Linguaggio Small21 - v.04.05.21

- **Questo documento** incorpora i progressi sulla definizione di Small21 nel corso della sua definizione e sviluppo

- **Aggiornare** Lo aggiorneremo per usarlo come documento di riferimento per le modifiche che effettueremo al termine del suo sviluppo

- **Errata Corrige** Recherà indicazione delle modifiche e correzioni (tecniche) sull'ultima versione.

  Correzioni e le modifiche fatte sul documento distribuito a conclusione dell'attività del giorno 03/05/21

  - BlockC usato erroneamente al posto di BlockS;
  - Regole Y3,Y4 e E7-E9 corrette - uso errato di struttura array
  - Regola Y7 aggiunta - omessa regola per `[emptyD]`
  - Signature funzione Semantica Y in slide9 - errata e' ora corretta
  - sintassi DCL in slide 10 - errata e' ora corretta.
  - Regola `[D1]` in slide 10 - errato uso di ρ corretto con stack Delta
  - Regola `[D3]` in slide 10 - conclusione errata e' ora corretta.

Sistema di Tipi Y di Small21

- **Sintassi Concreta**
  Type ::= Simple | void | Simple [Num]
  Simple ::= int | bool

- **Sintassi Astratta ed Espressioni di Tipo**
  Type ::= [int] | [bool] | [lab] | [void] | [arr] Type Num
  | [mut] Type | [abs] Type TypeSeq | [terr]
  | [unit] | [type]

- **Regole: Una Regola di $\mathtt{Y}_{\mathtt{DCL}^{\mathtt{Small21}}}$**

$$\frac{\langle \mathtt{e}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}', \mathtt{Y}_\rho) \qquad \mathtt{t}' \neq [\mathtt{terr}] \qquad \mathtt{t}' = \mathtt{t}}{\langle [\mathtt{const}]\ \mathtt{t}\ \mathtt{I}\ \mathtt{e}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} (\mathtt{Void}, [\mathtt{I}/\mathtt{t}] \circ \mathtt{Y}_\rho)}$$

Sono inferenze di un **Sistema Deduttivo** che associa ad ogni struttura del programma, s, con tipi degli identificatori definiti da $\mathtt{Y}_\rho$, una coppia $(\mathtt{t}', \mathtt{Y}'_\rho)$. Sopra, nella premessa della regola, nella formula:

$$\langle \mathtt{e}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}', \mathtt{Y}_\rho)$$

vediamo ciò. In particolare, la regola è applicabile quando il sistema ha dedotto che l'espressione e, dove i suoi identificatori liberi hanno tipi tutti legati in $\mathtt{Y}_\rho$, abbia tipo $\mathtt{t}'$ (i.e. $\mathtt{t}' = \mathtt{t}$). Ovvero:

$$\mathtt{Y}_\rho \vdash \mathtt{e} : \mathtt{t}.$$

Nella conclusione della regola sopra, la struttura di programma s è:

$$[\mathtt{const}]\ \mathtt{t}\ \mathtt{I}\ \mathtt{e}.$$

Quando la regola è applicabile allora il sistema è in grado di inferire

$$\langle \mathtt{s}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} (\mathtt{Void}, [\mathtt{I}/\mathtt{t}] \circ \mathtt{Y}_\rho)$$

ovvero il sistema deduce che la struttura s è/introduce una corretta dichiarazione che estende in modo consistente $\mathtt{Y}_\rho$ con il binding $[\mathtt{I}/\mathtt{t}]$. Ovvero:

$$\mathtt{Y}_\rho, \mathtt{I} : \mathtt{t} \vdash \mathtt{s} : \mathtt{Void}.$$

Sistema di Tipi: Regole, Strutture e Proprietà di Y

$$\frac{\langle \mathtt{e}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}', \mathtt{Y}_\rho) \qquad \mathtt{t}' \neq [\mathtt{terr}] \qquad \mathtt{t}' = \mathtt{t}}{\langle [\mathtt{const}]\ \mathtt{t}\ \mathtt{I}\ \mathtt{e}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} (\mathtt{Void}, [\mathtt{I}/\mathtt{t}] \circ \mathtt{Y}_\rho)}$$

- **Regole.** Sono inferenze di un sistema deduttivo che associano ad ogni struttura del programma, s, con tipi degli identificatori definiti da $\mathtt{Y}_\rho$, una coppia $(\mathtt{t}', \mathtt{Y}'_\rho)$.

  Sopra, in premessa $\langle \mathtt{e}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}', \mathtt{Y}_\rho)$ vediamo ciò. In particolare, il sistema ha dedotto che l'espressione e, dove i suoi identificatori liberi siano tutti legati in $\mathtt{Y}_\rho$, abbia tipo $\mathtt{t}' = \mathtt{t}$. Ovvero: $\mathtt{Y}_\rho \vdash \mathtt{e} : \mathtt{t}$.
  Sopra, nella conclusione, la struttura di programma s è: $[\mathtt{const}]\ \mathtt{t}\ \mathtt{I}\ \mathtt{e}$. Quando il sistema è in grado di inferire $\langle \mathtt{s}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} (\mathtt{Void}, [\mathtt{I}/\mathtt{t}] \circ \mathtt{Y}_\rho)$ allora il sistema ha dedotto che la struttura s introduce una corretta dichiarazione che estende in modo consistente $\mathtt{Y}_\rho$ con il binding $[\mathtt{I}/\mathtt{t}]$. Ovvero: $\mathtt{Y}_\rho, \mathtt{I} : \mathtt{t} \vdash \mathtt{s} : \mathtt{Void}$.

- **Identificatori.** Tutti gli identificatori introdotti nel programma hanno un tipo t (definito da un'espressione di tipo del linguaggio). Questo tipo è associato all'identificatore nell'ambiente $\mathtt{Y}_\rho$ come il binding dell'identificatore in $\mathtt{Y}_\rho$.

