

# 第 11 章 単純な拡張

テキストの解答要約はこんな感じで引用表現にする(引用じゃないけど)

## 演習 11.2.1. [★★★]

チャーチ数を次のように与える(型は省略)

- $c_2 = \lambda s. s \text{ (s unit)}$
- $c_n = \lambda s. \underbrace{s(s \dots (s \text{ unit}) \dots)}_{n \text{ times}} \dots$ 
  - つまりサイズ  $O(n)$

続いて、関数 power を次のように与える

- $\text{power} = \lambda n. \lambda m. m \ n$

最後に項  $t_n$  を次のように与える

- $t_n = \text{power } c_2 \ c_n$

これを評価すると、 $O(2^n)$  ステップの後、チャーチ数  $c_{2^n}$  となる。

解: 漸化式のように定義する。特に再帰部では、1つ前の添字の項をconst化して2回適用する。

定理 11.3.1. [逐次実行は派生形式である]

- 外部言語(表層構文)を内部言語の派生形式(糖衣構文とも)とすることで、証明をシンプルに維持する

## 演習 11.3.2. [★]

型付け規則

$$\frac{\_ : \text{Unit} \in \Gamma}{\Gamma \vdash \_ : \text{Unit}}$$

評価規則

$$(\lambda \_ : \text{Unit}. t) \ v \longrightarrow t$$

[証明]

- 型付け規則 T-Var の  $x$  を  $\_$  に置き換え、型  $T$  を  $\text{Unit}$  に置き換えればよい。
- 評価規則では、E-AppAbs を上記同様置き換え、代入の項を除去すればよい。

解: ワイルドカードの型をUnitに固定しない。また、型付け規則では、T-Absの前提部から  $x$  の型を取り除く。

## 演習 11.4.1. [推奨, ★★]

(1) 新しい派生形式

$$t \text{ as } T \stackrel{\text{def}}{=} (\lambda x : T. x) t$$

を与えると、脱糖衣した型付け規則は以下になる。

$$\frac{\Gamma \vdash t_1 : T}{\Gamma \vdash (\lambda x : T. x) t_1 : T} \quad (\text{T-Ascribe})$$

これは、単純型付きラムダ計算の型付け規則のみで導出できる。

$$\frac{\frac{x : T \in \Gamma, x : T}{\Gamma, x : T \vdash x : T} \text{T-Var} \quad \frac{\Gamma \vdash \lambda x : T. x : T \rightarrow T}{\Gamma \vdash \lambda x : T. x : T \rightarrow T} \text{T-Abs} \quad \Gamma \vdash t_1 : T}{\Gamma \vdash (\lambda x : T. x) t_1 : T} \text{T-App}$$

同様に、脱糖衣した評価規則 E-Ascribe は E-AppAbs によって、E-Ascribe1 は E-App2 によって、それぞれ導出木を構成できる。

(2) E-AscribeEager を純粋型付きラムダ計算の評価規則に脱糖衣できるような派生形式は作れない(根拠なし)。少なくとも、(1) の派生形式は、純粋型付きラムダ計算においてラムダ抽象に項を適用する評価規則がないため、扱うことはできない。

(2) では、次の脱糖衣を採用すれば実現できる。ただし、定理11.3.1の要求は、高水準の評価ステップと低水準の評価ステップが1対多となるように弱めなければならない(この脱糖衣によって1ステップの評価が2ステップかかるようになるため)。

$$t \text{ as } T \stackrel{\text{def}}{=} (\lambda x : \text{Unit} \rightarrow T. x \text{ unit}) (\lambda y : \text{Unit}. t) \quad \text{where } y \text{ is fresh}$$

## 演習 11.5.1. [推奨, ★ ★ ★]

→ 実装 ( `Sec11/SimplyTypedLet.hs` )

## 演習 11.5.2. [★★]

このアイデアは、 $t_2$  中で複数  $x$  が出現する場合に、もともとの `let` の定義を用いる場合に比べて評価回数が倍増するという問題がある。

1. 評価順を変えてしまう
2. well-typed でない項が、評価によって well-typed になるケースがある

## 演習 11.8.1. [★ →]

$$\{l_1 = v_1, \dots, l_i = v_i, \dots, l_n = v_n\}. l_i \rightarrow v_i \quad i \in 1..n$$

## 演習 11.8.2. [★ ★ ★ ★]

### (1) 型付け規則

パターン束縛に関して次の規則を加える。

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma, p : T_1 \vdash t_2 : T_2}{\Gamma \vdash \text{let } p = t_1 \text{ in } t_2 : T_2} \quad (\text{T-Ptn})$$

