多段階 let 挿入を行うコード生成言語の型システムの設計

大石純平　**亀山幸義**

筑波大学 コンピュータ・サイエンス専攻

2016/9/9

アウトライン

アウトライン

研究の目的

表現力と安全性を兼ね備えたコード生成言語の構築

- **表現力**: 多段階 let 挿入，メモ化等の技法を表現
- **安全性**: 生成されるコードの一定の性質を静的に検査

研究の目的

表現力と安全性を兼ね備えたコード生成言語の構築

- **表現力**: 多段階 let 挿入，メモ化等の技法を表現
- **安全性**: 生成されるコードの一定の性質を静的に検査

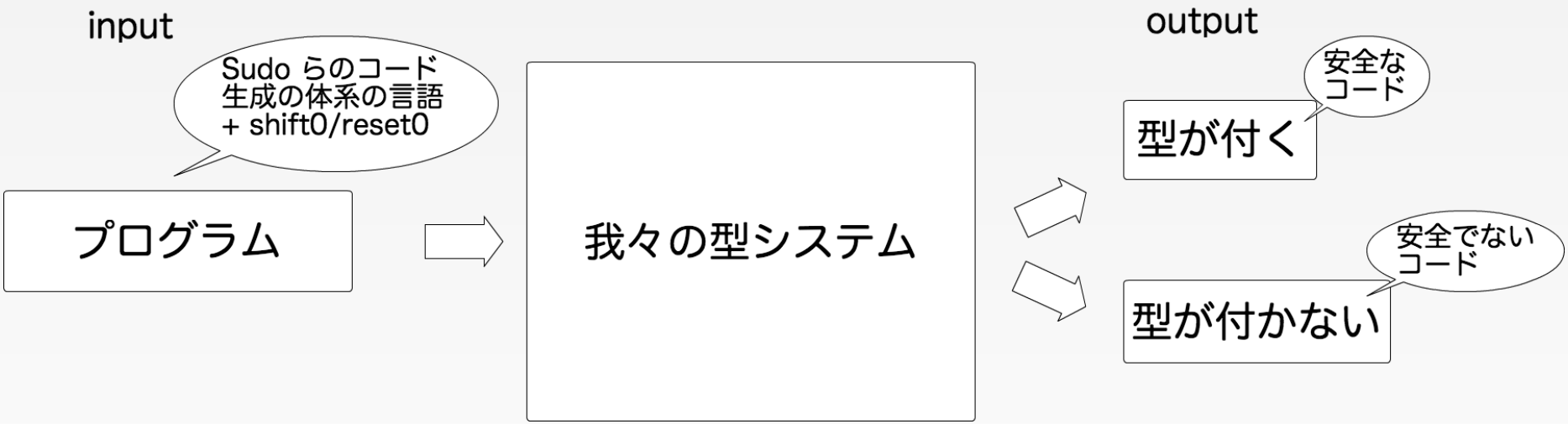
本研究: 簡潔で強力なコントロールオペレータに基づくコード生成体系の構築

- コントロールオペレータ shift0/reset0 を利用し，let 挿入などのコード生成技法を表現
- 型システムを構築して型安全性を保証

アウトライン

表現力を上げ（コードレベルでの多段階let挿入），安全性も保証するためにどうすればよいのか

本研究の手法


input
Sudo らのコード
生成の体系の言語
+ shift0/reset0
プログラム
我々の型システム
output
安全な
コード
型が付く
安全でない
コード
型が付かない

まず表現力について

コード生成器と生成されるコード

コード生成器

... **for** $x = e1$ **to** $e2$ **do**
... **for** $y = e3$ **to** $e4$ **do**
... **let** $u = t$ **in**
... **set** <a> (x, y) u

$\rightsquigarrow^*$

生成されるコード

$\langle$**let** $u' = t'$ **in**
for $x' = e1'$ **to** $e2'$ **do**
for $y' = e3'$ **to** $e4'$ **do**
$a[x', y'] \leftarrow u'\rangle$

shift0/reset0 による let 挿入

コード生成器:

$$\underline{\textbf{for}}\ x = e1\ \underline{\textbf{to}}\ e2\ \underline{\textbf{do}}$$
$$\underline{\textbf{for}}\ y = e3\ \underline{\textbf{to}}\ e4\ \underline{\textbf{do}}$$
$$\underline{\textbf{let}}\ u = t\ \underline{\textbf{in}}$$
$$(\underline{\textbf{set}}\ a\ (x, y)\ u)$$

