Laboratorio 7 - Transizioni SOS di Espressioni, Comandi Implementazione

Sommario: 17 maggio, 2019

- Raccomandazioni e Preliminari
- Linguaggio SmallC: Sintassi Astratta
- Sintassi Concreta: Una Grammatica EBNF per SmallC
- Linguaggio SmallC: Semantica SOS
- Linguaggio SmallC: Esecutore
- Varianti, Estensioni, Uso.
- Attività di Oggi: Esercizi 18, 19, 20

SMallC: All'inizio di Ogni Sessione

- Caricare il file SmallC, distribuito per la sessione, sulla WAD.
  Verifica: Caricarlo sul TLE OCaml e Controllare Esecuzione Tests Precedenti.
- Copiarlo in SmallC6.
- Seguire la Presentazione delle Attività della Sessione.
- Svogere le Attività modificando il file SmallC distribuito.

SmallC: Sintassi Astratta

$$\texttt{Dcl} ::= [\texttt{const}]\ \texttt{Ide}\ \texttt{Num} \mid [\texttt{var}]\ \texttt{Ide}\ \texttt{Num} \mid [\texttt{varN}]\ \texttt{Ide} \mid [\texttt{array}]\ \texttt{Ide}\ \texttt{Num} \mid [\texttt{emptyDCL}] \mid \texttt{Dcl}\ [\texttt{seqD}]\ \texttt{Dcl}$$

$$\begin{aligned}\texttt{Exp} ::= {} & [\texttt{val}]\ \texttt{Ide} \mid \texttt{Num} \mid \texttt{Ide}\ [\uparrow]\ \texttt{Exp} \\ & \mid \texttt{Exp}\ [+]\ \texttt{Exp} \mid \texttt{Exp}\ [-]\ \texttt{Exp} \mid \texttt{Exp}\ [*]\ \texttt{Exp} \mid \texttt{Exp}\ [\texttt{div}]\ \texttt{Exp} \\ & \mid \texttt{Exp}\ [==]\ \texttt{Exp} \mid \texttt{Exp}\ [<]\ \texttt{Exp} \mid \texttt{Exp}\ [>]\ \texttt{Exp} \\ & \mid [\texttt{not}]\ \texttt{Exp} \mid \texttt{Exp}\ [\texttt{or}]\ \texttt{Exp} \mid \texttt{Exp}\ [\texttt{and}]\ \texttt{Exp}\end{aligned}$$

$$\begin{aligned}\texttt{Cmd} ::= {} & [\texttt{ifE}]\ \texttt{Exp}\ \texttt{Cmd}\ \texttt{Cmd} \mid [\texttt{ifT}]\ \texttt{Exp}\ \texttt{Cmd} \\ & \mid \texttt{Ide}\ [=]\ \texttt{Exp} \mid \texttt{Ide}\ \texttt{Exp}\ [\leftarrow]\ \texttt{Exp} \\ & \mid [\texttt{while}]\ \texttt{Exp}\ \texttt{Cmd} \mid \texttt{Cmd}\ [\texttt{seqC}]\ \texttt{Cmd} \mid [\texttt{emptyCMD}]\end{aligned}$$

$$\texttt{Prog} ::= [\texttt{prog}]\ \texttt{Dcl}\ \texttt{Cmd} \mid [\texttt{progN}]\ \texttt{Cmd}$$

where:
– [xxx] indica indica un costruttore di alberi AST di nome xxx;
– $\uparrow$ costruttore termine (AST) accesso valore componente array;
– $\leftarrow$ costruttore termine modifica valore componente array;

Sintassi Concreta: Una Grammatica EBNF per SmallC

$$\texttt{Dcl} ::= \texttt{final ide = num} \mid \texttt{var ide = num} \mid \texttt{var ide} \mid \texttt{array ide[num]}$$
$$\texttt{Dcls} ::= \epsilon \mid \texttt{Dcl;Dcls}$$

