

- **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 Corrigere** 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

- AM21 - Lucidi ricollocati prima di Semantica Statica e Semantica Dinamica;
- slide 5. Sostituito Fp con FP e Ap con AP in lista parametri;
- slide 6. Corretta sintassi e aggiunto vincolo su numero e tipo dei parametri di Small21;
- slide 6. Sintassi aggiunto vincolo sui condizionali di Small21;
- slide 9. [12 cv¹] Regola E5 aggiunto caso Unit;
- slide 10. [13 cv] Regola [Y3], $t \in \text{Simple}$ sostituito da $t' \in \text{Simple}$, ancora $[\text{mut}]t$ da $[\text{mut}]t'$; [IB]²
- slide 15. Sintassi Dcl, BlockC sostituito da BlockS;
- slide 15. Regola [D2], $t' = t$ sostituito da $t = t_e$, ancora $Y_p|_0$ da $\Delta|_0$, ancora Y_p da Δ ; [IB]
- slide 15. Regola [D2'], t' sostituito da t_e , ancora $Y_p|_0$ da $\Delta|_0$, ancora Y_p da Δ ; [IB]
- slide 16. Regola [D5], Bc sostituito da d, s, ancora Y_p da Δ ; [IB]
- slide 16. Regola [D6], $t' \in \text{Simple}$ soppresso, Y_p sostituito da Δ ; [IB]

¹ cv = corrente versione, indica la slide corrispondente nella nuova versione

² IB = Ivan Bioli: Correzione ad opera di Ivan Bioli

Sistema di Tipi λ 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 $Y_{DCL}^{Small21}$**

$$\frac{\langle e, Y_\rho \rangle \rightarrow_Y (t', Y'_\rho) \quad t' \neq [\text{terr}] \quad t' = t}{\langle [\text{const}] \ t \ I \ e, Y_\rho \rangle \rightarrow_Y (\text{Void}, [I/t] \circ Y_\rho)}$$

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

$$\langle e, Y_\rho \rangle \rightarrow_Y (t', Y'_\rho)$$

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 Y_ρ , abbia tipo t' (i.e. $t' = t$). Ovvero:

$$Y_\rho \vdash e : t.$$

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

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

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

$$\langle s, Y_\rho \rangle \rightarrow_Y (\text{Void}, [I/t] \circ Y_\rho)$$

ovvero il sistema deduce che la struttura s è/introduce una corretta dichiarazione che estende in modo consistente Y_ρ con il binding $[I/t]$. Ovvero:

$$Y_\rho, I : t \vdash s : \text{Void}.$$

Sistema di Tipi: Regole, Strutture e Proprietà di Y

$$\frac{\langle e, Y_\rho \rangle \rightarrow_Y (t', Y_\rho) \quad t' \neq [\text{terr}] \quad t' = t}{\langle [\text{const}] t \text{ I } e, Y_\rho \rangle \rightarrow_Y (\text{Void}, [I/t] \circ Y_\rho)}$$

- **Regole.** Sono inferenze di un sistema deduttivo che associano ad ogni struttura del programma, s , con tipi degli identificatori definiti da Y_ρ , una coppia (t', Y'_ρ) .
Sopra, in premessa $\langle e, Y_\rho \rangle \rightarrow_Y (t', Y_\rho)$ vediamo ciò. In particolare, il sistema ha dedotto che l'espressione e , dove i suoi identificatori liberi siano tutti legati in Y_ρ , abbia tipo $t' = t$. Ovvero: $Y_\rho \vdash e : t$.
Sopra, nella conclusione, la struttura di programma s è: $[\text{const}] t \text{ I } e$. Quando il sistema è in grado di inferire $\langle s, Y_\rho \rangle \rightarrow_Y (\text{Void}, [I/t] \circ Y_\rho)$ allora il sistema ha dedotto che la struttura s introduce una corretta dichiarazione che estende in modo consistente Y_ρ con il binding $[I/t]$. Ovvero: $Y_\rho, I : t \vdash s : \text{Void}$.
- **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 Y_ρ come il binding dell'identificatore in Y_ρ .
- **Struttura Y_ρ .** 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 Y_ρ sia uno stack $\langle \rho :: Y'_\rho \rangle$ di ambienti senza ripetizioni con ρ ambiente del corrente blocco e Y'_ρ 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 Y_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: $\langle P, Y_L \rangle \rightarrow_Y (\text{Void}, Y_L)$. Ovvero: $Y_L \vdash P : \text{Void}$
 - discendenti: Per ogni termine s contenuto in P , $\langle s, Y_\rho \rangle \rightarrow_Y (t', Y'_\rho)$. Ovvero: $Y_L, Y_\rho \vdash s : t$
- **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 $t \neq [\text{terr}]$.

