

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

日本ソフトウェア科学会第 33 回大会

アウトライン

- ① 概要
- ② 研究の目的
- ③ 研究の内容
- ④ まとめと今後の課題

アウトライン

- ① 概要
- ② 研究の目的
- ③ 研究の内容
- ④ まとめと今後の課題

プログラムを生成するプログラミング言語 (=コード生成言語)
の安全性を保証する研究

プログラムを生成するプログラミング言語 (=コード生成言語)
の安全性を保証する研究

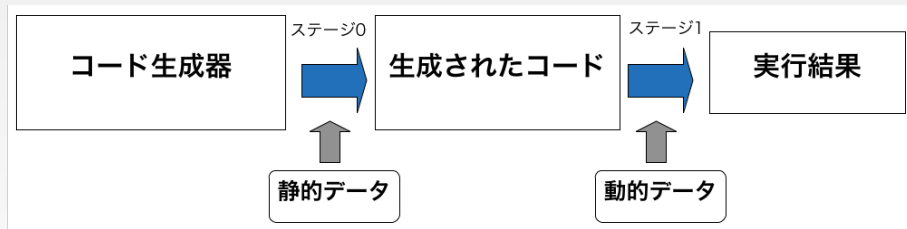
- 効率的なコードの生成
- 安全性の保証

プログラムを生成するプログラミング言語 (=コード生成言語)
の安全性を保証する研究

- 効率的なコードの生成
- 安全性の保証

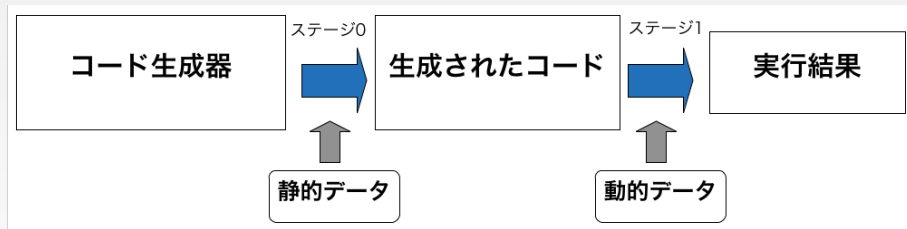
⇒ 多段階 let 挿入を効率的かつ安全に扱うための型システムを
構築

コード生成



- コード生成ステージとコード実行ステージ
- コード生成をサポートするプログラム言語 ⇒ コード生成言語

コード生成



- コード生成ステージとコード実行ステージ
- コード生成をサポートするプログラム言語 ⇒ コード生成言語
- 生成するプログラムだけでなく、生成されたプログラムも型の整合性が静的に（生成前に）保証される

コード生成言語による記述例

コード生成言語による記述例

コード生成器 \rightsquigarrow^* 生成されるコード

コード生成言語による記述例

コード生成器 \rightsquigarrow^* 生成されるコード
(int 3)

コード生成言語による記述例

コード生成器 \rightsquigarrow^* 生成されるコード

$(\underline{\text{int}}\ 3) \rightsquigarrow^* \langle 3 \rangle$

コード生成言語による記述例

コード生成器 \rightsquigarrow^* 生成されるコード

$(\underline{\text{int}}\ 3) \rightsquigarrow^* \langle 3 \rangle$

$(\underline{\text{int}}\ 3) \underline{+} (\underline{\text{int}}\ 5)$

コード生成言語による記述例

コード生成器 \rightsquigarrow^* 生成されるコード

$(\underline{\text{int}}\ 3) \rightsquigarrow^* \langle 3 \rangle$

$(\underline{\text{int}}\ 3) \underline{+} (\underline{\text{int}}\ 5) \rightsquigarrow^* \langle 3 + 5 \rangle$

コード生成言語による記述例

コード生成器 \rightsquigarrow^* 生成されるコード

$(\underline{\text{int}}\ 3) \rightsquigarrow^* \langle 3 \rangle$

$(\underline{\text{int}}\ 3) \underline{+} (\underline{\text{int}}\ 5) \rightsquigarrow^* \langle 3 + 5 \rangle$

