ソフトウェア基礎論配布資料 (4)
λ計算 (2)

五十嵐 淳
京都大学 大学院情報学研究科知能情報学専攻
e-mail: igarashi@kuis.kyoto-u.ac.jp

平成 16 年 10 月 26 日

1 形式化にむけて: 代入操作と一般化された変数参照

これまで β 簡約は β 基 $(\lambda x.t_1)\ t_2$ から t_1 中の x を t_2 で置換して得る，と説明してきた．この置換する操作を，**代入**(*substitution*) と呼び,「t_1 中の x を t_2 で置換して得」た項を $[t_1/x]t_2$ と記述する．

この代入操作の定義は基本的には t_1 の項構造を再帰的にみていき，x をみつけたら t_2 で置き換える，という操作であるが，いくつか注意が必要である．

具体的な例として $(\lambda\mathtt{x}.\lambda\mathtt{y}.\mathtt{x})\ \mathtt{y}$ という β 基を考える．この関数部分 $f \equiv \lambda\mathtt{x}.\lambda\mathtt{y}.\mathtt{x}$ は，ふたつの引数をとって最初のものを返す関数である．つまり，任意の項 t_1, t_2 に対し，$f\ t_1\ t_2 \longrightarrow_f^* t_2$ であることが期待される．しかし，$(\lambda\mathtt{x}.\lambda\mathtt{y}.\mathtt{x})\ \mathtt{y}$ から,「$\lambda\mathtt{y}.\mathtt{x}$ 中の $\mathtt{x}$ を $\mathtt{y}$ で文字通り置換」すると $\lambda\mathtt{y}.\mathtt{y}$ が得られる．これは恒等関数なので，$(\lambda\mathtt{x}.\lambda\mathtt{y}.\mathtt{x})\ \mathtt{y}\ t_2 \longrightarrow_f \lambda\mathtt{y}.\mathtt{y}\ t_2 \longrightarrow_f t_2$ になってしまう．

この原因は，β 基引数の $\mathtt{y}$ は自由変数であったのに，簡約後の項では束縛されてしまうことにある．一般的には t_2 内の自由変数 x が t_1 の中に $\lambda x.$ として出現する場合に，文字通りの置換を行なうと x が誤って束縛されてしまう．この現象を，自由変数が捕われる (capture される) という．正しい代入操作は，この capture を何らかの手段で避けるように定義する必要がある．

標準的な解決策としては，

- $\lambda\mathtt{x}.\mathtt{x}$ と $\lambda\mathtt{y}.\mathtt{y}$ のように束縛変数だけ違うような項は「等しい」とみなし (α **同値**(α *equivalence*) と呼ぶ),
- 簡約中に capture が発生しそうになったら「適宜」α 同値な項に置き換える

ということが行なわれる．上の例であれば，$(\lambda\mathtt{x}.\lambda\mathtt{y}.\mathtt{x})\ \mathtt{y}\ t_2$ は簡約時に，$(\lambda\mathtt{x}.\lambda\mathtt{z}.\mathtt{x})\ \mathtt{y}\ t_2$ と capture が発生しないように名前のつけ替えが行なわれ，$\longrightarrow_f (\lambda\mathtt{z}.\mathtt{y})\ t_2 \longrightarrow_f t_2$ となる．

上の解決策を数学的に定義しようとすると，(多くの人が目をつぶって見逃す) 微妙な点がいくつかあるので，ここでは別の策を取る．名前替えを行なう理由は，$\lambda\mathtt{y}.$ の有効範囲内で $\mathtt{y}$ という名前の自由変数を参照する手段がないため，とも言えるので,「有効範囲の外に飛び出す」ための機構を設けることで解決を図る．

具体的には，これまで変数項としてきたものを拡張し，$\#^i x$ という形の**変数参照**(*variable reference*) 項を導入する．x が，一番近くで宣言された x を参照しているのに対し，$\#^1 x$ という記法は「一番近

くの $\lambda x.$ の**外側の** x」という意味になる．上の例の簡約の過程は

$$\begin{aligned}(\lambda\mathtt{x}.\lambda\mathtt{y}.\mathtt{x})\ \mathtt{y}\ t_2 &\longrightarrow_f \lambda\mathtt{y}.\#^1\mathtt{y}\ t_2\\ &\longrightarrow_f \mathtt{y}\end{aligned}$$

