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. $[\star\star\star]$

チャーチ数を次のように与える（型は省略）

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

続いて、関数 power を次のように与える

- $\mathsf{power} = \lambda n.\, \lambda m.\, m\ n$

最後に項 t_n を次のように与える

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

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

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

評価規則

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

[証明]

- 型付け規則 T-Var の x を _ に置き換え、型 T を 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

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

を与えると、脱糖衣した型付け規則は以下になる。

$$\frac{\Gamma \vdash \mathsf{t}_1 : \mathsf{T}}{\Gamma \vdash (\lambda \mathsf{x} : \mathsf{T}.\, \mathsf{x})\ \mathsf{t}_1 : \mathsf{T}} \quad (\text{T-Ascribe})$$

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。

$$\dfrac{\dfrac{\dfrac{\mathsf{x} : \mathsf{T} \in \Gamma, \mathsf{x} : \mathsf{T}}{\Gamma, \mathsf{x} : \mathsf{T} \vdash \mathsf{x} : \mathsf{T}}\text{T-Var}}{\Gamma \vdash \lambda \mathsf{x} : \mathsf{T}.\, \mathsf{x} : \mathsf{T} \to \mathsf{T}}\text{T-Abs} \quad \Gamma \vdash \mathsf{t}_1 : \mathsf{T}}{\Gamma \vdash (\lambda \mathsf{x} : \mathsf{T}.\, \mathsf{x})\ \mathsf{t}_1 : \mathsf{T}}\text{T-App}$$

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

演習 11.5.1. [推奨, ⋆ ⋆ ⋆]

→実装（`Sec11/SimplyTypedLet.hs`）

演習 11.5.2. [⋆⋆]

このアイディアは、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. [⋆ ↛]

$$\{\mathsf{l}_1 = \mathsf{v}_1, \ldots, \mathsf{l}_i = \mathsf{v}_i, \ldots, \mathsf{l}_n = \mathsf{v}_n\}.\, \mathsf{l}_i \to \mathsf{v}_i \quad i \in 1..n$$

演習 11.8.2. [\star\star\star$]

(1) 型付け規則

パターン束縛に関して次の規則を加える。

$$\frac{\Gamma \vdash \mathsf{t}_1 : \mathsf{T}_1 \qquad \Gamma, \mathsf{p} : \mathsf{T}_1 \vdash \mathsf{t}_2 : \mathsf{T}_2}{\Gamma \vdash \mathsf{let}\ \mathsf{p} = \mathsf{t}_1\ \mathsf{in}\ \mathsf{t}_2 : \mathsf{T}_2} \quad (\text{T-Ptn})$$

ここでパターンに型が付くことが前提になっている ($\mathsf{p} : \mathsf{T}_1$) 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 $\Gamma \vdash \mathsf{let\ x} = \mathsf{t}_1\ \mathsf{in}\ \mathsf{t}_2 : \mathsf{T}_2$
    - t_2 に依らず以下の場合分けで評価が進行する
      - t_1 が値→E-LetV
      - t_1 が評価可能→E-Let
  - T-Rcd: 帰結部は $\Gamma \vdash \{\mathsf{l}_i = \mathsf{t}_i\ ^{i \in 1..n}\} : \{\mathsf{l}_i : \mathsf{T}_i\ ^{i \in 1..n}\}$
    - $\forall i \in 1..n.\ \mathsf{t}_i$ が値→レコード型への標準形補題
      - $\mathsf{v} : \{\mathsf{l}_i : \mathsf{T}_i\ ^{i \in 1..n}\} \longrightarrow \mathsf{v} = \{\mathsf{l}_i : \mathsf{v}_i\ ^{i \in 1..n}\}$ (証明略)
    - otherwise→ $\exists i \in 1..n.\ \mathsf{t}_i$ が評価可能→E-Rcd
  - T-Proj: 帰結部は $\Gamma \vdash \mathsf{t}_1.\mathsf{l}_j : \mathsf{T}_j$
    - $\mathsf{t}_1 = \{\mathsf{l}_i : \mathsf{v}_i\ ^{i \in 1..n}\}$（フィールドがすべて値）→E-ProjRcd
    - otherwise→E-Proj
  - T-Ptn ((1) で定義): 帰結部は $\Gamma \vdash \mathsf{let\ p} = \mathsf{t}_1\ \mathsf{in}\ \mathsf{t}_2 : \mathsf{T}_2$
    - 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 $\Gamma \vdash \mathsf{let\ x} = \mathsf{t}_1\ \mathsf{in}\ \mathsf{t}_2 : \mathsf{T}_2$
    - $\Gamma \vdash \mathsf{t}_1 : \mathsf{T}_1, \Gamma, \mathsf{x} : \mathsf{T}_1 \vdash \mathsf{t}_2 : \mathsf{T}_2$ 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: $\sf t_1 = v_1 \quad t' = [x\mapsto v_1]t_2$ → 代入補題
      - E-Let: $\sf t_1 \rarr t_1' \quad t' = let\ x=t_1'\ in\ t_2$ → T-Let
  - T-Rcd: 帰結部は $\Gamma \vdash \{\mathsf{l}_i = \mathsf{t}_i\ ^{i \in 1..n}\} : \{\mathsf{l}_i : \mathsf{T}_i\ ^{i \in 1..n}\}$
    - 各 i に対して $\Gamma \vdash \mathsf{t}_i : \mathsf{T}_i$ 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 → T-Rcd
  - T-Proj: 帰結部は $\Gamma \vdash \mathsf{t}_1.\mathsf{l}_j : \mathsf{T}_j$
    - $\Gamma \vdash \mathsf{t}_1 : \{\mathsf{l}_i : \mathsf{T}_i\ ^{i \in 1..n}\}$ 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 → 各 i に対して $\mathsf{l}_i : \mathsf{T}_i$ が部分導出で得られている
      - E-Proj → T-Proj
  - T-Ptn ((1) で定義): 帰結部は $\Gamma \vdash \mathsf{let\ p} = \mathsf{t}_1\ \mathsf{in}\ \mathsf{t}_2 : \mathsf{T}_2$
    - T-Let と同様