型輪講資料第9章
Simply Typed Lambda-Calculus

36118
立沢秀晃*

平成15年4月15日

1 関数型

8章で導入された、算術計算のための型 (Bool, Nat) の中で、簡略のためにこの章では Bool のみを扱う。関数型の導入に関して、→ 型を用いつが、単にこれを導入したのみだと関数であるという事実しか型情報としてえられない (以下の二つは区別できない) ので、引数と返り値のそれぞれの型を関数型の中に含めることにする。[1]
例：

$$\lambda x.\texttt{true} \quad : \quad \text{返り値の型は Bool}$$
$$\lambda x.\lambda y.y \quad : \quad \text{返り値の型は} \rightarrow$$

$$\begin{aligned} T(types) \quad ::= & \\ & \texttt{Bool}(type\ of\ booleans) \\ & T \rightarrow T(type\ of\ functions) \end{aligned}$$

2 型関係

関数型が定義されて、

- 関数適用が起こるときに何が起こるか
- 期待されている引数の型をどのように知るか

という問題がおこる。これを解決するためには二つの方針がある。

一つは型推論をすること。つまりλ抽象された式の本体を見て引数の値がどのように使われているかを判断するということである。このように型付けされる言語は「暗黙に型付けされる (implicityly typed)」言語という。

*e-mail: hideaki@yl.is.s.u-tokyo.ac.jp
[1] → は右結合。つまり、$T_1 \rightarrow T_2 \rightarrow T_3 = T_1 \rightarrow (T_2 \rightarrow T_3)$

もう一つは単にλ抽象の際に期待されている引数の型を付加する方法である。この方法では例えば単に「$\lambda x.t$」と書く代わりに「$\lambda x : \texttt{Bool}\ t$」等と書く。これによって term に明示的に型を付加することによって型付けされる言語は「明示的に型付けされる (explicitly typed)」という。

これを導入することによって、関数型を含めた型付け規則は次のように定義できる。ただし、以下ではΓは型環境を意味するものとし、また型環境の拡張として$\Gamma, x : T$と書く場合にはΓによって束縛される変数の中にはxは存在しないこととする。[2]

型付け規則

$$\Gamma \vdash \texttt{true} : \texttt{Bool} \text{(T-TRUE)}$$

$$\Gamma \vdash \texttt{false} : \texttt{Bool} \text{(T-FALSE)}$$

$$\frac{\Gamma \vdash t_1 : \texttt{Bool} \quad \Gamma \vdash t_2 : T \quad \Gamma \vdash t_3 : T}{\Gamma \vdash \texttt{if} t_1 \texttt{ then} t_2 \texttt{ else} t_3 : T} \text{ (T-IF)}$$

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \text{ (T-VAR)}$$

$$\frac{\Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x : T_1.t_2 : T_1 \to T_2} \text{ (T-ABS)}$$

$$\frac{\Gamma \vdash t_1 : T_{11} \to T_{12} \quad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1 t_2 : T_{12}} \text{ (T-APP)}$$

評価規則

$$\texttt{if true then } t_2 \texttt{ else } t_3 \to t_2 \text{(E-IFTRUE)}$$

$$\texttt{if false then } t_2 \texttt{ else } t_3 \to t_3 \text{(E-IFFALSE)}$$

$$\frac{t_1 \to t_1'}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \to \texttt{if } t_1' \texttt{ then } t_2 \texttt{ else } t_3} \text{ (E-IF)}$$

$$\frac{t_1 \to t_1'}{t_1 t_2 \to t_1' t_2} \text{ (E-APP1)}$$

$$\frac{t_2 \to t_2'}{v_1 t_2 \to v_1 t_2'} \text{ (E-APP2)}$$

$$(\lambda x : T_{11}.t_{12}) v_2 \to [x \mapsto v_2] t_{12} \text{(E-APPABS)}$$

練習 1 (ex9.2.1)
整数型や *boolean* 型等の基本型の存在しない、*pure simply typed lambda-calculus* ではよく型付けさ