$\underline{\lambda}x. x \underline{+} (\underline{\text{int}}\ 3)$

コード生成言語による記述例

コード生成器 \rightsquigarrow^* 生成されるコード

$(\underline{\text{int}}\ 3) \rightsquigarrow^* \langle 3 \rangle$

$(\underline{\text{int}}\ 3) \underline{+} (\underline{\text{int}}\ 5) \rightsquigarrow^* \langle 3 + 5 \rangle$

$\underline{\lambda}x. x \underline{+} (\underline{\text{int}}\ 3) \rightsquigarrow^* \langle \lambda x'. x' + 3 \rangle$

コード生成言語による記述例

コード生成器 \rightsquigarrow^* 生成されるコード

$(\underline{\text{int}}\ 3) \rightsquigarrow^* \langle 3 \rangle$

$(\underline{\text{int}}\ 3) \underline{+} (\underline{\text{int}}\ 5) \rightsquigarrow^* \langle 3 + 5 \rangle$

$\underline{\lambda}x. x \underline{+} (\underline{\text{int}}\ 3) \rightsquigarrow^* \langle \lambda x'. x' + 3 \rangle$

$\underline{\text{for}}\ x = \dots \underline{\text{to}} \dots \underline{\text{do}} \dots$

コード生成言語による記述例

コード生成器 \rightsquigarrow^* 生成されるコード

$(\underline{\text{int}}\ 3) \rightsquigarrow^* \langle 3 \rangle$

$(\underline{\text{int}}\ 3) \underline{+} (\underline{\text{int}}\ 5) \rightsquigarrow^* \langle 3 + 5 \rangle$

$\underline{\lambda}x. x \underline{+} (\underline{\text{int}}\ 3) \rightsquigarrow^* \langle \lambda x'. x' + 3 \rangle$

$\underline{\text{for}}\ x = \dots \underline{\text{to}} \dots \underline{\text{do}} \dots \rightsquigarrow^* \langle \text{for } x' = \dots \text{ to } \dots \text{ do } \dots \rangle$

コード生成言語による記述例

コード生成器 \rightsquigarrow^* 生成されるコード

$(\underline{\text{int}}\ 3) \rightsquigarrow^* \langle 3 \rangle$

$(\underline{\text{int}}\ 3) \underline{+} (\underline{\text{int}}\ 5) \rightsquigarrow^* \langle 3 + 5 \rangle$

$\underline{\lambda}x. x \underline{+} (\underline{\text{int}}\ 3) \rightsquigarrow^* \langle \lambda x'. x' + 3 \rangle$

$\underline{\text{for}}\ x = \dots \underline{\text{to}} \dots \underline{\text{do}} \dots \rightsquigarrow^* \langle \text{for } x' = \dots \text{ to } \dots \text{ do } \dots \rangle$

コードコンビネータ

- 下線付きの演算子
- コードを引数にとり，コードを返す

普通の power 関数

```
power =  $\lambda x.$  fix  $\lambda f.$   $\lambda n.$   
      if  $n = 0$  then 1  
      else  $x \times (f (n - 1))$ 
```

power 関数のコード生成器

gen_power: power コード生成器

$$\begin{aligned} \text{gen_power} = & \ \underline{\lambda}x.\mathbf{fix} \ \lambda f.\lambda n. \\ & \ \mathbf{if} \ n = 0 \ \mathbf{then} \ (\underline{\mathbf{int}} \ 1) \\ & \ \mathbf{else} \ x \ \underline{\times} \ (f \ (n - 1)) \end{aligned}$$

power 関数のコード生成器

gen_power: power コード生成器

$$\begin{aligned} \text{gen_power} = & \ \underline{\lambda}x.\mathbf{fix} \ \lambda f.\lambda n. \\ & \quad \mathbf{if} \ n = 0 \ \mathbf{then} \ (\underline{\mathbf{int}} \ 1) \\ & \quad \mathbf{else} \ x \ \underline{\times} \ (f \ (n - 1)) \end{aligned}$$