- **Struttura $\mathtt{Y}_\rho$.** È una struttura di ambiente con le stesse operazioni viste per gli ambienti di denotazioni. In un Linguaggio a blocchi, l'applicazione di Y in differenti blocchi di un programma richiede che $\mathtt{Y}_\rho$ sia uno stack $> \rho :: \mathtt{Y}'_\rho]$ di ambienti senza ripetizioni con ρ ambiente del corrente blocco e $\mathtt{Y}'_\rho$ lo stack di ambienti (uno per ogni blocco attraversato fino al corrente) creati staticamente dal sistema Y.

- **Completezza** di Y richiede che il sistema sia dotato di un insieme di regole tali che: L'applicazione di Y ad ogni programma (sintatticamente corretto e legale) P, nell'ambiente $\mathtt{Y}_\mathtt{L}$, contenente i binding per tutti e soli gli identificatori di primitive del Linguaggio, sia in grado di fornire una e una sola derivazione di tipi ovvero un albero dove ogni nodo è la conclusione di una regola e i figli sono le premessa di tale regola sotto una stessa istanza di simboli della regola con strutture di P. L'albero così ottenuto ha:

  - radice: $\langle \mathtt{P}, \mathtt{Y}_\mathtt{L} \rangle \rightarrow_{\mathtt{Y}} (\mathtt{Void}, \mathtt{Y}_\mathtt{L})$. Ovvero: $\mathtt{Y}_\mathtt{L} \vdash \mathtt{P} : \mathtt{Void}$
  - discendenti: Per ogni termine s contenuto in P, $\langle \mathtt{s}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} (\mathtt{t}', \mathtt{Y}'_\rho)$. Ovvero: $\mathtt{Y}_\mathtt{L}, \mathtt{Y}_\rho \vdash \mathtt{s} : \mathtt{t}$

- **Ben tipato** in Y, è ogni termine (costrutto) s di ogni programma P (sintatticamente corretto e legale) che abbia una derivazione di tipi con radice come sopra e con discendenti come sopra e tali che $\mathtt{t} \neq [\mathtt{terr}]$.

Sistema di Tipi: Regole, Strutture e Proprietà di Y - 2

$$\frac{\Gamma_1 \quad \ldots \quad \Gamma_n}{\langle s, Y_\rho \rangle \rightarrow_Y (t, Y'_\rho)}$$

- **Stato di Stuck** in un Linguaggio con un sistema di tipi Y, è ogni stato inconsistente rispetto a Y. Definiamo inconsistente ogni stato che contiene:
  - **binding** tra un identificatore di tipo `t1` e un valore (denotabile o memorizzabile) di tipo `t2`, con `t1` $\neq$ `t2`;
  - **valore atteso**, in qualche componente di un qualche AR (inclusi buffer di I/O), di tipo `t1` e un valore calcolato di tipo `t2`, con `t1` $\neq$ `t2`

  Ogni Linguaggio tipato ha una propria struttura di stato che include sempre l'ambiente dei bindings degli identificatori (introdotti nel programma), ρ, lo stack di AR e, secondo i casi, una (o più per heap, per variabili statiche) memoria μ per valori modificabili e ogni altra struttura (buffer di I/O, tabella delle classi, tabella dei moduli,..) che debba far parte dello stato. Il sistema di tipi provvede esplicitamente a definire la consistenza per la struttura di stato dello specifico Linguaggio tipato.
- **Stato NonStuck** in un Linguaggio con un sistema di tipi Y, è ogni stato che soddisfa la consistenza richiesta da Y.
- **Correttezza (Safety)** di Y richiede che esso sia definito in accordo alla semantica del Linguaggio garantendo le due proprietà sotto:
  - **Progress**: Ogni termine ben tipato `s` (avente $Y_\rho \vdash s : t$) o è esso stesso un valore di tipo `t`[1] che occorre in uno stato NonStuck, oppure in accordo alla semantica la computazione contiene un successivo stato;
  - **Preservation**: Se `s` nel successivo stato è stato rimpiazzato con s' allora anche s' è ben tipato ed anche per esso vale $Y_\rho \vdash s' : t$
- **Theorem Completezza+Correttezza (locale)** Ogni Linguaggio tipato con sistema di tipi Completo e Safe è tale che tutti i suoi programmi ben tipati hanno computazioni prive di stati di Stuck.

[1] Ad ogni tipo è associato un insieme di valori non vuoto. Il tipo Void contiene il solo valore: `void`

Sintassi Concreta: CFG per Small21 (;-terminatore)