[2] このことと Syntax から型環境に束縛される変数はすでに束縛されている全ての変数と異なることがいえる。

れた *(well-typed)* 項が存在しないのはなぜか。
解*:*
ある型 T に型付けされた項があったとする。次の表より、$T = T_1 \to T_2$ の形であり、また、それら T_1 や T_2 も関数型であるために $T_a \to T_b \to \ldots T_z \to \ldots$ と無限に続く。すなわち固定した有限の型自体が存在しない。

pure simply typed lambda-calculus
Syntax

$t ::=$	*(terms)*				
x	*(variable)*	$v ::=$	*(values)*	$T ::=$	*(types)*
$\lambda x : T.t$	*(abstraction)*	$\lambda x : T.t$	*(abstraction value)*	$T \to T$	*(type of function)*
tt	*(application)*				
		$\Gamma ::=$	*(contexts)*		
		ϕ	*(empty context)*		
		$\Gamma, x : T$	*(term variable binding)*		

ただしこの章では初めに書いた通り、この他に `Bool` 型の値として `true` と `false`、そして if 節を扱う。

練習 2 (ex9.2.2)
次の項の型導出木を構成せよ。

- $f : \texttt{Bool} \to \texttt{Bool} \vdash f(\texttt{if false then true else false}) : \texttt{Bool}$
- $f : \texttt{Bool} \to \texttt{Bool} \vdash \lambda x : \texttt{Bool}.f(\texttt{if } x \texttt{ then false else } x) : \texttt{Bool} \to \texttt{Bool}$

解*:*
1. $\Gamma = f : \texttt{Bool} \to \texttt{Bool}$ とする。

$$\dfrac{\dfrac{f : \texttt{Bool} \to \texttt{Bool} \in \Gamma}{\Gamma \vdash f : \texttt{Bool} \to \texttt{Bool}}\ T\text{-}VAR \quad \dfrac{\dfrac{\vdash \texttt{false} : \texttt{Bool}}{}\ T\text{-}FALSE \quad \dfrac{\vdash \texttt{true} : \texttt{Bool}}{}\ T\text{-}TRUE}{\Gamma \vdash \texttt{if false then true else false} : \texttt{Bool}}\ T\text{-}IF}{\Gamma \vdash f(\texttt{if false then true else false}) : \texttt{Bool}}\ T\text{-}APP$$

2. $\Gamma = f : \texttt{Bool} \to \texttt{Bool}$、$\Gamma' = \Gamma, x : \texttt{Bool}$ とする。

$$\dfrac{\dfrac{\dfrac{f : \texttt{Bool} \to \texttt{Bool} \in \Gamma'}{\Gamma' \vdash f : \texttt{Bool} \to \texttt{Bool}}\ T\text{-}VAR \quad \dfrac{\dfrac{x : \texttt{Bool} \in \Gamma'}{\Gamma' \vdash x : \texttt{Bool}}\ T\text{-}VAR \quad \dfrac{\vdash \texttt{false} : \texttt{Bool}}{}\ T\text{-}FALSE}{\Gamma' \vdash \texttt{if } x \texttt{ then false else } x : \texttt{Bool}}\ T\text{-}IF}{\Gamma' \vdash f(\texttt{if } x \texttt{ then false else } x) : \texttt{Bool}}\ T\text{-}APP}{\Gamma \vdash \lambda x : \texttt{Bool}.f(\texttt{if } x \texttt{ then false else } x) : \texttt{Bool} \to \texttt{Bool}}\ T\text{-}ABS$$

練習 3 (ex9.2.3)
項 fxy が型 `Bool` を持つような型環境 Γ を見つけよ。またそのような型環境の中でもっとも単純なものは何か。
解*:*

単純なもので項 fxy を型付けする環境としては f, x, y それぞれに対する束縛のみがある環境であるのは明らか。また、項の形から f は返り値の型が $\mathtt{Bool}$ の関数型である。これを $\alpha \to \mathtt{Bool}$ とおく。ここで、f は x, y の 2 引数を取っているので、x, y の型をそれぞれ β, γ とすると $\alpha = \beta \to \gamma$ が成り立つ。今考えている型でもっとも単純な型は $\mathtt{Bool}$ なので、$\beta = \gamma = \mathtt{Bool}$ とおくと、求めるもっとも単純な型環境 Γ は以下のようになる。

$$\Gamma = f : \mathtt{Bool} \to \mathtt{Bool} \to \mathtt{Bool}, x : \mathtt{Bool}, y : \mathtt{Bool}$$

3 型付けの性質

補題 1 (型関係の逆)

