TAPL Chap6
Nameless Representation of Terms

Kinebuchi Tomohiko

2011/01/09

まえがき

第 5 章で「束縛変数の名前付け替え」(α 変換) を扱った.
この方法は人間に分かりやすいし, 証明もしやすい.
しかし実装には向かない.
できれば一意な形になって欲しい.

代入の扱い方

1. 変数を記号で扱って, 名前付け替えで対応 (第 5 章での方法)
2. 変数を記号で扱うが, 束縛変数は他のどの変数とも名前がかぶらない. ただし代入で不安定. (Barendregt convention)
3. ある一意の形で項を表す. (この章で扱う方法)
4. explicit substitutions
5. 変数を使わない方法を取る. Combinatory logic

どれを選ぶかは好き好き.

代入の扱い方

この本では 3 番目の de Bruijn による方法を選ぶ.
この方法で実装すると複雑な例にも対応できるし, 間違えたときにすぐ分かる.

6.1 Terms and Contexts

de Bruijn のアイディアは読みづらいけど直接的.
変数を数字で書いてしまう.
これを de Bruijn index と呼ぶ.
コンパイラのを書く人は同じ概念を “static distance” と呼んだりする.

6.1.1 Exercise

やってみましょう. 端から順に当ててきます.
plus に誤植があります.
(n z s) の部分は (n s z) のはず.

6.1.2 Definition

Definition

$\mathcal{T} = \{\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \ldots\}$

1. $\mathsf{k} \in \mathcal{T}_n \Leftrightarrow 0 \leq \mathsf{k} < n$
2. $\mathsf{t}_1 \in \mathcal{T}_n \ (n > 0) \Rightarrow \lambda.\mathsf{t}_1 \in \mathcal{T}_{n-1}$
3. $\mathsf{t}_1 \in \mathcal{T}_n, \mathsf{t}_2 \in \mathcal{T}_n \Rightarrow (\mathsf{t}_1\mathsf{t}_2) \in \mathcal{T}_n$

5.3.1 (P.69) のような形式で定義する. 5.3.1 と違うのは自由変数の個数をきっちり見てるところ.
$\mathcal{T}_n$ (n-terms) =「多くとも n 個しか自由変数が無い項」t が閉項 (自由変数が 0 個) なら全ての $\mathcal{T}_n$ に属する.

閉項の場合 de Bruijn 表現は一意に決まる.
もちろん名前の付け替えで等しい項は de Bruijn 表現で同じ表現になる.
自由変数がある場合を扱うために naming context を用意する.
naming context = あらかじめ自由変数用の文字を用意して, その使う順序を決めたもの.

$$\begin{aligned}\Gamma = \mathsf{x} &\mapsto 4\\ \mathsf{y} &\mapsto 3\\ \mathsf{z} &\mapsto 2\\ \mathsf{a} &\mapsto 1\\ \mathsf{b} &\mapsto 0\end{aligned}$$

重なるとまずいので登場する束縛変数のの数だけずらして使う.

例

λx. x は λ. 0
x (y z) は 4 (3 2)
λw. y w は途中まで変換すると λ. y 0
ここで元々 w だった 0 と重ならないように context をずらす.

$$\begin{aligned}\Gamma = \mathsf{x} &\mapsto 5\\ \mathsf{y} &\mapsto 4\\ \mathsf{z} &\mapsto 3\\ \mathsf{a} &\mapsto 2\\ \mathsf{b} &\mapsto 1\end{aligned}$$

なので λ. 4 0
λw. λa. x は 2 つずれて λ. λ. 6

6.1.3 Definition

いちいち書いてると面倒なので naming context を
$\Gamma = \mathsf{x}_n, \mathsf{x}_{n-1}, \ldots, \mathsf{x}_0$ と書いて, $x_i \mapsto i$ という対応付けを表すとする.
$dom(\Gamma) \equiv \{\mathsf{x}_n, \mathsf{x}_{n-1}, \ldots, \mathsf{x}_0\}$

