多段階 let 挿入を行うコード生成言語の設計

大石純平

筑波大学 大学院
プログラム論理研究室

2016/7/12

アウトライン

アウトライン

概要

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

概要

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

概要

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

概要

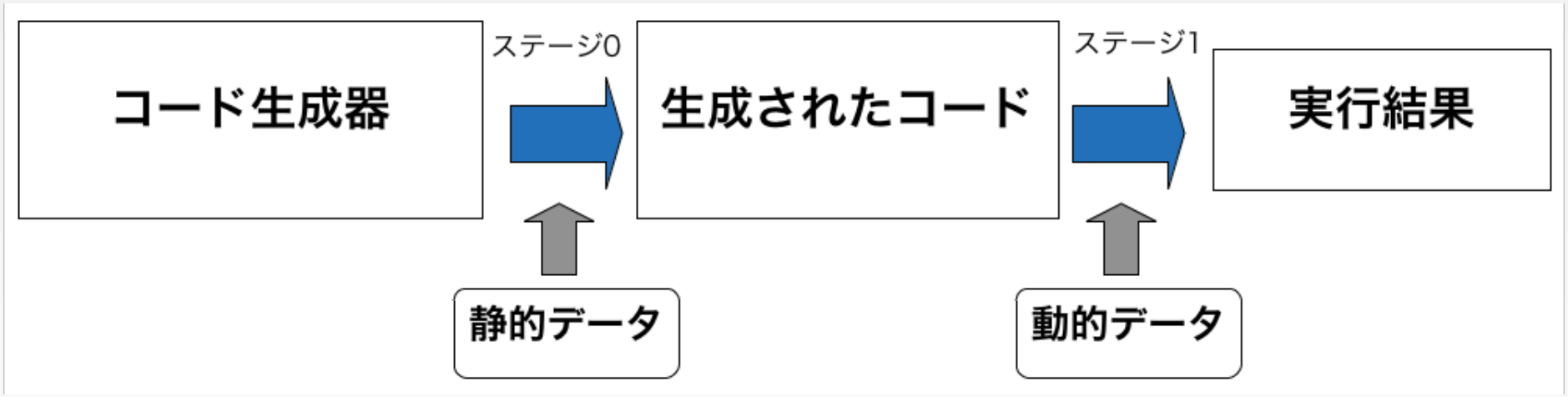
プログラムを生成するプログラミング言語 (=**コード生成言語**)
の安全性を保証する研究

- 効率的なコードの生成
- 安全性の保証

⇒ **多段階** let **挿入**を安全に扱うための型システムを構築

アウトライン

段階的計算 (Staged Computation)



- コード生成ステージとコード実行ステージ

⇒ 段階的計算をサポートするプログラム言語 ⇒ コード生成言語

power 関数のコード化

$$
\begin{aligned}
\text{power}\ x\ \ n\ \ &= x && \text{if}\ \ n = 1 \\
&\quad x * \text{power}\ x\ (n-1) && \text{if}\ \ n > 1
\end{aligned}
$$

power 関数のコード化

$$\begin{aligned} \text{power } x \;\; n \;\; &= \;\; x && \text{if} \;\; n = 1 \\ &\quad\; x * \text{power } x \; (n-1) && \text{if} \;\; n > 1 \end{aligned}$$

$n = 8$ に特化したコードの生成を行う

$$\text{gen\_power } x \;\; 8 \;\; = \;\; x \;*\; x \;*\; x \;*\; x \;*\; x \;*\; x \;*\; x \;*\; x$$

power 関数のコード化

$$\begin{aligned} \text{power } x \;\; n \;\; &= \;\; x && \text{if} \;\; n = 1 \\ & \quad\;\; x * \text{power } x \; (n-1) && \text{if} \;\; n > 1 \end{aligned}$$