$n = 5$ に特化したコード生成:

power 関数のコード生成器

gen_power: power コード生成器

$$\begin{aligned} \text{gen_power} = & \ \underline{\lambda}x.\mathbf{fix} \ \lambda f.\lambda n. \\ & \quad \mathbf{if} \ n = 0 \ \mathbf{then} \ (\underline{\mathbf{int}} \ 1) \\ & \quad \mathbf{else} \ x \ \underline{\times} \ (f \ (n - 1)) \end{aligned}$$

$n = 5$ に特化したコード生成:

$$\underline{\lambda}x. \text{gen_power } x \ 5 \rightsquigarrow^* \langle \underline{\lambda}x'. x' \times x' \times x' \times x' \times x' \times 1 \rangle$$

power 関数のコード生成器

gen_power: power コード生成器

$$\begin{aligned} \text{gen_power} = & \ \underline{\lambda}x.\mathbf{fix} \ \lambda f.\lambda n. \\ & \ \mathbf{if} \ n = 0 \ \mathbf{then} \ (\underline{\mathbf{int}} \ 1) \\ & \ \mathbf{else} \ x \ \underline{\times} \ (f \ (n - 1)) \end{aligned}$$

$n = 5$ に特化したコード生成:

$$\underline{\lambda}x. \text{gen_power } x \ 5 \rightsquigarrow^* \langle \lambda x'. x' \times x' \times x' \times x' \times x' \times 1 \rangle$$

gen_power 関数によって生成されたコードは power 関数より高速

二重 for ループのコード生成器

コード生成器

```
for  $x = (\underline{\text{int}}\ 0)\ \underline{\text{to}}\ (\underline{\text{int}}\ n)\ \underline{\text{do}}$   
  for  $y = (\underline{\text{int}}\ 0)\ \underline{\text{to}}\ (\underline{\text{int}}\ m)\ \underline{\text{do}}$   
    set  $a\ (x, y)$  complex calculation
```

二重 for ループのコード生成器

コード生成器

```
for  $x = (\underline{\text{int}}\ 0) \text{ to } (\underline{\text{int}}\ n) \text{ do}$   
  for  $y = (\underline{\text{int}}\ 0) \text{ to } (\underline{\text{int}}\ m) \text{ do}$   
    set  $a(x, y)$  complex calculation
```



生成されるコード

```
< for  $x' = 0 \text{ to } n$  do  
  for  $y' = 0 \text{ to } m$  do  
     $a[x', y'] \leftarrow$  complex calculation >
```

多段階 let 挿入の例

生成されるコード

< **for** $x' = 0$ **to** n **do**

for $y' = 0$ **to** m **do**

$a[x', y'] \leftarrow$ complex calculation

$b[x', y'] \leftarrow$ complex calculation >

多段階 let 挿入の例

生成されるコード

```
< let  $w = \text{complex calculation}$  in  
  for  $x' = 0$  to  $n$  do  
    let  $u = \text{complex calculation}$  in  
    for  $y' = 0$  to  $m$  do  
       $a[x', y'] \leftarrow u$   
       $b[x', y'] \leftarrow w$       >
```

多段階 let 挿入の例

生成されるコード

```
< let  $w = \text{complex calculation}$  in  
  for  $x' = 0$  to  $n$  do  
    let  $u = \text{complex calculation}$  in  
      for  $y' = 0$  to  $m$  do  
         $a[x', y'] \leftarrow u$   
         $b[x', y'] \leftarrow w$       >
```

多段階 let 挿入

- 入れ子になった for ループなどを飛び越えた**複数のコード移動**を許す仕組み
- ループ不変式の移動等によって、**効率的なコード生成**に必要なプログラミング技法

危険な例

危険なコード生成の例

生成される**危険な**コード

< **for** $x' = 0$ **to** n **do**

for $y' = 0$ **to** m **do**

$a[x', y'] \leftarrow$ **complex calculation**

$b[x', y'] \leftarrow$ **complex calculation**>

危険なコード生成の例

生成される**危険な**コード

< let w = complex calculation in

for $x' = 0$ to n do

let u = complex calculation in

for $y' = 0$ to m do

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

$b[x', y'] \leftarrow w$

>

危険なコード生成の例

生成される危険なコード

```
< let  $w = \text{complex calculation}$  in —  $w$  が  $x$  にも  $y$  にも依存する式
  for  $x' = 0$  to  $n$  do
    let  $u = \text{complex calculation}$  in —  $u$  が  $y$  に依存する式
    for  $y' = 0$  to  $m$  do
       $a[x', y'] \leftarrow u$ 
       $b[x', y'] \leftarrow w$ 
  >
```

