ソフトウェア構成特論 第9回

大学院理工学研究科 電気電子情報工学専攻　篠埜 功

2013年6月11日

1　はじめに

今回は、preservationの証明を行ってから、単純型付きラムダ計算 (simply typed lambda calculus) について紹介する。

2　preservationの証明

preservationは以下の命題である。

定理 1 (preservation)**.** $\mathtt{t} : \mathtt{T}$ かつ $\mathtt{t} \to \mathtt{t}'$ ならば $\mathtt{t}' : \mathtt{T}$ である。

証明. この命題を型判定の導出木に関する性質ととらえ、以下の性質 P を、導出木の構造に関する帰納法で証明する。

P(型判定の導出木 t) : 型判定の導出木 t の一番下の型判定が $\mathtt{t} : \mathtt{T}$ のとき、$\mathtt{t} \to \mathtt{t}'$が成り立つならば $\mathtt{t}' : \mathtt{T}$ が成り立つ

$\mathtt{t} : \mathtt{T}$ の導出木において一番最後に使われた（つまり一番下の）型付け規則で場合分けする。

T-TRUE の場合: 型判定の導出木は

$$\frac{}{\mathtt{true} : \mathtt{Bool}}\ \text{(T-TRUE)}$$

である。$\mathtt{true} \to \mathtt{t}'$ となる $\mathtt{t}'$ は存在しない。つまり、性質 P は前提（ならばの左側）が成り立たないので成り立つ。

T-FALSE の場合: 型判定の導出木は

$$\frac{}{\mathtt{false} : \mathtt{Bool}}\ \text{(T-FALSE)}$$

である。$\mathtt{false} \to \mathtt{t}'$ となる $\mathtt{t}'$ は存在しない。つまり、性質 P は前提（ならばの左側）が成り立たないので成り立つ。

T-ZERO の場合: 型判定の導出木は

$$\frac{}{\mathtt{0} : \mathtt{Nat}}\ \text{(T-ZERO)}$$

である。$\mathtt{0} \to \mathtt{t}'$ となる $\mathtt{t}'$ は存在しない。つまり、性質 P は前提（ならばの左側）が成り立たないので成り立つ。

T-IF の場合: 型判定の導出木の一番下の部分は

$$\frac{\mathtt{t}_1 : \mathtt{Bool} \quad \mathtt{t}_2 : \mathtt{T} \quad \mathtt{t}_3 : \mathtt{T}}{\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 : \mathtt{T}}\ \text{(T-IF)}$$

という形をしている。if 式の評価規則は 3 つあり、(E-IFTRUE), (E-IFFALSE), (E-IF) である。よって、$\mathtt{t} \to \mathtt{t}'$ となる $\mathtt{t}'$ が存在するとしたら、この 3 つの規則のどれかがこの評価判定の導出において一番最後に使われた規則である。この 3 つで場合分けをする。

E-IFTRUE の場合: 評価判定の導出木は

$$\frac{}{\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 : \mathtt{T} \to \mathtt{t}_2}\ \text{(E-IFTRUE)}$$

である。上で述べたように、$\mathtt{t}_2 : \mathtt{T}$ であり、性質 P は成り立つ。

E-IFFALSE の場合: E-IFTRUE の場合と同様。

E-IF の場合: 評価判定の導出木の一番下の部分は

$$\frac{\mathtt{t}_1 \to \mathtt{t}_1'}{\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \to \mathtt{if}\ \mathtt{t}_1'\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3}\ \text{(E-IF)}$$

という形をしている。帰納法の仮定より、$\mathtt{t}_1' : \mathtt{Bool}$ である。これと $\mathtt{t}_2 : \mathtt{T}$ および $\mathtt{t}_3 : \mathtt{T}$ に対して T-IF 規則を適用すると $\mathtt{if}\ \mathtt{t}_1'\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 : \mathtt{T}$、つまり、$\mathtt{t}' : \mathtt{T}$ が得られる。

T-SUCC の場合: 型判定の導出木の一番下の部分は

$$\frac{\mathtt{t}_1 : \mathtt{Nat}}{\mathtt{succ}\ \mathtt{t}_1 : \mathtt{Nat}}\ \text{(T-SUCC)}$$

