Implementación Correcta

David de Frutos Escrig

versión original elaborada por

Yolanda Ortega Mallén

Dpto. de Sistemas Informáticos y Computación

Universidad Complutense de Madrid

Sumario

- La máquina abstracta.
- Traducción: generación de código para la máquina.
- Corrección de la traducción.

Bibliografía

- Hanne Riis Nielson & Flemming Nielson,
  Semantics with Applications. An Appetizer, Springer, 2007.
  Capítulo 4.

Configuraciones + Instrucciones

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

Código c secuencia de instrucciones

Pila de evaluación $e \in \mathbf{Stack} = (\mathbb{Z} \cup \mathbf{T})^*$

Almacén s (Para While no necesitamos explicitar la memoria)

$$
\begin{array}{lcl}
inst & ::= & \textsc{push}\text{-}n \mid \textsc{add} \mid \textsc{mult} \mid \textsc{sub} \\
 & \mid & \textsc{true} \mid \textsc{false} \mid \textsc{eq} \mid \textsc{le} \mid \textsc{neg} \mid \textsc{and} \\
 & \mid & \textsc{fetch}\text{-}x \mid \textsc{store}\text{-}x \\
 & \mid & \textsc{noop} \mid \textsc{branch}(c,c) \mid \textsc{loop}(c,c) \\
c & ::= & \varepsilon \mid inst : c
\end{array}
$$

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

Funcionamiento de la máquina abstracta

Instrucciones de manejo de la pila (evaluación de expresiones)

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

$$\langle \text{ADD} : c, z_1 : z_2 : e, s\rangle \rhd \langle c, (z_1 \oplus z_2) : e, s\rangle \quad \text{si } z_1, z_2 \in \mathbb{Z}$$

$$\langle \text{MULT} : c, z_1 : z_2 : e, s\rangle \rhd \langle c, (z_1 \otimes z_2) : e, s\rangle \quad \text{si } z_1, z_2 \in \mathbb{Z}$$

$$\langle \text{SUB} : c, z_1 : z_2 : e, s\rangle \rhd \langle c, (z_1 \ominus z_2) : e, s\rangle \quad \text{si } z_1, z_2 \in \mathbb{Z}$$

$$\langle \text{TRUE} : c, e, s\rangle \rhd \langle c, \mathbf{tt} : e, s\rangle$$

$$\langle \text{FALSE} : c, e, s\rangle \rhd \langle c, \mathbf{ff} : e, s\rangle$$

$$\langle \text{EQ} : c, z_1 : z_2 : e, s\rangle \rhd \langle c, (z_1 = z_2) : e, s\rangle \quad \text{si } z_1, z_2 \in \mathbb{Z}$$

$$\langle \text{LE} : c, z_1 : z_2 : e, s\rangle \rhd \langle c, (z_1 \leq z_2) : e, s\rangle \quad \text{si } z_1, z_2 \in \mathbb{Z}$$

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

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

Funcionamiento de la máquina abstracta

Instrucciones de manejo de la memoria (acceso a variables)

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

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

Instrucciones estructuradas de control

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

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

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

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

- finita $\Leftrightarrow$ termina
- infinita $\Leftrightarrow$ cicla

Propiedades

Ejercicio 4.4

Demostrar que se puede extender al tiempo el código y la pila de evaluación de MA, sin que cambie su comportamiento, en tanto se va ejecutando dicho código:

$$\langle c_1, e_1, s\rangle \rhd^k \langle c', e', s'\rangle \implies \langle c_1 : c_2, e_1 : e_2, s\rangle \rhd^k \langle c' : c_2, e' : e_2, s'\rangle$$

Ejercicio 4.5

Demostrar que la ejecución completa de una secuencia de instrucciones siempre puede fraccionarse, instrucción a instrucción:

$$\langle c_1 : c_2, e, s\rangle \rhd^k \langle \varepsilon, e'', s''\rangle \implies$$
$$\exists \langle \varepsilon, e', s'\rangle \, \exists k_1, k_2 \;\; \langle c_1, e, s\rangle \;\; \rhd^{k_1} \langle \varepsilon, e', s'\rangle \wedge \langle c_2, e', s'\rangle \;\; \rhd^{k_2} \langle \varepsilon, e'', s''\rangle$$
$$\text{con} \;\; k = k_1 + k_2.$$

Inducción sobre la longitud de la secuencia de cómputo

Ejercicio 4.6

Demostrar que la semántica de MA es determinista:

$$\gamma \rhd \gamma' \;\; \wedge \;\; \gamma \rhd \gamma'' \implies \gamma' = \gamma''$$

Función semántica

Significado de una secuencia de instrucciones:

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

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

Modificaciones de la máquina MA

Ejercicio 4.7

Modificar MA para referirse a las variables por su dirección:

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

Dar la semántica operacional de la máquina modificada MA_1.

