

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^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)

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 | 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}

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

$\text{Prog} \rightarrow \text{Program } \text{ide } \{ \text{Dcls } \text{Cmds} \}$

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

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

$\text{Fp} \rightarrow \text{PPF } \text{Type } \text{ide}$

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

$\text{AP} \rightarrow \text{Ap } \text{APs} \mid \epsilon$ (per estensioni successive)

$\text{APs} \rightarrow, \text{Ap } \text{APs} \mid \epsilon$ (per estensioni successive)

$\text{Ap} \rightarrow \text{Exp}$

Vincoli Contestuali

- Identificatori usati devono essere dichiarati ed hanno scope statico.
- 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 solo nel blocco Programma (i.e. outermost block).

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
| 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 | [seqC] Cmd Cmd

Stm ::= DExp [=] Exp | [ifE] Exp 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 Cmd

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.: $x \uparrow 3$ 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.

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

● 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.
- $ar + \Delta$, stack con ar come top e Δ come coda dello stack.
- $\#\Delta$, 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: $\text{Ide} + \{\{\text{ipb}\}, [\text{E}], [\text{C}]\} + []$
- Chain, static(/dynamic) chain: Offset espresso da intero $k \geq 0$
- Frame, di ambiente: Vedi sotto
- Cont, continuazione: Sequenza di comandi: $c + C$ aggiunge c in testa alla continuazione C
- 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).
- $\star I, t, (f_1, \dots, f_k), d, c, k$ 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.
- (Lab, cnt) per label: Lab tipo di types per le labels, cnt un AST (nota: non chiusura - Perché?)

● 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

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

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

Gestione Errori di Tipo:

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

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

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

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

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

Notazione: Simboli e Strutture

- $[]$, ambiente vuoto
- $>\rho::Y_\rho$, 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

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

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

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]

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

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

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

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

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]}

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

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

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

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

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

Sistema Y: Regole per EXP

$Exp ::= DExp \mid [num] Num \mid [true] \mid [false] \mid [emptyE] \mid Exp \mathcal{O}_2 Exp \mid \mathcal{O}_1 Exp$
 $DExp ::= [val] Ide \mid Ide [\uparrow] Exp$

Sistema Y

$$\begin{array}{c} [Y8] \frac{}{\langle [num] N, Y_\rho \rangle \rightarrow_Y (\langle [int], Y_\rho \rangle)} \quad [Y9] \frac{}{\langle [true], Y_\rho \rangle \rightarrow_Y (\langle [bool], Y_\rho \rangle)} \quad [Y10] \frac{}{\langle [false], Y_\rho \rangle \rightarrow_Y (\langle [bool], Y_\rho \rangle)} \\ [Y11] \frac{}{\langle [emptyE], Y_\rho \rangle \rightarrow_Y (\langle [bool], Y_\rho \rangle)} \\ [Y12] \frac{Y_\rho(I) = [mut] t \quad t \in Simple}{\langle [val] I, Y_\rho \rangle \rightarrow_{DY} (\langle [mut] t, Y_\rho \rangle)} \quad [Y13] \frac{Y_\rho(I) = [mut] t \quad t \in Simple}{\langle [val] I, Y_\rho \rangle \rightarrow_Y (t, Y_\rho)} \quad [Y14] \frac{Y_\rho(I) = t \quad t \in Simple}{\langle [val] I, Y_\rho \rangle \rightarrow_Y (t, Y_\rho)} \\ [Y15] \frac{}{\langle I [\uparrow] e, Y_\rho \rangle \rightarrow_{DY}} \quad [Y16] \frac{}{\langle I [\uparrow] e, Y_\rho \rangle \rightarrow_Y} \\ [Y17] \frac{\langle e_1, Y_\rho \rangle \rightarrow_Y (t_1, Y_\rho) \quad \langle e_2, Y_\rho \rangle \rightarrow_Y (t_2, Y_\rho) \quad Y_\rho(op) = [abs] t[::]t'_1[::]t'_2 \quad op \in \mathcal{O}_2 \quad t'_1 = t_1 \quad t'_2 = t_2}{\langle e_1 [op] e_2, Y_\rho \rangle \rightarrow_Y (t, Y_\rho)} \quad [Y18] \frac{}{\langle [op] e, Y_\rho \rangle \rightarrow_Y} \end{array}$$

Notazione

- \rightarrow_{DY} è l'inferenza di Tipo del Valore Denotabile di espressioni con doppio significato.
- $\mathcal{O}_2 = \{+, =, >, <, or\}$; $\mathcal{O}_1 = \{\}$.
- $Simple = \{\langle [int], [bool] \rangle\}$

Sistema Y: Regole per EXP - Errori di tipo

$\text{Exp} ::= \text{DExp} \mid [\text{num}] \text{Num} \mid [\text{true}] \mid [\text{false}] \mid [\text{emptyE}] \mid \text{Exp } \mathcal{O}_2 \text{ Exp} \mid \mathcal{O}_1 \text{ Exp}$
 $\text{DExp} ::= [\text{val}] \text{Ide} \mid \text{Ide } [\uparrow] \text{ Exp}$

Sistema Y: Gestione Errori di Tipo

$$[\text{E16}] \frac{Y_\rho(\text{I}) = [\text{mut}] \text{t} \quad \text{t} \notin \text{Simple}}{\langle [\text{val}] \text{I}, Y_\rho \rangle \rightarrow_{\text{DY}} ([\text{terr}], Y_\rho)} \quad [\text{E17}] \frac{Y_\rho(\text{I}) = \text{t} \quad \text{t} \neq [\text{mut}] \text{t}'}{\langle [\text{val}] \text{I}, Y_\rho \rangle \rightarrow_{\text{DY}} ([\text{terr}], Y_\rho)} \quad [\text{E18}] \frac{Y_\rho(\text{I}) = [\text{mut}] \text{t}}{\langle [\text{val}] \text{I}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[\text{E19}] \frac{Y_\rho(\text{I}) = \perp}{\langle [\text{val}] \text{I}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[\text{E25}] \frac{\langle \text{e}_1, Y_\rho \rangle \rightarrow_Y (\text{t}_1, Y_\rho) \quad Y_\rho(\text{op}) = [\text{abs}] \text{t } \text{t}'_1[\text{x}]\text{t}'_2 \quad \text{t}'_1 \neq \text{t}_1}{\langle \text{e}_1 [\text{op}] \text{e}_2, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)} \quad [\text{E26}] \frac{\langle \text{e}_2, Y_\rho \rangle \rightarrow_Y (\text{t}_2, Y_\rho) \quad Y_\rho(\text{op}) = [\text{abs}] \text{t } \text{t}'_1[\text{x}]\text{t}'_2 \quad \text{t}'_2 \neq \text{t}_2}{\langle \text{e}_1 [\text{op}] \text{e}_2, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)} \quad [\text{E27}] \frac{\langle \text{e}, Y_\rho \rangle \rightarrow_Y (\text{t}_e, Y_\rho) \quad Y_\rho(\text{op}) = [\text{abs}] \text{t } \text{t}' \quad \text{t}' \neq \text{t}_e}{\langle [\text{op}] \text{e}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

Notazione

\rightarrow_{DY} è l'inferenza di Tipo del Valore Denotabile di espressioni con doppio significato.