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

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

- **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^3 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.

³ 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)

Type \rightarrow Simple | void | Simple [Num]

Simple \rightarrow int | bool

Dcl \rightarrow const Simple ide = Exp; | Simple ide = Exp;

| Type ide; | Type ide (FP) Block

Dcls \rightarrow Dcl Dcls | ϵ

Exp \rightarrow ExpB == ExpB | ExpB

ExpB \rightarrow ExpB or ExpR | truth | ExpR

ExpR \rightarrow ExpA > ExpA | ExpA < ExpA | ExpA

ExpA \rightarrow ExpA + ExpT | ExpT

ExpT \rightarrow num | (Exp) | ide (AP) | DExp

DExp \rightarrow ide | ide [ExpA]

Cmd \rightarrow lab : Stm | Stm

Cmds \rightarrow Cmd Cmds | Cmd

Stm \rightarrow if (Exp) Stm | OtherStm

OtherStm \rightarrow if (Exp) OtherStm else Stm | NonConditionalStm

NonConditionalStm \rightarrow DExp = Exp; | goto lab;

| return Exp; | ; | ide (AP); | Block

Stms \rightarrow Stm | Stm Stms

Block \rightarrow {Dcls Stms}

Prog \rightarrow Program ide {Dcls Cmds}

$FP \rightarrow Fp \ FPs \mid \epsilon$ (per estensioni successive)

$FPs \rightarrow , \ Fp \ FPs \mid \epsilon$ (per estensioni successive)

$Fp \rightarrow PPF \ Type \ ide$

$PPF \rightarrow \epsilon \mid \text{ref}$

$AP \rightarrow Ap \ APs \mid \epsilon$ (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 in uso di procedure o funzioni.
- Parametri solo di tipo simple.
- Procedure e funzioni con al più un parametro.
- Condizionale: Consentito uso del solo comando con guardia, `if(exp)stm`.
- 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.

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 valutato nello 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.

- $Y : AST \rightarrow Y_\rho \rightarrow Type * Y_\rho$
 $(\forall a, Y_\rho) \quad Y(a, Y_\rho) = (t, Y'_\rho) \quad \text{iff} \quad \langle a, Y_\rho \rangle \rightarrow_Y (t, Y'_\rho) \in \text{SistemaY}$
- $dclSem : Dcl \rightarrow State \rightarrow Type * State$
 $(\forall d, \sigma) \quad dclSem(d, \sigma) = (t, \sigma') \quad \text{iff} \quad \langle d, \sigma \rangle \rightarrow (t, \sigma') \in \text{SemDCL}$
- $expSem : exp \rightarrow State \rightarrow Type * Val * State$
 $(\forall e, \sigma) \quad expSem(e, \sigma) = (t, v, \sigma') \quad \text{iff} \quad \langle e, \sigma \rangle \rightarrow [t, v, \sigma'] \in \text{SemEXP}$
- $dexpSem : dexp \rightarrow State \rightarrow Type * DVal * State$
 $(\forall ed, \sigma) \quad expSem(ed, \sigma) = (td, vd, \sigma') \quad \text{iff} \quad \langle ed, \sigma \rangle \rightarrow [td, vd, \sigma'] \in \text{SemDEXP}$
- $cmdSem : cmd \rightarrow State \rightarrow Type * State$
 $(\forall c, \sigma) \quad cmdSem(c, \sigma) = (t, \sigma') \quad \text{iff} \quad \langle c, \sigma \rangle \rightarrow (t, \sigma') \in \text{SemCMD}$

● Stato: Strutture, Operazioni e Notazione

- $(\Delta, \mu) : \text{Stato con Stack di AR, } \Delta, \text{ e Store, } \mu.$
- 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

- $\langle ar_{top} \dots ar_1 \rangle$, struttura LIFO, con ar_{top} ultimo inserito e $top \geq 0$
- \rangle , Stack vuoto.
- $\#\Delta$, calcola size dello Stack, ie. numero di ar contenuti in esso.
- Pattern-Matching su termini Stack.

● Activation Record, AR.