$$
\begin{aligned}
&\texttt{Type} \rightarrow \texttt{Simple} \mid \texttt{void} \mid \texttt{Simple [Num]}\\
&\texttt{Simple} \rightarrow \texttt{int} \mid \texttt{bool}\\
\\
&\texttt{Dcl} \rightarrow \texttt{const Simple ide = Exp;} \mid \texttt{Simple ide = Exp;}\\
&\qquad\qquad \mid \texttt{Type ide;} \mid \texttt{Type ide (Fp) Block}\\
&\texttt{Dcls} \rightarrow \texttt{Dcl Dcls} \mid \epsilon\\
\\
&\texttt{Exp} \rightarrow \texttt{ExpB == ExpB} \mid \texttt{ExpB}\\
&\texttt{ExpB} \rightarrow \texttt{ExpB or ExpR} \mid \texttt{truth} \mid \texttt{ExpR}\\
&\texttt{ExpR} \rightarrow \texttt{ExpA} > \texttt{ExpA} \mid \texttt{ExpA} < \texttt{ExpA} \mid \texttt{ExpA}\\
&\texttt{ExpA} \rightarrow \texttt{ExpA} + \texttt{ExpT} \mid \texttt{ExpT}\\
&\texttt{ExpT} \rightarrow \texttt{num} \mid \texttt{(Exp)} \mid \texttt{ide (AP)} \mid \texttt{DExp}\\
&\texttt{DExp} \rightarrow \texttt{ide} \mid \texttt{ide [ExpA]}\\
\\
&\texttt{Cmd} \rightarrow \texttt{lab : Stm} \mid \texttt{Stm}\\
&\texttt{Cmds} \rightarrow \texttt{Cmd Cmds} \mid \texttt{Cmd}\\
&\texttt{Stm} \rightarrow \texttt{if (Exp) Stm} \mid \texttt{OtherStm}\\
&\texttt{OtherStm} \rightarrow \texttt{if (Exp) OtherStm else Stm} \mid \texttt{NonConditionalStm}\\
&\texttt{NonConditionalStm} \rightarrow \texttt{DExp = Exp;} \mid \texttt{goto lab;}\\
&\qquad\qquad \mid \texttt{return Exp;} \mid \texttt{;} \mid \texttt{ide (Ap);} \mid \texttt{Block}\\
&\texttt{Stms} \rightarrow \texttt{Stm} \mid \texttt{Stm Stms}\\
&\texttt{Block} \rightarrow \texttt{\{Dcls Stms\}}\\
\\
&\texttt{Prog} \rightarrow \texttt{Program ide \{Dcls Cmds\}}
\end{aligned}
$$

Sintassi Concreta: CFG per Small21 (;-terminatore) - 2

`FP` $\rightarrow$ `Fp FPs` $\mid \epsilon$	(per estensioni successive)
`FPs` $\rightarrow$ `,` `Fp FPs` $\mid \epsilon$	(per estensioni successive)
`Fp` $\rightarrow$ `PPF Type ide`	
`PPF` $\rightarrow \epsilon \mid$ `ref`	

`AP` $\rightarrow$ `Ap APs` $\mid \epsilon$	(per estensioni successive)
`APs` $\rightarrow$ `,` `Ap APs` $\mid \epsilon$	(per estensioni successive)
`Ap` $\rightarrow$ `Exp`	

Vincoli Contestuali

- Identificatori usati devono essere dichiarati.
- Vietata la molteplice dichiarazione di identificatori, anche con tipi diversi, in uno stesso blocco.
- Vietata dichiarazione procedure o funzioni overloaded.
- Corrispondenza in numero e tipo tra formali ed attuali nell'uso di procedure o funzioni
- Avvertenza: Uso di goto in inner block sconsigliato.

Sintassi Astratta di Small21

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

Dcl ::= [var] Type Ide Exp | [const] Type Ide Exp
      | [array] Type Ide | Dcl [seqD] Dcl
      | [pcd] Type Ide FPars BlockS | [emptyD]

Exp ::= DExp
      | [num] Num | Exp [+] Exp
      | Exp [==] Exp
      | [true] | [false] | Exp [or] Exp
      | Exp [>] Exp | Exp [<] Exp
      | [apply] Ide APars | [emptyE]
DExp ::= [val] Ide | DExp [↑] Exp | Ide [↑1] Exp

Cmd ::= Lab [:] Stm | Stm

Stm ::= DExp [=] Exp | [ifE] Exp Stm Stm | [goto] Lab
      | Stm [seqS] Stm | BlockS | [emptyS]
      | [call] Ide APars | [return] Exp

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

Sintassi Astratta di Small21 - 2

```
TypeSeq ::= Type [::] TypeSeq | Type

Num                          -- token del lessico degli interi
Ide                          -- token del lessico degli identificatori
Lab                          -- token del lessico delle labels di cmd

FPars ::= [fp] PPF Type Ide | [emptyFP]

PPF ::= [value] | [ref]      -- Parameter passing Form

BlockE ::= [blockE] Dcl Exp
BlockS ::= [blockS] Dcl Stm    -- in-line block
Block ::= [block] Dcl CmdSeq
CmdSeq ::= [seqC] Cmd CmdSeq | EC

APars ::= [ap] Exp | [emptyAP]
```

Osservazioni

- [mut]: Costruttore per Tipi di valori mutable. Ad es.: [mut][int], [mut]([arr]([int],30)), [arr]([mut][bool],15).
- ↑: Costruttore di termine di accesso a componente array. Ad es.: `x` ↑ 3 corrisponde, in C, all'espressione `x[3]` quando `x` sia una variabile di tipo [mut]([arr]([mut][int],30)).
- SeqTypes: Il tipo più a sinistra indica il tipo immagine i successivi, quando presenti, i tipi degli argomenti della astrazione funzionale, procedurale o di operatore primitivo.
- FPars e APars: Parametri per astrazioni al momento con al più un solo parametro e solo 2 forme di trasmissione.

Sistema Y: Regole per DCL

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

$$[\text{Y2}]\ \frac{\langle \texttt{e}, \texttt{Y}_\rho\rangle \rightarrow_{\texttt{Y}} (\texttt{t}', \texttt{Y}_\rho) \qquad \texttt{t} \in \texttt{Simple} \qquad (\texttt{t}' = \texttt{t}\ \texttt{or}\ \texttt{t}' = [\texttt{unit}]) \qquad \texttt{Y}_\rho|_0(\texttt{I}) = \bot}{\langle [\texttt{var}]\ \texttt{t}\ \texttt{I}\ \texttt{e}, \texttt{Y}_\rho\rangle \rightarrow_{\texttt{Y}} ([\texttt{void}], [\texttt{I}/[\texttt{mut}]\ \texttt{t}] \otimes \texttt{Y}_\rho)}$$

Gestione Errori di Tipo:

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

$$[\text{E3}]\ \frac{\texttt{Y}_\rho|_0(\texttt{I}) \neq \bot}{\langle [\texttt{const}]\ \texttt{t}\ \texttt{I}\ \texttt{e}, \texttt{Y}_\rho\rangle \rightarrow_{\texttt{Y}} ([\texttt{terr}], \texttt{Y}_\rho)}$$

