PROCEEDINGS OF THE YEREVAN STATE UNIVERSITY

Physical and Mathematical Sciences 2010, № 1, p. 37–46

Informatics

ON THE TYPE CORRECTNESS OF POLYMORPHIC λ-TERMS. 2

A. H. ARAKELYAN*

Chair of Programming and Information Thechnologies, YSU

In this paper the polymorphic lambda terms are considered, where no type information is provided for the variables. The aim of this work is to prove that presented typification algorithm [1] typifies such terms in most common way.

***Keywords*:** type, term, constraint, skeleton, expansion, principal typing.

1. Introduction. Types are used in programming languages to analyze programs without executing them, for purposes such as detecting programming errors earlier, for doing optimizations etc. In some programming languages no explicit type information is provided by the programmer, hence some system of type inference is required to recover the lost information and do compile time type checking. One of such type inference systems is the well known Hindley/Milner system [2], used in languages such as Haskell, SML, OCaml etc. An important property of the type systems is the property of *principal typings* [3, 4], which allows the compiler to do *compositional analysis*, i.e. analysis of modules in absence of information about other modules [3, 4]. Unfortunately the Hindley/Milner system doesn't support the property of *principal typings* [3]. This paper is the continuation of [1], in which we consider the extension of the type inference system called *System E*. In section 2 we prove that the type inference algorithm returns the principal typing of a term.

2. Principal Typing of a Term.

2.1. Preliminary Definitions and Facts. Before proving that the type inference algorithm returns the principal typing of a term let us present some definitions and facts.

Definition 2.1. Let $Q_1, Q_2 \in Skeleton$. Q_1 and Q_2 are equivalent, written $Q_1 \approx Q_2$, iff $term(Q_1)$=$term(Q_2)$, $typing(Q_1)$=$typing(Q_2)$, $constraint(Q_1)$= =$constraint(Q_1)$. In other words, $Q_1 \approx Q_2$, iff the judgements $(M \triangleright Q_1):(A \vdash \tau)/\Delta$ and $(M \triangleright Q_2):(A \vdash \tau)/\Delta$ are both inferable or not inferable.

Lemma 2.1. The following skeletons are equivalent:

1. $(Q_1 \cap (Q_2 \cap Q_3)) \approx ((Q_1 \cap Q_2) \cap Q_3)$; 2. $(Q_1 \cap Q_2) \approx (Q_2 \cap Q_1)$;

3. $(\omega^M \cap Q) \approx Q$; 4. $e(Q_1 \cap Q_2) \approx (eQ_1 \cap eQ_2)$; 5. $e\omega^M \approx \omega^M$,

where $Q_1, Q_2, Q_3 \in Skeleton$ and $M \in Term$ and $e \in ExpansionVariable$.

* E-mail: ara_arakelyan@yahoo.com

Let us consider the judgement $(M \rhd Q):(A \vdash \tau)/\Delta$ that is inferable. In many cases we will consider the maximal subtrees of the inference tree of that judgement that have root node corresponding to one of the following type inference rules: [VAR], [CONST], [OMEGA], [ABS] and [APP].

Lemma 2.2. Let $(M \rhd Q):(A \vdash \tau)/\Delta$ be an inferable judgement. Then there exist E-paths $e'_1,\ldots,e'_n$, environments $A_1,\ldots,A_n$, $Q_1,\ldots,Q_n \in Skeleton$, $\tau_1,\ldots,\tau_n \in Type$ and $\Delta_1,\ldots,\Delta_n \in Constraint$, $n \geq 1$, such that $Q \approx e'_1Q_1 \cap \ldots \cap e'_nQ_n$, $A{=}e'_1A_1 \cap \ldots \cap e'_nA_n$, $\tau{=}e'_1\tau_1 \cap \ldots \cap e'_n\tau_n$, $\Delta{=}e'_1\Delta_1 \cap \ldots \cap e'_n\Delta_n$ and judgements $(M \rhd Q_i):(A_i \vdash \tau_i)/\Delta_i$, $i{=}1,\ldots,n$, are inferable, and in the last step of inference of that judgements one of the following rules is used: [VAR], [CONST], [OMEGA], [ABS] and [APP].

A free occurrence of the subskeleton $x^{:\tau}$ in skeleton Q is defined in a conventional way, i.e. the occurrence of the subskeleton $x^{:\tau}$ in skeleton Q is called free, if it doesn't fall within the scope of a lambda that uses variable x, otherwise, the occurrence is called bounded. It is easy to see, that if the skeleton Q' is obtained from the skeleton Q by renaming some term variables, then $Q \approx Q'$. Let us introduce the following notations:

1. We denote by $Q\langle Q_1,\ldots,Q_n\rangle$, $n \geq 0$, the skeleton Q, in which mutually different subskeletons $Q_1,\ldots,Q_n$ are considered.

2. We denote by $Q\langle Q_1 := Q'_1,\ldots,Q_n := Q'_n\rangle$, $n \geq 0$, those skeletons that are obtained from the skeleton $Q\langle Q_1,\ldots,Q_n\rangle$ through substituting the subskeletons $Q_1,\ldots,Q_n$ by $Q'_1,\ldots,Q'_n$ respectively. The substitution mentioned above is called canonical, iff all free occurrences of subskeletons in Q_i are also free in $Q\langle Q_1,\ldots,Q_n\rangle$, and all free occurrences of subskeletons in Q'_i are also free in $Q\langle Q_1 := Q'_1,\ldots,Q_n := Q'_n\rangle$, $i{=}1,\ldots,n$. Henceforth only canonical substitutions of skeletons will be considered.

