

# 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

- ① 概要
- ② 研究の背景
- ③ 研究の目的
- ④ 研究の内容
- ⑤ まとめと今後

# アウトライン

- ① 概要
- ② 研究の背景
- ③ 研究の目的
- ④ 研究の内容
- ⑤ まとめと今後

- Sudo らが作ったコード体系の型システムでは、**破壊的代入**という副作用を扱うようなコード生成の安全性を保証した。
  - しかし、**多段階の let 挿入**を扱うようなコード生成の安全性は保証していなかった
- ⇒ **多段階の let 挿入**を安全に扱うために型システムを改良した。

# アウトライン

## ① 概要

## ② 研究の背景

段階的計算 (コード生成)

コード生成の例

段階的計算の課題

限定継続

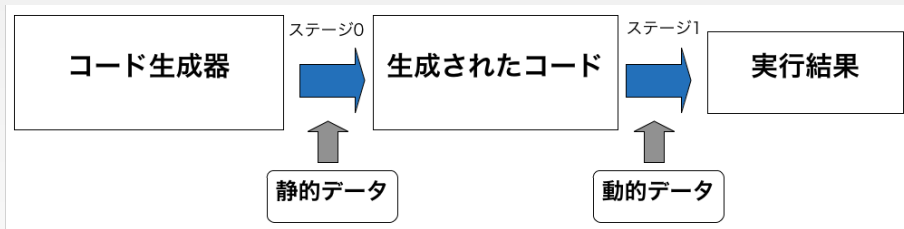
shift0/reset0

## ③ 研究の目的

## ④ 研究の内容

## ⑤ まとめと今後

# 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
  - 「保守性・再利用性の高さ」と「実行性能の高さ」の両立
- ⇒ 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

$\text{power } x \ n$	$= x$	if $n = 1$
	$x * \text{power}(x, n - 1)$	if $n > 1$

# power 関数のコード化

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

## 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

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

## 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

# コード生成における課題

## 生成されたコードの信頼性 (正しさ)

- 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

# コード生成における課題

## 生成されたコードの信頼性 (正しさ)

- 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保証

# コード生成における課題

## 生成されたコードの信頼性 (正しさ)

- 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保証
- 安全なコード: 構文, 型, 変数束縛が正しいプログラム

# コード生成における課題

## 生成されたコードの信頼性 (正しさ)

- 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保証
- 安全なコード: 構文, 型, 変数束縛が正しいプログラム
- **let 挿入**等を実現する**計算エフェクトを含む場合の安全性保証は研究途上**

## 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

# 多段階 let 挿入

clet  $x_1 = \%3$  in

clet  $x_2 = \%5$  in

clet  $y = t$  in

$(x_1 \pm x_2 \pm y)$

## 多段階 let 挿入

clet  $x_1 = \%3$  in

clet  $x_2 = \%5$  in

clet  $y = t$  in

$(x_1 \pm x_2 \pm y)$

reset0 clet  $x_1 = \%3$  in

reset0 clet  $x_2 = \%5$  in

shift0  $k_2 \rightarrow$  shift0  $k_1 \rightarrow$  clet  $y = t$  in

$k_1 (k_2 (x_1 \pm x_2 \pm y))$



# 多段階 let 挿入

reset0 clet  $x_1 = \%3$  in

reset0 clet  $x_2 = \%5$  in

shift0  $k_2 \rightarrow$  shift0  $k_1 \rightarrow$  clet  $y = t$  in

$k_1 (k_2 (x_1 \pm x_2 \pm y))$

# 多段階 let 挿入

clet  $y = t$  in  
clet  $x_1 = \%3$  in  
clet  $x_2 = \%5$  in  
 $(x_1 \text{ + } x_2 \text{ + } y)$

## 多段階 let 挿入の結果

clet  $x_1 = \%3$  in

clet  $x_2 = \%5$  in

clet  $y = t$  in

$(x_1 \pm x_2 \pm y)$

## 多段階 let 挿入の結果

clet  $x_1 = \%3$  in

clet  $x_2 = \%5$  in

clet  $y = t$  in

$(x_1 \text{ + } x_2 \text{ + } y)$

$\Rightarrow$  (shift0/reset0 を付与)

