Linguaggi di Programmazione con Laboratorio

Seminario di fine corso:
Il `for` del linguaggio Pascal in Small21

Alessio Marchetti

Università di Pisa, Dipartimento di Matematica

2 settembre 2021

Presentazione del ciclo For - 1

Il ciclo For del linguaggio Pascal è un costrutto di iterazione strutturata determinata con la seguente sintassi concreta:

$$\texttt{OtherStm} \rightarrow \texttt{For Ide from Exp to Exp do Stm} \mid \ldots$$

- L'identificatore è detto indice del ciclo.
- La prima espressione è il valore iniziale.
- La seconda espressione è il valore finale.
- Lo statement è il corpo del ciclo.
- I primi tre elementi formano la guardia.

Presentazione del ciclo For - 2

- Siano n e m i valori dell'espressione iniziale e finale, valutate in questo ordine a partire dallo stato in cui è valutato il costrutto.
- Il corpo del ciclo è eseguito un numero di volte pari al massimo tra 0 e la differenza $m - n$.
- La prima iterazione è eseguita nello stato risultante dalla valutazione delle due espressioni. Le altre iterazioni sono eseguite ciascuna nello stato finale dell'iterazione precedente.
- L'indice è un identificatore visibile solo al ciclo.
- In ogni iterazione del corpo, l'indice è associato a un valore pari alla somma di n e del numero di iterazioni già svolte.
- Il corpo del ciclo non può contenere dichiarazioni dell'indice nel suo blocco più esterno e nemmeno contenere costrutti che ne possano modificare il valore.

Espressività - 1

Domanda

Aumenta l'espressività?

Nella versione base di Small21 non sono presenti costrutti di iterazione strutturata.

Il comportamento del ciclo for può essere comunque emulato tramite goto (nel blocco più esterno del programma) oppure con funzioni ricorsive (ma AM21 non ha meccanismi per rendere efficiente la ricorsione di coda).

Esempio di utilizzo - 1

```
Program my_program {
      int[10] array;
      int sum = 0;
      For i from 0 to 10 do
         array[i] = i;
      For i from 0 to 10 do
         sum = (sum + array[i]);
      }
```

Esempio di utilizzo - 2

```
Program my_program {
    int n = 10;
    For i from 0 to n do
        n = n+1;
}
```

Esempio di programma non corretto

```
Program my_program {
    int n = 10;
    For i from 0 to n do
        i = 0;
}
```

Una prima proposta

Sintassi astratta:

$$\texttt{Stm} \rightarrow \texttt{[For]}\ \texttt{Ide}\ \texttt{Exp}\ \texttt{Exp}\ \texttt{Stm} \mid \cdots$$

Aggiunta al sistema Y:

$$\text{(YN1)}\quad \frac{\begin{array}{c}\langle \texttt{expi}, Y_\rho \rangle \rightarrow_Y (\texttt{[int]}, Y_\rho) \\ \langle \texttt{expf}, Y_\rho \rangle \rightarrow_Y (\texttt{[int]}, Y_\rho) \\ \texttt{[id/[int]]} \otimes Y_\rho = Y_{\rho'} \\ \langle \texttt{stm}, Y_{\rho'} \rangle \rightarrow_Y (\texttt{[void]}, Y_{\rho'})\end{array}}{\langle \texttt{[For]}\ \texttt{id}\ \texttt{expi}\ \texttt{expf}\ \texttt{stm}, Y_\rho \rangle \rightarrow_Y (\texttt{[void]}, Y_\rho)}$$

Tuttavia questo genere di controlli dovrebbe essere fatta staticamente nel front-end, mentre in AM21 il sistema Y è integrato nel back-end.

Proponiamo allora una soluzione alternativa, con l'introduzione di un nuovo tipo, detto `forIndex`, sottotipo del tipo `int`.

Per supportare il ciclo for in AM21, introduciamo anche uno statement ausiliario `ForInit`.

Sintassi Astratta - Finale

```
Type → [forIndex] | ...
 Stm → [For] DExp Exp Stm |
       [ForInit] Ide Exp Exp Stm | ...
```

Aggiornamento del sistema Y e di $\mathrm{SEM}_{\mathrm{EXP}}$ per `forIndex`

$$\text{(YU1)}\quad \frac{Y_\rho(\texttt{ide}) = \texttt{[forIndex]}}{\langle \texttt{[val] ide}, Y_\rho\rangle \to_Y (\texttt{[int]}, Y_\rho)}$$

$$\text{(E18)}\quad \frac{\begin{array}{c} Y_\rho(\texttt{ide}) = \texttt{t} \\ \texttt{t} \neq \texttt{[mut] t'} \qquad \texttt{t'} \in \texttt{Simple} \\ \texttt{t} \notin \texttt{Simple} \\ \texttt{t} \neq \texttt{[forIndex]} \end{array}}{\langle \texttt{[val] ide}, Y_\rho\rangle \to_Y (\texttt{[terr]}, Y_\rho)}$$

$$\text{(XU1)}\quad \frac{\begin{array}{c} \sigma = (\Delta, \mu) \\ \Delta(\texttt{ide}) = (\texttt{[forIndex]}, \texttt{loc}) \\ \mu(\texttt{loc}) = \texttt{v} \end{array}}{\langle \texttt{[val] ide}, \sigma\rangle \to \lfloor \texttt{[int]}, \texttt{v}, \sigma \rfloor}$$