と表現される．中間の項に出現する $\#^1\mathtt{y}$ は，自由変数の $\mathtt{y}$ を示すことになる．

2 形式的定義

2.1 項の文法

2.1.1 定義: λ 項の集合は以下の文法で定義される．

$$t ::= \#^i x \mid \lambda x.t \mid t\ t$$

□

$\#^0 x$ は単に x と略記される．$t_1\ t_2\ t_3$ は $(t_1\ t_2)\ t_3$ のことである．$\lambda x.t_1\ t_2$ は $\lambda x.(t_1\ t_2)$ を意味する．(有効範囲ができるだけ大きくなるように解釈する)．

項 t の自由変数参照の集合を $FV(t)$ と書き，以下のように t に関して帰納的に定義される．

$$\begin{aligned}FV(\#^i x) &= \{\#^i x\}\\ FV(\lambda x.t_0) &= \{\#^i y \mid \#^i y \in FV(t_0)\ \&\ x \neq y\} \cup \{\#^{i-1} x \mid \#^i x \in FV(t_0)\ \&\ n > 0\}\\ FV(t_1\ t_2) &= FV(t_1) \cup FV(t_2)\end{aligned}$$

2.2 shifting, 代入

上で見たように，$[t_1/x]\lambda y.t$ のように λ の奥に代入をしようとする時には，x に代入される t_1 中の自由変数 y を $\#^1 y$ に修正する動作が必要である．この変数参照の数字の操作を shifting 操作と呼び $\Uparrow_x t$ と記述する．例えば $\Uparrow_y(\mathtt{y}\ (\lambda\mathtt{y}.\mathtt{y}\ \#^1\mathtt{y})) \equiv \#^1\mathtt{y}\ (\lambda\mathtt{y}.\mathtt{y}\ \#^2\mathtt{y})$ である．

$\Uparrow_x t$ は以下のように $\Uparrow_x^j t$ という t に関して帰納的に以下のように定義される関数を経由して定義される．

$$\begin{aligned}\Uparrow_x t &= \Uparrow_x^0 t\\ \Uparrow_x^j \#^i x &= \#^{i+1} x && \text{if } i \geq j\\ \Uparrow_y^j \#^i x &= \#^i x && \text{if } i < j \text{ or } x \neq y\\ \Uparrow_x^j \lambda x.t_0 &= \lambda x.(\Uparrow_x^{j+1} t_0)\\ \Uparrow_y^j \lambda x.t_0 &= \lambda x.(\Uparrow_y^j t_0) && \text{if } x \neq y\\ \Uparrow_x^j (t_1\ t_2) &= (\Uparrow_x^j t_1)\ (\Uparrow_x^j t_2)\end{aligned}$$

これを用いて，項 t_2 中の自由変数参照 $\#^i x$ に t_1 を代入した項 $[t_1/\#^i x]t_2$ を以下のように t_2 に関する帰納法で定義する．

$$\begin{aligned}[t/\#^i x]\#^i x &= t\\ [t/\#^i x]\#^j y &= \#^j y && \text{if } i \neq j \text{ or } x \neq y\\ [t/\#^i x]\lambda x.t_0 &= \lambda x.([\Uparrow_x t/\#^{i+1} x]t_0\\ [t/\#^i x]\lambda y.t_0 &= \lambda x.([\Uparrow_y t/\#^i x]t_0 && \text{if } x \not\equiv y\\ [t/\#^i x](t_1\ t_2) &= ([t/\#^i x]t_1)\ ([t/\#^i x]t_2)\end{aligned}$$

また，$\Uparrow_x t$ とは逆の，変数参照の数字を減らす操作 $\Downarrow_x t$ も定義しておく．$\Downarrow_x t$ は $x \notin FV(t)$ の時のみ定義される．

$$
\begin{array}{lcll}
\Downarrow_x t & = & \Downarrow_x^0 t & \\
\Downarrow_x^j \#^i x & = & \#^{i-1} x & \text{if } i \geq j \text{ and } i > 0 \\
\Downarrow_y^j \#^i x & = & \#^i x & \text{if } i < j \text{ or } x \neq y \\
\Downarrow_x^j \lambda x.t_0 & = & \lambda x.(\Downarrow_x^{j+1} t_0) & \\
\Downarrow_y^j \lambda x.t_0 & = & \lambda x.(\Downarrow_y^j t_0) & \text{if } x \neq y \\
\Downarrow_x^j (t_1\ t_2) & = & (\Downarrow_x^j t_1)\ (\Downarrow_x^j t_2) &
\end{array}
$$

