プログラミング－言語・基礎・実践 12－2
（1993. 5. 28）

環境をファースト・クラス・オブジェクトとして扱えるλ計算

西崎真也

東京大学理学部情報科学科
sin@is.s.u-tokyo.ac.jp
京都大学理学研究科
sin@kurims.kyoto-u.ac.jp

Lispの方言の一つ，Schemeでは環境をファースト・クラス・オブジェクトとして取り扱うことができる．この機能を付加した単純型つきλ計算λ^{env}ををを提唱し，その簡約規則の停止性を示した．またDe-Bruijnインデックスやα同値関係などにおける，通常のλ計算との違いについて考察した．

λ-calculus with first-class environments

Shin-ya Nishizaki

The University of Tokyo,
Department of Information Science
sin@is.s.u-tokyo.ac.jp
Kyoto Univerisity
Research Institute for Mathematical Sciences
sin@kurims.kyoto-u.ac.jp

In Scheme, a programmer can handle environments as first-class object. We introduce a variant of simply typed λ-calculus with such first-class environments and show finiteness of its reduction. The remarkable difference between this calculus and the usual λ-calculus is also discussed.

1 はじめに

1.1 プログラミング言語における環境

プログラミング言語の基本的な概念の一つに環境(environment)と呼ばれるものがある。これは評価時の変数とそれに束縛されている値との対応である. ファースト・クラス・オブジェクトとは, 数, リスト, 文字列のように関数の引数・返値にすることができるデータのことであり, Schemeでは関数はもちろん, 継続もファースト・クラス・オブジェクトとして扱うことができる. そして, 上で説明した環境もSchemeの処理系の多くは, ファースト・クラス・オブジェクトとしてあつかうことを可能にしている [Sla87] [MIT] [Lau90]. 環境をファースト・クラス・オブジェクトとして取り扱う機能を提供するために導入されたプリミティブのうちで, 主要なものは, `the-environment` と `eval` である. `the-environment` は引数を取らないで, 現在の環境を返す. `eval` は第一引数にリストとして与えられた式を第二引数で与えられた環境のもとで評価して, その結果の値を返す. 簡単な例を次に示す:

```
bash$ elk
> (let ((x 1))
      ( (lambda (env) (eval 'x env))
        (the-environment)))
1
> (let ((x 1))
      ( (lambda (env) (eval 'x env))
        (let ((x 2))
            (the-environment))))
2
>
```

エーベルソンとサスマンの教科書では, ファースト・クラス・オブジェクトとして環境を使った手続きのパッケージングが紹介されている [AS85].

1.2 λ計算における環境

λ計算は, Schemeなどの Lispやその他の関数型言語の計算モデルである. 計算の過程はλ計算ではβ簡約に対応しているのであるが, β簡約には環境は現れてこない. これはλ計算では変数からそれに束縛されている変数を参照するという行為が, 代入操作によりメタな操作として定義されているからである.

$$
\begin{aligned}
& (\lambda x.\lambda y.x)M_1M_2 \\
\xrightarrow[\beta]{} \quad & (\lambda y.x)[x := M_1]M_2 \\
= \quad & (\lambda y.x[x := M_1])M_2 \\
= \quad & (\lambda y.M_1)M_2 \\
\xrightarrow[\beta]{} \quad & \cdots
\end{aligned}
$$

上のようなプログラム言語との隔たりを改善したλ計算として, Curienらはλσ計算を提唱した [nACCL90] [Cur91] [CHL92]. この体系では従来のλ計算では, メタな操作として扱われてきた代入を構文に導入し, 代入操作を計算規則の中に明示的に入れている (Explicit Substitution). こうすることによって, 環境は代入として表現される:

$$
\begin{aligned}
& (\lambda x.\lambda y.x)M_1M_2 \\
\xrightarrow[\beta]{} \quad & ((\lambda y.x)[(M_1/x)\cdot id])M_2 \\
\xrightarrow[\beta]{} \quad & x[(M_2/y)\cdot(M_1/x)\cdot id] \\
\xrightarrow[\sigma]{} \quad & x[(M_1/x)\cdot id] \\
\xrightarrow[\sigma]{} \quad & M_1 \\
\longrightarrow \quad & \cdots
\end{aligned}
$$

λ計算λ^{env}では, λσ計算のこの手法を応用することにより, λ計算においてファースト・クラス・オブジェクトとして環境を扱うことを可能にした. λσ計算では, 代入の構文範疇と項の構文範疇は別々であるのを, 一緒にすることにより, 代入をファースト・クラス・オブジェクトとして扱うことを可能にしたのである.

