the other combinations.

(a) and (i). From 10" and 11" $d \equiv d'$. Hence, by 11" and $p<q$, $(\varphi_1 \rightarrow \varphi_2)(p+1) \in T_d$, which is in contraddiction with the fact that $\forall \bar{b}.\varphi_1 \rightarrow \varphi_2 \in Q_r^{p+1,1}$.

(b) and (ii). 11" and $p<q$ imply that the leftmost variable of $\varphi_1 \rightarrow \varphi_2(p+1)$ is bound against the fact that $\forall \bar{b}.\varphi_1 \rightarrow \varphi_2 \in Q_r^{p+1,1}$ and hence $\varphi_1 \rightarrow \varphi_2(p+1) \in T_c$ where c is the r-th variable of $\bar{b}$. ◊

5. Some Properties of Type Scheme Systems

Canonical systems and normal forms

It is not known if the problem of finding a substitution satisfying a given type scheme system, if any, is decidable. There are, however, subclasses of systems for which we can show the existence of a solution. In this section we introduce one of such classes, the canonical systems, and show that we can always find a substitution satisfying a canonical system. Moreover we can prove that the algorithm Π applied to a λ-term in normal form gives a canonical system.

Definition 14. Let $\langle G,F\rangle$ be a type scheme system. $\langle G,F\rangle$ is *canonical* if and only if

(i) $G=\{\forall t_i.\beta_i \leq \sigma_i \mid 1 \leq i \leq n\}$ for some n and

(ii) $leftside(G) \cap domain(F) = \emptyset$. ◊

Property 15. If $\langle G,F\rangle$ is a canonical type scheme system then there exists always a simple substitution s such that s satisfies $\langle G,F\rangle$.

Proof. Consider a one-to-one mapping θ between the union of the sets of sequence variables and scheme variables and the set of type variables. ($\theta(t)$ is the type variable corresponding to the sequence variable t and $\theta(\alpha)$ is the type variable coresponding to the scheme variable α.) Consider the simple substitution s such that:

if $x \in rightside(G)-(leftside(G) \cup domain(F))$ then $s(x)=\theta(x)$,

if $\forall t.\beta \in leftside(G)$ then $s(t)=\theta(t)$ and $s(\beta)=\theta(t)$ and

if $t \in domain(F)$ then $s(t)=\varepsilon$ (the empty string).

It is easy to verify that s satisfies $\langle G,F\rangle$. ◊

Theorem 16. Let M be a λ-term in normal form and let $\Pi(M)$ be $\langle B,\sigma,\langle G,F\rangle\rangle$. $\langle G,F\rangle$ is a canonical type scheme system.

Proof. Define a scheme assignment B *canonical* if

$$B=\Sigma_{1 \leq i \leq n}\,[x_i \Rightarrow \forall t_i.\alpha_i]$$

for some n where for all j and i, $1 \leq j \neq i \leq n$, $t_i \neq t_j$ and $\alpha_i \neq \alpha_j$. It is easy to prove by induction on the structure of (the normal form) M that: if B is canonical and $t,\beta \notin FV(B)$ then $E(M,B,\forall t.\beta)=\langle S,\langle G,F\rangle\rangle$ where $\langle G,F\rangle$ is canonical. Then the proof follows from the definition of $\Pi(M)$. ◊

Connections with Curry's Typing

Since Curry's types [Curry 69] can be viewed as a subset of polymorphic types, the problem of deciding if a given term M has a type in Curry's type assignment or not can be reduced to the problem of searching a solution for the principal type scheme system of M with particular properties.

Property 17. Let $\Pi(M)$ be $\langle B,\sigma,\langle G,F\rangle\rangle$ and let V be $FV(B) \cup FV(\sigma) \cup FV(G) \cup FV(F)$. Let $\Re$ be the set of simple substitutions s such that: if $s \in \Re$ then

$s(t)$ is the empty string of type variables for all $t \in V$ and

$s(\alpha)$ is a Curry's type for all $\alpha \in V$.

It is decidable if G has a solution belonging to $\Re$ and it is possible to find the most general solution for G (within the solutions in $\Re$). If s is the most general solution then $s(\sigma)$ is the principal type for M (in Curry's type assignment). The method of finding s is an easy extension of Curry's technique for solving sets of equations between types, see [Curry 69] and [BenYelles 79]. ◊

6. Polymorphic type assignment systems of order n≥1

Girard, in [Girard 72], introduces a class of typed λ-calculi, which he calls F_n ($n \geq 0$), such that a term of F_n represents a proof in the intuitionistic propositional calculus of n+2-th order. The system F_0 is the second order polymorphic λ-calculus. Let

$\vdash_n$ be the type assignment system for λ-calculus corresponding to F_n. In the following we will sketch the proof that we can extend the results of this paper to $\vdash_n$ for any n.

