

SOMMARIO: 8 Maggio, 2020

- Sintassi Astratta: AST in Ocaml (invariata)
- Sintassi Concreta: Una CFG per Small20 (Invariata)
- Abstract Machine: AM20 - Stato (Invariato)
- Attività.
 - Semantica SOS con Controllo Dinamico dei Tipi:
 - Espressioni: Sistemi Sem_{EXP} e Sem_{DEN}
 - AM20 - Implementazione Esecutore:
 - Espressioni: Definizione Ocaml di expSem e dexpSem

Sintassi Astratta: Gli AST di Small20

Type ::= [int] | [bool] | [void]
 | [arr] Type Num | [mut] Type | [terr]
 | [abs] Type TypeSeq

TypeSeq ::= Type [x] Type | Type

Dcl ::= [const] Type Ide Exp | [var] Type Ide Exp
 | [varN] Type Ide
 | [array] Type Ide Num

Exp ::= [val] Ide | Num | Bool | Ide [↑] Exp | [-₁]Exp
 | Exp [+] Exp | Exp [-] Exp | Exp [*] Exp | Exp [div] Exp
 | Exp [==] Exp | Exp [<] Exp | Exp [>] Exp
 | [not] Exp | Exp [or] Exp | Exp [and] Exp
 | Exp [=] Exp | [emptyE]

Cmd ::= Cmd [seqC] Cmd
 | [ifE] Exp Cmd Cmd | [ifT] Exp Cmd
 | [for] Stm Exp Stm Cmd | Exp | [emptyC]

Stm ::= Dcl | Cmd | Stm [seqM] Stm | [emptyS]

Prog ::= [prog] Ide Stm

Sintassi Concreta: Una CFG per Small20

Type ::= Simple | void | Simple [Num]
Simple ::= int | bool

Dcl ::= final Simple ide = Exp | var Simple ide = Exp
| var Simple ide | Simple array ide[num]

Exp ::= Exp or ExpB1 | ExpB1
ExpB1 ::= ExpB1 and ExpB | ExpB
ExpB ::= not ExpB | truth | ExpR
ExpR ::= ExpR == ExpA2 | ExpR < ExpA2 | ExpR > ExpA2 | ExpA2
ExpA2 ::= ExpA2 + ExpA1 | ExpA2 - ExpA1 | ExpA1
ExpA1 ::= ExpA1 * ExpA | ExpA1 / ExpA | ExpA
ExpA ::= num | DExp | DExp = Exp | (Exp) | - ExpA2
DExp ::= ide | ide [ExpA2] | ϵ

Cmd ::= if (ExpB2) Cmd | OtherCmd
OtherCmd ::= if (ExpB2) OtherCmd else Cmd | NonConditionalCmd
NonConditionalCmd ::= for (Stm; Exp; Stm) Cmd | Exp | {cmds}
Cmds ::= Cmd; | Cmd; Cmds

Stm ::= Dcl | Cmd
Stms ::= Stm; | Stm; Stms

Prog ::= Program ide {Stms}

- **Memoria** Statica per variabili ed array di tipi diversi (interi e booleani)
Sequenza finita di **words** separate da "," e racchiusa in parentesi quadre:
 $[loc_1 \leftarrow Mv_1, \dots, loc_k \leftarrow Mv_k]$
 - **allocate**(μ, n): (alias, $\triangleright(\mu, n)$) alloca n words in sequenza;
 - **upd**(μ, loc, v): (alias, $\mu[loc \leftarrow v]$) modifica il contenuto di una word;
 - **getStore**(μ, loc): (alias, $\mu(loc)$) fornisce il contenuto di una word;
 - **emptyStore**(): (alias, 0^μ) crea uno store con words di contenuto indefinito.

- **Ambiente** per identificatori di costanti e di variabili (valori modificabili)
Sequenza finita di **bindings** separati da "," e racchiusa in parentesi quadre:
 $[Ide_1/Den_1, \dots, Ide_k/Den_k]$
 - **bind**(ρ, ide, den): (alias, $[ide/den] \circ \rho$) aggiunge un nuovo binding;
 - **getEnv**(ρ, ide): (alias, $\rho(ide)$) valore denotabile di un binding;
 - **emptyEnv**(): (alias, 0^ρ) crea un'ambiente senza bindings.

- **Stato**: Coppia (ρ, μ), indicante Ambiente e Memoria.
 - Activation Record (Stack di AR):
 - Unico: Si riduce al solo frame ρ (del blocco programma).
 - Altri componenti:
Assenti: Static/Dynamic chain, Return Address,...
Associati allo AST: Program Counter, Intermediate Results,...