2 λ計算系 λ^{env}

まず, 型と項の構文を定義し, そのあと, 型推論規則, 簡約規則をあたえる.

2.1 構文

定義 1 [λ^{env} の型と項]
原子型の可算集合*AtomicType*と変数の可算集合*Variable*があらかじめ与えられているとき, 型(type)の集合*Type*と項(term)の集合*Term*は次のように帰納的に定義される:

$$
\begin{aligned}
\text{型} \quad A \quad &::= \quad \alpha \mid A \to B \\
&\qquad\quad \mid \{x_1{:}A_1, \cdots, x_n{:}A_n\} \\
\text{項} \quad M \quad &::= \quad x \mid (MN) \mid \lambda x{:}A.M \\
&\qquad\quad \mid id \mid M \circ N \mid (M/x)\cdot N
\end{aligned}
$$

(但し, $n \geq 0$)
メタ変数としては, $A, B, C, \cdots$を型, $\alpha, \beta, \cdots$を原子型, $L, M, N, \cdots$を項, $x, y, z, \cdots$を変数につかう.

$\lambda x{:}A.M$をλ抽象(lambda-abstraction), (MN)を関数適用(function application), idを恒等環境(identical environment), $M \circ N$を合成(composition), $(M/x)\cdot N$を環境拡張(envionment extension)と呼ぶ.

また, $\{x_1{:}A_1, \cdots, x_n{:}A_n\}$の形をした型を特に環境型(environment type)と呼ぶこととし, $E, E_0, E_1, E_2, \cdots$を環境型のメタ変数につかう.

また, 略記として,

$$\{x_1{:}A_1 \cdots x_m{:}A_m\}\{y_1{:}B_1 \cdots y_n{:}B_n\}$$
$$= \{x_1{:}A_1 \cdots x_m{:}A_m, y_1{:}B_1 \cdots y_n{:}B_n\}$$

とする. □

項において通常の単純型つき λ 計算に比べて拡張されているのは, $id, M \circ N, (x/M) \cdot N$ である. id は, Scheme の the-environment に相当し, id が現われるところの環境を返す. (eval *expression environment*) に相当するのは, $M \circ N$ であり, 項 M を環境 N のもとで評価した値が計算結果である. また, $M \circ N$ において M と N の両方が環境の場合は, 環境 M のなかで束縛されている項に環境 N を適用したものが $M \circ N$ である. id は後で定義される計算規則で, $(-) \circ (-)$ に関して恒等元になっている. $(N/x) \cdot M$ は環境 M を変数 x と項 N の束縛により拡張した環境である.

この構文はもとは, $\lambda\sigma$ 計算に由来する. $\lambda\sigma$ 計算ではさらに, 項 M に対して代入 s を施す操作 $M[s]$ (closure) がある. λ^{env} では代入 (= 環境) と項は同じものとして扱われるので, これは環境合成 $(-) \circ (-)$ と同じものになってしまうのである.

このように定義された項に対し, 次のように型推論規則 (type inference rule) が定義される:

定義 2 [λ^{env} の型推論規則]

型判定式 (type judgement) とは環境型と項と型の間の三項関係 $E \vdash M : A$ であり, 以下のように帰納的に定義される.

$$\frac{(0 \le i \le n) \wedge x_i \notin \{x_0, \cdots, x_{i-1}\}}{\{x_0{:}A_0, \cdots, x_n{:}A_n\} \vdash x_i : A_i}\ Var$$

$$\frac{E \vdash M : A{\to}B \quad E \vdash N : A}{E \vdash MN : B}\ App$$

$$\frac{\{x{:}A\}E \vdash M : B}{E \vdash \lambda x{:}A.M : A{\to}B}\ Lam$$

$$\frac{}{E \vdash id : E}\ Id$$

$$\frac{E \vdash N : E' \quad E' \vdash M : A}{E \vdash M \circ N : A}\ Comp$$

$$\frac{E \vdash M : E' \quad E \vdash N : A}{E \vdash (N/x) \cdot M : \{x{:}A\}E'}\ Extn$$

□

定義 3 [型付き項と型推論木]

$E \vdash M : A$ が成り立っている時, 「項 M は環境型 E のもとで型 A をもつ」といい, このような M は型付き項 (typed term) とよぶ. 以下では特に指定がないかぎり, 項は型がついているものとする. また, $E \vdash M : A$ を導く推論木を型推論木 (type inference tree) と呼び, 型付き項の全体を *TypedTerm* と呼ぶ. □

このように与えられた型推論規則に対して型推論木の一意性が成り立つ:

補題 4 $E \vdash M : A$ かつ $E \vdash M : A'$ ならば $A = A'$ である.

