# コード生成 + Shift0/Reset0 の型システム

大石純平

平成 28 年 8 月 9 日

answer type は考えていない．
後で，answer type を加えたやつを考える．
answer type modification については考えない

## 1 Syntax

$$
\begin{aligned}
v &::= c_0 \mid \lambda x.e \mid \texttt{<}e\texttt{>} \\
e &::= x \mid c_0 \mid c_1\ e \mid c_2\ e_1\ e_2 \mid c_3\ e_1\ e_2\ e_3 \mid \lambda x.e \mid e_1\ e_2 \\
&\quad \mid \underline{\lambda} x.e \mid \underline{\underline{\lambda}} x.e \mid \textbf{reset0}\ e \mid \textbf{shift0}\ k \to e \mid \textbf{throw}\ k\ e \\
&\quad \mid \underline{\textbf{clet}}\ x = e_1\ \underline{\textbf{in}}\ e_2 \mid \textbf{if}\ e_1\ \textbf{then}\ e_2\ \textbf{else}\ e_3 \\
c_0 &::= N \mid B \\
&\quad \text{N is Integer numeric (1,2,3,}\cdots\text{), B is Bool (true or false)} \\
c_1 &::= \underline{\textbf{cint}} \mid \textbf{fix} \mid \underline{\textbf{fix}} \\
c_2 &::= \underline{@} \mid + \mid \underline{+} \\
c_3 &::= \underline{\textbf{cif}}
\end{aligned}
$$

## 2 Semantics

left-to-right, call-by-value

### 2.1 Evaluation Context

$$
\begin{aligned}
E &::= [\,] \mid E\ e \mid v\ E \\
&\quad \mid c_1\ E \mid c_2\ E\ e \mid c_2\ v\ E \\
&\quad \mid \textbf{if}\ E\ \textbf{then}\ e_1\ \textbf{else}\ e_2 \mid \textbf{reset0}\ E \mid \underline{\underline{\lambda}} x.E
\end{aligned}
$$

### 2.2 Operation Semantics

underline 付きのものは，コードコンビネータであり，なにか値を受け取ってコードを出すもの
underline がないもの： present stage で動く
underline があるもの： present stage で動かない
**shift0 reset0 throw** は コードの型を持つ $e$ のみを引数に取ることにする？ ⇒ する
コードレベルで shift0/reset0 throw は出てこないようにする？ ⇒ する
**throw** $k\ e$ ってあるけど，これ，**throw** $e$ にしたほうがいい？ ⇒ 良くない．

$$\frac{E[e] \rightsquigarrow E[e']}{e \rightsquigarrow e'}$$

$$
\begin{aligned}
(\lambda x.e)\ v &\rightsquigarrow e\{x := v\} \\
\mathbf{let}\ x = v\ \mathbf{in}\ e &\rightsquigarrow e\{x := v\} \\
\mathbf{if}\ \mathit{true}\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2 &\rightsquigarrow e_1 \\
\mathbf{if}\ \mathit{else}\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2 &\rightsquigarrow e_2 \\
\underline{\lambda} x.e &\rightsquigarrow \underline{\lambda} y.e\{x := <y>\} \\
& \quad y \text{ is fresh variable} \\
\underline{\lambda} y.<e> &\rightsquigarrow <\lambda y.e> \\
\mathbf{reset0}\ v &\rightsquigarrow v \\
\mathbf{reset0}(E[\mathbf{shift0}\ k \to e]) &\rightsquigarrow e\{k := \underline{\lambda} x.\mathbf{reset0}(E[x])\} \\
& \quad x \text{ is fresh variable} \\
\mathbf{throw}\ k\ v &\rightsquigarrow k\ v
\end{aligned}
$$