- $\{\text{head, chain, frame, cont, val}\}$: AR a 5 componenti
- Head, intestazione: $\text{Ide} + \{[E], [C]\} + []$
- Chain, static(/dynamic) chain: Offest espresso da intero $k \geq 0$
- Frame, di ambiente: Vedi sotto
- Cont, continuazione: Sequenza di comandi
- Val, return value: $\text{Val} + []$
- Pattern-Matching su termini AR.

● Frame: Operazioni.

- $[]$ Crea un frame vuoto per un AR.
- $[I/D] \otimes \Delta$ aggiunge il binding $[I/D]$ al frame di ar_{top} di Δ
- $\Delta|_k$ frame dello AR k posizioni sotto il top dello stack Δ , se esiste.
- $\Delta(I)$ denotazione, D , del binding di I in Δ , se esiste ed in accordo allo scope di Small21.
- Pattern-Matching su termini frame

● Store: Operazioni

- $\triangleright(\mu, n)$ alloca in μ , n locazioni libere, in sequenza da loc , modifica μ in μ' , restituisce (loc, μ')
- $\triangleleft(\mu, \rho)$ dealloca da μ , le locazioni in ρ , che tornano allocabili. Restituisce la memoria modificata
- $\mu(\text{loc})$ (loc deve essere una locazione già allocata) restituisce il valore di loc
- $\mu[\text{loc} \leftarrow n]$ (loc deve essere già allocata) modifica il valore di loc con il valore n
- $\text{loc} \oplus k$ 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.

- $INT = \{0, 1, -1, \dots\}$
- $BOOL = \{True, False\}$
- $Bool = \{Cast(Bool, 0), Cast(Bool, 1)\}$ – per interi usati come booleani
- $DVal = \{loc_i^t \mid i \in [0..StoreSize], t \in \{Int, Bool, [Arr] t' k\}\}$ – memorizzabili per puntatori

● Denotazioni.

- (t, v) per costanti: $t \in Type, v \in \{t\}$.
- $([Mut] t, loc_{t1})$ per modificabili: $[Mut] t \in Type, loc_{t1}$ locazione per valori (memorizzabili) in $\{t\}$
Per i castable il valore utilizzato per la rappresentazione: Ad es. $([Mut] Bool, loc_{Int})$ per una variabile booleana quando i booleani sono memorizzabili ma non implementati (primitivi).
- $([Mut] ([Arr] t N), loc_{t1})$ per variabili array: loc_{t1} locazione per valori nell'insieme utilizzato per la memorizzazione (vedi sopra, valori memorizzabili).
- $\star I, t, (f_1, \dots, f_k), d, c, k \star$ closure per Procedure e Funzioni Procedurali con parametri.
I = nome, t = tipo, (f_1, \dots, f_k) = formali, d = dichiarazioni, c = comando, 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, loc^t (anche con pedici), Locazione di memoria o word (di tipo t).
- ff, Sequenza, non vuota, parametri formali.
- aa, Sequenza, non vuota, parametri attuali.

Sistema Y: Regole per DCL

$$[Y1] \frac{\langle e, Y_\rho \rangle \rightarrow_Y (t', Y_\rho) \quad t \in \text{Simple} \quad t' = t \quad Y_\rho = \triangleright \rho :: Y'_\rho \quad \rho(I) = \perp \quad \triangleright [I/t] \circ \rho :: Y''_\rho = Y''_\rho}{\langle [\text{const}] t I e, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y''_\rho)}$$

$$[Y2] \frac{\langle e, Y_\rho \rangle \rightarrow_Y (t', Y_\rho) \quad t \in \text{Simple} \quad (t' = t \text{ or } t' = [\text{unit}]) \quad Y_\rho |_0(I) = \perp}{\langle [\text{var}] t I e, Y_\rho \rangle \rightarrow_Y ([\text{void}], [I/[\text{mut}]] t \otimes Y_\rho)}$$

Gestione Errori di Tipo:

$$[E1] \frac{t \notin \text{Simple}}{\langle [\text{const}] t I e, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)} \quad [E2] \frac{\langle e, Y_\rho \rangle \rightarrow_Y (t', Y_\rho) \quad t' \neq t}{\langle [\text{const}] t I e, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E3] \frac{Y_\rho |_0(I) \neq \perp}{\langle [\text{const}] t I e, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E4] \frac{t \notin \text{Simple}}{\langle [\text{var}] t I e, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E5] \frac{\langle e, Y_\rho \rangle \rightarrow_Y (t', Y_\rho) \quad t' \neq t \quad t' \neq \text{Unit}}{\langle [\text{var}] t I e, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E6] \frac{Y_\rho |_0(I) \neq \perp}{\langle [\text{var}] t I e, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

