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

コード生成法：プログラムの実行性能の高さ，保守性・再利用性を両立

概要

コード生成法：プログラムの実行性能の高さ，保守性・再利用性を両立

- **コード生成言語の型安全性に関して破壊的代入を持つ言語体系に対する** Sudo **らの研究がある**

概要

コード生成法：プログラムの実行性能の高さ，保守性・再利用性を両立

- コード生成言語の型安全性に関して破壊的代入を持つ言語体系に対する Sudo らの研究がある
- しかし，**多段階の let 挿入**を扱うようなコード生成言語の安全性は保証していなかった

概要

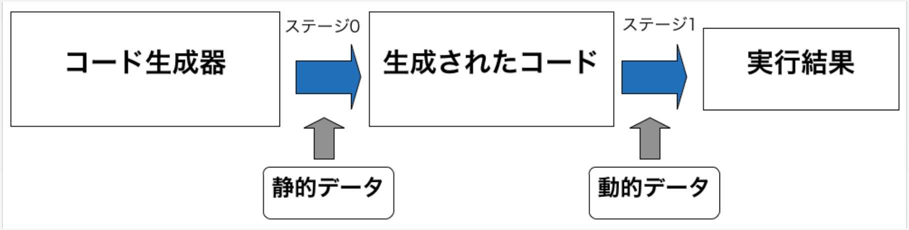
コード生成法：プログラムの実行性能の高さ，保守性・再利用性を両立

- コード生成言語の型安全性に関して破壊的代入を持つ言語体系に対する Sudo らの研究がある
- しかし，多段階の let 挿入を扱うようなコード生成言語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

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

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

power 関数のコード化

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

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

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

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 挿入の例

for $i = 1$ **in**

for $j = 1$ **in**

$y = t$

$a[i][j] = b[i] + y$

多段階 let 挿入の例

for $i = 1$ **in**
for $j = 1$ **in**
$y = t$
$a[i][j] = b[i] + y$

$$\Downarrow$$

for $i = 1$ **in**
for $j = 1$ **in**
$y = t$ — t が i,j に依存した式
$a[i][j] = b[i] + y$

多段階 let 挿入の例

for $i = 1$ **in**
for $j = 1$ **in**
$y = t$
$a[i][j] = b[i] + y$

$$\Downarrow$$

for $i = 1$ **in**
$y = t$ — t が i にのみ依存し j には依存しない式
for $j = 1$ **in**
$a[i][j] = b[i] + y$

多段階 let 挿入の例

for $i = 1$ **in**
for $j = 1$ **in**
$y = t$
$a[i][j] = b[i] + y$

$$\Downarrow$$

$y = t$ — t が i にも j にも依存しない式
for $i = 1$ **in**
for $j = 1$ **in**
$a[i][j] = b[i] + y$

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

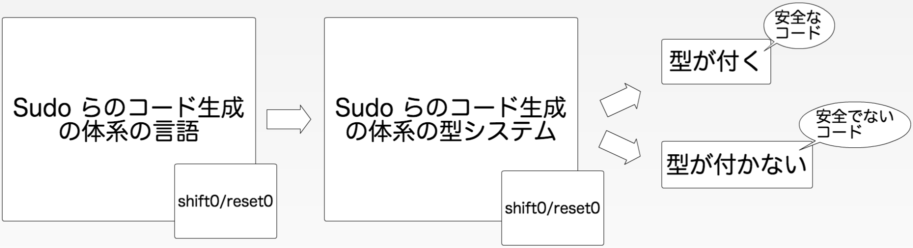
行うこと

- **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


Sudo らのコード生成
の体系の言語
shift0/reset0
Sudo らのコード生成
の体系の型システム
shift0/reset0
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

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

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 $\to$ **shift0** k_1 $\to$ **clet** $y = t$ **in**
throw k_1 (**throw** k_2 $(x_1 \mathbin{\underline{+}} x_2 \mathbin{\underline{+}} y)$)

shift0/reset0 による多段階 let 挿入

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

$$k_1 = \underline{\textbf{clet}}\ x_1 = \%3\ \underline{\textbf{in}}$$
$$k_2 = \underline{\textbf{clet}}\ x_2 = \%5\ \underline{\textbf{in}}$$

shift0/reset0 による多段階 let 挿入

$$\underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}}$$

$$\underline{\textbf{throw}}\ k_1\ (\underline{\textbf{throw}}\ k_2\ (x_1\ \underline{+}\ x_2\ \underline{+}\ y))$$

$$k_1 = \underline{\textbf{clet}}\ x_1 = \%3\ \underline{\textbf{in}}$$

$$k_2 = \underline{\textbf{clet}}\ x_2 = \%5\ \underline{\textbf{in}}$$

shift0/reset0 による多段階 let 挿入

$\underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}}$

$\underline{\textbf{clet}}\ x_1 = \%3\ \underline{\textbf{in}}$

$\underline{\textbf{clet}}\ x_2 = \%5\ \underline{\textbf{in}}$

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

shift0/reset0 による多段階 let 挿入の結果

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

shift0/reset0 による多段階 let 挿入の結果

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

$\Rightarrow$ (shift0/reset0 **を付与**)

shift0/reset0 による多段階 let 挿入の結果

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

$\Rightarrow$ (shift0/reset0 **を付与**)