危険なコード生成の例

生成される危険なコード

```
< let  $w = \text{complex calculation}$  in —  $w$  が  $x$  にも  $y$  にも依存する式
  for  $x' = 0$  to  $n$  do
    let  $u = \text{complex calculation}$  in —  $u$  が  $y$  に依存する式
    for  $y' = 0$  to  $m$  do
       $a[x', y'] \leftarrow u$ 
       $b[x', y'] \leftarrow w$ 
  >
```

complex calculation によって挿入できる場所が異なる

- 多段階 let 挿入が可能となっても、安全に挿入できる場所とそうでない場所がある
- 安全に let 挿入を行うためにどうすればよいかを考える必要がある

コード生成の利点と課題

利点

- 「保守性・再利用性の高さ」と「実行性能の高さ」の両立

コード生成の利点と課題

利点

- 「保守性・再利用性の高さ」と「実行性能の高さ」の両立

課題

- パラメータに応じて、非常に多数のコードが生成される
 - 生成したコードのデバッグが容易ではない
- ⇒ コード生成の前に安全性を保証したい

アウトライン

- ① 概要
- ② 研究の目的
- ③ 研究の内容
- ④ まとめと今後の課題

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

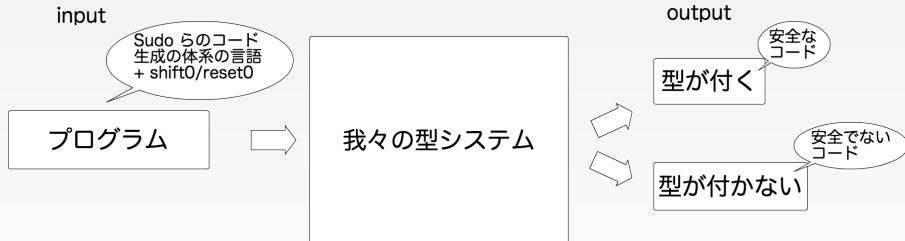
- コントロールオペレータ `shift0/reset0` を利用し, `let` 挿入などのコード生成技法を表現
- 型システムを構築して型安全性を保証

アウトライン

- ① 概要
- ② 研究の目的
- ③ 研究の内容**
- ④ まとめと今後の課題

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



まず表現力について

let 挿入の実現方法

コード生成器

```
... for  $x = e1$  to  $e2$  do  
  ... for  $y = e3$  to  $e4$  do  
    set  $\langle a \rangle (x, y)$  ... comp calc
```

\rightsquigarrow^*

生成されるコード

```
<let  $u' = \text{comp calc}'$  in  
  for  $x' = e1'$  to  $e2'$  do  
    for  $y' = e3'$  to  $e4'$  do  
       $a[x', y'] \leftarrow u'$ >
```

let 挿入の実現方法

コード生成器

```
... for  $x = e1$  to  $e2$  do  
... for  $y = e3$  to  $e4$  do  
    set  $\langle a \rangle (x, y)$  ... comp calc
```

\rightsquigarrow^*

生成されるコード

```
<let  $u' = \text{comp calc}'$  in  
  for  $x' = e1'$  to  $e2'$  do  
    for  $y' = e3'$  to  $e4'$  do  
       $a[x', y'] \leftarrow u'$ >
```

shift0/reset0 の導入

... のところに後に説明する **shift0/reset0** というコントロールオペレータを用いることで、多段階 let 挿入を行う

コード生成器:

```
for  $x = e1$  to  $e2$  do  
  for  $y = e3$  to  $e4$  do  
    set  $a(x, y)$   $cc$ 
```

shift0/reset0 による let 挿入

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

shift0/reset0 による let 挿入

reset0 ($E[\text{shift0 } k \rightarrow e]$) $\rightarrow e\{k \Leftarrow E\}$

コード生成器: **reset0** for $x = e1$ to $e2$ do

for $y = e3$ to $e4$ do

set $a(x, y)$ **shift0** $k \rightarrow$ let $u = cc$ in **throw** k u

$k \Leftarrow$ for $x = e1$ to $e2$ do

for $y = e3$ to $e4$ do

set $a(x, y)$ []

shift0/reset0 による let 挿入

reset0 ($E[\text{shift0 } k \rightarrow e] \rightarrow e\{k \Leftarrow E\}$)