Definition 2.2. Let $Q\langle Q'\rangle \in Skeleton$. Then the *E-Path* of the skeleton Q' in Q, written as $E\text{-}Path(Q\langle Q'\rangle)$, is calculated as follows:

1. If $Q{=}Q'$, then $E\text{-}Path(Q\langle Q'\rangle) = \varepsilon$;

2. If $Q{=}eQ_1$, and Q' is a subskeleton of Q_1, then $E\text{-}Path(Q\langle Q'\rangle) =$ $=eE\text{-}Path(Q_1\langle Q'\rangle)$, where $Q_1 \in Skeleton$ and $e \in ExpansionVariable$;

3. If $Q{=}(\lambda x.Q_1)$, and Q' is a subskeleton of Q_1, then $E\text{-}Path(Q\langle Q'\rangle) =$ $=E\text{-}Path(Q_1\langle Q'\rangle)$, where $Q_1 \in Skeleton$ and $x \in TermVariable$;

4. If $Q{=}(Q_1 \cap Q_2)$, and Q' is a subskeleton of Q_1, then $E\text{-}Path(Q\langle Q'\rangle) =$ $=E\text{-}Path(Q_1\langle Q'\rangle)$, where $Q_1, Q_2 \in Skeleton$;

5. If $Q{=}(Q_1 \cap Q_2)$, and Q' is a subskeleton of Q_2, then $E\text{-}Path(Q\langle Q'\rangle) =$

$=$*E-Path*$\left(Q_2\langle Q'\rangle\right)$, where $Q_1, Q_2 \in Skeleton$;

6. If Q=$(Q_1Q_2)^{:\tau}$, and Q' is a subskeleton of Q_1, then *E-Path*$\left(Q\langle Q'\rangle\right)=$ $=$*E-Path*$\left(Q_1\langle Q'\rangle\right)$, where $Q_1, Q_2 \in Skeleton$ and $\tau \in Type$;

7. If Q=$(Q_1Q_2)^{:\tau}$, and Q' is a subskeleton of Q_2, then *E-Path*$\left(Q\langle Q'\rangle\right)=$ $=$*E-Path*$\left(Q_2\langle Q'\rangle\right)$, where $Q_1, Q_2 \in Skeleton$ and $\tau \in Type$.

Let us present some simple propositions without proof.

Proposition 2.1. Let $Q \in Skeleton$ and $type(Q) = \vec{e}_1\tau_1 \cap \ldots \cap \vec{e}_n\tau_n$, $n \geq 1$, where $\tau_1,\ldots,\tau_n \in Type$ and $\vec{e}_1,\ldots,\vec{e}_n$ are *E-Paths*. Then $\exists Q_1,\ldots,Q_n \in Skeleton$ such that $Q \approx \vec{e}_1Q_1 \cap \ldots \cap \vec{e}_nQ_n$ and $type(Q_i) = \tau_i$, i=$1,\ldots,n$.

Proposition 2.2. Let $Q\langle Q_1,\ldots,Q_n\rangle \in Skeleton$, $n \geq 0$, and $Q'_1,\ldots,Q'_n \in Skeleton$. Then if $type(Q_i)$=$type(Q'_i)$ $\forall i$=$1,\ldots,n$, then $type\left(Q\langle Q_1,\ldots,Q_n\rangle\right)$=$type\left(Q\langle Q_1 := Q'_1,\ldots,Q_n := Q'_n\rangle\right)$.

Proposition 2.3. Let $Q\left\langle x^{:\tau_1},\ldots,x^{:\tau_n}\right\rangle \in Skeleton$ and only subskeletons $x^{:\tau_1},\ldots,x^{:\tau_n}$ have free occurrence in Q and $Q_1,\ldots,Q_n \in Skeleton$, where $x \in TermVariable$ and $\tau_1,\ldots,\tau_n \in Type$, $n \geq 0$. Then if $term\left(Q\left\langle x^{:\tau_1},\ldots,x^{:\tau_n}\right\rangle\right) = M_1$ and $term(Q_i) = M_2$ $\forall i$=$1,\ldots,n$, then $term\left(Q\left\langle x^{:\tau_1} := Q_1,\ldots,x^{:\tau_n} := Q_n\right\rangle\right) =$ $= M_1[x := M_2]$.

Proposition 2.4. Let $Q\left\langle x^{:\tau}\right\rangle \in Skeleton$ and $Q' \in Skeleton$ and $type(Q')$=τ, where $\tau \in Type$ and $x \in TermVariable$. Then $constraint\left(Q\left\langle x^{:\tau} := Q'\right\rangle\right)$= $=constraint\left(Q\left\langle x^{:\tau}\right\rangle\right) \cap$ *E-Path*$\left(Q\left\langle x^{:\tau}\right\rangle\right)constraint\ (Q')$.

Let us consider the term $M \notin \beta$-*NF* and one step of β-reduction: $M \to_\beta M'$. Now we are going to show that if $(A \vdash \tau)$ is a typing of term M, then it is also a typing of term M'.

Lemma 2.3. Let $Q\left\langle x^{:\tau_1},\ldots,x^{:\tau_n}\right\rangle \in Skeleton$ and only subskeletons $x^{:\tau_1},\ldots,x^{:\tau_n}$ have free occurrence in Q and $\vec{e}_i$= *E-Path* $\left(Q\left\langle x^{:\tau_i}\right\rangle\right)$, i=$1,\ldots,n$, where $\tau_1,\ldots,\tau_n \in Type$ and $x \in TermVariable$, $n \geq 0$. Then $env(Q)(x) = \vec{e}_1\tau_1 \cap \ldots \cap \vec{e}_n\tau_n$.

Proof. By induction on form of skeleton Q.

1. Let Q=ω^M, where $M \in Term$. We must show that $env(Q)(x) = \omega$. By the rule [OMEGA], $env(Q) = env_\omega \Rightarrow env(Q)(x) = \omega$.

2. Let Q=$c^{:\tau}$, where $c \in Constant$ and $\tau \in Type$. We must show that $env(Q)(x) = \omega$. By the rule [CONST], $env(Q) = env_\omega \Rightarrow env(Q)(x) = \omega$.

3. Let $Q{=}y^{:\tau}$, where $x \neq y \in TermVariable$ and $\tau \in Type$. We must show that $env(Q)(x) = \omega$. By the rule [VAR], $env(Q) = env_{\omega}[y \to \tau] \Rightarrow env(Q)(x) = \omega$.

4. Let $Q{=}x^{:\tau}$, where $\tau \in Type$. We must show that $env(Q)(x) = \tau$. By the rule [VAR], $env(Q) = env_{\omega}[x \to \tau] \Rightarrow env(Q)(x) = \tau$.

