Espressione array in Small20

Lorenzo Demeio

29 Maggio 2020

Introduzione del costrutto

- Vogliamo introdurre il costrutto di espressione array $\{\texttt{ex}_0, \ldots, \texttt{ex}_{\texttt{k-1}}\}$ che crea un array di `k` elementi.

Introduzione del costrutto

- Vogliamo introdurre il costrutto di espressione array $\{\texttt{ex}_0, \ldots, \texttt{ex}_{\texttt{k-1}}\}$ che crea un array di `k` elementi.
- Una volta che aggiungiamo per le espressioni la possibilità di calcolare valori array, possiamo associare un `l-value` e un `r-value` anche agli identificatori relativi agli array.

Introduzione del costrutto

- Vogliamo introdurre il costrutto di espressione array $\{\texttt{ex}_0, \ldots, \texttt{ex}_{\texttt{k-1}}\}$ che crea un array di `k` elementi.
- Una volta che aggiungiamo per le espressioni la possibilità di calcolare valori array, possiamo associare un `l-value` e un `r-value` anche agli identificatori relativi agli array.
- L'ultimo passo sarà quindi quello di aggiornare l'espressione "`Upd el er`" per renderla compatibile con gli array.

Sintassi del nuovo costrutto

- Per prima cosa descriviamo la Sintassi Astratta del nuovo costrutto

Sintassi del nuovo costrutto

- Per prima cosa descriviamo la Sintassi Astratta del nuovo costrutto

$$\texttt{Exp} ::= \ldots \mid \texttt{[arrayE]}\ \texttt{ExpSeq} \mid \ldots$$

$$\texttt{ExpSeq} ::= \texttt{Exp}\ \texttt{[x]}\ \texttt{ExpSeq} \mid \texttt{Exp}$$

Sintassi del nuovo costrutto

- Per prima cosa descriviamo la Sintassi Astratta del nuovo costrutto

$$\texttt{Exp} ::= \ldots \mid \texttt{[arrayE] ExpSeq} \mid \ldots$$
$$\texttt{ExpSeq} ::= \texttt{Exp [x] ExpSeq} \mid \texttt{Exp}$$

- e la Sintassi Concreta

$$\texttt{Exp} ::= \texttt{Exp or ExpB1} \mid \texttt{ExpB1} \mid \{\texttt{ExpSeq}\}$$
$$\texttt{ExpSeq} ::= \texttt{Exp, ExpSeq} \mid \texttt{Exp}$$

Aggiornamento del Sistema di Tipi Y

Aggiungiamo al Sistema di Tipi le regole relative al nuovo costrutto, ai nuovi valori restituibili da `[val] I` e al nuovo caso di `Upd el er`:

Aggiornamento del Sistema di Tipi Y

Aggiungiamo al Sistema di Tipi le regole relative al nuovo costrutto, ai nuovi valori restituibili da `[val] I` e al nuovo caso di `Upd el er`:

- Costrutto array

$$\text{Y22:}\ \frac{\begin{array}{c}\langle \mathtt{e}_0, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_0}, \mathtt{Y}_\rho) \\ \langle \mathtt{e}_1, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_1}, \mathtt{Y}_\rho) \\ \vdots \\ \langle \mathtt{e}_{\mathtt{k\text{-}1}}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_{\mathtt{k\text{-}1}}}, \mathtt{Y}_\rho) \\ \mathtt{t}_{\mathtt{e}_0} = \ldots = \mathtt{t}_{\mathtt{e}_{\mathtt{k\text{-}1}}} = \mathtt{t} \quad \mathtt{t} \in \mathtt{Simple}\end{array}}{\langle \mathtt{[arrayE]}\ \ [\mathtt{e}_0, \ldots, \mathtt{e}_{\mathtt{k\text{-}1}}], \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{Arr(t,\ k)}, \mathtt{Y}_\rho)}$$

Aggiornamento del Sistema di Tipi Y

Aggiungiamo al Sistema di Tipi le regole relative al nuovo costrutto, ai nuovi valori restituibili da `[val] I` e al nuovo caso di `Upd el er`:

- Costrutto array

$$\text{Y22:}\ \frac{\begin{array}{c}\langle \mathtt{e}_0, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_0}, \mathtt{Y}_\rho) \\ \langle \mathtt{e}_1, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_1}, \mathtt{Y}_\rho) \\ \vdots \\ \langle \mathtt{e}_{\mathtt{k\text{-}1}}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_{\mathtt{k\text{-}1}}}, \mathtt{Y}_\rho) \\ \mathtt{t}_{\mathtt{e}_0} = \ldots = \mathtt{t}_{\mathtt{e}_{\mathtt{k\text{-}1}}} = \mathtt{t} \quad \mathtt{t} \in \mathtt{Simple}\end{array}}{\langle \mathtt{[arrayE]}\ \ [\mathtt{e}_0, \ldots, \mathtt{e}_{\mathtt{k\text{-}1}}], \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{Arr(t,\ k)}, \mathtt{Y}_\rho)}$$

- `l-value` per gli `Ide` di array

$$\text{Y8.1:}\ \frac{\begin{array}{c}\mathtt{Y}_\rho(\mathtt{I}) = \mathtt{[mut]([arr]([mut]\ t, N))} \\ \mathtt{t} \in \mathtt{Simple}\end{array}}{\langle \mathtt{[val]\ I}, \mathtt{Y}_\rho\rangle \to_{\mathtt{DY}} (\mathtt{[mut]([arr]([mut]\ t, N))}, \mathtt{Y}_\rho)}$$

Aggiornamento del Sistema di Tipi Y

Aggiungiamo al Sistema di Tipi le regole relative al nuovo costrutto, ai nuovi valori restituibili da `[val] I` e al nuovo caso di `Upd el er`:

- Costrutto array

$$\text{Y22:}\ \frac{\begin{array}{c}\langle \mathtt{e}_0, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_0}, \mathtt{Y}_\rho) \\ \langle \mathtt{e}_1, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_1}, \mathtt{Y}_\rho) \\ \vdots \\ \langle \mathtt{e}_{\mathtt{k\text{-}1}}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_{\mathtt{k\text{-}1}}}, \mathtt{Y}_\rho) \\ \mathtt{t}_{\mathtt{e}_0} = \ldots = \mathtt{t}_{\mathtt{e}_{\mathtt{k\text{-}1}}} = \mathtt{t} \quad \mathtt{t} \in \mathtt{Simple}\end{array}}{\langle \mathtt{[arrayE]}\ \ [\mathtt{e}_0, \ldots, \mathtt{e}_{\mathtt{k\text{-}1}}], \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{Arr(t,\ k)}, \mathtt{Y}_\rho)}$$

- `l-value` per gli `Ide` di array

$$\text{Y8.1:}\ \frac{\begin{array}{c}\mathtt{Y}_\rho(\mathtt{I}) = \mathtt{[mut]([arr]([mut]\ \ t, N))} \\ \mathtt{t} \in \mathtt{Simple}\end{array}}{\langle \mathtt{[val]\ \ I}, \mathtt{Y}_\rho\rangle \to_{\mathtt{DY}} (\mathtt{[mut]([arr]([mut]\ \ t, N))}, \mathtt{Y}_\rho)}$$

- `r-value` per gli `Ide` di array

$$\text{Y9.1:}\ \frac{\begin{array}{c}\mathtt{Y}_\rho(\mathtt{I}) = \mathtt{[mut]([arr]([mut]\ \ t, N))} \\ \mathtt{t} \in \mathtt{Simple}\end{array}}{\langle \mathtt{[val]\ \ I}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{[arr](t, n)}, \mathtt{Y}_\rho)}$$

Aggiornamento del Sistema di Tipi Y

Aggiungiamo al Sistema di Tipi le regole relative al nuovo costrutto, ai nuovi valori restituibili da `[val] I` e al nuovo caso di `Upd el er`:

- Costrutto array

$$\text{Y22:}\ \frac{\begin{array}{c}\langle \mathtt{e}_0, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_0}, \mathtt{Y}_\rho) \\ \langle \mathtt{e}_1, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_1}, \mathtt{Y}_\rho) \\ \vdots \\ \langle \mathtt{e}_{\mathtt{k\text{-}1}}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_{\mathtt{k\text{-}1}}}, \mathtt{Y}_\rho) \\ \mathtt{t}_{\mathtt{e}_0} = \ldots = \mathtt{t}_{\mathtt{e}_{\mathtt{k\text{-}1}}} = \mathtt{t} \qquad \mathtt{t} \in \mathtt{Simple}\end{array}}{\langle \mathtt{[arrayE]}\ \ [\mathtt{e}_0, \ldots, \mathtt{e}_{\mathtt{k\text{-}1}}], \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{Arr(t,\ k)}, \mathtt{Y}_\rho)}$$

