3E−2

古典的証明に基づく値呼びプログラミング言語の型推論アルゴリズムについて

堺 美子　藤田 憲悦*
九州工業大学 情報工学部 知能情報工学科

1 はじめに

カリー・ハーワードの原理 [H80] は，直観主義論理と型付き λ 計算との同型対応を与える重要な原理であり，プログラミングに応用されている．この原理は Griffin[G90] によって古典論理まで拡張された．一方，我々は単純型理論に排中律に相当する推論規則を追加して，古典命題論理の体系である λ^{v}_{exc}[F97] を提案した．そこでは排中律に相当する推論規則に関する簡約規則により継続の取り扱いが可能である．本稿では λ^{v}_{exc} に再帰プログラムと多相型を導入して拡張した体系を提案する．そして，拡張した体系に対する型推論アルゴリズム $\mathcal{W}$ を与え，$\mathcal{W}$ の健全性と完全性の証明をする．

2 古典命題論理に基づく値呼び言語

本稿で導入する値呼び言語の項 (プログラム)M，型 τ，型スキーム σ，型環境 Γ を，以下に定義する．ここで，c は定数，b は型定数 ($\mathtt{bool}, \bot$ を含む)，型 $\neg\tau$ は $\tau \to \bot$ を表す．変数は，x と y の 2 種類を用意し，y は型 $\neg\tau$ に対してのみ宣言されるものとする．また，$\tau \leq \forall\vec{\alpha}.\tau'$ を，$\tau = [\tau_1/\alpha_1, \cdots, \tau_n/\alpha_n]\tau'$ の時と定義する．ただし，$\vec{\alpha} = \alpha_1 \cdots \alpha_n$ とし，τ_i は型であり，$[\tau_1/\alpha_1, \cdots, \tau_n/\alpha_n]$ は，型変数 α_i への τ_i の代入を表す $(1 \leq i \leq n)$.

$$
\begin{aligned}
M &::= c \mid x \mid \lambda x.M \mid MM \\
&\quad \mid \mathtt{if}\ M\ \mathtt{then}\ M\ \mathtt{else}\ M \\
&\quad \mid \mathtt{let}\ x{=}M\ \mathtt{in}\ M \mid yM \\
&\quad \mid \mathtt{fix}\ x.M \mid \mathtt{raise}(M) \mid \{y\}M
\end{aligned}
$$

$$
\begin{aligned}
\tau &::= b \mid \alpha \mid \tau \to \tau \\
\sigma &::= \tau \mid \forall\alpha.\sigma \\
\Gamma &::= \langle\rangle \mid x{:}\sigma, \Gamma \mid y{:}\neg\tau, \Gamma \mid c{:}\sigma, \Gamma
\end{aligned}
$$

項に対する型判定システムを次に定義する．

$$\Gamma \vdash c : \tau \quad (\tau \leq \Gamma(c))$$

$$\Gamma \vdash x : \tau \quad (\tau \leq \Gamma(x))$$

$$\frac{\Gamma, x{:}\tau_1 \vdash M : \tau}{\Gamma \vdash \lambda x.M : \tau_1 \to \tau}\ (\to I)$$

$$\frac{\Gamma \vdash M_1 : \tau_1 \to \tau \quad \Gamma \vdash M_2 : \tau_1}{\Gamma \vdash M_1 M_2 : \tau}\ (\to E)$$

$$\frac{\Gamma \vdash M : \mathtt{bool} \quad \Gamma \vdash M_1 : \tau \quad \Gamma \vdash M_2 : \tau}{\Gamma \vdash \mathtt{if}\ M\ \mathtt{then}\ M_1\ \mathtt{else}\ M_2 : \tau}\ (\mathrm{if})$$

$$\frac{\Gamma, x{:}\tau \vdash M : \tau}{\Gamma \vdash \mathtt{fix}\ x.M : \tau}\ (\mathrm{fix})$$

$$\frac{\Gamma \vdash M_1 : \tau' \quad \Gamma,\ x{:}\forall\vec{\alpha}.\tau' \vdash M : \tau}{\Gamma \vdash \mathtt{let}\ x{=}M_1\ \mathtt{in}\ M : \tau}\ (\mathrm{let})$$

$$\frac{\Gamma \vdash\ M : \tau}{\Gamma \vdash\ yM\ : \bot}\ (\bot I)\ \ if\ \Gamma(y) \equiv \neg\tau$$

$$\frac{\Gamma \vdash\ M : \bot}{\Gamma \vdash\ \mathtt{raise}(M)\ : \tau}\ (\bot E)$$

$$\frac{\Gamma, y{:}\neg\tau \vdash M : \tau}{\Gamma \vdash \{y\}M : \tau}\ (exc)$$

ただし，(let) の適用では $\vec{\alpha}$ は Γ に自由に出現しない型変数の列である．

3 型推論アルゴリズム

先に定義した言語に対し，型環境 Γ と項 M を入力として，型の代入 S と主要型 τ を返す型推論アルゴリズムが存在する．項 M の主要型が τ_p であるとは，次の時である:
(1) ある型環境 Γ に対して $\Gamma \vdash M : \tau_p$ であり，かつ
(2) ある型環境 Γ' と型 τ に対し $\Gamma' \vdash M : \tau$ ならば，ある代入 S に対して $\tau = S\tau_p$ である．
型推論アルゴリズム $\mathcal{W}$ の定義を [M78, DM82, O97] にしたがって以下に与える．ここで，2

Type Inference Algorithm for Call-by-Value Languages based on Classical Proofs
Yoshiko Sakai and Ken-etsu Fujita
Department of Artificial Intelligence, Kyushu Institute of Technology