context をずらす操作についてもう少し

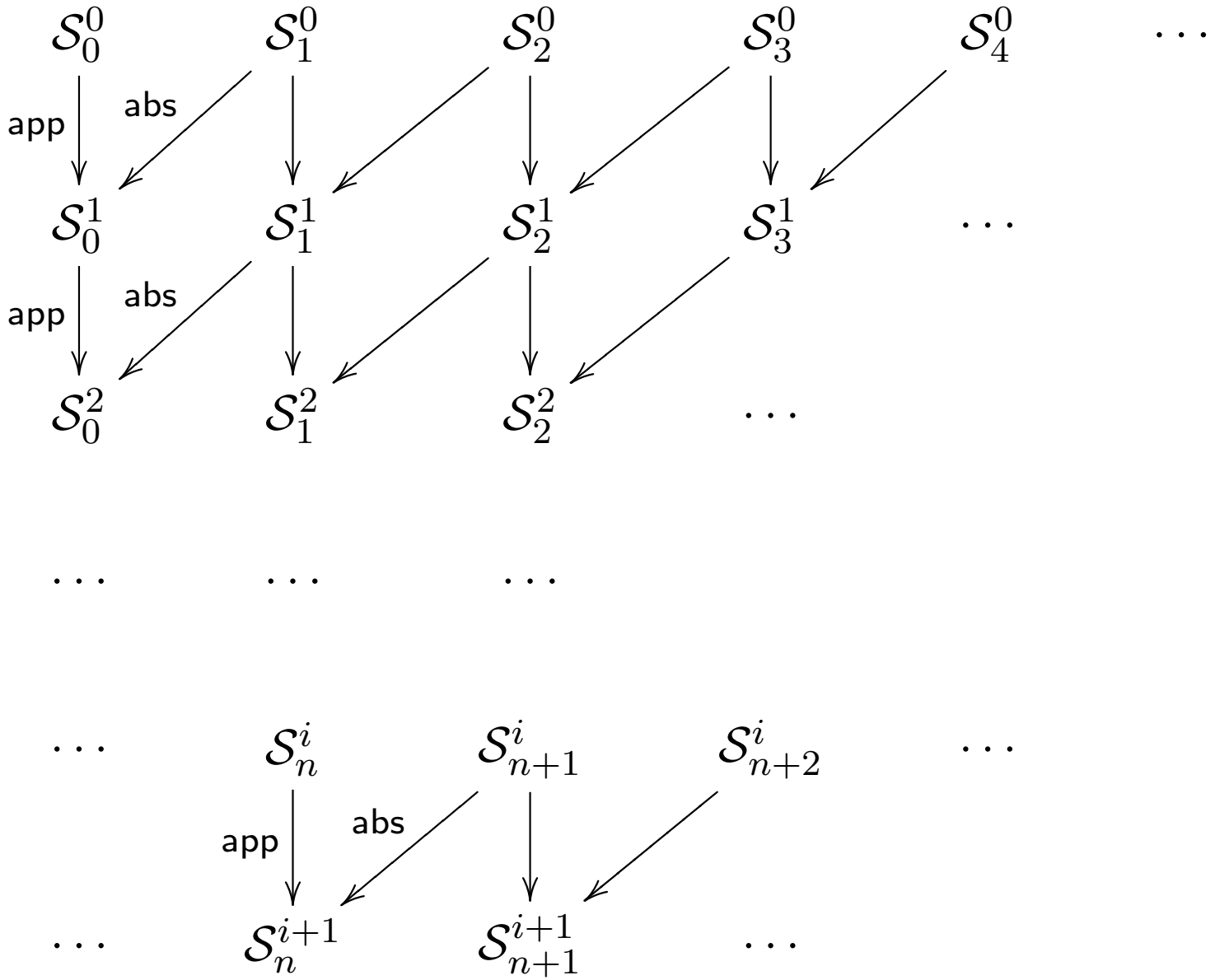
$\lambda w.\ \lambda a.\ x$ の 1 番外側では context は $\Gamma = x, y, z, a, b.$
もちろん, これらは全て自由変数.
↓
abstraction の 1 つ内側 ($\lambda a.\ x$ の部分) に入るとき w と衝突しないようにずらさないといけない.
新しい context $\Gamma' = x, y, z, a, b, w$ になると考えてもよい.
↓
もう一度 abstraction の内側 (x の部分) に入るとき a と衝突しないようにずらさないといけないので,
新しい context $\Gamma' = x, y, z, a, b, w, a$ になる.

6.1.4 Exercise

さっきの $\mathcal{T} = \{\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \ldots\}$ を 3.2.3 (P. 27) みたいに定義し直して, 最初の定義と同等であることを示せ.
以下のように構成すれば良い.

$$
\begin{aligned}
\mathcal{S}_n^{i+1} =& \{0, 1, 2, ..., n-1\} \\
& \cup \{\lambda.t | t \in \mathcal{S}_{n+1}^i\} \\
& \cup \{t_1 t_2 | t_1, t_2 \in \mathcal{S}_n^i\} \\
\mathcal{S}_n =& \bigcup_i \mathcal{S}_n^i
\end{aligned}
$$

ちなみにこの定義から $\mathcal{S}_n \subset \mathcal{S}_{n+1}$ が分かる.