(証明) M の構造に関する帰納法による. □

この補題により, 型つきの項に関して, 項の構造に関する帰納法のかわりに型推論木の構造に関する帰納法が使用できることがわかる.

2.2 簡約規則

この節では, 前節で与えた項に対して, 簡約規則 (reduction rule) を定義することにより, λ^{env} における計算をあたえる.

定義 5 [λ^{env} の簡約]

λ^{env} の簡約規則 (reduction rule) は, 項の間の二項関係 $(-) \underset{wr}{\to} (-)$ であり, 次のように帰納的に定義される. また, $\underset{wr}{\to}$ の部分関係として *Beta1*, *Beta2* を除いたものを $\underset{cr}{\to}$ とする:

$$\frac{}{(L \circ M) \circ N \underset{wr}{\to} L \circ (M \circ N)}\ Ass$$

$$\frac{}{id \circ M \underset{wr}{\to} M}\ IdL$$

$$\frac{}{M \circ id \underset{wr}{\to} M}\ IdR$$

$$\frac{}{((L/x) \cdot M) \circ N \underset{wr}{\to} ((L \circ N)/x) \cdot (M \circ N)}\ DExtn$$

$$\frac{}{x \circ ((M/x) \cdot N) \underset{wr}{\to} M}\ VarRef$$

$$\frac{x \not\equiv y}{y \circ ((M/x) \cdot N) \underset{wr}{\to} y \circ N}\ VarSkip$$

$$\frac{}{(M_1 M_2) \circ N \underset{wr}{\to} (M_1 \circ N)(M_2 \circ N)}\ DApp$$

$$\frac{}{((\lambda x{:}A.M) \circ L)N \underset{wr}{\to} M \circ ((N/x) \cdot L)}\ Beta1$$

$$\frac{}{(\lambda x{:}A.M)N \underset{wr}{\to} M \circ ((N/x) \cdot id)}\ Beta2$$

$$\frac{M \underset{wr}{\to} N}{(ML) \underset{wr}{\to} (NL)}\ AppLeft$$

$$\frac{M \underset{wr}{\to} N}{(LM) \underset{wr}{\to} (LN)}\ AppRight$$

$$\frac{M \underset{wr}{\to} N}{\lambda x{:}A.M \underset{wr}{\to} \lambda x{:}A.N}\ Lam$$

$$\frac{M \underset{wr}{\to} N}{M \circ L \underset{wr}{\to} N \circ L}\ CompLeft$$

$$\frac{M \underset{wr}{\to} N}{L \circ M \underset{wr}{\to} L \circ N}\ CompRight$$

□

通常の単純型つきλ計算の場合と同様にこのように定義された簡約化は，型を保存する：

定理 6 [Subject Reduction Property]　$E \vdash M : A$ かつ $M \underset{wr}{\rightarrow} M'$ ならば，$E \vdash M' : A$ が成り立つ．

（証明）$M \underset{wr}{\rightarrow} M'$ を導く証明図に関する帰納法を用いる．□

この節で定義された簡約規則による簡約例をあげる：

$$
\begin{array}{rc}
 & (\lambda x{:}A.(\lambda y{:}A.x))M_1M_2 \\
\overset{Beta2}{\rightarrow} & ((\lambda y{:}A.x) \circ ((M_1/x) \cdot id))M_2 \\
\overset{Beta1}{\rightarrow} & x \circ ((M_2/y) \cdot (M_1/x) \cdot id) \\
\overset{VarSkip}{\rightarrow} & x \circ ((M_1/x) \cdot id) \\
\overset{VarRef}{\rightarrow} & M_1
\end{array}
$$

もう一つ例を見てみよう：

$$
\begin{array}{rc}
 & (\lambda y{:}A.\lambda z{:}A.z)M \\
\overset{Beta2}{\rightarrow} & (\lambda z{:}A.z) \circ ((M/y) \cdot id)
\end{array}
$$