という形をしている。$\mathtt{succ}\ \mathtt{t}_1$ の形に適用できる評価規則は E-SUCC のみであるので、$\mathtt{t} \to \mathtt{t}'$ となる $\mathtt{t}'$ が存在する場合は、その評価判定に対する導出木の一番下の部分が

$$\frac{\mathtt{t}_1 \to \mathtt{t}_1'}{\mathtt{succ}\ \mathtt{t}_1 \to \mathtt{succ}\ \mathtt{t}_1'}\ \text{(E-SUCC)}$$

という形をしている。帰納法の仮定より、$\mathtt{t}_1' : \mathtt{Nat}$ であり、これに T-SUCC 規則を適用すると $\mathtt{succ}\ \mathtt{t}_1' : \mathtt{Nat}$、つまり $\mathtt{t}' : \mathtt{Nat}$ となり、成立する。

T-PRED の場合: 型判定の導出木の一番下の部分は

$$\frac{\mathtt{t}_1 : \mathtt{Nat}}{\mathtt{pred}\ \mathtt{t}_1 : \mathtt{Nat}}\ \text{(T-PRED)}$$

という形をしている。pred 式の評価規則は 3 つあり、(E-PREDZERO), (E-PREDSUCC), (E-PRED) である。よって、$\mathtt{t} \to \mathtt{t}'$ となる $\mathtt{t}'$ が存在する場合は、この 3 つの規則のどれかがこの評価判定の導出において一番最後に使われた規則である。この 3 つで場合分けする。

E-PREDZERO の場合: 評価判定の導出木は

$$\frac{}{\texttt{pred 0} \to \texttt{0}}\ \text{(E-PREDZERO)}$$

であり、(T-ZERO) 規則より $\texttt{0} : \texttt{Nat}$ であり、成立する。

E-PREDSUCC の場合: 評価判定の導出木の一番下の部分は

$$\frac{}{\texttt{pred}\ (\texttt{succ}\ \texttt{nv}_1) \to \texttt{nv}_1}\ \text{(E-PREDSUCC)}$$

という形をしている。inversion lemma より $\texttt{nv}_1 : \texttt{Nat}$ であり、成立する。

E-PRED の場合: 評価判定の導出木の一番下の部分は

$$\frac{\texttt{t}_1 \to \texttt{t}_1'}{\texttt{pred t}_1 \to \texttt{pred t}_1'}\ \text{(E-PRED)}$$

という形をしている。よって、帰納法の仮定より $\texttt{t}_1' : \texttt{Nat}$ が成り立ち、(T-PRED) 規則より $\texttt{pred t}_1' : \texttt{Nat}$ となり、成立する。

T-ISZERO の場合: 型判定の導出木の一番下の部分は

$$\frac{\texttt{t}_1 : \texttt{Nat}}{\texttt{iszero t}_1 : \texttt{Bool}}\ \text{(T-ISZERO)}$$

という形をしている。iszero 式の評価規則は 3 つあり、(E-ISZEROZERO), (E-ISZEROSUCC), (E-ISZERO) である。よって、$\texttt{t} \to \texttt{t}'$ となる $\texttt{t}'$ が存在する場合は、この 3 つの規則のどれかがこの評価判定の導出において一番最後に使われた規則である。この 3 つで場合分けする。

E-ISZEROZERO の場合: 評価判定の導出木は

$$\frac{}{\texttt{iszero 0} \to \texttt{true}}\ \text{(E-ISZEROZERO)}$$

であり、inversion lemma より $\texttt{true} : \texttt{Bool}$ であり、成立する。

E-ISZEROSUCC の場合: 評価判定の導出木は

$$\frac{}{\texttt{iszero}\ (\texttt{succ nv}) \to \texttt{false}}\ \text{(E-ISZEROSUCC)}$$

という形をしている。inversion lemma より $\texttt{false} : \texttt{Bool}$ であり、成立する。

E-ISZERO の場合: 評価判定の導出木の一番下の部分は

$$\frac{\texttt{t}_1 \to \texttt{t}_1'}{\texttt{iszero t}_1 \to \texttt{iszero t}_1'}\ \text{(E-ISZERO)}$$