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

$$[\text{E6}]\ \frac{\texttt{Y}_\rho|_0(\texttt{I}) \neq \bot}{\langle [\texttt{var}]\ \texttt{t}\ \texttt{I}\ \texttt{e}, \texttt{Y}_\rho\rangle \rightarrow_{\texttt{Y}} ([\texttt{terr}], \texttt{Y}_\rho)}$$

Notazione: Simboli e Strutture

- ${>}\rho :: \texttt{Y}_\rho]$, stack non vuoto di ambienti di legami `[I/t]`, avente ρ come top e $\texttt{Y}_\rho$ come stack residuo.
- $\texttt{Y}_\rho|_{\texttt{n}}$, n-esimo (`n>=0`) ambiente precedente il top dello stack $\texttt{Y}_\rho$.
- $[\texttt{I}/\texttt{t}] \otimes \texttt{Y}_\rho$, aggiunta del binding `[I/t]` nel top dello stack $\texttt{Y}_\rho$.
- `N`, Intero; • `I` (anche con apici), Identificatore; • `t` (anche con apici), Tipo; • `e`, espressione.
- $\texttt{Simple} = \{[\texttt{int}], [\texttt{bool}]\}$

Sistema Y: Regole per DCL - 2

$$[\text{Y3})\ \frac{\begin{array}{c}\texttt{t} = [\texttt{arr}]\,\texttt{t}'\,\texttt{N} \qquad \texttt{N} > 0 \\ \texttt{t} \in \texttt{Simple} \qquad \texttt{Y}_\rho|_0(\texttt{I}) = \bot\end{array}}{\langle [\texttt{array}]\,\texttt{t}\,\texttt{I}, \texttt{Y}_\rho\rangle \rightarrow_\texttt{Y} ([\texttt{void}], [\texttt{I}/[\texttt{arr}]([\texttt{mut}]\ \texttt{t})\ \texttt{N}] \otimes \texttt{Y}_\rho)}\ **$$

$$[\text{Y4})\ \frac{\begin{array}{c}\langle \texttt{d1}, \texttt{Y}_\rho\rangle \rightarrow_\texttt{Y} ([\texttt{void}], \texttt{Y1}_\rho) \\ \langle \texttt{d2}, \texttt{Y1}_\rho\rangle \rightarrow_\texttt{Y} ([\texttt{void}], \texttt{Y2}_\rho)\end{array}}{\langle \texttt{d1}\ [\texttt{seqD}]\ \texttt{d2}, \texttt{Y}_\rho\rangle \rightarrow_\texttt{Y} ([\texttt{void}], \texttt{Y2}_\rho)}\ **$$

Gestione Errori di Tipo:

$$[\text{E7})\ \frac{\texttt{t} = [\texttt{arr}]\,\texttt{t}'\,\texttt{N} \quad \texttt{t}' \notin \texttt{Simple}}{\langle [\texttt{array}]\,\texttt{t}\,\texttt{I}, \texttt{Y}_\rho\rangle \rightarrow_\texttt{Y} ([\texttt{terr}], \texttt{Y}_\rho)} \qquad [\text{E8})\ \frac{\texttt{t} = [\texttt{arr}]\,\texttt{t}'\,\texttt{N} \quad \texttt{N} <= 0}{\langle [\texttt{array}]\,\texttt{t}\,\texttt{I}, \texttt{Y}_\rho\rangle \rightarrow_\texttt{Y} ([\texttt{terr}], \texttt{Y}_\rho)}$$

$$[\text{E9})\ \frac{\texttt{Y}_\rho|_0(\texttt{I}) \neq \bot}{\langle [\texttt{array}]\,\texttt{t}\,\texttt{I}, \texttt{Y}_\rho\rangle \rightarrow_\texttt{Y} ([\texttt{terr}], \texttt{Y}_\rho)}$$

$$[\text{E10})\ \frac{\langle \texttt{d1}, \texttt{Y}_\rho\rangle \rightarrow_\texttt{Y} ([\texttt{terr}], \texttt{Y1}_\rho)}{\langle \texttt{d1}\ [\texttt{seqD}]\ \texttt{d2}, \texttt{Y}_\rho\rangle \rightarrow_\texttt{Y} ([\texttt{terr}], \texttt{Y}_\rho)} \qquad [\text{E11})\ \frac{\begin{array}{c}\langle \texttt{d1}, \texttt{Y}_\rho\rangle \rightarrow_\texttt{Y} ([\texttt{void}], \texttt{Y1}_\rho) \\ \langle \texttt{d2}, \texttt{Y1}_\rho\rangle \rightarrow_\texttt{Y} ([\texttt{terr}], \texttt{Y2}_\rho)\end{array}}{\langle \texttt{d1}\ [\texttt{seqD}]\ \texttt{d2}, \texttt{Y}_\rho\rangle \rightarrow_\texttt{Y} ([\texttt{terr}], \texttt{Y1}_\rho)}$$

Notazione: Simboli e Strutture

- [], ambiente vuoto
- $>\rho :: \texttt{Y}_\rho]$, stack non vuoto di ambienti di legami [I/t], avente ρ come top e $\texttt{Y}_\rho$ come stack residuo.
- $\texttt{Y}_\rho|_\texttt{n}$, n-esimo (n>=0) ambiente precedente il top dello stack $\texttt{Y}_\rho$.
- $[\texttt{I}/\texttt{t}] \otimes \texttt{Y}_\rho$, aggiunta del binding [I/t] nel top dello stack $\texttt{Y}_\rho$.