これは Lisp の関数クロージャに対応していて，クロージャの本体部分が $(\lambda z{:}A.z)$ であり，局所変数に関する情報の部分が，$((M/y) \cdot id)$ である．

この簡約規則に対して次のことが証明されている [西崎 93]：

定理 7 [$\underset{wr}{\rightarrow}$ の合流性]　$M \overset{*}{\underset{wr}{\rightarrow}} N_1$ かつ $M \overset{*}{\underset{wr}{\rightarrow}} N_2$ ならば，$N_1 \overset{*}{\underset{wr}{\rightarrow}} L$ かつ $N_2 \overset{*}{\underset{wr}{\rightarrow}} L$ をみたす L が存在する．□

3　停止性

次に簡約規則wrの停止性を証明する．証明の概略は次の通り：

1. λ^{env} の型つき項から対 (pairing) をもつ単純型つきλ計算 $\lambda^{\rightarrow}_{\times}$ の項への変換を定義する．
2. 簡約規則wrが $\lambda^{\rightarrow}_{\times}$ の上でシミュレートできる．
3. $\lambda^{\rightarrow}_{\times}$ は停止的なので，wrも停止的．

$\lambda^{\rightarrow}_{\times}$ は単純型つきλ計算に対を付加したものである．ただし，ここで付加する対はいわゆる *surjective* pairing ではない．定義はこの論文の最後にあげる．

λ^{env} から対をもつ単純型つきλ計算 $\lambda^{\rightarrow}_{\times}$ への変換を定義する．この変換は型の変換 $[\![-]\!]^\circ$ と項の変換 $[\![-]\!]^*$ とからなる：

定義 8　[型の変換 $[\![-]\!]^\circ$]

λ^{env} の型から $\lambda^{\rightarrow}_{\times}$ の型への変換 $[\![-]\!]^\circ$ は次のように帰納的に定義される：

$$
\begin{array}{l}
[\![\alpha]\!]^\circ = \alpha \\
[\![A{\rightarrow}B]\!]^\circ = [\![A]\!]^\circ{\rightarrow}[\![B]\!]^\circ \\
[\![\{x_1{:}A_1, \cdots, x_n{:}A_n\}]\!]^\circ \\
\qquad = A_1 \times (\cdots \times (A_n \times \mathbf{1}))
\end{array}
$$

□

定義 9　[項の変換 $[\![-]\!]^*$]

λ^{env} の型つきの項から $\lambda^{\rightarrow}_{\times}$ の項への変換 $[\![-]\!]^*$ は次のように帰納的に定義される：

$E \vdash M : A$ なる型つきの項 M に対して，

$$[\![M]\!]^* = \lambda env{:}[\![E]\!]^\circ.[\![E \vdash M : A]\!]^*(env)$$

但し，$[\![-]\!]^*(-)$ は，次のように定義される．□

定義 10　[変換 $[\![-]\!]^*(-)$]

λ^{env} の型推論木と $\lambda^{\rightarrow}_{\times}$ の項から $\lambda^{\rightarrow}_{\times}$ の項への変換 $[\![-]\!]^*(-)$ は次のように帰納的に定義される：

$$
\begin{array}{l}
[\![\{x_0{:}A_0, \cdots, x_n{:}A_n\} \vdash x_i : A_i]\!]^*(Env) \\
= \mathsf{fst}(\mathsf{snd}^i(Env))
\end{array}
$$

$$
\begin{array}{l}
[\![E \vdash MN : B]\!]^*(Env) = \\
([\![E \vdash M : A{\rightarrow}B]\!]^*(Env))([\![E \vdash N : A]\!]^*(Env))
\end{array}
$$

$$
\begin{array}{l}
[\![E \vdash \lambda x{:}A.M : A{\rightarrow}B]\!]^*(Env) \\
= \lambda x'{:}[\![A]\!]^\circ.([\![\{x{:}A\}E \vdash M : B]\!]^*((x', Env))) \\
\text{（但し，} x' \text{は} Env \text{に出現していない変数）}
\end{array}
$$

$$[\![E \vdash id : E]\!]^*(Env) = Env$$

$$
\begin{array}{l}
[\![E \vdash M \circ N : A]\!]^*(Env) \\
= [\![E' \vdash M : A]\!]^*([\![E \vdash N : E']\!]^*(Env))
\end{array}
$$

$$
\begin{array}{l}
[\![E \vdash (M/x) \cdot N : \{x{:}A\}E']\!]^*(Env) = \\
([\![E \vdash M : A]\!]^*(Env), [\![E \vdash N : E']\!]^*(Env))
\end{array}
$$