$n = 8$ **に特化したコードの生成を行う**

$$\text{gen\_power } x \;\; 8 \;\; = \;\; x \;*\; x \;*\; x \;*\; x \;*\; x \;*\; x \;*\; x \;*\; x$$

gen_power $x \;\; 8$ **は** power $x \;\; 8$ **より高速**

- **関数呼び出しがない**
- **条件式がない**

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

従来研究

- コード生成プログラムが，安全なコードのみを生成する事を静的に保証
- 安全なコード: 構文，型，変数束縛が正しいプログラム

従来研究

- コード生成プログラムが，安全なコードのみを生成する事を静的に保証
- 安全なコード: 構文，型，変数束縛が正しいプログラム

しかし多段階 let 挿入等を実現する計算エフェクトを含む場合のコード生成の安全性保証は研究途上

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コード移動を許す仕組み
- ループ不変式の移動によって，効率的なコード生成に必要なプログラミング技法

多段階 let 挿入，let 挿入の例

$$
\begin{array}{l}
\textbf{for } i = 0 \textbf{ to } n \textbf{ in} \\
\quad \textbf{for } j = 0 \textbf{ to } m \textbf{ in} \\
\quad\quad \textbf{let } y \ = \ t \textbf{ in} \\
\quad\quad\quad a[i][j] = b[i] + y
\end{array}
$$

多段階 let 挿入，let 挿入の例

for $i = 0$ **to** n **in**
for $j = 0$ **to** m **in**
let $y = t$ **in**
$a[i][j] = b[i] + y$

多段階 let 挿入

$\Downarrow$

let $y = t$ **in** — t が i にも j にも依存しない式
for $i = 0$ **to** n **in**
for $j = 0$ **to** m **in**
$a[i][j] = b[i] + y$

多段階 let 挿入，let 挿入の例

for $i = 0$ **to** n **in**
for $j = 0$ **to** m **in**
let $y = t$ **in**
$a[i][j] = b[i] + y$

普通の let 挿入

$\Downarrow$

for $i = 0$ **to** n **in**
let $y = t$ **in**
for $j = 0$ **to** m **in**
$a[i][j] = b[i] + y$

— t が i にのみ依存し j には依存しない式

多段階 let 挿入，let 挿入の例

for $i = 0$ **to** n **in**
for $j = 0$ **to** m **in**
let $y = t$ **in**
$a[i][j] = b[i] + y$

多段階 let 挿入でも let 挿入でもない

$\Downarrow$

for $i = 0$ **to** n **in**
for $j = 0$ **to** m **in**
let $y = t$ **in**
$a[i][j] = b[i] + y$

— t が i,j に依存した式

コントロールオペレータ

プログラミング言語におけるプログラムを制御するプリミティブ

- exception (**例外**): C++, Java, ML
- call/cc (**第一級継続**): Scheme, SML/NJ
- shift/reset (**限定継続**): Racket, Scala, OCaml
  - 1989 **年以降多数研究がある**
  - **コード生成における** let **挿入が実現可能**
- shift0/reset0
  - 2011 **年以降研究が活発化.**
  - **コード生成における多段階** let **挿入が可能**