5. Let $Q{=}eQ'$, where $e \in ExpansionVariable$ and $Q' \in Skeleton$. Assume that only subskeletons $x^{:\tau_1},\ldots,x^{:\tau_n}$ have free occurrence in Q' and $\vec{e}_i'{=}\,E\text{-}Path\left(Q'\left\langle x^{:\tau_i}\right\rangle\right)$, where $i{=}1,\ldots,n$, $n \geq 0$. We must show that $env(Q)(x){=}$ ${=}\vec{e}_1\tau_1 \cap \ldots \cap \vec{e}_n\tau_n$. By induction hypothesis, $env(Q')(x){=}\vec{e}_1'\tau_1 \cap \ldots \cap \vec{e}_n'\tau_n$. By the rule [E-VAR], $env(Q){=}eenv(Q') \Rightarrow env(Q)(x){=}eenv(Q')(x){=}e(\vec{e}_1'\tau_1 \cap$ $\ldots \cap \vec{e}_n'\tau_n){=}\vec{e}_1\tau_1 \cap \ldots \cap \vec{e}_n\tau_n$.

6. Let $Q{=}(\lambda y.Q')$, where $y \in TermVariable$ and $Q' \in Skeleton$. Assume that only subskeletons $x^{:\tau_1},\ldots,x^{:\tau_n}$ have free occurrence in Q' and $\vec{e}_i'{=}\,E\text{-}Path\left(Q'\left\langle x^{:\tau_i}\right\rangle\right)$, where $i{=}1,\ldots,n$, $n \geq 0$. We must show that $env(Q)(x){=}$ ${=}\vec{e}_1\tau_1 \cap \ldots \cap \vec{e}_n\tau_n$. By induction hypothesis, $env(Q')(x){=}\vec{e}_1'\tau_1 \cap \ldots \cap \vec{e}_n'\tau_n$. By the rule [ABS], $env(Q) = env(Q')[y \to \omega] \Rightarrow env(Q)(x) = env(Q')(x) =$ $= \vec{e}_1'\tau_1 \cap \ldots \cap \vec{e}_n'\tau_n{=}\vec{e}_1\tau_1 \cap \ldots \cap \vec{e}_n\tau_n$.

7. Let $Q{=}(\lambda y.Q')$, where $Q' \in Skeleton$. We must show that $env(Q)(x) = \omega$. By the rule [ABS], $env(Q) = env(Q')[x \to \omega] \Rightarrow env(Q)(x) = \omega$.

8. Let $Q{=}(Q_1 \cap Q_2)$, where $Q_1, Q_2 \in Skeleton$. Assume that only subskeletons $x^{:\tau_1'},\ldots,x^{:\tau_m'}$ have free occurrence in Q_1 and only subskeletons $x^{:\tau_1''},\ldots,x^{:\tau_k''}$ have free occurrence in Q_2, and $\vec{e}_i'{=}\,E\text{-}Path\left(Q_1\left\langle x^{:\tau_i'}\right\rangle\right)$, $\vec{e}_j''{=}\,E\text{-}Path\left(Q_2\left\langle x^{:\tau_j''}\right\rangle\right)$, where $i{=}1,\ldots,m$, $j{=}1,\ldots,k$ and $m,k \geq 0$. We must show that $env(Q)(x) = \vec{e}_1'\tau_1' \cap$ $\ldots \cap \vec{e}_m'\tau_1' \cap \vec{e}_1''\tau_1'' \cap \ldots \cap \vec{e}_k''\tau_k''$. By induction hypothesis, $env(Q_1)(x) = \vec{e}\,'_1\tau'_1 \cap$ $\ldots \cap \vec{e}\,'_m\tau'_1$ and $env(Q_2)(x) = \vec{e}_1''\tau_1'' \cap \ldots \cap \vec{e}_k''\tau_k''$. By the rule [INT], $env(Q) = env(Q_1)$ $\cap env(Q_2) \Rightarrow env(Q)(x) = env(Q_1)(x) \cap env(Q_2)(x) = \vec{e}_1'\tau_1' \cap \ldots \cap \vec{e}_m'\tau_1' \cap \vec{e}_1''\tau_1'' \cap \ldots \cap \vec{e}_k''\tau_k''$.

9. Let $Q{=}(Q_1Q_2)^{:\tau}$, where $Q_1, Q_2 \in Skeleton$ and $\tau \in Type$. Assume that only subskeletons $x^{:\tau_1'},\ldots,x^{:\tau_m'}$ have free occurrence in Q_1, only subskeletons $x^{:\tau_1''},\ldots,x^{:\tau_k''}$ have free occurrence in Q_2, $\vec{e}_i'{=}\,E\text{-}Path\left(Q_1\left\langle x^{:\tau_i'}\right\rangle\right)$ and $\vec{e}_j''{=}\,E\text{-}Path\left(Q_2\left\langle x^{:\tau_j''}\right\rangle\right)$, where $i{=}1,\ldots,m$, $j{=}1,\ldots,k$ and $m,k \geq 0$. We must show that $env(Q)(x) = \vec{e}_1'\tau_1' \cap$ $\ldots \cap \vec{e}_m'\tau_1' \cap \vec{e}_1''\tau_1'' \cap \ldots \cap \vec{e}_k''\tau_k''$. By induction hypothesis, $env(Q_1)(x) = \vec{e}_1'\tau_1' \cap \ldots \cap \vec{e}_m'\tau_1'$ and $env(Q_2)(x) = \vec{e}_1''\tau_1'' \cap \ldots \cap \vec{e}_k''\tau_k''$. By the rule [APP], $env(Q) = env(Q_1) \cap$ $env(Q_2) \Rightarrow env(Q)(x) = env(Q_1)(x) \cap env(Q_2)(x) = \vec{e}_1'\tau_1' \cap \ldots \cap \vec{e}_m'\tau_1' \cap \vec{e}_1''\tau_1'' \cap \ldots \cap \vec{e}_k''\tau_k''$.