- `l-value` per gli `Ide` di array

$$\text{Y8.1:}\ \frac{\begin{array}{c}\mathtt{Y}_\rho(\mathtt{I}) = \mathtt{[mut]([arr]([mut]\ \ t, N))} \\ \mathtt{t} \in \mathtt{Simple}\end{array}}{\langle \mathtt{[val]\ \ I}, \mathtt{Y}_\rho\rangle \to_{\mathtt{DY}} (\mathtt{[mut]([arr]([mut]\ \ t, N))}, \mathtt{Y}_\rho)}$$

- `r-value` per gli `Ide` di array

$$\text{Y9.1:}\ \frac{\begin{array}{c}\mathtt{Y}_\rho(\mathtt{I}) = \mathtt{[mut]([arr]([mut]\ \ t, N))} \\ \mathtt{t} \in \mathtt{Simple}\end{array}}{\langle \mathtt{[val]\ \ I}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{[arr](t, n)}, \mathtt{Y}_\rho)}$$

- `Upd el er` nel caso di array

$$\text{Y15.1}\ \frac{\begin{array}{c}\langle \mathtt{e_r}, \mathtt{Y}_\rho\rangle \to (\mathtt{t_r}, \mathtt{Y}_\rho) \qquad \langle \mathtt{e_l}, \mathtt{Y}_\rho\rangle \to_{\mathtt{DY}} (\mathtt{t_l}, \mathtt{Y}_\rho) \\ \mathtt{t_l} = \mathtt{[mut]([arr]([mut]\ \ t, N))} \\ \mathtt{t_r} = \mathtt{[arr]}(\mathtt{t_l}, \mathtt{k}) \qquad \mathtt{t_l} = \mathtt{t} \qquad \mathtt{N} = \mathtt{k}\end{array}}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{[arr](t, N)}, \mathtt{Y}_\rho)}$$

Aggiornamento della Semantica

Passiamo ora a descrivere come si devono comportare le nuove funzionalità che vogliamo implementare. Descriviamo cioè le regole per la Semantica

Aggiornamento della Semantica

Passiamo ora a descrivere come si devono comportare le nuove funzionalità che vogliamo implementare. Descriviamo cioè le regole per la Semantica

- Costrutto array

$$\text{X12:}\ \frac{\begin{array}{c}\texttt{k > 0} \\ \langle \texttt{e}_0, \sigma\rangle \to \lfloor \texttt{t}_{\texttt{e}_0}, \texttt{v}_{\texttt{e}_0}, (\rho, \mu_{\texttt{e}_0})\rfloor \\ \langle \texttt{e}_1, (\rho, \mu_{\texttt{e}_0})\rangle \to \lfloor \texttt{t}_{\texttt{e}_1}, \texttt{v}_{\texttt{e}_1}, (\rho, \mu_{\texttt{e}_1})\rfloor \\ \vdots \\ \langle \texttt{e}_{\texttt{k-1}}, (\rho, \mu_{\texttt{e}_{\texttt{k-2}}})\rangle \to \lfloor \texttt{t}_{\texttt{e}_{\texttt{k-1}}}, \texttt{v}_{\texttt{e}_{\texttt{k-1}}}, (\rho, \mu_{\texttt{e}_{\texttt{k-1}}})\rfloor \\ \texttt{t}_{\texttt{e}_0} = \ldots = \texttt{t}_{\texttt{e}_{\texttt{k-1}}} = \texttt{t} \qquad \texttt{t} \in \texttt{Simple} \\ [\texttt{v}_{\texttt{e}_0}, \ldots, \texttt{v}_{\texttt{e}_{\texttt{k-1}}}] \texttt{ = } \texttt{v}_{\texttt{t}}\end{array}}{\langle \texttt{[arrayE] [e}_0, \ldots, \texttt{e}_{\texttt{k-1}}\texttt{]}, \sigma\rangle \to \lfloor \texttt{[arr](t, k)}, \texttt{v}_{\texttt{t}}, (\rho, \mu_{\texttt{e}_{\texttt{k-1}}})\rfloor}$$

Aggiornamento della Semantica

Passiamo ora a descrivere come si devono comportare le nuove funzionalità che vogliamo implementare. Descriviamo cioè le regole per la Semantica

- Costrutto array

$$\text{X12:}\ \frac{\begin{array}{c}\texttt{k > 0} \\ \langle \texttt{e}_0, \sigma\rangle \to \lfloor \texttt{t}_{\texttt{e}_0}, \texttt{v}_{\texttt{e}_0}, (\rho, \mu_{\texttt{e}_0})\rfloor \\ \langle \texttt{e}_1, (\rho, \mu_{\texttt{e}_0})\rangle \to \lfloor \texttt{t}_{\texttt{e}_1}, \texttt{v}_{\texttt{e}_1}, (\rho, \mu_{\texttt{e}_1})\rfloor \\ \vdots \\ \langle \texttt{e}_{\texttt{k-1}}, (\rho, \mu_{\texttt{e}_{\texttt{k-2}}})\rangle \to \lfloor \texttt{t}_{\texttt{e}_{\texttt{k-1}}}, \texttt{v}_{\texttt{e}_{\texttt{k-1}}}, (\rho, \mu_{\texttt{e}_{\texttt{k-1}}})\rfloor \\ \texttt{t}_{\texttt{e}_0} = \ldots = \texttt{t}_{\texttt{e}_{\texttt{k-1}}} = \texttt{t} \qquad \texttt{t} \in \texttt{Simple} \\ \texttt{[}\texttt{v}_{\texttt{e}_0}, \ldots, \texttt{v}_{\texttt{e}_{\texttt{k-1}}}\texttt{]} \texttt{ = } \texttt{v}_\texttt{t}\end{array}}{\langle \texttt{[arrayE] [}\texttt{e}_0, \ldots, \texttt{e}_{\texttt{k-1}}\texttt{]}, \sigma\rangle \to \lfloor \texttt{[arr](t, k)}, \texttt{v}_\texttt{t}, (\rho, \mu_{\texttt{e}_{\texttt{k-1}}})\rfloor}$$

- `l-value` per gli `Ide` di array

$$\text{X4.1:}\ \frac{\begin{array}{c}\rho(\texttt{I}) = (\texttt{[mut]([arr](t,N))}, \texttt{loc}_\texttt{t}\texttt{'}) \\ \texttt{t' } \in \texttt{ Simple} \qquad \texttt{t} = \texttt{[mut] t'}\end{array}}{\langle \texttt{[val] I}, (\rho, \mu)\rangle \to_{\texttt{Den}} \lfloor \texttt{[mut]([arr](t,N))}, \texttt{loc}_\texttt{t}\texttt{'}, (\rho, \mu)\rfloor}$$

Aggiornamento della Semantica

Passiamo ora a descrivere come si devono comportare le nuove funzionalità che vogliamo implementare. Descriviamo cioè le regole per la Semantica

- Costrutto array

$$\text{X12:}\ \frac{\begin{array}{c}\texttt{k} > 0 \\ \langle \texttt{e}_0, \sigma\rangle \to \lfloor \texttt{t}_{\texttt{e}_0}, \texttt{v}_{\texttt{e}_0}, (\rho, \mu_{\texttt{e}_0})\rfloor \\ \langle \texttt{e}_1, (\rho, \mu_{\texttt{e}_0})\rangle \to \lfloor \texttt{t}_{\texttt{e}_1}, \texttt{v}_{\texttt{e}_1}, (\rho, \mu_{\texttt{e}_1})\rfloor \\ \vdots \\ \langle \texttt{e}_{\texttt{k-1}}, (\rho, \mu_{\texttt{e}_{\texttt{k-2}}})\rangle \to \lfloor \texttt{t}_{\texttt{e}_{\texttt{k-1}}}, \texttt{v}_{\texttt{e}_{\texttt{k-1}}}, (\rho, \mu_{\texttt{e}_{\texttt{k-1}}})\rfloor \\ \texttt{t}_{\texttt{e}_0} = \ldots = \texttt{t}_{\texttt{e}_{\texttt{k-1}}} = \texttt{t} \qquad \texttt{t} \in \texttt{Simple} \\ [\texttt{v}_{\texttt{e}_0}, \ldots, \texttt{v}_{\texttt{e}_{\texttt{k-1}}}] \texttt{ = } \texttt{v}_\texttt{t}\end{array}}{\langle \texttt{[arrayE] [e}_0, \ldots, \texttt{e}_{\texttt{k-1}}\texttt{]}, \sigma\rangle \to \lfloor \texttt{[arr](t, k)}, \texttt{v}_\texttt{t}, (\rho, \mu_{\texttt{e}_{\texttt{k-1}}})\rfloor}$$