Aggiornamento del sistema Y e di $\mathrm{SEM}_{\mathtt{DEN}}$ per `forIndex`

$$\text{(YU2)} \quad \frac{Y_\rho(\mathtt{ide}) = \mathtt{[forIndex]}}{\langle \mathtt{[val]\ \ ide}, Y_\rho\rangle \to_{DY} (\mathtt{[forIndex]}, Y_\rho)}$$

$$\text{(E17)} \quad \frac{\begin{array}{c} Y_\rho(\mathtt{ide}) = \mathtt{t} \\ \mathtt{t} \neq \mathtt{[mut]\ t'} \qquad \mathtt{t} \neq \mathtt{[forIndex]} \end{array}}{\langle \mathtt{[val]\ \ ide}, Y_\rho\rangle \to_{DY} (\mathtt{[terr]}, Y_\rho)}$$

$$\text{(XU2)} \quad \frac{\begin{array}{c} \sigma = (\Delta, \mu) \\ \Delta(\mathtt{ide}) = (\mathtt{[forIndex]}, \mathtt{loc}) \end{array}}{\langle \mathtt{[val]\ \ ide}, \sigma\rangle \to_{\mathtt{DEN}} \lfloor \mathtt{[forIndex]}, \mathtt{loc}, \sigma \rfloor}$$

Aggiornamento del sistema Y per `For`

$$\text{(YU3)}\quad \frac{\begin{array}{c}\langle \texttt{dexp}, Y_\rho\rangle \to_{DY} (\texttt{t}_1, Y_\rho) \\ \texttt{t}_1 = \texttt{[forIndex]} \\ \langle \texttt{exp}, Y_\rho\rangle \to_{Y} (\texttt{t}_2, Y_\rho) \\ \texttt{t}_2 = \texttt{[int]} \\ \langle \texttt{stm}, Y_\rho\rangle \to_{Y} (\texttt{[void]}, Y_\rho)\end{array}}{\langle \texttt{[For] dexp exp stm}, Y_\rho\rangle \to_{Y} (\texttt{[void]}, Y_\rho)}$$

Aggiornamento del sistema Y per `For`

$$\text{(YU3)}\quad \frac{\begin{array}{c}\langle \texttt{dexp}, Y_\rho\rangle \rightarrow_{DY} (\texttt{t}_1, Y_\rho) \\ \texttt{t}_1 = \texttt{[forIndex]} \\ \langle \texttt{exp}, Y_\rho\rangle \rightarrow_{Y} (\texttt{t}_2, Y_\rho) \\ \texttt{t}_2 = \texttt{[int]} \\ \langle \texttt{stm}, Y_\rho\rangle \rightarrow_{Y} (\texttt{[void]}, Y_\rho)\end{array}}{\langle \texttt{[For] dexp exp stm}, Y_\rho\rangle \rightarrow_{Y} (\texttt{[void]}, Y_\rho)}$$

Aggiornamento del sistema Y per `For`

$$\text{(YU3)}\quad \frac{\begin{array}{c}\langle \texttt{dexp}, Y_\rho\rangle \rightarrow_{DY} (\texttt{t}_1, Y_\rho) \\ \texttt{t}_1 = \texttt{[forIndex]} \\ \langle \texttt{exp}, Y_\rho\rangle \rightarrow_{Y} (\texttt{t}_2, Y_\rho) \\ \texttt{t}_2 = \texttt{[int]} \\ \langle \texttt{stm}, Y_\rho\rangle \rightarrow_{Y} (\texttt{[void]}, Y_\rho)\end{array}}{\langle \texttt{[For] dexp exp stm}, Y_\rho\rangle \rightarrow_{Y} (\texttt{[void]}, Y_\rho)}$$

Aggiornamento del sistema Y per `For`

$$\text{(YU3)}\quad \frac{\begin{array}{c}\langle \texttt{dexp}, Y_\rho\rangle \to_{DY} (\texttt{t}_1, Y_\rho) \\ \texttt{t}_1 = \texttt{[forIndex]} \\ \langle \texttt{exp}, Y_\rho\rangle \to_{Y} (\texttt{t}_2, Y_\rho) \\ \texttt{t}_2 = \texttt{[int]} \\ \langle \texttt{stm}, Y_\rho\rangle \to_{Y} (\texttt{[void]}, Y_\rho)\end{array}}{\langle \texttt{[For] dexp exp stm}, Y_\rho\rangle \to_{Y} (\texttt{[void]}, Y_\rho)}$$

Aggiornamento del sistema Y per `For` - Errori

$$\text{(EU1)}\quad \frac{\langle \texttt{dexp}, Y_\rho\rangle \rightarrow_{DY} (\texttt{t}_1, Y_\rho) \qquad \texttt{t}_1 \neq \texttt{[forIndex]}}{\langle \texttt{[For]}\ \ \texttt{dexp}\ \ \texttt{exp}\ \ \texttt{stm}, Y_\rho\rangle \rightarrow_{Y} (\texttt{[terr]}, Y_\rho)}$$

$$\text{(EU2)}\quad \frac{\langle \texttt{exp}, Y_\rho\rangle \rightarrow_{Y} (\texttt{t}_2, Y_\rho) \qquad \texttt{t}_2 \neq \texttt{[Int]}}{\langle \texttt{[For]}\ \ \texttt{dexp}\ \ \texttt{exp}\ \ \texttt{stm}, Y_\rho\rangle \rightarrow_{Y} (\texttt{[terr]}, Y_\rho)}$$

$$\text{(EU3)}\quad \frac{\langle \texttt{stm}, Y_\rho\rangle \rightarrow_{Y} (\texttt{[terr]}, Y_\rho)}{\langle \texttt{[For]}\ \ \texttt{dexp}\ \ \texttt{exp}\ \ \texttt{stm}, Y_\rho\rangle \rightarrow_{Y} (\texttt{[terr]}, Y_\rho)}$$

