型付きλ計算(続)

Jacques Garrigue, 2007年6月15日

単純型付ラムダ計算は，型なしラムダ計算の表現力を大きく制限している．論理との関係を見ても，表現できるのはせいぜい命題論理である．しかし，型の表現力を増やす，表現力の大部分を取り戻したり，もっと強い論理を表現できたりする．

1 二次ラムダ計算 λ2

型なしラムダ計算で利用していた表現方法を見ると，あるパターンがよく現れる．

$$
\begin{aligned}
&\mathsf{t} = \lambda x.\lambda y.x \\
&\mathsf{f} = \lambda x.\lambda y.y \\
&\mathsf{pair}\ a\ b = \lambda s.(sab) \\
&\mathsf{fst} = \lambda p.p\mathsf{t} \\
&\mathsf{snd} = \lambda p.p\mathsf{f}
\end{aligned}
$$

これらに共通するのは，引数の型を固定できないことである．型なしλ計算で任意のものが渡せるから使えるが，型付λ計算ではうまく行かない．同じ項が複数の型として使えることを「多相性」という．

二次λ計算は多相型を導入することによって，上記の表現を使えるようにする．

$$
\begin{array}{rcll}
t & ::= & \alpha & \text{型変数} \\
 & | & \forall\alpha.t & \text{多相型} \\
 & | & t \to t & \text{関数型} \\
M & ::= & x \mid \lambda x{:}t.M \mid (MM) & \\
 & | & \Lambda\tau.M & \text{型抽象} \\
 & | & M[t] & \text{型適用}
\end{array}
$$

新しい推論規則が必要になる．

$$
\textbf{型抽象}\quad \frac{\Gamma, \alpha : * \vdash M : \tau}{\Gamma \vdash \Lambda\alpha.M : \forall\alpha.\tau}
$$

$$
\textbf{型適用}\quad \frac{\Gamma \vdash M : \forall\alpha.\tau}{\Gamma \vdash M[t] : [t/\alpha]\tau}
$$

上記の項は以下の型が付く．

$$
\begin{array}{ll}
\mathsf{Bool} = \forall\alpha.\alpha \to \alpha \to \alpha & \\
\vdash \mathsf{t} = \Lambda\alpha.\lambda x{:}\alpha.\lambda y{:}\alpha.x & : \mathsf{Bool} \\
\vdash \mathsf{f} = \Lambda\alpha.\lambda x{:}\alpha.\lambda y{:}\alpha.y & : \mathsf{Bool} \\
\mathsf{Pair}[t_1, t_2] = \forall\gamma.(t_1 \to t_2 \to \gamma) \to \gamma & \\
\vdash \mathsf{pair} = \Lambda\alpha.\Lambda\beta.\lambda a{:}\alpha.\lambda b{:}\alpha.\Lambda\gamma.\lambda s{:}\alpha \to \beta \to \gamma.(s\ a\ b) & : \forall\alpha.\forall\beta.\alpha \to \beta \to \mathsf{Pair}[\alpha,\beta] \\
\vdash \mathsf{fst} = \Lambda\alpha.\Lambda\beta.\lambda p{:}\mathsf{Pair}[\alpha,\beta].p[\alpha]\ (\lambda x{:}\alpha.\lambda y{:}\beta.x) & : \forall\alpha.\forall\beta.\mathsf{Pair}[\alpha,\beta] \to \alpha \\
\vdash \mathsf{snd} = \Lambda\alpha.\Lambda\beta.\lambda p{:}\mathsf{Pair}[\alpha,\beta].p[\beta]\ (\lambda x{:}\alpha.\lambda y{:}\beta.y) & : \forall\alpha.\forall\beta.\mathsf{Pair}[\alpha,\beta] \to \beta
\end{array}
$$

チャーチ自然数も同様に型が付く.

$$
\begin{array}{ll}
\mathsf{Nat} = \forall\alpha.(\alpha \to \alpha) \to \alpha \to \alpha & \\
\vdash \mathsf{c}_n = \Lambda\alpha.\lambda f : \alpha \to \alpha.\lambda x : \alpha.f^n x & : \mathsf{Nat} \\
\vdash \mathsf{c}_+ = \lambda m : \mathsf{Nat}.\lambda n : \mathsf{Nat}.\Lambda\alpha.\lambda f : \alpha \to \alpha.\lambda x : \alpha.(m[\alpha]\ x\ (n[\alpha]\ f\ x)) & : \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat} \\
\vdash \mathsf{c}_\times = \lambda m : \mathsf{Nat}.\lambda n : \mathsf{Nat}.\Lambda\alpha.\lambda f : \alpha \to \alpha.(m[\alpha]\ (n[\alpha]\ f)) & : \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat} \\
\vdash \mathsf{c}_{\mathsf{exp}} = \lambda m : \mathsf{Nat}.\lambda n : \mathsf{Nat}.\Lambda\alpha.n[\alpha \to \alpha]\ (m[\alpha]) & : \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat}
\end{array}
$$

小さな拡張で表現力が大きく上がる．ただし，二次型付λ計算は単純型付λ計算を持っているので，δ規則なしに Y を表現することができない．逆に言えば，Y さえ追加すれば，Turing マシンと同等の計算能力を持つ.