- l-value per gli Ide di array

$$\text{X4.1:}\ \frac{\begin{array}{c}\rho(\texttt{I}) = (\texttt{[mut]([arr](t,N))}, \texttt{loc}_\texttt{t}\texttt{'}) \\ \texttt{t' } \in \texttt{ Simple} \qquad \texttt{t} = \texttt{[mut] t'}\end{array}}{\langle \texttt{[val] I}, (\rho, \mu)\rangle \to_{\texttt{Den}} \lfloor \texttt{[mut]([arr](t,N))}, \texttt{loc}_\texttt{t}\texttt{'}, (\rho, \mu)\rfloor}$$

- r-value per gli Ide di array

$$\text{X5.1:}\ \frac{\begin{array}{c}\rho(\texttt{I}) = (\texttt{[mut]([arr]([mut] t,N))}, \texttt{loc}_\texttt{t}) \\ \texttt{t} \in \texttt{Simple} \\ \mu(\texttt{loc}_\texttt{t}) = \texttt{v}_0 \ldots \mu(\texttt{loc}_\texttt{d} \oplus \texttt{N-1}) = \texttt{v}_{\texttt{N-1}} \\ [\texttt{v}_0, \ldots, \texttt{v}_{\texttt{N-1}}] = \texttt{v}\end{array}}{\langle \texttt{[val] I}, (\rho, \mu)\rangle \to \lfloor \texttt{[arr]}(\texttt{t}, \texttt{n}), \texttt{v}, (\rho, \mu)\rfloor}$$

Aggiornamento della Semantica

Passiamo ora a descrivere come si devono comportare le nuove funzionalità che vogliamo implementare. Descriviamo cioè le regole per la Semantica

- Costrutto array

$$\text{X12:}\ \frac{\begin{array}{c}\texttt{k > 0}\\ \langle \texttt{e}_0, \sigma\rangle \to \lfloor \texttt{t}_{\texttt{e}_0}, \texttt{v}_{\texttt{e}_0}, (\rho, \mu_{\texttt{e}_0})\rfloor\\ \langle \texttt{e}_1, (\rho, \mu_{\texttt{e}_0})\rangle \to \lfloor \texttt{t}_{\texttt{e}_1}, \texttt{v}_{\texttt{e}_1}, (\rho, \mu_{\texttt{e}_1})\rfloor\\ \vdots\\ \langle \texttt{e}_{\texttt{k-1}}, (\rho, \mu_{\texttt{e}_{\texttt{k-2}}})\rangle \to \lfloor \texttt{t}_{\texttt{e}_{\texttt{k-1}}}, \texttt{v}_{\texttt{e}_{\texttt{k-1}}}, (\rho, \mu_{\texttt{e}_{\texttt{k-1}}})\rfloor\\ \texttt{t}_{\texttt{e}_0} = \ldots = \texttt{t}_{\texttt{e}_{\texttt{k-1}}} = \texttt{t} \qquad \texttt{t} \in \texttt{Simple}\\ [\texttt{v}_{\texttt{e}_0}, \ldots, \texttt{v}_{\texttt{e}_{\texttt{k-1}}}] \texttt{ = } \texttt{v}_\texttt{t}\end{array}}{\langle \texttt{[arrayE] [e}_0, \ldots, \texttt{e}_{\texttt{k-1}}\texttt{]}, \sigma\rangle \to \lfloor \texttt{[arr](t, k)}, \texttt{v}_\texttt{t}, (\rho, \mu_{\texttt{e}_{\texttt{k-1}}})\rfloor}$$

- l-value per gli Ide di array

$$\text{X4.1:}\ \frac{\begin{array}{c}\rho(\texttt{I}) = (\texttt{[mut]([arr](t,N))}, \texttt{loc}_\texttt{t}\texttt{'})\\ \texttt{t' } \in \texttt{ Simple} \qquad \texttt{t = [mut] t'}\end{array}}{\langle \texttt{[val] I}, (\rho, \mu)\rangle \to_{\texttt{Den}} \lfloor \texttt{[mut]([arr](t,N))}, \texttt{loc}_\texttt{t}\texttt{'}, (\rho, \mu)\rfloor}$$

- r-value per gli Ide di array

$$\text{X5.1:}\ \frac{\begin{array}{c}\rho(\texttt{I}) = (\texttt{[mut]([arr]([mut] t,N))}, \texttt{loc}_\texttt{t})\\ \texttt{t} \in \texttt{Simple}\\ \mu(\texttt{loc}_\texttt{t}) = \texttt{v}_0 \ldots \mu(\texttt{loc}_\texttt{d} \oplus \texttt{N-1}) = \texttt{v}_{\texttt{N-1}}\\ [\texttt{v}_0, \ldots, \texttt{v}_{\texttt{N-1}}] = \texttt{v}\end{array}}{\langle \texttt{[val] I}, (\rho, \mu)\rangle \to \lfloor \texttt{[arr](t,n)}, \texttt{v}, (\rho, \mu)\rfloor}$$

- Upd el er nel caso di array

$$\text{X11.1}\ \frac{\begin{array}{c}\langle \texttt{e}_\texttt{r}, \sigma\rangle \to \lfloor \texttt{t}_\texttt{r}, \texttt{v}_\texttt{r}, \sigma_\texttt{r}\rfloor\\ \langle \texttt{e}_\texttt{l}, \sigma_\texttt{r}\rangle \to_{\texttt{DEN}} \lfloor \texttt{t}_\texttt{l}, \texttt{loc}_\texttt{l}, (\rho_\texttt{l}, \mu_\texttt{l})\rfloor\\ \texttt{t}_\texttt{l} = \texttt{[mut]([arr]([mut] t,N))}\\ \texttt{t}_\texttt{r} = \texttt{[arr]}(\texttt{t}_\texttt{l}, \texttt{k}) \qquad \texttt{t}_\texttt{l} = \texttt{t} \qquad \texttt{t} \in \texttt{Simple}\\ \texttt{N = k} \qquad \texttt{v}_\texttt{r} = [\texttt{v}_0, \ldots, \texttt{v}_{\texttt{k-1}}]\\ \mu_\texttt{l}(\texttt{loc}_\texttt{l} \leftarrow \texttt{v}_0) = \mu_1 \ldots \mu_{\texttt{k-1}}(\texttt{loc}_\texttt{l} \oplus (\texttt{k-1}) \leftarrow \texttt{v}_{\texttt{k-1}}) = \mu_\texttt{F}\end{array}}{\langle \texttt{e}_\texttt{l} \texttt{ [=] } \texttt{e}_\texttt{r}, \sigma\rangle \to \lfloor \texttt{[arr]}(\texttt{t}, \texttt{N}), \texttt{v}_\texttt{r}, (\rho_\texttt{l}, \mu_\texttt{F})\rfloor}$$

Gestione degli Errori di Tipo

Dobbiamo modificare alcuni errori precedenti e introdurre i nuovi Errori di Tipo per il costrutto e per le altre espressioni calcolate.