Sistema Y: Regole per DCL - 3

$$[\text{Y5}]\ \frac{\mathtt{t} \in \mathtt{Simple} \cup \{[\mathtt{void}]\} \qquad \mathtt{F} = [\mathtt{fp}]\,\mathtt{p}\,\mathtt{t}'\,\mathtt{I}' \qquad \mathtt{t}' \in \mathtt{Simple} \qquad \mathtt{Y}_\rho|_0(\mathtt{I}) = \bot \qquad > [\mathtt{I}'/\mathtt{t}'] \circ [] :: \mathtt{Y}_\rho] = \mathtt{Y}'_\rho \qquad \langle \mathtt{Bc}, \mathtt{Y}'_\rho \rangle \rightarrow_{\mathtt{Y}} ([\mathtt{void}], \mathtt{Y}''_\rho)}{\langle [\mathtt{pcd}]\,\mathtt{t}\,\mathtt{I}\,\mathtt{F}\,\mathtt{Bc}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} ([\mathtt{void}], [\mathtt{I}/[\mathtt{abs}]\,\mathtt{t}\,[::]\,\mathtt{t}'] \otimes \mathtt{Y}_\rho)}\ **$$

$$[\text{Y6}]\ \frac{\mathtt{t} \in \mathtt{Simple} \cup \{[\mathtt{void}]\} \qquad \mathtt{F} = [\mathtt{emptyFP}] \qquad \mathtt{Y}_\rho|_0(\mathtt{I}) = \bot \qquad > [] :: \mathtt{Y}_\rho] = \mathtt{Y}'_\rho \qquad \langle \mathtt{Bc}, \mathtt{Y}'_\rho \rangle \rightarrow_{\mathtt{Y}} ([\mathtt{void}], \mathtt{Y}''_\rho)}{\langle [\mathtt{pcd}]\,\mathtt{t}\,\mathtt{I}\,\mathtt{F}\,\mathtt{Bc}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} ([\mathtt{void}], [\mathtt{I}/[\mathtt{abs}]\,\mathtt{t}] \otimes \mathtt{Y}_\rho)}\ **$$

$$[\text{Y7}]\ \frac{}{\langle [\mathtt{emptyD}], \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} ([\mathtt{void}], \mathtt{Y}_\rho)}$$

Gestione Errori di Tipo:

$$[\text{E12}]\ \frac{\mathtt{t} \notin \mathtt{Simple} \cup \{[\mathtt{void}]\}}{\langle [\mathtt{pcd}]\,\mathtt{t}\,\mathtt{I}\,\mathtt{F}\,\mathtt{Bc}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}_\rho)}$$

$$[\text{E13}]\ \frac{\mathtt{F} = [\mathtt{fp}]\,\mathtt{p}\,\mathtt{t}'\,\mathtt{I}' \qquad \mathtt{t}' \notin \mathtt{Simple}}{\langle [\mathtt{pcd}]\,\mathtt{t}\,\mathtt{I}\,\mathtt{F}\,\mathtt{Bc}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}_\rho)}$$

$$[\text{E14}]\ \frac{\mathtt{Y}_\rho|_0(\mathtt{I}) \neq \bot}{\langle [\mathtt{pcd}]\,\mathtt{t}\,\mathtt{I}\,\mathtt{F}\,\mathtt{Bc}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}_\rho)}$$

$$[\text{E15}]\ \frac{\ldots\ldots \quad \langle \mathtt{Bc}, > \rho :: \mathtt{Y}_\rho] \rangle \rightarrow_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}'_\rho)}{\langle [\mathtt{pcd}]\,\mathtt{t}\,\mathtt{I}\,\mathtt{F}\,\mathtt{Bc}, \mathtt{Y}_\rho \rangle \rightarrow_{\mathtt{Y}} ([\mathtt{terr}], \mathtt{Y}_\rho)}$$

Notazione: Simboli e Strutture

- [], ambiente vuoto
- $> \rho :: \mathtt{Y}_\rho]$, stack non vuoto di ambienti di legami [I/t], avente ρ come top e $\mathtt{Y}_\rho$ come stack residuo.
- $\mathtt{Y}_\rho|_{\mathtt{n}}$, n-esimo (n>=0) ambiente precedente il top dello stack $\mathtt{Y}_\rho$.
- $[\mathtt{I}/\mathtt{t}] \otimes \mathtt{Y}_\rho$, aggiunta del binding [I/t] nel top dello stack $\mathtt{Y}_\rho$.
- $[\mathtt{abs}]\,\mathtt{t}\,[::]\,\mathtt{t}'$, $[\mathtt{abs}]\,\mathtt{t}$, Tipi per procedura e funzione;
- F, parametro formale;
- Bc blocco di statements;