## 多段階 let 挿入の結果

clet  $x_1 = \%3$  in

clet  $x_2 = \%5$  in

clet  $y = t$  in

$(x_1 \text{ + } x_2 \text{ + } y)$

$\Rightarrow$  (shift0/reset0 を付与)

clet  $y = t$  in

clet  $x_1 = \%3$  in

clet  $x_2 = \%5$  in

$(x_1 \text{ + } x_2 \text{ + } y)$

# アウトライン

- ① 概要
- ② 研究の背景
- ③ 研究の目的**
- ④ 研究の内容
- ⑤ まとめと今後

# 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 研究の背景

## ③ 研究の目的

## ④ 研究の内容

行うこと

困難・問題点

本研究の手法

多段階 let 挿入

型システム

型付けの例

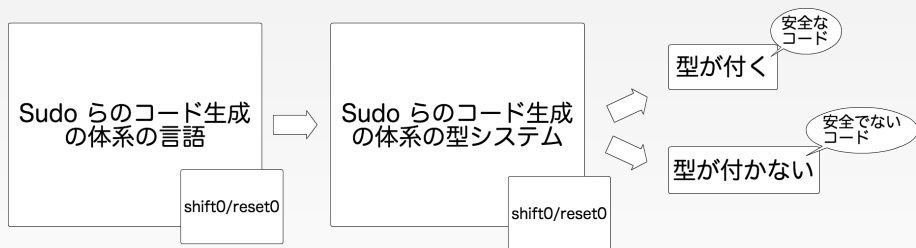
## ⑤ まとめと今後

- **コントロールオペレータ `shift0/reset0` を利用し、深く入れ子になった内側の `shift0` からの `let` 挿入 (多段階 `let` 挿入) などのコード生成技法を行える言語の設計**
- **`shift0/reset0` を持つコード生成言語の型システムの設計**

# 困難・問題点

- shift0/reset0 は shift/reset より強力であるため、型システムが非常に複雑
- コード生成言語の型システムも一定の複雑さを持つ
- ⇒ 単純な融合は困難

# 本研究の手法



# 多段階 let 挿入

clet  $x_1 = \%3$  in

clet  $x_2 = \%5$  in

clet  $y = t$  in

$(x_1 \text{ + } x_2 \text{ + } y)$

## 多段階 let 挿入

clet  $x_1 = \%3$  in

clet  $x_2 = \%5$  in

clet  $y = t$  in

$(x_1 \oplus x_2 \oplus y)$

reset0 clet  $x_1 = \%3$  in

reset0 clet  $x_2 = \%5$  in

shift0  $k_2 \rightarrow$  shift0  $k_1 \rightarrow$  clet  $y = t$  in

throw  $k_1$  (throw  $k_2$   $(x_1 \oplus x_2 \oplus y)$ )

## shift0/reset0 による多段階 let 挿入

reset0 clet  $x_1 = \%3$  in

reset0 clet  $x_2 = \%5$  in

shift0  $k_2 \rightarrow$  shift0  $k_1 \rightarrow$  clet  $y = t$  in

throw  $k_1$  (throw  $k_2$  ( $x_1 \pm x_2 \pm y$ ))

# shift0/reset0 による多段階 let 挿入

reset0 clet  $x_1 = \%3$  in

reset0 clet  $x_2 = \%5$  in

shift0  $k_2 \rightarrow$  shift0  $k_1 \rightarrow$  clet  $y = t$  in

throw  $k_1$  (throw  $k_2$  ( $x_1 \oplus x_2 \oplus y$ ))

$k_1 =$  clet  $x_1 = \%3$  in

$k_2 =$  clet  $x_2 = \%5$  in



clet  $y = t$  in

throw  $k_1$  (throw  $k_2$  ( $x_1 \pm x_2 \pm y$ ))

$k_1 = \text{clet } x_1 = \%3 \text{ in}$

$k_2 = \text{clet } x_2 = \%5 \text{ in}$

clet  $y = t$  in

clet  $x_1 = \%3$  in

clet  $x_2 = \%5$  in

$(x_1 \text{ + } x_2 \text{ + } y)$

## shift0/reset0 による多段階 let 挿入の結果

clet  $x_1 = \%3$  in

clet  $x_2 = \%5$  in

clet  $y = t$  in

$(x_1 \pm x_2 \pm y)$

## shift0/reset0 による多段階 let 挿入の結果

clet  $x_1 = \%3$  in

clet  $x_2 = \%5$  in

clet  $y = t$  in

$(x_1 \pm x_2 \pm y)$

$\Rightarrow$  (shift0/reset0 を付与)

