Laboratorio 5 - Small20.
Abstract Machine di Small20: Esecutore di Dichiarazioni

SOMMARIO: 5 Maggio, 2020

- Sintassi Astratta: AST in Ocaml (invariata)
- Sintassi Concreta: Una CFG per Small20 (Invariata)
- Abstract Machine: AM20
  - Stato (Corretto: allocate - Rimosso Arg. per Tipo)
- Attività.
  - Semantica SOS con Controllo Dinamico dei Tipi:
    - Dichiarazioni: Il Sistema $\texttt{Sem}_{\texttt{Dcl}}$
  - AM20 - Implementazione Esecutore:
    - Dichiarazioni: definizione Ocaml di `dclSem`

Sintassi Astratta: Gli AST di Small20

```
Type ::= [int] | [bool] | [void]
       | [arr] Type Num | [mut] Type | [terr]
       | [abs] Type TypeSeq
TypeSeq ::= Type [x] Type | Type

Dcl ::= [const] Type Ide Exp | [var] Type Ide Exp
      | [varN] Type Ide
      | [array] Type Ide Num

Exp ::= [val] Ide | Num | Bool | Ide [↑] Exp | [−1]Exp
      | Exp [+] Exp | Exp [−] Exp | Exp [∗] Exp | Exp [div] Exp
      | Exp [==] Exp | Exp [<] Exp | Exp [>] Exp
      | [not] Exp | Exp [or] Exp | Exp [and] Exp
      | Exp [=] Exp | [emptyE]

Cmd ::= Cmd [seqC] Cmd
      | [ifE] Exp Cmd Cmd | [ifT] Exp Cmd
      | [for] Stm Exp Stm Cmd | Exp | [emptyC]

Stm ::= Dcl | Cmd | Stm [seqM] Stm | [emptyS]

Prog ::= [prog] Ide Stm
```

Sintassi Concreta: Una CFG per Small20

$$
\begin{aligned}
&\texttt{Type} ::= \texttt{Simple} \mid \texttt{void} \mid \texttt{Simple}\ [\texttt{Num}]\\
&\texttt{Simple} ::= \texttt{int} \mid \texttt{bool}\\
\\
&\texttt{Dcl} ::= \texttt{final Simple ide} = \texttt{Exp} \mid \texttt{var Simple ide} = \texttt{Exp}\\
&\qquad\qquad \mid \texttt{var Simple ide} \mid \texttt{Simple array ide}\,[\texttt{num}]\\
\\
&\texttt{Exp} ::= \texttt{Exp or ExpB1} \mid \texttt{ExpB1}\\
&\texttt{ExpB1} ::= \texttt{ExpB1 and ExpB} \mid \texttt{ExpB}\\
&\texttt{ExpB} ::= \texttt{not ExpB} \mid \texttt{truth} \mid \texttt{ExpR}\\
&\texttt{ExpR} ::= \texttt{ExpR} == \texttt{ExpA2} \mid \texttt{ExpR} < \texttt{ExpA2} \mid \texttt{ExpR} > \texttt{ExpA2} \mid \texttt{ExpA2}\\
&\texttt{ExpA2} ::= \texttt{ExpA2} + \texttt{ExpA1} \mid \texttt{ExpA2} - \texttt{ExpA1} \mid \texttt{ExpA1}\\
&\texttt{ExpA1} ::= \texttt{ExpA1} * \texttt{ExpA} \mid \texttt{ExpA1} \,/\, \texttt{ExpA} \mid \texttt{ExpA}\\
&\texttt{ExpA} ::= \texttt{num} \mid \texttt{DExp} \mid \texttt{DExp} = \texttt{Exp} \mid (\texttt{Exp}) \mid -\texttt{ExpA2}\\
&\texttt{DExp} ::= \texttt{ide} \mid \texttt{ide}\,[\texttt{ExpA2}] \mid \epsilon\\
\\
&\texttt{Cmd} ::= \texttt{if (ExpB2) Cmd} \mid \texttt{OtherCmd}\\
&\texttt{OtherCmd} ::= \texttt{if (ExpB2) OtherCmd else Cmd} \mid \texttt{NonConditionalCmd}\\
&\texttt{NonConditionalCmd} ::= \texttt{for (Stm; Exp; Stm) Cmd} \mid \texttt{Exp} \mid \{\texttt{cmds}\}\\
&\texttt{Cmds} ::= \texttt{Cmd;} \mid \texttt{Cmd; Cmds}\\
\\
&\texttt{Stm} ::= \texttt{Dcl} \mid \texttt{Cmd}\\
&\texttt{Stms} ::= \texttt{Stm;} \mid \texttt{Stm; Stms}\\
\\
&\texttt{Prog} ::= \texttt{Program ide}\ \{\texttt{Stms}\}
\end{aligned}
$$