$$\texttt{ExpB2} ::= \texttt{ExpB2 or ExpB1} \mid \texttt{ExpB1}$$
$$\texttt{ExpB1} ::= \texttt{ExpB1 and ExpB} \mid \texttt{ExpB}$$
$$\texttt{ExpB} ::= \texttt{not ExpB} \mid \texttt{ExpR}$$
$$\texttt{ExpR} ::= \texttt{ExpR} == \texttt{ExpA2} \mid \texttt{ExpR} < \texttt{ExpA2} \mid \texttt{ExpR} > \texttt{ExpA2} \mid \texttt{ExpA2}$$
$$\texttt{ExpA2} ::= \texttt{ExpA2} + \texttt{ExpA1} \mid \texttt{ExpA2} - \texttt{ExpA1} \mid \texttt{ExpA1}$$
$$\texttt{ExpA1} ::= \texttt{ExpA1} * \texttt{ExpA} \mid \texttt{ExpA1 div ExpA} \mid \texttt{ExpA}$$
$$\texttt{ExpA} ::= \texttt{ide} \mid \texttt{num} \mid \texttt{ide[ExpA2]} \mid \texttt{(ExpB2)}$$

$$\texttt{Cmd} ::= \texttt{if (ExpB2) Cmd; else Cmd;} \mid \texttt{OtherCmd;}$$
$$\texttt{OtherCmd} ::= \texttt{if (ExpB2) OtherCmd} \mid \texttt{NonCondionalCmd}$$
$$\texttt{NonCondionalCmd} ::= \texttt{ide = ExpA2} \mid \texttt{Ide[ExpA2] = ExpA2}$$
$$\mid \texttt{while (ExpB2)} \widehat{\{} \texttt{ Cmd } \widehat{\}} \mid \{\texttt{Cmd}\}$$
$$\texttt{Cmds} ::= \texttt{Cmd} \mid \texttt{Cmd Cmds}$$

$$\texttt{Prog} ::= \widehat{\{} \texttt{ Dcls } \{\texttt{Cmd}\} \widehat{\}}$$

where: `ide`,`=`,`num`,`var`,`[`, `]`,`;`,`or`,`and`,`not`,`==`,`<`,`>`,`+`,`-`,`*`,`div`,`(`,`)`,`if`,`then`,`else`,`while`,$\widehat{\{}$,$\widehat{\}}$ sono categorie lessicali (contenenti 1 solo lessema, ad eccezione di ide e num): $\widehat{\{}$ ha lessema "{", $\widehat{\}}$ ha lessema "}".

SmallC: Semantica SOS

- **Transizione**.
  - Esecuzione di un costrutto c nello stato σ della Macchina Astratta
  - La esprimiamo con:
    - $\langle \mathtt{c}, \sigma \rangle \rightarrow \sigma'$, indicante ...
    - $\langle \mathtt{c}, \sigma \rangle \rightarrow \langle \mathtt{c}', \sigma' \rangle$, indicante ...
    - $\langle \mathtt{c}_1, \sigma_1 \rangle \rightarrow \langle \mathtt{c}'_1, \sigma'_1 \rangle, \cdots, \langle \mathtt{c}_k, \sigma_k \rangle \rightarrow \langle \mathtt{c}'_k, \sigma'_k \rangle \ / \ \langle \mathtt{c}, \sigma \rangle \rightarrow \langle \mathtt{c}', \sigma' \rangle$, k-premesse/1-conclusione, indicante ...

- **Compurazione** La sequenza $\sigma_1, ..., \sigma_k, ...$ degli stati effettivamente calcolati dalle transizioni usate nella valutazione/esecuzione del programma.

- **Notazione**.
  - (ρ, μ) stato con ambiente ρ (anche con apici/pedici) e memoria μ (anche con apici/pedici)
  - N (anche con pedici) intero, I (anche con pedici) identificatore, D (anche con pedici) valore denotabile
  - $[\mathtt{I}/\mathtt{D}] \circ \rho$ crea un nuovo ambiente che estende ρ con il binding `[I/D]`
  - $\rho(\mathtt{I})$ denotazione del binding di `I` in ρ, se esiste
  - $\perp_{\mathtt{mem}}$ indefinito nei memorizzabili
  - $\mathtt{allocate}(\mu, \mathtt{n})$ alloca `n` locazioni libere, in sequenza, a partire da `loc`, modificando μ in μ' e restituisce $(\mathtt{loc}, \mu')$
  - $\mu(\mathtt{loc})$ (`loc` deve essere una locazione giá allocata) restituisce il valore di `loc`
  - $\mu[\mathtt{loc} \leftarrow \mathtt{n}]$ (`loc` deve essere giá allocata) modifica il valore di `loc` con il valore `n`
  - $\mathtt{loc} \oplus \mathtt{k}$ locazione k posizioni dopo loc.
  - `true, false` le costanti booleane, `1,0` gli interi utilizzati in SmallC per le costanti booleane
  - `[+],[-],[*],[div],[==],[>],[<]` operazione somma, sottrazione,..., minore_di su interi

