Information Processing Letters 45 (1993) 1–3.

Normal Forms have Partial Types

Jens Palsberg
palsberg@daimi.aau.dk

Computer Science Department, Aarhus University
Ny Munkegade, DK–8000 Aarhus C, Denmark

Abstract

We prove that every λ-term in normal form has one of Thatte's partial types.

Keywords: Functional programming, partial types.

Partial types for the pure λ-calculus [1] were introduced by Thatte in 1988 [5] as a way to type certain λ-terms that are untypable in the simply-typed λ-calculus. Any λ-term that has a simple type also has a partial type. Moreover, any λ-term that has a partial type is strongly normalizing [6].

Type inference for partial types can be performed in cubic time, as demonstrated by Kozen and Schwartzbach together with the present author [3]. Our algorithm improved the exponential time algorithm of O'Keefe and Wand [4].

In this paper we prove that every λ-term in normal form has a partial type. This property is shared by few other type systems, one example being the simple intersection types of Coppo and Giannini [2].

The set of partial types is defined by the grammar

$$t ::= \Omega \mid t \to t$$

Partial types are ordered by $\leq$ as follows:

1. $t \leq \Omega$ for any t;
2. $s \to t \leq s' \to t'$ if and only if $s' \leq s$ and $t \leq t'$

Thus, partial types have a largest type Ω and involve the usual contravariant rule for function types. Typical inclusions are $\Omega \to \Omega \leq \Omega$ and $\Omega \to \Omega \leq (\Omega \to \Omega) \to \Omega$. Intuitively, $s \leq t$ allows a coercion from s to t that forgets some type structure. The type Ω contains only the information "well-typed". O'Keefe and

Wand [4] have shown that $\leq$ is a partial order. Note that there is no least partial type, and that $\leq$ is not well-founded.

Given a λ-term E, the question of whether it has the partial type t can be phrased using the judgement $A \vdash E : t$, where A is a type environment, i.e., a finite association of variables with partial types; if $A \vdash E : t$ is derivable, then E has the partial type t. There are four rules:

$$A \vdash x : t \quad (A(x) = t)$$

$$\frac{A[x \leftarrow t_1] \vdash E : t_2}{A \vdash (\lambda x.E) : t_1 \to t_2}$$

$$\frac{A \vdash E_1 : t_1 \to t_2 \quad A \vdash E_2 : t_1}{A \vdash (E_1 E_2) : t_2}$$

$$\frac{A \vdash E : t \quad t \leq t'}{A \vdash E : t'}$$

The first three rules are those of type inference for simple types and the last rule is that of subsumption.

As an example of a λ-term that does not have a simple type but does have a partial type, consider $\lambda f.(fK(fI))$, where $K = \lambda x.(\lambda y.x)$ and $I = \lambda z.z$. This λ-term has partial type $(\Omega \to (\Omega \to \Omega)) \to \Omega$ but no simple type.

As an example of a λ-term that does not even have a partial type, consider $(\lambda x.xx)(\lambda x.xx)$. Since this term is not strongly normalizing, it cannot have a partial type. A direct proof of the untypability of $(\lambda x.xx)(\lambda x.xx)$ has been presented by O'Keefe and Wand [4].

Definition 1 An occurrence of a subterm of a λ-term E is said to be *maximal* in E if it is not the function part of an application in E. If E_1 is a subterm of a λ-term E, then the *weight* of E_1 in E is the greatest natural number in the set $\{n \mid E_1 E_2 \ldots E_n \text{ is maximal in } E\}$. We use the convention that application associates to the left. $\square$

For example, the weight of f in $\lambda f.(fK(fI))$ is 3.

Definition 2 If E is a λ-term, then T_E maps subterms and bound variables of E to partial types as follows. If E' is a subterm of E, then define $T_E(E') = \Omega^n$, where n is the weight of E' in E, and where, for $n > 0$, the symbol Ω^n is inductively defined by $\Omega^1 = \Omega$ and $\Omega^{n+1} = \Omega \to \Omega^n$. If x is a bound variable in E that does not occur otherwise in E, then define $T_E(x) = \Omega$. Finally, A_E is T_E restricted to variables. $\square$

For example, for $E = \lambda f.(fK(fI))$, we get that $T_E(f) = \Omega^3$.

Theorem 3 *If E is a λ-term in normal form and E' is a subterm of E, then $A_E \vdash E' : T_E(E')$ is derivable.*

Proof. We proceed by induction in the structure of E'. In the base case, consider x. Clearly, $A_E \vdash x : T_E(x)$ is derivable, since $A_E(x) = T_E(x)$.

In the induction step, consider first $\lambda x.E''$. By the induction hypothesis $A_E \vdash E'' : T_E(E'')$ is derivable. Since $A_E = A_E[x \leftarrow T_E(x)]$ we get that also $A_E \vdash (\lambda x.E'') : T_E(x) \rightarrow T_E(E'')$ is derivable. Since E is in normal form, $\lambda x.E''$ is maximal in E and the weight of $\lambda x.E''$ in E is 1. Hence, $T_E(\lambda x.E'') = \Omega \geq T_E(x) \rightarrow T_E(E'')$, so, using the subsumption rule, $A_E \vdash (\lambda x.E'') : T_E(\lambda x.E'')$ is derivable.

Consider then E_1E_2. Let n be the weight of E_1 in E. Notice that $n > 1$ and that E_1E_2 has weight $n-1$ in E. By the induction hypothesis both $A_E \vdash E_1 : \Omega^n$ and $A_E \vdash E_2 : T_E(E_2)$ are derivable. Hence, using the subsumption rule, also $A_E \vdash E_2 : \Omega$ is derivable. Since $\Omega^n = \Omega \rightarrow \Omega^{n-1}$, we get that $A_E \vdash (E_1E_2) : T_E(E_1E_2)$ is derivable. □

Corollary 4 *Every λ-term in normal form has a partial type.*

Proof. Immediate from theorem 3. □

Acknowledgement. The author thanks Torben Amtoft, Uffe Engberg, Fritz Henglein, Kim Skak Larsen, Peter Mosses, Hanne Riis Nielson, Michael Schwartzbach, Satish Thatte, Mitchell Wand, and the anonymous referees for helpful comments on drafts of the paper.

References

[1] Henk P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* North-Holland, 1981.

[2] Mario Coppo and Paola Giannini. A complete type inference algorithm for simple intersection types. In *Proc. CAAP'92*, pages 102–123. Springer-Verlag (*LNCS* 581), 1992.

[3] Dexter Kozen, Jens Palsberg, and Michael I. Schwartzbach. Efficient inference of partial types. *Journal of Computer and System Sciences*, 49(2):306–324, 1994. Also in Proc. FOCS'92, 33rd IEEE Symposium on Foundations of Computer Science, pages 363–371, Pittsburgh, Pennsylvania, October 1992.

[4] Patrick M. O'Keefe and Mitchell Wand. Type inference for partial types is decidable. In *Proc. ESOP'92, European Symposium on Programming*, pages 408–417. Springer-Verlag (*LNCS* 582), 1992.

[5] Satish Thatte. Type inference with partial types. In *Proc. International Colloquium on Automata, Languages, and Programming 1988*, pages 615–629. Springer-Verlag (*LNCS* 317), 1988.

[6] Mitchell Wand and Patrick M. O'Keefe. Partially typed terms are strongly normalizing. Manuscript, December 1991.