Lemma 2.4. Let $M_1, M_2 \in Term$ and $x \in TermVariable$. If $(A \vdash \tau)$ is a typing of term $((\lambda x.M_1)M_2)$, then it is also a typing of term $M_1[x := M_2]$.

Proof. Because $(A \vdash \tau)$ is a typing of term $((\lambda x.M_1)M_2)$, there exist $Q \in Skeleton$ and $\Delta \in Constraint$ such that the judgement $(((\lambda x.M_1)M_2) \rhd Q) : (A \vdash \tau) / \Delta$ is inferable, and Δ is solved. There are three cases to consider:

1. In the last step of inference of the judgement $(((\lambda x.M_1)M_2) \rhd Q) : (A \vdash \tau) / \Delta$ the rule [OMEGA] is used. Then $Q{=}\omega^{((\lambda x.M_1)M_2)}$, $(A \vdash \tau) = (env_\omega \vdash \omega)$ and $\Delta{=}\omega$. Because $(env_\omega \vdash \omega)$ is a typing of any term, it is also a typing of term $M_1[x := M_2]$.

2. In the last step of inference of the judgement $(((\lambda x.M_1)M_2) \rhd Q) : (A \vdash \tau) / \Delta$ rule [APP] is used. Then $Q{=}(Q_1Q_2)^{:\tau}$ and the judgements $((\lambda x.M_1) \rhd Q_1) : (A_1 \vdash \tau_1) / \Delta_1$ and $(M_2 \rhd Q_2) : (A_2 \vdash \tau_2) / \Delta_2$ are inferable, and $A{=}A_1 \cap A_2$ and $\tau{=}\tau_1 \cap \tau_2$, and $\Delta{=}\Delta_1 \cap \Delta_2 \cap (\tau_1 \doteq (\tau_2 \to \tau))$ (1). Because Δ is solved, (1) $\Rightarrow \tau_1{=}(\tau_2 \to \tau)$ (2), and constraints Δ_1, Δ_2 are solved. (2) $\Rightarrow$ in the last step of inference of the judgement $((\lambda x.M_1) \rhd Q_1) : (A_1 \vdash \tau_1) / \Delta_1$ the rule [ABS] is used. Hence, $Q_1{=}(\lambda x.Q')$ and the judgement $(M_1 \rhd Q') : (A' \vdash \tau') / \Delta'$ is inferable, and $A_1{=}A'[x \to \omega]$ and $\tau_1{=}(\tau_2 \to \tau) = (A'(x) \to \tau')$, and $\Delta_1{=}\Delta'$ (3). (3) $\Rightarrow \tau_2{=}A'(x)$ and $\tau{=}\tau'$, and constraint Δ' is solved (4). Assume that only subskeletons $x^{:\tau'_1}, \ldots, x^{:\tau'_n}$, $n \geq 0$, have free occurrence in Q'. By Lemma 2.3, $A'(x) = \vec{e}_1\tau'_1 \cap \ldots \cap \vec{e}_n\tau'_n$ (5), where $\vec{e}_i{=}E\text{-}Path\left(Q'\left\langle x^{:\tau'_i}\right\rangle\right)$, $i{=}1,\ldots,n$. (4),(5) $\Rightarrow$ $\Rightarrow \tau_2{=}\vec{e}_1\tau'_1 \cap \ \ldots \cap \vec{e}_n\tau'_n$ (6). By Proposition 2.1 and (6), $Q_2 \approx \vec{e}_1Q'_1 \cap \ldots \cap \vec{e}_nQ'_n$ and $type(Q'_i) = \tau'_i$, $i{=}1,\ldots,n$ (7). Let us consider the following skeleton: $Q'' = Q'\left\langle x^{:\tau'_1} := Q'_1, \ldots, x^{:\tau'_n} := Q'_n\right\rangle$. Now we will calculate $term(Q''), env(Q''),$ $type(Q'')$ and $constraint(Q'')$.

2a. (1), (3) $\Rightarrow term(Q') = term\left(Q'\left\langle x^{:\tau'_1}, \ldots, x^{:\tau'_n}\right\rangle\right)$ and $term(Q_2) = M_2$. (7) $\Rightarrow term(Q_2) = term(\vec{e}_1Q'_1 \cap \ldots \cap \vec{e}_nQ'_n) = term(Q') = \ldots = term(Q'_n) = M_2$. Hence by Proposition 2.3, $term(Q'') = M_1[x := M_2]$ (8).

2b. Let us show that $env(Q'')(y) = A(y)$ $\forall y \in TermVariable$ such that $y \neq x$. By Lemma 2.3, $A(y)$ depends on subskeletons of the form $y^{:\tau''}$ that have free occurrence in skeleton $Q{=}((\lambda x.Q')Q_2)$, and their *E-Paths* in that skeleton and $env(Q'')(y)$ depend on subskeletons of the form $y^{:\tau''}$ that have free occurrences in skeleton $Q'' = Q'\left\langle x^{:\tau'_1} := Q'_1, \ldots, x^{:\tau'_n} := Q'_n\right\rangle$ and their *E-Paths* in that skeleton. Due to (7), $Q \approx ((\lambda x.Q')(\vec{e}_1Q'_1 \cap \ldots \cap \vec{e}_nQ'_n))$. Hence, it is easy to see that the both skeletons Q and Q'' have the same free occurrences of subskeletons of the form $y^{:\tau''}$ with the same *E-Path* $\Rightarrow env(Q'')(y) = A(y)$ (9). Now let us show that $env(Q'')(x) =$ $= A(x)$. Assume that there is no subskeleton of form $x^{:\tau''}$ that have free occurrence in skeleton Q_2, otherwise, we would rename the bound variable x in skeleton

$(\lambda x.Q')$. Hence, it is easy to see that both skeletons Q and Q'' have no free occurrences of subskeletons of form $x^{:\tau''} \Rightarrow env(Q'')(x)=A(x)=\omega$ (10). (9), (10) $\Rightarrow env(Q'') = A$ (11).