Dichiarazioni SMallC: Le Transizioni $\mathrm{SEM}_{\mathrm{DCL}}$

$$\mathtt{Dcl} ::= [\mathtt{const}]\ \mathtt{Ide}\ \mathtt{Num} \mid [\mathtt{var}]\ \mathtt{Ide}\ \mathtt{Num} \mid [\mathtt{varN}]\ \mathtt{Ide} \mid [\mathtt{array}]\ \mathtt{Ide}\ \mathtt{Num} \mid [\mathtt{emptyDCL}] \mid \mathtt{Dcl}\ [\mathtt{seqD}]\ \mathtt{Dcl}$$

- Il sistema di regole $\mathrm{Sem}_{\mathtt{Dcl}}$, sotto riportato, definisce il comportamento delle dichiarazioni SmallC durante la computazione dei Programmi.

$$\frac{}{\langle [\mathtt{const}]\ \mathtt{I}\ \mathtt{N}, (\rho, \mu)\rangle \rightarrow ([\mathtt{I}/\mathtt{N}]\circ\rho, \mu)}$$

$$\frac{\mathtt{allocate}(\mu, 1) = (\mathtt{loc}, \mu')}{\langle [\mathtt{var}]\ \mathtt{I}\ \mathtt{N}, (\rho, \mu)\rangle \rightarrow ([\mathtt{I}/\mathtt{loc}]\circ\rho, \mu'[\mathtt{loc} \leftarrow \mathtt{N}])} \qquad \frac{\mathtt{allocate}(\mu, 1) = (\mathtt{loc}, \mu')}{\langle [\mathtt{varN}]\ \mathtt{I}, (\rho, \mu)\rangle \rightarrow ([\mathtt{I}/\mathtt{loc}]\circ\rho, \mu')}$$

$$\frac{\mathtt{allocate}(\mu, \mathtt{N}) = (\mathtt{loc}, \mu')}{\langle [\mathtt{array}]\ \mathtt{I}\ \mathtt{N}, (\rho, \mu)\rangle \rightarrow ([\mathtt{I}/\mathtt{loc}]\circ\rho, \mu')}$$

$$\frac{}{\langle [\mathtt{emptyDCL}], \sigma\rangle \rightarrow \sigma} \qquad \frac{\langle \mathtt{d}_1, (\rho, \mu)\rangle \rightarrow (\rho', \mu')}{\langle \mathtt{d}_1\ [\mathtt{seqD}]\ \mathtt{d}_2, (\rho, \mu)\rangle \rightarrow \langle \mathtt{d}_2, (\rho', \mu')\rangle}$$

Espressioni SMallC: Le Transizioni $\mathrm{Sem}_{\mathrm{EXP}}$ - Parte1

$$\mathtt{Exp} ::= [\mathtt{val}]\ \mathtt{Ide} \mid [\mathtt{num}]\ \mathtt{Num} \mid \mathtt{Ide}\,[\uparrow]\,\mathtt{Exp}$$

$$\frac{\rho(\mathtt{I}) = \mathtt{N}}{\langle [\mathtt{val}]\,\mathtt{I}, (\rho, \mu)\rangle \to \mathtt{N}} \qquad \frac{\rho(\mathtt{I}) = \mathtt{loc} \qquad \mu(\mathtt{loc}) = \mathtt{N}}{\langle [\mathtt{val}]\,\mathtt{I}, (\rho, \mu)\rangle \to \mathtt{N}}$$

