Laboratorio 4- Transizioni SOS delle Dichiarazioni, Stato e Implementazione

Sommario: 12 aprile, 2019

- Semantica SOS - Transizioni e Computazione
- Dichiarazioni: Le Transizioni

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 SmallCXXX, con l'indice corretto.
- Seguire la Presentazione delle Attività della Sessione.
- Svogere le Attività modificando il file SmallC distribuito.

Semantica SOS di SmallC: 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)
  - N (anche con pedici) intero, I (anche con pedici) identificatore, D (anche con pedici) valore denotabile
  - $[I/D] \circ \rho$ crea un nuovo ambiente che estende ρ con il binding [I/D]
  - $\rho(I)$ denotazione del binding di I in ρ, se esiste
  - $\perp_{mem}$ indefinito nei memorizzabili
  - allocate(μ, n) alloca n locazioni libere, in sequenza, a partire da loc, modificando μ in μ' e restituisce (loc, μ')
  - μ(loc) (loc deve essere una locazione giá allocata) restituisce il valore di loc
  - μ[loc $\leftarrow$ n] (loc deve essere giá allocata) modifica il valore di loc con il valore n

Dichiarazioni SmallC: Le Transizioni $\mathtt{Sem}_{\mathtt{Dcl}}$

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

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

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

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

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

$$\frac{}{\langle \mathtt{emptyDCL}, \sigma\rangle \rightarrow \sigma}$$

$$\frac{\langle \mathtt{d}_1, (\rho, \mu)\rangle \rightarrow (\rho', \mu')}{\langle \mathtt{d}_1; \mathtt{d}_2, (\rho, \mu)\rangle \rightarrow \langle \mathtt{d}_2, (\rho', \mu')\rangle}$$

Implementazione: Memoria, Ambiente e Stato

Se esaminiamo le transizioni, vediamo le operazioni sulla Memoria e sull'Ambiente richieste per descrivere il comportamento delle dichiarazioni (semantica SOS).

- **Memoria**.
  - **allocate**(μ, **n**): alloca `n words` (per interi) in sequenza
  - **upd**(μ, **loc**, **n**): (alias, $\mu[\texttt{loc} \leftarrow \texttt{n}]$) modifica il valore di una locazione
  - **getStore**(μ, **loc**): (alias, $\mu(\texttt{loc})$) fornisce il valore di una locazione
  - **emptyStore**(): crea uno store iniziale con `words` libere, a valore indefinito
- **Ambiente**.
  - **bind**(ρ, **ide**, **den**): (alias, $[\texttt{ide}/\texttt{den}] \circ \rho$) aggiunge un nuovo binding
  - **getEnv**(ρ, **ide**): (alias, $\rho(\texttt{ide})$) valore denotabile di un binding
  - **emptyEnv**(): crea un'ambiente senza bindings

Per l'implementazione vedere il codice OCaml nel listing allegato all'attività di oggi. In particolare le sezioni:

1) **Store: Definition**.
   Operazioni per l'introduzione, manipolazione e presentazione di valori store;
2) **Environment: Definition**.
   Operazioni per l'introduzione, manipolazione e presentazione di valori env;
3) **State: A pair...**.
   Operazioni per la presentazione dei valori state, e definizione dello stato iniziale sigma0.

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 di SmallC

- Introduciamo una funzione che chiameremo `dclSem`.
- `dclSem` deve definire una trasformazione che data una dichiarazione `d` e uno stato σ calcola lo stato prodotto usando le transizioni di $\texttt{SEM}_{\texttt{DCL}}$. Ovvero:
$$(\forall \texttt{d}, \sigma) \quad \texttt{dclSem}(\texttt{d}, \sigma) = \sigma' \quad \texttt{iff} \quad \langle \texttt{d}, \sigma \rangle \rightarrow^* \sigma' \in \texttt{Sem}_{\texttt{DCL}}$$ [1]
- `dclSem` ha segnatura:
$$\texttt{dclSem} : \texttt{Dcl} \rightarrow \texttt{State} \rightarrow \texttt{State}$$
La presenza di premesse contenenti $\rightarrow$ nelle regole di inferenza conduce a definizioni ricorsive della funzione semantica.
- `dclSem` associa ad ogni dichiarazione una funzione di tipo $\texttt{State} \rightarrow \texttt{State}$

[1] $\rightarrow^*$ indica una computazione terminante in k-applicazioni delle regole $\texttt{Sem}_{\texttt{Dcl}}$, per qualche $\texttt{k} \geq 1$, ovvero:
$$\langle \texttt{d}, \sigma \rangle \equiv \langle \texttt{d}_1, \sigma_1 \rangle \rightarrow \ldots \rightarrow \langle \texttt{d}_{\texttt{k}-1}, \sigma_{\texttt{k}-1} \rangle \rightarrow \sigma_{\texttt{k}} \equiv \sigma'$$

SMallC: 2 Esercizi

Esercizio (1)

Nello spazio riservato ai "Semantic Functions; dclSem", si fornisca la definizione per un AST, dd, la cui sintassi concreta è mostrata sotto (con l'ultima riga vuota indicante una dicharazione vuota):

```
k = 12;
var x = 12;
var z;
A[7];
```

(a) *Si verifichi la corrispondenza di quanto scritto: Check;*
(b) *Si applichi la funzione semantica "dclSem" e si calcoli lo stato generato;*
(c) *Si commenti tale stato: Check*
Perchè quel risultato? Quali errori nell'uso delle operazioni? Quali errori nella programmazione? ;

Esercizio (2)

(a) *Si corregga la definizione per casi della funzione "dclSem" in modo tale che dopo la correzione di ogni caso si verifichi la soluzione data: La verifica deve consistere nel calcolare la semantica dell'AST dd, modificato per i soli casi corretti.*
(b) *Ottenuta la definizione corretta la si applichi alla semantica dell'AST dd definito inizialmente e si mostri lo stato finale prodotto, utilizzando le operazioni appropriate Check*

SmallC: Al termine di Ogni Sessione

- Riportare nel file SmallC.ml ogni progresso nel codice sviluppato durante la sessione.
- La prossima sessione:
  - Semantica SOS delle espressioni: Che Rivisiteremo Ancora
  - Implementeremo la Funzione Semantica per l'interpretazione [decodifica e modifica_dello_stato] delle espressioni di SmallC