## shift0/reset0 による多段階 let 挿入の結果

clet  $x_1 = \%3$  in

clet  $x_2 = \%5$  in

clet  $y = t$  in

$(x_1 \pm x_2 \pm y)$

$\Rightarrow$  (shift0/reset0 を付与)

clet  $y = t$  in

clet  $x_1 = \%3$  in

clet  $x_2 = \%5$  in

$(x_1 \pm x_2 \pm y)$

# 変数スコープの利用 by Sudo

一般的な `shift0/reset0 (throw)` の式の形

$$(\underline{\text{reset0}} \dots (\underline{\text{shift0}} \ k \rightarrow \dots (\underline{\text{throw}} \ k \dots )))$$

# 変数スコープの利用 by Sudo

- $\gamma$  は変数のスコープを表す.
- その時点で使える自由変数の集合と思ってもらえば良い.
- $\gamma$  には, 包含関係があり, それを  $\gamma_0 \geq \gamma_1$  というような順序で表している.
- 直感的には  $\gamma_0$  より  $\gamma_1$  のほうが使える自由変数が多いという意味である.

$(\text{reset0 } \gamma_0 (\text{shift0 } k \rightarrow \gamma_1 (\text{throw } k \gamma_3)))$

# shift0/reset0 による let 挿入が型安全性を保つ条件

一般的な shift0/reset0 (throw) の式の形

$$(\underline{\text{reset0}} \dots (\underline{\text{shift0}} \ k \rightarrow \dots (\underline{\text{throw}} \ k \dots)))$$

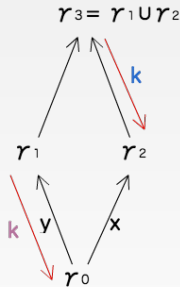


# shift0/reset0 による let 挿入が型安全性を保つ条件

$(\underline{\text{reset0}} \ \underline{\text{clet}} \ y = e_1 \ \underline{\text{in}} \ \dots$   
 $(\underline{\text{shift0}} \ k \rightarrow \underline{\text{clet}} \ x = \dots \ \underline{\text{in}} \ \dots$   
 $(\underline{\text{throw}} \ k \ \dots \ )))$

$\Rightarrow$

$(\underline{\text{reset0}} \ \underline{\text{clet}} \ y = e_1 \ \underline{\text{in}} \ \gamma_0$   
 $(\underline{\text{shift0}} \ k \rightarrow \underline{\text{clet}} \ x = \gamma_1 \ \underline{\text{in}} \ \gamma_2$   
 $(\underline{\text{throw}} \ k \ \gamma_3 \ )))$



$k : (\langle \cdot \rangle^{\gamma_1} \Rightarrow \langle \cdot \rangle^{\gamma_0})$

$k : (\langle \cdot \rangle^{\gamma_3} \Rightarrow \langle \cdot \rangle^{\gamma_2})$

# 型が付く例/付かない例

$e_1 =$  reset0 clet  $x_1 = \%3$  in  
    reset0 clet  $x_2 = \%5$  in  
    shift0  $k \rightarrow$  clet  $y = t$  in  
    throw  $k (x_1 \pm x_2 \pm y)$

# 型が付く例/付かない例

$$e_1 = \underline{\text{reset0}} \quad \underline{\text{clet}} \ x_1 = \%3 \ \underline{\text{in}} \\ \underline{\text{reset0}} \quad \underline{\text{clet}} \ x_2 = \%5 \ \underline{\text{in}} \\ \underline{\text{shift0}} \ k \rightarrow \text{clet } y = t \ \underline{\text{in}} \\ \underline{\text{throw}} \ k \ (x_1 \ \underline{+} \ x_2 \ \underline{+} \ y)$$
$$e_1 \rightsquigarrow^* \underline{\text{reset0}} \quad \underline{\text{clet}} \ x_1 = \%3 \ \underline{\text{in}} \\ \text{clet } y = t \ \underline{\text{in}} \\ \underline{\text{reset0}} \quad \underline{\text{clet}} \ x_2 = \%5 \ \underline{\text{in}} \\ (x_1 \ \underline{+} \ x_2 \ \underline{+} \ y)$$