First let us recall some definitions. A *kind* is defined by the following grammar.

$$\theta ::= 0 \mid \theta \to \theta$$

Let θ be a kind, a^θ, b^θ, c^θ and d^θ range over *variables of kind* θ and φ^θ, χ^θ, ψ^θ, and ω^θ range over *connectives of kind* θ. The set of connective of kind $\theta \equiv \theta_1 \to \cdots \to \theta_m \to 0$ (it is immediate to verify that any kind can be written in this way) is defined by the following grammar.

$$\varphi^\theta ::= a^\theta \mid \lambda a_1{}^{\theta_1} \cdots a_m{}^{\theta_m}.\varphi^0$$

The connectives of kind 0, the *types*, are defined by:

$$\varphi^0 ::= a^0 \mid \varphi^0 \to \varphi^0 \mid \forall a^\theta.\varphi^0 \mid \varphi^{\theta'}(\varphi_1{}^{\theta_1}, \ldots, \varphi_m{}^{\theta_m})$$

where $\theta' \equiv \theta_1 \to \cdots \to \theta_m \to 0$. In the following we omit the superscript denoting the kind when this is 0. To any kind we can associate an integer, its *order*, in the following way:

$order(0)=0$,

$order(\theta_1 \to \cdots \to \theta_m \to 0) = \max_{1 \le i \le m}(order(\theta_i))+1$.

(The order of a kind is the deepest nesting level of the arrows in it.) The *order* of a connective φ^θ is the maximum of the orders of the kinds of its subterms. Hence a connective may have kind 0 and order n where $n>0$.

Type assignment system of order n

$$\text{(var)}\quad A+[x \Rightarrow \varphi] \vdash_n x:\varphi$$

$$(\to\text{I})\ \frac{A+[x \Rightarrow \varphi] \vdash_n M:\psi}{A \vdash_n \lambda x.M:\varphi \to \psi}$$

$$(\to\text{E})\ \frac{A \vdash_n M:\varphi \to \psi \qquad A \vdash_n N:\varphi}{A \vdash_n MN:\psi}$$

$$(\forall\text{I})\ \frac{A \vdash_n M:\varphi}{A \vdash_n M:\forall a^\theta.\varphi}\ (*) \qquad (\forall\text{E})\ \frac{A \vdash_n M:\forall a^\theta.\varphi}{A \vdash_n M:[\psi^\theta / a^\theta]\varphi}$$

(*) a^θ is bindable in M with respect to A.

All the connectives both in A and in the derivation must be of order less or equal than n.

The containement relation of order n is defined as follows.

Definition 18. $\forall a_1{}^{\theta_1} \cdots a_m{}^{\theta_m}.\varphi \le_n [\chi_i^\vartheta i / a_i^{\theta_i} i]\varphi \quad (1 \le i \le m)$ where both the orders of θ_i $(1 \le i \le m)$ and the order of φ are less or equal than n. ◊

All the results of this papers hold in $\vdash_n$,using $\le_n$ instead of $\le$. However it is not known whether the relation $\le_n$ is decidable. The decidibility of $\le_n$ is related to the problem of higher order pattern matching which is not known to be decidable.

It is natural to ask if the term $R = (\lambda xy.y(xI)(xK))\Delta$, which was proved not have type in the polymorphic type assignment system, can have type in $\vdash_n$ for some $n \ge 1$.

Theorem 19. For every n, $n \ge 1$, there is a type φ' such that $\vdash_n R:\varphi'$.

Proof. Let $\varphi \equiv (c \to c) \to \psi \to b$, where $\psi \equiv (c \to c \to c) \to c \to c \to c$, $\omega \equiv \forall b.b \to a^{0 \to 0}(b)$ and $\chi \equiv \forall a^{0 \to 0}.\omega \to a^{0 \to 0}(c \to a^{0 \to 0}(c))$. Let A be $[x \Rightarrow \chi, y \Rightarrow \varphi]$. Let D_1 be the following derivation:

$$\dfrac{\dfrac{A \vdash x:\chi}{(\forall\text{E})} \qquad \dfrac{\dfrac{A+[z \Rightarrow b] \vdash z:b}{(\to\text{I})}}{A \vdash \lambda z.z:b \to b}}{A \vdash x:(\forall b.b \to b) \to c \to c \quad (1)} \qquad \dfrac{}{(\forall\text{I})}\ A \vdash \lambda z.z:\forall b.b \to b$$

$$(\to\text{E})\ \frac{}{A \vdash xI:c \to c}$$

(1) $a^{0 \to 0}$ is replaced by $\lambda d.d$ in χ.

Let D_2 be the following derivation:

$$\dfrac{\dfrac{A \vdash x:\chi}{(\forall\text{E})} \qquad \dfrac{\dfrac{\dfrac{A+[z \Rightarrow b, w \Rightarrow b] \vdash z:b}{(\to\text{I})}}{A+[z \Rightarrow b] \vdash \lambda w.z:b \to b}}{(\to\text{I})}}{A \vdash x:(\forall b.b \to b \to b) \to \psi \quad (2)} \qquad \dfrac{}{(\forall\text{I})}\ A \vdash \lambda zw.z:b \to b \to b$$

$$(\to\text{E})\ \frac{}{A \vdash xK:\psi} \qquad \dfrac{}{A \vdash K:\forall b.b \to b \to b}$$

(2) $a^{0 \to 0}$ is replaced by $\lambda d.d \to d$ in χ.

