

コード生成 + Shift0/Reset0 の型システム

大石純平

平成 28 年 8 月 29 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

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

2 Semantics

left-to-right, call-by-value

2.1 Evaluation Context

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

2.2 Operation Semantics

underline 付きのものは、コードコンピネータであり、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？ \Rightarrow する
コードレベルで **shift0/reset0 throw** は出てこないようにする？ \Rightarrow する
throw $k \ e$ ってあるけど、これ、**throw e** にしたほうがいい？ \Rightarrow 良くない。

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

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

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？

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

簡約例

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

$$\begin{aligned}
[e_1] &\rightsquigarrow [\text{reset0}(\underline{\text{clet}}\ x_1 = \%3\ \underline{\text{in}} \\
&\quad \text{reset0}\ \underline{\text{clet}}\ x_2 = \%5\ \underline{\text{in}} \\
&\quad [\text{shift0}\ k \rightarrow \underline{\text{clet}}\ y = t\ \underline{\text{in}} \\
&\quad [\text{throw}\ k\ (x_1 \pm x_2 \pm y)]])] \\
&\rightsquigarrow [\underline{\text{clet}}\ y = t\ \underline{\text{in}} \\
&\quad [\underline{\lambda x}.\text{reset0}\ (\underline{\text{clet}}\ x_1 = \%3\ \underline{\text{in}}\ \text{reset0}\ (\underline{\text{clet}}\ x_2 = \%5\ \underline{\text{in}}[x]))(x_1 \pm x_2 \pm y)]] \\
&\rightsquigarrow [\underline{\lambda y}.\underline{\lambda x}.\text{reset0}\ (\underline{\text{clet}}\ x_1 = \%3\ \underline{\text{in}}\ \text{reset0}\ (\underline{\text{clet}}\ x_2 = \%5\ \underline{\text{in}}[x]))(x_1 \pm x_2 \pm y))\ @\ t] \\
&\rightsquigarrow [[\underline{\lambda y}.\underline{\lambda x}.\text{reset0}\ (\underline{\text{clet}}\ x_1 = \%3\ \underline{\text{in}}\ \text{reset0}\ (\underline{\text{clet}}\ x_2 = \%5\ \underline{\text{in}}[x]))(x_1 \pm x_2 \pm y))]\ @\ t] \\
&\rightsquigarrow [[\underline{\lambda y_1}.\underline{\lambda x}.\text{reset0}\ (\underline{\text{clet}}\ x_1 = \%3\ \underline{\text{in}}\ \text{reset0}\ (\underline{\text{clet}}\ x_2 = \%5\ \underline{\text{in}}[x]))(x_1 \pm x_2 \pm \langle y_1 \rangle)]\ @\ t] \\
&\rightsquigarrow
\end{aligned}$$

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

BasicType ::= Int | Bool

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

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

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:

$$\begin{aligned}
&\frac{}{\Gamma, \gamma_1 \geq \gamma_2 \models \gamma_1 \geq \gamma_2} \\
&\frac{\Gamma \models \gamma_1 \geq \gamma_2 \quad \Gamma \models \gamma_2 \geq \gamma_3}{\Gamma \models \gamma_1 \geq \gamma_3}
\end{aligned}$$

Typing rule for code-level lambda:

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

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

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

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

Typing rule for code-level reset0:

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

Typing rule for code-level shift0:

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

Typing rule for code-level throw:

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

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:

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

Typing rule for App:

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

Typing rule for Abs:

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

Typing rule for If:

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

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

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

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

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

Typing rule for Code:

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

Typing rule for Const:

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

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

```
e1 = reset0  clet x1 = %3 in
           reset0  clet x2 = %5 in
           shift0 k  → clet y = t in
           throw k (x1 ± x2 ± y)
```

If $t = \%7$ or $t = x_1$, then e_1 is typable.

If $t = x_2$, then e_1 is not typable.

$$\begin{aligned} e_2 = & \text{reset0 } \underline{\text{clet}} \ x_1 = \%3 \ \underline{\text{in}} \\ & \text{reset0 } \underline{\text{clet}} \ x_2 = \%5 \ \underline{\text{in}} \\ & \text{shift0 } k_2 \rightarrow \text{shift0 } k_1 \rightarrow \underline{\text{clet}} \ y = t \ \underline{\text{in}} \\ & \text{throw } k_1 \ (\text{throw } k_2 \ (x_1 \ \underline{\pm} \ x_2 \ \underline{\pm} \ y)) \end{aligned}$$

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， $\Gamma, \gamma_1 \geq \gamma, (y : t_1)^{\gamma_1}; \sigma \vdash e_3\{x := \langle y \rangle\} : \langle t_2 \rangle^{\gamma_1}; \sigma$

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， $\Gamma, (y : t_1)^{\gamma} \vdash^{\gamma} e : t_2; \sigma$

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， $\vdash v : \langle t_1 \rangle^{\gamma_3}$

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 $e = \text{reset0 } e_1$ とする．帰納法の仮定より評価文脈における $\text{reset0 } E$ 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，