Gestione degli Errori di Tipo

Dobbiamo modificare alcuni errori precedenti e introdurre i nuovi Errori di Tipo per il costrutto e per le altre espressioni calcolate.
Iniziamo con gli errori possibili per il nuovo costrutto

Gestione degli Errori di Tipo

Dobbiamo modificare alcuni errori precedenti e introdurre i nuovi Errori di Tipo per il costrutto e per le altre espressioni calcolate.
Iniziamo con gli errori possibili per il nuovo costrutto

- Il primo errore lo abbiamo se
  $\mathtt{t}_{\mathtt{e}_0} \notin \mathtt{Simple}$

$$\text{E37:}\ \frac{\begin{array}{c}\mathtt{k} > 0 \\ \langle \mathtt{e}_0, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_0}, \mathtt{Y}_\rho) \\ \langle \mathtt{e}_1, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_1}, \mathtt{Y}_\rho) \\ \vdots \\ \langle \mathtt{e}_{\mathtt{k-1}}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_{\mathtt{k-1}}}, \mathtt{Y}_\rho) \\ \mathtt{t}_{\mathtt{e}_0} \notin \mathtt{Simple}\end{array}}{\langle \mathtt{[arrayE]}\ \ [\mathtt{e}_0, \ldots, \mathtt{e}_{\mathtt{k-1}}], \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}_\rho))}$$

Gestione degli Errori di Tipo

Dobbiamo modificare alcuni errori precedenti e introdurre i nuovi Errori di Tipo per il costrutto e per le altre espressioni calcolate.
Iniziamo con gli errori possibili per il nuovo costrutto

- Il primo errore lo abbiamo se $\texttt{t}_{\texttt{e}_0} \notin \texttt{Simple}$
- Il secondo se per qualche i si ha $\texttt{t}_{\texttt{e}_i} \neq \texttt{t}_{\texttt{e}_0}$

$$\text{E37:}\ \frac{\begin{array}{c}\texttt{k} > 0 \\ \langle \texttt{e}_0, \texttt{Y}_\rho \rangle \rightarrow_{\texttt{Y}} (\texttt{t}_{\texttt{e}_0}, \texttt{Y}_\rho) \\ \langle \texttt{e}_1, \texttt{Y}_\rho \rangle \rightarrow_{\texttt{Y}} (\texttt{t}_{\texttt{e}_1}, \texttt{Y}_\rho) \\ \vdots \\ \langle \texttt{e}_{\texttt{k-1}}, \texttt{Y}_\rho \rangle \rightarrow_{\texttt{Y}} (\texttt{t}_{\texttt{e}_{\texttt{k-1}}}, \texttt{Y}_\rho) \\ \texttt{t}_{\texttt{e}_0} \notin \texttt{Simple}\end{array}}{\langle [\texttt{arrayE}]\ \ [\texttt{e}_0, \ldots, \texttt{e}_{\texttt{k-1}}], \texttt{Y}_\rho \rangle \rightarrow_{\texttt{Y}} ([\texttt{terr}], \texttt{Y}_\rho))}$$

$$\text{E38:}\ \frac{\begin{array}{c}\texttt{k} > 0 \\ \langle \texttt{e}_0, \texttt{Y}_\rho \rangle \rightarrow_{\texttt{Y}} (\texttt{t}_{\texttt{e}_0}, \texttt{Y}_\rho) \\ \langle \texttt{e}_1, \texttt{Y}_\rho \rangle \rightarrow_{\texttt{Y}} (\texttt{t}_{\texttt{e}_1}, \texttt{Y}_\rho) \\ \vdots \\ \langle \texttt{e}_{\texttt{k-1}}, \texttt{Y}_\rho \rangle \rightarrow_{\texttt{Y}} (\texttt{t}_{\texttt{e}_{\texttt{k-1}}}, \texttt{Y}_\rho) \\ \texttt{t}_{\texttt{e}_0} \in \texttt{Simple} \quad \exists \texttt{i} \in \{0, \ldots, \texttt{k-1}\}\ \texttt{t.c.}\ \ \texttt{t}_{\texttt{e}_0} \neq \texttt{t}_{\texttt{e}_i}\end{array}}{\langle [\texttt{arrayE}]\ \ [\texttt{e}_0, \ldots, \texttt{e}_{\texttt{k-1}}], \texttt{Y}_\rho \rangle \rightarrow_{\texttt{Y}} ([\texttt{terr}], \texttt{Y}_\rho))}$$

Gestione degli Errori di Tipo

Dobbiamo modificare alcuni errori precedenti e introdurre i nuovi Errori di Tipo per il costrutto e per le altre espressioni calcolate.
Iniziamo con gli errori possibili per il nuovo costrutto

- Il primo errore lo abbiamo se $\mathtt{t}_{\mathtt{e}_0} \notin \mathtt{Simple}$

$$\text{E37:}\ \frac{\begin{array}{c}\mathtt{k} > 0 \\ \langle \mathtt{e}_0, \mathtt{Y}_\rho \rangle \to_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_0}, \mathtt{Y}_\rho) \\ \langle \mathtt{e}_1, \mathtt{Y}_\rho \rangle \to_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_1}, \mathtt{Y}_\rho) \\ \vdots \\ \langle \mathtt{e}_{\mathtt{k-1}}, \mathtt{Y}_\rho \rangle \to_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_{\mathtt{k-1}}}, \mathtt{Y}_\rho) \\ \mathtt{t}_{\mathtt{e}_0} \notin \mathtt{Simple}\end{array}}{\langle [\mathtt{arrayE}]\ \ [\mathtt{e}_0, \ldots, \mathtt{e}_{\mathtt{k-1}}], \mathtt{Y}_\rho \rangle \to_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}_\rho))}$$

- Il secondo se per qualche i si ha $\mathtt{t}_{\mathtt{e}_i} \neq \mathtt{t}_{\mathtt{e}_0}$

$$\text{E38:}\ \frac{\begin{array}{c}\mathtt{k} > 0 \\ \langle \mathtt{e}_0, \mathtt{Y}_\rho \rangle \to_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_0}, \mathtt{Y}_\rho) \\ \langle \mathtt{e}_1, \mathtt{Y}_\rho \rangle \to_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_1}, \mathtt{Y}_\rho) \\ \vdots \\ \langle \mathtt{e}_{\mathtt{k-1}}, \mathtt{Y}_\rho \rangle \to_{\mathtt{Y}} (\mathtt{t}_{\mathtt{e}_{\mathtt{k-1}}}, \mathtt{Y}_\rho) \\ \mathtt{t}_{\mathtt{e}_0} \in \mathtt{Simple} \quad \exists \mathtt{i} \in \{0, \ldots, \mathtt{k-1}\}\ \mathtt{t.c.}\ \ \mathtt{t}_{\mathtt{e}_0} \neq \mathtt{t}_{\mathtt{e}_i}\end{array}}{\langle [\mathtt{arrayE}]\ \ [\mathtt{e}_0, \ldots, \mathtt{e}_{\mathtt{k-1}}], \mathtt{Y}_\rho \rangle \to_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}_\rho))}$$

- Il terzo se la lista di espressioni è vuota

$$\text{E39:}\ \frac{}{\langle [\mathtt{arrayE}]\ \ [], \mathtt{Y}_\rho \rangle \to_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}_\rho))}$$

Gestione degli Errori di Tipo

Per gli `r-value` di `Ide` relativi ad array modifichiamo la regola E8 e aggiungiamo un altro errore possibile.

Gestione degli Errori di Tipo

Per gli `r-value` di `Ide` relativi ad array modifichiamo la regola E8 e aggiungiamo un altro errore possibile.

$$\text{E8:}\ \frac{\begin{array}{c}\mathtt{Y}_\rho(\mathtt{I}) = \texttt{[mut] t} \\ \texttt{t} \notin \texttt{Simple AND t} \neq \texttt{[arr]([mut] t'},\mathtt{N})\end{array}}{\langle \texttt{[val] I}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\texttt{[terr]}, \mathtt{Y}_\rho)}$$

Gestione degli Errori di Tipo

Per gli `r-value` di `Ide` relativi ad array modifichiamo la regola E8 e aggiungiamo un altro errore possibile.

$$\text{E8:}\ \frac{\mathtt{Y}_\rho(\mathtt{I}) = \mathtt{[mut]\ t} \qquad \mathtt{t} \notin \mathtt{Simple\ \ AND\ \ t} \neq \mathtt{[arr]([mut]\ t',N)}}{\langle \mathtt{[val]\ \ I}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E9.1:}\ \frac{\mathtt{Y}_\rho(\mathtt{I}) = \mathtt{[mut]([arr]([mut]\ t,N))} \qquad \mathtt{t} \notin \mathtt{Simple}}{\langle \mathtt{[val]\ \ I}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

Gestione degli Errori di Tipo

Per gli `r-value` di `Ide` relativi ad array modifichiamo la regola E8 e aggiungiamo un altro errore possibile.

$$\text{E8:}\ \frac{\mathtt{Y}_\rho(\mathtt{I}) = \mathtt{[mut]\ t} \qquad \mathtt{t} \notin \mathtt{Simple\ AND\ t} \neq \mathtt{[arr]([mut]\ t',N)}}{\langle \mathtt{[val]\ I}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E9.1:}\ \frac{\mathtt{Y}_\rho(\mathtt{I}) = \mathtt{[mut]([arr]([mut]\ t,N))} \qquad \mathtt{t} \notin \mathtt{Simple}}{\langle \mathtt{[val]\ I}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

Per gli `l-value` aggiungiamo un ulteriore errore possibile.

$$\text{E9.2:}\ \frac{\mathtt{Y}_\rho(\mathtt{I}) = \mathtt{[mut]([arr](t,N))} \qquad \mathtt{t} \neq \mathtt{[mut]\ t'}}{\langle \mathtt{[val]\ I}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{DY}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

Gestione degli Errori di Tipo

Per l'espressione `Upd el er` dobbiamo modificare gli errori `E17`, `E18` ed `E19` e aggiungerne altri tre relativi agli array.