2c. (3) $\Rightarrow type(Q') = type\left(Q'\left\langle x^{:\tau_1'},\dots,x^{:\tau_n'}\right\rangle\right) = \tau'$. By proposition 2.2, (4) and (7), $type(Q'') = type(Q') = \tau' = \tau \Rightarrow type(Q'') = \tau$ (12).

2d. (1),(3) $\Rightarrow constraint(Q_2)=\Delta_2$ and $constraint(Q')=\Delta'=\Delta_1$. (7) $\Rightarrow$ $\Rightarrow constraint(Q_2)=\vec{e}_1 constraint\ (Q_1') \cap \dots \cap \vec{e}_n constraint(Q_n')$ (13). By Proposition 2.4, $constraint(Q'')=constraint(Q') \cap$ $E\text{-}Path\left(Q'\left\langle x^{:\tau_1'}\right\rangle\right) constraint(Q_1') \cap \dots$ $\cap E\text{-}Path\left(Q'\left\langle x^{:\tau_n'}\right\rangle\right) constraint(Q_n') = constraint(Q') \cap \vec{e}_1 constraint(Q_1') \cap \dots \cap \vec{e}_n constraint(Q_n')$ (14). (13), (14) $\Rightarrow constraint(Q'') = \Delta_1 \cap \Delta_2$ (15). (8), (11), (12), (15) $\Rightarrow$ $\Rightarrow \left(M_1[x := M_2] \triangleright Q'\left\langle x^{:\tau_1'} := Q_1',\dots,x^{:\tau_n'} := Q_n'\right\rangle\right) : (A \vdash \tau) / \Delta_1 \cap \Delta_2$ is inferable, and constraints Δ_1, Δ_2 are solved. Hence, $(A \vdash \tau)$ is typing of term $M_1[x := M_2]$.

3. In the last step of inference of the judgement $(((\lambda x.M_1)M_2) \triangleright Q) : (A \vdash \tau) / \Delta$ the rules [OMEGA] and [APP] is not used. By Lemma 2.2, $Q \approx \vec{e}_1 Q_1 \cap \dots \cap \vec{e}_n Q_n$, $A=\vec{e}_1 A_1 \cap \dots \cap \vec{e}_n A_n$, $\tau=\vec{e}_1 \tau_1 \cap \dots \cap \vec{e}_n \tau_n$ and $\Delta=\vec{e}_1 \Delta_1 \cap \dots \cap \vec{e}_n \Delta_n$, and the following judgements are inferable: $(((\lambda x.M_1)M_2) \triangleright Q_i) : (A_i \vdash \tau_i) / \Delta$, $i=1,\dots,n$, $n \geq 1$. In our case in the last step of inference of the judgements $(((\lambda x.M_1)M_2) \triangleright Q_i) : (A_i \vdash \tau_i) / \Delta$, $i=1,\dots,n$, the rule [OMEGA] or rule [APP] is used $\Rightarrow$ by 1st and 2nd points of our proof, $(A_i \vdash \tau_i)$ is typing of term $M_1[x := M_2] \Rightarrow$ by the rules [INT], [E-VAR], $(\vec{e}_1 A_1 \cap \dots \cap \vec{e}_n A_n \vdash \vec{e}_1 \tau_1 \cap \dots \cap \vec{e}_n \tau_n) = (A \vdash \tau)$ is the typing of term $M_1[x := M_2]$.

Lemma 2.5. Let $M, M' \in Term$ and $M \to_\beta M'$. If $(A \vdash \tau)$ is a typing of term M, then it is also a typing of term M'.

Proof. Let us denote by M_β the β-redex corresponding to the one step of beta reduction $M \to_\beta M'$. We will prove Lemma by induction on the form of term M.

1. Let $M=M_\beta$. By Lemma 2.4, $(A \vdash \tau)$ is typing of term M'.

2. Let $M=(\lambda x.M_1)$, where $M_1 \in Term$ and M_β is subterm of M_1. $(A \vdash \tau)$ is typing of term $(\lambda x.M_1) \Rightarrow \exists Q \in Skeleton$, s.t. the judgement $((\lambda x.M_1) \triangleright Q) : (A \vdash \tau) / \Delta$ is inferable and Δ is solved. By Lemma 2.2, $Q \approx \vec{e}_1 Q_1 \cap \dots \cap \vec{e}_n Q_n$, $A=\vec{e}_1 A_1 \cap \dots \cap \vec{e}_n A_n$, $\tau=\vec{e}_1 \tau_1 \cap \dots \cap \vec{e}_n \tau_n$ and $\Delta=\vec{e}_1 \Delta_1 \cap \dots \cap \vec{e}_n \Delta_n$, and the following judgements are inferable: $((\lambda x.M_1) \triangleright Q_i) : (A_i \vdash \tau_i)/\Delta_i$, $i=1,\dots,n$, $n \geq 1$. In our case in the last step of inference of the judgements $((\lambda x.M_1) \triangleright Q_i) : (A_i \vdash \tau_i) / \Delta_i$, $i=1,\dots,n$, the rule [OMEGA] or rule [ABS] is used. Let us show that $(A_i \vdash \tau_i)$ is a typing of term M'. In that case $(A \vdash \tau) = (\vec{e}_1 A_1 \cap \dots \cap \vec{e}_n A_n \vdash \vec{e}_1 \tau_1 \cap \dots \cap \vec{e}_n \tau_n)$ will also be typing of term M'

(using the rules [E-VAR] and [INT]), which we need to prove. $M \to_\beta M' \Rightarrow \exists M_1' \in Term$ such that $M' = (\lambda x.M_1')$ and $M_1 \to_\beta M_1'$. There are two cases to consider:

2a. In the last step of inference of the judgement $((\lambda x.M_1) \rhd Q_i) : (A_i \vdash \tau_i)/\Delta_i$ the rule [OMEGA] is used. Then $(A_i \vdash \tau_i) = (env_\omega \vdash \omega)$. Because $(env_\omega \vdash \omega)$ is a typing of any term, it is also a typing of term M'.