という形をしている。よって、帰納法の仮定より $\texttt{t}_1' : \texttt{Nat}$ が成り立ち、(T-ISZERO) 規則より $\texttt{iszero t}_1' : \texttt{Bool}$ となり、成立する。

□

参考　この preservation 定理は subject reduction あるいは subject evaluation とも呼ばれる。型判定 $\texttt{t} : \texttt{T}$ は「算術式 t が型 T を持つ」という文（statement）であり、この文の主語が t で、t を reduce（あるいは evaluate）してもこの文が成立するということからこのように呼ばれる。

3　単純型付きラムダ計算

前回紹介したラムダ計算に型システムを導入する。

3.1　関数型

まず、ラムダ抽象が直感的には関数を表すので、ラムダ抽象に関数型を付けるという型付け規則

$$\frac{}{\lambda \mathtt{x}.\mathtt{t} : \rightarrow}$$

を考えてみる。これだと、ラムダ抽象式を何かのラムダ式に適用した場合の関数適用式の型が分からなくなってしまう。よってラムダ抽象式の型には結果の型が必要である。また、ラムダ抽象式が何らかのラムダ式に適用される場合に、ラムダ抽象式がどういう型の式を引数として期待しているかについての情報が必要である。そこで、$\rightarrow$ の代わりに $\mathtt{T}_1 \rightarrow \mathtt{T}_2$ という形の型（関数型という）を用いる。

ここでは、以下のように定義される、`Bool` 型と関数型からなる型を用いる。

定義 1. `Bool` 型上の単純型（*simple types over the type* `Bool`）の集合は以下のように定義される。

$$\begin{array}{rcl} \mathtt{T} & ::= & \mathtt{T} \rightarrow \mathtt{T} \\ & | & \mathtt{Bool} \end{array}$$

型構成子（*type constructor*）$\rightarrow$ は右結合とする。つまり、$\mathtt{T}_1 \rightarrow \mathtt{T}_2 \rightarrow \mathtt{T}_3$ という型式（*type expression*）は $\mathtt{T}_1 \rightarrow (\mathtt{T}_2 \rightarrow \mathtt{T}_3)$ を表す。

3.2　型付け関係

まず、ラムダ抽象式に対する型付け規則をどのように与えたらよいかを考える。これには 2 つの可能性があり、1 つは、ラムダ抽象の束縛変数の部分に

$$\lambda \mathtt{x} : \mathtt{T}_1.\ \mathtt{t}_2$$

のように型を書くというものである。もう 1 つは、束縛変数がラムダ抽象の本体内でどのように使われるかを解析して束縛変数の型を計算するという方法である。ここでは、束縛変数の部分に型を書くという方法をとる。このようにプログラム中に書かれた型情報を**型注釈** (type annotation) という。

一般に、型注釈のある言語を**陽に型付けられた**言語 (*explicitly typed* language) といい、それに対し、型注釈がなく、変数の型を推論する (infer または reconstruct) 言語を**暗に型付けられた**言語 (*implicitly typed* language) という。この講義では陽に型付けられた言語を主に扱う。

型注釈によりラムダ抽象の束縛変数の型が分かると、あとはラムダ抽象の本体の型が返り値の型となる。これを規則にすると以下の形になる。

$$\frac{\mathtt{x} : \mathtt{T}_1 \vdash \mathtt{t}_2 \ :\ \mathtt{T}_2}{\vdash \lambda \mathtt{x} : \mathtt{T}_1.\ \mathtt{t}_2 \ :\ \mathtt{T}_1 \rightarrow \mathtt{T}_2}$$

ラムダ抽象 $\lambda \mathtt{x} : \mathtt{T}_1.\ \mathtt{t}_2$ の本体 $\mathtt{t}_2$ についての型判定においては、変数 $\mathtt{x}$ の型が $\mathtt{T}_1$ であることを使う（かもしれない）ので、型判定は、$\Gamma \vdash \mathtt{t} \ : \ \mathtt{T}$ の 3 項関係の形で表される。Γ はラムダ式 $\mathtt{t}$ 中の各自由変数とその型の集合であり、**型環境** (*typing context* または *type environment*) と言う。形式的には、型環境は変数と型の列であり、**コンマ演算子**により型環境に対して右側に新しい binding(変数と型の対応、$\mathtt{x} \ : \ \mathtt{T}$ の形で記述) を加える。空の型環境は $\emptyset$ と書かれることもあるが、単に、$\vdash \mathtt{t} \ : \ \mathtt{T}$ のような形で何も書かない場合もある。