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

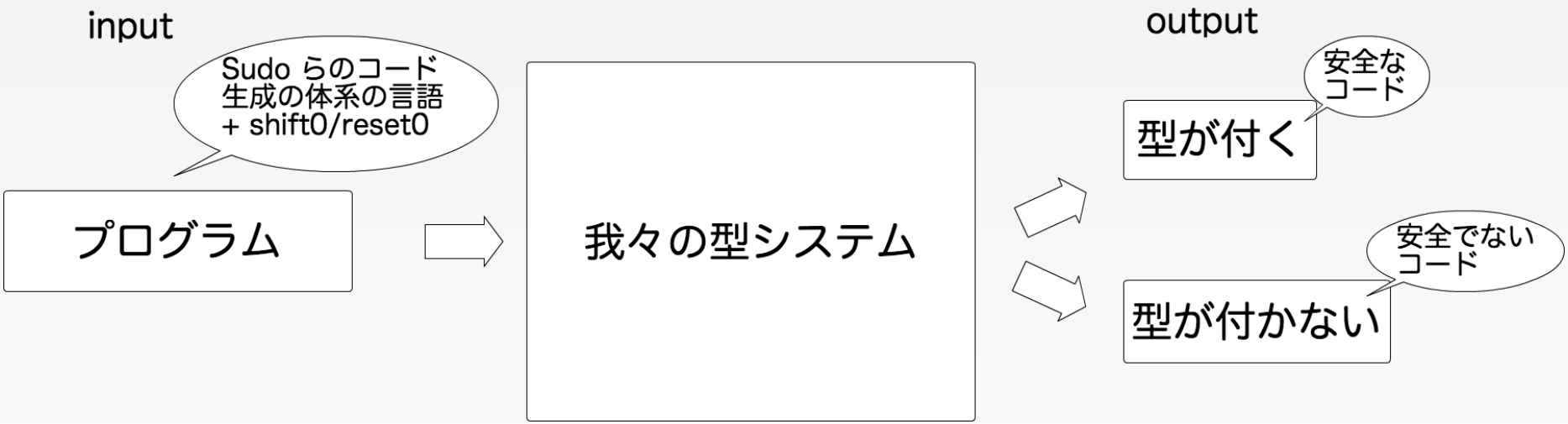
- 表現力: 多段階 let 挿入，メモ化等の技法を表現
- 安全性: 生成されるコードの一定の性質を静的に検査

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

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

shift0/reset0 による多段階 let 挿入

コード生成前

$$\underline{\textbf{cfor}}\ i = 0\ \underline{\textbf{to}}\ n\ \underline{\textbf{in}}$$
$$\underline{\textbf{cfor}}\ j = 0\ \underline{\textbf{to}}\ m\ \underline{\textbf{in}}$$
$$\underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}}$$
$$(a\underline{[i][j]} = b\underline{[i]} + y)$$

shift0/reset0 による多段階 let 挿入

コード生成前

$$\begin{array}{l}
\textbf{reset0}\ \ \underline{\textbf{cfor}}\ i = 0\ \underline{\textbf{to}}\ n\ \underline{\textbf{in}} \\
\quad \underline{\textbf{reset0}}\ \ \underline{\textbf{cfor}}\ j = 0\ \underline{\textbf{to}}\ m\ \underline{\textbf{in}} \\
\quad\quad \underline{\textbf{shift0}}\ k_2 \rightarrow \underline{\textbf{shift0}}\ k_1 \rightarrow \underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}} \\
\quad\quad\quad \underline{\textbf{throw}}\ k_1\ (\underline{\textbf{throw}}\ k_2\ (a\underline{[i][j]} = b\underline{[i]} + y))
\end{array}$$

shift0/reset0 による多段階 let 挿入

コード生成前

$$\begin{array}{l}\textbf{reset0}\ \ \underline{\textbf{cfor}}\ i = 0\ \underline{\textbf{to}}\ n\ \underline{\textbf{in}} \\ \quad \textbf{reset0}\ \ \underline{\textbf{cfor}}\ j = 0\ \underline{\textbf{to}}\ m\ \underline{\textbf{in}} \\ \quad\quad \textbf{shift0}\ k_2\ \rightarrow\ \textbf{shift0}\ k_1\ \rightarrow\ \textbf{clet}\ y = t\ \textbf{in} \\ \quad\quad\quad \underline{\textbf{throw}}\ k_1\ (\underline{\textbf{throw}}\ k_2\ (a\underline{[i][j]} = b\underline{[i]} + y))\end{array}$$

$$\begin{array}{rl}k_1 = & \underline{\textbf{cfor}}\ i = 0\ \underline{\textbf{to}}\ n\ \underline{\textbf{in}} \\ k_2 = & \underline{\textbf{cfor}}\ j = 0\ \underline{\textbf{to}}\ m\ \underline{\textbf{in}}\end{array}$$

shift0/reset0 による多段階 let 挿入

コード生成前

$$\underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}}$$
$$\quad \underline{\textbf{throw}}\ k_1\ (\underline{\textbf{throw}}\ k_2\ (a\underline{[i][j]} = b\underline{[i]} + y))$$