$\mathcal{O}_2 = \{+, -, *, \text{div}, ==, >, <, \text{or}, \text{and}\}; \quad \mathcal{O}_1 = \{-1, \text{not}\}.$

$\text{Simple} = \{[\text{int}], [\text{bool}]\}$

Espressioni con Stato: Le Transizioni SEM_{EXP} e SEM_{DEN}

Il Sistema di regole SEM_{EXP} (risp. SEM_{DEN}) definisce il comportamento delle r-espressioni (risp. l-espressioni) durante la computazione dei Programmi Small21 sulla Macchina Astratta AM21.

Controllo dei Tipi Dinamico: I Sistemi SEM_{EXP} e SEM_{DEN} sono Integrati con il Sistema Y (ultime 2 slides precedenti, per le espressioni).

$Exp ::= [val] Ide \mid [num] N \mid [true] \mid [false] \mid [emptyE] \mid \dots$

Semantica Dinamica: Regole di Inferenza

$$X1: \frac{}{\langle [num] N, \sigma \rangle \rightarrow \llbracket [int], N, \sigma \rrbracket}$$

$$X2: \frac{}{\langle [true], \sigma \rangle \rightarrow \llbracket [bool], [true], \sigma \rrbracket}$$

$$X2': \frac{}{\langle [false], \sigma \rangle \rightarrow \llbracket [bool], [true], \sigma \rrbracket}$$

$$X3: \frac{}{\langle [emptyE], \sigma \rangle \rightarrow \llbracket [unit], [emptyE], \sigma \rrbracket}$$

$$X4: \frac{\begin{array}{l} \sigma = (\Delta, \mu) \\ \Delta(I) = (t, loc_{t'}) \\ t = [mut] t' \\ t' \in Simple \end{array}}{\langle [val] I, \sigma \rangle \rightarrow_{Den} \llbracket t, loc_{t'}, \sigma \rrbracket}$$

$$X5: \frac{\begin{array}{l} \sigma = (\Delta, \mu) \\ \Delta(I) = ([mut] t, loc_t) \\ t \in Simple \\ \mu(loc_t) = v_t \end{array}}{\langle [val] I, \sigma \rangle \rightarrow \llbracket t, v_t, \sigma \rrbracket}$$

$$X6: \frac{\begin{array}{l} \sigma = (\Delta, \mu) \\ \Delta(I) = (t, v_t) \\ t \in Simple \end{array}}{\langle [val] I, \sigma \rangle \rightarrow \llbracket t, v_t, \sigma \rrbracket}$$

Notazione e Osservazioni

- \rightarrow ha termine destro una tripla indicante tipo, valore (esprimibile/denotabile), stato, racchiusa da $\llbracket \cdot \rrbracket$.
- \rightarrow_{DEN} è la relazione per di Tipo-ValoreDenotato di espressioni con doppio significato.
- $N \in [0..N_k - 1]$ è il controllo a run-time sui bounds inferiore e superiore per i componenti di array statico.

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

$\text{Exp} ::= \text{Exp} [\text{op}] \text{Exp} \mid \dots$

Semantica Dinamica: Regole di Inferenza

$$\begin{array}{c} \langle e_1, \sigma \rangle \rightarrow [t_1, v_1, \sigma_1] \\ \langle e_2, \sigma_1 \rangle \rightarrow [t_2, v_2, \sigma_2] \\ \Delta(\text{op}) = [\text{abs}] \ t[::]t'_1[::]t'_2 \\ t'_1 = t_1 \quad t'_2 = t_2 \\ \text{op} \in \mathcal{O}_2 \quad \overline{\text{op}}(v_1, v_2) = v \\ \hline [\text{X9}] \ \langle e_1 [\text{op}] e_2, \sigma \rangle \rightarrow [t, v, \sigma_2] \end{array}$$

Osservazioni (Dispatcher)

La regola [X9] è adatta all'uso del modello dispatcher per la gestione delle operazioni.

Dispatcher è un meccanismo hw-sw per la selezione dell'operazione (procedura, funzione, ect..) data una sigature, i.e. nome e tipo, dell'operazione richiesta.

Notazione

- $\Delta(\text{op})$ fornisce il tipo di operatori primitivi (forniti dallo RTS di AM21)
- $\mathcal{O}_2 = \{+, -, *, \text{div}, ==, >, <, \text{or}, \text{and}\}$; $\mathcal{O}_1 = \{-1, \text{not}\}$.
- $\overline{\text{op}}$ è l'operazione dello RTS con cui implementiamo l'operatore op: Usiamo le corrispondenti op di Ocaml (se definite), implementazioni ad hoc, altrimenti.
- $\text{Simple} = \{\text{int}, \text{bool}\}$

Sistema Y: Regole per CMD, STM, PROG

$\text{Cmd} ::= \text{Lab} [:] \text{Stm} \mid \text{Stm} \mid [\text{seqC}] \text{Cmd} \text{Cmd}$
 $\text{Stm} ::= \dots \mid [\text{goto}] \text{Lab} \mid \dots$

Sistema Y

$$\text{[Y19]} \frac{\begin{array}{c} \#Y_\rho = 1 \\ Y_\rho |_0(L) = \perp \quad [L/[\text{lab}]] \otimes Y_\rho = Y'_\rho \\ \langle s, Y'_\rho \rangle \rightarrow_Y ([\text{void}], Y'_\rho) \end{array}}{\langle L [:] s, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y'_\rho)}$$

$$\text{[Y22]} \frac{\begin{array}{c} \#Y_\rho = 1 \\ Y_\rho |_0(L) = [\text{lab}] \end{array}}{\langle [\text{goto}] L, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho)} \quad \text{[Y23]} \frac{\begin{array}{c} \#Y_\rho = 1 \\ Y_\rho |_0(L) = \perp \\ \text{JumpAheadCheck} \end{array}}{\langle [\text{goto}] L, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho)} \quad (***)$$

$$\text{[Y24]} \frac{\begin{array}{c} \#Y_\rho = 1 \\ \langle c_1, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y'_\rho) \\ \langle s_2, Y'_\rho \rangle \rightarrow_Y ([\text{void}], Y''_\rho) \end{array}}{\langle [\text{seqC}] c_1 c_2, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y''_\rho)}$$


Gestione Errori di Tipo:

$$\text{[E35]} \frac{\#Y_\rho \neq 1}{\langle L [:] s, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y'_\rho)}$$

$$\text{[E36]} \frac{Y_\rho |_0(L) \neq \perp}{\langle L [:] s, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y'_\rho)}$$

$$\text{[E37]} \frac{\dots \dots \dots \langle s, Y'_\rho \rangle \rightarrow_Y ([\text{terr}], Y'_\rho)}{\langle L [:] s, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y'_\rho)}$$

Notazione e Osservazioni: Costruttori Aggiuntivi

- **JumpAheadCheck**: Richiede Controllo a run-time (trattabile in Semantica dinamica);
- *******: In [Y22] e [Y23] dove affermiamo che goto può occore solo nel blocco più esterno di un programma?;
- **[E36]**: Non includere nella Semantica Statica (vedi slide successiva) 

Sistema Y: La regola [E36]

La regola [E36] esprime correttamente l'unicità di una label in un blocco (in questo caso il blocco più esterno)

$$[E36] \frac{Y_\rho |_0(L) \neq \perp}{\langle L [:] s, Y_\rho \rangle \rightarrow_Y (\langle \text{terr} \rangle, Y'_\rho)}$$

Occorre tuttavia, fare alcune osservazioni circa il suo impiego nella successiva semantica dinamica (con controllo dinamico dei tipi, delle proprietà statiche e dei vincoli di sintassi contestuale).