紛らわしくならないようにするため、$\Gamma, \mathtt{x} : \mathtt{T}$ のように書くときには Γ 中に $\mathtt{x}$ はないものとする。この講義では、ラムダ抽象の束縛変数の名前は自由に付け替えてよいとしたので、この条件に合致するように名前の付け替えを行うことができる。Γ は変数から型への関数と考えることもできる。この直観に従い、$dom(\Gamma)$ により、Γ で束縛される変数の集合を表す。

以上より、ラムダ抽象に対する型付け規則は以下の形になる。

$$\frac{\Gamma, \mathtt{x} : \mathtt{T}_1 \vdash \mathtt{t}_2 \ : \ \mathtt{T}_2}{\Gamma \vdash \lambda \mathtt{x} : \mathtt{T}_1.\ \mathtt{t}_2 \ : \ \mathtt{T}_1 \to \mathtt{T}_2} \text{ (T-ABS)}$$

また、変数に対する型付け規則は以下のように与えればよい。

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

関数適用に対する型付け規則は以下のように与えればよい。

$$\frac{\Gamma \vdash \mathtt{t}_1 \ : \ \mathtt{T}_{11} \to \mathtt{T}_{12} \qquad \Gamma \vdash \mathtt{t}_2 \ : \ \mathtt{T}_{11}}{\Gamma \vdash \mathtt{t}_1 \ \mathtt{t}_2 \ : \ \mathtt{T}_{12}} \text{ (T-APP)}$$

if 式に対する型付け規則は以下のように与えればよい。

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{Bool} \qquad \Gamma \vdash \mathtt{t}_2 : \mathtt{T} \qquad \Gamma \vdash \mathtt{t}_3 : \mathtt{T}}{\Gamma \vdash \mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \ : \ \mathtt{T}} \text{ (T-IF)}$$

これらをまとめると以下のようになる。

ラムダ式の集合は以下のように定義する。

$$\begin{aligned} \mathtt{t} \ &:= \ \mathtt{x} \mid \lambda \mathtt{x} : \mathtt{T}.\mathtt{t} \mid \mathtt{t}\ \mathtt{t} \\ &\mid \ \mathtt{true} \mid \mathtt{false} \mid \mathtt{if}\ \mathtt{t}\ \mathtt{then}\ \mathtt{t}\ \mathtt{else}\ \mathtt{t} \end{aligned}$$

値は以下のように定義する。

$$\mathtt{v} ::= \lambda \mathtt{x} : \mathtt{T}.\mathtt{t} \mid \mathtt{true} \mid \mathtt{false}$$

型は以下のように定義する。

$$\mathtt{T} ::= \mathtt{T} \to \mathtt{T} \mid \mathtt{Bool}$$

型環境は以下のように定義する。

$$\Gamma ::= \emptyset \mid \Gamma, \mathtt{x} : \mathtt{T}$$

評価規則は以下のように定義する。

$$\frac{\mathtt{t}_1 \to \mathtt{t}_1'}{\mathtt{t}_1\ \mathtt{t}_2 \to \mathtt{t}_1'\ \mathtt{t}_2} \text{ (E-APP1)} \quad \frac{\mathtt{t}_2 \to \mathtt{t}_2'}{\mathtt{v}_1\ \mathtt{t}_2 \to \mathtt{v}_1\ \mathtt{t}_2'} \text{ (E-APP2)}$$

$$\frac{}{(\lambda \mathtt{x} : \mathtt{T}_{11}.\mathtt{t}_{12})\ \mathtt{v}_2 \to [\mathtt{x} \mapsto \mathtt{v}_2]\mathtt{t}_{12}}\ \text{(E-APPABS)}$$

$$\frac{}{\mathtt{if\ true\ then\ t}_2\ \mathtt{else\ t}_3 \to \mathtt{t}_2}\ \text{(E-IFTRUE)}$$

$$\frac{}{\mathtt{if\ false\ then\ t}_2\ \mathtt{else\ t}_3 \to \mathtt{t}_3}\ \text{(E-IFFALSE)}$$

$$\frac{\mathtt{t}_1 \to \mathtt{t}_1'}{\mathtt{if\ t}_1\ \mathtt{then\ t}_2\ \mathtt{else\ t}_3 \to \mathtt{if\ t}_1'\ \mathtt{then\ t}_2\ \mathtt{else\ t}_3}\ \text{(E-IF)}$$