* 本研究の一部は実吉奨学会の援助を受けている

つの型を型変数への適当な代入によって構文的に同一な型にするために単一化アルゴリズム $\mathcal{U}$ を援用する.

$$\mathcal{W}(\Gamma; M) = (S, \tau)$$

1) M が定数, 又は変数の時 ($n \geq 0$ として)
$\Gamma(M)$ が $\forall\alpha_1\forall\alpha_2\cdots\forall\alpha_n.\tau_1$ であれば,
$\tau = [\beta_1/\alpha_1, \beta_2/\alpha_2, \cdots, \beta_n/\alpha_n]\tau_1$
($\beta_1, \beta_2, \cdots, \beta_n$ は新しい型変数)
$S = I$(恒等代入);

2), 3), 4), 5) M が $\lambda x.M_1$, M_1M_2,
`if` M_0 `then` M_1 `else` M_2, 又は
`fix` $x.M_1$ の時は省略する;

6) M が `let` $x = M_1$ `in` M_2 の時
$\mathcal{W}(\Gamma; M_1) = (S_1, \tau_1)$
$\mathcal{W}(S_1\Gamma, x{:}\forall\vec{\alpha}.\tau_1; M_2) = (S_2, \tau_2)$
($\vec{\alpha}$ は $S_1\Gamma$ に現れない τ_1 のすべての自由な型変数の並び) であり,
$S = S_2 \circ S_1$, $\tau = \tau_2$;

7) M が yM_1 の時
$\Gamma(y) = \neg\tau_1$
$\mathcal{W}(\Gamma; M_1) = (S_1, \tau_2)$
$U = \mathcal{U}(\tau_1, \tau_2)$ であり,
$S = U \circ S_1$, $\tau = \bot$;

8) M が `raise`(M_1) の時
$\mathcal{W}(\Gamma; M_1) = (S_1, \tau_1)$
$U = \mathcal{U}(\tau_1, \bot)$ であり,
$S = U \circ S_1$, $\tau = \alpha$ (α は新しい型変数);

9) M が $\{y\}M_1$ の時
$W(\Gamma, y{:}\neg\alpha; M_1) = (S_1, \tau_1)$
(α は新しい型変数)
$U = [\tau_1/\alpha]$ であり,
$S = U \circ S_1$, $\tau = \tau_1$.

$\mathcal{W}$ は型判定システムに対して健全かつ完全であることが項の構造に関する帰納法で証明できる.

定理 1 (健全性) 任意の項 M と型環境 Γ について, もし $W(\Gamma; M) = (S, \tau)$ なら, $S\Gamma \vdash M : \tau$ である.

定理 2 (完全性) 任意の項 M と型環境 Γ について $S\Gamma \vdash M : \tau$ となる型の代入 S と型 τ が存在するなら, $W(\Gamma; M) = (S_1, \tau_1)$, かつある型の代入 S_2 が存在して, $S\Gamma = S_2 \circ S_1\Gamma$, かつ $\tau = S_2\tau_1$ である.

上の定理より, $\mathcal{W}$ は Γ の下で主要型を計算することがわかる.

4 おわりに

$\mathcal{W}$ は作用的プログラミングのスタイルで与えられているが, 実現の際には参照型を利用した効率的アルゴリズム $\mathcal{I}$ が知られている [M78]. そのような型推論アルゴリズムも容易に構成できる. また, 本言語は, 関数型言語 ML の `callcc` と次の意味で対応している.

$\lceil$`callcc`$(M)\rceil = \{y\}(\lceil M\rceil(\lambda x.yx))$;
$\lceil$`throw` M_1 $M_2\rceil$ = `raise`$(\lceil M_1\rceil\lceil M_2\rceil)$.
$\lfloor\{y\}M\rfloor$ = `callcc`$(\lambda y.\lfloor M\rfloor)$;
$\lfloor$`raise`$(yM)\rfloor$ = `throw` y $\lfloor M\rfloor$.

`callcc` の型 $((1\alpha$ `cont` $\to 1\alpha) \to 1\alpha)$ は, 弱い型変数に制限されている. 一方, 我々の言語では, 1α `cont` に対応する y の型 $(\neg\tau)$ は型変数を含む型で定義されており, ML では型エラーになる次のようなプログラム F

```
let f = callcc(λk.λx.throw k (λx'.x))
        in (λx'.λx.x)(f1)(f true)
```

も $\lceil F\rceil$: `bool` と型付け可能である.
ただし, $\lceil F\rceil$ は

```
let f = {y}(λx.raise(y(λx'.x))
        in (λx'.λx.x)(f1)(f true)
```

であり, その評価結果は ML で期待される値と異なり, 簡約規則により `true` となる.

参考文献

[DM82] Damas, L., Milner, R.: *Principal type-schemes for functional programs*, Proc. 9th POPL, 207–212, 1982.

[F97] Fujita, K.: *Calculus of Classical Proofs I*, LNCS **1345**, 321–335, 1997.

[G90] Griffin, T.G.: *A Formulae-as-Types Notion of Control*, Proc. 17th POPL, 47–58, 1990.

[H80] Howard, W.: *The Formulae-as-Types Notion of Constructions*, Academic Press, 479-490, 1980.

[HDM93] Harper, R., Duba, B.F., MacQueen, D.: *Typing First-Class Continuations in ML*, J. Func. Prog. **3**, 465–484, 1993.

[M78] Milner, R.: *A Theory of Type Polymorphism in Programming*, J. Comp. Sys. Sci. **17**, 348–375, 1978.

[O97] 大堀 淳: プログラミング言語の基礎理論, 共立出版, 148–236, 1997.