Gestione degli Errori di Tipo

Per l'espressione `Upd el er` dobbiamo modificare gli errori `E17`, `E18` ed `E19` e aggiungerne altri tre relativi agli array.

$$\text{E17:}\ \frac{\langle \texttt{e}_\texttt{l}, \texttt{Y}_\rho \rangle \rightarrow_{\texttt{DY}} (\texttt{[mut]}\ \ \texttt{t}, \texttt{Y}_\rho) \quad \texttt{t} \notin \texttt{Simple AND t} \neq \texttt{[mut]([arr]([mut] t'}, \texttt{N}))}{\langle \texttt{e}_\texttt{l}\ \ \texttt{[=]}\ \ \texttt{e}_\texttt{r}, \texttt{Y}_\rho \rangle \rightarrow_{\texttt{Y}} (\texttt{[terr]}, \texttt{Y}_\rho)}$$

Gestione degli Errori di Tipo

Per l'espressione `Upd el er` dobbiamo modificare gli errori `E17`, `E18` ed `E19` e aggiungerne altri tre relativi agli array.

$$\text{E17:}\ \frac{\langle \texttt{e}_\texttt{l}, \texttt{Y}_\rho \rangle \rightarrow_{\texttt{DY}} (\texttt{[mut]}\ \ \texttt{t}, \texttt{Y}_\rho) \quad \texttt{t} \notin \texttt{Simple AND t} \neq \texttt{[mut]([arr]([mut]\ t'},\texttt{N}))}{\langle \texttt{e}_\texttt{l}\ \ \texttt{[=]}\ \ \texttt{e}_\texttt{r}, \texttt{Y}_\rho \rangle \rightarrow_{\texttt{Y}} (\texttt{[terr]}, \texttt{Y}_\rho)}$$

$$\text{E18:}\ \frac{\langle \texttt{e}_\texttt{r}, \texttt{Y}_\rho \rangle \rightarrow_{\texttt{Y}} (\texttt{t}, \texttt{Y}_\rho) \quad \texttt{t} \notin \texttt{Simple AND t} \neq \texttt{([arr](t'},\texttt{N}))}{\langle \texttt{e}_\texttt{l}\ \ \texttt{[=]}\ \ \texttt{e}_\texttt{r}, \texttt{Y}_\rho \rangle \rightarrow_{\texttt{Y}} (\texttt{[terr]}, \texttt{Y}_\rho)}$$

Gestione degli Errori di Tipo

Per l'espressione `Upd el er` dobbiamo modificare gli errori `E17`, `E18` ed `E19` e aggiungerne altri tre relativi agli array.

$$\text{E17:}\ \frac{\langle \mathtt{e_l}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{DY}} (\mathtt{[mut]\ t}, \mathtt{Y}_\rho) \quad \mathtt{t} \notin \mathtt{Simple\ \ AND\ t} \neq \mathtt{[mut]([arr]([mut]\ t'}, \mathtt{N}))}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E18:}\ \frac{\langle \mathtt{e_r}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}, \mathtt{Y}_\rho) \quad \mathtt{t} \notin \mathtt{Simple\ \ AND\ t} \neq \mathtt{([arr](t'}, \mathtt{N}))}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E19:}\ \frac{\langle \mathtt{e_l}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{DY}} (\mathtt{[mut]t_l}, \mathtt{Y}_\rho) \quad \langle \mathtt{e_r}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{t_r}, \mathtt{Y}_\rho) \quad \mathtt{t_l} \neq \mathtt{t_r} \quad \mathtt{t_l} \in \mathtt{Simple}}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

Gestione degli Errori di Tipo

Per l'espressione `Upd el er` dobbiamo modificare gli errori `E17`, `E18` ed `E19` e aggiungerne altri tre relativi agli array.

$$\text{E17:}\ \frac{\langle \mathtt{e_l}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{DY}} (\mathtt{[mut]\ t}, \mathtt{Y}_\rho) \quad \mathtt{t} \notin \mathtt{Simple\ \ AND\ t} \neq \mathtt{[mut]([arr]([mut]\ t'}, \mathtt{N}))}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E18:}\ \frac{\langle \mathtt{e_r}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}, \mathtt{Y}_\rho) \quad \mathtt{t} \notin \mathtt{Simple\ \ AND\ t} \neq \mathtt{([arr](t'}, \mathtt{N}))}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E19:}\ \frac{\langle \mathtt{e_l}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{DY}} (\mathtt{[mut]t_1}, \mathtt{Y}_\rho) \quad \langle \mathtt{e_r}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{t_r}, \mathtt{Y}_\rho) \quad \mathtt{t_1} \neq \mathtt{t_r} \quad \mathtt{t_1} \in \mathtt{Simple}}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E19.1:}\ \frac{\langle \mathtt{e_r}, \mathtt{Y}_\rho\rangle \rightarrow (\mathtt{t_r}, \mathtt{Y}_\rho) \quad \langle \mathtt{e_l}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{DY}} (\mathtt{t_1}, \mathtt{Y}_\rho) \quad \mathtt{t_1} = \mathtt{[mut]([arr]([mut]\ t}, \mathtt{N})) \quad \mathtt{t_r} = \mathtt{[arr]}(\mathtt{t_1}, \mathtt{k}) \quad \mathtt{t_1} \neq \mathtt{t}}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

Gestione degli Errori di Tipo

Per l'espressione `Upd el er` dobbiamo modificare gli errori `E17`, `E18` ed `E19` e aggiungerne altri tre relativi agli array.

$$\text{E17:}\ \frac{\langle \mathtt{e_l}, \mathtt{Y}_\rho\rangle \to_{\mathtt{DY}} (\mathtt{[mut]\ t}, \mathtt{Y}_\rho) \quad \mathtt{t} \notin \mathtt{Simple\ \ AND\ t} \neq \mathtt{[mut]([arr]([mut]\ t'}, \mathtt{N}))}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E18:}\ \frac{\langle \mathtt{e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{t}, \mathtt{Y}_\rho) \quad \mathtt{t} \notin \mathtt{Simple\ \ AND\ t} \neq \mathtt{([arr](t'}, \mathtt{N}))}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E19:}\ \frac{\begin{array}{c}\langle \mathtt{e_l}, \mathtt{Y}_\rho\rangle \to_{\mathtt{DY}} (\mathtt{[mut]t_l}, \mathtt{Y}_\rho) \\ \langle \mathtt{e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{t_r}, \mathtt{Y}_\rho) \\ \mathtt{t_l} \neq \mathtt{t_r} \qquad \mathtt{t_l} \in \mathtt{Simple}\end{array}}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E19.1:}\ \frac{\begin{array}{c}\langle \mathtt{e_r}, \mathtt{Y}_\rho\rangle \to (\mathtt{t_r}, \mathtt{Y}_\rho) \quad \langle \mathtt{e_l}, \mathtt{Y}_\rho\rangle \to_{\mathtt{DY}} (\mathtt{t_l}, \mathtt{Y}_\rho) \\ \mathtt{t_l} = \mathtt{[mut]([arr]([mut]\ t}, \mathtt{N})) \\ \mathtt{t_r} = \mathtt{[arr]}(\mathtt{t_1}, \mathtt{k}) \qquad \mathtt{t_1} \neq \mathtt{t}\end{array}}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E19.2:}\ \frac{\begin{array}{c}\langle \mathtt{e_r}, \mathtt{Y}_\rho\rangle \to (\mathtt{t_r}, \mathtt{Y}_\rho) \quad \langle \mathtt{e_l}, \mathtt{Y}_\rho\rangle \to_{\mathtt{DY}} (\mathtt{t_l}, \mathtt{Y}_\rho) \\ \mathtt{t_l} = \mathtt{[mut]([arr]([mut]\ t}, \mathtt{N})) \\ \mathtt{t_r} = \mathtt{[arr]}(\mathtt{t_1}, \mathtt{k}) \qquad \mathtt{t_1} = \mathtt{t} \qquad \mathtt{N} \neq \mathtt{k}\end{array}}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