2b. In the last step of inference of the judgement $((\lambda x.M_1) \rhd Q_i) : (A_i \vdash \tau_i)/\Delta_i$ the rule [ABS] is used. Then $\exists \tau_i' \in Type$ and environment A_i' such that $(A_i' \vdash \tau_i')$ is a typing of term M_1, $A_i = A_i'[x \to \omega]$ and $\tau_i = (A_i'(x) \to \tau_i')$. By induction hypothesis, $(A_i' \vdash \tau_i')$ is a typing of term $M_1' \Rightarrow (A_i \vdash \tau_i) = (A_i'[x \to \omega] \vdash (A_i'(x) \to \tau_i'))$ is a typing of term M' (using the rule [ABS]).

3. Let $M = (M_1 M_2)$, where $M_1, M_2 \in Term$ and M_β is a subterm of M_1. $(A \vdash \tau)$ is typing of term $(M_1 M_2) \Rightarrow \exists Q \in Skeleton$, s.t. the judgement $((M_1 M_2) \rhd Q) : (A \vdash \tau) / \Delta$ is inferable and Δ is solved. By Lemma 2.2, $Q \approx \vec{e}_1 Q_1 \cap \ldots \cap \vec{e}_n Q_n$, $A = \vec{e}_1 A_1 \cap \ldots \cap \vec{e}_n A_n$, $\tau = \vec{e}_1 \tau_1 \cap \ldots \cap \vec{e}_n \tau_n$ and $\Delta = \vec{e}_1 \Delta_1 \cap \ldots \cap \vec{e}_n \Delta_n$, and the following judgements are inferable: $((M_1 M_2) \rhd Q_i) : (A_i \vdash \tau_i) / \Delta_i$, $i = 1, \ldots, n$, $n \geq 1$. In our case in the last step of inference of the judgements $((M_1 M_2) \rhd Q_i) : (A_i \vdash \tau_i) / \Delta_i$, $i = 1, \ldots, n$, the rule [OMEGA] or rule [APP] is used. Let us show that $(A_i \vdash \tau_i)$ is a typing of term M'. In that case $(A \vdash \tau) = (\vec{e}_1 A_1 \cap \ldots \cap \vec{e}_n A_n \vdash \vec{e}_1 \tau_1 \cap \ldots \cap \vec{e}_n \tau_n)$ will also be typing of term M' (using the rules [E-VAR] and [INT]), which we need to prove. $M \to_\beta M' \Rightarrow \exists M_1' \in Term$ such that $M' = (M_1' M_2)$ and $M_1 \to_\beta M_1'$. There are two cases to consider:

3a. In the last step of inference of the judgement $((M_1 M_2) \rhd Q_i) : (A_i \vdash \tau_i) / \Delta_i$ the rule [OMEGA] is used. Then $(A_i \vdash \tau_i) = (env_\omega \vdash \omega)$. Because $(env_\omega \vdash \omega)$ is a typing of any term, it is also a typing of term M'.

3b. In the last step of inference of the judgement $((M_1 M_2) \rhd Q_i) : (A_i \vdash \tau_i) / \Delta_i$ the rule [APP] is used. Then $\exists \tau_i^1, \tau_i^2 \in Type$ and environments A_i^1, A_i^2 such that $(A_i^1 \vdash \tau_i^1)$ is a typing of term M_1, and $(A_i^2 \vdash \tau_i^2)$ is a typing of term M_2 and $A_i = A_i^1 \cap A_i^2$, $\tau_i^1 = (\tau_i^2 \to \tau_i)$. By induction hypothesis, $(A_i^1 \vdash \tau_i^1)$ is typing of term $M_1' \Rightarrow (A_i \vdash \tau_i) = (A_i^1 \cap A_i^2 \vdash \tau_i)$, is a typing of term $M' = (M_1' M_2)$ (using the rule [APP]).

4. Let $M = (M_1 M_2)$, where $M_1, M_2 \in Term$ and M_β is a subterm of M_2. The proof is similar to the proof of 3rd point.

2.2 Type Inference Algorithm and Principal Typing of Term. In this subsection we will prove that in case of success the type inference algorithm returns the principal typing of term. First of all let us consider terms that are in β-normal form.

Lemma 2.6. Let $M \in Term$ and $M \in \beta$-NF. Then if the judgement $(M \triangleright Q'):(A' \vdash \tau')/\Delta'$ is inferable, constraint Δ' is solved, and the rule [OMEGA] is not used during the inference of that judgement, then:

1. $(A \vdash \tau) = Typify(M)$, i.e. the type inference algorithm succeeds for input M.

2. $\exists E \in Expansion$, s.t. $A' = [E]A$ and $\tau' = [E]\tau$.

3. If in the last step of inference of the judgement $(M \triangleright Q'):(A' \vdash \tau')/\Delta'$ one of the rules [VAR], [CONST], [ABS] or [APP] is used, then the expansion E is a subtitution of the following form: $E=\{a_0 := \tau_{a_0}\}$ in case of [VAR]; $E=\varepsilon$ in case of [CONST]; $E=\{e_0 := E_{e_0}\}$ in case of [ABS]; $E=\{a_0 := \tau_{a_0}, e_1 := E_{e_1}, e_2 := E_{e_2}\}$ in case of [APP].

Proof. By induction on the form of term M.

1. Let $M=x$, where $x \in TermVariable$. Then by the type inference algorithm definition, $A=env_\omega[x \to a_0]$ and $\tau=a_0$, which is the proof of first part of Lemma's statement. There are two cases to consider:

1a. In the last step of inference of the judgement $(M \triangleright Q'):(A' \vdash \tau')/\Delta'$ the rule [VAR] is used. Then $Q' = x^{:\tau'}$, $A' = env_\omega[x \to \tau']$ and $\Delta' = \omega$. Let $E=\{a_0 := \tau'\} \Rightarrow A = [E]A$ and $\tau' = [E]\tau$, which we need to prove.