AM21: Il Modello Astratto - Le Operazioni

- **Relazione tra Stato, Semantica ed Operazioni**.
  - Lo Stato esprime tutte e sole le configurazioni della AM21
  - Computazione di un programma Small21 P a partire da uno stato S è la Sequenza di Stati che la Semantica associa a P valutatonello stato iniziale S.
  - La Semantica è espressa nelle sole operazioni di AM21 (che definiremo a partire da oggi)
- **Funzioni Semantiche e Sistema di Inferenza associato**.
  - **Y** : $\texttt{AST} \rightarrow \texttt{Y}_\rho \rightarrow \texttt{Type} * \texttt{Y}_\rho$
    $(\forall \texttt{a}, \texttt{Y}_\rho)$ $\mathbf{Y}(\texttt{a}, \texttt{Y}_\rho) = (\texttt{t}, \texttt{Y}'_\rho)$ **iff** $\langle \texttt{a}, \texttt{Y}_\rho \rangle \rightarrow_{\texttt{Y}} (\texttt{t}, \texttt{Y}'_\rho) \in$ **SistemaY**
  - **dclSem** : $\texttt{Dcl} \rightarrow \texttt{State} \rightarrow \texttt{Type} * \texttt{State}$
    $(\forall \texttt{d}, \sigma)$ $\mathbf{dclSem}(\texttt{d}, \sigma) = (\texttt{t}, \sigma')$ **iff** $\langle \texttt{d}, \sigma \rangle \rightarrow (\texttt{t}, \sigma') \in \mathbf{Sem_{DCL}}$
  - **expSem** : $\texttt{exp} \rightarrow \texttt{State} \rightarrow \texttt{Type} * \texttt{Val} * \texttt{State}$
    $(\forall \texttt{e}, \sigma)$ $\mathbf{expSem}(\texttt{e}, \sigma) = (\texttt{t}, \texttt{v}, \sigma')$ **iff** $\langle \texttt{e}, \sigma \rangle \rightarrow \lfloor \texttt{t}, \texttt{v}, \sigma' \rfloor \in \mathbf{Sem_{EXP}}$
  - **dexpSem** : $\texttt{dexp} \rightarrow \texttt{State} \rightarrow \texttt{Type} * \texttt{DVal} * \texttt{State}$
    $(\forall \texttt{e}_\texttt{d}, \sigma)$ $\mathbf{expSem}(\texttt{e}_\texttt{d}, \sigma) = (\texttt{t}_\texttt{d}, \texttt{v}_\texttt{d}, \sigma')$ **iff** $\langle \texttt{e}_\texttt{d}, \sigma \rangle \rightarrow \lfloor \texttt{t}_\texttt{d}, \texttt{v}_\texttt{d}, \sigma' \rfloor \in \mathbf{Sem_{DEXP}}$
  - **cmdSem** : $\texttt{cmd} \rightarrow \texttt{State} \rightarrow \texttt{Type} * \texttt{State}$
    $(\forall \texttt{c}, \sigma)$ $\mathbf{cmdSem}(\texttt{c}, \sigma) = (\texttt{t}, \sigma')$ **iff** $\langle \texttt{c}, \sigma \rangle \rightarrow (\texttt{t}, \sigma') \in \mathbf{Sem_{CMD}}$
- **Stato: Strutture, Operazioni e Notazione**
  - (Δ, μ) : Stato con Stack di AR, Δ, e Store, μ.
  - Costruttori: Tutte le strutture del Modello (Frame, AR, ...) hanno costruttori che forniscono Termini (Algebra libera) con cui esprimere tutte e sole le strutture di AM21.
  - Operazioni: Tutte le strutture del Modello hanno operazioni di Accesso, Selezione, Modifica componenti per Pattern-Matching.
- **Stack di AR**
  - $>\texttt{ar}_{\texttt{top}} \ldots \texttt{ar}_1]$, struttura LIFO, con $\texttt{ar}_{\texttt{top}}$ ultimo inserito e $\texttt{top} \geq 0$
  - $>]$, Stack vuoto.
  - $\#\Delta$, calcola size dello Stack, ie. numero di ar contenuti in esso.
  - `Pattern-Matching` su termini Stack.

AM21: Il Modello Astratto - Le Operazioni 2

- **Activation Record, AR**.
  - $\{\texttt{head}, \texttt{chain}, \texttt{frame}, \texttt{cont}, \texttt{val}\}$: AR a 5 componenti
  - `Head`, intestazione: `Ide` $+ \{[\texttt{E}], [\texttt{C}]\} + []$
  - `Chain`, static(/dynamic) chain: Offest espresso da intero `k` ≥ 0
  - `Frame`, di ambiente: Vedi sotto
  - `Cont`, continuazione: Sequenza di comandi
  - `Val`, return value: `Val` $+ []$
  - `Pattern-Matching` su termini AR.
- **Frame: Operazioni**.
  - [] Crea un frame vuoto per un AR.
  - $[\texttt{I/D}] \otimes \Delta$ aggiunge il binding `[I/D]` al frame di $\texttt{ar}_{\texttt{top}}$ di Δ
  - $\Delta|_{\texttt{k}}$ frame dello AR k posizioni sotto il top dello stack Δ, se esiste.
  - $\Delta(\texttt{I})$ denotazione, D, del binding di `I` in Δ, se esiste ed in accordo allo scope di Small21.
  - `Pattern-Matching` su termini frame
- **Store: Operazioni**
  - $\triangleright(\mu, \texttt{n})$ alloca in μ, `n` locazioni libere, in sequenza da `loc`, modifica μ in μ', restituisce $(\texttt{loc}, \mu')$
  - $\triangleleft(\mu, \rho)$ dealloca da μ, le locazioni in ρ, che tornano allocabili. Restituisce la memoria modificata
  - $\mu(\texttt{loc})$ (`loc` deve essere una locazione giá allocata) restituisce il valore di `loc`
  - $\mu[\texttt{loc} \leftarrow \texttt{n}]$ (`loc` deve essere giá allocata) modifica il valore di `loc` con il valore `n`
  - `loc` $\oplus$ `k` locazione k words dopo loc.
  - `Pattern-Matching` su termini store.
- **Pattern-Matching**
  - Termine = Pattern: Lega le variabili del Pattern a termini quando può essere reso identico al Termine. Fallisce altrimenti.
  - Uso. Posto in premessa di regole di inferenza, seleziona componenti del termine con cui instanziare la regola. In caso di fallimento rende la regola inapplicabile.