Gestione degli Errori di Tipo

Per l'espressione `Upd el er` dobbiamo modificare gli errori `E17`, `E18` ed `E19` e aggiungerne altri tre relativi agli array.

$$\text{E17:}\ \frac{\langle \mathtt{e_l}, \mathtt{Y}_\rho\rangle \to_{\mathtt{DY}} (\mathtt{[mut]\ t}, \mathtt{Y}_\rho) \quad \mathtt{t} \notin \mathtt{Simple\ \ AND\ \ t} \neq \mathtt{[mut]([arr]([mut]\ t'}, \mathtt{N}))}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E18:}\ \frac{\langle \mathtt{e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{t}, \mathtt{Y}_\rho) \quad \mathtt{t} \notin \mathtt{Simple\ \ AND\ \ t} \neq \mathtt{([arr](t'}, \mathtt{N}))}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E19:}\ \frac{\begin{array}{c}\langle \mathtt{e_l}, \mathtt{Y}_\rho\rangle \to_{\mathtt{DY}} (\mathtt{[mut]t_l}, \mathtt{Y}_\rho) \\ \langle \mathtt{e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{t_r}, \mathtt{Y}_\rho) \\ \mathtt{t_l} \neq \mathtt{t_r} \qquad \mathtt{t_l} \in \mathtt{Simple}\end{array}}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E19.1:}\ \frac{\begin{array}{c}\langle \mathtt{e_r}, \mathtt{Y}_\rho\rangle \to (\mathtt{t_r}, \mathtt{Y}_\rho) \quad \langle \mathtt{e_l}, \mathtt{Y}_\rho\rangle \to_{\mathtt{DY}} (\mathtt{t_l}, \mathtt{Y}_\rho) \\ \mathtt{t_l} = \mathtt{[mut]([arr]([mut]\ t}, \mathtt{N})) \\ \mathtt{t_r} = \mathtt{[arr]}(\mathtt{t_1}, \mathtt{k}) \qquad \mathtt{t_1} \neq \mathtt{t}\end{array}}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E19.2:}\ \frac{\begin{array}{c}\langle \mathtt{e_r}, \mathtt{Y}_\rho\rangle \to (\mathtt{t_r}, \mathtt{Y}_\rho) \quad \langle \mathtt{e_l}, \mathtt{Y}_\rho\rangle \to_{\mathtt{DY}} (\mathtt{t_l}, \mathtt{Y}_\rho) \\ \mathtt{t_l} = \mathtt{[mut]([arr]([mut]\ t}, \mathtt{N})) \\ \mathtt{t_r} = \mathtt{[arr]}(\mathtt{t_1}, \mathtt{k}) \qquad \mathtt{t_1} = \mathtt{t} \qquad \mathtt{N} \neq \mathtt{k}\end{array}}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

$$\text{E19.3:}\ \frac{\begin{array}{c}\langle \mathtt{e_r}, \mathtt{Y}_\rho\rangle \to (\mathtt{t_r}, \mathtt{Y}_\rho) \quad \langle \mathtt{e_l}, \mathtt{Y}_\rho\rangle \to_{\mathtt{DY}} (\mathtt{t_l}, \mathtt{Y}_\rho) \\ \mathtt{t_l} = \mathtt{[mut]([arr]([mut]\ t}, \mathtt{N})) \\ \mathtt{t_r} \neq \mathtt{[arr]}(\mathtt{t_1}, \mathtt{k})\end{array}}{\langle \mathtt{e_l\ \ [=]\ \ e_r}, \mathtt{Y}_\rho\rangle \to_{\mathtt{Y}} (\mathtt{[terr]}, \mathtt{Y}_\rho)}$$

Passiamo all'implementazione!

```
1  type eval =
2     Ival of num
3     | Bval of bool
4     | Aval of evalSeq
5     | E
6
7     and
8
9  evalSeq = eval list;;
```

- Aggiungiamo ai possibili valori calcolati da un'espressione quello di Array

Passiamo all'implementazione!

```
1  type eval =
2     Ival of num
3     | Bval of bool
4     | Aval of evalSeq
5     | E
6
7     and
8
9  evalSeq = eval list;;
```

- Aggiungiamo ai possibili valori calcolati da un'espressione quello di Array

- Passiamo ora ad modificare la funzione `expSem`

Passiamo all'implementazione!

```
| ArrayE expS
   when (List.length expS > 0) -> (
       let (te0, _, _) = expSem (nth expS 0) (rho, mu) in
       if (isSimple te0) then
           let f (valS, (rho, mu)) exp = (
               let (te, ve, (rho1, mu1)) = expSem exp (rho,
               ↪  mu) in
               if (ysame te te0) then
                   (valS@[ve], (rho1, mu1))
               else
                   let msg = "Expression '" ^ (toStringExp
                   ↪  exp)
                   ^ "' has type " ^ (toStringTye te)
                   ^ " but type " ^ (toStringTye te0) ^ " is
                   ↪  required" in
                   raise(SystemErrorE("E38 expSem: " ^ msg))
           ) in
           let (valS, (rhoF, muF)) = List.fold_left f ([],
           ↪  (rho, mu)) expS in
           (Arr(te0, List.length expS), Aval valS, (rhoF,
           ↪  muF))
       else
           let msg = "Type " ^ (toStringTye te1) ^ " is not
           ↪  simple" in
               raise(SystemErrorE("E37 expSem: " ^ msg))
       )
```

- Abbiamo aggiunto alla funzione expSem il caso `ArrayE`

Passiamo all'implementazione!

```
| ArrayE expS
   when (List.length expS > 0) -> (
       let (te0, _, _) = expSem (nth expS 0) (rho, mu) in
       if (isSimple te0) then
           let f (valS, (rho, mu)) exp = (
               let (te, ve, (rho1, mu1)) = expSem exp (rho,
               ↪  mu) in
               if (ysame te te0) then
                   (valS@[ve], (rho1, mu1))
               else
                   let msg = "Expression '" ^ (toStringExp
                   ↪  exp)
                   ^ "' has type " ^ (toStringTye te)
                   ^ " but type " ^ (toStringTye te0) ^ " is
                   ↪  required" in
                   raise(SystemErrorE("E38 expSem: " ^ msg))
           ) in
           let (valS, (rhoF, muF)) = List.fold_left f ([],
           ↪  (rho, mu)) expS in
           (Arr(te0, List.length expS), Aval valS, (rhoF,
           ↪  muF))
       else
           let msg = "Type " ^ (toStringTye te1) ^ " is not
           ↪  simple" in
               raise(SystemErrorE("E37 expSem: " ^ msg))
       )
```

- Abbiamo aggiunto alla funzione `expSem` il caso `ArrayE`
- Nel caso in cui $t_{e_0} \in$ `Simple` prova a calcolare i valori delle espressioni

Passiamo all'implementazione!

```
| ArrayE expS
   when (List.length expS > 0) -> (
       let (te0, _, _) = expSem (nth expS 0) (rho, mu) in
       if (isSimple te0) then
           let f (valS, (rho, mu)) exp = (
               let (te, ve, (rho1, mu1)) = expSem exp (rho,
               ↪  mu) in
               if (ysame te te0) then
                   (valS@[ve], (rho1, mu1))
               else
                   let msg = "Expression '" ^ (toStringExp
                   ↪  exp)
                   ^ "' has type " ^ (toStringTye te)
                   ^ " but type " ^ (toStringTye te0) ^ " is
                   ↪  required" in
                   raise(SystemErrorE("E38 expSem: " ^ msg))
           ) in
           let (valS, (rhoF, muF)) = List.fold_left f ([],
           ↪  (rho, mu)) expS in
           (Arr(te0, List.length expS), Aval valS, (rhoF,
           ↪  muF))
       else
           let msg = "Type " ^ (toStringTye te1) ^ " is not
           ↪  simple" in
               raise(SystemErrorE("E37 expSem: " ^ msg))
       )
```