● Transizione.

- Esecuzione di un costrutto c nello stato σ della Macchina Astratta
- La esprimiamo con:
 - $\langle c, \sigma \rangle \rightarrow \sigma'$, indicante ...
 - $\langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle$, indicante ...
 - $\langle c_1, \sigma_1 \rangle \rightarrow \langle c'_1, \sigma'_1 \rangle, \dots, \langle c_k, \sigma_k \rangle \rightarrow \langle c'_k, \sigma'_k \rangle / \langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle$,
k-premesse/1-conclusione, indicante ...

● Computazione La sequenza $\sigma_1, \dots, \sigma_k, \dots$ degli stati effettivamente calcolati dalle transizioni usate nella valutazione/esecuzione del programma.

● Notazione.

- (ρ, μ) stato con ambiente ρ (anche con apici/pedici) e memoria μ (anche con apici/pedici)
- 0^ρ crea un ambiente vuoto: Senza bindings.
- 0^μ crea una memoria vuota: Tutte le words sono indefinite e allocabili.
- \perp_{mem} memorizzabile indefinito, contenuto di word indefinita: Indica valore misleading o ingannevole
- N (anche con pedici) intero, B (anche con pedici) booleano, I (anche con pedici) identificatore, D (anche con pedici), una coppia (t, d_t) , indicante un tipo e una denotazione di tale tipo.
- $[I/D] \circ \rho$ crea un nuovo ambiente che estende ρ con il binding $[I/D]$
- $\rho(I)$ denotazione del binding di I in ρ , se esiste
- $\triangleright(\mu, n)$ 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
- $\mu[\text{loc} \leftarrow v]$ (loc deve essere già allocata) modifica il valore di loc con il memorizzabile v
- $\text{loc} \oplus k$ locazione k posizioni dopo loc .
- Introduzione o Vincolo. Posto in premessa con forma: espressione = pattern.

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 Small20 sulla Macchina Astratta AM20.

Controllo dei Tipi Dinamico: I Sistemi Sem_{EXP} e Sem_{DEN} sono Integrati con il Sistema Y (slide7, per le espressioni).

$Exp ::= [val] Ide \mid Num \mid Bool \mid Ide [\uparrow] Exp \mid \dots$

$$X1: \frac{\dots}{\langle [num] N, \sigma \rangle \rightarrow [\dots]} \quad X2: \frac{\dots}{\langle [truth] B, \sigma \rangle \rightarrow [\dots]} \quad X3: \frac{\dots}{\langle [emptyE], \sigma \rangle \rightarrow_Y [\dots]}$$

$$X4: \frac{\begin{array}{l} \sigma = (\rho, \mu) \\ \dots \\ t' \in Simple \\ t = [mut] t' \end{array}}{\langle [val] I, \sigma \rangle \rightarrow_{DEN} [\dots, loc'_t, \dots]} \quad X5: \frac{\begin{array}{l} \sigma = (\rho, \mu) \\ \rho(I) = ([mut] t, loc_t) \\ t \in Simple \\ \dots \end{array}}{\langle [val] I, \sigma \rangle \rightarrow [\dots, v_t, \dots]} \quad X6: \frac{\begin{array}{l} \sigma = (\rho, \mu) \\ \rho(I) = \dots \\ t \in Simple \end{array}}{\langle [val] I, \sigma \rangle \rightarrow [\dots]}$$

$$X7: \frac{\begin{array}{l} \rho(I) = \dots \\ \dots \rightarrow \dots \\ N \in [0..N_k - 1] \quad t \in Simple \\ loc_a \oplus N = \dots \end{array}}{\langle I [\uparrow] e, \sigma \rangle \rightarrow_{DEN} [\dots, loc_t, \dots]} \quad X8: \frac{\dots}{\langle I [\uparrow] e, \sigma \rangle \rightarrow [\dots]}$$

Notazione e Osservazioni

- \rightarrow ha termine destro una tripla indicante tipo, valore (esprimibile/denotabile), stato, racchiusa da $[\]$.
- $?$ è l'unico valore di tipo void: La sua presenza ha solo ragioni tecniche.
- \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.