Ejercicio 4.8

Modificar MA_1 para pasar a tener instrucciones de salto no estructuradas:

- las configuraciones pasan a ser $\langle pc, c, e, m \rangle$, donde $pc \in \mathbb{N}$ es el contador de programa, que apunta a la siguiente instrucción a ejecutar en c ($c[pc]$).
- sustituir BRANCH$(\ldots,\ldots)$ y LOOP$(\ldots,\ldots)$ por LABEL-l, JUMP-l y JUMPFALSE-l, representando LABEL-l, con $l \in \mathbb{N}$, una posición en c.

Dar la semántica operacional de la máquina modificada MA_2.

Traducción de expresiones de While

Expresiones aritméticas

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

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

Expresiones booleanas

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

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

Corrección de la implementación de expresiones

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, podemos capturar en qué sentido se *comportan* de forma *suficientemente* similar.

Lema 8:

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

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:

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

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

Traducción de sentencias de While

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

$$\begin{aligned}
\mathcal{CS}[\![x := a]\!] &= \mathcal{CA}[\![a]\!] : \text{STORE-}x \\
\mathcal{CS}[\![\texttt{skip}]\!] &= \text{NOOP} \\
\mathcal{CS}[\![S_1; S_2]\!] &= \mathcal{CS}[\![S_1]\!] : \mathcal{CS}[\![S_2]\!] \\
\mathcal{CS}[\![\texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2]\!] &= \mathcal{CB}[\![b]\!] : \text{BRANCH}(\mathcal{CS}[\![S_1]\!], \mathcal{CS}[\![S_2]\!]) \\
\mathcal{CS}[\![\texttt{while } b \texttt{ do } S]\!] &= \text{LOOP}(\mathcal{CB}[\![b]\!], \mathcal{CS}[\![S]\!])
\end{aligned}$$

Ejercicio 4.14

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

Ejercicio 4.16

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

Ejercicio 4.17

Generar el código para traducir **While** a la máquina MA_2.
Garantizar la unicidad de las etiquetas incorporando un parámetro adicional:
siguiente etiqueta sin usar.

Corrección de la implementación de sentencias

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

$$\mathcal{S}_{\text{ma}} : \textbf{Stm} \longrightarrow (\textbf{State} \hookrightarrow \textbf{State}) \quad \mathcal{S}_{\text{ma}}[\![S]\!] = (\mathcal{M} \circ \mathcal{CS})[\![S]\!]$$

Teorema 9:

$$\forall S \in \textbf{Stm} \quad \mathcal{S}_{\text{bs}}[\![S]\!] = \mathcal{S}_{\text{ma}}[\![S]\!]$$

Lema 10: $\forall S \in \textbf{Stm}, \forall s, s' \in \textbf{State} \quad \langle S, s\rangle \to s' \implies \langle \mathcal{CS}[\![S]\!], \varepsilon, s\rangle \rhd^* \langle \varepsilon, \varepsilon, s'\rangle$

Lema 11: $\forall S \in \textbf{Stm}, \forall s, s' \in \textbf{State}$

$$\langle \mathcal{CS}[\![S]\!], \varepsilon, s\rangle \rhd^k \langle \varepsilon, e, s'\rangle \implies \langle S, s\rangle \to s' \wedge e = \varepsilon$$

Resumen de la demostración de 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 pasando a tener $\mathcal{CS}[\![\texttt{skip}]\!] = \varepsilon$.

¿Cómo afecta este cambio a la demostración del Teorema 9?

Ejercicio 4.24

Extender la demostración del Teorema 9 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 sobre el valor de env?

Ejercicio 4.26

Suponer que la pila de evaluación solamente puede contener valores enteros. En consecuencia, tendremos que representar $\mathbf{ff}$ y $\mathbf{tt}$ mediante $\mathbf{0}$ y $\mathbf{1}$.

Introducir todos los cambios necesarios para que la traducción de While siga funcionando correctamente.