- Abbiamo aggiunto alla funzione `expSem` il caso `ArrayE`
- Nel caso in cui $t_{e_0} \in$ `Simple` prova a calcolare i valori delle espressioni
- Se qualcuna di queste restituisce un tipo diverso da t_{e_0} dà l'errore `E38`

Passiamo all'implementazione!

```
| ArrayE expS
   when (List.length expS > 0) -> (
       let (te0, _, _) = expSem (nth expS 0) (rho, mu) in
       if (isSimple te0) then
           let f (valS, (rho, mu)) exp = (
               let (te, ve, (rho1, mu1)) = expSem exp (rho,
               ↪  mu) in
               if (ysame te te0) then
                   (valS@[ve], (rho1, mu1))
               else
                   let msg = "Expression '" ^ (toStringExp
                   ↪  exp)
                   ^ "' has type " ^ (toStringTye te)
                   ^ " but type " ^ (toStringTye te0) ^ " is
                   ↪  required" in
                   raise(SystemErrorE("E38 expSem: " ^ msg))
           ) in
           let (valS, (rhoF, muF)) = List.fold_left f ([],
           ↪  (rho, mu)) expS in
           (Arr(te0, List.length expS), Aval valS, (rhoF,
           ↪  muF))
       else
           let msg = "Type " ^ (toStringTye te1) ^ " is not
           ↪  simple" in
           raise(SystemErrorE("E37 expSem: " ^ msg))
       )
```

- Abbiamo aggiunto alla funzione `expSem` il caso `ArrayE`
- Nel caso in cui $t_{e_0} \in$ `Simple` prova a calcolare i valori delle espressioni
- Se qualcuna di queste restituisce un tipo diverso da t_{e_0} dà l'errore `E38`
- Se $t_{e_0} \notin$ `Simple` allora dà l'errore `E37`

Passiamo all'implementazione!

```
1  | ArrayE expS ->
2        let msg = "Array cannot be empty" in
3        raise(SystemErrorE("E39 expSem: " ^ msg))
```

- Nel caso in cui l'array passato sia vuoto dà l'errore E39

Passiamo all'implementazione!

```
1  | ArrayE expS ->
2        let msg = "Array cannot be empty" in
3        raise(SystemErrorE("E39 expSem: " ^ msg))
```

- Nel caso in cui l'array passato sia vuoto dà l'errore E39

Aggiorniamo ora il caso `Val ide` della funzione

Passiamo all'implementazione!

```
| ArrayE expS ->
      let msg = "Array cannot be empty" in
      raise(SystemErrorE("E39 expSem: " ^ msg))
```

- Nel caso in cui l'array passato sia vuoto dà l'errore E39

Aggiorniamo ora il caso `Val ide` della funzione

```
let rec locRange a b =
   if a > b then []
   else (Loc a) :: locRange (a+1) b
   ;;
```

- Per farlo ci serviremo di questa funzione per creare un array delle locazioni che ci interessano

Implementazione nel nuovo r-value per Val I

```
| Val ide -> (
    try (let den = getEnv rho ide in
       match den with
       ...
       | DArray(Mut(Arr(Mut t, num)), (Loc n))
           when (isSimple t)
           -> (let locL = locRange n (n + num - 1) in
           let f loc = (
               try mTOe(getStore mu loc)
               with
                   | (UndefinedLoc(_,loc2))
                       -> (let msg = "Alloc. Problems in "
                       ^ (toStringLoc loc)
                       ^ " or in " ^ (toStringLoc loc2) in
                       raise(SystemErrorE("expSem: "^msg)))
                   | (SystemErrorM _)
                       -> (let msg = "Integrity Problems in
                       ↪  Store State" in
                       raise(SystemErrorE("expSem: " ^ msg)))
           ) in
           let valS = List.map f locL in
           (Arr(t, num), Aval valS, (rho, mu)))
       | DArray(Mut(Arr(Mut t, num)), (Loc n))
           -> raise(TypeErrorE("E9.1: expSem - " ^
           ↪  (toStringExp exp)))
```

- Nel caso in cui den sia di tipo array prova ad ottenere i valori dell'array dallo store.

Implementazione nel nuovo r-value per Val I

```
| Val ide -> (
    try (let den = getEnv rho ide in
       match den with
       ...
       | DArray(Mut(Arr(Mut t, num)), (Loc n))
           when (isSimple t)
           -> (let locL = locRange n (n + num - 1) in
           let f loc = (
               try mTOe(getStore mu loc)
               with
                   | (UndefinedLoc(_,loc2))
                       -> (let msg = "Alloc. Problems in "
                       ^ (toStringLoc loc)
                       ^ " or in " ^ (toStringLoc loc2) in
                       raise(SystemErrorE("expSem: "^msg)))
                   | (SystemErrorM _)
                       -> (let msg = "Integrity Problems in
                       ↪  Store State" in
                       raise(SystemErrorE("expSem: " ^ msg)))
           ) in
           let valS = List.map f locL in
           (Arr(t, num), Aval valS, (rho, mu)))
       | DArray(Mut(Arr(Mut t, num)), (Loc n))
           -> raise(TypeErrorE("E9.1: expSem - " ^
           ↪  (toStringExp exp)))
```

- Nel caso in cui den sia di tipo array prova ad ottenere i valori dell'array dallo store.
- Nel caso ci siano problemi di allocazione restituisce errore

Implementazione nel nuovo r-value per Val I

```
| Val ide -> (
    try (let den = getEnv rho ide in
       match den with
       ...
       | DArray(Mut(Arr(Mut t, num)), (Loc n))
           when (isSimple t)
           -> (let locL = locRange n (n + num - 1) in
           let f loc = (
               try mTOe(getStore mu loc)
               with
                   | (UndefinedLoc(_,loc2))
                       -> (let msg = "Alloc. Problems in "
                       ^ (toStringLoc loc)
                       ^ " or in " ^ (toStringLoc loc2) in
                       raise(SystemErrorE("expSem: "^msg)))
                   | (SystemErrorM _)
                       -> (let msg = "Integrity Problems in
                       ↪  Store State" in
                       raise(SystemErrorE("expSem: " ^ msg)))
           ) in
           let valS = List.map f locL in
           (Arr(t, num), Aval valS, (rho, mu)))
       | DArray(Mut(Arr(Mut t, num)), (Loc n))
           -> raise(TypeErrorE("E9.1: expSem - " ^
           ↪  (toStringExp exp)))
```

- Nel caso in cui den sia di tipo array prova ad ottenere i valori dell'array dallo store.
- Nel caso ci siano problemi di allocazione restituisce errore
- Altrimenti restituisce un valore array

Implementazione nel nuovo r-value per Val I

```
| Val ide -> (
    try (let den = getEnv rho ide in
       match den with
       ...
       | DArray(Mut(Arr(Mut t, num)), (Loc n))
           when (isSimple t)
           -> (let locL = locRange n (n + num - 1) in
           let f loc = (
               try mTOe(getStore mu loc)
               with
                   | (UndefinedLoc(_,loc2))
                       -> (let msg = "Alloc. Problems in "
                       ^ (toStringLoc loc)
                       ^ " or in " ^ (toStringLoc loc2) in
                       raise(SystemErrorE("expSem: "^msg)))
                   | (SystemErrorM _)
                       -> (let msg = "Integrity Problems in
                       ↪  Store State" in
                       raise(SystemErrorE("expSem: " ^ msg)))
           ) in
           let valS = List.map f locL in
           (Arr(t, num), Aval valS, (rho, mu)))
       | DArray(Mut(Arr(Mut t, num)), (Loc n))
           -> raise(TypeErrorE("E9.1: expSem - " ^
           ↪  (toStringExp exp)))
```

- Nel caso in cui `den` sia di tipo array prova ad ottenere i valori dell'array dallo store.
- Nel caso ci siano problemi di allocazione restituisce errore
- Altrimenti restituisce un valore array
- Se `t` $\notin$ `Simple` allora dà l'errore `E9.1`

Implementazione nel nuovo `l-value` per `Val I`

Andiamo ora a modificare la funzione `dexpSem` per aggiungere il nuovo valore denotabile calcolabile