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

型が付く例/付かない例

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

型が付く例/付かない例

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

$$
\begin{aligned}
e_2 \rightsquigarrow^{*} {} & \underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}} \\
& \underline{\textbf{reset0}}\ \ \underline{\textbf{clet}}\ x_1 = \%3\ \underline{\textbf{in}} \\
& \underline{\textbf{reset0}}\ \ \underline{\textbf{clet}}\ x_2 = \%5\ \underline{\textbf{in}} \\
& (x_1 \mathbin{\underline{+}} x_2 \mathbin{\underline{+}} y)
\end{aligned}
$$

型が付く例/付かない例

$$
\begin{aligned}
e_2 = {} & \underline{\textbf{reset0}}\ \ \underline{\textbf{clet}}\ x_1 = \%3\ \underline{\textbf{in}} \\
& \underline{\textbf{reset0}}\ \ \underline{\textbf{clet}}\ x_2 = \%5\ \underline{\textbf{in}} \\
& \underline{\textbf{shift0}}\ k_2\ \to\ \underline{\textbf{shift0}}\ k_1\ \to\ \color{red}{\underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}}} \\
& \underline{\textbf{throw}}\ k_1\ (\underline{\textbf{throw}}\ k_2\ (x_1 \mathbin{\underline{+}} x_2 \mathbin{\underline{+}} y))
\end{aligned}
$$

$$
\begin{aligned}
e_2 \rightsquigarrow^* {} & \color{red}{\underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}}} \\
& \underline{\textbf{reset0}}\ \ \underline{\textbf{clet}}\ x_1 = \%3\ \underline{\textbf{in}} \\
& \underline{\textbf{reset0}}\ \ \underline{\textbf{clet}}\ x_2 = \%5\ \underline{\textbf{in}} \\
& (x_1 \mathbin{\underline{+}} x_2 \mathbin{\underline{+}} y)
\end{aligned}
$$

$t = \%7$ **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**

型が付く例/付かない例

$$
\begin{aligned}
e_1 = {} & \underline{\textbf{reset0}}\ \ \underline{\textbf{clet}}\ x_1 = \%3\ \underline{\textbf{in}} \\
& \underline{\textbf{reset0}}\ \ \underline{\textbf{clet}}\ x_2 = \%5\ \underline{\textbf{in}} \\
& \underline{\textbf{shift0}}\ k\ \to\ \color{red}{\underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}}} \\
& \underline{\textbf{throw}}\ k\ (x_1 \mathbin{\underline{+}} x_2 \mathbin{\underline{+}} y)
\end{aligned}
$$

型が付く例/付かない例

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

$$\begin{aligned} e_1 \leadsto^* \underline{\textbf{reset0}}\ \ & \underline{\textbf{clet}}\ x_1 = \%3\ \underline{\textbf{in}} \\ & \underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}} \\ & \underline{\textbf{reset0}}\ \ \underline{\textbf{clet}}\ x_2 = \%5\ \underline{\textbf{in}} \\ & (x_1 \mathbin{\underline{+}} x_2 \mathbin{\underline{+}} y) \end{aligned}$$

型が付く例/付かない例

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

$$
\begin{aligned}
e_1 \leadsto^* {} & \underline{\textbf{reset0}}\ \ \underline{\textbf{clet}}\ x_1 = \%3\ \underline{\textbf{in}} \\
& \underline{\textbf{clet}}\ y = t\ \underline{\textbf{in}} \\
& \underline{\textbf{reset0}}\ \ \underline{\textbf{clet}}\ x_2 = \%5\ \underline{\textbf{in}} \\
& (x_1 \ \underline{+}\ x_2\ \underline{+}\ y)
\end{aligned}
$$

$t = \%7$ 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

変数スコープの利用 by Sudo

一般的な shift0/reset0 (throw) **の式の形**

$$(\underline{\textbf{reset0}}\ \ldots\ (\underline{\textbf{shift0}}\ k \rightarrow\ \ \ldots\ (\underline{\textbf{throw}}\ k\ \ldots\ )))$$

変数スコープの利用 by Sudo

- γ 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.
- γ には，包含関係があり，それを $\gamma_1 \geq \gamma_0$ というような順序で表している.
- 直感的には γ_0 より γ_1 のほうが使える自由変数が多いという意味である.

$$(\underline{\textbf{reset0}}\ \gamma_0\ (\underline{\textbf{shift0}}\ k \to\ \gamma_1\ (\underline{\textbf{throw}}\ k\ \gamma_3)))$$

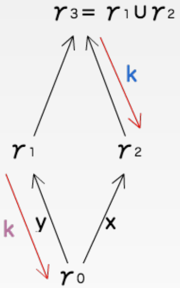
shift0/reset0 による let 挿入が型安全性を保つ条件

一般的な shift0/reset0 (throw) **の式の形**

$$(\underline{\textbf{reset0}}\ \ldots\ (\underline{\textbf{shift0}}\ k \rightarrow\ \ \ldots\ (\underline{\textbf{throw}}\ k\ \ldots\ )))$$

shift0/reset0 による let 挿入が型安全性を保つ条件

(**reset0** **clet** $y = \ldots$ **in** $\ldots$
(**shift0** $k \rightarrow$ **clet** $x = \ldots$ **in** $\ldots$
(**throw** k $\ldots$)))
=>
(**reset0** **clet** $y = \gamma_0$ **in** γ_1
(**shift0** $k \rightarrow$ **clet** $x = \gamma_1$ **in** γ_2
(**throw** k γ_3)))



$$k \ : \ (\langle\cdot\rangle^{\gamma_1} \Rightarrow \langle\cdot\rangle^{\gamma_0})$$
$$k \ : \ (\langle\cdot\rangle^{\gamma_3} \Rightarrow \langle\cdot\rangle^{\gamma_2})$$

アウトライン

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