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

生成コード: < **let** $u' = cc'$ **in**
 for $x' = e1'$ **to** $e2'$ **do**
 for $y' = e3'$ **to** $e4'$ **do**
 $a[x', y'] \leftarrow u'$ >

shift0/reset0 による多段階 let 挿入

reset0 ($E[\text{shift0 } k \rightarrow e]$) $\rightarrow e\{k \Leftarrow E\}$

コード生成器: **reset0** for $x = e1$ to $e2$ do

reset0 for $y = e3$ to $e4$ do

set $a(x, y)$ **shift0** $k_1 \rightarrow$ let $u = cc1$ in **throw** k_1 u

set $b(x, y)$ **shift0** $k_2 \rightarrow$ let $w = cc2$ in **throw** k_2 w

shift0/reset0 による多段階 let 挿入

reset0 ($E[\text{shift0 } k \rightarrow e] \rightarrow e\{k \Leftarrow E\}$)

コード生成器: **reset0** for $x = e1$ to $e2$ do

reset0 for $y = e3$ to $e4$ do

set $a(x, y)$ **shift0** $k_1 \rightarrow$ let $u = cc1$ in **throw** k_1 u

set $b(x, y)$ **shift0** $k_2 \rightarrow$ let $w = cc2$ in **throw** k_2 w

生成コード: < **let** $w' = cc2'$ **in**

for $x' = e1'$ **to** $e2'$ **do**

let $u' = cc1'$ **in**

for $y' = e3'$ **to** $e4'$ **do**

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

$b[x', y'] \leftarrow w' >$

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

γ_0 **for** x = e1 **to** e2 **do**
 γ_1 **for** y = e3 **to** e4 **do**
 γ_2 **set** a (x,y) complex calc

スコープ	使えるコード変数
γ_0	なし
γ_1	x
γ_2	x,y

型システムでコード変数のスコープを表現:

$$\Gamma = \gamma_2 \geq \gamma_1, \ x : \langle \mathbf{int} \rangle^{\gamma_1}, \ y : \langle \mathbf{int} \rangle^{\gamma_2}$$

型システムでコード変数のスコープを表現:

$$\Gamma = \gamma_2 \geq \gamma_1, \ x : \langle \mathbf{int} \rangle^{\gamma_1}, \ y : \langle \mathbf{int} \rangle^{\gamma_2}$$

$$\Gamma \vdash x : \langle \mathbf{int} \rangle^{\gamma_2} \text{ OK}$$

型システムでコード変数のスコープを表現:

$$\Gamma = \gamma_2 \geq \gamma_1, \ x : \langle \mathbf{int} \rangle^{\gamma_1}, \ y : \langle \mathbf{int} \rangle^{\gamma_2}$$

$$\Gamma \vdash x : \langle \mathbf{int} \rangle^{\gamma_2} \text{ OK}$$

$$\Gamma \vdash y : \langle \mathbf{int} \rangle^{\gamma_1} \text{ NG}$$

型システムでコード変数のスコープを表現:

$$\Gamma = \gamma_2 \geq \gamma_1, x : \langle \text{int} \rangle^{\gamma_1}, y : \langle \text{int} \rangle^{\gamma_2}$$

$$\Gamma \vdash x : \langle \text{int} \rangle^{\gamma_2} \text{ OK}$$

$$\Gamma \vdash y : \langle \text{int} \rangle^{\gamma_1} \text{ NG}$$

$$\Gamma \vdash x \underline{+} y : \langle \text{int} \rangle^{\gamma_1} \text{ NG}$$