Aggiornamento di SEM_{STM} per For - 1

$$\text{(SU1)}\quad \frac{\begin{array}{c}\langle \texttt{dexp}, \sigma\rangle \to_{\texttt{DEN}} \lfloor \texttt{[forIndex]}, \texttt{loc}, \sigma_1 \rfloor \\ \langle \texttt{exp}, \sigma\rangle \to \lfloor \texttt{[int]}, \texttt{m}, \sigma_2 \rfloor \\ \sigma_2 = (\Delta_2, \mu_2) \qquad \mu_2(\texttt{loc}) = \texttt{n} \\ \langle \texttt{stm}, \sigma_2\rangle \to (\texttt{[void]}, \sigma_3) \\ \sigma_3 = (\Delta_3, \mu_3) \qquad \mu_4 = \mu_3[\texttt{loc} \leftarrow \texttt{n} + 1] \\ \texttt{n} < \texttt{m} \qquad \sigma_4 = (\Delta_3, \mu_4) \\ \texttt{loop} = \texttt{[For] dexp ([num] m) stm} \\ \langle \texttt{loop}, \sigma_4\rangle \to (\texttt{[void]}, \sigma_5)\end{array}}{\langle \texttt{[For] dexp exp stm}, \sigma\rangle \to (\texttt{[void]}, \sigma_5)}$$

Aggiornamento di $\mathrm{SEM}_{\mathrm{STM}}$ per `For` - 1

$$\text{(SU1)}\quad \frac{\begin{array}{c}\langle \texttt{dexp}, \sigma\rangle \to_{\texttt{DEN}} \lfloor \texttt{[forIndex]}, \texttt{loc}, \sigma_1 \rfloor \\ \langle \texttt{exp}, \sigma\rangle \to \lfloor \texttt{[int]}, \texttt{m}, \sigma_2 \rfloor \\ \sigma_2 = (\Delta_2, \mu_2) \qquad \mu_2(\texttt{loc}) = \texttt{n} \\ \langle \texttt{stm}, \sigma_2\rangle \to (\texttt{[void]}, \sigma_3) \\ \sigma_3 = (\Delta_3, \mu_3) \qquad \mu_4 = \mu_3[\texttt{loc} \leftarrow \texttt{n} + 1] \\ \texttt{n} < \texttt{m} \qquad \sigma_4 = (\Delta_3, \mu_4) \\ \texttt{loop} = \texttt{[For] dexp ([num] m) stm} \\ \langle \texttt{loop}, \sigma_4\rangle \to (\texttt{[void]}, \sigma_5)\end{array}}{\langle \texttt{[For] dexp exp stm}, \sigma\rangle \to (\texttt{[void]}, \sigma_5)}$$

Aggiornamento di $\mathrm{SEM}_{\mathrm{STM}}$ per `For` - 1

$$\text{(SU1)}\quad \frac{\begin{array}{c}\langle \texttt{dexp}, \sigma\rangle \rightarrow_{\texttt{DEN}} \lfloor \texttt{[forIndex]}, \texttt{loc}, \sigma_1 \rfloor \\ \langle \texttt{exp}, \sigma\rangle \rightarrow \lfloor \texttt{[int]}, \texttt{m}, \sigma_2 \rfloor \\ \sigma_2 = (\Delta_2, \mu_2) \qquad \mu_2(\texttt{loc}) = \texttt{n} \\ \langle \texttt{stm}, \sigma_2\rangle \rightarrow (\texttt{[void]}, \sigma_3) \\ \sigma_3 = (\Delta_3, \mu_3) \qquad \mu_4 = \mu_3[\texttt{loc} \leftarrow \texttt{n} + 1] \\ \texttt{n} < \texttt{m} \qquad \sigma_4 = (\Delta_3, \mu_4) \\ \texttt{loop} = \texttt{[For] dexp ([num] m) stm} \\ \langle \texttt{loop}, \sigma_4\rangle \rightarrow (\texttt{[void]}, \sigma_5)\end{array}}{\langle \texttt{[For] dexp exp stm}, \sigma\rangle \rightarrow (\texttt{[void]}, \sigma_5)}$$

Aggiornamento di SEM_{STM} per For - 1

$$\text{(SU1)}\quad \frac{\begin{array}{c}\langle \texttt{dexp}, \sigma\rangle \to_{\texttt{DEN}} \lfloor \texttt{[forIndex]}, \texttt{loc}, \sigma_1 \rfloor \\ \langle \texttt{exp}, \sigma\rangle \to \lfloor \texttt{[int]}, \texttt{m}, \sigma_2 \rfloor \\ \sigma_2 = (\Delta_2, \mu_2) \qquad \mu_2(\texttt{loc}) = \texttt{n} \\ \langle \texttt{stm}, \sigma_2\rangle \to (\texttt{[void]}, \sigma_3) \\ \sigma_3 = (\Delta_3, \mu_3) \qquad \mu_4 = \mu_3[\texttt{loc} \leftarrow \texttt{n} + 1] \\ \texttt{n} < \texttt{m} \qquad \sigma_4 = (\Delta_3, \mu_4) \\ \texttt{loop} = \texttt{[For] dexp ([num] m) stm} \\ \langle \texttt{loop}, \sigma_4\rangle \to (\texttt{[void]}, \sigma_5)\end{array}}{\langle \texttt{[For] dexp exp stm}, \sigma\rangle \to (\texttt{[void]}, \sigma_5)}$$