生成コード:

shift0/reset0 による let 挿入

コード生成器:

$$\underline{\textbf{for}}\ x = e1\ \underline{\textbf{to}}\ e2\ \underline{\textbf{do}}$$
$$\textbf{reset0}\ \ \underline{\textbf{for}}\ y = e3\ \underline{\textbf{to}}\ e4\ \underline{\textbf{do}}$$
$$\textbf{shift0}\ k\ \rightarrow\ \underline{\textbf{let}}\ u = t\ \underline{\textbf{in}}$$
$$(\textbf{throw}\ k\ (\underline{\textbf{set}}\ a\ (x, y)\ u))$$

生成コード:

shift0/reset0 による let 挿入

$$\textbf{reset0}\ (E[\textbf{shift0}\ k \to e]) \to e\{k \Leftarrow E\}$$

コード生成器:

$$\begin{array}{l}
\underline{\textbf{for}}\ x = e1\ \underline{\textbf{to}}\ e2\ \underline{\textbf{do}} \\
\quad \textbf{reset0}\ \ \underline{\textbf{for}}\ y = e3\ \underline{\textbf{to}}\ e4\ \underline{\textbf{do}} \\
\quad\quad \textbf{shift0}\ k\ \to\ \underline{\textbf{let}}\ u = t\ \underline{\textbf{in}} \\
\quad\quad\quad (\textbf{throw}\ k\ (\underline{\textbf{set}}\ a\ (x, y)\ u)) \\
k \Leftarrow\ \ \underline{\textbf{for}}\ y = e3\ \underline{\textbf{to}}\ e4\ \underline{\textbf{do}}\ [\ ]
\end{array}$$

生成コード:

$$\begin{array}{l}
\langle\ \textbf{for}\ x' = e1'\ \textbf{to}\ e2'\ \textbf{do} \\
\quad \textbf{let}\ u'\ =\ t'\ \textbf{in} \\
\quad\quad \textbf{for}\ y' = e3'\ \textbf{to}\ e4'\ \textbf{do} \\
\quad\quad\quad a[x', y'] \leftarrow u'\ \rangle
\end{array}$$

shift0/reset0 による let 挿入

$$\textbf{reset0}\ (E[\textbf{shift0}\ k \to e]) \to e\{k \Leftarrow E\}$$

コード生成器: **reset0** **for** $x = e1$ **to** $e2$ **do**
for $y = e3$ **to** $e4$ **do**
shift0 k $\to$ **let** $u = t$ **in**
(**throw** k (**set** a (x, y) u))

生成コード:

shift0/reset0 による let 挿入

$$\textbf{reset0}\ (E[\textbf{shift0}\ k \to e]) \to e\{k \Leftarrow E\}$$

コード生成器: $\textbf{reset0}\ \underline{\textbf{for}}\ x = e1\ \underline{\textbf{to}}\ e2\ \underline{\textbf{do}}$

$\underline{\textbf{for}}\ y = e3\ \underline{\textbf{to}}\ e4\ \underline{\textbf{do}}$

$\textbf{shift0}\ k\ \to\ \underline{\textbf{let}}\ u = t\ \underline{\textbf{in}}$

$(\textbf{throw}\ k\ (\underline{\textbf{set}}\ a\ (x, y)\ u))$

$k \Leftarrow\ \ \underline{\textbf{for}}\ x = e1\ \underline{\textbf{to}}\ e2\ \underline{\textbf{do}}\ \ \underline{\textbf{for}}\ y = e3\ \underline{\textbf{to}}\ e4\ \underline{\textbf{do}}\ \ [\ ]$