- Non è adatta in caso di controllo dinamico e in presenza di statement goto.
- Gli statement goto richiedono la gestione delle sezioni di programma etichettate:
Ad esempio, sia $s = l1:c1;s1;l2:c2;s2;l3:c3;s3$; una sequenza del programma. Allora s contiene:
per goto l1 la sezione $c1;s1;l2:c2;s2;l3:c3;s3$; e per goto l2 la sezione $c2;s2;l3:c3;s3$;
che condividono una sequenza contenente l3.
Ciò conduce ad una multipla occorrenza di l3 nelle sezioni.
- Come gestire [E36] quando attraversiamo la sezione di goto l2 dopo aver attraversato quella di goto l1?
- Le soluzioni per farlo sono inutilmente complicate nel caso di Small21.

La **Soluzione** che seguiremo è quella più semplice:

- La regola [E36] non è inclusa nella semantica dinamica - Vedi regola [C1'];
- Il controllo dell'Unicità delle labels è condotto:
 - staticamente dal Front-End ed assunto verificato sui programmi usati, oppure
 - staticamente da un controllo (pre-processor) condotto dal nostro esecutore attraversando i comandi dell'outermost block, prima di procedere con il calcolo della computazione.

Osservazioni

- Unicità della label: È possibile definire Linguaggi con multipla occorrenza di label di statement all'interno di uno stesso blocco.

Sistema Y: Regole per CMD, STM, PROG - 2

$\text{Cmd} ::= \text{Lab} [:] \text{Stm} \mid \text{Stm} \mid [\text{seqC}] \text{Cmd} \text{Cmd}$

$\text{Stm} ::= \dots \mid [\text{goto}] \text{Lab} \mid \dots$

Gestione Errori di Tipo:

$$[\text{E43}] \frac{Y_\rho |_0(L) = t \quad t \neq [\text{lab}]}{\langle [\text{goto}] L, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)} \quad [\text{E44}] \frac{Y_\rho |_0(L) = \perp \quad \text{JumpAheadCheckFail}}{\langle [\text{goto}] L, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)} \quad [\text{E45}] \frac{\#Y_\rho \neq 1}{\langle [\text{goto}] L, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[\text{E56}] \frac{\#Y_\rho \neq 1}{\langle [\text{seqC}] c_1 c_2, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y'_\rho)}$$

$$[\text{E57}] \frac{\langle c_1, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y'_\rho)}{\langle [\text{seqC}] c_1 c_2, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y'_\rho)}$$

$$[\text{E58}] \frac{\langle c_1, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y'_\rho) \quad \langle s_2, Y'_\rho \rangle \rightarrow_Y ([\text{terr}], Y''_\rho)}{\langle [\text{seqC}] c_1 c_2, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y''_\rho)}$$

Notazione e Osservazioni: Costruttori Aggiuntivi

- [E45]: Perché anche questa regola?;

Comandi Small21 - Regole di Inferenza SEM_{CMD}, SEM_{STM}

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

Controllo dei Tipi Dinamico: Il Sistema SEM_{CMD} è Integrato con il Sistema Y (ultime 2 slide sopra, per i comandi).

Cmd ::= Lab [:] Stm | Stm | [seqC] Cmd Cmd | ...

Semantica Dinamica: Regole di Inferenza

$$[C1] \frac{\begin{array}{l} \#\Delta = 1 \quad \Delta = \text{>ar}_1 \\ \text{ar}_1 = \{\text{hd, cs, fr, cnt, v}\} \\ [\text{seqC}] \text{ s cnt} = \text{c}' \\ \Delta|_0(L) = \perp \quad [L/([\text{lab}], \text{c}')] \otimes \Delta = \Delta' \\ \langle \text{s}, (\Delta', \mu) \rangle \rightarrow ([\text{void}], \sigma_F) \end{array}}{\langle L [:] \text{s}, (\Delta, \mu) \rangle \rightarrow ([\text{void}], \sigma_F)}$$
$$[C1'] \frac{\begin{array}{l} \#\Delta = 1 \\ \Delta|_0(L) \neq \perp \\ \langle \text{s}, (\Delta', \mu) \rangle \rightarrow ([\text{void}], \sigma_F) \end{array}}{\langle L [:] \text{s}, (\Delta, \mu) \rangle \rightarrow ([\text{void}], \sigma_F)}$$

$$[C2] \frac{\begin{array}{l} \#\Delta = 1 \quad \Delta = \text{>ar}_1 \\ \text{ar}_1 = \{\text{hd, cs, fr, cnt, v}\} \\ [\text{seqC}] \text{ c}_2 \text{ cnt} = \text{cnt}' \\ \{\text{hd, cs, fr, cnt}', \text{v}\} = \text{ar}'_1 \\ \text{>ar}'_1 = \Delta' \\ \langle \text{c}_1, (\Delta', \mu) \rangle \rightarrow ([\text{void}], \sigma_F) \end{array}}{\langle [\text{seqC}] \text{ c}_1 \text{ c}_2, (\Delta, \mu) \rangle \rightarrow ([\text{void}], \sigma_F)}$$

Notazione e Osservazioni: Costruttori Aggiuntivi

- Denotazione: Estese con (t, C) per le label con $t = \text{Lab}$ e C una continuazione (rappresentata come sotto)
- Continuazione C : Rappresentata con AST di Small21e c in testa alla continuazione C ;
- $[\text{seqC}] c C$: Aggiunge c in testa alla continuazione C ;
- (*corretto13/05/21*): Perché una volta incontrata e legata nel frame, una label deve essere ignorata?;
- Il predicato $\Delta = \text{>ar}_1$ implica $\Delta = \text{>ar}_1$: Perché ?;

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

Controllo dei Tipi Dinamico: Il Sistema SEM_{CMD} è Integrato con il Sistema Y.

Stm ::= [goto] Lab | ...