Semantica SOS di SMall20: Stato di AM20

- **Memoria** Statica per variabili ed array di tipi diversi (interi e booleani)
  Sequenza finita di **words** separate da "," e racchiusa in parentesi quadre:
  $$[\texttt{loc}_1 \leftarrow \texttt{Mv}_1, ..., \texttt{loc}_k \leftarrow \texttt{Mv}_k]$$
  - **allocate**$(\mu, \textbf{n})$: (alias, $\triangleright(\mu, \texttt{n})$) alloca `n words` in sequenza;
  - **upd**$(\mu, \textbf{loc}, \textbf{v})$: (alias, $\mu[\texttt{loc} \leftarrow \texttt{v}]$) modifica il contenuto di una word;
  - **getStore**$(\mu, \textbf{loc})$: (alias, $\mu(\texttt{loc})$) fornisce il contenuto di una word;
  - **emptyStore**(): (alias, 0^{μ}) crea uno store con `words` di contenuto indefinito.

- **Ambiente** per identificatori di costanti e di variabili (valori modificabili)
  Sequenza finita di **bindings** separati da "," e racchiusa in parentesi quadre:
  $$[\texttt{Ide}_1/\texttt{Den}_1, ..., \texttt{Ide}_k/\texttt{Den}_k]$$
  - **bind**$(\rho, \textbf{ide}, \textbf{den})$: (alias, $[\texttt{ide}/\texttt{den}] \circ \rho$) aggiunge un nuovo binding;
  - **getEnv**$(\rho, \textbf{ide})$: (alias, $\rho(\texttt{ide})$) valore denotabile di un binding;
  - **emptyEnv**(): (alias, 0^{ρ}) crea un'ambiente senza bindings.

- **Stato**: Coppia (ρ, μ), indicante Ambiente e Memoria.
  - Activation Record (Stack di AR):
    - Unico: Si riduce al solo frame ρ (del blocco programma).
    - Altri componenti:
      Assenti: Static/Dynamic chain, Return Address,...
      Associati allo AST: Program Counter, Intermediate Results,..

Semantica SOS di Small120: Transizione e Computazione

- **Transizione**.
  - Esecuzione di un costrutto c nello stato σ della Macchina Astratta
  - La esprimiamo con:
    - $\langle c, \sigma\rangle \rightarrow \sigma'$, indicante ...
    - $\langle c, \sigma\rangle \rightarrow \langle c', \sigma'\rangle$, indicante ...
    - $\langle c_1, \sigma_1\rangle \rightarrow \langle c'_1, \sigma'_1\rangle, \cdots, \langle c_k, \sigma_k\rangle \rightarrow \langle c'_k, \sigma'_k\rangle \ / \ \langle c, \sigma\rangle \rightarrow \langle c', \sigma'\rangle$, 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)
  - 0^{ρ} crea un ambiente vuoto: Senza bindings.
  - 0^{μ} crea una memoria vuota: Tutte le words sono indefinite e allocabili.
  - $\perp_{mem}$ memorizzabile indefinito, contenuto di word indefinita: Indica valore misleading o ingannevole
  - N (anche con pedici) intero, B (anche con pedici) booleano, I (anche con pedici) identificatore, D (anche con pedici), una coppia (t, d_t), indicante un tipo e una denotazione di tale tipo.
  - $[I/D]\circ\rho$ crea un nuovo ambiente che estende ρ con il binding [I/D]
  - $\rho(I)$ denotazione del binding di I in ρ, se esiste
  - $\triangleright(\mu, n)$ alloca in μ, n locazioni libere, in sequenza da loc, modifica μ in μ', restituisce (loc, μ')
  - $\triangleleft(\mu, \rho)$ dealloca da μ, le locazioni in ρ, che tornano allocabili. Restituisce la memoria modificata.
  - $\mu(loc)$ (loc deve essere una locazione giá allocata) restituisce il valore di loc
  - $\mu[loc \leftarrow v]$ (loc deve essere giá allocata) modifica il valore di loc con il memorizzabile v
  - $loc \oplus k$ locazione k posizioni dopo loc.
  - Introduzione o Vincolo. Posto in premessa con forma: espressione = pattern.

