4.正しさが証明可能な実装

抽象機械へのコンパイル

初期状態→意味論の解釈器→最終状態

↑

While言語

↓

コンパイラ

↓抽象機械語

初期状態→抽象機械の解釈器→最終状態

抽象機械

- 計算状態<c,e,s>
  - c: 実行コード
  - e: 算術式や論理式の評価のためのスタック
  - s: メモリ
- 命令の実行は計算状態を別の計算状態に写すこと

より現実的な機械

- c: 固定されたコード列とプログラムカウンタ
- e: 評価スタック
- s: メモリ＝アドレスで参照可能なセルの列

単純な命令セット

$$
\begin{array}{lll}
inst & ::= & \text{PUSH-}n \mid \text{ADD} \mid \text{MULT} \mid \text{SUB} \\
& \mid & \text{TRUE} \mid \text{FALSE} \mid \text{EQ} \mid \text{LE} \mid \text{AND} \mid \text{NEG} \\
& \mid & \text{FETCH-}x \mid \text{STORE-}x \\
& \mid & \text{NOOP} \mid \text{BRANCH}(c,\ c) \mid \text{LOOP}(c,\ c) \\
c & ::= & \varepsilon \mid inst{:}c
\end{array}
$$

コード例

```
PUSH-0:STORE-z:FETCH-x:STORE-r:
LOOP(  FETCH-r:FETCH-y:LE,
       FETCH-y:FETCH-r:SUB:STORE-r:
       PUSH-1:FETCH-z:ADD:STORE-z)
```

z:=0; r:=x; while y≦r do (r:=r-y; z:=z+1)
というコードから生成される

遷移関係 $\langle c,\ e,\ s\rangle \triangleright \langle c',\ e',\ s'\rangle$

<PUSH-n:c, e, s> ▷ <c, N[[n]]:e, s>

- PUSH-nが次の命令
- PUSH-nの実行後は命令列がcに
- nの値N[[n]]がプッシュされる
- 状態は変わらない

<ADD:c, z1:z2:e, s> ▷ <c, (z1+z2):e, s>　if z1, z2∈Z

- スタックの上に2つの数　→　ポップされる
- 和がプッシュされる

命令の意味論

$$\langle \text{PUSH-}n : c, e, s\rangle \triangleright \langle c, \mathcal{N}[\![n]\!] : e, s\rangle$$

$$\langle \text{ADD} : c, z_1 : z_2 : e, s\rangle \triangleright \langle c, (z_1 + z_2) : e, s\rangle \quad \text{if } z_1, z_2 \in \mathbf{Z}$$

$$\langle \text{MULT} : c, z_1 : z_2 : e, s\rangle \triangleright \langle c, (z_1 \star z_2) : e, s\rangle \quad \text{if } z_1, z_2 \in \mathbf{Z}$$

$$\langle \text{SUB} : c, z_1 : z_2 : e, s\rangle \triangleright \langle c, (z_1 - z_2) : e, s\rangle \quad \text{if } z_1, z_2 \in \mathbf{Z}$$

$$\langle \text{TRUE} : c, e, s\rangle \triangleright \langle c, \mathsf{tt} : e, s\rangle$$

$$\langle \text{FALSE} : c, e, s\rangle \triangleright \langle c, \mathsf{ff} : e, s\rangle$$

$$\langle \text{EQ} : c, z_1 : z_2 : e, s\rangle \triangleright \langle c, (z_1 = z_2) : e, s\rangle \quad \text{if } z_1, z_2 \in \mathbf{Z}$$

$$\langle \text{LE} : c, z_1 : z_2 : e, s\rangle \triangleright \langle c, (z_1 \leq z_2) : e, s\rangle \quad \text{if } z_1, z_2 \in \mathbf{Z}$$

$$\langle \text{AND} : c, t_1 : t_2 : e, s\rangle \triangleright \begin{cases} \langle c, \mathsf{tt} : e, s\rangle & \text{if } t_1 = \mathsf{tt} \text{ and } t_2 = \mathsf{tt} \\ \langle c, \mathsf{ff} : e, s\rangle & \text{if } t_1 = \mathsf{ff} \text{ or } t_2 = \mathsf{ff}, t_1, t_2 \in \mathbf{T} \end{cases}$$

