ソフトウェア基礎論配布資料 (5)
単純型付 λ 計算

五十嵐 淳
京都大学 大学院情報学研究科知能情報学専攻
e-mail: igarashi@kuis.kyoto-u.ac.jp

平成 16 年 11 月 9 日

1 動機づけ

- 項 `1 + true` はエラーの発生を示していると考えられる.
- プログラム (項) を実行 (簡約) する前に，こういったデータ型のミスマッチに起因するエラーの発生を検知したい．安全性の保証
- 検知する問題自体は決定不能 $\Longrightarrow$ 保守的な方法: エラーが発生するプログラムは全て検出するが，エラーが発生しないプログラムも危険と判断されうる.

2 型システムとは

プログラム (項) を，その構文的情報のみを使って，その予想される実行結果 (簡約の正規形) の種類で分類するための仕組み

2.1 型

分類情報を**型**(*type*) と呼ぶ．型としてどんなものを用意するかで，受理できるプログラムの集合の大きさ，保証できる安全性 (すなわち，排除できるエラーの種類) が大きく異なる．ここでは

- 自然数に評価される項のための型 `nat`
- 真偽値に評価される項のための型 `bool`
- 関数値 (λ 抽象) に評価される項のための型

を考える．さらに，関数の型は定義域 (の型) と値域 (の型) によって細かく分け，$T_1 \to T_2$ で「T_1 型の項を引数として T_2 型の項を返す関数」の型を表わす．例えば `nat` $\to$ `nat` は自然数上の関数の型であり，(`nat` $\to$ `nat`) $\to$ `bool` は自然数上の関数を引数として真偽値を返す関数の型である．`nat`, `bool` のように 型の原子とも言える型を**基底型**(*base type*) と呼ぶことがある． また，このように基底型と関数型だけからなる型 (の構造) を**単純型**(*simple type*) と呼ぶ.

2.2 型付け関係 (型判断)，型環境と型付け規則

項 t が何らかの型 T の分類にあてはまる時，$t:T$ と書き，項 t の型が T である，とか，単に項が**型付け可能**(*typable*) という．例えば $\texttt{true}:\texttt{bool}$ や $\texttt{0}:\texttt{nat}$ が成立すると考えられる．また，$\texttt{S}(t)$ のような部分式を持つ項に関しては,「t が nat 型ならば $\texttt{S}(t)$ は nat 型である」 という規則が考えられる．このような項に型を与えるための規則を**型付け規則**(*typing rule*) と呼ぶ．通常，型付け規則は項の構成要素毎に用意される．

関数適用項 $t_1\ t_2$ の型付け規則は,

$$t_1 : T_1 \to T_2 \text{ かつ } t_2 : T_1 \to \text{ならば } t_1\ t_2 : T_2$$

とすることができる．さて，関数を構成するための λ 抽象項には期待通り関数型が与えられる．定義域・値域の型はどのように決まるのだろう．ここでは，定義域の型は，プログラマが宣言するものと考え，項の文法に型宣言を加え，λ 抽象項は $\lambda x{:}T.t$ と記述することにする．値域の型は，関数本体 t の型となる．ただし，関数本体中ではパラメータである変数項 x が参照されているかもしれないので，これらを総合して λ 抽象項の型付け規則は

$$t : T_2 \text{ が } x : T_1 \textbf{ という仮定の下で成立するとき}, \ \lambda x{:}T_1.t : T_1 \to T_2$$

となる．上のような変数項の持つ型に関する仮定を扱うために，より一般的には，**型環境**(*type environment, typing context*) と呼ばれる概念を導入する．型環境 (メタ変数 Γ を使用する) は $x_1{:}T_1, \ldots, x_n{:}T_n$ という形の列で記述され，x_i が T_i 型を持つ，という仮定を示す．そして，項が型を持つかの議論は型環境という仮定のもとで行なわれるので，これを明示して $\Gamma \vdash t{:}T$ (Γ の下で，t は型 T を持つ) と記述する．この $\Gamma \vdash t : T$ の形を**型判断**(*type judgment*) もしくは**型付け判断**(*typing judgment*) ということがある．また，この $\cdot \vdash \cdot : \cdot$ を型環境・項・型の三項関係と考えて**型付け関係**(*typing relation*) と呼ぶこともある．このように考えると，型付け規則は型付け関係という帰納的に定義される集合を定義するための規則として記述することができる．例えば，上述の非形式的な規則は，それぞれ,

$$\frac{\Gamma \vdash t_1 : T_1 \to T_2 \qquad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1\ t_2 : T_2} \tag{T-App}$$

$$\frac{\Gamma, x{:}T_1 \vdash t_0 : T_2}{\Gamma \vdash \lambda x{:}T_1.t_0 : T_1 \to T_2} \tag{T-Abs}$$

と記述することができる．

以上のようにして得られる計算体系 (今までと同様の手法で定義される簡約関係も含む) を**単純型付 λ 計算**(*simply typed λ-calculus*) と呼ぶ．これに対して，前回まで扱った体系を**型無し λ 計算**(*untyped λ-calculus*) と呼ぶことがある．

2.3 型安全性

型付け可能な項に関しては，そもそもの動機づけで触れた，エラーが発生しない，という性質が満たされていてほしい．このような型システムが提供する安全性を**型安全性**(*type safety*) という．型安全性は

- $\Gamma \vdash t : T$ かつ $t \longrightarrow_v t'$ ならば，$\Gamma \vdash t' : T$ である．
- $\vdash t : T$ かつ t が値でなければ，ある項 t' が存在して $t \longrightarrow_v t'$ である．