1. $\Gamma \vdash x : R$ ならば、$x : R \in \Gamma$
2. $\Gamma \vdash \lambda x : T_1.t_2 : R$ ならば、$\Gamma, x : T_1 \vdash t_2 : R_2$ という R_2 に対して、$R = T_1 \to R_2$
3. $\Gamma \vdash t_1 t_2 : R$ ならば、$\Gamma \vdash t_2 : T_{11}$ という T_{11} に対して、$\Gamma \vdash t_1 : T_{11} \to R$
4. $\Gamma \vdash \mathtt{true} : R$ ならば、$R = \mathtt{Bool}$
5. $\Gamma \vdash \mathtt{false} : R$ ならば、$R = \mathtt{Bool}$
6. $\Gamma \vdash \mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 : R$ ならば、$\Gamma \vdash t_1 : \mathtt{Bool}$ かつ、$\Gamma \vdash t_2, t_3 : R$

証明*:*
型付け規則の定義からこの定理は得られる。

練習 4 (ex9.3.3)
$\Gamma \vdash xx : T$ となるような型環境 Γ と型 T は存在するか。存在するならばそれらを与え、その型導出木をもとめ、存在しないならばそれを証明せよ。
解*:*
項 xx が型 T を持つとすると、型関係の逆から x*(*左の x*)* は型 $T_1 \to T$ を持ち、かつ x*(*右の x*)* は型 T_1 を持つことになる。それら x は同じ項であり同じ型を持つので $T_1 \to T = T_1$ となり、矛盾する。よって題意を満たすような型環境と型は存在しない。

定理 1 (型の単一性)
型環境 Γ が与えられたとき、Γ のもとで、現れる自由変数は Γ に全て束縛されている項 t は多くとも一つの型しか持たない *(*つまり型付け可能な項は唯一の型を持つ*)*。また、その型導出木も唯一に決まる。[3]
証明*:*
$\Gamma \vdash t : T_1$ かつ $\Gamma \vdash t : T_2$ としたとき $T_1 = T_2$ を示す。導出木に関する帰納法を用いる。

1. $t = \mathtt{true}$ のとき、明らか。
2. $t = \mathtt{false}$ のとき、明らか。
3. $t = \mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3$ のとき、型関係の逆から、$\Gamma \vdash t_1 : \mathtt{Bool}$ かつ $\Gamma \vdash t_2 : T_1$ かつ $\Gamma \vdash t_3 : T_1$。また、$\Gamma \vdash t_1 : \mathtt{Bool}$ かつ $\Gamma \vdash t_2 : T_2$ かつ $\Gamma \vdash t_3 : T_2$。帰納法の仮定より項 t_2, t_3 は多くとも一つの型しか持たないので、$T_1 = T_2$。

[3] この定理により明示的に型付けされる言語の型の注釈が λ 抽象のときのみで十分であることが分かる。

4. $t = x$ のとき、型関係の逆から、$x : T_1 \in \Gamma$。また、$x : T_2 \in \Gamma$。型環境 Γ の *Syntax* より、ある変数については一つの型しか対応していないので、$T_1 = T_2$。

5. $t = \lambda x : T'.t2$ のとき、$T_1 = T' \to T_1'$、$T_2 = T' \to T_2'$ とすると、型関係の逆から、$\Gamma, x : T' \vdash t2 : T_1'$。また、$\Gamma, x : T' \vdash t2 : T_2'$。帰納法の仮定より、項 t_2 は多くとも一つの型しか持たないので、$T_1' = T_2'$。
   $\therefore T_1 = T_2$。

6. $t = t_1 t_2$ のとき、型関係の逆から、$\Gamma \vdash t_1 : T_1' \to T_1$ かつ、$\Gamma \vdash t_2 : T_1'$。また、$\Gamma \vdash t_1 : T_2' \to T_2$ かつ、$\Gamma \vdash t_2 : T_2'$。帰納法の仮定より項 t_1, t_2 は多くとも一つの型しか持たないので、$T_1' \to T_1 = T_2' \to T_2$ かつ、$T_1' = T_2'$。
   $\therefore T_1 = T_2$

補題 2 (標準形)