1b. In the last step of inference of the judgement $(M \triangleright Q'):(A' \vdash \tau')/\Delta'$ the rule [E-VAR] or rule [INT] is used. By Lemma 2.2, $Q' \approx \vec{e}_1 Q'_1 \cap \ldots \cap \vec{e}_n Q'_n$, $A'=\vec{e}_1 A'_1 \cap \ldots \cap \vec{e}_n A'_n$, $\tau'=\vec{e}_1 \tau'_1 \cap \ldots \cap \vec{e}_n \tau'_n$ and $\Delta' = \vec{e}_1 \Delta'_1 \cap \ldots \cap \vec{e}_n \Delta'_n$, and the following judgements are inferable: $(M \triangleright Q'_i):(A'_i \vdash \tau'_i)/\Delta'_i$, $i=1,\ldots,n$, $n \geq 1$. It is easy to see that the condition of Lemma holds also for the judgements $(M \triangleright Q'_i):(A'_i \vdash \tau'_i)/\Delta'_i$, $i=1,\ldots,n$. In our case in the last step of inference of the judgements $(M \triangleright Q'_i):(A'_i \vdash \tau'_i)/\Delta'_i$, $i=1,\ldots,n$, the rule [VAR] is used. Hence, by point 1a, $\exists E_1,\ldots,E_n \in Expansion$, s.t. $A'_i=[E_i]A_i$, $\tau'_i=[E_i]\tau_i$, $i=1,\ldots,n$ (1).

Let $E=\vec{e}_1 E_1 \cap \ldots \cap \vec{e}_n E_n$. By (1), $[E]A=\vec{e}_1[E_1]A \cap \ldots \cap \vec{e}_n[E_n]A=\vec{e}_1 A'_1 \cap \ldots \cap \vec{e}_n A'_n=A'$ and $[E]\tau=\vec{e}_1[E_1]\tau \cap \ldots \cap \vec{e}_n[E_n]\tau=\vec{e}_1\tau'_1 \cap \ldots \cap \vec{e}_n \tau'_n=\tau'$, which we need to prove.

2. Let $M=c$, where $c \in Constant$. Then by the type inference algorithm definition [1], $A=env_\omega$ and $\tau=\Sigma(c)$, which is the proof of first part of Lemma's statement. There are two cases to consider:

2a. In the last step of inference of the judgement $(M \triangleright Q'):(A' \vdash \tau')/\Delta'$ the rule [CONST] is used. Then $Q' = x^{:\Sigma(c)}$, $A' = env_\omega$, $\tau' = \Sigma(c)$ and $\Delta' = \omega$. Let $E=\varepsilon \Rightarrow A' = [E]A$ and $\tau' = [E]\tau$, which is to be proved.

2b. In the last step of inference of the judgement $(M \triangleright Q'):(A' \vdash \tau')/\Delta'$ the rule [E-VAR] or the rule [INT] is used. The proof is similar to the proof of point 1b.

3. Let $M=(\lambda x.M_1)$, where $x \in TermVariable$ and $M_1 \in Term$. Let $P_1=initial(M_1)$ and $P= initial(M)=(\lambda x.e_0 P_1) \Rightarrow constraint(P)=e_0 constraint(P_1)$. Hence, by definition of the unification algorithm [1] and unification rules $unify_\beta$,

$unify_x$, $unify_c$ [1], $\sigma = Unify(constraint(P)) \Leftrightarrow \sigma_1 = Unify\ (constraint(P_1))$, where $\sigma = e_0 / \sigma_1$ (2). Due to (2), due to the definition of the type inference algorithm [1] and definitions of algorithms *env* and *type* [1], $(A \vdash \tau) = Typify(M) \Leftrightarrow (A_1 \vdash \tau_1) = Typify(M_1)$, where $A = e_0 A_1[x \to \omega]$ and $\tau = (e_0 A_1(x) \to e_0 \tau_1)$ (3). There are two cases to consider:

3a. In the last step of inference of the judgement $(M \triangleright Q') : (A' \vdash \tau') / \Delta'$ the rule [ABS] is used. Then $Q' = (\lambda x.Q_1')$, $A' = A_1'[x \to \omega]$, $\tau' = (A_1'(x) \to \tau_1')$ and the judgement $(M_1 \triangleright Q_1') : (A_1' \vdash \tau_1') / \Delta_1'$ is inferable (4). It is easy to see that the condition of Lemma holds also for the judgement $(M_1 \triangleright Q_1') : (A_1' \vdash \tau_1') / \Delta_1'$. Hence, by the induction hypothesis, $Typify(M_1)$ succeeds and $\exists E_1 \in Expansion$, s.t. $A_1' = [E_1] A_1$ and $\tau_1' = [E_1]\tau_1$ (5). By (3) and (5), the first part of Lemma's statement is proved. Let $E = \{e_0 := E_1\}$. By (3), (4) and (5), $[E]A = [\{e_0 := E_1\}] e_0 A_1[x \to \omega] = [E_1]$, $A_1[x \to \omega] = A_1'[x \to \omega] = A'$ and $[E]\tau = [\{e_0 := E_1\}](e_0 A_1(x) \to e_0 \tau_1) = ([E_1] A_1(x) \to [E_1]\tau_1) = (A_1'(x) \to \tau_1') = \tau'$, which is to be proved.

3b. In the last step of the inference of the judgement $(M \triangleright Q') : (A' \vdash \tau') / \Delta'$ the rule [E-VAR] or the rule [INT] is used. The proof is similiar to the proof of point 1b.

4. Let $M = (M_1 M_2)$. We will not present the proof of this case.

Lemma 2.7. Let $M \in Term$ and $M \in \beta$-*NF*. Then if $(A' \vdash \tau')$ is a typing of term M and $(A \vdash \tau) = Typify(M)$, then:

1. $\exists E \in Expansion$, s.t. $A' = [E]A$ and $\tau' = [E]\tau$.

2. If in the last step of inference of the judgement $(M \triangleright Q') : (A' \vdash \tau') / \Delta'$ one of the rules [VAR], [CONST], [ABS] or [APP] is used, then $E = \omega$ or E is a subtitution of the following form: $E = \{a_0 := \tau_{a_0}\}$ in case of [VAR]; $E = \varepsilon$ in case of [CONST]; $E = \{e_0 := E_{e_0}\}$ in case of [ABS]; $E = \{a_0 := \tau_{a_0}, e_1 := E_{e_1}, e_2 := E_{e_2}\}$ in case of [APP].

