Implementación Correcta

Yolanda Ortega Mallén

Dpto. de Sistemas Informáticos y Computación

Universidad Complutense de Madrid

Sumario

- La máquina abstracta.
- Traducción.
- Corrrección.
- Bisimulación

Configuraciones + Instrucciones

Configuraciones $\langle c, e, s\rangle \in \textbf{Code} \times \textbf{Stack} \times \textbf{State}$

- c código, secuencia de instrucciones
- e pila de evaluación, $\textbf{Stack} = (\mathbb{Z} \cup \mathbf{T})^*$
- s almacén

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

Configuración final $\langle \varepsilon, e, s\rangle$

Semántica operacional

$$
\begin{array}{lcll}
\langle \mathrm{PUSH}\text{-}n : c,\, e,\, s\rangle & \rhd & \langle c,\, \mathcal{N}[\![n]\!] : e,\, s\rangle & \\
\langle \mathrm{ADD} : c,\, z_1 : z_2 : e,\, s\rangle & \rhd & \langle c,\, (z_1 \oplus z_2) : e,\, s\rangle & \text{si } z_1, z_2 \in \mathbb{Z} \\
\langle \mathrm{MULT} : c,\, z_1 : z_2 : e,\, s\rangle & \rhd & \langle c,\, (z_1 \otimes z_2) : e,\, s\rangle & \text{si } z_1, z_2 \in \mathbb{Z} \\
\langle \mathrm{SUB} : c,\, z_1 : z_2 : e,\, s\rangle & \rhd & \langle c,\, (z_1 \ominus z_2) : e,\, s\rangle & \text{si } z_1, z_2 \in \mathbb{Z} \\
\langle \mathrm{TRUE} : c,\, e,\, s\rangle & \rhd & \langle c,\, \mathbf{tt} : e,\, s\rangle & \\
\langle \mathrm{FALSE} : c,\, e,\, s\rangle & \rhd & \langle c,\, \mathbf{ff} : e,\, s\rangle & \\
\langle \mathrm{EQ} : c,\, z_1 : z_2 : e,\, s\rangle & \rhd & \langle c,\, (z_1 = z_2) : e,\, s\rangle & \text{si } z_1, z_2 \in \mathbb{Z} \\
\langle \mathrm{LE} : c,\, z_1 : z_2 : e,\, s\rangle & \rhd & \langle c,\, (z_1 \leq z_2) : e,\, s\rangle & \text{si } z_1, z_2 \in \mathbb{Z}
\end{array}
$$

$$
\langle \mathrm{AND} : c,\, t_1 : t_2 : e,\, s\rangle \quad \rhd \quad \begin{cases} \langle c,\, \mathbf{tt} : e,\, s\rangle & \text{si } t_1 = \mathbf{tt} \text{ y } t_2 = \mathbf{tt} \\ \langle c,\, \mathbf{ff} : e,\, s\rangle & \text{si } t_1 = \mathbf{ff} \text{ o } t_2 = \mathbf{ff},\ \text{y } t_1, t_2 \in \mathbf{T} \end{cases}
$$

$$
\langle \mathrm{NEG} : c,\, t : e,\, s\rangle \quad \rhd \quad \begin{cases} \langle c,\, \mathbf{ff} : e,\, s\rangle & \text{si } t = \mathbf{tt} \\ \langle c,\, \mathbf{tt} : e,\, s\rangle & \text{si } t = \mathbf{ff} \end{cases}
$$

Semántica operacional

$$
\begin{array}{lcl}
\langle \mathrm{FETCH}\text{-}x : c,\, e,\, s\rangle & \rhd & \langle c,\, (s\ x) : e,\, s\rangle \\
\langle \mathrm{STORE}\text{-}x : c,\, z : e,\, s\rangle & \rhd & \langle c,\, e,\, s[x \mapsto z]\rangle \quad \textbf{si } z \in \mathbb{Z} \\
\langle \mathrm{NOOP} : c,\, e,\, s\rangle & \rhd & \langle c,\, e,\, s\rangle \\
\langle \mathrm{BRANCH}(c_1, c_2) : c,\, t : e,\, s\rangle & \rhd & \begin{cases} \langle c_1 : c,\, e,\, s\rangle & \textbf{si } t = \mathbf{tt} \\ \langle c_2 : c,\, e,\, s\rangle & \textbf{si } t = \mathbf{ff} \end{cases} \\
\langle \mathrm{LOOP}(c_1, c_2) : c,\, e,\, s\rangle & \rhd & \langle c_1 : \mathrm{BRANCH}(c_2 : \mathrm{LOOP}(c_1, c_2), \mathrm{NOOP}) : c,\, e,\, s\rangle
\end{array}
$$