**throw** の簡約は，代入に lambda-substitution のような感じで定義することで無くす？

$$
\begin{aligned}
\mathbf{fix}\ e &\rightsquigarrow e\ (\mathbf{fix}\ e) \\
\underline{\mathbf{fix}}\ e &\rightsquigarrow <\mathbf{fix}\ e> \\
\underline{\mathbf{cint}}\ n &\rightsquigarrow <n> \\
<e_1>\ \underline{@}\ <e_2> &\rightsquigarrow <e_1\ e_2> \\
<e_1>\ \underline{+}\ <e_2> &\rightsquigarrow <e_1 + e_2> \\
\underline{\mathbf{cif}}\ <e_1>\ <e_2>\ <e_3> &\rightsquigarrow <\mathbf{if}\ e_1\ \mathbf{then}\ e_2\ \mathbf{else}\ e_3> \\
\underline{\mathbf{clet}}\ x = e_1\ \underline{\mathbf{in}}\ e_2 &\rightsquigarrow <\mathbf{let}\ x = e_1\ \mathbf{in}\ e_2>
\end{aligned}
$$

**簡約例**

$$
\begin{aligned}
e_1 = {} & \mathbf{reset0}\ \ \underline{\mathbf{clet}}\ x_1 = \%3\ \underline{\mathbf{in}} \\
& \mathbf{reset0}\ \ \underline{\mathbf{clet}}\ x_2 = \%5\ \underline{\mathbf{in}} \\
& \mathbf{shift0}\ k\ \to\ \underline{\mathbf{clet}}\ y = t\ \underline{\mathbf{in}} \\
& \mathbf{throw}\ k\ (x_1\ \underline{+}\ x_2\ \underline{+}\ y)
\end{aligned}
$$

$$
\begin{aligned}
&[e_1] \rightsquigarrow [\mathbf{reset0}(\underline{\mathbf{clet}}\ x_1 = \%3\ \underline{\mathbf{in}} \\
&\quad \mathbf{reset0}\ \underline{\mathbf{clet}}\ x_2 = \%5\ \underline{\mathbf{in}} \\
&\quad [\mathbf{shift0}\ k \ \rightarrow\ \underline{\mathbf{clet}}\ y = t\ \underline{\mathbf{in}} \\
&\quad [\mathbf{throw}\ k\ (x_1 \mathbin{\underline{+}} x_2 \mathbin{\underline{+}} y)]]])] \\
&\quad \rightsquigarrow [\underline{\mathbf{clet}}\ y = t\ \underline{\mathbf{in}} \\
&\quad [\underline{\lambda} x.\mathbf{reset0}\ (\underline{\mathbf{clet}}\ x_1 = \%3\ \underline{\mathbf{in}}\ \mathbf{reset0}\ (\underline{\mathbf{clet}}\ x_2 = \%5\ \underline{\mathbf{in}}[x]))(x_1 \mathbin{\underline{+}} x_2 \mathbin{\underline{+}} y)]] \\
&\quad \rightsquigarrow [\underline{\lambda} y.(\underline{\lambda} x.\mathbf{reset0}\ (\underline{\mathbf{clet}}\ x_1 = \%3\ \underline{\mathbf{in}}\ \mathbf{reset0}\ (\underline{\mathbf{clet}}\ x_2 = \%5\ \underline{\mathbf{in}}[x]))(x_1 \mathbin{\underline{+}} x_2 \mathbin{\underline{+}} y))\ \underline{@}\ t] \\
&\quad \rightsquigarrow [[\underline{\lambda} y.(\underline{\lambda} x.\mathbf{reset0}\ (\underline{\mathbf{clet}}\ x_1 = \%3\ \underline{\mathbf{in}}\ \mathbf{reset0}\ (\underline{\mathbf{clet}}\ x_2 = \%5\ \underline{\mathbf{in}}[x]))(x_1 \mathbin{\underline{+}} x_2 \mathbin{\underline{+}} y))]\ \underline{@}\ t] \\
&\quad \rightsquigarrow [[\underline{\lambda} y_1.(\underline{\lambda} x.\mathbf{reset0}\ (\underline{\mathbf{clet}}\ x_1 = \%3\ \underline{\mathbf{in}}\ \mathbf{reset0}\ (\underline{\mathbf{clet}}\ x_2 = \%5\ \underline{\mathbf{in}}[x]))(x_1 \mathbin{\underline{+}} x_2 \mathbin{\underline{+}} \texttt{<}y_1\texttt{>}))]\ \underline{@}\ t] \\
&\quad \rightsquigarrow
\end{aligned}
$$