□

このように変換された項は型つきがつく：

命題 11　$E \vdash M : A$ なる型つき項 M に対し，

$$
\begin{array}{rl}
\{env{:}[\![E]\!]^\circ\} \vdash [\![E \vdash M : A]\!]^*(env) : [\![A]\!]^\circ & (1) \\
\vdash [\![M]\!]^* : [\![E]\!]^\circ{\rightarrow}[\![A]\!]^\circ & (2)
\end{array}
$$

証明：式 (2) は $[\![M]\!]^*$ の定義から，式 (1) により明らか．

式 (1) は型推論木の構造に関する帰納法による．□

次に λ^{env} における簡約列が $[\![-]\!]^*$ により $\lambda^{\rightarrow}_{\times}$ に翻訳されることをしめす．

命題 12　[$[\![-]\!]^*$ による $\underset{wr}{\rightarrow}$ の翻訳]　λ^{env} の型つき項 M, N に対して $M \underset{wr}{\rightarrow} N$ ならば，$[\![M]\!]^* \underset{\beta P}{\rightarrow} [\![N]\!]^*$，もしくは $[\![M]\!]^* = [\![N]\!]^*$ である．とくに，

$$M \underset{VarRef}{\rightarrow} N \Rightarrow [\![M]\!]^*(env) \underset{Fst}{\rightarrow} [\![N]\!]^*(env),$$

$$M \underset{VarSkip}{\rightarrow} N \Rightarrow [\![M]\!]^*(env) \underset{Snd}{\rightarrow} [\![N]\!]^*(env),$$

$$M \underset{Beta1}{\rightarrow} N \Rightarrow [\![M]\!]^*(env) \underset{Beta}{\rightarrow} [\![N]\!]^*(env),$$

$$M \underset{Beta2}{\rightarrow} N \Rightarrow [\![M]\!]^*(env) \underset{Beta}{\rightarrow} [\![N]\!]^*(env).$$

である．（但し，env は変数） □

この命題を使い，λ^{env} の停止性を証明するのだが，この命題だけから停止性は導くことはできない．というのは，無限列

$$M_0 \underset{wr}{\rightarrow} M_1 \underset{wr}{\rightarrow} M_2 \underset{wr}{\rightarrow} \cdots$$

という無限列で，

$$[\![M_0]\!]^* = [\![M_1]\!]^* = [\![M_2]\!]^* = \cdots$$

というものの存在を上の命題からは否定できず，このような無限列は $\lambda_{\times}^{\rightarrow}$ の停止性と矛盾しないからである．しかし，次の命題からこのような無限列は存在しないことがいえる．

命題 13 [$\underset{cr}{\rightarrow}$ の停止性] $\underset{cr}{\rightarrow}$ は停止性をもつ．
（$\underset{cr}{\rightarrow}$ は $\underset{wr}{\rightarrow}$ から$Beta1$,$Beta2$を除いた簡約規則）証明：合成 > 環境拡張 > 関数適用 というオペレータの順序による lexicographic path ordering により，$\underset{cr}{\rightarrow}$ の各規則は真に減少していて，このように定義された lexicographic path ordering は well-founded なので． □

以上，これらの命題により，次の定理が証明できる：

定理 14 [$\underset{wr}{\rightarrow}$ の停止性] $\underset{wr}{\rightarrow}$ は停止性を持つ．
証明：

$$M_0 \underset{wr}{\rightarrow} M_1 \underset{wr}{\rightarrow} \cdots$$

という無限列があったとしよう．$\underset{cr}{\rightarrow}$ の停止性より，$\underset{Beta1}{\rightarrow}$ もしくは $\underset{Beta2}{\rightarrow}$ を含むような $\underset{wr}{\rightarrow}$ は無限個存在する．$\underset{Beta1}{\rightarrow}$ と $\underset{Beta2}{\rightarrow}$ はおのおの，$\underset{Beta}{\rightarrow}$ に変換されるから，$[\![-]\!]^*$ によって変換された列も（前で述べたような場合はあり得ず）真の無限列になっている．これは，$\underset{\beta P}{\rightarrow}$ の停止性に矛盾する． □