Secuencia de cómputo $\gamma_0 \rhd \gamma_1 \rhd \gamma_2 \rhd \cdots \rhd \gamma_k$

- finita / termina
- infinita / cicla

Propiedades

Inducción sobre la longitud de la secuencia de cómputo

Ejercicio 4.4

Demostrar que se puede extender el código y la pila de evaluación de MA sin que cambie su comportamiento:

$$\langle c_1, e_1, s\rangle \rhd^k \langle c', e', s'\rangle \text{ entonces } \langle c_1 : c_2, e_1 : e_2, s\rangle \rhd^k \langle c' : c_2, e' : e_2, s'\rangle$$

Ejercicio 4.5

Demostrar que la ejecución de una secuencia de instrucciones puede fraccionarse:

$\langle c_1 : c_2, e, s\rangle \rhd^k \langle \varepsilon, e'', s''\rangle$ entonces

$\exists\langle \varepsilon, e', s'\rangle\, \exists k_1, k_2. \langle c_1, e, s\rangle \rhd^{k_1} \langle \varepsilon, e', s'\rangle \wedge \langle c_2, e', s'\rangle \rhd^{k_2} \langle \varepsilon, e'', s''\rangle$

con $k = k_1 + k_2$.

Ejercicio 4.6

Demostrar que la semántica de MA es determinista:

$$\gamma \rhd \gamma' \wedge \gamma \rhd \gamma'' \text{ entonces } \gamma' = \gamma''$$

Función semántica

Significado de una secuencia de instrucciones:

$$\mathcal{M} : \textbf{Code} \longrightarrow (\textbf{State} \hookrightarrow \textbf{State})$$

$$\mathcal{M}[\![c]\!]s = \begin{cases} s' & \text{si } \langle c, \varepsilon, s\rangle \rhd^* \langle \varepsilon, e, s'\rangle \\ \text{INDEFINIDO} & \text{e.c.c.} \end{cases}$$

Modificaciones

Ejercicio 4.7

Modificar MA para referirse a las variables por su dirección:

- las configuraciones son $\langle c, e, m\rangle$, donde $m \in \mathbb{Z}^*$ representa la memoria ($m[n]$ selecciona el n-ésimo valor de la lista m);
- sustituir FETCH-x y STORE-x por GET-n y PUT-n, siendo $n \in \mathbb{N}$ una dirección.

Dar la semántica operacional de la máquina modificada MA_1.

Ejercicio 4.8

Modificar MA_1 para introducir instrucciones de salto:

- las configuraciones son $\langle pc, c, e, m\rangle$, donde $pc \in \mathbb{N}$ es el contador de programa que apunta a alguna instrucción en c ($c[pc]$ indica la instrucción en c apuntada por pc);
- sustituir BRANCH$(\ldots,\ldots)$ y LOOP$(\ldots,\ldots)$ por LABEL-l, JUMP-l y JUMPFALSE-l, siendo $l \in \mathbb{N}$ una etiqueta.

Dar la semántica operacional de la máquina modificada MA_2.

Traducción de expresiones

Expresiones aritméticas

$$\mathcal{CA} : \mathbf{Aexp} \longrightarrow \mathbf{Code}$$

$$\begin{array}{rcl} \mathcal{CA}[\![n]\!] & = & \text{PUSH-}n \\ \mathcal{CA}[\![x]\!] & = & \text{FETCH-}x \\ \mathcal{CA}[\![a_1 + a_2]\!] & = & \mathcal{CA}[\![a_2]\!] : \mathcal{CA}[\![a_1]\!] : \text{ADD} \\ \mathcal{CA}[\![a_1 \times a_2]\!] & = & \mathcal{CA}[\![a_2]\!] : \mathcal{CA}[\![a_1]\!] : \text{MULT} \\ \mathcal{CA}[\![a_1 - a_2]\!] & = & \mathcal{CA}[\![a_2]\!] : \mathcal{CA}[\![a_1]\!]\text{SUB} \end{array}$$

Expresiones booleanas

$$\mathcal{CB} : \mathbf{Bexp} \longrightarrow \mathbf{Code}$$

$$\begin{array}{rcl} \mathcal{CB}[\![\mathtt{true}]\!] & = & \text{TRUE} \\ \mathcal{CB}[\![\mathtt{false}]\!] & = & \text{FALSE} \\ \mathcal{CB}[\![a_1 = a_2]\!] & = & \mathcal{CA}[\![a_2]\!] : \mathcal{CA}[\![a_1]\!] : \text{EQ} \\ \mathcal{CB}[\![a_1 \leq a_2]\!] & = & \mathcal{CA}[\![a_2]\!] : \mathcal{CA}[\![a_1]\!] : \text{LE} \\ \mathcal{CB}[\![\neg b]\!] & = & \mathcal{CB}[\![b]\!] : \text{NEG} \\ \mathcal{CB}[\![b_1 \wedge b_2]\!] & = & \mathcal{CB}[\![b_2]\!] : \mathcal{CB}[\![b_1]\!] : \text{AND} \end{array}$$