let ref の e1 e2 の制限 scope extrusion 問題への対処 shift reset で同じようなことをかけるので，これについて考える

# 3 Type System

$$\text{BasicType} ::= \text{Int} \mid \text{Bool}$$

$$
\begin{aligned}
t &::= \text{BasicType} \mid t \to t \mid \langle t \rangle^{\gamma} \\
\sigma &::= \epsilon \mid t, \sigma
\end{aligned}
$$

$$\Gamma ::= \emptyset \mid \Gamma, (\gamma \geq \gamma) \mid \Gamma, (x : t)^{L}; \sigma \mid \Gamma, (u^{1} : t)^{\gamma}; \sigma$$

Typing judgements take the form:

$$
\begin{aligned}
&\Gamma \vdash^{L} e : t; \sigma \\
&\Gamma \models \gamma_1 \geq \gamma_2
\end{aligned}
$$

Environment Classifier の rule:

$$\frac{}{\Gamma, \gamma_1 \geq \gamma_2 \models \gamma_1 \geq \gamma_2}$$

$$\frac{\Gamma \models \gamma_1 \geq \gamma_2 \quad \Gamma \models \gamma_2 \geq \gamma_3}{\Gamma \models \gamma_1 \geq \gamma_3}$$

Typing rule for code-level lambda:

$$\frac{\Gamma,\ \gamma_1 \geq \gamma,\ x : \langle t_1 \rangle^{\gamma_1}; \sigma \vdash e : \langle t_2 \rangle^{\gamma_1}; \sigma}{\Gamma \vdash \underline{\lambda} x.e : \langle t_1 \to t_2 \rangle^{\gamma}; \sigma}\ (\gamma_1 \text{ is eigen var})$$

Typing rule for code-level let (derived rule):

$$\frac{\Gamma \vdash e_1 : \langle t_1 \rangle^{\gamma}; \sigma \quad \Gamma,\ \gamma_1 \geq \gamma,\ x : \langle t_1 \rangle^{\gamma_1}; \sigma \vdash e_2 : \langle t_2 \rangle^{\gamma_1}; \sigma}{\Gamma \vdash \underline{\mathbf{clet}}\ x = e_1\ \underline{\mathbf{in}}\ e_2 : \langle t_2 \rangle^{\gamma}; \sigma}\ (\gamma_1 \text{ is eigen var})$$

reset0, shift0, throw は，present-stage で動き，引数はコード型のみ

Typing rule for code-level reset0:

$$\frac{\Gamma \vdash e : \langle t \rangle^{\gamma}; \langle t \rangle^{\gamma}, \sigma}{\Gamma \vdash \mathbf{reset0}\ e : \langle t \rangle^{\gamma}; \sigma}$$

Typing rule for code-level shift0:

$$\frac{\Gamma,\ k : (\langle t_1\rangle^{\gamma_1} \Rightarrow \langle t_0\rangle^{\gamma_0};\sigma) \vdash e : \langle t_0\rangle^{\gamma_0};\sigma \quad \Gamma \models \gamma_1 \geq \gamma_0}{\Gamma \vdash \textbf{shift0}\ k \to e : \langle t_1\rangle^{\gamma_1};\sigma}$$