型システムでコード変数のスコープを表現:

$$\Gamma = \gamma_2 \geq \gamma_1, \ x : \langle \text{int} \rangle^{\gamma_1}, \ y : \langle \text{int} \rangle^{\gamma_2}$$

$$\Gamma \vdash x : \langle \text{int} \rangle^{\gamma_2} \quad \text{OK}$$

$$\Gamma \vdash y : \langle \text{int} \rangle^{\gamma_1} \quad \text{NG}$$

$$\Gamma \vdash x \underline{+} y : \langle \text{int} \rangle^{\gamma_1} \quad \text{NG}$$

$$\Gamma \vdash x \underline{+} y : \langle \text{int} \rangle^{\gamma_2} \quad \text{OK}$$

型システムでコード変数のスコープを表現:

$$\Gamma = \gamma_2 \geq \gamma_1, x : \langle \text{int} \rangle^{\gamma_1}, y : \langle \text{int} \rangle^{\gamma_2}$$

$$\Gamma \vdash x : \langle \text{int} \rangle^{\gamma_2} \text{ OK}$$

$$\Gamma \vdash y : \langle \text{int} \rangle^{\gamma_1} \text{ NG}$$

$$\Gamma \vdash x + y : \langle \text{int} \rangle^{\gamma_1} \text{ NG}$$

$$\Gamma \vdash x + y : \langle \text{int} \rangle^{\gamma_2} \text{ OK}$$

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:

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

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

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。

環境識別子 (EC) の問題点

コード生成前・後でスコープの包含関係が逆転

コード生成器：

The diagram illustrates nested code blocks with labels γ_0 , γ_1 , γ_2 , and γ_3 indicating the scope level. The blocks are nested, with γ_3 being the innermost and γ_0 the outermost.

```
for  $x = e1$  to  $e2$  do  
 $\gamma_0$  reset0 for  $y = e3$  to  $e4$  do  
 $\gamma_1$  shift0  $k \rightarrow$  let  $u = cc$  in  
 $\gamma_2$  throw  $k$   
 $\gamma_3$  set  $a(x, y)$   $u$ 
```


コード生成前・後でスコープの包含関係が逆転

コード生成器：

for $x = e1$ to $e2$ do

γ_0

reset0 for $y = e3$ to $e4$ do

γ_1

shift0 $k \rightarrow$ let $u = cc$ in

γ_2

throw k

γ_3

set $a(x, y) u$

生成コード：

for $x' = e1'$ to $e2'$ do

γ_0

let $u' = cc'$ in

γ_2

for $y' = e3'$ to $e4'$ do

γ_1

γ_3

$a[(x', y')] \leftarrow u'$

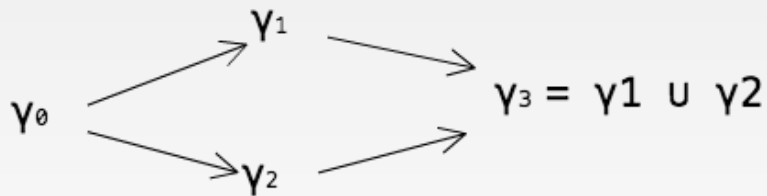
コード生成前・後でスコープの包含関係が逆転

コード生成器 :	スコープ	使えるコード変数
	γ_0	x
	γ_1	x, y
	γ_2	x, y, u
	γ_3	x, y, u

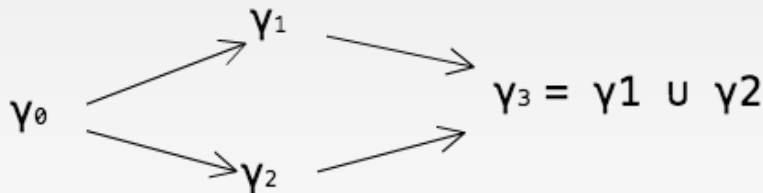
生成コード :	スコープ	使えるコード変数
	γ_0	x
	γ_1	x, u
	γ_2	x, y, u
	γ_3	x, y, u