$$k_1 = \quad \underline{\textbf{cfor}}\ i = 0\ \underline{\textbf{to}}\ n\ \underline{\textbf{in}}$$
$$k_2 = \quad \underline{\textbf{cfor}}\ j = 0\ \underline{\textbf{to}}\ m\ \underline{\textbf{in}}$$

shift0/reset0 による多段階 let 挿入

コード生成前

$$
\begin{array}{l}
\underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}} \\
\quad \underline{\textbf{cfor}}\ i = 0\ \underline{\textbf{to}}\ n\ \underline{\textbf{in}} \\
\quad\quad \underline{\textbf{cfor}}\ j = 0\ \underline{\textbf{to}}\ m\ \underline{\textbf{in}} \\
\quad\quad\quad (a\underline{[i][j]} = b\underline{[i]} + y)
\end{array}
$$

型が付く例/付かない例

コード生成前

$$
\begin{aligned}
e = \ & \textbf{reset0}\ \ \underline{\textbf{cfor}}\ i = 0\ \underline{\textbf{to}}\ n\ \underline{\textbf{in}} \\
& \quad \underline{\textbf{reset0}}\ \ \underline{\textbf{cfor}}\ j = 0\ \underline{\textbf{to}}\ m\ \underline{\textbf{in}} \\
& \quad\quad \underline{\textbf{shift0}}\ k_2 \rightarrow \underline{\textbf{shift0}}\ k_1 \rightarrow \underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}} \\
& \quad\quad\quad \underline{\textbf{throw}}\ k_1\ (\underline{\textbf{throw}}\ k_2\ (a\underline{[i][j]} = b\underline{[i]} + y))
\end{aligned}
$$

型が付く例/付かない例

コード生成前

$$e = \underline{\textbf{reset0}}\ \ \underline{\textbf{cfor}}\ i = 0\ \underline{\textbf{to}}\ n\ \underline{\textbf{in}}$$
$$\underline{\textbf{reset0}}\ \ \underline{\textbf{cfor}}\ j = 0\ \underline{\textbf{to}}\ m\ \underline{\textbf{in}}$$
$$\underline{\textbf{shift0}}\ k_2\ \rightarrow\ \underline{\textbf{shift0}}\ k_1\ \rightarrow\ \underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}}$$
$$\underline{\textbf{throw}}\ k_1\ (\underline{\textbf{throw}}\ k_2\ (a\underline{[i][j]} = b\underline{[i]} + y))$$

$$e \rightsquigarrow^* \underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}}$$
$$\underline{\textbf{cfor}}\ i = 0\ \underline{\textbf{to}}\ n\ \underline{\textbf{in}}$$
$$\underline{\textbf{cfor}}\ j = 0\ \underline{\textbf{to}}\ m\ \underline{\textbf{in}}$$
$$(a\underline{[i][j]} = b\underline{[i]} + y)$$

型が付く例/付かない例

コード生成前

$$
\begin{aligned}
e = \ & \underline{\textbf{reset0}}\ \ \underline{\textbf{cfor}}\ i = 0\ \underline{\textbf{to}}\ n\ \underline{\textbf{in}} \\
& \underline{\textbf{reset0}}\ \ \underline{\textbf{cfor}}\ j = 0\ \underline{\textbf{to}}\ m\ \underline{\textbf{in}} \\
& \underline{\textbf{shift0}}\ k_2\ \rightarrow\ \underline{\textbf{shift0}}\ k_1\ \rightarrow\ \underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}} \\
& \underline{\textbf{throw}}\ k_1\ (\underline{\textbf{throw}}\ k_2\ (a\underline{[i][j]} = b\underline{[i]} + y))
\end{aligned}
$$