Proof. The proof is very similiar to the proof of Lemma 2.6.

Now let us present the main theorem on the principal typing of a term.

Theorem 2.1. Let $M \in Term$ and $M \in Term$, s.t. $M \twoheadrightarrow_\beta M'$ and $M' \in \beta$-*NF*.

1. If there exists a typing of term M' such that during the inference of the corresponding judgement the rule [OMEGA] is not used, then $Typify(M)$ succeeds.

2. If $(A \vdash \tau) = Typify(M)$, then $(A \vdash \tau)$ is the principal typing of term M.

Proof. Let $\Delta = constraint(initial(M))$ and $\Delta' = constraint(initial(M'))$. By Lemma 2.12 of [1], $Unify(\Delta) = [Unify(\Delta')]\sigma$, where $\sigma = [\sigma_m] \ldots [\sigma_2]\sigma_1$ (1), and substitutions $\sigma_1, \ldots, \sigma_m$, $m \geq 0$, are created by the rule $unify_\beta$ during the work of the unification algorithm for input Δ. Hence, by definition of the type inference algorithm, both $Typify(M)$ and $Typify(M')$ are simultaneously executed or fail.

1. Because there exists a typing of term M' such that during the inference of the corresponding judgement the rule [OMEGA] is not used, then due to Lemma 2.6, $\mathit{Typify}(M')$ succeeds $\Rightarrow \mathit{Typify}(M)$ succeeds as well, which is proves the first part of the Theorem.

2. We have that $(A \vdash \tau) = \mathit{Typify}(M)$. By Lemma 2.6 of [1], each application of the rule unify_β corresponds to one step of β-reduction. Hence, $\exists M_1,\ldots,M_m \in \mathit{Term}$, such that $M{=}M_0 \to_\beta M_1 \to_\beta \ldots \to_\beta M_m{=}M'$, $A_i{=}[\sigma_i]A_{i-1}$, $\tau_i{=}[\sigma_i]\tau_{i-1}$, where $A_j{=}\mathit{env}(\mathit{initial}(M_j))$ and $\tau_j{=}\mathit{type}(\mathit{initial}\ (M_j))$, $i{=}1,\ldots,m$, $j{=}0,\ldots,m$ (2). (1),(2) $\to A_m{=}\mathit{env}(\mathit{initial}(M')){=}[\sigma]A_0{=}[\sigma]A$ and $\tau_m{=}\mathit{type}(\mathit{initial}$ $(M')){=}[\sigma]\tau_0{=}[\sigma]\tau$ (3). By (1) and (3), $(A \vdash \tau) = \mathit{Typify}(M) = ([[\mathit{Unify}(\Delta')]\sigma]A_0 \vdash$ $\vdash [[\mathit{Unify}(\Delta')]\sigma]\tau_0) = ([\mathit{Unfiy}(\Delta')][\sigma]A_0 \vdash [\mathit{Unfiy}(\Delta')][\sigma]\tau_0) = ([\mathit{Unify}(\Delta)]A_0 \vdash [\mathit{Unify}(\Delta)]\tau_0) =$ $= ([\mathit{Unfiy}(\Delta')]A_m \vdash [\mathit{Unfiy}(\Delta')]\tau_m) = \mathit{Typify}(M')$ (4). Let $(A' \vdash \tau')$ is a typing of term M. By Lemma 2.5, $(A' \vdash \tau')$ is also a typing of term $M' \in \beta$-*NF*. Hence by (4) and Lemma 2.7, $\exists E \in \mathit{Expansion}$, s.t. $A' = [E]A$ and $\tau' = [E]\tau$, which means that $\mathit{Typify}(M)$ is the principal typing of term M.

Remark 2.1. The type inference algorithm returns the principal typing of a term that has a β-normal form, except for the situations, when it is impossible to type a β-normal form of the given term without using the rule [OMEGA]. For terms that do not have a β-normal form the type inference algorithm never returns.

Received 06.04.2009

REFERENCES

1. **Arakelyan A.H.** Proceedings of the YSU, Phys. Math. Sciences, 2009, № 3, p. 42–51.
2. **Milner R.** Journal of Computer and System Sciences, 1978, № 17, p. 348–375.
3. **Wells J.B.** In Proc. 29th Int'l Coll. Automata, Languages, and Programming. V. 2380 of LNCS. Springer-Verlag, 2002.
4. **Jim T.** What are Principal Typings and What are They Good for? In Conf. Rec. POPL '96: 23rd ACM Symp. Princ. of Prog. Langs., 1996.
5. **Carlier S., Wells J.B.** Type Inference with Expansion Variables and Intersection Types in System E and an Exact Correspondence with β-reduction. In Proc. 6th Int'l Conf. Principles & Practice Declarative Programming, 2004.
6. **Carlier S., Polakow J., Wells J.B., Kfoury A.J.** System E: Expansion Variables for Flexible Typing with Linear and Non-linear Types and Intersection Types. In Programming Languages & Systems, 13th European Symp. Programming. V. 2986 of LNCS. Springer-Verlag, 2004.
7. **Barendregt H.P.** The Lambda Calculus: Its Syntax and Semantics. Amsterdam, North Holand, 1981.

Proc. of the Yerevan State Univ. Phys. and Mathem. Sci., 2010, № 1, p. 37–46.

Ա. Հ. Առաքելյան

Պոլիմորֆ -թերմերի տիպային կոռեկտության մասին: 2

Աշխատանքում դիտարկվում են պոլիմորֆ -թերմերը, որոնցում չկա ինֆորմացիա փոփոխականների տիպերի մասին: Աշխատանքի նպատակն է ապացուցել, որ [1]-ում ներկայացված փոփոխականների տիպի արտածման ալգորիթմը տիպայնացնում է այդպիսի թերմերն ամենաընդհանուր ձևով:

А. Г. Аракелян.

О типовой корректности полиморфных λ -термов. 2

В работе рассматриваются полиморфные λ -термы, в которых отсутствует информация о типах переменных. Цель даной работы – доказать, что представленный в [1] алгоритм типизации выводит самый общий тип таких термов.