# 型が付く例/付かない例

$$e_1 = \underline{\text{reset0}} \quad \underline{\text{clet}} \ x_1 = \%3 \ \underline{\text{in}} \\ \underline{\text{reset0}} \quad \underline{\text{clet}} \ x_2 = \%5 \ \underline{\text{in}} \\ \underline{\text{shift0}} \ k \rightarrow \text{clet } y = t \ \underline{\text{in}} \\ \underline{\text{throw}} \ k \ (x_1 \ \underline{+} \ x_2 \ \underline{+} \ y)$$
$$e_1 \rightsquigarrow^* \underline{\text{reset0}} \quad \underline{\text{clet}} \ x_1 = \%3 \ \underline{\text{in}} \\ \text{clet } y = t \ \underline{\text{in}} \\ \underline{\text{reset0}} \quad \underline{\text{clet}} \ x_2 = \%5 \ \underline{\text{in}} \\ (x_1 \ \underline{+} \ x_2 \ \underline{+} \ y)$$

$t = \%7$  か  $t = x_1$  のとき  $e_1$  は型が付く  
 $t = x_2$  のとき  $e_1$  は型が付かない

# 型が付く例/付かない例

$e_2 =$  reset0 clet  $x_1 = \%3$  in  
    reset0 clet  $x_2 = \%5$  in  
    shift0  $k_2 \rightarrow$  shift0  $k_1 \rightarrow$  clet  $y = t$  in  
    throw  $k_1$  (throw  $k_2$  ( $x_1 \pm x_2 \pm y$ ))

# 型が付く例/付かない例

$e_2 = \underline{\text{reset0}} \ \underline{\text{clet}} \ x_1 = \%3 \ \underline{\text{in}}$   
 $\quad \underline{\text{reset0}} \ \underline{\text{clet}} \ x_2 = \%5 \ \underline{\text{in}}$   
 $\quad \underline{\text{shift0}} \ k_2 \rightarrow \underline{\text{shift0}} \ k_1 \rightarrow \text{clet } y = t \ \underline{\text{in}}$   
 $\quad \underline{\text{throw}} \ k_1 \ (\underline{\text{throw}} \ k_2 \ (x_1 \ \underline{+} \ x_2 \ \underline{+} \ y))$

$e_2 \rightsquigarrow^* \text{clet } y = t \ \underline{\text{in}}$   
 $\quad \underline{\text{reset0}} \ \underline{\text{clet}} \ x_1 = \%3 \ \underline{\text{in}}$   
 $\quad \underline{\text{reset0}} \ \underline{\text{clet}} \ x_2 = \%5 \ \underline{\text{in}}$   
 $\quad (x_1 \ \underline{+} \ x_2 \ \underline{+} \ y)$

# 型が付く例/付かない例

$$e_2 = \underline{\text{reset0}} \ \underline{\text{clet}} \ x_1 = \%3 \ \underline{\text{in}} \\ \underline{\text{reset0}} \ \underline{\text{clet}} \ x_2 = \%5 \ \underline{\text{in}} \\ \underline{\text{shift0}} \ k_2 \rightarrow \underline{\text{shift0}} \ k_1 \rightarrow \text{clet } y = t \ \underline{\text{in}} \\ \underline{\text{throw}} \ k_1 \ (\underline{\text{throw}} \ k_2 \ (x_1 \ \underline{+} \ x_2 \ \underline{+} \ y))$$
$$e_2 \rightsquigarrow^* \text{clet } y = t \ \underline{\text{in}} \\ \underline{\text{reset0}} \ \underline{\text{clet}} \ x_1 = \%3 \ \underline{\text{in}} \\ \underline{\text{reset0}} \ \underline{\text{clet}} \ x_2 = \%5 \ \underline{\text{in}} \\ (x_1 \ \underline{+} \ x_2 \ \underline{+} \ y)$$

$t = \%7$  のとき  $e_2$  は型が付く

$t = x_2$  か  $t = x_1$  のとき  $e_2$  は型が付かない

# アウトライン

- ① 概要
- ② 研究の背景
- ③ 研究の目的
- ④ 研究の内容
- ⑤ **まとめと今後**



# まとめと今後

- コードの型システムに `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 `answer type modification` に対応した型システムを設計し, (`subject reduction` 等の) 健全性の証明を行う