1. v が型 `Bool` の値を持つとき、それは `true` または `false` である。

2. v が型 $T_1 \to T_2$ の値を持つとき、$v = \lambda x : T_1.t_2$ である。

証明*:*
値の定義から明らか。

定理 2 (進行)
t が閉じた、型付けされる項だとするとき *(*すなわち、ある型 T に対して $\vdash t : T$ が成り立つとき*)*、t は値であるか、他の項 t' に対して $t \to t'$ となる簡約が存在する。
証明
$\vdash t : T$ とする。
項の構造に関する帰納法を用いる。

1. $t =$ `true` のとき、t は値である。

2. $t =$ `false` のとき、t は値である。

3. $t =$ `if` t_1 `then` t_2 `else` t_3 のとき、型関係の逆から、$\vdash t_1 :$ `Bool` かつ $\vdash t_2 : T$ かつ $\vdash t_3 : T$。帰納法の仮定より項 t_1 は値であるか他の項に対する簡約が存在する。

   - t_1 が値のとき、t_1 は `Bool` 型を持つので、標準形の補題から `true` または `false` である。それぞれ *E-IFTRUE* と *E-IFFALSE* を適用することにより簡約できる。
   - そうでないとき、帰納法の仮定より $t_1 \to t_1'$ と簡約できるので、$t \to$ `if` t_1' `then` t_2 `else` t_3 と簡約できる。

4. $t = x$ のときは t が閉じた項であるという定理の仮定より起こり得ない。

5. $t = \lambda x : T'.t2$ のとき、t は値 *(abstraction value)* である。

6. $t = t_1 t_2$ のとき、型関係の逆から、$\vdash t_1 : T_2 \to T$ かつ、$\vdash t_2 : T_2$ である。帰納法の仮定より、t_1, t_2 は値であるか、他の項に関する簡約が存在する。

   - t_1 に他の項に関する簡約が存在するとき、*E-APP1* を適用できる。
   - t_1 が値で t_2 に他の項に関する簡約が存在するとき、*E-APP2* が適用できる。

- t_1, t_2 の両方が値のとき、t_1 は型 $T_2 \to T$ を持つので、標準形の補題より $\lambda x : T_2.t_1'$ の形であるので、*E-APPABS* が適用できる。

補題 3 (置換)
$\Gamma \vdash t : T$ かつ、Δ が Γ の置換であるとき、$\Delta \vdash t : T$ である。また、導出木の深さもこの二つは同じである。

補題 4 (弱化)
$\Gamma \vdash t : T$ かつ、$x \notin dom(\Gamma)$ のとき、$\Gamma, x : S \vdash t : T$ である。また、導出木の深さもこの二つは同じである。

補題 5 (代入における型保存)
$\Gamma, x : S \vdash t : T$ かつ、$\Gamma \vdash s : S$ のとき、$\Gamma \vdash [x \mapsto s]t : T$。
証明:
$\Gamma, x : S \vdash t : T$ の導出に関する帰納法を用いる。

1. $t = \texttt{true}$ のとき、$T = \texttt{Bool}$ なので、$[x \mapsto s]t = true$。よって、弱化の補題より、$\Gamma \vdash [x \mapsto s]t : T$。
2. $t = \texttt{false}$ のとき、上と同様である。
3. $t = \texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3$ のとき、型関係の逆から、
   - $\Gamma, x : S \vdash t_1 : \texttt{Bool}$
   - $\Gamma, x : S \vdash t_2 : T$
   - $\Gamma, x : S \vdash t_3 : T$

   帰納法の仮定より、
   - $\Gamma \vdash [x \mapsto s]t_1 : \texttt{Bool}$
   - $\Gamma \vdash [x \mapsto s]t_2 : T$
   - $\Gamma \vdash [x \mapsto s]t_3 : T$