Aggiornamento di $\mathrm{SEM}_{\mathtt{STM}}$ per `For` - 1

$$\text{(SU1)}\quad \frac{\begin{array}{c}\langle \mathtt{dexp}, \sigma\rangle \to_{\mathtt{DEN}} \lfloor \mathtt{[forIndex]}, \mathtt{loc}, \sigma_1 \rfloor \\ \langle \mathtt{exp}, \sigma\rangle \to \lfloor \mathtt{[int]}, \mathtt{m}, \sigma_2 \rfloor \\ \sigma_2 = (\Delta_2, \mu_2) \qquad \mu_2(\mathtt{loc}) = \mathtt{n} \\ \langle \mathtt{stm}, \sigma_2\rangle \to (\mathtt{[void]}, \sigma_3) \\ \sigma_3 = (\Delta_3, \mu_3) \qquad \mu_4 = \mu_3[\mathtt{loc} \leftarrow \mathtt{n}+1] \\ \mathtt{n} < \mathtt{m} \qquad \sigma_4 = (\Delta_3, \mu_4) \\ \mathtt{loop} = \mathtt{[For]\ dexp\ ([num]\ m)\ stm} \\ \langle \mathtt{loop}, \sigma_4\rangle \to (\mathtt{[void]}, \sigma_5)\end{array}}{\langle \mathtt{[For]\ dexp\ exp\ stm}, \sigma\rangle \to (\mathtt{[void]}, \sigma_5)}$$

Aggiornamento di SEM_{STM} per For - 1

$$\text{(SU1)}\quad \frac{\begin{array}{c}\langle \texttt{dexp}, \sigma\rangle \to_{\texttt{DEN}} \lfloor \texttt{[forIndex]}, \texttt{loc}, \sigma_1 \rfloor \\ \langle \texttt{exp}, \sigma\rangle \to \lfloor \texttt{[int]}, \texttt{m}, \sigma_2 \rfloor \\ \sigma_2 = (\Delta_2, \mu_2) \qquad \mu_2(\texttt{loc}) = \texttt{n} \\ \langle \texttt{stm}, \sigma_2\rangle \to (\texttt{[void]}, \sigma_3) \\ \sigma_3 = (\Delta_3, \mu_3) \qquad \mu_4 = \mu_3[\texttt{loc} \leftarrow \texttt{n} + 1] \\ \texttt{n} < \texttt{m} \qquad \sigma_4 = (\Delta_3, \mu_4) \\ \texttt{loop} = \texttt{[For] dexp ([num] m) stm} \\ \langle \texttt{loop}, \sigma_4\rangle \to (\texttt{[void]}, \sigma_5)\end{array}}{\langle \texttt{[For] dexp exp stm}, \sigma\rangle \to (\texttt{[void]}, \sigma_5)}$$

Aggiornamento di $\mathtt{SEM}_{\mathtt{STM}}$ per `For` - 1

$$\text{(SU1)}\quad \frac{\begin{array}{c}\langle \mathtt{dexp}, \sigma\rangle \rightarrow_{\mathtt{DEN}} \lfloor \mathtt{[forIndex]}, \mathtt{loc}, \sigma_1 \rfloor \\ \langle \mathtt{exp}, \sigma\rangle \rightarrow \lfloor \mathtt{[int]}, \mathtt{m}, \sigma_2 \rfloor \\ \sigma_2 = (\Delta_2, \mu_2) \qquad \mu_2(\mathtt{loc}) = \mathtt{n} \\ \langle \mathtt{stm}, \sigma_2\rangle \rightarrow (\mathtt{[void]}, \sigma_3) \\ \sigma_3 = (\Delta_3, \mu_3) \qquad \mu_4 = \mu_3[\mathtt{loc} \leftarrow \mathtt{n}+1] \\ \mathtt{n} < \mathtt{m} \qquad \sigma_4 = (\Delta_3, \mu_4) \\ \mathtt{loop} = \mathtt{[For]\ dexp\ ([num]\ m)\ stm} \\ \langle \mathtt{loop}, \sigma_4\rangle \rightarrow (\mathtt{[void]}, \sigma_5)\end{array}}{\langle \mathtt{[For]\ dexp\ exp\ stm}, \sigma\rangle \rightarrow (\mathtt{[void]}, \sigma_5)}$$

Aggiornamento di $\mathrm{SEM}_{\mathtt{STM}}$ per `For` - 1

$$\textbf{(SU1)}\quad \frac{\begin{array}{c}\langle \mathtt{dexp}, \sigma\rangle \to_{\mathtt{DEN}} \lfloor [\mathtt{forIndex}], \mathtt{loc}, \sigma_1 \rfloor \\ \langle \mathtt{exp}, \sigma\rangle \to \lfloor [\mathtt{int}], \mathtt{m}, \sigma_2 \rfloor \\ \sigma_2 = (\Delta_2, \mu_2) \qquad \mu_2(\mathtt{loc}) = \mathtt{n} \\ \langle \mathtt{stm}, \sigma_2\rangle \to ([\mathtt{void}], \sigma_3) \\ \sigma_3 = (\Delta_3, \mu_3) \qquad \mu_4 = \mu_3[\mathtt{loc} \leftarrow \mathtt{n}+1] \\ \mathtt{n} < \mathtt{m} \qquad \sigma_4 = (\Delta_3, \mu_4) \\ \mathtt{loop} = \mathtt{[For]\ dexp\ ([num]\ m)\ stm} \\ \langle \mathtt{loop}, \sigma_4\rangle \to ([\mathtt{void}], \sigma_5)\end{array}}{\langle \mathtt{[For]\ dexp\ exp\ stm}, \sigma\rangle \to ([\mathtt{void}], \sigma_5)}$$