- 生成前と生成後の γ_1 と γ_2 の間には順序関係がない

本研究の解決策 EC のジョイン

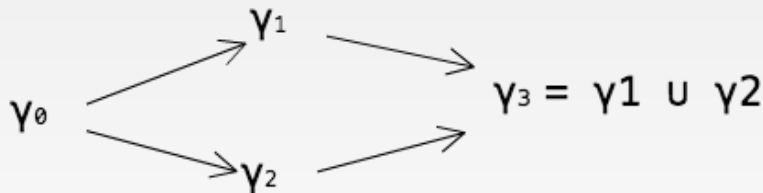


本研究の解決策 EC のジョイン



- 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

本研究の解決策 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
- ⇒ 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)

reset0:

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

shift0:

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

throw:

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

型が付く例/付かない例

コード生成器

```
e = reset0 for i = 0 to n do  
    reset0 for j = 0 to m do  
        shift0 k2 → shift0 k1 → let y = t in  
            k1 (k2 (set <a> (i, j) b[i] + y))
```

型が付く例/付かない例

コード生成器

```
e = reset0 for i = 0 to n do  
    reset0 for j = 0 to m do  
        shift0 k2 → shift0 k1 → let y = t in  
            k1 (k2 (set <a> (i, j) b[i] + y))
```

生成されるコード



```
e  $\rightsquigarrow^*$  let y = a[i][j] in  
    for i = 0 to n do  
        for j = 0 to m do  
            a[i, j]  $\leftarrow$  b[i] + y
```

```
e  $\rightsquigarrow^*$  let y = 7 in  
    for i = 0 to n do  
        for j = 0 to m do  
            a[i, j]  $\leftarrow$  b[i] + y
```


型付けの例 (1)

$e1 = \text{reset0} \ (\underline{\text{for}} \ x = e1 \ \underline{\text{to}} \ e2 \ \underline{\text{do}}$
 $\quad \text{shift0} \ k \rightarrow \underline{\text{let}} \ u = \boxed{} \ \underline{\text{in}} \ \text{throw} \ k \ u)$

$$\begin{array}{c}
 \dfrac{\dfrac{\Gamma2 \vdash u : \langle t \rangle^{\gamma1} \cup \gamma2}{\Gamma2 \vdash \text{throw } k \ u : \langle t \rangle^{\gamma2}; \epsilon} \quad \dfrac{\vdots}{\Gamma1 \vdash \boxed{} : \langle t \rangle^{\gamma0}; \epsilon}}{\Gamma1 \vdash \underline{\text{let}} \ u = \dots : \langle t \rangle^{\gamma0}; \epsilon} \\
 \dfrac{\gamma1 \geq \gamma0, \ x : \langle t \rangle^{\gamma1} \vdash \text{shift0} \ k \rightarrow \dots : \langle t \rangle^{\gamma1}; \langle t \rangle^{\gamma0}}{\vdash \underline{\text{for}} \ x = \dots : \langle t \rangle^{\gamma0}; \langle t \rangle^{\gamma0}} \\
 \hline
 \vdash e1 : \langle t \rangle^{\gamma0}; \epsilon
 \end{array}$$

$\Gamma1 = \gamma1 \geq \gamma0, \ x : \langle t \rangle^{\gamma1}, \ k : (\langle t \rangle^{\gamma1} \Rightarrow \langle t \rangle^{\gamma0})$

$\Gamma2 = \Gamma1, \ \gamma2 \geq \gamma0, \ u : \langle t \rangle^{\gamma2}$

型付けの例 (1)

$e1 = \text{reset0 } (\text{for } x = e1 \text{ to } e2 \text{ do}$
 $\text{shift0 } k \rightarrow \text{let } u = \boxed{\text{int3}} \text{ in throw } k \text{ } u)$