Sistema Y: Regole per EXP - Parte1

$$\begin{array}{l}
 \text{Y5: } \frac{}{\langle [\text{num}] N, \sigma \rangle \rightarrow_Y ([\text{int}], Y_\rho)} \quad \text{Y6: } \frac{}{\langle [\text{truth}] B, \sigma \rangle \rightarrow_Y ([\text{bool}], Y_\rho)} \quad \text{Y7: } \frac{}{\langle [\text{emptyE}] , \sigma \rangle \rightarrow_Y ([\text{void}], Y_\rho)} \\
 \\
 \text{Y8: } \frac{Y_\rho(I) = [\text{mut}] t \quad t \in \text{Simple}}{\langle [\text{val}] I, Y_\rho \rangle \rightarrow_{DY} ([\text{mut}] t, Y_\rho)} \quad \text{Y9: } \frac{Y_\rho(I) = [\text{mut}] t \quad t \in \text{Simple}}{\langle [\text{val}] I, Y_\rho \rangle \rightarrow_Y (t, Y_\rho)} \quad \text{Y10: } \frac{Y_\rho(I) = t \quad t \in \text{Simple}}{\langle [\text{val}] I, Y_\rho \rangle \rightarrow_Y (t, Y_\rho)} \\
 \\
 \text{Y11: } \frac{\begin{array}{l} Y_\rho(I) = [\text{mut}][[\text{arr}]([\text{mut}]t)N] \\ \langle e, Y_\rho \rangle \rightarrow_Y ([\text{int}], Y_\rho) \quad t \in \text{Simple} \\ \text{InBoundDinamycCheck} \end{array}}{\langle I \uparrow e, Y_\rho \rangle \rightarrow_{DY} ([\text{mut}] t, Y_\rho)} \quad \text{Y12: } \frac{\begin{array}{l} Y_\rho(I) = [\text{mut}][[\text{arr}]([\text{mut}]t)N] \\ \langle e, Y_\rho \rangle \rightarrow_Y ([\text{int}], Y_\rho) \quad t \in \text{Simple} \\ \text{InBoundDinamycCheck} \end{array}}{\langle I \uparrow e, Y_\rho \rangle \rightarrow_Y (t, Y_\rho)}
 \end{array}$$

Gestione Errori di Tipo:

$$\begin{array}{l}
 \text{E8: } \frac{Y_\rho(I) = [\text{mut}] t \quad t \notin \text{Simple}}{\langle [\text{val}] I, Y_\rho \rangle \rightarrow_{DY} ([\text{terr}], Y_\rho)} \quad \text{E9: } \frac{Y_\rho(I) = t \quad t \neq [\text{mut}] t'}{\langle [\text{val}] I, Y_\rho \rangle \rightarrow_{DY} ([\text{terr}], Y_\rho)} \quad \text{E10: } \frac{Y_\rho(I) \neq [\text{mut}] t}{\langle [\text{val}] I, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)} \\
 \\
 \text{E11: } \frac{\begin{array}{l} Y_\rho(I) = [\text{mut}][[\text{arr}]([\text{mut}]t)N] \\ \langle e, Y_\rho \rangle \rightarrow_Y (t_e, Y_\rho) \quad t_e \neq [\text{int}] \end{array}}{\langle I \uparrow e, Y_\rho \rangle \rightarrow_{DY} ([\text{terr}], Y_\rho)} \quad \text{E12: } \frac{\begin{array}{l} Y_\rho(I) = [\text{mut}][[\text{arr}]([\text{mut}]t)N] \\ t \notin \text{Simple} \end{array}}{\langle I \uparrow e, Y_\rho \rangle \rightarrow_{DY} ([\text{terr}], Y_\rho)} \quad \text{E13: } \frac{\begin{array}{l} Y_\rho(I) = [\text{mut}][[\text{arr}]([\text{mut}]t)N] \\ \langle e, Y_\rho \rangle \rightarrow_Y ([\text{int}], Y_\rho) \\ \text{OutOfBoundDinamycCheck} \end{array}}{\langle I \uparrow e, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}
 \end{array}$$

Notazione

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

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

Espressioni con Stato: Le Transizioni SEM_{EXP} e SEM_{DEN} - S1

Il Sistema di regole SEM_{EXP} (risp. SEM_{DEN}) definisce il comportamento delle r-espressioni (risp. l-espressioni) durante la computazione dei Programmi Small20 sulla Macchina Astratta AM20.

Controllo dei Tipi Dinamico: I Sistemi SEM_{EXP} e SEM_{DEN} sono Integrati con il Sistema Y (slide7, per le espressioni).