$$\frac{}{\langle [\mathtt{num}]\,\mathtt{N}, \sigma\rangle \to \mathtt{N}}$$

$$\frac{\rho(\mathtt{I}) = \mathtt{loc}' \qquad \langle \mathtt{e}, (\rho, \mu)\rangle \to \mathtt{N}' \qquad \mathtt{loc}' \oplus \mathtt{N}' = \mathtt{loc} \qquad \mu(\mathtt{loc}) = \mathtt{N}}{\langle \mathtt{I}\,[\uparrow]\,\mathtt{e}, (\rho, \mu)\rangle \to \mathtt{N}}$$

$$\mid \mathtt{Exp}\,[+]\,\mathtt{Exp} \mid \mathtt{Exp}\,[-]\,\mathtt{Exp} \mid \mathtt{Exp}\,[*]\,\mathtt{Exp} \mid \mathtt{Exp}\,[\mathtt{div}]\,\mathtt{Exp}$$

$$\frac{\begin{array}{c}\langle \mathtt{e}_1, \sigma\rangle \to \mathtt{N}_1 \qquad \langle \mathtt{e}_2, \sigma\rangle \to \mathtt{N}_2 \\ \mathtt{N}_1\,[\mathtt{op}]\,\mathtt{N}_2 = \mathtt{N} \qquad \mathtt{op} \in \{+, -, *, \mathtt{div}\}\end{array}}{\langle \mathtt{e}_1\,[\mathtt{op}]\,\mathtt{e}_2, \sigma\rangle \to \mathtt{N}}$$

Espressioni SMallC: Le Transizioni Sem_{EXP} - Parte2

$$\mid \text{Exp}\,[==]\,\text{Exp} \mid \text{Exp}\,[<]\,\text{Exp} \mid \text{Exp}\,[>]\,\text{Exp}$$

$$\frac{\langle e_1, \sigma\rangle \to N_1 \qquad \langle e_2, \sigma\rangle \to N_2 \qquad N_1\,[op]\,N_2 = \texttt{true} \qquad op \in \{==, <, >\}}{\langle e_1\,[op]\,e_2, \sigma\rangle \to 1}$$

$$\frac{\langle e_1, \sigma\rangle \to N_1 \qquad \langle e_2, \sigma\rangle \to N_2 \qquad N_1\,[op]\,N_2 = \texttt{false} \qquad op \in \{==, <, >\}}{\langle e_1\,[op]\,e_2, \sigma\rangle \to 0}$$

$$\mid [\text{not}]\,\text{Exp} \mid \text{Exp}\,[\text{or}]\,\text{Exp} \mid \text{Exp}\,[\text{and}]\,\text{Exp}$$

$$\frac{\langle e, \sigma\rangle \to 0}{\langle [\text{not}]\ e, \sigma\rangle \to 1} \qquad \frac{\langle e, \sigma\rangle \to 1}{\langle [\text{not}]\ e, \sigma\rangle \to 0}$$

$$\frac{\langle e_1, \sigma\rangle \to N_1 \qquad \langle e_2, \sigma\rangle \to N_2 \qquad N_1 = 1}{\langle e_1\,[\text{or}]\,e_2, \sigma\rangle \to 1} \qquad \frac{\langle e_1, \sigma\rangle \to N_1 \qquad \langle e_2, \sigma\rangle \to N_2 \qquad N_2 = 1}{\langle e_1\,[\text{or}]\,e_2, \sigma\rangle \to 1} \qquad \frac{\langle e_1, \sigma\rangle \to N_1 \qquad \langle e_2, \sigma\rangle \to N_2 \qquad N_1 = 0 \qquad N_2 = 0}{\langle e_1\,[\text{or}]\,e_2, \sigma\rangle \to 0}$$

$$\frac{\langle e_1, \sigma\rangle \to N_1 \qquad \langle e_2, \sigma\rangle \to N_2 \qquad N_1 = 0}{\langle e_1\,[\text{and}]\,e_2, \sigma\rangle \to 0} \qquad \frac{\langle e_1, \sigma\rangle \to N_1 \qquad \langle e_2, \sigma\rangle \to N_2 \qquad N_2 = 0}{\langle e_1\,[\text{and}]\,e_2, \sigma\rangle \to 0} \qquad \frac{\langle e_1, \sigma\rangle \to N_1 \qquad \langle e_2, \sigma\rangle \to N_2 \qquad N_1 = 1 \qquad N_2 = 1}{\langle e_1\,[\text{and}]\,e_2, \sigma\rangle \to 1}$$