型付け規則は以下のように定義する。

$$\frac{\mathtt{x} : \mathtt{T} \in \Gamma}{\Gamma \vdash \mathtt{x}\ :\ \mathtt{T}}\ \text{(T-VAR)} \qquad \frac{\Gamma, \mathtt{x} : \mathtt{T}_1 \vdash \mathtt{t}_2\ :\ \mathtt{T}_2}{\Gamma \vdash \lambda \mathtt{x} : \mathtt{T}_1.\ \mathtt{t}_2\ :\ \mathtt{T}_1 \to \mathtt{T}_2}\ \text{(T-ABS)}$$

$$\frac{\Gamma \vdash \mathtt{t}_1\ :\ \mathtt{T}_{11} \to \mathtt{T}_{12} \qquad \Gamma \vdash \mathtt{t}_2\ :\ \mathtt{T}_{11}}{\Gamma \vdash \mathtt{t}_1\ \mathtt{t}_2\ :\ \mathtt{T}_{12}}\ \text{(T-APP)}$$

$$\frac{}{\Gamma \vdash \mathtt{true} : \mathtt{Bool}}\ \text{(T-TRUE)} \qquad \frac{}{\Gamma \vdash \mathtt{false} : \mathtt{Bool}}\ \text{(T-FALSE)}$$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{Bool} \qquad \Gamma \vdash \mathtt{t}_2 : \mathtt{T} \qquad \Gamma \vdash \mathtt{t}_3 : \mathtt{T}}{\Gamma \vdash \mathtt{if\ t}_1\ \mathtt{then\ t}_2\ \mathtt{else\ t}_3\ :\ \mathtt{T}}\ \text{(T-IF)}$$

置換は以下のように定義される。

$$\begin{array}{ll}
[\mathtt{x} \mapsto \mathtt{s}]\mathtt{x} = \mathtt{s} & \\
[\mathtt{x} \mapsto \mathtt{s}]\mathtt{y} = \mathtt{y} & \text{if } \mathtt{y} \neq \mathtt{x} \\
[\mathtt{x} \mapsto \mathtt{s}](\lambda \mathtt{y} : \mathtt{T}.\ \mathtt{t}_1) = \lambda \mathtt{y} : \mathtt{T}.\ [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_1 & \text{if } \mathtt{y} \neq \mathtt{x} \text{ and } \mathtt{y} \notin FV(\mathtt{s}) \\
[\mathtt{x} \mapsto \mathtt{s}](\mathtt{t}_1\ \mathtt{t}_2) = ([\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_1)([\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_2) & \\
[\mathtt{x} \mapsto \mathtt{s}]\mathtt{true} = \mathtt{true} & \\
[\mathtt{x} \mapsto \mathtt{s}]\mathtt{false} = \mathtt{false} & \\
[\mathtt{x} \mapsto \mathtt{s}](\mathtt{if\ t}_1\ \mathtt{then\ t}_2\ \mathtt{else\ t}_3) = & \\
\quad \mathtt{if}\ [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_1\ \mathtt{then}\ [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_2\ \mathtt{else}\ [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_3 &
\end{array}$$

ただし、第 6 回の講義資料で説明した通り、ラムダ抽象に対する置換適用時には、必要に応じて置換対象のラムダ抽象式の束縛変数の付け替えを行う。

以降、単純型付きラムダ計算を $\lambda_{\to}$ と書く。

例 1. 型判定 $\vdash$ (λx:Bool.x) true : Bool は成立する。これの導出は黒板で行う。

練習問題 1. 以下の型判定の導出木を書け。

1. f:Bool→Bool ⊢ f (if false then true else false) : Bool
2. f:Bool→Bool ⊢ λx:Bool. f (if x then false else x) : Bool → Bool