   $\therefore$*T-IF* より、$\Gamma \vdash [x \mapsto s]t : T$。
4. $t = z$ のとき、*(z は変数)*
   - $z = x$ のとき、$[x \mapsto s]z = s$ である。また $T = S$ であるので、求められている結果は $\Gamma \vdash [x \mapsto s]z : S$。よって得られた。
   - z が x 以外の変数のとき、$[x \mapsto s]z = z$ である。$\Gamma \vdash z : T$ なので、求められている結果が得られた。
5. $t = \lambda y : T_2.t_1$ のとき、$T = T_2 \to T_1$ かつ $\Gamma, x : S, y : T_2 \vdash t_1 : T_1$ とおく。このとき、型環境の定義から $x \neq y$ かつ、$y \notin FV(s)$。置換の補題から、$\Gamma, y : T_2, x : S \vdash t_1 : T_1$。また、弱化の補題から、$\Gamma, y : T_2 \vdash s : S$。よって、帰納法の仮定から、$\Gamma, y : T_2 \vdash [x \mapsto s]t_1 : T_1$。これに *T-ABS* を適用して、$\Gamma \vdash \lambda y : T_2.[x \mapsto s]t_1 : T_2 \to T_1$ が得られる。
6. $t = t_1 t_2$ のとき、型関係の逆から、$\Gamma, x : S \vdash t_1 : T_2 \to T$ かつ、$\Gamma, x : S \vdash t_2 : T_2$ である。帰納法の仮定より、$\Gamma \vdash [x \mapsto s]t_1 : T_2 \to T$ かつ、$\Gamma \vdash [x \mapsto s]t_2 : T_2$。よって *T-APP* より、$\Gamma \vdash [x \mapsto s]t_1[x \mapsto s]t_2 : T$ すなわち $\Gamma \vdash [x \mapsto s](t_1 t_2) : T$。

定理 3 (評価における型保存)
$\Gamma \vdash t : T$ かつ、$t \to t'$ のとき、$\Gamma \vdash t' : T$。
証明:
$\Gamma \vdash t : T$ の導出に関する帰納法を用いる。

1. $t = \texttt{true}$ のとき、t と異なる t' に対して $t \to t'$ となる評価は存在しない。よって、定理の前提が偽になるため定理を満たしている。

2. $t = \texttt{false}$ のとき、上と同様である。

3. $t = \texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3$ のとき、型関係の逆から、$\Gamma \vdash t_1 : \texttt{Bool}$ かつ、$\Gamma \vdash t_2 : T$ かつ、$\Gamma \vdash t_3 : T$。このとき、評価の適用によって次のように場合わけする。

   - *E-IFTRUE* のとき、$t_1 = \texttt{true}$ かつ、$t' = t_2$。t_2 は型 T を持つので題意を満たす。
   - *E-IFFALSE* のとき、上と同様。
   - *E-IF* のとき、$t_1 \to t_1'$ かつ、$t' = \texttt{if}\ t_1'\ \texttt{then}\ t_2\ \texttt{else}\ t_3$。また、$t_1$ は型 Bool を持つので、帰納法の仮定から t_1' は型 Bool を持つ。さらに、$\Gamma \vdash t_2 : T$ かつ、$\Gamma \vdash t_3 : T$ なので、*T-IF* より $\Gamma \vdash \texttt{if}\ t_1'\ \texttt{then}\ t_2\ \texttt{else}\ t_3 : T$。

4. $t = x$ のとき (x は変数)、t と異なる t' に対して $t \to t'$ となる評価は存在しない。よって、定理の前提が偽になるため定理を満たしている。

5. $t = \lambda x : T_1.t_1$ のとき、t は値なので t と異なる t' に対して $t \to t'$ となる評価は存在しない。よって、定理の前提が偽になるため定理を満たしている。

6. $t = t_1 t_2$ のとき、型関係の逆から、$\Gamma \vdash t_1 : T_1 \to T$ かつ、$\Gamma \vdash t_2 : T_1$。このとき、評価の適用によって次のように場合わけする。

   - *E-APP1* のとき、$t_1 \to t_1'$ かつ、$t' = t_1' t_2$。帰納法の仮定より、型環境 Γ のもとで t_1' は t_1 と同じ型をもつ、すなわち、$\Gamma \vdash t_1' : T_1 \to T$。よって *T-APP* より、$\Gamma \vdash t_1' t_2 : T$。
   - *E-APP2* のとき、上と同様。
   - *E-APPABS* のとき、$t_1 = \lambda x : T_1.t_{12}$ かつ、$t' = [x \mapsto t_2]t_{12}$ (ただし $x \notin \Gamma$)。また、$\Gamma \vdash \lambda x : T_1.t_{12} : T_1 \to T$ なので、型関係の逆より、$\Gamma, x : T_1 \vdash t_{12} : T$ さらに、$\Gamma \vdash t_2 : T_1$ なので、代入における型保存の補題より、$\Gamma \vdash [x \mapsto t_2]t_{12} : T$。すなわち、$\Gamma \vdash t' : T$。