$Exp ::= [val] Ide \mid Num \mid Bool \mid Ide [\uparrow] Exp \mid \dots$

$$X1: \frac{}{\langle [num] N, \sigma \rangle \rightarrow [[int], N, \sigma]} \quad X2: \frac{}{\langle [truth] B, \sigma \rangle \rightarrow [[bool], B, \sigma]} \quad X3: \frac{}{\langle [emptyE], \sigma \rangle \rightarrow [[void], ?, \sigma]}$$

$$X4: \frac{\begin{array}{l} \sigma = (\rho, \mu) \\ \dots \\ t' \in Simple \\ t = [mut] t' \end{array}}{\langle [val] I, \sigma \rangle \rightarrow_{DEN} [\dots, loc'_t, \dots]} \quad X5: \frac{\begin{array}{l} \sigma = (\rho, \mu) \\ \rho(I) = ([mut] t, loc_t) \\ t \in Simple \\ \dots \end{array}}{\langle [val] I, \sigma \rangle \rightarrow [\dots, v_t, \dots]} \quad X6: \frac{\begin{array}{l} \sigma = (\rho, \mu) \\ \rho(I) = \dots \\ t \in Simple \end{array}}{\langle [val] I, \sigma \rangle \rightarrow [\dots]}$$

$$X7: \frac{\begin{array}{l} \rho(I) = \dots \\ \dots \rightarrow \dots \\ N \in [0..N_k - 1] \quad t \in Simple \\ loc_a \oplus N = \dots \end{array}}{\langle I [\uparrow] e, \sigma \rangle \rightarrow_{DEN} [\dots, loc_t, \dots]} \quad X8: \frac{\dots}{\langle I [\uparrow] e, \sigma \rangle \rightarrow [\dots]}$$

Notazione e Osservazioni

- \rightarrow ha termine destro una tripla indicante tipo, valore (esprimibile/denotabile), stato, racchiusa da $[\]$.
- $?$ è l'unico valore di tipo void: La sua presenza ha solo ragioni tecniche.
- \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.

Sistema Y: Regole per EXP - Parte1

$$\begin{array}{l}
Y5: \frac{}{\langle [\text{num}] N, \sigma \rangle \rightarrow_Y ([\text{int}], Y_\rho)} \quad Y6: \frac{}{\langle [\text{truth}] B, \sigma \rangle \rightarrow_Y ([\text{bool}], Y_\rho)} \quad Y7: \frac{}{\langle [\text{emptyE}] , \sigma \rangle \rightarrow_Y ([\text{void}], Y_\rho)} \\
Y8: \frac{Y_\rho(I) = [\text{mut}] t \quad t \in \text{Simple}}{\langle [\text{val}] I, Y_\rho \rangle \rightarrow_{DY} ([\text{mut}] t, Y_\rho)} \quad Y9: \frac{Y_\rho(I) = [\text{mut}] t \quad t \in \text{Simple}}{\langle [\text{val}] I, Y_\rho \rangle \rightarrow_Y (t, Y_\rho)} \quad Y10: \frac{Y_\rho(I) = t \quad t \in \text{Simple}}{\langle [\text{val}] I, Y_\rho \rangle \rightarrow_Y (t, Y_\rho)} \\
Y11: \frac{\begin{array}{l} Y_\rho(I) = [\text{mut}][[\text{arr}]]([\text{mut}]t)N \\ \langle e, Y_\rho \rangle \rightarrow_Y ([\text{int}], Y_\rho) \quad t \in \text{Simple} \\ \text{InBoundDinamycCheck} \end{array}}{\langle I \uparrow e, Y_\rho \rangle \rightarrow_{DY} ([\text{mut}] t, Y_\rho)} \quad Y12: \frac{\begin{array}{l} Y_\rho(I) = [\text{mut}][[\text{arr}]]([\text{mut}]t)N \\ \langle e, Y_\rho \rangle \rightarrow_Y ([\text{int}], Y_\rho) \quad t \in \text{Simple} \\ \text{InBoundDinamycCheck} \end{array}}{\langle I \uparrow e, Y_\rho \rangle \rightarrow_Y (t, Y_\rho)}
\end{array}$$

Gestione Errori di Tipo:

$$\begin{array}{l}
E8: \frac{Y_\rho(I) = [\text{mut}] t \quad t \notin \text{Simple}}{\langle [\text{val}] I, Y_\rho \rangle \rightarrow_{DY} ([\text{terr}], Y_\rho)} \quad E9: \frac{Y_\rho(I) = t \quad t \neq [\text{mut}] t'}{\langle [\text{val}] I, Y_\rho \rangle \rightarrow_{DY} ([\text{terr}], Y_\rho)} \quad E10: \frac{Y_\rho(I) \neq [\text{mut}] t}{\langle [\text{val}] I, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)} \\
E11: \frac{\begin{array}{l} Y_\rho(I) = [\text{mut}][[\text{arr}]]([\text{mut}]t)N \\ \langle e, Y_\rho \rangle \rightarrow_Y (t_e, Y_\rho) \quad t_e \neq [\text{int}] \end{array}}{\langle I \uparrow e, Y_\rho \rangle \rightarrow_{DY} ([\text{terr}], Y_\rho)} \quad E12: \frac{\begin{array}{l} Y_\rho(I) = [\text{mut}][[\text{arr}]]([\text{mut}]t)N \\ t \notin \text{Simple} \end{array}}{\langle I \uparrow e, Y_\rho \rangle \rightarrow_{DY} ([\text{terr}], Y_\rho)} \quad E13: \frac{\begin{array}{l} Y_\rho(I) = [\text{mut}][[\text{arr}]]([\text{mut}]t)N \\ \langle e, Y_\rho \rangle \rightarrow_Y ([\text{int}], Y_\rho) \\ \text{OutOfBoundDinamycCheck} \end{array}}{\langle I \uparrow e, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}
\end{array}$$

Notazione

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

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

Implementazione: La Funzione di Interpretazione expSem

Dobbiamo introdurre ed implementare in OCaml una funzione che esprima il comportamento del sistema SEM_{EXP} definito per la semantica SOS delle r-Espressioni Small20

- Introduciamo una funzione che chiameremo expSem.
- expSem deve definire una trasformazione che data una espressione e ed uno Stato σ calcola la tripla (Tipo,r-Valore,Stato) prodotto usando le transizioni di SEM_{EXP} :

$$(\forall e, \sigma) \quad \text{expSem}(e, \sigma) = (t_e, v_e, \sigma_e) \quad \text{iff} \quad \langle e, \sigma \rangle \rightarrow [t_e, v_e, \sigma_e] \in \text{Sem}_{EXP}$$

- expSem ha segnatura:

$$\text{expSem} : e \rightarrow \text{State} \rightarrow \text{Type} * \text{rValue} * \text{State}$$

La presenza di premesse contenenti \rightarrow_{EXP} nelle regole di inferenza conduce a definizioni ricorsive della funzione semantica.

- expSem associa ad ogni espressione una funzione:

$$\text{State} \rightarrow \text{Type} * \text{rValue} * \text{State}$$

dove il primo componente dell'immagine è il tipo, associato, dal Sistema Y, alla espressione e i cui identificatori hanno tipo e valori come specificato dallo Stato sorgente.

Implementazione: La Funzione di Interpretazione dexpSem

Dobbiamo introdurre ed implementare in OCaml una funzione che esprima il comportamento del sistema SEM_{DEN} definito per la semantica SOS delle l-Espressioni Small20

- Introduciamo una funzione che chiameremo dexpSem.
- dexpSem definisce una trasformazione che data una espressione e ed uno Stato σ calcola la tripla (Tipo,l-Valore,Stato) prodotto usando le transizioni di SEM_{DEXP} :

$$(\forall e, \sigma) \quad \text{dexpSem}(e, \sigma) = (t_e, l_e, \sigma_e) \quad \text{iff} \quad \langle e, \sigma \rangle \rightarrow [t_e, l_e, \sigma_e] \in \text{Sem}_{DEN}$$

- dexpSem ha segnatura:
 $\text{dexpSem} : e \rightarrow \text{State} \rightarrow \text{Type} * l\text{Value} * \text{State}$

La presenza di premesse contenenti \rightarrow_{DEN} nelle regole di inferenza conduce a definizioni ricorsive della funzione semantica.

- dexpSem associa ad ogni l-espressione una funzione:
 $\text{State} \rightarrow \text{Type} * l\text{Value} * \text{State}$,
dove il primo componente dell'immagine è il tipo, associato dal Sistema Y, alla espressione e, i cui identificatori hanno tipo e valori come specificato dallo Stato sorgente.

Esercizio (1)

Regole X1, X2, X3: Completare premessa e termine dx della conclusione

Funzione Semantica expSem: Correggere il Codice e fornire Test Prima Verifica (vedi Listing allegato)

Esercizio (2)

Regola X4, X5, x6: Completare le regole per il calcolo del valore di un identificatore.

Funzioni Semantiche dexpSem/expSem: Estendere il Codice e fornire Test Prima Verifica (vedi Listing allegato)

Esercizio (3)

Regola X7: Completare regola per valore denotabile di un componente array.

Funzione Semantica dexpSem: Estendere Codice e fornire Test Prima Verifica (vedi Listing allegato)

Esercizio (4)

Regola X8: Completare regola per valore di un componente array.

Funzione Semantica expSem: Estendere Codice e fornire Test Prima Verifica (vedi Listing allegato)