Typing rule for code-level throw:

$$\frac{\Gamma,\ \gamma_3 \geq \gamma_1,\ \gamma_3 \geq \gamma_2 \vdash e : \langle t_1\rangle^{\gamma_3};\sigma \quad \Gamma \models \gamma_2 \geq \gamma_0}{\Gamma,\ k : (\langle t_1\rangle^{\gamma_1} \Rightarrow \langle t_0\rangle^{\gamma_0});\sigma \vdash \textbf{throw}\ k\ e : \langle t_0\rangle^{\gamma_2};\sigma}\ (\gamma_3 \text{ is eigen var})$$

Typing rule for Subs-0:

$$\frac{\Gamma \vdash e : \langle t\rangle^{\gamma_2};\sigma \quad \Gamma \models \gamma_1 \geq \gamma_2}{\Gamma \vdash e : \langle t\rangle^{\gamma_1};\sigma}$$

Typing rule for Subs-1:

$$\frac{\Gamma \vdash e : \langle t\rangle^{\gamma_2};\sigma \quad \Gamma \models \gamma_1 \geq \gamma_2}{\Gamma \vdash e : \langle t\rangle^{\gamma_1};\sigma}$$

Typing rule for Var:

$$\frac{}{\Gamma, (x:t)^L;\sigma \vdash^L x : t;\sigma}$$

Typing rule for App:

$$\frac{\Gamma \vdash^L e_1 : t_2 \to t_1;\sigma \quad \Gamma \vdash^L e_2 : t_2;\sigma}{\Gamma \vdash^L e_1\ e_2 : t_1;\sigma}$$

Typing rule for Abs:

$$\frac{\Gamma,\ (x:t_1)^L;\sigma \vdash^L e : t_2;\sigma}{\Gamma \vdash^L \lambda x.e : t_1 \to t_2\sigma}$$

Typing rule for If:

$$\frac{\Gamma \vdash^L e_1 : \mathrm{Bool};\sigma \quad \Gamma \vdash^L e_3 : t;\sigma}{\Gamma \vdash^L \textbf{if}\ e_1\ \textbf{then}\ e_2\ \textbf{else}\ e_3 : t;\sigma}$$

Typing rule for CAbs: $\gamma_1 \notin FCV(\Gamma, \langle t_1 \to t_2\rangle^{\gamma})$

$$\frac{\Gamma, \gamma_1 \geq \gamma, x : \langle t_1\rangle^{\gamma_1};\sigma \vdash e : \langle t_2\rangle^{\gamma_1};\sigma}{\Gamma \vdash \underline{\lambda} x.e : \langle t_1 \to t_2\rangle^{\gamma};\sigma}$$

Typing rule for IAbs: $\gamma_1 \notin FCV(\Gamma, \langle t_1 \to t_2\rangle^{\gamma})$

$$\frac{\Gamma, \gamma_1 \geq \gamma, x : (u : t_1)^{\gamma_1};\sigma \vdash e : \langle t_2\rangle^{\gamma_1};\sigma}{\Gamma \vdash \underline{\underline{\lambda}} u^1.e : \langle t_1 \to t_2\rangle^{\gamma};\sigma}$$

Typing rule for Code:

$$\frac{\Gamma \vdash^{\gamma} e^1 : t^1;\sigma}{\Gamma \vdash \texttt{<}e^1\texttt{>} : \langle t^1\rangle^{\gamma};\sigma}$$

Typing rule for Const:

$$\frac{}{\Gamma \vdash^L c : t^c}$$

コードレベルの変数 $u^1$，コードレベルの項 $e^1$，コードレベルの型 $t^1$，あと，$t^c$ についてなぜ必要なのかよく分かっていない...