Notazione: Simboli e Strutture

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

Sistema Y: Regole per DCL - 2

$$[Y3] \frac{\begin{array}{l} t = [\text{arr}] t' N \quad N > 0 \\ t' \in \text{Simple} \quad Y_\rho |_0(I) = \perp \end{array}}{\langle [\text{array}] t I, Y_\rho \rangle \rightarrow_Y ([\text{void}], [I/[\text{arr}]([\text{mut}] t) N] \otimes Y_\rho)} **$$

$$[Y4] \frac{\begin{array}{l} \langle d1, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y1_\rho) \\ \langle d2, Y1_\rho \rangle \rightarrow_Y ([\text{void}], Y2_\rho) \end{array}}{\langle d1 [\text{seqD}] d2, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y2_\rho)} **$$

Gestione Errori di Tipo:

$$[E7] \frac{t = [\text{arr}] t' N \quad t' \notin \text{Simple}}{\langle [\text{array}] t I, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E8] \frac{t = [\text{arr}] t' N \quad N \leq 0}{\langle [\text{array}] t I, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E9] \frac{Y_\rho |_0(I) \neq \perp}{\langle [\text{array}] t I, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E10] \frac{\langle d1, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y1_\rho)}{\langle d1 [\text{seqD}] d2, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E11] \frac{\begin{array}{l} \langle d1, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y1_\rho) \\ \langle d2, Y1_\rho \rangle \rightarrow_Y ([\text{terr}], Y2_\rho) \end{array}}{\langle d1 [\text{seqD}] d2, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y1_\rho)}$$

Notazione: Simboli e Strutture

- $[]$, ambiente vuoto
- $\rangle_\rho :: Y_\rho$, stack non vuoto di ambienti di legami $[I/t]$, avente ρ come top e Y_ρ come stack residuo.
- $Y_\rho |_n$, n-esimo ($n \geq 0$) ambiente precedente il top dello stack Y_ρ .
- $[I/t] \otimes Y_\rho$, aggiunta del binding $[I/t]$ nel top dello stack Y_ρ .

Sistema Y: Regole per DCL - 3

$$[Y5] \frac{t \in \text{Simple} \cup \{\text{void}\} \quad F = [\text{fp}] p t' I' \quad t' \in \text{Simple} \quad Y_\rho |_0(I) = \perp \quad \triangleright [I'/t'] \circ [] :: Y_\rho = Y'_\rho \quad \langle \text{Bs}, Y'_\rho \rangle \rightarrow_Y ([\text{void}], Y'_\rho)}{\langle [\text{pcd}] t I F \text{Bs}, Y_\rho \rangle \rightarrow_Y ([\text{void}], [I/[\text{abs}] t [::] t'] \otimes Y_\rho)} **$$

$$[Y6] \frac{t \in \text{Simple} \cup \{\text{void}\} \quad F = [\text{emptyFP}] \quad Y_\rho |_0(I) = \perp \quad \triangleright [] :: Y_\rho = Y'_\rho \quad \langle \text{Bs}, Y'_\rho \rangle \rightarrow_Y ([\text{void}], Y'_\rho)}{\langle [\text{pcd}] t I F \text{Bs}, Y_\rho \rangle \rightarrow_Y ([\text{void}], [I/[\text{abs}] t] \otimes Y_\rho)} ** \quad [Y7] \frac{}{\langle [\text{emptyD}], Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho)}$$

Gestione Errori di Tipo:

$$[E12] \frac{t \notin \text{Simple} \cup \{\text{void}\}}{\langle [\text{pcd}] t I F \text{Bs}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E13] \frac{F = [\text{fp}] p t' I' \quad t' \notin \text{Simple}}{\langle [\text{pcd}] t I F \text{Bs}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E14] \frac{Y_\rho |_0(I) \neq \perp}{\langle [\text{pcd}] t I F \text{Bs}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E15] \frac{\dots \quad \langle \text{Bs}, \triangleright \rho :: Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y'_\rho)}{\langle [\text{pcd}] t I F \text{Bs}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

Notazione: Simboli e Strutture

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

Dichiarazioni Small21: Le Transizioni SEM_{DCL}

Il Sistema di regole SEM_{DCL} definisce il comportamento delle dichiarazioni durante la computazione dei Programmi Small21 sulla Macchina Astratta AM21.