Regole di Inferenza

$$\begin{array}{c}
 \Delta|_0(L) = ([lab], c) \\
 \Delta = >ar_1 \\
 ar_1 = \{hd, cs, fr, cnt, v\} \\
 \{hd, cs, fr, emptyC, v\} = ar'_1 \\
 (>ar'_1, \mu) = \sigma' \\
 [S3] \frac{\langle c, \sigma' \rangle \rightarrow ([void], \sigma_F)}{\langle [goto] L, (\Delta, \mu) \rangle \rightarrow ([void], \sigma_F)}
 \end{array}$$

$$\begin{array}{c}
 \Delta|_0(L) = \perp \quad \Delta = >ar_1 \\
 ar_1 = \{hd, cs, fr, cnt, v\} \\
 \text{JACL cnt} = [\text{seqC}] c cs \\
 >ar'_1 = \Delta' \quad c = L [:] s \\
 \{hd, cs, fr, cs, v\} = ar'_1 \\
 [\text{seqC}] s cs = c' \\
 [L/([lab], c')] \otimes \Delta' = \Delta'' \\
 [S3'] \frac{\langle s, (\Delta'', \mu) \rangle \rightarrow ([void], \sigma_F)}{\langle [goto] L, (\Delta, \mu) \rangle \rightarrow ([void], \sigma_F)} (***)
 \end{array}$$

Notazione ed Osservazioni: Costruttori Aggiuntivi

- [seqC] c C: Aggiunge c in testa alla continuazione C;
- JACL C = C': Vera quando C' è la sotto-sequenza terminale con primo comando etichettato con L. Falsa altr.
- emptyC: Indica "nessuna continuazione"
- (**): Usa l'invariante di inserire il binding di una label e successivamente ignorarne la presenza ogni prima volta che si seleziona un comando etichettato e se ne considera la trasformazione definita.

Nucleo di Small21 per Macchina a Stato Minsky-Wang

Costrutti Minori: Semantica Statica e Dinamica

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

$Stm ::= [ifT] Exp Stm \mid \dots$

Sistema Y:

$$[Y21] \frac{\langle e, Y_\rho \rangle \rightarrow_Y ([bool], Y_\rho) \quad \langle s_1, Y_\rho \rangle \rightarrow_Y ([void], Y_\rho)}{\langle [ifT] e s_1, Y_\rho \rangle \rightarrow_Y ([void], Y_\rho)}$$

Gestione Errori di Tipo:

$$[E41] \frac{\langle e, Y_\rho \rangle \rightarrow_Y (t, Y_\rho) \quad t \neq [bool]}{\langle [ifT] e s_1, Y_\rho \rangle \rightarrow_Y ([terr], Y_\rho)}$$

$$[E42] \frac{\langle s_1, Y_\rho \rangle \rightarrow_Y ([terr], Y_\rho)}{\langle [ifT] e s_1, Y_\rho \rangle \rightarrow_Y ([terr], Y_\rho)}$$

Regole di Inferenza SEM_{CMD}

$$[S2] \frac{\langle e, \sigma \rangle \rightarrow [[bool], true, \sigma_e] \quad \langle c, \sigma_e \rangle \rightarrow ([void], \sigma_1)}{\langle [ifT] e c, \sigma \rangle \rightarrow ([void], \sigma_1)}$$

$$[S2'] \frac{\langle e, \sigma \rangle \rightarrow [[bool], false, \sigma_e]}{\langle [ifT] e c, \sigma \rangle \rightarrow ([void], \sigma_e)}$$

Osservazioni

- Definizione Completa dello Statement con guardia: `Costrutto condizionale if;`

Nucleo di Small21 per Macchina a Stato Minsky-Wang

Costrutti Minori: Semantica Statica e Dinamica - 2

$Stm ::= Exp [=] Exp \mid ES \mid \dots$

Sistema Y:

$$[Y15] \frac{\langle e_1, Y_\rho \rangle \rightarrow_{DY} (t_1, Y_\rho) \quad \langle e_r, Y_\rho \rangle \rightarrow_Y (t_r, Y_\rho) \quad t_1 = [Mut] t \quad t = t_r \quad t \in Simple}{\langle e_1 [=] e_r, Y_\rho \rangle \rightarrow_Y (t, Y_\rho)}$$
$$[Y16] \frac{}{\langle ES, Y_\rho \rangle \rightarrow ([void], Y_\rho)}$$

Gestione Errori di Tipo:

$$[E17] \frac{\langle e_1, Y_\rho \rangle \rightarrow_{DY} ([mut] t, Y_\rho) \quad t \notin Simple}{\langle e_1 [=] e_r, Y_\rho \rangle \rightarrow_Y ([terr], Y_\rho)}$$
$$[E18] \frac{\langle e_r, Y_\rho \rangle \rightarrow_Y (t, Y_\rho) \quad t \notin Simple}{\langle e_1 [=] e_r, Y_\rho \rangle \rightarrow_Y ([terr], Y_\rho)}$$
$$[E19] \frac{\langle e_1, Y_\rho \rangle \rightarrow_{DY} (t_1, Y_\rho) \quad \langle e_r, Y_\rho \rangle \rightarrow_Y (t_r, Y_\rho) \quad t_1 = [Mut] t \quad t \neq t_r}{\langle e_1 [=] e_r, Y_\rho \rangle \rightarrow_Y (t, Y_\rho)}$$

Regole di Inferenza SEM_{CMD}

$$[S1] \frac{\langle e_r, \sigma \rangle \rightarrow [t_r, v_r, \sigma_r] \quad \langle e_1, \sigma_r \rangle \rightarrow_{DEN} [t_1, loc_t, \sigma_1] \quad t_1 = [mut] t \quad t = t_r \quad t \in Simple \quad \sigma_1 = (\Delta_1, \mu_1) \quad \mu_1[loc_t \leftarrow v_r] = \mu_F}{\langle e_1 [=] e_r, \sigma \rangle \rightarrow ([void], (\Delta_1, \mu_F))}$$
$$[S8] \frac{}{\langle ES, \sigma \rangle \rightarrow ([void], \sigma)}$$

Notazione e Osservazioni

- \rightarrow_{DY} è l'inferenza di Tipo del Valore Denotabile di espressioni con doppio significato.
- \rightarrow_{DEN} è l'inferenza semantica dinamica di espressioni con doppio significato.
- $=$ valuta gli argomenti da destra a sinistra come in C/C++
- $Simple = \{[int], [bool]\}$

Small21: Programma e outermost inline block

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

Prog ::= [prog] Ide Block

Block ::= [block] Dcl Cmd

Sistema Y:

$$\begin{array}{c} Bc = [\text{block}]dscs \\ \langle ds, Y_\rho^0 \rangle \rightarrow_Y ([\text{void}], Y_\rho^1) \\ \langle cs, Y_\rho^1 \rangle \rightarrow_Y ([\text{void}], Y_\rho^F) \\ \hline [\text{Y44}] \frac{}{\langle [\text{prog}] I Bc, Y_\rho^0 \rangle \rightarrow_Y ([\text{void}], Y_\rho^F)} \end{array}$$

Gestione Errori di Tipo:

$$[\text{E96}] \frac{\langle ds, Y_\rho^0 \rangle \rightarrow_Y ([\text{terr}], Y_\rho^1)}{\langle [\text{prog}] I Bc, Y_\rho^0 \rangle \rightarrow_Y ([\text{terr}], Y_\rho^1)}$$

$$[\text{E97}] \frac{\langle cs, Y_\rho^1 \rangle \rightarrow_Y ([\text{terr}], Y_\rho^F)}{\langle [\text{prog}] I Bc, Y_\rho^0 \rangle \rightarrow_Y ([\text{terr}], Y_\rho^F)}$$