# 4 Example

$$\begin{aligned} e_1 = {} & \textbf{reset0}\ \ \underline{\textbf{clet}}\ x_1 = \%3\ \underline{\textbf{in}} \\ & \textbf{reset0}\ \ \underline{\textbf{clet}}\ x_2 = \%5\ \underline{\textbf{in}} \\ & \textbf{shift0}\ k\ \to\ \underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}} \\ & \textbf{throw}\ k\ (x_1 \mathbin{\underline{+}} x_2 \mathbin{\underline{+}} y) \end{aligned}$$

If $t = \%7$ or $t = x_1$, then $e_1$ is typable.
If $t = x_2$, then $e_1$ is not typable.

$$
\begin{aligned}
e_2 = \mathbf{reset0}\ \ & \underline{\mathbf{clet}}\ x_1 = \%3\ \underline{\mathbf{in}} \\
& \mathbf{reset0}\ \ \underline{\mathbf{clet}}\ x_2 = \%5\ \underline{\mathbf{in}} \\
& \mathbf{shift0}\ k_2\ \to\ \mathbf{shift0}\ k_1\ \to\ \underline{\mathbf{clet}}\ y = t\ \underline{\mathbf{in}} \\
& \mathbf{throw}\ k_1\ (\mathbf{throw}\ k_2\ (x_1 \mathbin{\underline{+}} x_2 \mathbin{\underline{+}} y))
\end{aligned}
$$

If $t = \%7$, then $e_1$ is typable.
If $t = x_2$ or $t = x_1$, then $e_1$ is not typable.

# 5 型安全性の証明

型システムの健全性を型保存定理，進行定理によって証明する

## 5.1 型保存 (subject reduction)

**定理 5.1 (型保存)**
$\vdash e : t$ かつ $e \rightsquigarrow e'$ であれば，$\vdash e' : t$ である

**補題 5.1 (代入)**
$\Gamma_1, \Gamma_2, x : t_1 \vdash e : t_2$ かつ $\Gamma_1 \vdash v : t_1$ ならば，$\Gamma_1, \Gamma_2 \vdash e\{x := v\} : t_2$

**補題 5.2 (**$\underline{\lambda}$**)**
$\Gamma, \gamma_1 \geq \gamma, x : \langle t_1 \rangle^{\gamma_1}; \sigma \vdash e : \langle t_2 \rangle^{\gamma_1}; \sigma$ ならば，$\Gamma, \gamma_1 \geq \gamma, (y : t_1)^{\gamma_1}; \sigma \vdash e_3\{x := \texttt{<}y\texttt{>}\} : \langle t_2 \rangle^{\gamma_1}; \sigma$

**補題 5.3 (**$\underline{\lambda}$**)**
$\gamma_1 \geq \gamma, \Gamma, (y : t_1)^{\gamma_1}; \sigma \vdash^{\gamma_1} e : t_2; \sigma$ ならば，$\Gamma, (y : t_1)^{\gamma} \vdash^{\gamma} e : t_2$

**補題 5.4 (reset0 – shift0)**
$\Gamma \models \gamma_1 \geq \gamma_0 \Gamma, k : (\langle t_1 \rangle^{\gamma_1} \Rightarrow \langle t_0 \rangle^{\gamma_0}); \sigma \vdash e : \langle t_0 \rangle^{\gamma_0}; \langle t_0 \rangle^{\gamma_0}$ ならば，$k : (\langle t_1 \rangle^{\gamma_1} \Rightarrow \langle t_2 \rangle^{\gamma_0}) \vdash v : \langle t_1 \rangle^{\gamma_3}$

answer type のところ，よく考える

**証明**

## 5.2 進行

**定理 5.2 (進行)**
$\vdash e : t$ が導出可能であれば，$e$ は 値 $v$ である．または，$e \rightsquigarrow e'$ であるような 項 $e'$ が存在する

**証明**　$\vdash e : t$ の導出に関する帰納法による．
Const, Abs, Code 規則の場合 $e$ は値である．
Var 規則の場合 $\vdash e : t$ は導出可能でない．
Throw 規則の場合 $\vdash e : t$ は導出可能でない．
Reset0 規則の場合 $e = \mathbf{reset0}\ e_1$ とする．帰納法の仮定より評価文脈における $\mathbf{reset0}E$ より簡約が進み，
$e_1$ が値のとき，$e \rightsquigarrow v$ となるような $v$ が存在する．
$e_1$ が値でないとき，