練習 5
Subject expansion の性質、すなわち $t \to t'$ かつ、$\Gamma \vdash t' : T$ ならば $\Gamma \vdash t : T$ は関数部分については (t が *if* 節を含まない場合) 成り立つか。[4]
解:
$t = (\lambda x : T_{11}.t_{12})t_2$ かつ、$t' = [x \mapsto t_2]t_{12}$ のとき、x と t_{12} の型が異なる場合、t は型付けできない。よって、この命題は成り立たない。
具体例:
$t = (\lambda x : \texttt{Bool} \to \texttt{Bool}.x)\texttt{true}$ かつ、$t' = [x \mapsto \texttt{true}]x$ のとき、t は型付けできないが t' は型 Bool を持つ。

[4]参考として、if 節のある場合は、$t = \texttt{if true then}\ t'\ \texttt{else true}$ かつ、$t' = \lambda x : \texttt{Bool}.\texttt{true}$ などが反例として挙げられる。

4 カリーハワード対応

論理	プログラム言語
命題	型
命題 P ならば Q	型 $P \to Q$
命題 P かつ Q	型 $P \times Q$(ペアの型)
命題 P の証明	型 P の項
命題 P が証明可能	型 P の項が存在する
カット除法による証明の単純化の論理的手続き	λ 項の簡約による計算

このカリーハワード対応[5]はある特定の型システムと論理の対応だけでなく、様々な型システムと論理の対応に拡張することができる。実際この対応は新しい成果をそれぞれの分野に翻訳するために使われてきた。例としては線型論理によって線型型システムが生まれたり、様相論理によって部分評価と実行時コード生成のフレームワークを作る際のヒントになった。

5 型の削除

プログラムを実行 (評価) するときには型情報はチェックしていないので、評価する前に型付きの項を型無しの項に変換する。この変換を pure simply typed lambda-calculus 上において次の *erase* によって定義する。

定義 1
erase の定義

$$\begin{aligned} erase(x) &= x \\ erase(\lambda x : T_1.t_2) &= \lambda x.erase(t_2) \\ erase(t_1 t_2) &= erase(t_1) erase(t_2) \end{aligned}$$

この変換によって評価結果が変わらない、つまり、評価してから型をなくすのと、型をなくしてから評価するのが等価であることは次の定理から得られる。

定理 4

1. 型付けされた項 t に対して、$t \to t'$ ならば、$erase(t) \to erase(t')$
2. 型付けされた項 t に対して、$erase(t) \to m'$ ならば、$t \to t'$ かつ、$erase(t') = m'$ となるような型付けされる項 t' がひとつ存在する。

証明:
評価導出における帰納法を用いる (省略)。

ここで考えているコンパイルが非常に単純なためにこの定理は明らかであると考えられるが、より複雑な言語ではプログラマが書く高レベルの言語と処理をする低レベルの言語の等価性を意味する。また、型無し λ 計算の項に対する型付け可能性によって、次のように「型付け可能」を定義する。

定義 2
$erase(t) = m$ かつ、$\Gamma \vdash t : T$ であるようなある単純型付けされる項 t、型 T、型環境 Γ が存在するとき、型無し λ 計算の項 m が $\lambda_{\to}$ において型付け可能という。

[5]simply typed lambda-calculus においての対応に関してはカリー対応という

6　Curry スタイルと Church スタイル

型と意味論の定義の仕方には二つの形式がある。Curry-style ではまず項を定義し、次にそれらがどのように振る舞うかという意味論を定義して、最後に望まない振る舞いをする項をふるい落とすような型システムを与える。これに対して、Church-style ではまず項を定義し、次によく型付けされた (well-typed) 項を定義し、最後にそれらの項に対して意味論を与える。つまり、Curry-style では型付けより意味論の方が優位であり、逆に、Church-style では意味論より型付けの方が優位である。つまり、Church-style では型付けされない項の振る舞いがどうなるかということは意味の無いことである。厳密にはこの本では Church-style のシステムで実際に評価しているのは項ではなく型導出である。歴史的には暗黙に型付けされる λ 計算ではよく Curry-style が使われ、明示的に型付けされる λ 計算ではよく Church-style が使われたため、Curry-style という言葉が暗黙に型付けされる λ 計算の Syntax という意味で使われ、Church-style という言葉が明示的にに型付けされる λ 計算の Syntax という意味で使われることが時々ある。