Osservazioni

- Y_ρ^0 : Contesto contenente i binding di tipo per tutti gli identificatori di primitive del Linguaggio;

Small21: Programma e outermost inline block - 2

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

Prog ::= [prog] Ide Block

Block ::= [block] Decl Cmd

Regole di Inferenza SEM_{CMD}

$$\begin{array}{c}
 \Delta = \text{ar} + \Delta_R \\
 \text{ar} = \{\text{id}, \text{lc}, \text{fr}, \text{c} + \text{cnt}, \text{v}\} \\
 \{\text{id}, \text{lc}, \text{fr}, \text{cnt}, \text{v}\} = \text{ar}_1 \\
 \langle \text{c}, (\text{ar}_1 + \Delta_R, \mu) \rangle \rightarrow ([\text{void}], (\text{ar}_c + \Delta_R, \mu_c)) \\
 \text{ar} = \{\text{id}', \text{lc}', \text{fr}', \text{cnt}', \text{v}'\} \\
 \text{cnt} = [] \\
 (\text{ar}_c + \Delta_R, \mu_c) = \sigma_{\text{next}} \\
 \hline
 [\text{R1}] \frac{}{(\Delta, \mu) \rightarrow_{R^*} \sigma_{\text{next}}}
 \end{array}$$

$$\begin{array}{c}
 \Delta = \text{ar} + \Delta_R \\
 \text{ar} = \{\text{id}, \text{lc}, \text{fr}, \text{c} + \text{cnt}, \text{v}\} \\
 \{\text{id}, \text{lc}, \text{fr}, \text{cnt}, \text{v}\} = \text{ar}_1 \\
 \langle \text{c}, (\text{ar}_1 + \Delta_R, \mu) \rangle \rightarrow ([\text{void}], (\text{ar}_c + \Delta_R, \mu_c)) \\
 \text{ar} = \{\text{id}', \text{lc}', \text{fr}', \text{cnt}', \text{v}'\} \\
 \text{cnt} \neq [] \\
 (\text{ar}_c + \Delta_R, \mu_c) \rightarrow_{R^*} \sigma_{\text{next}} \\
 \hline
 [\text{R2}] \frac{}{(\Delta, \mu) \rightarrow_{R^*} \sigma_{\text{next}}}
 \end{array}$$

$$\begin{array}{c}
 \{I, 0, \rho^0, [], []\} = \text{ar}_0 \\
 > \text{ar}_0 = \Delta_0 \quad [] = \mu_0 \\
 (\Delta_0, \mu_0) = \sigma_0 \quad \text{Bc} = [\text{block}] \text{ ds cs} \\
 \langle \text{ds}, \sigma_0 \rangle \rightarrow ([\text{void}], (\text{ar}_1 + \Delta', \mu_1)) \\
 \text{ar}_1 = \{\text{id}, \text{lc}, \text{fr}, \text{cnt}, \text{v}\} \\
 \{\text{id}, \text{lc}, \text{fr}, \text{cs}, \text{v}\} = \text{ar}_2 \\
 > \text{ar}_2 = \Delta_2 \quad (\Delta_2, \mu_1) = \sigma_2 \\
 \sigma_2 \rightarrow_{R^*} (\Delta_F, \mu_F) \quad \# \Delta_F = 1 \\
 \hline
 [\text{P1}] \frac{}{\langle [\text{prog}] I \text{ Bc} \rangle \rightarrow_P ([\text{void}], (\Delta_F, \mu_F))}
 \end{array}$$

Osservazioni

- \rightarrow_{R^*} e \rightarrow_P : Il sistema per l'iteratore di Continuzione e quello per la semantica del Programma di Small21
- $\text{ar} + \Delta$: indica stack con ar come top record e Δ come coda;
- $\text{c} + \text{cnt}$: sta per SeqC $\text{c} \text{ cnt}$, quando la continuazione, cnt , sia una sequenza di comandi;

Small21: inline block

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

BlockS ::= [blockS] Decl Stm

Sistema Y:

$$[Y32] \frac{\langle ds, Y_\rho \rangle \rightarrow_Y ([void], Y'_\rho) \quad \langle sts, Y'_\rho \rangle \rightarrow_Y ([void], Y''_\rho)}{\langle [blockS] ds sts, Y_\rho \rangle \rightarrow_Y ([void], Y_\rho)}$$

Gestione Errori di Tipo:

$$[E51] \frac{\langle ds, Y_\rho \rangle \rightarrow_Y ([terr], Y'_\rho)}{\langle [blockS] ds sts, Y_\rho \rangle \rightarrow_Y ([terr], Y'_\rho)} \quad [E52] \frac{\langle ds, Y_\rho \rangle \rightarrow_Y ([void], Y'_\rho) \quad \langle sts, Y'_\rho \rangle \rightarrow_Y ([terr], Y''_\rho)}{\langle [blockS] ds sts, Y_\rho \rangle \rightarrow_Y ([terr], Y''_\rho)}$$

Regole di Inferenza SEM_{CHD}

$$[S13] \frac{\begin{array}{l} \sigma = (ar + \Delta, \mu) \quad ar = \{h, l, f, c, r\} \\ \{g(h), 1, [], [], []\} = ar_{top} \\ ar_{top} + ar + \Delta = \Delta_1 \\ (\Delta_1, \mu) = \sigma_1 \\ \langle ds, \sigma_1 \rangle \rightarrow ([void], (ar_2 + \Delta_2, \mu_2)) \\ ar_2 = \{id, lc, fr, cnt, v\} \\ \{id, lc, fr, sts, v\} = ar_3 \\ (ar_3 + \Delta_2, \mu_2) \rightarrow_{R^*} (ar'_3 + \Delta'_2, \mu_3) \\ (\Delta'_2, \mu_3) = \sigma_F \end{array}}{\langle [blockS] ds sts, \sigma \rangle \rightarrow ([void], \sigma_F)}$$

Osservazioni

- $ar + \Delta$: indica l'estensione dello stack Δ con il record ar posto come top del nuovo stack;
- $g(h)$: Avremmo dovuto scrivere 2 distinte regole per i possibili valori di $g(h)$: $g(Idle) = [ipb]$, $g([ipb]) = [ipb]$, per ogni altro valore h , $g[h] = []$.