$$\langle \text{NEG} : c, t : e, s\rangle \triangleright \begin{cases} \langle c, \mathsf{ff} : e, s\rangle & \text{if } t = \mathsf{tt} \\ \langle c, \mathsf{tt} : e, s\rangle & \text{if } t = \mathsf{ff} \end{cases}$$

$$\langle \text{FETCH-}x : c, e, s\rangle \triangleright \langle c, (s\ x) : e, s\rangle$$

$$\langle \text{STORE-}x : c, z : e, s\rangle \triangleright \langle c, e, s[x \mapsto z]\rangle \quad \text{if } z \in \mathbf{Z}$$

$$\langle \text{NOOP} : c, e, s\rangle \triangleright \langle c, e, s\rangle$$

$$\langle \text{BRANCH}(c_1, c_2) : c, t : e, s\rangle \triangleright \begin{cases} \langle c_1 : c, e, s\rangle & \text{if } t = \mathsf{tt} \\ \langle c_2 : c, e, s\rangle & \text{if } t = \mathsf{ff} \end{cases}$$

$$\langle \text{LOOP}(c_1, c_2) : c, e, s\rangle \triangleright \langle c_1 : \text{BRANCH}(c_2 : \text{LOOP}(c_1, c_2), \text{NOOP}) : c, e, s\rangle$$

例

- s x = 3 であるような状態s

$$\langle \text{PUSH-1:FETCH-x:ADD:STORE-x}, \varepsilon, s \rangle$$

$$\triangleright \langle \text{FETCH-x:ADD:STORE-x}, \mathbf{1}, s \rangle$$

$$\triangleright \langle \text{ADD:STORE-x}, \mathbf{3{:}1}, s \rangle$$

$$\triangleright \langle \text{STORE-x}, \mathbf{4}, s \rangle$$

$$\triangleright \langle \varepsilon, \varepsilon, s[\mathbf{x} \longmapsto \mathbf{4}] \rangle$$

例

$$\langle \textsc{loop}(\bar{}\textsc{true},\ \textsc{noop}),\ \varepsilon,\ s\rangle$$

$$\triangleright\ \langle \textsc{true:branch}(\bar{}\textsc{noop:loop}(\textsc{true},\ \textsc{noop}),\ \textsc{noop}),\ \varepsilon,\ s\rangle$$

$$\triangleright\ \langle \textsc{branch}(\bar{}\textsc{noop:loop}(\textsc{true},\ \textsc{noop}),\ \textsc{noop}),\ \mathbf{tt},\ s\rangle$$

$$\triangleright\ \langle \textsc{noop:loop}(\textsc{true},\ \textsc{noop}),\ \varepsilon,\ s\rangle$$

$$\triangleright\ \langle \textsc{loop}(\textsc{true},\ \textsc{noop}),\ \varepsilon,\ s\rangle$$

$$\triangleright\ \cdots$$

コード生成

$$\mathcal{CA}[\![n]\!] = \text{PUSH-}n$$

$$\mathcal{CA}[\![x]\!] = \text{FETCH-}x$$

$$\mathcal{CA}[\![a_1 + a_2]\!] = \mathcal{CA}[\![a_2]\!] : \mathcal{CA}[\![a_1]\!] : \text{ADD}$$

$$\mathcal{CA}[\![a_1 \star a_2]\!] = \mathcal{CA}[\![a_2]\!] : \mathcal{CA}[\![a_1]\!] : \text{MULT}$$

$$\mathcal{CA}[\![a_1 - a_2]\!] = \mathcal{CA}[\![a_2]\!] : \mathcal{CA}[\![a_1]\!] : \text{SUB}$$

$$\mathcal{CB}[\![\texttt{true}]\!] = \text{TRUE}$$

$$\mathcal{CB}[\![\texttt{false}]\!] = \text{FALSE}$$

$$\mathcal{CB}[\![a_1 = a_2]\!] = \mathcal{CA}[\![a_2]\!] : \mathcal{CA}[\![a_1]\!] : \text{EQ}$$