$$\begin{array}{c}
 \frac{\Gamma 2 \vdash u : \langle t \rangle^{\gamma_1} \cup \gamma_2}{\Gamma 2 \vdash \text{throw } k \text{ } u : \langle t \rangle^{\gamma_2}; \epsilon} \quad \vdots \quad \Gamma 1 \vdash \boxed{\text{int3}} : \langle t \rangle^{\gamma_0}; \epsilon \\
 \hline
 \Gamma 1 \vdash \text{let } u = \dots : \langle t \rangle^{\gamma_0}; \epsilon \\
 \hline
 \gamma_1 \geq \gamma_0, x : \langle t \rangle^{\gamma_1} \vdash \text{shift0 } k \rightarrow \dots : \langle t \rangle^{\gamma_1}; \langle t \rangle^{\gamma_0} \\
 \hline
 \vdash \text{for } x = \dots : \langle t \rangle^{\gamma_0}; \langle t \rangle^{\gamma_0} \\
 \hline
 \vdash e1 : \langle t \rangle^{\gamma_0}; \epsilon
 \end{array}$$

$$\Gamma 1 = \gamma_1 \geq \gamma_0, x : \langle t \rangle^{\gamma_1}, k : (\langle t \rangle^{\gamma_1} \Rightarrow \langle t \rangle^{\gamma_0})$$

$$\Gamma 2 = \Gamma 1, \gamma_2 \geq \gamma_0, u : \langle t \rangle^{\gamma_2}$$

型付けの例 (1)

$e1 = \text{reset0} \ (\underline{\text{for}} \ x = e1 \ \underline{\text{to}} \ e2 \ \underline{\text{do}} \ \underline{\text{shift0}} \ k \rightarrow \underline{\text{let}} \ u = \boxed{\times \underline{+} \ (\underline{\text{int3}})} \ \underline{\text{in}} \ \text{throw } k \ u)$

$$\begin{array}{c}
 \frac{\Gamma 2 \vdash u : \langle t \rangle^{\gamma_1} \cup \gamma_2}{\Gamma 2 \vdash \text{throw } k \ u : \langle t \rangle^{\gamma_2}; \epsilon} \quad \vdots \quad \Gamma 1 \vdash \boxed{\times \underline{+} \ (\underline{\text{int3}})} : \langle t \rangle^{\gamma_0}; \epsilon \\
 \hline
 \Gamma 1 \vdash \underline{\text{let}} \ u = \dots : \langle t \rangle^{\gamma_0}; \epsilon \\
 \hline
 \frac{\gamma_1 \geq \gamma_0, \ x : \langle t \rangle^{\gamma_1} \vdash \underline{\text{shift0}} \ k \rightarrow \dots : \langle t \rangle^{\gamma_1}; \langle t \rangle^{\gamma_0}}{\vdash \underline{\text{for}} \ x = \dots : \langle t \rangle^{\gamma_0}; \langle t \rangle^{\gamma_0}} \\
 \hline
 \vdash e1 : \langle t \rangle^{\gamma_0}; \epsilon
 \end{array}$$

$\Gamma 1 = \gamma_1 \geq \gamma_0, \ x : \langle t \rangle^{\gamma_1}, \ k : (\langle t \rangle^{\gamma_1} \Rightarrow \langle t \rangle^{\gamma_0})$

$\Gamma 2 = \Gamma 1, \ \gamma_2 \geq \gamma_0, \ u : \langle t \rangle^{\gamma_2}$

型付けの例 (2)

$e1 =$ **reset0** for $i = 0$ to n do
 reset0 for $j = 0$ to m do
 shift0 $k_2 \rightarrow$ **shift0** $k_1 \rightarrow$ let $y = t$ in
 k_1 (k_2 (set $\langle a \rangle (i, j)$ $b[i] + y$))

アウトライン

- ① 概要
- ② 研究の目的
- ③ 研究の内容
- ④ **まとめと今後の課題**

まとめと今後の課題

まとめ

- コード生成言語の型システムに `shift0/reset0` を組み込んだ型システムの設計を完成させた.
- 安全なコードの場合に型が付くこと, 安全でないコードの場合には型が付かないように意図通りに型システムが設計できていることをみた

今後の課題

- 設計した型システムの健全性の証明 (Subject reduction 等)
- 型推論アルゴリズムの開発
- 言語の拡張
 - グローバルな参照 (OCaml の `let ref`)
 - 生成したコードの実行 (MetaOCaml の `run`)

APPENDIX

⑤ 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)