Comandi SMallC: Le Transizioni $\mathtt{Sem}_{\mathtt{CMD}}$ - Parte1

$\mathtt{Cmd} ::= \mathtt{[ifE]\ Exp\ Cmd\ Cmd} \mid \mathtt{[ifT]\ Exp\ Cmd}$

$$\frac{\langle \mathtt{e}, (\rho, \mu)\rangle \to \mathtt{1} \qquad \langle \mathtt{c}_1, (\rho, \mu)\rangle \to (\rho, \mu')}{\langle \mathtt{[ifE]\ e\ c_1\ c_2}, (\rho, \mu)\rangle \to (\rho, \mu')}$$

$$\frac{\langle \mathtt{e}, (\rho, \mu)\rangle \to \mathtt{0} \qquad \langle \mathtt{c}_2, (\rho, \mu)\rangle \to (\rho, \mu')}{\langle \mathtt{[ifE]\ e\ c_1\ c_2}, (\rho, \mu)\rangle \to (\rho, \mu')}$$

$$\frac{\langle \mathtt{e}, (\rho, \mu)\rangle \to \mathtt{1} \qquad \langle \mathtt{c}, (\rho, \mu)\rangle \to (\rho, \mu')}{\langle \mathtt{[ifT]\ e\ c}, (\rho, \mu)\rangle \to (\rho, \mu')}$$

$$\frac{\langle \mathtt{e}, \sigma\rangle \to \mathtt{0}}{\langle \mathtt{[ifT]\ e\ c}, (\rho, \mu)\rangle \to \sigma}$$

$\mid \mathtt{Ide\ [=]\ Exp} \mid \mathtt{Ide\ Exp\ [\leftarrow]\ Exp} \mid \mathtt{Cmd\ [seqC]\ Cmd}$

$$\frac{\rho(\mathtt{I}) = \mathtt{loc} \qquad \langle \mathtt{e}, (\rho, \mu)\rangle \to \mathtt{N} \qquad \mu[\mathtt{loc} \leftarrow \mathtt{N}] = \mu'}{\langle \mathtt{I\ [=]\ e}, (\rho, \mu)\rangle \to (\rho, \mu')}$$

$$\frac{\rho(\mathtt{I}) = \mathtt{loc} \qquad \langle \mathtt{e}_1, (\rho, \mu)\rangle \to \mathtt{N}_1 \qquad \langle \mathtt{e}_2, (\rho, \mu)\rangle \to \mathtt{N}_2 \qquad \mathtt{loc} \oplus \mathtt{N}_1 = \mathtt{loc}' \qquad \mu[\mathtt{loc}' \leftarrow \mathtt{N}_2] = \mu'}{\langle \mathtt{I\ e_1[\leftarrow]\ e_2}, (\rho, \mu)\rangle \to (\rho, \mu')}$$

$$\frac{\langle \mathtt{c}_1, (\rho, \mu)\rangle \to (\rho, \mu_1) \qquad \langle \mathtt{c}_2, (\rho, \mu_1)\rangle \to (\rho, \mu_2)}{\langle \mathtt{c_1\ [seqC]\ c_2}, (\rho, \mu)\rangle \to (\rho, \mu_2)}$$

Le Transizioni Sem_{CMD} (Parte2) e Le Transizioni Sem_{PROG}

| [while] Exp Cmd | [emptyCMD]

$$\frac{\langle \texttt{e}, (\rho, \mu)\rangle \to 0}{\langle \texttt{[while] e c}, (\rho, \mu)\rangle \to (\rho, \mu)} \qquad \frac{\langle \texttt{e}, (\rho, \mu)\rangle \to 1 \quad \langle \texttt{c}, (\rho, \mu)\rangle \to (\rho, \mu_1) \quad \langle \texttt{[while] e c}, (\rho, \mu_1)\rangle \to (\rho, \mu')}{\langle \texttt{[while] e c}, (\rho, \mu)\rangle \to (\rho, \mu')}$$

$$\frac{}{\langle \texttt{[emptyCMD]}, (\rho, \mu)\rangle \to (\rho, \mu)}$$

Prog ::= [prog] Dcl Cmd | [progN] Cmd

$$\frac{\langle \texttt{c}, (\rho, \mu)\rangle \to (\rho, \mu')}{\langle \texttt{[progN] c}, (\rho, \mu)\rangle \to (\rho, \mu')} \qquad \frac{\langle \texttt{d}, (\rho, \mu)\rangle \to (\rho', \mu) \quad \langle \texttt{c}, (\rho', \mu)\rangle \to (\rho', \mu')}{\langle \texttt{[prog] d c}, (\rho, \mu)\rangle \to (\rho', \mu')}$$