Controllo dei Tipi Dinamico: Il Sistema SEM_{DCL} è Integrato con il Sistema Y (slides sopra, per le dichiarazioni).

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

$$[D1] \frac{\langle e, (\Delta, \mu) \rangle \rightarrow \lfloor t_e, v, (\Delta, \mu_e) \rfloor \quad t \in \text{Simple} \quad t = t_e \quad \Delta|_0(I) = \perp \quad [I/(t,v)] \otimes \Delta = \Delta_I}{\langle [\text{const}] t I e, (\Delta, \mu) \rangle \rightarrow ([\text{void}], (\Delta_I, \mu_e))}$$

$$[D2] \frac{\langle e, (\Delta, \mu) \rangle \rightarrow \lfloor t_e, v, (\Delta, \mu_e) \rfloor \quad t \in \text{Simple} \quad t = t_e \quad \Delta|_0(I) = \perp \quad \triangleright(\mu_e, t, 1) = (\text{loc}_t, \mu_a) \quad \mu_a[\text{loc}_t \leftarrow v] = \mu'_a \quad [I/([\text{Mut}] t_e, \text{loc}_t)] \otimes \Delta = \Delta_I}{\langle [\text{var}] t I e, Y_\rho \rangle \rightarrow_Y ([\text{void}], (\Delta_I, \mu'_a))}$$

$$[D2'] \frac{\langle e, (\Delta, \mu) \rangle \rightarrow \lfloor t_e, v, (\Delta, \mu_e) \rfloor \quad t \in \text{Simple} \quad t_e = [\text{unit}] \quad \Delta|_0(I) = \perp \quad \triangleright(\mu_e, t, 1) = (\text{loc}_t, \mu_a) \quad [I/([\text{Mut}] t_e, \text{loc}_t)] \otimes \Delta = \Delta_I}{\langle [\text{var}] t I e, \Delta \rangle \rightarrow_Y ([\text{void}], (\Delta_I, \mu_a))}$$

$$[D3] \frac{t = [\text{arr}] t' N \quad N > 0 \quad t' \in \text{Simple} \quad Y_\rho|_0(I) = \perp \quad \triangleright(\mu, t, N) = (\text{loc}_t, \mu_a) \quad [I/([\text{Arr}] ([\text{Mut}] t) N, \text{loc}_t)] \otimes \Delta = \Delta_I B_s}{\langle [\text{array}] t I, (\rho, \mu) \rangle \rightarrow ([\text{Void}], (\Delta_I, \mu_a))}$$

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.
- Simple = {[Int], [Bool]}

$$[D4] \frac{\begin{array}{l} \langle d_1, \sigma \rangle \rightarrow ([\text{Void}], \sigma_1) \\ \langle d_2, \sigma_1 \rangle \rightarrow ([\text{Void}], \sigma_2) \end{array}}{\langle d_1 [\text{seqD}] d_2, \sigma \rangle \rightarrow ([\text{Void}], \sigma_2)}$$

$$[D5] \frac{\begin{array}{l} t \in \text{Simple} \cup \{\text{void}\} \quad F = [\text{fp}] p t' I' \\ t' \in \text{Simple} \quad \Delta|_0(I) = \perp \quad [\text{Abs}] t :: t' = t_r \\ \text{Bs} = [\text{BlockS}] d s \quad *I, t_r, F, d, s, \#\Delta* = v_r \end{array}}{\langle [\text{pcd}] t I F \text{Bs}, \Delta \rangle \rightarrow (\text{Void}, ([I/(t_r, v_r)] \otimes \Delta, \mu))}$$

$$[D6] \frac{\begin{array}{l} t \in \text{Simple} \cup \{\text{void}\} \quad \Delta|_0(I) = \perp \\ F = [\text{emptyFP}] \quad [\text{Abs}] t = t_r \\ \text{Bs} = [\text{BlockS}] d s \quad *I, t_r, F, d, s, \#\Delta* = v_r \end{array}}{\langle [\text{pcd}] t I F \text{Bs}, \Delta \rangle \rightarrow (\text{Void}, ([I/(t_r, v_r)] \otimes \Delta, \mu))}$$

$$[D7] \frac{}{\langle [\text{emptyD}], \sigma \rangle \rightarrow ([\text{void}], \sigma)}$$

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.
- $\text{Simple} = \{[\text{Int}], [\text{Bool}]\}$
- $*I, t_r, F, d, s, \#\Delta*$ Chiusura per procedure/funzioni