Aggiornamento di $\mathtt{SEM}_{\mathtt{STM}}$ per `For` - 2

$$\text{(SU2)}\quad \frac{\begin{array}{c}\langle \mathtt{dexp}, \sigma\rangle \rightarrow_{\mathtt{DEN}} \lfloor [\mathtt{forIndex}], \mathtt{loc}, \sigma_1 \rfloor \\ \langle \mathtt{exp}, \sigma\rangle \rightarrow \lfloor [\mathtt{int}], \mathtt{m}, \sigma_2 \rfloor \\ \sigma_2 = (\Delta_2, \mu_2) \qquad \mu_2(\mathtt{loc}) = \mathtt{n} \\ \mathtt{n} \geq \mathtt{m}\end{array}}{\langle [\mathtt{For}]\ \ \mathtt{dexp}\ \ \mathtt{exp}\ \ \mathtt{stm}, \sigma\rangle \rightarrow ([\mathtt{void}], \sigma_2)}$$

Aggiornamento di $\mathtt{SEM}_{\mathtt{STM}}$ per `For` - 2

$$\text{(SU2)}\quad \frac{\begin{array}{c}\langle \mathtt{dexp}, \sigma\rangle \rightarrow_{\mathtt{DEN}} \lfloor \mathtt{[forIndex]}, \mathtt{loc}, \sigma_1 \rfloor \\ \langle \mathtt{exp}, \sigma\rangle \rightarrow \lfloor \mathtt{[int]}, \mathtt{m}, \sigma_2 \rfloor \\ \sigma_2 = (\Delta_2, \mu_2) \qquad \mu_2(\mathtt{loc}) = \mathtt{n} \\ \mathtt{n} \geq \mathtt{m}\end{array}}{\langle \mathtt{[For]\ dexp\ exp\ stm}, \sigma\rangle \rightarrow (\mathtt{[void]}, \sigma_2)}$$

Aggiornamento di $\mathtt{SEM}_{\mathtt{STM}}$ per `For` - 2

$$\text{(SU2)}\quad \frac{\begin{array}{c}\langle \mathtt{dexp}, \sigma\rangle \rightarrow_{\mathtt{DEN}} \lfloor \mathtt{[forIndex]}, \mathtt{loc}, \sigma_1 \rfloor \\ \langle \mathtt{exp}, \sigma\rangle \rightarrow \lfloor \mathtt{[int]}, \mathtt{m}, \sigma_2 \rfloor \\ \sigma_2 = (\Delta_2, \mu_2) \qquad \mu_2(\mathtt{loc}) = \mathtt{n} \\ \mathtt{n} \geq \mathtt{m}\end{array}}{\langle \mathtt{[For]\ dexp\ exp\ stm}, \sigma\rangle \rightarrow (\mathtt{[void]}, \sigma_2)}$$

Aggiornamento di $\mathtt{SEM}_{\mathtt{STM}}$ per `For` - 2

$$\text{(SU2)}\quad \frac{\begin{array}{c}\langle \mathtt{dexp}, \sigma\rangle \rightarrow_{\mathtt{DEN}} \lfloor [\mathtt{forIndex}], \mathtt{loc}, \sigma_1 \rfloor \\ \langle \mathtt{exp}, \sigma\rangle \rightarrow \lfloor [\mathtt{int}], \mathtt{m}, \sigma_2 \rfloor \\ \sigma_2 = (\Delta_2, \mu_2) \qquad \mu_2(\mathtt{loc}) = \mathtt{n} \\ \mathtt{n} \geq \mathtt{m}\end{array}}{\langle [\mathtt{For}]\ \mathtt{dexp}\ \mathtt{exp}\ \mathtt{stm}, \sigma\rangle \rightarrow ([\mathtt{void}], \sigma_2)}$$

Aggiornamento di $\mathtt{SEM}_{\mathtt{STM}}$ per `For` - 2

$$\text{(SU2)}\quad \frac{\begin{array}{c}\langle \mathtt{dexp}, \sigma\rangle \rightarrow_{\mathtt{DEN}} \lfloor [\mathtt{forIndex}], \mathtt{loc}, \sigma_1 \rfloor \\ \langle \mathtt{exp}, \sigma\rangle \rightarrow \lfloor [\mathtt{int}], \mathtt{m}, \sigma_2 \rfloor \\ \sigma_2 = (\Delta_2, \mu_2) \qquad \mu_2(\mathtt{loc}) = \mathtt{n} \\ \mathtt{n} \geq \mathtt{m}\end{array}}{\langle [\mathtt{For}]\ \mathtt{dexp}\ \mathtt{exp}\ \mathtt{stm}, \sigma\rangle \rightarrow ([\mathtt{void}], \sigma_2)}$$

Aggiornamento del sistema Y per `ForInit`

$$\text{(YU4)}\quad \frac{\begin{array}{c}\langle \texttt{exp1}, Y_\rho\rangle \to_Y (\texttt{[int]}, Y_\rho) \\ \langle \texttt{exp2}, Y_\rho\rangle \to_Y (\texttt{[int]}, Y_\rho) \\ Y'_\rho = \texttt{[ide/[forIndex]]} \otimes Y_\rho \\ \langle \texttt{For ([val] ide) exp2 stm}, Y'_\rho\rangle \to_Y (\texttt{[void]}, Y'_\rho)\end{array}}{\langle \texttt{[ForInit] ide exp1 exp2 stm}, Y_\rho\rangle \to_Y (\texttt{[void]}, Y'_\rho)}$$