2.3 簡約

β 簡約は，代入操作と shifting を使って，

$$(\lambda x.t_1)\ t_2 \longrightarrow \Downarrow_x([(\Uparrow_x t_2)/x]t_1)$$

と表すことができる．shifting が必要なのは，t_1 は x の束縛が消えるため，変数参照 $\#^i x$ を $\#^{i-1} x$ に調整しておく必要がある．t_2 は，添字があとで減らされる分，予め増やしておく必要があるので，このような定義になる．

以下は，前回触れた様々な簡約関係の形式的な定義である．

2.3.1 定義 [full β-reduction]: λ 項上の二項関係 $\longrightarrow_f$ は以下の規則で定義される．

$$\frac{}{(\lambda x.t_1)\ t_2 \longrightarrow_f \Downarrow_x([(\Uparrow_x t_2)/x]t_1)} \qquad \frac{t_1 \longrightarrow_f t_1'}{t_1\ t_2 \longrightarrow_f t_1'\ t_2}$$

$$\frac{t \longrightarrow_f t'}{\lambda x.t \longrightarrow_f \lambda x.t'} \qquad \frac{t_2 \longrightarrow_f t_2'}{t_1\ t_2 \longrightarrow_f t_1\ t_2'}$$

□

2.3.2 定義 [normal order reduction]: λ 項上の二項関係 $\longrightarrow_n$ は以下の規則で定義される．

$$\frac{}{(\lambda x.t_1)\ t_2 \longrightarrow_n \Downarrow_x([(\Uparrow_x t_2)/x]t_1)} \qquad \frac{na_1 \longrightarrow_n na_1'}{na_1\ t_2 \longrightarrow_n na_1'\ t_2}$$

$$\frac{t \longrightarrow_n t'}{\lambda x.t \longrightarrow_n \lambda x.t'} \qquad \frac{t_2 \longrightarrow_n t_2'}{nanf_1\ t_2 \longrightarrow_n nanf_1\ t_2'}$$

ただし，規則中の nf, $nanf$, na は，それぞれ以下の文法で定義される (特殊な形の) 項を示す．

$$
\begin{array}{ll}
nf & ::= \lambda x.nf \mid nanf \\
nanf & ::= x \mid nanf\ nf \\
na & ::= x \mid t_1\ t_2
\end{array}
$$

□

2.3.3 定義: 値 (メタ変数 v) の集合を以下の文法で定義する.

$$v ::= \#^i x \mid \lambda x.t$$

2.3.4 定義 [call-by-value reduction]: λ 項上の二項関係 $\longrightarrow_v$ は以下の規則で定義される.

$$\frac{}{(\lambda x.t_1)\ v_2 \longrightarrow_v \Downarrow_x([(\Uparrow_x v_2)/x]t_1)} \qquad \frac{t_1 \longrightarrow_v t_1'}{t_1\ t_2 \longrightarrow_v t_1'\ t_2} \qquad \frac{t_2 \longrightarrow_v t_2'}{v_1\ t_2 \longrightarrow_v v_1\ t_2'}$$

□

2.3.5 定義 [call-by-name reduction]: λ 項上の二項関係 $\longrightarrow_{cbn}$ は以下の規則で定義される.

$$\frac{}{(\lambda x.t_1)\ t_2 \longrightarrow_{cbn} \Downarrow_x([(\Uparrow_x t_2)/x]t_1)} \qquad \frac{t_1 \longrightarrow_{cbn} t_1'}{t_1\ t_2 \longrightarrow_{cbn} t_1'\ t_2}$$

□

3 自然数 + 真偽値をもつ値呼び λ 計算

3.1 動機

- Church 数表現は full β reduction (か normal order reduction) でないと，きれいにいかない. (`plus 1 1` $\longrightarrow_v$ λ`s.λz.1 s (1 s z)` であり 2 には簡約されない. ただし，振舞としては λ`s.λz.1 s (1 s z)` も 2 の代わりに使える.)
- すこぶる読みにくい
- `plus true 1` みたいな式でも，簡約がすすむ. (その結果，意味のわからない λ 項が得られる.)

$\Longrightarrow$ 自然数・真偽値などの基本的なデータ型を言語のプリミティブとして導入する.