生成コード: $\langle\ \textbf{let}\ u' = t'\ \textbf{in}$

$\textbf{for}\ x' = e1'\ \textbf{to}\ e2'\ \textbf{do}$

$\textbf{for}\ y' = e3'\ \textbf{to}\ e4'\ \textbf{do}$

$a[x', y'] \leftarrow u'\ \rangle$

shift0/reset0 による多段階 let 挿入

$$\textbf{reset0}\ (E[\textbf{shift0}\ k \to e]) \to e\{k \Leftarrow E\}$$

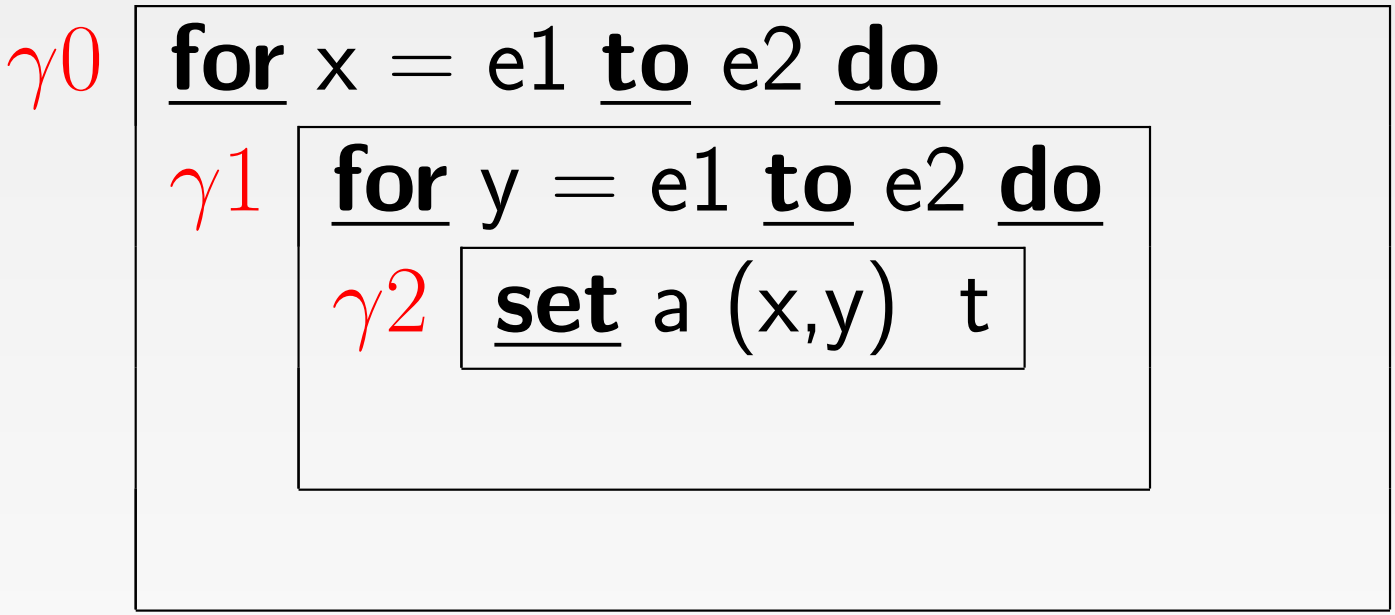
コード生成器: reset0 for $x = e1$ to $e2$ do
reset0 for $y = e3$ to $e4$ do
shift0 k_2 → shift0 k_1 → let $u = t$ in
throw k_1 (throw k_2 (set a (x, y) u))

生成コード: $\langle$ let $u' = t'$ in
for $x' = e1'$ to $e2'$ do
for $y' = e3'$ to $e4'$ do
$a[x', y'] \leftarrow u'$ $\rangle$

次に安全性

コード生成前の段階で，安全なコードかどうかを判断する

環境識別子(EC)を利用したスコープ表現 [Sudo+2014]



スコープ	使えるコード変数
$\gamma 0$	なし
$\gamma 1$	x
$\gamma 2$	x,y

環境識別子 (EC) を利用したスコープ表現 [Sudo+2014]

型システムでコード変数のスコープを表現:

$$\gamma 2 \geq \gamma 1,\ x : \langle \mathbf{int} \rangle !\gamma 1,\ y : \langle \mathbf{int} \rangle !\gamma 2 \ \vdash\ x : \langle \mathbf{float} \rangle !\gamma 2 \qquad \text{OK}$$

$$\gamma 2 \geq \gamma 1,\ x : \langle \mathbf{int} \rangle !\gamma 1,\ y : \langle \mathbf{int} \rangle !\gamma 2 \ \vdash\ y : \langle \mathbf{float} \rangle !\gamma 1 \qquad \text{NG}$$

$$\gamma 2 \geq \gamma 1,\ x : \langle \mathbf{int} \rangle !\gamma 1,\ y : \langle \mathbf{int} \rangle !\gamma 2 \ \vdash\ x \underline{+} y : \langle \mathbf{int} \rangle !\gamma 2 \qquad \text{OK}$$

コードレベルのラムダ抽象の型付け規則で，固有変数条件を利用:

$$\frac{\Gamma,\ \gamma_2 \geq \gamma_1,\ x : \langle t_1 \rangle !\gamma_2 \vdash e : \langle t_2 \rangle !\gamma_2}{\Gamma \vdash \underline{\lambda} x.e : \langle t_1 \to t_2 \rangle !\gamma_1}\ (\gamma_2 \text{ is eigen var})$$