たとえば，リストを以下のような型で表現できる.

$$\mathsf{List}[t] = \forall\alpha.(t \to \alpha \to \alpha) \to \alpha \to \alpha$$

2 依存型とλP

上記の二次λ計算は計算能力という面で単純型付λ計算を拡張するが，論理の面では相変わらず命題論理にしか対応していない．述語論理に対応するために，異なる種類の拡張が必要になる.

述語論理の命題は多相型に似ている.

$$
\begin{array}{rcll}
t & ::= & x \mid a \mid f(t,\ldots) & \text{項} \\
A & ::= & \bot \mid A \to A \mid A \wedge A \mid A \vee A & \\
 & \mid & p(t,\ldots) & \text{述語} \\
 & \mid & \forall x.A & \text{全称} \\
 & \mid & \exists x.A & \text{存在}
\end{array}
$$

しかし，よく見ると，ここでは t や x で表されているのは型ではなく，項である．そこで，二次λ計算と少し違う拡張が考えられる．二次λ計算では項の中に型が入ってもよかったが，今度は型の中に項を入れる.

$$
\begin{array}{rcll}
t & ::= & b \mid \bot \mid p_t\ M \mid \Pi x{:}t.t \mid t \times t & \text{型} \\
M & ::= & x \mid c_t \mid \lambda x{:}t.M \mid (MM) \mid (M,M) & \text{項}
\end{array}
$$

二つ目の型に x が現れない場合，$\Pi x{:}t_1.t_2$ (依存関数型) は $t_1 \to t_2$ と書ける．

推論規則も増やす．特に，型の構成を確認する必要がある．

型 $\dfrac{\Gamma \vdash M : t}{\Gamma \vdash p_t\ M\ \mathrm{ok}}$ $\dfrac{\Gamma, x : t \vdash t'\ \mathrm{ok}}{\Gamma \vdash \Pi x{:}t.t'\ \mathrm{ok}}$ $\dfrac{\Gamma, x : t \vdash t'\ \mathrm{ok}}{\Gamma \vdash \Sigma x{:}t.t'\ \mathrm{ok}}$

抽象 $\dfrac{\Gamma, x : t \vdash M : t'}{\Gamma \vdash \lambda x{:}t.M : \Pi x{:}t.t'}$

適用 $\dfrac{\Gamma \vdash M : \Pi x{:}t.t' \quad \Gamma \vdash N : t}{\Gamma \vdash (M\ N) : [N/x]t'}$

変換 $\dfrac{\Gamma \vdash M : [N/x]t \quad N =_{\beta\delta} N'}{\Gamma \vdash M : [N'/x]t}$

否定 $\dfrac{\Gamma \vdash M : \bot \quad \Gamma \vdash t\ \mathrm{ok}}{\Gamma \vdash M : t}$

以上の定義を使えば，述語論理を表現することができる．実は，述語論理で表現できないものも入ってしまうので，準同型にしかならないが，さらなる制限を加えると同型になる．

例えば,「人間は死ぬ，ソクラテスは人間である，すなわちソクラテスは死ぬ」は以下の型として表現できる．

$$(\Pi x{:}Name.Human\ x \to Mortal\ x) \to Human\ Socrates \to Mortal\ Socrates$$

以下の項によって証明される．

$$\lambda mortal{:}(\Pi x{:}Name.Human\ x \to Mortal\ x).\lambda human{:}(Human\ Socrates).mortal\ Socrates\ human$$

式「$\forall x.x + x = 2 \times x$」も以下のように書ける．

$$\Pi x{:}\mathsf{Nat}.eqnat(add\ x\ x, mult\ 2\ x)$$

δ 規則は以下のとおり．

$$\begin{aligned}
&add\ 0\ n \to n\\
&add\ (sm)\ n \to s\ (add\ m\ n)\\
&mult\ 0\ n \to 0\\
&mult\ (sm)\ n \to add\ n\ (mult\ m\ n)
\end{aligned}$$

以下の定数 (公理) を使って証明できる．

$$\begin{aligned}
&add_sym : \Pi m{:}\mathsf{Nat}.\Pi n{:}\mathsf{Nat}.eqnat(add\ m\ n, add\ n\ m)\\
&eq_sub : \Pi f{:}(\mathsf{Nat} \to \mathsf{Nat}).\Pi m{:}\mathsf{Nat}.\Pi n{:}\mathsf{Nat}.eqnat(m, n) \to eqnat(f\ m, f\ n)
\end{aligned}$$

以下の項によって証明される．

$$\lambda x{:}\mathsf{Nat}.eq\_sub\ (\lambda y{:}\mathsf{Nat}.add\ x\ y)\ (add\_sym\ 0\ x)$$

3 λP2と証明器

λPでは述語論理が表現できるものの，まだ一般的な原理が書けないので，それほど強くない．しかし，λPとλ2を組み合わせると表現力が劇的に増え，実際の定理証明に使われる．