Traducción de sentencias

$$\mathcal{CS} : \mathbf{Stm} \longrightarrow \mathbf{Code}$$

$$\begin{array}{rcl}
\mathcal{CS}[\![x := a]\!] & = & \mathcal{CA}[\![a]\!] : \textsc{store}\text{-}x \\
\mathcal{CS}[\![\texttt{skip}]\!] & = & \textsc{noop} \\
\mathcal{CS}[\![S_1; S_2]\!] & = & \mathcal{CS}[\![S_1]\!] : \mathcal{CS}[\![S_2]\!] \\
\mathcal{CS}[\![\texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2]\!] & = & \mathcal{CB}[\![b]\!] : \textsc{branch}(\mathcal{CS}[\![S_1]\!], \mathcal{CS}[\![S_2]\!]) \\
\mathcal{CS}[\![\texttt{while } b \texttt{ do } S]\!] & = & \textsc{loop}(\mathcal{CB}[\![b]\!], \mathcal{CS}[\![S]\!])
\end{array}$$

Ejercicio 4.11

Aunque $\mathcal{A}[\![(a_1 + a_2) + a_3]\!] = \mathcal{A}[\![a_1 + (a_2 + a_3)]\!]$, demostrar que, en general, $\mathcal{CA}[\![(a_1 + a_2) + a_3]\!] \neq \mathcal{CA}[\![a_1 + (a_2 + a_3)]\!]$;
sin embargo se *comportan* de forma similar.

Ejercicio 4.14

Extender el lenguaje **While** con la sentencia `repeat` S `until` b y generar código para esta sentencia, sin ampliar el conjunto de instrucciones de MA.

Función semántica

Significado de una sentencia: Traducir a código de MA y ejecutarlo.

$$\mathcal{S}_{\mathrm{ma}} : \mathbf{Stm} \longrightarrow (\mathbf{State} \hookrightarrow \mathbf{State})$$

$$\mathcal{S}_{\mathrm{ma}}[\![S]\!] = (\mathcal{M} \circ \mathcal{CS})[\![S]\!]$$

Ejercicio 4.16

Modificar la generación de código para traducir **While** a MA_1.
Utilizar entornos $env : \mathbf{Var} \longrightarrow \mathbb{N}$ que asocian cada variable a su dirección.

Ejercicio 4.17

Modificar la generación de código para traducir **While** a MA_2.
Garantizar la unicidad de las etiquetas mediante un parámero adicional: siguiente etiqueta sin usar.

Corrección de la implementación de expresiones

Lema 4.18:

$$\forall a \in \mathbf{Aexp}, \forall s \in \mathbf{State}. \langle \mathcal{CA}[\![a]\!], \varepsilon, s\rangle \rhd^* \langle \varepsilon, \mathcal{A}[\![a]\!]s, s\rangle$$

Además, en todas las configuraciones intermedias de esta secuencia de cómputo, la pila de evaluación es no vacía.

Ejercicio 4.19

Demostrar un resultado similar para las expresiones booleanas:

$$\forall b \in \mathbf{Bexp}, \forall s \in \mathbf{State}. \langle \mathcal{CB}[\![b]\!], \varepsilon, s\rangle \rhd^* \langle \varepsilon, \mathcal{B}[\![b]\!]s, s\rangle$$

Demostrar además que en todas las configuraciones intermedias de esta secuencia de cómputo, la pila de evaluación es no vacía.

Corrección de la implementación de sentencias

Teorema 4.20:

$$\forall S \in \mathbf{Stm}. \mathcal{S}_{\mathrm{bs}}[\![S]\!] = \mathcal{S}_{\mathrm{ma}}[\![S]\!]$$

Lema 4.21:

$$\forall S \in \mathbf{Stm}, \forall s, s' \in \mathbf{State}. \langle S, s\rangle \to s' \text{ entonces } \langle \mathcal{CS}[\![S]\!], \varepsilon, s\rangle \rhd^* \langle \varepsilon, \varepsilon, s'\rangle$$

Lema 4.22:

$$\forall S \in \mathbf{Stm}, \forall s, s' \in \mathbf{State}. \langle \mathcal{CS}[\![S]\!], \varepsilon, s\rangle \rhd^k \langle \varepsilon, e, s'\rangle \text{ entonces } \langle S, s\rangle \to s' \wedge e = \varepsilon$$

Resumen demostración corrección

1. Inducción sobre el árbol de derivación
   Para cada árbol de derivación en la semántica de paso largo existe la correspondiente secuencia de cómputo finita en la máquina abstracta.
2. Inducción sobre la longitud de la secuencia de cómputo
   Para cada secuencia de cómputo finita obtenida al ejecutar una sentencia del lenguaje **While** en la máquina abstracta existe el correspondiente árbol de derivación en la semántica de paso largo.