Dichiarazioni Small20: Le Transizioni $\mathrm{SEM}_{\mathrm{DCL}}$

Il Sistema di regole $\mathtt{Sem}_{\mathrm{DCL}}$ definisce il comportamento delle dichiarazioni durante la computazione dei Programmi Small20 sulla Macchina Astratta AM20.
Controllo dei Tipi Dinamico: Il Sistema $\mathtt{Sem}_{\mathrm{DCL}}$ è Integrato con il Sistema Y (prossima slide, per le dichiarazioni).

$\mathtt{Dcl} ::= \mathtt{[const]\ Type\ Ide\ Exp} \mid \mathtt{[var]\ Type\ Ide\ Exp}$
$\quad \mid \mathtt{[varN]\ Type\ Ide} \mid \mathtt{[array]\ Type\ Ide\ Num}$

$$\text{D1:}\ \frac{\langle \mathtt{e}, (\rho, \mu)\rangle \to \langle \mathtt{t_e}, \mathtt{v}, (\rho, \mu_\mathtt{e})\rangle \quad \mathtt{t} \in \mathtt{Simple} \quad \mathtt{t} = \mathtt{t_e} \quad [\mathtt{I}/(\mathtt{t}, \mathtt{v})]\circ\rho = \rho_\mathtt{I}}{\langle \mathtt{[const]\ t\ I\ e}, (\rho, \mu)\rangle \to (\mathtt{[void]}, (\rho_\mathtt{I}, \mu_\mathtt{e}))}$$

$$\text{D2:}\ \frac{\langle \mathtt{e}, (\rho, \mu)\rangle \to \langle \mathtt{t_e}, \mathtt{v}, (\rho, \mu_\mathtt{e})\rangle \quad \mathtt{t} = \mathtt{t_e} \quad \mathtt{t} \in \mathtt{Simple} \quad \rhd(\mu_\mathtt{e}, 1) = (\mathtt{loc_t}, \mu_\mathtt{a}) \quad [\mathtt{I}/(\mathtt{[mut]\ t_e}, \mathtt{loc_t})]\circ\rho = \rho_\mathtt{F} \quad \mu_\mathtt{a}[\mathtt{loc_t} \leftarrow \mathtt{v}] = \mu_\mathtt{F}}{\langle \mathtt{[var]\ t\ I\ e}, (\rho, \mu)\rangle \to (\mathtt{[void]}, (\rho_\mathtt{F}, \mu_\mathtt{F}))}$$

$$\text{D3:}\ \frac{\mathtt{t} \in \mathtt{Simple} \quad \rhd(\mu, 1) = (\mathtt{loc_t}, \mu_\mathtt{a}) \quad [\mathtt{I}/(\mathtt{[mut]\ t}, \mathtt{loc_t})]\circ\rho = \rho_\mathtt{I}}{\langle \mathtt{[varN]\ t\ I}, (\rho, \mu)\rangle \to (\mathtt{[void]}, (\rho_\mathtt{F}, \mu_\mathtt{F}))}$$