4 考察

この章では λ^{env} のいくつかの視点から考察をおこなう．

4.1 変数名について

前章で λ^{env} を $\lambda_{\times}^{\rightarrow}$ によって解釈することにより，停止性を証明した．その時に用いた項の変換 $[\![-]\!]^*$ について，まず考察する．実は次のことがいえる：

命題 15 M を λ^{env} の型つき項とする．$[\![M]\!]^*$ は閉項 (closed term) である．
証明：$[\![E \vdash M : A]\!]^*(Env)$ において自由に出現する変数は全て項 Env においても自由に出現しなければならないことを，型推論木の構造に関する帰納法により証明する． □

例えば，$\{x{:}A\} \vdash x : A$ なる変数 x は，

$$[\![x]\!]^* = \lambda env{:}A \times 1.\mathrm{fst}(env)$$

というように変換される．変数名による参照が，インデックスによる参照にコンパイルされているのである．この部分で，型推論規則において $\vdash$ の左側，（例えば，$\{x_0{:}A_0, \cdots, x_n{:}A_n\}$）が集合ではなくリストであることが本質的である．変数は変数に関する型推論規則 $\{x_0{:}A_0, \cdots, x_n{:}A_n\} \vdash x_i : A_i$ における環境型のエントリの順位 i を使って変換される．これは，ちょうど，DeBruijn インデックスになっている．実際，λ^{env} の DeBruijn スタイルの名前なし λ 計算 λ_{nl}^{env} が定義できる．これは，本論文の最後にあげる．

4.2 α 同値と型なし項

λ^{env} における α 同値は定義されていない．通常の λ 計算では，まず最初に項の間に α 同値関係を定義して，その後，項の α 同値類に対して型推論規則を定義し，簡約規則を定義する．しかし λ^{env} では α 同値関係は通常の λ 計算のように束縛変数の名前のつけかえとして，定義することはできない．例えば，次の例は束縛変数の名前が違うだけであり，通常の λ 計算では α 同値であるが，λ^{env} では簡約結果が変わってくる：

$$x((\lambda x{:}A.\lambda y{:}B.id)MN) \underset{wr}{\rightarrow} \cdots \underset{wr}{\rightarrow} M$$
$$x((\lambda y{:}A.\lambda x{:}B.id)MN) \underset{wr}{\rightarrow} \cdots \underset{wr}{\rightarrow} N$$

とはいうものの，（DeBruijn スタイルの）名前なし項に変換した時に等しくなる二つの項を α 同値であると定義すれば，λ^{env} に α 同値という概念を定義することができる．論文の後ろにあげた名前なし項への変換 $NameElim$ は型推論木に対して定義されているので，型がつかない項に対してはこの方法では定義できない．通常の λ 計算の場合，型がつかない λ 項であっても，DeBruijn スタイルの項に変換できるが，λ^{env} では難しいように思われる．たとえば，次の例を考えてみよう：

$$\lambda env.x \circ env$$

を考えてみよう．xをどのようなDeBruijnインデックスにすればいいのかは，envにどのような環境がわたってくるのかに依存している．

$NameElim(\lambda env.\{x{:}A\}x \circ env)$
$= \lambda\{A\}.0 \circ 0;\ NameElim(\lambda env.\{y{:}B, x{:}A\}x \circ env)$
$= \lambda\{A\}.1 \circ 0.$

もっとも，型なし項を考える目的がチューリングマシンと同等の計算能力を得るため出あるのなら，Yのような再帰演算子を導入すればよい．

5 結論

環境をファースト・クラス・オブジェクトとして扱うことのできる単純型つきλ計算λ^{env}を定義し，その停止性を示した．そして，いくつかの考察をおこなった．

6 将来の課題

6.1 簡約規則について

$(\lambda y.M_1) \circ ((M_2/x) \cdot id)$といった“関数クロージャ”の形をした項に対する簡約は定義してないという点で今回定義した簡約規則は通常のλ計算のベータ簡約に比べて弱いものである．

これに対する簡約を定義するためには，$\lambda\sigma$計算のように shift-オペレータ“$\uparrow$”を導入する必要がある．この拡張で，合流性がどうなるかは今後の課題である．

6.2 型推論アルゴリズムについて

今回の型体系はいわゆる明示的型体系と呼ばれるもので，束縛変数に型を明示的に書いていかなければならず，実際にプログラミング言語としては現実的でない．したがって，MLのような型推論アルゴリズムを与えること，もしくは，型推論アルゴリズムを与えることが可能なような適切な言語のサブクラスを見つけることが，この体系を実際のプログラミング言語に使うという点において不可欠である．