Aggiornamento del sistema Y per `ForInit` - Errori

$$\text{(EU4)}\quad \frac{\langle \texttt{exp1}, Y_\rho\rangle \rightarrow_Y (\texttt{t}, Y_\rho) \quad \texttt{t} \neq \texttt{[int]}}{\langle \texttt{[ForInit] ide exp1 exp2 stm}, Y_\rho\rangle \rightarrow_Y (\texttt{[terr]}, Y'_\rho)}$$

$$\text{(EU5)}\quad \frac{\langle \texttt{exp2}, Y_\rho\rangle \rightarrow_Y (\texttt{t}, Y_\rho) \quad \texttt{t} \neq \texttt{[int]}}{\langle \texttt{[ForInit] ide exp1 exp2 stm}, Y_\rho\rangle \rightarrow_Y (\texttt{[terr]}, Y'_\rho)}$$

$$\text{(EU6)}\quad \frac{\langle \texttt{exp1}, Y_\rho\rangle \rightarrow_Y (\texttt{[int]}, Y_\rho) \quad Y'_\rho = \texttt{[ide/[forIndex]]} \otimes Y_\rho \quad \langle \texttt{[For] ([val] ide) exp2 stm}, Y'_\rho\rangle \rightarrow_Y (\texttt{[terr]}, Y'_\rho)}{\langle \texttt{[ForInit] ide exp1 exp2 stm}, Y_\rho\rangle \rightarrow_Y (\texttt{[terr]}, Y'_\rho)}$$

Aggiornamento di SEM_{STM} per `ForInit`

$$\text{(SU3)}\quad \frac{\begin{array}{c}\langle \texttt{exp1}, \sigma\rangle \to \lfloor [\texttt{int}], \texttt{n}, \sigma' \rfloor \\ \langle \texttt{exp2}, \sigma'\rangle \to \lfloor [\texttt{int}], \texttt{m}, \sigma'' \rfloor \\ \texttt{ar} = \{[], 1, [], [], []\} \\ \sigma'' = (\Delta, \mu) \qquad \Delta_1 = \texttt{ar} + \Delta \\ (\texttt{loc}, \mu') = \triangleright(\mu, [\texttt{forIndex}], 1) \\ \mu_2 = \mu'[\texttt{loc} \leftarrow \texttt{n}] \\ \Delta_2 = [\texttt{ide}/[\texttt{forIndex}], \texttt{loc}] \otimes \Delta_1 \\ \texttt{loop} = \texttt{For ([val] ide) ([num] m) stm} \\ \langle \texttt{loop}, (\Delta_2, \mu_2)\rangle \to ([\texttt{void}], \sigma_F) \\ \sigma_F = (\texttt{ar}_F + \Delta_F, \mu_F) \\ \sigma'_F = (\Delta_F, \mu_F)\end{array}}{\langle [\texttt{ForInit}]\ \texttt{ide exp1 exp2 stm}, \sigma\rangle \to ([\texttt{void}], \sigma'_F)}$$

Aggiornamento di $\mathrm{SEM}_{\mathrm{STM}}$ per ForInit

$$\text{(SU3)}\quad \frac{\begin{array}{c}\langle \texttt{exp1}, \sigma\rangle \to \lfloor [\texttt{int}], \texttt{n}, \sigma' \rfloor \\ \langle \texttt{exp2}, \sigma'\rangle \to \lfloor [\texttt{int}], \texttt{m}, \sigma'' \rfloor \\ \texttt{ar} = \{[], 1, [], [], []\} \\ \sigma'' = (\Delta, \mu) \qquad \Delta_1 = \texttt{ar} + \Delta \\ (\texttt{loc}, \mu') = \triangleright(\mu, [\texttt{forIndex}], 1) \\ \mu_2 = \mu'[\texttt{loc} \leftarrow \texttt{n}] \\ \Delta_2 = [\texttt{ide}/[\texttt{forIndex}], \texttt{loc}] \otimes \Delta_1 \\ \texttt{loop} = \texttt{For ([val] ide) ([num] m) stm} \\ \langle \texttt{loop}, (\Delta_2, \mu_2)\rangle \to ([\texttt{void}], \sigma_F) \\ \sigma_F = (\texttt{ar}_F + \Delta_F, \mu_F) \\ \sigma'_F = (\Delta_F, \mu_F)\end{array}}{\langle [\texttt{ForInit}]\ \texttt{ide exp1 exp2 stm}, \sigma\rangle \to ([\texttt{void}], \sigma'_F)}$$

Aggiornamento di $\mathrm{SEM}_{\mathtt{STM}}$ per `ForInit`

$$
\frac{\begin{array}{c}
\langle \texttt{exp1}, \sigma \rangle \rightarrow \lfloor [\texttt{int}], \texttt{n}, \sigma' \rfloor \\
\langle \texttt{exp2}, \sigma' \rangle \rightarrow \lfloor [\texttt{int}], \texttt{m}, \sigma'' \rfloor \\
\texttt{ar} = \{[], 1, [], [], []\} \\
\sigma'' = (\Delta, \mu) \qquad \Delta_1 = \texttt{ar} + \Delta \\
(\texttt{loc}, \mu') = \triangleright(\mu, [\texttt{forIndex}], 1) \\
\mu_2 = \mu'[\texttt{loc} \leftarrow \texttt{n}] \\
\Delta_2 = [\texttt{ide}/[\texttt{forIndex}], \texttt{loc}] \otimes \Delta_1 \\
\texttt{loop} = \texttt{For ([val] ide) ([num] m) stm} \\
\langle \texttt{loop}, (\Delta_2, \mu_2) \rangle \rightarrow ([\texttt{void}], \sigma_F) \\
\sigma_F = (\texttt{ar}_F + \Delta_F, \mu_F) \\
\sigma'_F = (\Delta_F, \mu_F)
\end{array}}{\langle [\texttt{ForInit}] \ \texttt{ide exp1 exp2 stm}, \sigma \rangle \rightarrow ([\texttt{void}], \sigma'_F)} \ (\texttt{SU3})
$$