$$\text{D4:}\ \frac{\mathtt{N} > 0 \quad \mathtt{t} \in \mathtt{Simple} \quad \rhd(\mu, \mathtt{N}) = (\mathtt{loc_t}, \mu_\mathtt{F}) \quad [\mathtt{I}/(\mathtt{[mut]([Arr]\ t\ N)}, \mathtt{loc_t})]\circ\rho = \rho_\mathtt{F}}{\langle \mathtt{[array]\ t\ I\ N}, (\rho, \mu)\rangle \to (\mathtt{[void]}, (\rho_\mathtt{F}, \mu_\mathtt{F}))}$$

Notazione, Osservazioni

- [xxx] è un costruttore di AST (i.e, Albero di Sintassi Astratta). I rimanenti identificatori che occorrono in una regola sono nomi di variabile quantificate universalmente se introdotte, i.e. bound, nel termine sinistro della conclusione, esistenzialmente se introdotte nel termine destro di una premessa o di una uguaglianza.
- $\mathtt{Simple} = \{\mathtt{[Int]}, \mathtt{[Bool]}\}$

Sistema Y: Regole per DCL

$$\text{Y1:}\ \frac{\langle \mathtt{e}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}', \mathtt{Y}_\rho) \qquad \mathtt{t} \in \mathtt{Simple} \qquad \mathtt{t}' = \mathtt{t}}{\langle [\mathtt{const}]\ \mathtt{t}\ \mathtt{I}\ \mathtt{e}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} ([\mathtt{void}], [\mathtt{I}/\mathtt{t}]\circ\mathtt{Y}_\rho)}$$

$$\text{Y2:}\ \frac{\langle \mathtt{e}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}', \mathtt{Y}_\rho) \qquad \mathtt{t} \in \mathtt{Simple} \qquad \mathtt{t}' = \mathtt{t}}{\langle [\mathtt{var}]\ \mathtt{t}\ \mathtt{I}\ \mathtt{e}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} ([\mathtt{void}], [\mathtt{I}/[\mathtt{mut}]\ \mathtt{t}]\circ\mathtt{Y}_\rho)} \qquad \text{Y3:}\ \frac{\mathtt{t} \in \mathtt{Simple}}{\langle [\mathtt{varN}]\ \mathtt{t}\ \mathtt{I}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} ([\mathtt{void}], [\mathtt{I}/[\mathtt{mut}]\ \mathtt{t}]\circ\mathtt{Y}_\rho)}$$

$$\text{Y4:}\ \frac{\mathtt{t} \in \mathtt{Simple} \qquad \mathtt{N} > 0}{\langle [\mathtt{array}]\ \mathtt{t}\ \mathtt{I}\ \mathtt{N}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} ([\mathtt{void}], [\mathtt{I}/[\mathtt{mut}]([\mathtt{arr}]\ \mathtt{t}\ \mathtt{N})]\circ\mathtt{Y}_\rho)}$$

Gestione Errori di Tipo:

$$\text{E1:}\ \frac{\mathtt{t} \notin \mathtt{Simple}}{\langle [\mathtt{const}]\ \mathtt{t}\ \mathtt{I}\ \mathtt{e}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}_\rho)} \qquad \text{E2:}\ \frac{\langle \mathtt{e}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}', \mathtt{Y}_\rho) \qquad \mathtt{t}' \neq \mathtt{t}}{\langle [\mathtt{const}]\ \mathtt{t}\ \mathtt{I}\ \mathtt{e}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}_\rho)}$$

$$\text{E3:}\ \frac{\mathtt{t} \notin \mathtt{Simple}}{\langle [\mathtt{var}]\mathtt{t}\ \mathtt{I}\ \mathtt{e}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}_\rho)} \quad \text{E4:}\ \frac{\langle \mathtt{e}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}', \mathtt{Y}_\rho) \qquad \mathtt{t}' \neq \mathtt{t}}{\langle [\mathtt{var}]\mathtt{t}\ \mathtt{I}\ \mathtt{e}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}_\rho)} \quad \text{E5:}\ \frac{\mathtt{t} \notin \mathtt{Simple}}{\langle [\mathtt{varN}]\mathtt{t}\ \mathtt{I}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}_\rho)}$$