6.3 レコード型との関連

近年，オブジェクト指向プログラミングを型理論へ移入するために，レコード型が盛んに研究されている[Oho92]．名前と型の対の並びという点でレコード型と環境型は似ている．しかし，レコードと環境はそれにほどこす操作という点ではことなる．これら二つの意味論的な関連も興味深いと思われる．

6.4 応用例

環境をファーストクラス・オブジェクトとして扱ったプログラミングの応用例としては，手続きのパッケージング[AS85]などが知られている．デバッギングのメタ言語としても役に立つのではないかと思われるが，いかなる応用例があるかを見つけることも重要な課題である．

参考文献

[AS85] Harold Abelson and Gerald Jay Sussman. *Structure and Interpretation of Computer Programs*. The MIT Press, 1985.

[CHL92] Pierre-Luis Curien, Thérèse Hardin, and Jean-Jacques Lévy. Confluence properties of weak and strong calculi of explicit substitutions. Rapports de Recherche 1617, INRIA, February 1992.

[Cur91] Pierre-Luis Curien. An abstract framework for environment machines. *Theoretical Computer Science*, Vol. 82, pp. 389–402, 1991.

[Lau90] Oliver Laumann. *Reference Manual for the Elk Extension Language Interpreter*, 1990.

[MIT] MIT. *MIT Scheme Reference Manual.*

[nACCL90] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. In *proceedings of the Seventeenth Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, San Francisco, California, January 1990.

[Oho92] Atsushi Ohori. A compilation method for ml-style polymorphic record calculi. In *Conference Record of the Nineteenth Annual ACM Symposium on Principles of Programming Languages*, pp. 154–165, 1992.

[Sla87] Stephen Slade. *The T Programming Language: A Dialect of LISP*. Prentice-Hall, 1987.

[西崎93] 西崎真也. 環境を扱うプリミィティブをもつ型つきλ計算. 関数プログラミングJSSST'93, 1993.

A 対を持つ単純型つきλ計算

対を持つ単純型つきλ計算(simply typed lambda-calculus with pairing)を定義する．

定義 16 [対をもつ単純型つきλ計算]

あらかじめ，変数の可算集合$\mathcal{V}$と原始型の可算集合$\mathcal{AT}$があらかじめ与えられているものとする．型と項は次のように定義される：

$$\begin{array}{lll} \text{型} & A & ::= \alpha \mid A{\to}B \mid A \times B \\ \text{項} & M & ::= x \mid (MN) \mid \lambda x{:}A.M \\ & & \quad \mid (M,N) \mid \mathsf{fst}(M) \mid \mathsf{snd}(N) \end{array}$$

原始型のうち0と書かれるものがあるとする．そして，$0{\to}0$を1と略記することとする．[1]

型割り当て(type assignment)Hとは，変数と型の対の並びである：

$$H ::= \{x_1{:}A_1, \cdots, x_n{:}A_n\} \ (n \geq 0)$$

型判定式(type judgement)は，型割り当てHと項Mと型Aの間の三項関係$H \vdash M : A$で，次のように帰納的に定義される．また，$H \vdash M : A$を導く推論木を型推論木と呼ぶ：

$$\frac{(0 \leq i \leq n) \wedge x_i \notin \{x_0, \cdots, x_{i-1}\}}{\{x_0{:}A_0, \cdots, x_n{:}A_n\} \vdash x_i : A_i}\ Var$$

$$\frac{E \vdash M : A{\to}B \quad E \vdash N : A}{E \vdash MN : B}\ App$$

$$\frac{\{x{:}A\}E \vdash M : B}{E \vdash \lambda x{:}A.M : A{\to}B}\ Lam$$

$$\frac{E \vdash M : A \quad E \vdash N : B}{E \vdash (M,N) : A \times B}\ Pair$$

$$\frac{E \vdash M : A \times B}{E \vdash \mathsf{fst}(M) : A}\ Fst$$

$$\frac{E \vdash M : A \times B}{E \vdash \mathsf{snd}(M) : B}\ Snd$$

□

定義 17 $[\lambda_{\times}^{\to}$ の簡約$]$

$\lambda_{\times}^{\to}$の簡約規則(reduction rule)は，項の間の二項関係$(-) \underset{\beta P}{\to} (-)$であり，次のように帰納的に定義される：