Aggiornamento di $\mathrm{SEM}_{\mathtt{STM}}$ per ForInit

$$\text{(SU3)}\quad \frac{\begin{array}{c}\langle \texttt{exp1}, \sigma\rangle \to \lfloor [\texttt{int}], \texttt{n}, \sigma' \rfloor \\ \langle \texttt{exp2}, \sigma'\rangle \to \lfloor [\texttt{int}], \texttt{m}, \sigma'' \rfloor \\ \texttt{ar} = \{[], 1, [], [], []\} \\ \sigma'' = (\Delta, \mu) \qquad \Delta_1 = \texttt{ar} + \Delta \\ (\texttt{loc}, \mu') = \triangleright(\mu, [\texttt{forIndex}], 1) \\ \mu_2 = \mu'[\texttt{loc} \leftarrow \texttt{n}] \\ \Delta_2 = [\texttt{ide}/[\texttt{forIndex}], \texttt{loc}] \otimes \Delta_1 \\ \texttt{loop} = \texttt{For ([val] ide) ([num] m) stm} \\ \langle \texttt{loop}, (\Delta_2, \mu_2)\rangle \to ([\texttt{void}], \sigma_F) \\ \sigma_F = (\texttt{ar}_F + \Delta_F, \mu_F) \\ \sigma'_F = (\Delta_F, \mu_F)\end{array}}{\langle [\texttt{ForInit}]\ \texttt{ide exp1 exp2 stm}, \sigma\rangle \to ([\texttt{void}], \sigma'_F)}$$

Aggiornamento di SEM_{STM} per ForInit

$$\text{(SU3)}\quad \frac{\begin{array}{c}\langle \texttt{exp1}, \sigma\rangle \to \lfloor [\texttt{int}], \texttt{n}, \sigma' \rfloor \\ \langle \texttt{exp2}, \sigma'\rangle \to \lfloor [\texttt{int}], \texttt{m}, \sigma'' \rfloor \\ \texttt{ar} = \{[], 1, [], [], []\} \\ \sigma'' = (\Delta, \mu) \qquad \Delta_1 = \texttt{ar} + \Delta \\ (\texttt{loc}, \mu') = \triangleright(\mu, [\texttt{forIndex}], 1) \\ \mu_2 = \mu'[\texttt{loc} \leftarrow \texttt{n}] \\ \Delta_2 = [\texttt{ide}/[\texttt{forIndex}], \texttt{loc}] \otimes \Delta_1 \\ \texttt{loop} = \texttt{For ([val] ide) ([num] m) stm} \\ \langle \texttt{loop}, (\Delta_2, \mu_2)\rangle \to ([\texttt{void}], \sigma_F) \\ \sigma_F = (\texttt{ar}_F + \Delta_F, \mu_F) \\ \sigma'_F = (\Delta_F, \mu_F)\end{array}}{\langle [\texttt{ForInit}]\ \texttt{ide exp1 exp2 stm}, \sigma\rangle \to ([\texttt{void}], \sigma'_F)}$$

Aggiornamento di $\mathrm{SEM}_{\mathrm{STM}}$ per ForInit

$$\frac{\begin{array}{c}\langle \texttt{exp1}, \sigma\rangle \rightarrow \lfloor [\texttt{int}], \texttt{n}, \sigma' \rfloor \\ \langle \texttt{exp2}, \sigma'\rangle \rightarrow \lfloor [\texttt{int}], \texttt{m}, \sigma'' \rfloor \\ \texttt{ar} = \{[], 1, [], [], []\} \\ \sigma'' = (\Delta, \mu) \qquad \Delta_1 = \texttt{ar} + \Delta \\ (\texttt{loc}, \mu') = \triangleright(\mu, [\texttt{forIndex}], 1) \\ \mu_2 = \mu'[\texttt{loc} \leftarrow \texttt{n}] \\ \Delta_2 = [\texttt{ide}/[\texttt{forIndex}], \texttt{loc}] \otimes \Delta_1 \\ \texttt{loop} = \texttt{For ([val] ide) ([num] m) stm} \\ \langle \texttt{loop}, (\Delta_2, \mu_2)\rangle \rightarrow ([\texttt{void}], \sigma_F) \\ \sigma_F = (\texttt{ar}_F + \Delta_F, \mu_F) \\ \sigma'_F = (\Delta_F, \mu_F)\end{array}}{\langle [\texttt{ForInit}]\ \texttt{ide exp1 exp2 stm}, \sigma\rangle \rightarrow ([\texttt{void}], \sigma'_F)}\ (\texttt{SU3})$$

Codice - 1

```
type tye =
     Int
     | Bool
     | Lab
     | Void
     | Mut of tye
     | Arr of tye * num
     | Terr
     | Abs of tye * tyeSeq
     | Unit
     | ForIndex
```

Codice - 2

```
stm =
    Upd of exp * exp
    | IfT of exp * stm
    | Goto of lab
    | SeqS of stm * stm
    | BlockS of dcl * stm
    | Call of ide * apars
    | Return of exp
    | ES
    | For of exp * exp * stm
    | ForInit of ide * exp * exp * stm
```