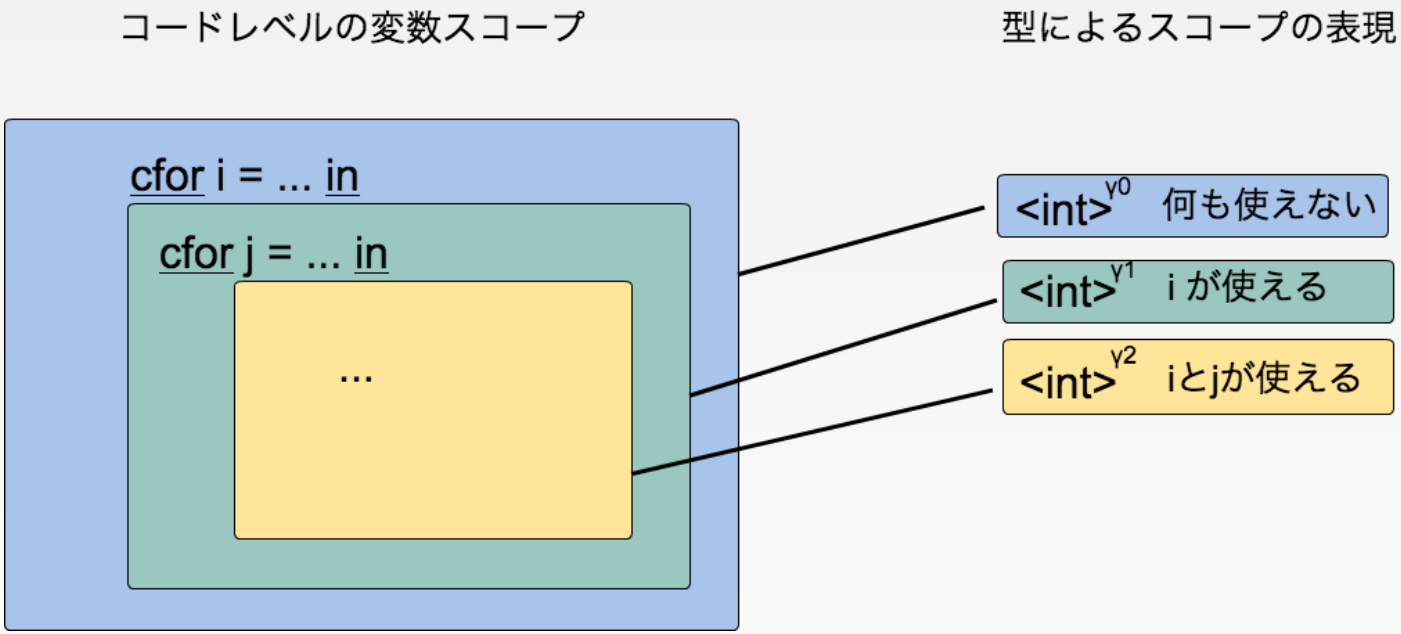
$$
\begin{aligned}
e \rightsquigarrow^* \ & \underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}} \\
& \underline{\textbf{cfor}}\ i = 0\ \underline{\textbf{to}}\ n\ \underline{\textbf{in}} \\
& \underline{\textbf{cfor}}\ j = 0\ \underline{\textbf{to}}\ m\ \underline{\textbf{in}} \\
& (a\underline{[i][j]} = b\underline{[i]} + y)
\end{aligned}
$$