Corrección de la implementación de sentencias

Ejercicio 4.23

Modificar la función de traducción de forma que $\mathcal{CS}[\![\texttt{skip}]\!] = \varepsilon$.
¿Cómo afecta este cambio a la demostración del Teorema 4.20?

Ejercicio 4.24

Extender la demostración del Teorema 4.20 para incluir la sentencia
$\texttt{repeat}\ S\ \texttt{until}\ b$.

Ejercicio 4.25

Demostrar la corrección del código generado para MA_1.
¿Qué hay que asumir para env?

Ejercicio 4.26

Suponer que la pila de evaluación solamente contiene valores enteros, de forma que $\mathbf{ff}$ y $\mathbf{tt}$ se representan mediante $\mathbf{0}$ y $\mathbf{1}$.
Modificar adecuadamente todo lo visto en este tema.

Demostración alternativa de corrección

- Utilizar la semántica de paso corto.
- Definir una relación de bisimulación:

$$\begin{aligned} \langle S, s\rangle &\approx \langle \mathcal{CS}[\![S]\!], \varepsilon, s\rangle \\ s &\approx \langle \varepsilon, \varepsilon, s\rangle \end{aligned}$$

Ejercicio 4.27 + 4.28

1. Demostrar que a cada paso de la semántica de paso corto le corresponde una secuencia de pasos en la máquina abstracta:

$$\gamma_{\mathrm{ss}} \approx \gamma_{\mathrm{ma}} \wedge \gamma_{\mathrm{ss}} \Rightarrow \gamma'_{\mathrm{ss}} \text{ entonces } \exists \gamma'_{\mathrm{ma}} \,.\, \gamma'_{\mathrm{ss}} \approx \gamma'_{\mathrm{ma}} \wedge \gamma_{\mathrm{ma}} \rhd^{+} \gamma'_{\mathrm{ma}}$$

   Inferir que $\langle S, s\rangle \Rightarrow^* s'$ entonces $\langle \mathcal{CS}[\![S]\!], \varepsilon, s\rangle \rhd^* \langle \varepsilon, \varepsilon, s'\rangle$

2. Consideramos secuencias de cómputo que parten de una pila de evaluación vacía y terminan con la pila vacía.
   Asumiendo que $\gamma_{\mathrm{ss}} \approx \gamma^1_{\mathrm{ma}}$ y que $\gamma^1_{\mathrm{ma}} \rhd \gamma^2_{\mathrm{ma}} \rhd \cdots \rhd \gamma^k_{\mathrm{ma}}$, con $k > 1$ y la pila de evaluación solo es vacía en γ^1_{ma} y γ^k_{ma}, demostrar que existe γ'_{ss} tal que

$$\gamma'_{\mathrm{ss}} \approx \gamma^k_{\mathrm{ma}} \wedge \gamma_{\mathrm{ss}} \Rightarrow \gamma'_{\mathrm{ss}}$$

   Inferir que $\langle \mathcal{CS}[\![S]\!], \varepsilon, s\rangle \rhd^* \langle \varepsilon, \varepsilon, s'\rangle$ entonces $\langle S, s\rangle \Rightarrow^* s'$

Demostración alternativa de corrección

$$\forall S \in \mathbf{Stm}.\mathcal{S}_{\text{ss}}[\![S]\!] = \mathcal{S}_{\text{ma}}[\![S]\!]$$

Resumen demostración corrección con bisimulación

1. Un paso en la semántica de paso corto puede simularse mediante una secuencia no vacía de pasos en la máquina abstracta.
   Extender a secuencias de pasos en la semántica de paso corto.
2. Una secuencia no vacía de pasos en la máquina abstracta que cumple algunas restricciones puede simularse mediante un paso en la semántica de paso corto. Extender a secuencias más generales en la máquina abstracta.

Ejercicio 4.30

Sustituir la regla para los bucles en la semántica de paso corto por dos axiomas:

$$\begin{array}{ll} \langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \Rightarrow \langle S; \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle & \text{si } \mathcal{B}[\![b]\!]s = \mathbf{tt} \\ \langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \Rightarrow s & \text{si } \mathcal{B}[\![b]\!]s = \mathbf{ff} \end{array}$$

Demostrar que la nueva semántica es equivalente a la antigua:

$$\forall S \in \mathbf{Stm}.\mathcal{S}_{\text{ss}}[\![S]\!] = \mathcal{S}'_{\text{ss}}[\![S]\!]$$

¿Se complican / simplifican las demostraciones de corrección?