Implementazione nel nuovo `l-value` per `Val I`

Andiamo ora a modificare la funzione `dexpSem` per aggiungere il nuovo valore denotabile calcolabile

```
1  | Val ide -> (
2     try (let den = getEnv rho ide in
3     ...
4     | DArray(Mut(Arr(Mut t, num)), loct)
5         when isSimple t
6         -> (Mut(Arr(Mut t, num)), loct, (rho, mu))
7     | DArray(Mut(Arr(Mut t, num)), loct)
8         -> raise(TypeErrorE("E9.2: dexpSem - " ^
          ↪  (toStringExp dexp)))
```

- Nel caso i cui `t` $\in$ `Simple` restituisce il valore array e la locazione relativa, altrimenti dà l'errore `E9.2`

Implementazione nel nuovo `l-value` per `Val I`

Andiamo ora a modificare la funzione `dexpSem` per aggiungere il nuovo valore denotabile calcolabile

```
| Val ide -> (
   try (let den = getEnv rho ide in
   ...
   | DArray(Mut(Arr(Mut t, num)), loct)
       when isSimple t
       -> (Mut(Arr(Mut t, num)), loct, (rho, mu))
   | DArray(Mut(Arr(Mut t, num)), loct)
       -> raise(TypeErrorE("E9.2: dexpSem - " ^
       ↪  (toStringExp dexp)))
```

- Nel caso i cui `t` $\in$ `Simple` restituisce il valore array e la locazione relativa, altrimenti dà l'errore `E9.2`

Possiamo quindi aggiungere al caso `Upd(el, er)` di `expSem` la possibilità di aggiornare degli array

Upd(el, er) per array

```
1  match (dexpSem el sgr) with
2   | (Mut(Arr(Mut t, num)), (Loc n), (rhol, mul))
3      -> (match tr with
4          | Arr(te, k)
5            when (ysame te t) && (k = num)
6            -> (match vr with
7                | Aval valS ->
8                    (try (let ll = locRange n (n+k-1) in
9                    let muF = List.fold_left2 upd mul ll
                     ↪  (List.map eTOm valS) in
10                   (Arr(t, k), Aval valS, (rhol, muF))
11                   ) with
12                       | (IllegalAddress _)
13                           -> (let msg = "expSem: wrong
                             ↪  Mutable/location found " in
14                           raise(SystemErrorE(msg ^
                             ↪  (toStringExp exp)))))
15               | _ -> (let msg = "expSem: wrong value in " in
16                   raise(SystemErrorE(msg ^ (toStringExp
                     ↪  exp)))))
17         | Arr(te, k)
18           when (ysame te t)
19               -> (let msg = "E19.2: expSem - " in
20               raise(SystemErrorE(msg ^ (toStringExp exp))))
21         | Arr(te, k)
22           -> (let msg = "E19.1 expSem - " in
23           raise(SystemErrorE(msg ^ (toStringExp exp))))
24         | _ -> (let msg = "E19.3 expSem - " in
25           raise(SystemErrorE(msg ^ (toStringExp exp)))))
```

- Nel caso in cui `tr` sia un array di tipo `t` e dimensione `num` proviamo ad aggiornare i valori nelle giuste locazioni della memoria

Upd(el, er) per array

```
1  match (dexpSem el sgr) with
2   | (Mut(Arr(Mut t, num)), (Loc n), (rhol, mul))
3      -> (match tr with
4          | Arr(te, k)
5            when (ysame te t) && (k = num)
6            -> (match vr with
7                | Aval valS ->
8                    (try (let ll = locRange n (n+k-1) in
9                    let muF = List.fold_left2 upd mul ll
                     ↪  (List.map eTOm valS) in
10                   (Arr(t, k), Aval valS, (rhol, muF))
11                   ) with
12                       | (IllegalAddress _)
13                           -> (let msg = "expSem: wrong
                             ↪  Mutable/location found " in
14                           raise(SystemErrorE(msg ^
                             ↪  (toStringExp exp)))))
15               | _ -> (let msg = "expSem: wrong value in " in
16                   raise(SystemErrorE(msg ^ (toStringExp
                     ↪  exp)))))
17         | Arr(te, k)
18           when (ysame te t)
19               -> (let msg = "E19.2: expSem - " in
20               raise(SystemErrorE(msg ^ (toStringExp exp))))
21         | Arr(te, k)
22           -> (let msg = "E19.1 expSem - " in
23           raise(SystemErrorE(msg ^ (toStringExp exp))))
24         | _ -> (let msg = "E19.3 expSem - " in
25           raise(SystemErrorE(msg ^ (toStringExp exp)))))
```

- Nel caso in cui `tr` sia un array di tipo `t` e dimensione `num` proviamo ad aggiornare i valori nelle giuste locazioni della memoria
- Se il tipo `tr` di `er` è diverso da quello cercato restituiamo l'errore relativo al caso in cui siamo

Alcuni esempi di programma

Riportiamo ora alcuni esempi di programmi che mettono in evidenza le funzionalità del nuovo costrutto e i tipi di errori che si possono ottenere

```
let d1 = Array(Int, "A", 3) in
let exp = Upd(Val "A", ArrayE [N 1; N 2; N 3]) in
let c = Cmd exp in
let p = Prog("myprog", Seq [StmD d1; StmC c]) in
let s = toStringProg p in
let _ = printf "\n%s\n" s in
run p;;
```

- Qui verifichiamo la semplice esecuzione della nuova funzionalità di `Upd`

```
Program myprog{
    int array  A[3];
    A = {1, 2, 3};
}

========== Traccia del Programma mypro ==========
=================================================
Stack: [A/(int[3],L0)]
Store: [L0<-1,L1<-2,L2<-3]
=================================================
================= Traccia: Fine =================
```

- Osserviamo che l'array `A` è stato modificato correttamente

Alcuni esempi di programma

```
let d1 = Array(Bool, "A", 3) in
let exp1 = Upd(Val "A", ArrayE [B True; B False; B True]) in
let d2 = Array(Bool, "B", 3) in
let exp2 = Upd(Val "B", exp1) in
let c = Cmd exp2 in
let p = Prog("myprog", Seq [StmD d1; StmD d2; StmC c]) in
let s = toStringProg p in
let _ = printf "\n%s\n" s in
run p;;
```

- Complichiamo leggermente il programma per verificare che Upd restituisca il valore corretto

```
Program mypro{
   bool array  A[3];
   bool array  B[3];
   B = (A = {true, false, true});
}

========== Traccia del Programma mypro ==========
=================================================
Stack: [B/(bool[3],L3); A/(bool[3],L0)]
Store: [L0<-true,L1<-false,L2<-true,L3<-true,
 L4<-false,L5<-true]
=================================================
================= Traccia: Fine =================
```

- Anche l'array B è stato modificato correttamente

Alcuni esempi di programma

```
let d1 = Array(Int, "A", 3) in
let exp = Upd(Val "A", ArrayE [B True; B False; B
↪  False]) in
let c = Cmd exp in
let p = Prog("mypro", Seq [StmD d1; StmC c]) in
let s = toStringProg p in
let _ = printf "\n%s\n" s in
run p;;
```

- Proviamo a verificare che il controllo dei tipi funzioni correttamente

```
Program mypro{
    int array  A[3];
    A = {true, false, false};
}
Exception: SystemErrorE "E19.1 expSem - (A = {true,
↪  false, false})".
```

- Ci viene correttamente restituito l'errore E19.1, cioè quello relativo a array di tipo diverso calcolati da `el` e `er`

Alcuni esempi di programma

```
let d1 = Array(Int, "A", 3) in
let exp = Upd(Val "A", ArrayE [N 1; B False]) in
let c = Cmd exp in
let p = Prog("mypro", Seq [StmD d1; StmC c]) in
let s = toStringProg p in
let _ = printf "\n%s\n" s in
run p;;
run p;;
```

- Infine verifichiamo che anche il controllo dei tipi relativo al costrutto `ArrayE` funzioni correttamente

```
Program mypro{
   int array  A[3];
   A = {1, false};
}
Exception:
SystemErrorE
 "E33 expSem: Expression 'false' has type bool but
 ↪  type int is required".
```

- Otteniamo l'errore `E33` e l'interprete ci avverte qual è l'espressione con tipo sbagliato.

Grazie per l'attenzione