$$\mathcal{CB}[\![a_1 \leq a_2]\!] = \mathcal{CA}[\![a_2]\!] : \mathcal{CA}[\![a_1]\!] : \text{LE}$$

$$\mathcal{CB}[\![\neg b]\!] = \mathcal{CB}[\![b]\!] : \text{NEG}$$

$$\mathcal{CB}[\![b_1 \wedge b_2]\!] = \mathcal{CB}[\![b_2]\!] : \mathcal{CB}[\![b_1]\!] : \text{AND}$$

$$\mathcal{CS}[\![x := a]\!] = \mathcal{CA}[\![a]\!] : \text{STORE-}x$$

$$\mathcal{CS}[\![\texttt{skip}]\!] = \text{NOOP}$$

$$\mathcal{CS}[\![S_1; S_2]\!] = \mathcal{CS}[\![S_1]\!] : \mathcal{CS}[\![S_2]\!]$$

$$\mathcal{CS}[\![\texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2]\!] = \mathcal{CB}[\![b]\!] : \text{BRANCH}(\mathcal{CS}[\![S_1]\!], \mathcal{CS}[\![S_2]\!])$$

$$\mathcal{CS}[\![\texttt{while } b \texttt{ do } S]\!] = \text{LOOP}(\mathcal{CB}[\![b]\!], \mathcal{CS}[\![S]\!])$$

例：階乗計算

$$
\begin{aligned}
&\mathcal{CS}\,[\![\mathtt{y{:=}1;\ while}\ \neg(\mathtt{x{=}1})\ \mathtt{do}\ (\mathtt{y{:=}y}\star\mathtt{x;\ x{:=}x{-}1})]\!] \\
&\quad = \mathcal{CS}[\![\mathtt{y{:=}1}]\!]{:}\mathcal{CS}[\![\mathtt{while}\ \neg(\mathtt{x{=}1})\ \mathtt{do}\ (\mathtt{y{:=}y}\star\mathtt{x;\ x{:=}x{-}1})]\!] \\
&\quad = \mathcal{CA}[\![1]\!]{:}\textsc{store-y}{:}\textsc{loop}(\mathcal{CB}[\![\neg(\mathtt{x{=}1})]\!], \mathcal{CS}[\![\mathtt{y{:=}y}\star\mathtt{x;\ x{:=}x{-}1}]\!]) \\
&\quad = \textsc{push-1}{:}\textsc{store-y}{:}\textsc{loop}(\mathcal{CB}[\![\mathtt{x{=}1}]\!]{:}\textsc{neg}, \mathcal{CS}[\![\mathtt{y{:=}y}\star\mathtt{x}]\!]{:}\mathcal{CS}[\![\mathtt{x{:=}x{-}1}]\!]) \\
&\quad \vdots \\
&\quad = \textsc{push-1}{:}\textsc{store-y}{:}\textsc{loop}(\ \textsc{push-1}{:}\textsc{fetch-x}{:}\textsc{eq}{:}\textsc{neg}, \\
&\qquad\qquad \textsc{fetch-x}{:}\textsc{fetch-y}{:}\textsc{mult}{:}\textsc{store-y}{:} \\
&\qquad\qquad\quad \textsc{push-1}{:}\textsc{fetch-x}{:}\textsc{sub}{:}\textsc{store-x})
\end{aligned}
$$

定理

$$\langle \mathcal{CS}[\![S]\!], \epsilon, s\rangle \triangleright^* \langle \epsilon, \epsilon, s'\rangle \quad \text{if and only if} \quad \langle S\,,\ s\rangle \rightarrow s'$$

証明方法？

- 構造帰納法？
- 導出木の形に関する帰納法？
- 導出列の長さに関する帰納法？
- ？

まとめ

- 抽象機械アーキテクチャ
- この機械のための操作的意味論
- While言語から機械語への翻訳
- 定理：正しさが証明された実装