環境識別子 (EC) を利用したスコープ表現

先行研究:

- **局所的なスコープをもつ破壊的変数をもつコード生成の体系に対する (型安全な) 型システムの構築**
  [Sudo,Kiselyov,Kameyama 2014]
- **グローバルなスコープをもつ破壊的変数への拡張**
  [Kiselyov,Kameyama,Sudo 2016]
- **コントロールオペレータには非対応**

問題点: shift0/reset0 **などのコントロールオペレータは，スコープの包含関係を逆転させてしまう．**

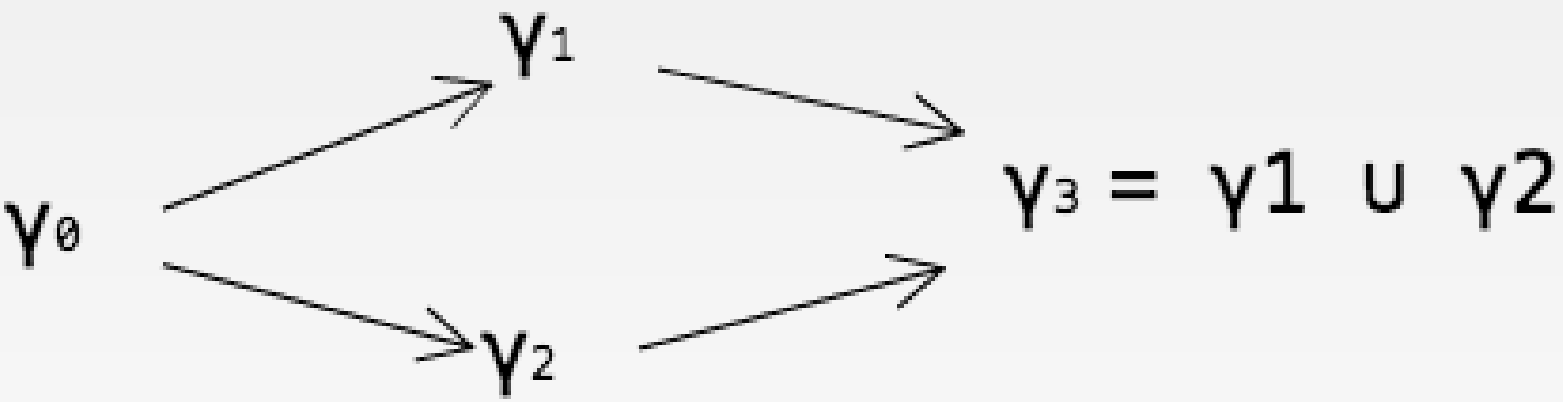
ここに，for **ループと** shift0/reset0 **の例を再掲する．**
もとのスライドの 24 **ページの絵をかく．**

本研究での解決策

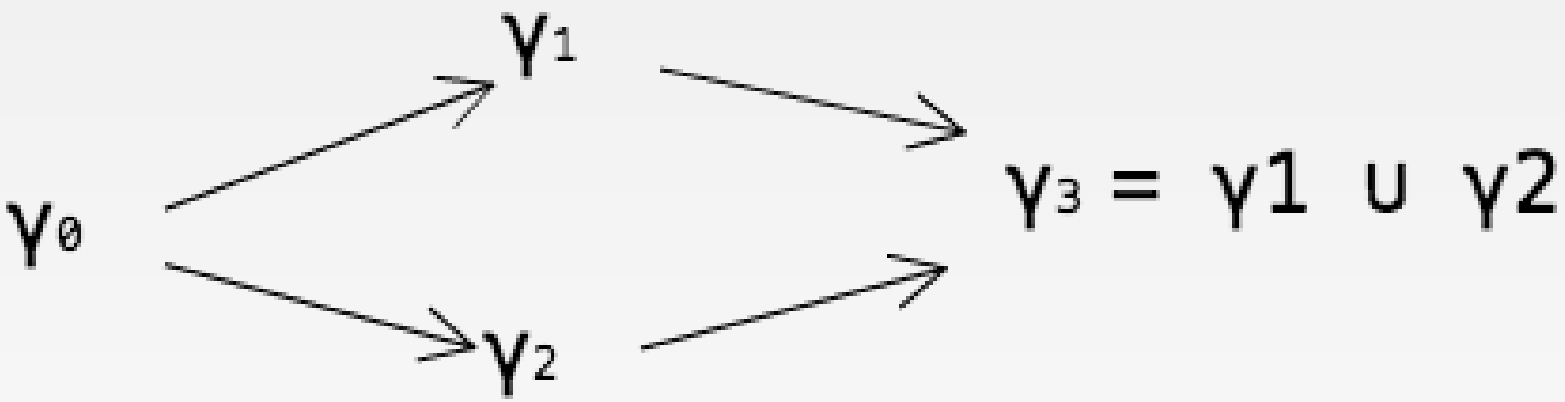
3 **つのアイディア**:

- **包含関係にない** EC
- **ジョイン演算子**
- EC **に関する多相性**

EC のジョイン

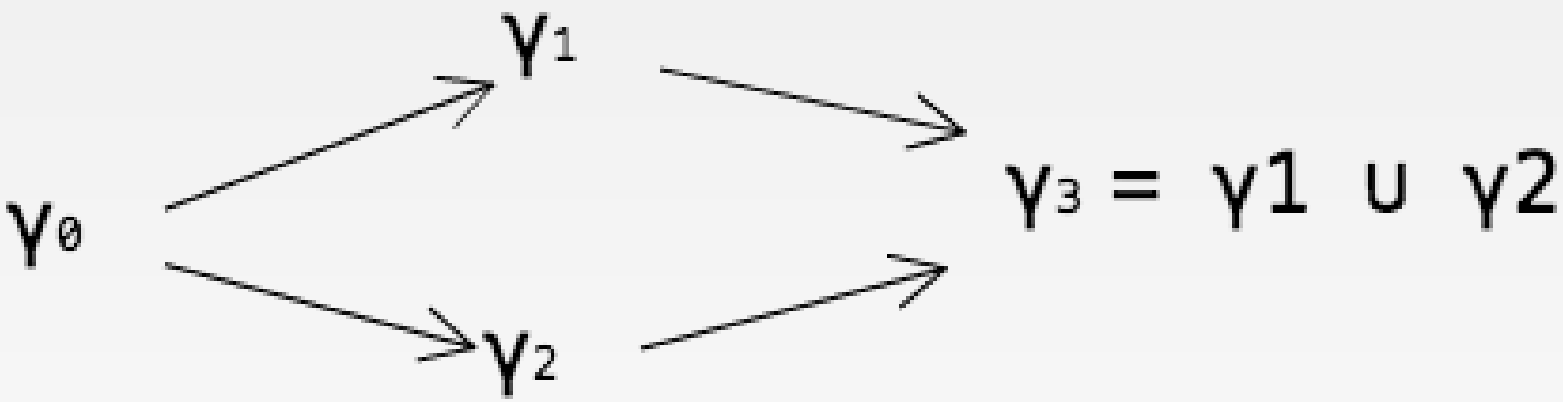


EC のジョイン



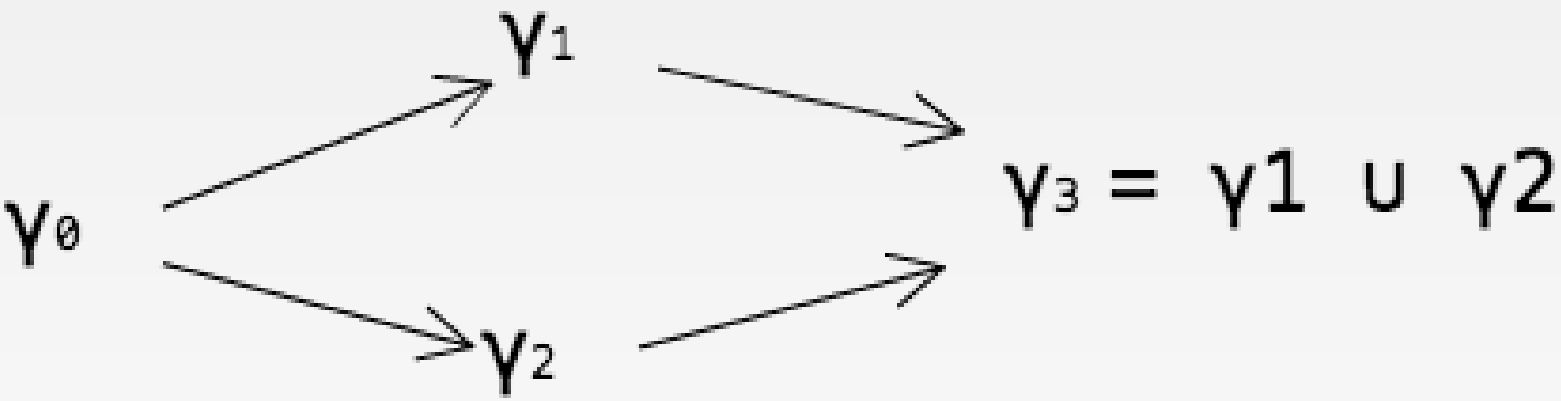
- γ_1 **のコードレベル変数は** γ_2 **では使えない**

EC のジョイン



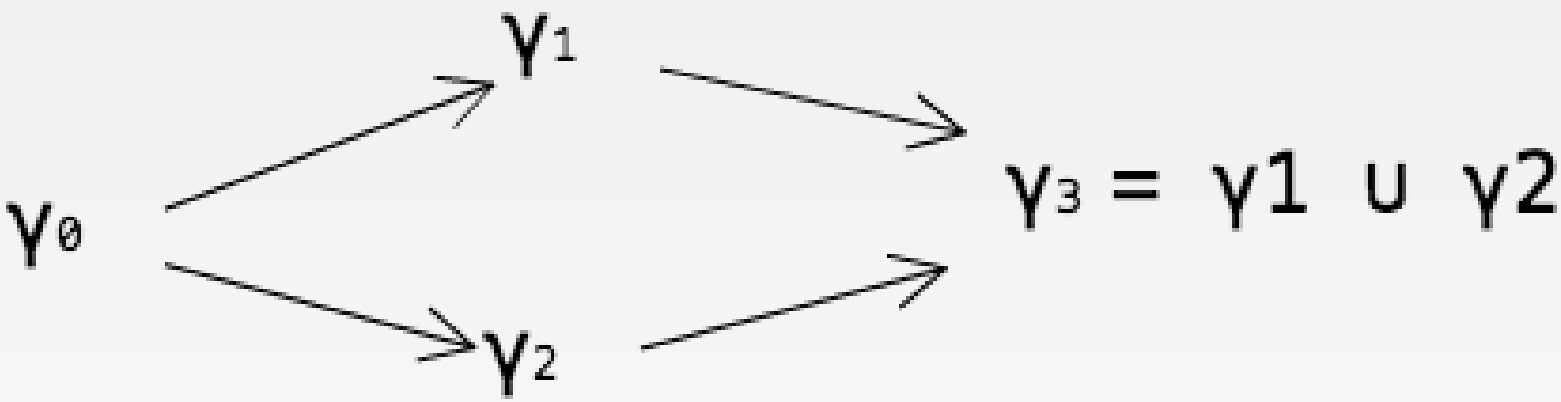
- γ_1 のコードレベル変数は γ_2 では使えない
- γ_2 のコードレベル変数は γ_1 では使えない

EC のジョイン



- γ_1 のコードレベル変数は γ_2 では使えない
- γ_2 のコードレベル変数は γ_1 では使えない
- γ_1, γ_2 のコードレベル変数は γ_3 で使える

EC のジョイン



- γ_1 **のコードレベル変数は** γ_2 **では使えない**
- γ_2 **のコードレベル変数は** γ_1 **では使えない**
- γ_1, γ_2 **のコードレベル変数は** γ_3 **で使える**

$\Rightarrow$ Sudo **らの体系に** $\cup$ **を追加**

コード生成+shift0/reset0 の型システム (の一部)

コードレベルのラムダ抽象:

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

reset0:

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

shift0:

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

throw:

$$\frac{\Gamma \vdash v : \langle t_1 \rangle^{\gamma_1 \cup \gamma_2}; \sigma \quad \Gamma \models \gamma_2 \geq \gamma_0}{\Gamma,\ k : (\langle t_1 \rangle^{\gamma_1} \Rightarrow \langle t_0 \rangle^{\gamma_0})\sigma \vdash \textbf{throw}\ k\ v : \langle t_0 \rangle^{\gamma_2}; \sigma}$$

APPENDIX

アウトライン

❸ 健全性の証明

健全性の証明 (Subject Reduction)

型安全性 (型システムの健全性; Subject Reduction **等の性質) を厳密に証明する.**

Subject Redcution Property

$\Gamma \vdash M : \tau$ **が導ければ (プログラム** M **が型検査を通れば)，**M **を計算して得られる任意の** N **に対して，**$\Gamma \vdash N : \tau$ **が導ける (**N **も型検査を通り，**M **と同じ型，同じ自由変数を持つ)**