AM21: Il Modello Astratto - Le Operazioni 3

- **Valori Calcolabili e Memorizzabili**.
  - $\mathtt{INT} = \{0, 1, -1, \ldots$
  - $\mathtt{BOOL} = \{\mathtt{Cast(Bool, 0)}, \mathtt{Cast(Bool, 1)}\}$
  - $\mathtt{DVal} = \{\mathtt{loc}_{\mathtt{i}}^{\mathtt{t}} \mid \mathtt{i} \in [0..\mathtt{StoreSize}], \mathtt{t} \in \{\mathtt{Int}, \mathtt{Bool}, [\mathtt{Arr}]\,\mathtt{t}'\,\mathtt{k}\}\}$ – memorizzabili per puntatori

- **Denotazioni**.
  - $(\mathtt{t}, \mathtt{v})$ per costanti: $\mathtt{t} \in \mathtt{Type}$, $\mathtt{v} \in \{\mathtt{t}\}$.
  - $([\mathtt{Mut}]\,\mathtt{t}, \mathtt{loc}_{\mathtt{t1}})$ per modificabili: $[\mathtt{Mut}]\,\mathtt{t} \in \mathtt{Type}$, $\mathtt{loc}_{\mathtt{t1}}$ locazione per valori (memorizzabili) in $\{\mathtt{t}\}$ Per i castable il valore utilizzato per la rappresentazione: Ad es. $([\mathtt{Mut}]\,\mathtt{Bool}, \mathtt{loc}_{\mathtt{Int}})$ per una variabile booleana quando i booleani sono memorizzabili ma non implementati (primitivi).
  - $([\mathtt{Mut}]\,([\mathtt{Arr}]\,\mathtt{t}\,\mathtt{N}), \mathtt{loc}_{\mathtt{t1}})$ per variabili array: $\mathtt{loc}_{\mathtt{t1}}$ locazione per valori nell'insieme utilizzato per la memorizzazione (vedi sopra, valori memorizzabili).
  - $\star\mathtt{I}, \mathtt{t}, (\mathtt{f}_1, \ldots, \mathtt{f}_k), \mathtt{d}, \mathtt{c}, \mathtt{k}\star$ closure per Procedure e Funzioni Procedurali con parametri.
    $\mathtt{I}$ = nome, $\mathtt{t}$ = tipo, $(\mathtt{f}_1, \ldots, \mathtt{f}_k)$ = formali, $\mathtt{d}$ = dichiarazioni, $\mathtt{c}$ = comando, $\mathtt{k}$ = posizione AR di dichiarazione nello Stack, della Procedura, Funzione Procedurale, dichiarata.

- **Notazione: Simboli**
  - [], Struttura Vuota o Valore di Default.
  - $\perp$, Valore Indefinito.
  - σ (anche con pedici, apici), Stato.
  - Δ (anche con pedici, apici), Stack di AR.
  - μ (anche con pedici, apici), Store.
  - ρ (anche con pedici, apici), Frame.
  - N (anche con pedici, apici), Intero.
  - I (anche con pedici, apici), Identificatore.
  - D, den (anche con pedici, apici), Denotazione.
  - loc, l, $\mathrm{loc}^{\mathrm{t}}$ (anche con pedici), Locazione di memoria o word (di tipo t).
  - ff, Sequenza, non vuota, parametri formali.
  - aa, Sequenza, non vuota, parametri attuali.

Dichiarazioni Small21: Le Transizioni $\mathrm{SEM}_{\mathrm{DCL}}$

Il Sistema di regole $\mathtt{Sem}_{\mathrm{DCL}}$ definisce il comportamento delle dichiarazioni durante la computazione dei Programmi Small21 sulla Macchina Astratta AM21.

Controllo dei Tipi Dinamico: Il Sistema $\mathtt{Sem}_{\mathrm{DCL}}$ è Integrato con il Sistema Y (slides sopra, per le dichiarazioni).

$\mathtt{Dcl} ::= [\mathtt{var}]\ \mathtt{Type\ Ide\ Exp} \mid [\mathtt{const}]\ \mathtt{Type\ Ide\ Exp}$
$\mid [\mathtt{array}]\ \mathtt{Type\ Ide} \mid \mathtt{Dcl}\ [\mathtt{seqD}]\ \mathtt{Dcl}$
$\mid [\mathtt{pcd}]\ \mathtt{Type\ Ide\ FPars\ BlockC} \mid [\mathtt{emptyD}]$

$$[\mathrm{D1}]\ \frac{\begin{array}{c}\langle \mathtt{e}, (\Delta, \mu)\rangle \rightarrow \lfloor \mathtt{t_e}, \mathtt{v}, (\Delta, \mu_{\mathtt{e}})\rfloor \\ \mathtt{t} \in \mathtt{Simple} \qquad \mathtt{t} = \mathtt{t_e} \\ \Delta|_0(\mathtt{I}) = \bot \qquad [\mathtt{I}/(\mathtt{t},\mathtt{v})]\otimes\Delta = \Delta_{\mathtt{I}}\end{array}}{\langle [\mathtt{const}]\ \mathtt{t}\ \mathtt{I}\ \mathtt{e}, (\Delta, \mu)\rangle \rightarrow ([\mathtt{void}], (\Delta_{\mathtt{I}}, \mu_{\mathtt{e}}))}$$

$$[\mathrm{D2}]\ \frac{\begin{array}{c}\langle \mathtt{e}, (\Delta, \mu)\rangle \rightarrow \lfloor \mathtt{t_e}, \mathtt{v}, (\Delta, \mu_{\mathtt{e}})\rfloor \\ \mathtt{t} \in \mathtt{Simple} \quad \mathtt{t}' = \mathtt{t} \quad \mathtt{Y}_\rho|_0(\mathtt{I}) = \bot \\ \rhd(\mu_{\mathtt{e}}, \mathtt{t}, 1) = (\mathtt{loc_t}, \mu_{\mathtt{a}}) \quad \mu_{\mathtt{a}}[\mathtt{loc_t} \leftarrow \mathtt{v}] = \mu'_{\mathtt{a}} \\ [\mathtt{I}/([\mathtt{Mut}]\ \mathtt{t_e}, \mathtt{loc_t})]\otimes\Delta = \Delta_{\mathtt{I}}\end{array}}{\langle [\mathtt{var}]\ \mathtt{t}\ \mathtt{I}\ \mathtt{e}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} ([\mathtt{void}], (\Delta_{\mathtt{I}}, \mu'_{\mathtt{a}}))}$$

