Wejście - wyjście: semantyka denotacyjna

Rozszerzamy składnię języka o wejście-wyjście:

Składnia

- $e ::= m \mid v \mid (e + e') \mid (e - e') \mid (e * e') \mid$ read
- $b ::= t \mid b \vee b' \mid e = e' \mid \neg b$
- $d ::=$ var $v \mid d_1; d_2$
- $c ::=$ skip $\mid v := e \mid c; c' \mid$ if b then c else $c' \mid$ while b do $c \mid$ begin $d; c$ end $\mid$ write(e)

Modyfikujemy dziedziny semantyczne. Dotychczas mieli-smy

$$S = [Loc \to Sv + \{\ \underline{\text{wolna}}\ \}] - \text{stany (=pamięć)}$$

Teraz rozszerzamy stany o wejście i wyjście. Mamy:

- $Val = \{N + B + \{\perp\}\}$
- $In = Val^*$
- $Out = Val^*$
- pusty ciąg wartości będziemy oznaczać ϵ
- $S = M \times In \times Out$
- $M = [Loc \to Sv + \{\ \underline{\text{wolna}}\ \}] - \text{stany (pamięć)}$

Będziemy używać pomocniczych funkcji:

- $hd : Val^* \to Val$
- $tl : Val^* \to Val*$

Funkcje semantyczne

$$\mathcal{A} : Aexp \to [Env \to [S \to N_\perp \times In]]$$

$$\mathcal{B} : Bexp \to [Env \to [S \to B_\perp \times In]]$$

$$\mathcal{D} : Decl \to [Env \to [M \to Env \times M]]$$

$$\mathcal{C} : Com \to [Env \to [S \to S_\perp]]$$

Równania semantyczne

Aexp

$$\mathcal{A}[[n]] = \lambda\rho \in Env.\lambda s \in S.(n, (s)_2)$$

$$\mathcal{A}[[v]] = \lambda\rho \in Env.\lambda(m,i,o) \in S.\textbf{if } \rho v \notin Loc \textbf{ then } \bot \textbf{ else}$$
$$\textbf{if } m(\rho v) = \bot_N \textbf{ then } \bot_N \textbf{ else } (m(\rho v), i)$$

$$\mathcal{A}[[\ \underline{\text{read}}\ ]] = \lambda\rho \in Env.\lambda s \in S.\textbf{if } i = \epsilon \textbf{ then } \bot \textbf{ else } (\mathit{hd}(i), \mathit{tl}(i))$$

$$\mathcal{A}[[e_0 + e_1]] = \lambda\rho \in Env.\lambda(m,i,o) \in S.\textbf{let } (w', i') = \mathcal{A}[[e_0]]\rho(m,i,o)$$
$$\textbf{in if } w' = \bot_N \textbf{ then } \bot \textbf{ else let } (w'', i'') = \mathcal{A}[[e_1]]\rho(m,i',o)$$
$$\textbf{in if } w'' = \bot_N \textbf{ then } \bot \textbf{ else } (w' + w'', i'')$$

Pozostałe wyrażenia analogicznie.

Decl

$$\mathcal{D}[[\ \underline{\text{var}}\ v]] = \lambda\rho \in Env.\lambda s \in S.(\rho[(new(s))_1/v], (new(s))_2)$$

$$\mathcal{D}[[d_1; d_2]] = \lambda\rho \in Env.\lambda s \in S.\mathcal{D}[[d_2]](\mathcal{D}[[d_1]]\rho s)_1(\mathcal{D}[[d_1]]\rho s)_2$$

Com

$$\mathcal{C}[[\underline{\text{skip}}]] = \lambda\rho \in Env.\lambda s \in S.s$$

$$\mathcal{C}[[v := e]] = \lambda\rho \in Env.\lambda(m,i,o) \in S.\textbf{if } \rho v \notin Loc \textbf{ then } \perp_S$$
$$\textbf{else if } (\mathcal{A}[[e]]\rho(m,i,o))_1 = \perp_N \textbf{ then } \perp_S \textbf{ else}$$
$$(m[(\mathcal{A}[[e]]\rho(m,i,o))_1/\rho v], (\mathcal{A}[[e]]\rho(m,i,o))_2, o)$$

$$\mathcal{C}[[\ \underline{\text{write}}(e)\ ]] = \lambda\rho \in Env.\lambda(m,i,o) \in S.\textbf{let } (w,i') = \mathcal{A}[[e]]\rho(m,i,o)$$
$$\textbf{in if } w = \perp_N \textbf{ then } \perp_S \textbf{ else } (m,i',w.o)$$

Pozostałe instrukcje analogicznie.