Small21. Proc-Fun block: Call

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

Stm ::= [call] Ide Avars

Sistema Y:

$$[Y25] \frac{\begin{array}{l} Y_\rho(\text{ide}) = [\text{abs}] [\text{void}] [::] t' \\ \langle \text{aps}, Y_\rho \rangle \rightarrow_Y (\text{ta}, Y_\rho) \\ t' = \text{ta} \end{array}}{\langle [\text{call}] \text{ ide aps}, Y_\rho \rangle \rightarrow_Y (\text{void}, Y_\rho)}$$

Gestione Errori di Tipo:

$$[E45] \frac{Y_\rho(\text{ide}) \neq [\text{abs}] [\text{void}] [::] t'}{\langle [\text{call}] \text{ ide aps}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E46] \frac{Y_\rho(\text{ide}) = \perp}{\langle [\text{call}] \text{ ide aps}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E47] \frac{\begin{array}{l} Y_\rho(\text{ide}) = [\text{abs}] [\text{void}] [::] t' \\ \langle \text{aps}, Y_\rho \rangle \rightarrow_Y (\text{ta}, Y_\rho) \\ t' \neq \text{ta} \end{array}}{\langle [\text{call}] \text{ ide aps}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

Osservazioni

- **Controllo Differito:** Il controllo $t' = \text{ta}$ è nelle regole di trasmissione dei parametri (vedi sistema \rightarrow_{TR1});

Small21. Proc-Fun block: Call - 2

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

Stm ::= [call] Ide Apars

Regole di Inferenza SEM_{CMD}

$$\begin{array}{l} \sigma = (\Delta, \mu) \quad \Delta(\text{ide}) = (t_r, v_r) \\ v_r = \star \text{ide}, t, \text{fps}, \text{dcl}, \text{sts}, k \star \\ [\text{abs}]t_r[::]aa = t \quad t_r = [\text{void}] \\ \text{ar} = \{\text{ide}, \#\Delta - k + 1, [], [], []\} \\ \Delta_c = \text{ar} + \Delta \\ \langle \text{fps} \triangleleft \text{aps}, (\Delta, \Delta_c, \mu) \rangle \rightarrow_{\text{TR1}} (\sigma_r, \text{epi}_r) \\ \langle \text{dcl}, \sigma_r \rangle \rightarrow ([\text{void}], (\text{ar}_2 + \Delta_2, \mu_2)) \\ \text{ar}_2 = \{\text{ide}, \text{lc}, \text{fr}, \text{cnt}, v\} \\ \{\text{ide}, \text{lc}, \text{fr}, \text{sts}, v\} = \text{ar}_3 \\ [\text{S14}] \frac{(\text{ar}_3 + \Delta_2, \mu_2) \rightarrow_{R^*} (\text{ar}'_3 + \Delta'_2, \mu_3)}{\langle [\text{call}] \text{ide apars}, \sigma \rangle \rightarrow ([\text{void}], (\Delta'_2, \mu_3))} \end{array}$$

Osservazioni

- \rightarrow_{TR1} : Introduce una nuova inferenza per trattare la trasmissione di parametri;
- epi_r : Le forme di trasmissione attualmente in Small21 non richiedono un epilogo, pertanto il valore di epi_r è irrilevante e la regola di call non provvede ad avviare alcun epilogo prima di rimuovere il corrente ar top ed uscire dal blocco procedura.
- Deallocazione di Memoria: All'uscita di un blocco la maggior parte dei Linguaggi provvedono a Deallocare la memoria allocata o comunque acceduta dal blocco e non più accessibile dal Programma. AM21 ha operazioni per farlo, ma la semantica dinamica data non provvede ad alcuna forma di deallocazione.

Small21. Proc-Fun block: Apply

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

Exp ::= [apply] Ide Apars

Sistema Y:

$$[Y26] \frac{\begin{array}{l} Y_\rho(\text{ide}) = [\text{abs}] \text{t}[:,:] \text{t}' \\ \langle \text{aps}, Y_\rho \rangle \rightarrow_Y (\text{ta}, Y_\rho) \\ \text{t} \in \text{Simple} \quad \text{t}' = \text{ta} \end{array}}{\langle [\text{apply}] \text{ide apars}, Y_\rho \rangle \rightarrow_Y (\text{t}, Y_\rho)}$$

Gestione Errori di Tipo:

$$[E32] \frac{Y_\rho(\text{ide}) \neq [\text{abs}] \text{t}[:,:] \text{t}'}{\langle [\text{apply}] \text{ide apars}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E33] \frac{Y_\rho(\text{ide}) = \perp}{\langle [\text{apply}] \text{ide apars}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E34] \frac{\begin{array}{l} Y_\rho(\text{ide}) = [\text{abs}] \text{t}[:,:] \text{t}' \\ \langle \text{aps}, Y_\rho \rangle \rightarrow_Y (\text{ta}, Y_\rho) \\ \text{t}' \neq \text{ta} \end{array}}{\langle [\text{apply}] \text{ide apars}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[E34.1] \frac{\begin{array}{l} Y_\rho(\text{ide}) = [\text{abs}] \text{t}[:,:] \text{t}' \\ \langle \text{aps}, Y_\rho \rangle \rightarrow_Y (\text{ta}, Y_\rho) \\ \text{t}' \notin \text{Simple} \end{array}}{\langle [\text{apply}] \text{ide apars}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

Osservazioni

- **Controllo Differito:** Il controllo $\text{t}' = \text{ta}$ è nelle regole di trasmissione dei parametri (vedi sistema \rightarrow_{TR1});

Small21. Proc-Fun block: Apply - 2

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

Exp ::= [Apply] Ide Apars

Regole di Inferenza SEM_{EXP}

$$\begin{array}{c} \sigma = (\Delta, \mu) \quad \Delta(\text{ide}) = (t_r, c_r) \\ c_r = \star \text{ide}, t, \text{fps}, \text{dcl}, \text{sts}, k \star \\ [\text{abs}]t_r[::]aa = t \quad t_r \in \text{Simple} \\ \text{ar} = \{\text{ide}, \#\Delta - k + 1, [], [], []\} \\ \text{ar} + \Delta = \Delta_c \\ \langle \text{fps} \triangleleft \text{aps}, (\Delta, \Delta_c, \mu) \rangle \rightarrow_{\text{TR1}} (\sigma_r, \text{epi}_r) \\ \langle \text{dcl}, \sigma_r \rangle \rightarrow ([\text{void}], (\text{ar}_2 + \Delta_2, \mu_2)) \\ \text{ar}_2 = \{\text{ide}, \text{lc}, \text{fr}, \text{cnt}, \text{v}\} \\ \{\text{ide}, \text{lc}, \text{fr}, \text{sts}, \text{v}\} = \text{ar}_3 \\ (\text{ar}_3 + \Delta_2, \mu_2) \rightarrow_{\text{R}^*} (\text{ar}'_3 + \Delta'_2, \mu_3) \\ \text{ar}'_3 = \{\text{ide}, \text{lc}, \text{fr}, \text{cnt}, \text{v}_r\} \\ \text{[S15]} \frac{}{\langle [\text{apply}] \text{ide aps}, \sigma \rangle \rightarrow [t_r, v_r, (\Delta'_2, \mu_3)]} \end{array}$$

Osservazioni

- \rightarrow_{TR1} : Introduce una nuova inferenza per trattare la trasmissione di parametri;
- epi_r : Le forme di trasmissione attualmente in Small21 non richiedono un epilogo (ovvero una lista di parametri per l'epilogo della trasmissione), pertanto il valore di epi_r è irrilevante e la regola di call non provvede ad avviare alcun epilogo prima di rimuovere il corrente ar top ed uscire dal blocco procedura.
- Deallocazione di Memoria: All'uscita di un blocco la maggior parte dei Linguaggi provvedono a Deallocare la memoria allocata o comunque acceduta dal blocco e non più accessibile dal Programma. AM21 ha operazioni per farlo, ma la semantica dinamica data npn provvede ad alcuna forma di deallocazione.
- call ed apply: hanno semantica dinamica apparentemente identica: Il confronto tra le regole dice questo. In effetti, anche apply opera su una procedura (funzionale, o una funzione procedurale). Tuttavia, le 2 regole hanno comportamenti diversi nel tipo t e nel valore calcolato, ovvero l'ultimo componente dell'Activation Record con cui le due invocazione sono gestite (vedi comportamento del comando return).