$$[\mathrm{D2'}]\ \frac{\begin{array}{c}\langle \mathtt{e}, (\Delta, \mu)\rangle \rightarrow \lfloor \mathtt{t_e}, \mathtt{v}, (\Delta, \mu_{\mathtt{e}})\rfloor \\ \mathtt{t} \in \mathtt{Simple} \quad \mathtt{t}' = [\mathtt{unit}] \quad \mathtt{Y}_\rho|_0(\mathtt{I}) = \bot \\ \rhd(\mu_{\mathtt{e}}, \mathtt{t}, 1) = (\mathtt{loc_t}, \mu_{\mathtt{a}}) \\ [\mathtt{I}/([\mathtt{Mut}]\ \mathtt{t_e}, \mathtt{loc_t})]\otimes\Delta = \Delta_{\mathtt{I}}\end{array}}{\langle [\mathtt{var}]\ \mathtt{t}\ \mathtt{I}\ \mathtt{e}, \mathtt{Y}_\rho\rangle \rightarrow_{\mathtt{Y}} ([\mathtt{void}], (\Delta_{\mathtt{I}}, \mu_{\mathtt{a}}))}$$

$$[\mathrm{D3}]\ \frac{\begin{array}{c}\mathtt{t} = [\mathtt{arr}]\ \mathtt{t}'\ \mathtt{N} \qquad \mathtt{N} > 0 \\ \mathtt{t}' \in \mathtt{Simple} \qquad \mathtt{Y}_\rho|_0(\mathtt{I}) = \bot \\ \rhd(\mu, \mathtt{t}, \mathtt{N}) = (\mathtt{loc_t}, \mu_{\mathtt{a}}) \\ [\mathtt{I}/([\mathtt{Arr}]\ ([\mathtt{Mut}]\ \mathtt{t})\ \mathtt{N}, \mathtt{loc_t})]\otimes\Delta = \Delta_{\mathtt{I}}\end{array}}{\langle [\mathtt{array}]\ \mathtt{t}\ \mathtt{I}, (\rho, \mu)\rangle \rightarrow ([\mathtt{Void}], (\Delta_{\mathtt{I}}, \mu_{\mathtt{a}}))}$$

Notazione, Osservazioni

- [xxx] è un costruttore di AST (i.e, Albero di Sintassi Astratta). I rimanenti identificatori (inessenziale se minuscoli o Maiuscoli o se con indici) 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.
- $\mathtt{Simple} = \{[\mathtt{Int}], [\mathtt{Bool}]\}$

Dichiarazioni Small21: Le Transizioni $\mathrm{SEM}_{\mathrm{DCL}}$ — 2

$$[\mathrm{D4}]\ \frac{\langle \mathtt{d}_1, \sigma\rangle \to (\mathtt{[Void]}, \sigma_1) \quad \langle \mathtt{d}_2, \sigma_1\rangle \to (\mathtt{[Void]}, \sigma_2)}{\langle \mathtt{d}_1\ \mathtt{[seqD]}\ \mathtt{d}_2, \sigma\rangle \to (\mathtt{[Void]}, \sigma_2)}$$

$$[\mathrm{D5}]\ \frac{\begin{array}{c}\mathtt{t} \in \mathtt{Simple} \cup \{\mathtt{[void]}\} \qquad \mathtt{F} = \mathtt{[fp]\,p\,t'\,I'} \\ \mathtt{t'} \in \mathtt{Simple} \qquad \Delta|_0(\mathtt{I}) = \bot \qquad \mathtt{[Abs]\,t :: t'} = \mathtt{t_r} \\ \mathtt{Bc} = \mathtt{[BlockS]\,d\,s} \qquad \star\mathtt{I}, \mathtt{t_r}, \mathtt{F}, \mathtt{Bc}, \#\Delta\star = \mathtt{v_r}\end{array}}{\langle \mathtt{[pcd]\,t\,I\,F\,Bc}, \mathtt{Y}_\rho\rangle \to (\mathtt{Void}, (\mathtt{[I/(t_r, v_r)]} \otimes \Delta, \mu))}$$

$$[\mathrm{D6}]\ \frac{\begin{array}{c}\mathtt{t} \in \mathtt{Simple} \cup \{\mathtt{[void]}\} \qquad \Delta|_0(\mathtt{I}) = \bot \\ \mathtt{t'} \in \mathtt{Simple} \qquad \mathtt{F} = \mathtt{[emptyFP]} \qquad \mathtt{[Abs]\,t} = \mathtt{t_r} \\ \mathtt{Bc} = \mathtt{[BlockS]\,d\,s} \qquad \star\mathtt{I}, \mathtt{t_r}, \mathtt{F}, \mathtt{d}, \mathtt{s}, \#\Delta\star = \mathtt{v_r}\end{array}}{\langle \mathtt{[pcd]\,t\,I\,F\,Bc}, \mathtt{Y}_\rho\rangle \to (\mathtt{Void}, (\mathtt{[I/(t_r, v_r)]} \otimes \Delta, \mu))}$$

$$[\mathrm{D7}]\ \frac{}{\langle \mathtt{[emptyD]}, \sigma\rangle \to (\mathtt{[void]}, \sigma)}$$

Notazione, Osservazioni

- [xxx] è un costruttore di AST (i.e, Albero di Sintassi Astratta). I rimanenti identificatori (inessenziale se minuscoli o Maiuscoli o se con indici) 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.
- $\mathtt{Simple} = \{\mathtt{[Int]}, \mathtt{[Bool]}\}$
- $\star\mathtt{I}, \mathtt{t_r}, \mathtt{F}, \mathtt{d}, \mathtt{s}, \#\Delta\star$ Chiusura per procedure/funzioni