$$\text{E6:}\ \frac{\mathtt{t} \notin \mathtt{Simple}}{\langle [\mathtt{array}]\ \mathtt{t}\ \mathtt{I}\ \mathtt{N}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}_\rho)} \qquad \text{E7:}\ \frac{\mathtt{N} \leq 0}{\langle [\mathtt{array}]\ \mathtt{t}\ \mathtt{I}\ \mathtt{N}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}_\rho)}$$

Notazione: Simboli e Strutture

- N (anche con pedici, apici), Intero
- B (anche con pedici, apici), Booleans
- I (anche con pedici, apici), Identificatore
- t (anche con pedici, apici), (Espressione di) Tipo
- d (anche con pedici, apici), dichiarazione
- e (anche con pedici, apici), espressione
- c (anche con pedici, apici), comando
- m (anche con pedici, apici), statement
- $\mathtt{Simple} = \{[\mathtt{int}], [\mathtt{bool}]\}$

Implementazione: La Funzione di Interpretazione delle Dichiarazioni

Dobbiamo introdurre ed implementare in OCaml una funzione che esprima il comportamento del sistema $\texttt{SEM}_{\texttt{DCL}}$ definito per la semantica SOS delle Dichiarazioni Small20

- Introduciamo una funzione che chiameremo `dclSem`.
- `dclSem` deve definire una trasformazione che data una dichiarazione `d` e uno Stato σ calcola la coppia Tipo-Stato prodotto usando le transizioni di $\texttt{SEM}_{\texttt{DCL}}$:

$$(\forall \texttt{d}, \sigma) \quad \texttt{dclSem}(\texttt{d}, \sigma) = (\texttt{t}_\texttt{d}, \sigma_\texttt{d}) \quad \texttt{iff} \quad \langle \texttt{d}, \sigma \rangle \rightarrow (\texttt{t}_\texttt{d}, \sigma_\texttt{d}) \in \texttt{Sem}_{\texttt{DCL}}$$

- `dclSem` ha segnatura:

$$\texttt{dclSem} : \texttt{d} \rightarrow \texttt{State} \rightarrow \texttt{Type} * \texttt{State}$$

  La presenza di premesse contenenti $\rightarrow_{\texttt{DCL}}$ nelle regole di inferenza conduce a definizioni ricorsive della funzione semantica.

- `dclSem` associa ad ogni dichiarazione una funzione $\texttt{State} \rightarrow \texttt{Type} * \texttt{State}$, dove il primo (risp. secondo) componente dell'immagine è il tipo, associato dal Sistema Y, alla (risp. lo stato prodotto dalla Semantica per la) dichiarazione d, i cui identificatori hanno tipo (risp. valori) come specificato dallo Stato sorgente.

Implementazione Dichiarazioni di AM20: 4 Esercizi

Esercizio (1)

Regola D1: Verificare correttezza Tipo associato e Stato prodotto per dichiarazione di Costante.
Funzione Semantica dclSem: Verificare il Codice e fornire Test Prima Verifica (vedi Listing allegato)

Esercizio (2)

Regola D2: Completare Premessa per dichiarazione di Variabile Inizializzata.
Funzione Semantica dclSem: Estendere Codice e fornire Test Prima Verifica (vedi Listing allegato)

Esercizio (3)

Regola D3: Completare Premessa per dichiarazione di Variabile Non Inizializzata.
Funzione Semantica dclSem: Estendere Codice e fornire Test Prima Verifica (vedi Listing allegato)

Esercizio (4)

Regola D4: Completare Premessa per dichiarazione di Array.
Funzione Semantica dclSem: Estendere Codice e fornire Test Prima Verifica (vedi Listing allegato)