Small21: Trasmissione Parametri

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

$FPars ::= [fp] PPF\ Type\ Ide \mid [emptyFP]$

$PPF ::= [value] \mid [ref]$

$APars ::= [ap] Exp \mid [emptyAP]$

Sistema Y:

$$\begin{array}{c}
 \text{fps} = [fp][value] t\ ide \\
 Y_\rho |_0(ide) = \perp \\
 \text{aps} = [ap] exp \\
 \langle exp, Y_\rho \rangle \rightarrow_Y (ta, Y_\rho) \\
 t = ta \quad t \in Simple
 \end{array}
 \quad
 \begin{array}{c}
 \text{fps} = [fp][ref] t\ ide \\
 Y_\rho |_0(ide) = \perp \\
 \text{aps} = [ap] exp \\
 \langle exp, Y_\rho \rangle \rightarrow_{DY} ([mut] ta, Y_\rho) \\
 t = ta \quad t \in Simple
 \end{array}
 \quad
 \begin{array}{c}
 \text{fps} = [emptyFp] \\
 \text{aps} = [emptyAp]
 \end{array}$$

$$[Y35] \frac{\dots}{\langle \text{fps} \triangleleft \text{aps}, Y_\rho \rangle \rightarrow_Y ([void], Y_\rho)} \quad
 [Y36] \frac{\dots}{\langle \text{fps} \triangleleft \text{aps}, Y_\rho \rangle \rightarrow_Y ([void], Y_\rho)} \quad
 [Y37] \frac{\dots}{\langle \text{fps} \triangleleft \text{aps}, Y_\rho \rangle \rightarrow_Y ([void], Y_\rho)}$$

Gestione Errori di Tipo:

$$[E59] \frac{\dots}{\langle \text{fps} \triangleleft \text{aps}, Y_\rho \rangle \rightarrow_Y ([terr], Y_\rho)} \quad
 [E60] \frac{\dots}{\langle \text{fps} \triangleleft \text{aps}, Y_\rho \rangle \rightarrow_Y ([terr], Y_\rho)}$$

$$[E61] \frac{Y_\rho |_0(ide) \neq \perp}{\langle \text{fps} \triangleleft \text{aps}, Y_\rho \rangle \rightarrow_Y ([void], Y_\rho)} \quad
 [E61.1] \frac{t \neq ta}{\langle \text{fps} \triangleleft \text{aps}, Y_\rho \rangle \rightarrow_Y ([void], Y_\rho)} \quad
 [E61.2] \frac{t \notin Simple}{\langle \text{fps} \triangleleft \text{aps}, Y_\rho \rangle \rightarrow_Y ([void], Y_\rho)}$$

$$[E62] \frac{\text{fps} = [emptyFp] \quad \text{aps} \neq [emptyAp]}{\langle \text{fps} \triangleleft \text{aps}, Y_\rho \rangle \rightarrow_Y ([void], Y_\rho)} \quad
 [E63] \frac{\text{fps} \neq [emptyFp] \quad \text{aps} = [emptyAp]}{\langle \text{fps} \triangleleft \text{aps}, Y_\rho \rangle \rightarrow_Y ([void], Y_\rho)}$$

Osservazioni

- **Compatibilità:** degli attuali rispetto alla forma di trasmissione usata e al binding da produrre. Se le 2 regole sono applicabili allora la trasmissione è corretta ovvero ha tipo [void].
- Quando la trasmissione è corretta rispetto ai tipi e alla compatibilità, allora possiamo applicarla durante l'invocazione di procedure e funzioni, utilizzando l'inferenza \rightarrow_{TR1} .

Small21: Trasmissione Parametri

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

$\text{FPars} ::= [\text{fp}] \text{ PPF Type Ide} \mid [\text{emptyFP}]$

$\text{PPF} ::= [\text{value}] \mid [\text{ref}]$

$\text{APars} ::= [\text{ap}] \text{ Exp} \mid [\text{emptyAP}]$

Regole di Inferenza SEM_{CMD}

$$[S16] \frac{\begin{array}{l} \text{fps} = [\text{fp}][\text{value}] \text{ t I} \\ \text{aps} = [\text{ap}] \text{ exp} \\ \langle \text{exp}, (\Delta, \mu) \rangle \rightarrow \llbracket \text{ta}, \text{va}, (\Delta_1, \mu_1) \rrbracket \\ \text{t} \in \text{Simple} \quad \text{t} = \text{ta} \quad \Delta_c |_0(\text{I}) = \perp \\ \quad \triangleright \langle \mu_1, \text{t}, \text{I} \rangle = (\text{loc}_t, \mu_2) \\ \quad \quad \mu_2[\text{loc}_t \leftarrow \text{va}] = \mu_3 \\ \quad \quad \llbracket \text{I} / \llbracket \text{Mut} \rrbracket \text{t}, \text{loc}_t \rrbracket \otimes \Delta_c = \Delta_c^F \\ \quad \quad (\Delta_c^F, \mu_3) = \sigma_r \quad \llbracket \rrbracket = \text{epi}_r \end{array}}{\langle \text{fps} \triangleleft \text{aps}, (\Delta, \Delta_c, \mu) \rangle \rightarrow_{\text{TR1}} (\sigma_r, \text{epi}_r)}$$

$$[S17] \frac{\begin{array}{l} \text{fps} = [\text{fp}][\text{Ref}] \text{ t I} \\ \text{aps} = [\text{ap}] \text{ exp} \\ \langle \text{exp}, (\Delta, \mu) \rangle \rightarrow_{\text{DEN}} \llbracket \llbracket \text{Mut} \rrbracket \text{ta}, \text{loca}, (\Delta_1, \mu_1) \rrbracket \\ \text{t} \in \text{Simple} \quad \text{t} = \text{ta} \quad \Delta_c |_0(\text{I}) = \perp \\ \quad \quad \llbracket \text{I} / \llbracket \llbracket \text{Mut} \rrbracket \text{ta}, \text{loca} \rrbracket \rrbracket \otimes \Delta_c = \Delta_c^F \\ \quad \quad (\Delta_c^F, \mu_1) = \sigma_r \quad \llbracket \rrbracket = \text{epi}_r \end{array}}{\langle \text{fps} \triangleleft \text{aps}, (\Delta, \Delta_c, \mu) \rangle \rightarrow_{\text{TR1}} (\sigma_r, \text{epi}_r)}$$

$$[S17.1] \frac{\begin{array}{l} \text{fps} = [\text{emptyFP}] \\ \text{aps} = [\text{emptyAP}] \\ (\Delta_c, \mu) = \sigma_r \quad \llbracket \rrbracket = \text{epi}_r \end{array}}{\langle \text{fps} \triangleleft \text{aps}, (\Delta, \Delta_c, \mu) \rangle \rightarrow_{\text{TR1}} (\sigma_r, \text{epi}_r)}$$

Osservazioni

- **Controllo della corrispondenza** dei tipi dei parametri data in accordo alla forma di trasmissione usata.