Let $\varphi_1 \equiv (c \to a^{0 \to 0}(c)) \to a^{0 \to 0}(c \to a^{0 \to 0}(c))$ and D_3 be the following derivation:

$$\dfrac{\dfrac{\dfrac{[x\Rightarrow\omega]\ \vdash x:\omega}{[x\Rightarrow\omega]\ \vdash x:\varphi_1\ (3)}(\forall E) \qquad \dfrac{[x\Rightarrow\omega]\ \vdash x:\omega}{[x\Rightarrow\omega]\ \vdash x:c\to a^{0\to 0}(c)\ (4)}(\forall E)}{[x\Rightarrow\omega]\ \vdash xx:a^{0\to 0}(c\to a^{0\to 0}(c))}(\to E)}{\vdash \lambda x.xx:\omega\to a^{0\to 0}(c\to a^{0\to 0}(c))}(\to I)$$

$$\dfrac{\vdash \lambda x.xx:\omega\to a^{0\to 0}(c\to a^{0\to 0}(c))}{\vdash \lambda x.xx:\forall a^{0\to 0}.\omega\to a^{0\to 0}(c\to a^{0\to 0}(c))\equiv\chi}(\forall I)$$

(3) b is replaced by $c\to a^{0\to 0}(c)$ in ω,
(4) b is replaced by c in ω.
We can finally build the following derivation D for R.

$$\dfrac{\dfrac{\dfrac{\dfrac{A\vdash y:\varphi \qquad D_1:\ A\vdash xI:c\to c}{A\vdash y(xI):\psi\to b}(\to E) \qquad D_2:\ \vdash xK:\psi}{A\ \vdash y(xI)(xK):b}(\to E)}{[x\Rightarrow\chi]\ \vdash \lambda y.y(xI)(xK):\varphi\to b}(\to I)}{\vdash \lambda xy.y(xI)(xK):\ \chi\to\varphi\to b}(\to I) \qquad D_3:\ \vdash\lambda x.xx:\chi$$

$$\dfrac{\vdash \lambda xy.y(xI)(xK):\ \chi\to\varphi\to b \qquad D_3:\ \vdash\lambda x.xx:\chi}{\vdash (\lambda xy.y(xI)(xK))(\lambda x.xx):\varphi\to b}(\to E)$$

Analizing the previous derivation we can see that we have never used connectives of order greater than 1. Hence the previous is a derivation in a type assignment of any order n such that $n\geq 1$. ◊

Acknoledgments

We are indebted with Furio Honsell, who first conjectured that the strongly normalizing term R, defined in section 4, could not be typed in the polymorphic type discipline. Moreover we would like to thank John Mitchell for the helpful discussions we had in occasion of his visit in Torino in January 1987.

References

[Barendregt 84] Barendregt, H.P., *The Lambda Calculus: its Syntax and Semantics*. North Holland, 1984 (revised version).

[BenYelles 79] Ben-Yelles, C., *Type Assignment in the Lambda-Calculus: Syntax and Semantics*. Ph. D. Thesis, University College of Swansea, 1979.

[Coppo 85] Coppo, M., *Tipi e polimorfismo nei linguaggi di programmazione,* Atti degli Incontri di Logica Matematica, Siena, 1985.

[Curry 69] Curry, H.B., Modified Basic Functionality in Combinatory Logic. *Dialectica*, 1969.

[Damas and Milner 82] Damas, L. and Milner, R., The Principal Type Schemes for Functional Programs. In *Symposium on Principles of Programming Languages*, ACM, 1982.

[Giannini et al 87] Giannini,P.,Honsell , F., Ronchi Della Rocca, S., *A strongly normalizing term having no type in the system F (second order λ-calculus)*, Rapporto Interno, Dipartimento di Informatica, Torino, 1987.

[Girard 72] Girard, J.Y., *Interpretation Fonctionelle et Elimination des Coupures de l'Arithmetique d'Ordre Superieur.* These D'Etat, Universite Paris VII, 1972.

[Hindley 69] Hindley, R.,The Principal Type Scheme of an Object in Combinatory Logic. *Transactions of American Mathematical Society*, 1969.

[Leivant 83] Leivant, D., Polymorphic Type Inference. In *Symposium on Principles of Programming Languages*, ACM, 1983.

[Mitchell 87] Mitchell, J.C., Polymorphic Type Inference and Containment. *Information and Control*, to appear.

[Prawitz 65] Prawitz, D., *Natural Deduction, a Proof Theoretic Study*. Almquist and Wiksell, Amsterdam, 1965.

[Reynolds 74] Reynolds, J.C., Towards a Theory of Type Structures. In *Paris Colloquium on Programming*, Springer Verlag, 1974.

[Robinson 65] Robinson, J.A., A Machine Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 1965.