$t = \%7$ **のとき** e **は型が付く**

$t = a\underline{[i][j]}$ **や** $t = b\underline{[j]}$ **のとき** e **は型が付かない**

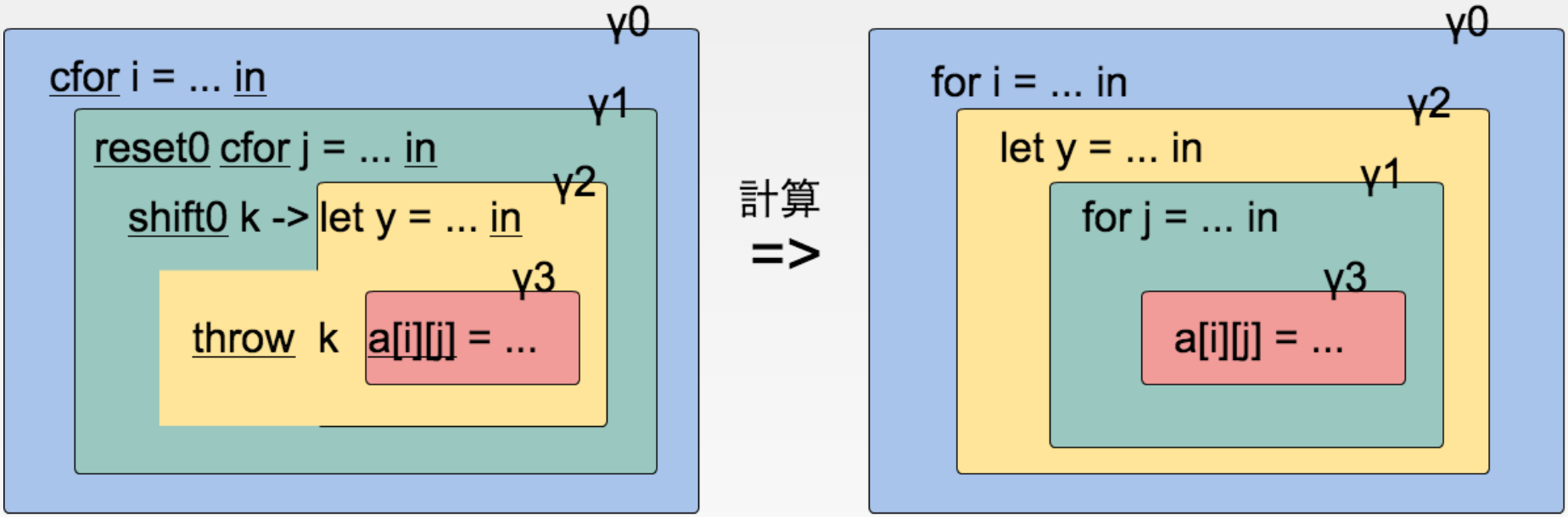
安全なコードにのみ型をつけるにはどうすればよいか

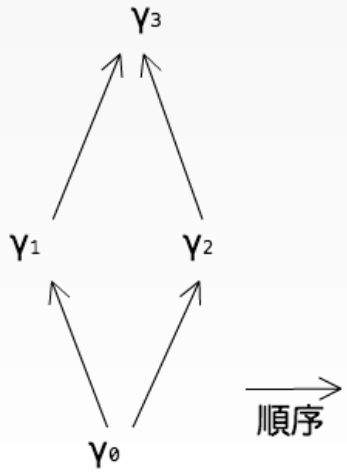
環境識別子 EC によるスコープ表現
[Taha+2003] [Sudo+2014]