Codice - 3

```
expSem exp (sk,(Store(d,g)as mu)) =
    match exp with
    | Val ide ->
        (match getS sk ide with
        | DVar(Mut t,loct) when (isSimple t)
            -> let vt = mTOe(getStore mu loct) in
                (t,vt,(sk,mu))
        | DVar(Mut t,loct)
            -> raise(TypeErrorE("E16: expSem -", exp))
        | DVar(ForIndex, loc)
            -> let vt = mTOe(getStore mu loc) in
            (Int, vt, (sk, mu))
        | DConst(t,v) -> (t,v,(sk,mu))
        |_ -> let msg = "UnExpected binding in Env" in
            raise(SystemErrorE("expSem: "^msg)))
```

Codice - 4

```
dexpSem dexp (sk,(Store(d,g)as mu)) =
    match dexp with
        | Val ide -> (
            match getS sk ide with
            | DVar(Mut t,loct)
                when isSimple t
                -> (Mut t,loct,(sk,mu))
            | DVar(Mut t,loct)
                -> raise(TypeErrorE("E16: dexpSem - ",dexp))
            | DVar (ForIndex, loct)
                -> (ForIndex, loct, (sk, mu))
            | _  -> raise(TypeErrorE("E17: dexpSem - ",dexp)
              ))
```

Codice - 5

```
stmSem stm (sk,(Store(d,g)as mu)) =
    match stm with
    | For (dexp, exp, innerstm) -> (
        match dexpSem dexp (sk,mu) with
        | (ForIndex, loc, sigma1) -> (
            match expSem exp sigma1 with
            | (Int, Ival m, sigma2) ->
                let (_, mu2) = sigma2 in (
                match getStore mu2 loc with
                | MvalI n ->
                    if m > n then
                        let (_, sigma3) = stmSem innerstm
                            sigma2 in
                        let (delta3, mu3) = sigma3 in
                        let mu4 = upd mu3 loc (MvalI (n+1))
                            in
                        let sigma4 = (delta3, mu4) in
                        let loop = For (dexp, (N m),
                            innerstm) in
                        stmSem loop sigma4
                    else
                       (Void, sigma2)
                | _ ->
                    raise(SystemError("System␣error")))
            | _ -> raise(TypeErrorS("EU2:␣stmSem", stm)))
        | _ -> raise(TypeErrorS("EU1:␣stmSem", stm)))
```

Codice - 6

```
stmSem stm (sk,(Store(d,g)as mu)) =
    match stm with
    | ForInit (ide, exp1, exp2, innerstm) -> (
        match expSem exp1 (sk, mu) with
        | (Int, vi, sigma) -> (
        match expSem exp2 sigma with
        | (Int, Ival vf, (delta, mue)) -> (
            let artop = mkAR5 NoneH 1 (emptyEnv()) [] None
                in
            let delta1 = push sk artop in
            let (loc, mup) = allocate mue 1 in
            let den = DVar (ForIndex, loc) in
            let delta2 = bindS delta1 ide den in
            let mu2 = upd mup loc (eTOm vi) in
            let loop = For ((Val ide), (N vf), innerstm) in
            let (_, (deltaF, muF)) = stmSem loop (delta2,
                mu2) in
            let sigmaF = (pop deltaF, muF) in
            (Void, sigmaF)
        )
        | _ -> raise(TypeErrorS("EU5:␣stmSem", stm))
        )
        | _ -> raise(TypeErrorS("EU4:␣stmSem", stm))
    )
```

Esempi - 1

```
Program my_program {
      int[10] array;
      int sum = 0;
      For i from 0 to 10 do
         array[i] = i;
      For i from 0 to 10 do
         sum = (sum + array[i]);
 }
```

```
Stack:
>{my_program,0,[sum/(Mint,L10);
 array/(:Mint[10],L0)],:cmdNext:,[N]}
]
Store:
[L0<-0,L1<-1,L2<-2,L3<-3,L4<-4,L5<-5,L6<-6,L7<-7,L8
    <-8,L9<-9,L10<-45,L11<-10,L12<-10]
```

Esempi - 2

```
Program my_program {
    void foo(value int x){
    x = 100;
    }
    For i from 0 to 10 do
        foo(i);
}
```

```
Stack:
>{my_program,0,[foo/([void::int],$foo,[void::int],:
    fpar:,:cmd:,1$)],:cmdNext:,[N]}
]
Store:
[L0<-10,L1<-100,L2<-100,L3<-100,L4<-100,L5<-100,L6
    <-100,L7<-100,L8<-100,L9<-100,L10<-100]
```

Esempi - 3

```
Program my_program {
    void foo(ref int x){
    x = 100;
    }
    For i from 0 to 10 do
        foo(i);
}
```

```
Exception: TypeErrorT "E61.1: Trasmission: Types
    Mismatch".
```

Esempi - 4

```
Program my_program {
      For i from 0 to 10 do
          i = 0;
}
```

```
Exception: TypeErrorS ("E38: stmSem", Upd (Val "i", N
    0)).
```

Esempi - 5

```
Program my_program {
      int n = 10;
      For i from 0 to n do
         n = (n + 1);
}
```

```
Stack:
>{my_program,0,[n/(Mint,L0)],:cmdNext:,[N]} ]
Store:
[L0<-20,L1<-10]
```

Grazie per l'attenzione!

- Marco Bellia, *Linguaggi di Programmazione con Laboratorio*, *http://pages.di.unipi.it/bellia/2021/Matematica/Registro.html*, 2021.
- *K. Jensen, N. Wirth*, *Pascal User Manual and Report*, *Springer-Verlag, New York 1991*.

Gli esempi sono stati prodotti utilizzando OCaml versione 4.12.0.