という，それぞれ Type Preservation, Progress と呼ばれる，ふたつの性質を組合せることで示される．つまり，型付け可能なプログラム (自由変数のない項) は `1 + true` のような，値でもないのに簡約しようのない項に辿りつくことがないことが示される．

3 形式的定義

3.1 定義: 型 T，項 t，数値 n，値 v，型環境 Γ は以下の文法で定義される．

$$
\begin{array}{rcl}
T & ::= & \texttt{nat} \mid \texttt{bool} \mid T \to T \\
t & ::= & \#^i x \mid \lambda x{:}T.t \mid t\ t \mid \texttt{fix} \\
 & \mid & \texttt{0} \mid \texttt{S}(t) \mid \texttt{pred}\ t \mid t\ \texttt{+}\ t \mid t\ \texttt{*}\ t \\
 & \mid & \texttt{true} \mid \texttt{false} \mid \texttt{if}\ t\ \texttt{then}\ t\ \texttt{else}\ t \\
n & ::= & \texttt{0} \mid \texttt{S}(n) \\
v & ::= & \#^i x \mid \lambda x{:}T.t \mid n \mid \texttt{true} \mid \texttt{false} \\
\Gamma & ::= & \bullet \mid \Gamma, x{:}T
\end{array}
$$

□

$FV(t)$, $\Uparrow_x t$, $\Downarrow_x t$, $[t_1/x]t_2$ は形無し λ 計算のそれらと同様な定義．$\Downarrow_x([(\Uparrow_x t_2)/x]t_1)$ を $[x \mapsto t_1]t_2$ と書く．

$$\frac{}{\Gamma, x{:}T \vdash \#^0 x : T}\ (\text{T-Var}) \qquad \frac{\Gamma \vdash t : \texttt{nat}}{\Gamma \vdash \texttt{S}(t) : \texttt{nat}}\ (\text{T-Succ})$$

$$\frac{\Gamma \vdash \#^n x : T}{\Gamma, x{:}T' \vdash \#^{n+1} x : T}\ (\text{T-Weak1}) \qquad \frac{\Gamma \vdash t : \texttt{nat}}{\Gamma \vdash \texttt{pred}\ t : \texttt{nat}}\ (\text{T-Pred})$$

$$\frac{\Gamma \vdash \#^n x : T \qquad x \neq y}{\Gamma, y{:}T' \vdash \#^n x : T}\ (\text{T-Weak2}) \qquad \frac{\Gamma \vdash t_1 : \texttt{nat} \qquad \Gamma \vdash t_2 : \texttt{nat}}{\Gamma \vdash t_1\ \texttt{+}\ t_2 : \texttt{nat}}\ (\text{T-Plus})$$

$$\frac{\Gamma, x{:}T_1 \vdash t_0 : T_2}{\Gamma \vdash \lambda x{:}T_1.t_0 : T_1 \to T_2}\ (\text{T-Abs}) \qquad \frac{\Gamma \vdash t_1 : \texttt{nat} \qquad \Gamma \vdash t_2 : \texttt{nat}}{\Gamma \vdash t_1\ \texttt{*}\ t_2 : \texttt{nat}}\ (\text{T-Mult})$$

$$\frac{\Gamma \vdash t_1 : T_1 \to T_2 \qquad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1\ t_2 : T_2}\ (\text{T-App}) \qquad \frac{}{\Gamma \vdash \texttt{true} : \texttt{bool}}\ (\text{T-True})$$

$$\frac{\Gamma \vdash t : T \to T}{\Gamma \vdash \texttt{fix}\ t : T}\ (\text{T-Fix}) \qquad \frac{}{\Gamma \vdash \texttt{false} : \texttt{bool}}\ (\text{T-False})$$

$$\frac{}{\Gamma \vdash \texttt{0} : \texttt{nat}}\ (\text{T-Zero}) \qquad \frac{\Gamma \vdash t_1 : \texttt{bool} \qquad \Gamma \vdash t_2 : T \qquad \Gamma \vdash t_3 : T}{\Gamma \vdash \texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 : T}\ (\text{T-If})$$

3.2 定義: 簡約関係 $\longrightarrow_v$ は以下の規則で定義される.

$$\frac{}{(\lambda x{:}T.t_1)\ v_2 \longrightarrow_v [x \mapsto v_2]t_1} \quad \text{(Ev-AbsApp)}$$

$$\frac{}{\texttt{fix}\ (\lambda x{:}T.t_0) \longrightarrow_v [x \mapsto \texttt{fix}\ (\lambda x{:}T.t_0)]t_0} \quad \text{(Ev-FixAbs)}$$

□

4 諸性質

4.1 定理 [Uniquness of Typing]: $\Gamma \vdash t : T$ かつ $\Gamma \vdash t : T'$ ならば $T \equiv T'$ である.

4.2 定理 [Type Preservation]: $\Gamma \vdash t : T$ かつ $t \longrightarrow_v t'$ ならば, $\Gamma \vdash t' : T$ である.

4.3 定理 [Progress]: $\bullet \vdash t : T$ かつ t が値でなければ, ある項 t' が存在して $t \longrightarrow_v t'$ である.

4.4 定理 [Normalization]: $\Gamma \vdash t : T$ かつ t が `fix` を含まないならば $t \longrightarrow_v t_1 \longrightarrow_v \cdots \longrightarrow_v t_n \longrightarrow_v \cdots$ なる無限列は存在しない.