Small21: Return e Uscita da un Proc-Fun Block

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

Cmd ::= [return] Exp

Sistema Y:

$$[Y38] \frac{\langle e, Y_\rho \rangle \rightarrow_Y (t, Y_\rho) \quad t \in \text{Simple} \cup \{\text{void}\}}{\langle [\text{return}] e, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho)}$$

Gestione Errori di Tipo:

$$[E64] \frac{\langle e, Y_\rho \rangle \rightarrow ([\text{terr}], Y_\rho)}{\langle [\text{return}] e, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)} \quad [E65] \frac{t \notin \text{Simple} \cup \{\text{void}\}}{\langle [\text{return}] e, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

Regole di Inferenza SEM_{CMD}

$$[S18] \frac{\begin{array}{l} \langle e, \sigma \rangle \rightarrow [t_e, v_e, (\Delta_e, \mu_e)] \\ \Delta_e = \text{ar} + \Delta_r \\ \text{ar} = \{\text{id}, \text{ls}, \text{fr}, \text{cnt}, v\} \\ \text{id} \in \text{Ide} \quad \Delta_e(\text{id}) = (t_r, c_r) \\ t_r = [\text{abs}]t[::]t' \\ (t = t_e) \vee (t = [\text{void}] \wedge t_e = [\text{unit}]) \\ \{\text{id}, \text{ls}, \text{fr}, [], v_e\} = \text{ar1} \\ (\text{ar1} + \Delta_r, \mu_e) = \sigma_F \end{array}}{\langle [\text{return}] e, \sigma \rangle \rightarrow ([\text{void}], \sigma_F)}$$

Osservazioni

- Controlla corrispondenza tra tipo del valore restituito e tipo atteso dall'invocazione
- Controlla applicazione, [S18.1], in blocchi inline annidati in blocchi procedura avente.
- Controlla divieto di return in blocchi diversi da blocchi procedura o inline annidati in essi
- Applicabile anche al return da procedura dove Exp è [emptyE].

Small21: Return e Uscita da un Proc-Fun Block - 2

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

Cmd ::= [return] Exp

Come distinguere a runtime Ar di blocchi inline annidati in blocchi procedura dagli altri blocchi inline?: Return può occorrere solo nei primi (ed ovviamente nei blocchi procedura).

- AM21: Il componente Head (= Ide + {[E], [C], [IPB]} + []) degli AR ha il valore [IPB] proprio per gli inline annidati in blocchi procedura.
- [S13]: pone [IPB] nel campo Head di AR di blocchi inline annidati in blocchi procedura o in blocchi con intestazione [IPB].

Regole di Inferenza SEM_{CMD}

$$\begin{array}{l} \langle e, \sigma \rangle \rightarrow \lfloor t_e, v_e, (\Delta_e, \mu_e) \rfloor \\ \Delta_e = ar + \Delta_r \\ ar = \{ [ipb], ls, fr, cnt, v \} \\ \{ [ipb], ls, fr, [], v \} = ar1 \\ \Delta_r = ar2 + \Delta'_r \\ ar2 = \{ i, l, f, c, r \} \\ [return] v_e = stm \\ \{ i, l, f, [stm], r \} = ar2 \\ (ar1 + ar2 + \Delta'_r, \mu_e) = \sigma_F \\ \text{[S18.1]} \frac{}{\langle [return] e, \sigma \rangle \rightarrow \langle [void], \sigma_F \rangle} \end{array}$$

Osservazioni

- Controlla corrispondenza tra tipo del valore restituito e tipo atteso dall'invocazione
- Controlla applicazione, [S18.1], in blocchi inline annidati in blocchi procedura avente.
- Controlla divieto di return in blocchi diversi da blocchi procedura o inline annidati in essi
- Applicabile anche al return da procedura dove Exp è [emptyE].

Small21: Sequenza di Statements

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

$\text{Stm} ::= [\text{seqS}] \text{Stm} \text{Stm}$

Sistema Y:

$$[\text{Y39}] \frac{\langle s_1, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y'_\rho) \quad \langle s_2, Y'_\rho \rangle \rightarrow_Y ([\text{void}], Y''_\rho)}{\langle [\text{seqS}] s_1 s_2, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y''_\rho)}$$

Gestione Errori di Tipo:

$$[\text{E66}] \frac{\langle s_1, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y'_\rho)}{\langle [\text{seqS}] s_1 s_2, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y'_\rho)} \quad [\text{E67}] \frac{\langle s_1, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y'_\rho) \quad \langle s_2, Y'_\rho \rangle \rightarrow_Y ([\text{terr}], Y''_\rho)}{\langle [\text{seqC}] s_1 s_2, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y''_\rho)}$$

Regole di Inferenza SEM_{CMD}

$$[\text{S19}] \frac{\begin{array}{l} \Delta = \text{ar} + \Delta_r \\ \text{ar} = \{\text{hd}, \text{cs}, \text{fr}, \text{cnt}, \text{v}\} \\ [\text{seqC}] s_2 \text{cnt} = \text{cnt1} \\ \{\text{hd}, \text{cs}, \text{fr}, \text{cnt1}, \text{v}\} = \text{ar1} \\ \text{ar1} + \Delta_r = \Delta_2 \end{array} \quad \langle s_1, (\Delta_2, \mu) \rangle \rightarrow ([\text{void}], \sigma_F)}{\langle [\text{seqS}] s_1 s_2, (\Delta, \mu) \rangle \rightarrow ([\text{void}], \sigma_F)}$$

Osservazioni

- Sottotipo: Nell'uso degli AST, le categorie della Sintassi Astratta sono assimilate ai Tipi. Pertanto la scrittura

$\text{Cmd} ::= \dots \mid \text{Stm} \mid \dots$,

utilizzata nella Sintassi Astratta, dice che in AM21 il tipo `Stm` è sottotipo di `Cmd` e i suoi valori (ed espressioni) possono essere utilizzati ovunque sia richiesto un tipo `Cmd`.

Attenzione. In Ocaml, questo non è vero e dobbiamo usare il costruttore `UnL` per fare coersione di tipo.

Small21: Operazioni su Array

Il Sistema definisce il comportamento dei comandi sulla Macchina Astratta AM21.

$\text{Exp} ::= \text{DExp} \mid \dots$

$\text{DExp} ::= \text{Ide} [\uparrow 1] \text{Exp} \mid \dots$

Sistema Y:

$[Y15] \frac{}{\langle I [\uparrow 1] e, Y_\rho \rangle \rightarrow_{DY}}$

$[Y16] \frac{}{\langle I [\uparrow 1] e, Y_\rho \rangle \rightarrow_Y}$

Gestione Errori di Tipo:

Regole di Inferenza $\text{SEM}_{\text{EXP}}, \text{SEM}_{\text{DEXP}}$

Osservazioni

Controlli: `DynamicOutBoundCheck` e `DynamicOutBoundCheck:Failure`

Errori: E28-E31 e E28.1-E31.1