ここでパターンに型が付くことが前提になっている ( $p : T_1$ ) が、パターンは変数かレコードであるため、それらの型付け規則はT-VarおよびT-Rcdで賄うことができる。

## (2) 進行、保存則の証明概略

- 進行 (項がwell-typedならば値or評価可能): 型付け導出に関する帰納法
  - T-Let: 帰結部は  $\Gamma \vdash \text{let } x = t_1 \text{ in } t_2 : T_2$ 
    - $t_2$  に依らず以下の場合分けて評価が進行する
      - $t_1$  が値  $\rightarrow$  E-LetV
      - $t_1$  が評価可能  $\rightarrow$  E-Let
  - T-Rcd: 帰結部は  $\Gamma \vdash \{l_i = t_i \mid i \in 1..n\} : \{l_i : T_i \mid i \in 1..n\}$ 
    - $\forall i \in 1..n. t_i$  が値  $\rightarrow$  レコード型への標準形補題
      - $v : \{l_i : T_i \mid i \in 1..n\} \longrightarrow v = \{l_i : v_i \mid i \in 1..n\}$  (証明略)
    - otherwise  $\rightarrow \exists i \in 1..n. t_i$  が評価可能  $\rightarrow$  E-Rcd
  - T-Proj: 帰結部は  $\Gamma \vdash t_1.l_j : T_j$ 
    - $t_1 = \{l_i : v_i \mid i \in 1..n\}$  (フィールドがすべて値)  $\rightarrow$  E-ProjRcd
    - otherwise  $\rightarrow$  E-Proj
  - T-Ptn ((1) で定義): 帰結部は  $\Gamma \vdash \text{let } p = t_1 \text{ in } t_2 : T_2$ 
    - T-Let と同様
- 保存 (well-typedな項を評価して得た項はwell-typed)
  - T-Let: 帰結部は  $\Gamma \vdash \text{let } x = t_1 \text{ in } t_2 : T_2$ 
    - $\Gamma \vdash t_1 : T_1, \Gamma, x : T_1 \vdash t_2 : T_2$  を帰結とする部分導出がある
    - 適用可能な評価規則は E-LetV, E-Let
      - E-LetV:  $t_1 = v_1 \quad t' = [x \mapsto v_1]t_2 \rightarrow$  代入補題
      - E-Let:  $t_1 \rightarrow t'_1 \quad t' = \text{let } x = t'_1 \text{ in } t_2 \rightarrow$  T-Let
  - T-Rcd: 帰結部は  $\Gamma \vdash \{l_i = t_i \mid i \in 1..n\} : \{l_i : T_i \mid i \in 1..n\}$ 
    - 各  $i$  に対して  $\Gamma \vdash t_i : T_i$  を帰結とする部分導出がある
    - 適用可能な評価規則は E-Rcd  $\rightarrow$  T-Rcd
  - T-Proj: 帰結部は  $\Gamma \vdash t_1.l_j : T_j$ 
    - $\Gamma \vdash t_1 : \{l_i : T_i \mid i \in 1..n\}$  を帰結とする部分導出がある
    - 適用可能な評価規則は E-ProjRcd, E-Proj
      - E-ProjRcd  $\rightarrow$  各  $i$  に対して  $l_i : T_i$  が部分導出で得られている
      - E-Proj  $\rightarrow$  T-Proj
  - T-Ptn ((1) で定義): 帰結部は  $\Gamma \vdash \text{let } p = t_1 \text{ in } t_2 : T_2$ 
    - T-Let と同様

(1) パターン型付け規則を導入し、 $\Gamma$  に加えて  $\Delta$  を環境として持つ。レコードパターン型付けを行う際、マッチ対象の値が実際に持つフィールドよりも少ないフィールドにしか言及しないように改造すると、レコード射影はletへの糖衣構文になる

(2) パターン型付け関係と実行時のマッチ演算を対応させる補題が必要

- $\Gamma \vdash \sigma \models \Delta$  は、 $\Delta$  の定義域である各パターンについて、代入  $\sigma$  を実行しても  $\Gamma$  上の型はそのままである...みたいな

- 項もパターンも同じ型で、環境  $\Delta$  に含まれるなら、マッチは代入に評価され、 $\Gamma \vdash \sigma \models \Delta$  が成り立つという補題
  - さらに、代入補題化:  $\Gamma, \Delta \vdash t : T$  かつ  $\Gamma \vdash \sigma \models \Delta$  ならば  $\Delta \vdash \sigma t : T$