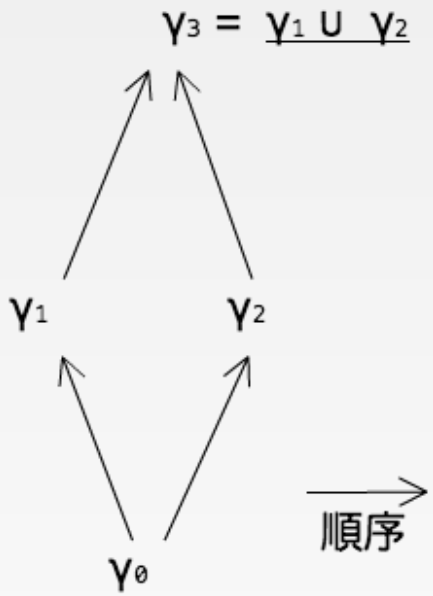
γ_i...Refined Environment Classifier

EC の洗練化 （本研究）

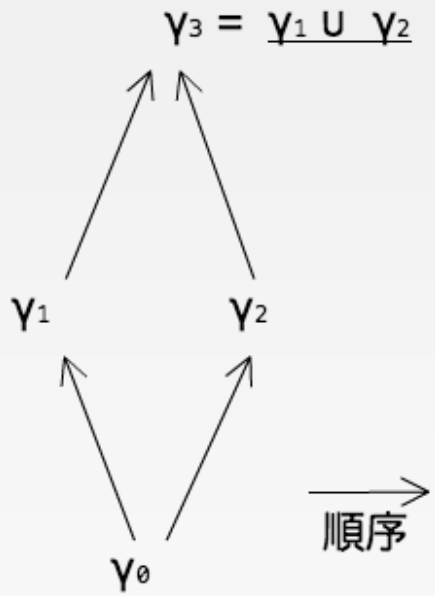

γ0
cfor i = ... in
γ1
reset0 cfor j = ... in
γ2
shift0 k -> let y = ... in
γ3
throw k
a[i][j] = ...
計算
=>
γ0
for i = ... in
γ2
let y = ... in
γ1
for j = ... in
γ3
a[i][j] = ...


γ3
γ1
γ2
γ0
順序

EC のジョイン

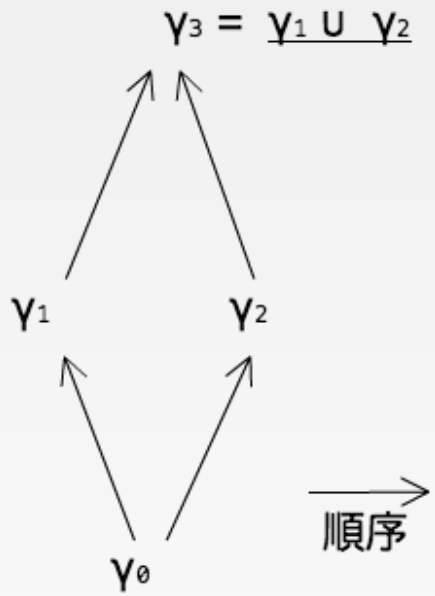


EC のジョイン



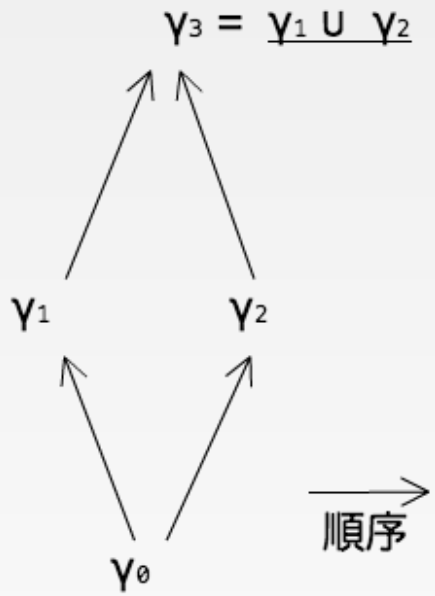
- γ_1 のコードレベル変数は γ_2 では使えない

EC のジョイン



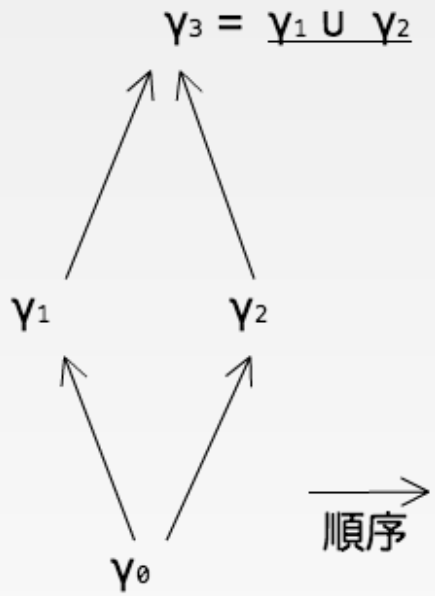
- γ_1 のコードレベル変数は γ_2 では使えない
- γ_2 のコードレベル変数は γ_1 では使えない

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

アウトライン

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行った
- その型システムによって型が付く場合と付かない場合の例をみた

今後の課題

- 設計した型システムの健全性の証明（Subject recudtion 等）を行い，実装を完成させる