$\mathcal{S}_0^0$ $\mathcal{S}_1^0$ $\mathcal{S}_2^0$ $\mathcal{S}_3^0$ $\mathcal{S}_4^0$ $\cdots$
app abs
$\mathcal{S}_0^1$ $\mathcal{S}_1^1$ $\mathcal{S}_2^1$ $\mathcal{S}_3^1$ $\cdots$
app abs
$\mathcal{S}_0^2$ $\mathcal{S}_1^2$ $\mathcal{S}_2^2$ $\cdots$
$\cdots$ $\cdots$ $\cdots$
$\cdots$ $\mathcal{S}_n^i$ $\mathcal{S}_{n+1}^i$ $\mathcal{S}_{n+2}^i$ $\cdots$
app abs
$\cdots$ $\mathcal{S}_n^{i+1}$ $\mathcal{S}_{n+1}^{i+1}$ $\cdots$

6.1.5 Exercise

通常の項の表現から de Bruijn 表現へ変換する関数
$removenames_{\Gamma}(\mathrm{t})$ を書け. $FV(\mathrm{t}) \subset dom(\Gamma)$
de Bruijn 表現から通常の項の表現へ変換する関数
$restorenames_{\Gamma}(\mathrm{t})$ を書け.
当然この 2 つは逆関数の関係にないといけない.
基本のアイディアは 11 シート目の内容

巻末の解答について

application の場合は context は変化しないと思う.
変化してしまうと 6.2.1 の定義と食い違うし, x がどこから出てきたか分からない. 誤植?
以下が正しいと思う.

$$removenames_\Gamma(t_1\ t_2) = removenames_\Gamma(t_1)\ removenames_\Gamma(t_2)$$

6.2 Shifting and Substitution

代入を扱う前にシフト操作を定義する.
11 シート目の内容を厳密に定義する.

6.2.1 Definition

これを使うと自由変数だけシフトすることができる.

Definition

$$\uparrow_c^d (\mathsf{k}) = \begin{cases} \mathsf{k} & \text{if } \mathsf{k} < c \\ \mathsf{k} + d & \text{if } \mathsf{k} \geq c \end{cases}$$

$$\uparrow_c^d (\lambda.\ \mathsf{t}_1) = \lambda.\ \uparrow_{c+1}^d (\mathsf{t}_1)$$

$$\uparrow_c^d (\mathsf{t}_1\ \mathsf{t}_2) = \uparrow_c^d (\mathsf{t}_1)\ \uparrow_c^d (\mathsf{t}_2)$$

6.2.2 Exercise

計算してみる.

6.2.3 Exercise

項のサイズについての帰納法で証明してみる.

6.2.4 Definition

代入が書けるようになった.

Definition

$$[\mathsf{j} \mapsto \mathsf{s}]\mathsf{k} = \begin{cases} s & \text{if } \mathsf{k} = \mathsf{j} \\ k & \text{otherwise} \end{cases}$$

$$[\mathsf{j} \mapsto \mathsf{s}](\lambda.\ \mathsf{t}_1) = \lambda.\ [\mathsf{j}+1 \mapsto \uparrow^1 (\mathsf{s})]\mathsf{t}_1$$

$$[\mathsf{j} \mapsto \mathsf{s}](\mathsf{t}_1\ \mathsf{t}_2) = [\mathsf{j} \mapsto \mathsf{s}]\mathsf{t}_1\ [\mathsf{j} \mapsto \mathsf{s}]\mathsf{t}_2$$

6.2.5 Exercise

4 では代入した後の自由変数の index の調整が必要

6.2.6 Exercise

6.2.3 の結果を使う.

6.2.7 Exercise

定義をそらで書けるかのチェック. 各自やってください.

6.2.8 Exercise

「nameless term を使った代入操作と ordinary term を使った代入操作が一致する」という主張を言うのに必要な定理を書き, 証明する.
6.1.5 で $removenames_\Gamma$ と $restorenames_\Gamma$ は作ったので $restorenames_\Gamma([\mathsf{j} \mapsto \mathsf{s}]removenames(\lambda \mathsf{x}.\mathsf{t}_1)) = [\mathsf{j} \mapsto \mathsf{s}](\lambda \mathsf{x}.\mathsf{t}_1)$ を示せば良い. (残りは楽なので省略)

6.3 Evaluation

代入までできたので, 残るは β 簡約の定義.
もちろん代入を使う.
ポイントとなるのは代入を行うと束縛変数が 1 つ消える.
その空いた分を詰めなくてはいけない.
(6.2.5 の 4 を解くと分かる.)
その再調整操作を入れて
$(\lambda.\ t_{12})\ \mathsf{v}_2 \longrightarrow \uparrow^{-1}([0 \mapsto \uparrow^{1}(\mathsf{v}_2)]\ t_{12})$
こんなルールになる.

6.3.1 Exercise

-1 のシフトがあるけど大丈夫か?
項 $([0 \mapsto \uparrow^1 (v_2)]\ t_{12})$ に 0 は出てこないので -1 シフトしても大丈夫

6.3.2 Exercise

de Bruijn level (reversed de Bruijn repr.) naming context
$\Gamma = x_0, x_2, \ldots, x_n$ 構成中...