3.2 定義

3.2.1 定義: 項 t, 数値 n, 値 v は以下の文法で定義される.

$$\begin{aligned} t ::=&\ \#^i x \mid \lambda x.t \mid t\ t \mid \texttt{fix} \\ \mid&\ \texttt{0} \mid \texttt{S(}t\texttt{)} \mid t\ \texttt{+}\ t \mid t\ \texttt{*}\ t \\ \mid&\ \texttt{true} \mid \texttt{false} \mid \texttt{if}\ t\ \texttt{then}\ t\ \texttt{else}\ t \ \square \\ n ::=&\ \texttt{0} \mid \texttt{S(}n\texttt{)} \\ v ::=&\ \#^i x \mid \lambda x.t \mid n \mid \texttt{true} \mid \texttt{false} \end{aligned}$$

$FV(t)$, $\Uparrow_x t$, $\Downarrow_x t$, $[t_1/x]t_2$ は同様な定義.

3.2.2 定義: 簡約関係 $\longrightarrow_v$ は以下の規則で定義される.

$$\frac{t \longrightarrow_v t'}{\texttt{S}(t) \longrightarrow_v \texttt{S}(t')} \quad \text{(Ev-Succ)}$$

$$\frac{}{(\lambda x.t_1)\ v_2 \longrightarrow_v \Downarrow_x([(\Uparrow_x v_2)/x]t_1)} \quad \text{(Ev-AbsApp)}$$

$$\frac{t_1 \longrightarrow_v t_1'}{t_1 \texttt{ + } t_2 \longrightarrow_v t_1' \texttt{ + } t_2} \quad \text{(Ev-PlusL)}$$

$$\frac{t_1 \longrightarrow_v t_1'}{t_1\ t_2 \longrightarrow_v t_1'\ t_2} \quad \text{(Ev-App1)}$$

$$\frac{t_2 \longrightarrow_v t_2'}{v_1 \texttt{ + } t_2 \longrightarrow_v v_1 \texttt{ + } t_2'} \quad \text{(Ev-PlusR)}$$

$$\frac{t_2 \longrightarrow_v t_2'}{v_1\ t_2 \longrightarrow_v v_1\ t_2'} \quad \text{(Ev-App2)}$$

$$\frac{t_1 \longrightarrow_v t_1'}{t_1 \texttt{ * } t_2 \longrightarrow_v t_1' \texttt{ * } t_2} \quad \text{(Ev-MultL)}$$

$$\frac{}{\texttt{fix}\ (\lambda x.t_0) \longrightarrow_v \Downarrow_x[\Uparrow_x(\texttt{fix}\ (\lambda x.t_0))/x]t_0} \quad \text{(Ev-FixAbs)}$$

$$\frac{t_2 \longrightarrow_v t_2'}{v_1 \texttt{ * } t_2 \longrightarrow_v t_1 \texttt{ * } a_2'} \quad \text{(Ev-MultR)}$$

$$\frac{t_0 \longrightarrow_v t_0'}{\texttt{fix}\ t_0 \longrightarrow_v \texttt{fix}\ t_0'} \quad \text{(Ev-Fix)}$$

$$\frac{}{n_0 \texttt{ + 0} \longrightarrow_v n_0} \quad \text{(Ev-PlusZero)} \qquad \frac{}{\texttt{if true then}\ t_1\ \texttt{else}\ t_2 \longrightarrow_v t_1} \quad \text{(Ev-IfTrue)}$$

$$\frac{}{n_1 \texttt{ + S(}n_2\texttt{)} \longrightarrow_v \texttt{S(}n_1 \texttt{ + } n_2\texttt{)}} \quad \text{(Ev-PlusSucc)} \qquad \frac{}{\texttt{if false then}\ t_1\ \texttt{else}\ t_2 \longrightarrow_v t_2} \quad \text{(Ev-IfFalse)}$$

$$\frac{}{n_0 \texttt{ * 0} \longrightarrow_v \texttt{0}} \quad \text{(Ev-MultZero)}$$

$$\frac{t_0 \longrightarrow_v t_0'}{\texttt{if}\ t_0\ \texttt{then}\ t_1\ \texttt{else}\ t_2 \longrightarrow_v \texttt{if}\ t_0'\ \texttt{then}\ t_1\ \texttt{else}\ t_2} \quad \text{(Ev-If)}$$

$$\frac{}{n_1 \texttt{ * S(}n_2\texttt{)} \longrightarrow_v n_1 \texttt{ * } n_2 \texttt{ + } n_1} \quad \text{(Ev-MultSucc)}$$

□