Implementazione: Esecutore

- La funzione $\texttt{dclSem} : \texttt{Dcl} \to \texttt{State} \to \texttt{State}$.
$$(\forall \texttt{d}, \sigma) \quad \texttt{dclSem}(\texttt{d}, \sigma) = \sigma' \quad \texttt{iff} \quad \langle \texttt{d}, \sigma \rangle \to^* \sigma' \in \texttt{Sem}_{\texttt{DCL}}$$

- La funzione $\texttt{expSem} : \texttt{Exp} \to \texttt{State} \to \texttt{Num}$
$$(\forall \texttt{e}, \sigma) \quad \texttt{expSem}(\texttt{e}, \sigma) = \texttt{N} \quad \texttt{iff} \quad \langle \texttt{e}, \sigma \rangle \to^* \texttt{N} \in \texttt{Sem}_{\texttt{EXP}}$$

- La funzione $\texttt{cmdSem} : \texttt{cmd} \to \texttt{State} \to \texttt{State}$.
$$(\forall \texttt{c}, \sigma) \quad \texttt{cmdSem}(\texttt{c}, \sigma) = \sigma' \quad \texttt{iff} \quad \langle \texttt{c}, \sigma \rangle \to^* \sigma' \in \texttt{Sem}_{\texttt{CMD}}$$

- La funzione $\texttt{progSem} : \texttt{Prog} \to \texttt{State} \to \texttt{State}$.
$$(\forall \texttt{p}, \sigma) \quad \texttt{progSem}(\texttt{p}, \sigma) = \sigma' \quad \texttt{iff} \quad \langle \texttt{p}, \sigma \rangle \to^* \sigma' \in \texttt{Sem}_{\texttt{PROG}}$$

Varianti, Esetnsioni, Uso

- **Varianti.** Il costrutto per la dichiarazione di Array non prevede alcun controllo sul rispetto del bound superiore. Il controllo non può essere risolto a compile time. Allo stesso tempo, la mancanza di un tale controllo può condurre a stati di stuck

- **Estensioni.** Il linguaggio C fornisce anche un operatore condizionale per le espressioni con la seguente struttura sintattica:

  `guard ? exp1 : exp2`

  dove, `guard` è valutato come un'espressione boolean con cui selezionare il valore calcolato dall'espressione exp1 oppure exp2.

- **Uso.** SmallC è un Linguaggio Imperativo con 1 solo tipo di valore, 1 solo tipo di dato strutturato, senza blocchi e ad allocazione statica. Ha espressività limitata ma i suoi programmi definiscono tutte le funzioni calcolabili.

Gli Esercizi di Oggi: 3 esercizi

Esercizio (18)

a) Si scriva e si esegua un programma SmallC la cui esecuzione conduce ad uno stato di stuck.
b) Si modifichi la semantica per includere un controllo sul bound superiore dell'indice di accesso di un array. Check
c) Si modifichi l'implementazione in accordo alla nuova semantica.
d) Si esegua il programma in (a) col nuovo esecutore. Al termine, si mostri l'intero procedimento seguito. Check

Esercizio (19)

a) Si scriva un programma SmallC per il calcolo del bubblesort di un array.
b) Lo si esegua e si mostri lo stato risultante Check

Esercizio (20)

a) Si modifichi la Sintassi Concreta aggiungendo l'operatore condizionale alle espressione. Il nuovo operatore ha priorità più bassa di ogni altro operatore.
b) Si modifichi la Sintassi Astratta e si mostri l'AST della seguente espressione: `x+y?10:w-x*3`.
c) Si fornisca la semantica del nuovo operatore.
d) Si modifichi l'implementazione in accordo alla nuova semantica. Al termine si mostri il procedimento seguito. Check;