$$\frac{}{(\lambda x{:}A.M)N \underset{\beta P}{\to} [N/x]M}\ Beta$$

$$\frac{}{\mathsf{fst}((M,N)) \underset{\beta P}{\to} M}\ Fst$$

$$\frac{}{\mathsf{snd}((M,N)) \underset{\beta P}{\to} N}\ Snd$$

$$\frac{M \underset{\beta P}{\to} N}{(ML) \underset{\beta P}{\to} (NL)}\ AppLeft$$

$$\frac{M \underset{\beta P}{\to} N}{(LM) \underset{\beta P}{\to} (LN)}\ AppRight$$

$$\frac{M \underset{\beta P}{\to} N}{\lambda x{:}A.M \underset{\beta P}{\to} \lambda x{:}A.N}\ Lam$$

[1] これは Curry-Howard の対応のもとで，おのおの最小論理での $\bot, \top$ に対応するものである

ただし，$[N/x]M$はMにおいてxをNに代入したものとする． □

次のことが成り立つことはよく知られている．

定理 18 $[\lambda_{\times}^{\to}$ の強正規化可能性$]$ $\lambda_{\times}^{\to}$は強正規化可能である． □

B 名前なし計算 λ_{nl}^{env}

定義 19 $[\lambda_{nl}^{env}$ の構文$]$

構文は次の通り：

$$\begin{array}{lll} \text{型} & A & ::= \alpha \mid A{\to}B \quad \mid \{A_1, \cdots, A_n\} \\ \text{項} & M & ::= 0 \mid 1 \mid 2 \mid \cdots \quad \mid (MN) \mid \lambda A.M \\ & & \quad \mid id \mid M \circ N \mid M \cdot N \end{array}$$

□

定義 20 $[\lambda_{nl}^{env}$ の型推論規則$]$

$$\frac{0 \leq i \leq n}{\{A_0, \cdots, A_n\} \vdash i : A_i}\ Var$$

$$\frac{\{A\}E \vdash M : B}{E \vdash \lambda A.M : A{\to}B}\ Lam$$

$$\frac{E \vdash M : A \quad E \vdash M' : E'}{E \vdash M \cdot N : \{A\}E'}\ Extn$$

他はλ^{env}と同様． □

定義 21 $[\lambda_{nl}^{env}$ の簡約規則$]$

$$\frac{}{(L \cdot M) \circ N \underset{wr}{\overset{nl}{\to}} (L \circ N) \cdot (M \circ N)}\ DExtn$$

$$\frac{}{0 \circ (M \cdot N) \underset{wr}{\overset{nl}{\to}} M}\ VarRef$$

$$\frac{}{(i+1) \circ (M \cdot N) \underset{wr}{\overset{nl}{\to}} i \circ N}\ VarSkip$$

$$\frac{}{((\lambda A.M) \circ L)N \underset{wr}{\overset{nl}{\to}} M \circ (N \cdot L)}\ Beta1$$

$$\frac{}{(\lambda A.M)N \underset{wr}{\overset{nl}{\to}} M \circ (N \cdot id)}\ Beta2$$

$$\frac{M \underset{wr}{\overset{nl}{\to}} N}{(\lambda A.M) \underset{wr}{\overset{nl}{\to}} (\lambda A.N)}\ Beta$$

他は前と同じ． □

定義 22 $[$変換$NameElim]$

$NameElim$は名前つき計算λ^{env}の型推論木から名前なし計算λ_{nl}^{env}の項への変換であり，型推論木に関する帰納法により定義する：

$$NameElim(\{x_0{:}A_0, \cdots, x_n{:}A_n\} \vdash x_i : A_i)$$
$$= i\ NameElim(E \vdash M\,N : B)$$

$$= (NameElim(E \vdash M : A{\to}B)$$
$$\quad NameElim(E \vdash N : A))$$

$$NameElim(E \vdash \lambda x{:}A.M : A{\to}B)$$
$$= \lambda A.NameElim(\{x{:}A\}E \vdash M : B)$$

$$NameElim(E \vdash id : E)$$
$$= id$$

$$NameElim(E \vdash M \circ N : A)$$
$$= NameElim(E \vdash M : E')$$
$$\circ\, NameElim(E' \vdash N : A)$$

$$NameElim(E \vdash (M/x) \cdot N : \{x{:}A\}E')$$
$$= NameElim(E \vdash M : A) \cdot NameElim(E \vdash N : E')$$

□