ANALISI LINGUAGGIO L

Semantica Statica

1. Diamo nomi alle parti del programma $P = D_3 ; C_2 \qquad D_3 = D_1 ; D_2$
2. Costruiamo albero di sintassi astratta a cui applichiamo le definizioni semantiche 
3. Attraverso assiomi e R.I dimostriamo la correttezza statica, ovvero se vengono effettuate operazioni corrette tra tipi

Semantica dinamica

1. Diamo nomi alle parti del programma $P = D_3 ; C_2 \qquad D_3 = D_1 ; D_2$
2. Costruiamo albero di sintassi astratta a cui applichiamo le definizioni semantiche 
3. Costruiamo la sequenza di stati attraverso cui passa la macchina astratta 
4. Dimostriamo le transizioni che connettono gli stati (uso delle RI)

Estensione linguaggio

1. Comprensione del nuovo costrutto
2. Definire FI del nuovo costrutto
3. Definire la semantica statica del nuovo costrutto
4. Definire la semantica dinamica del nuovo costrutto
5. Verificare se il nuovo costrutto influisce su costrutti già esistenti e in quel caso aggiornare le regole di semantica statica e dinamica

Verifica sintattica

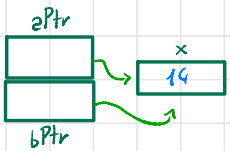
1. Diamo nomi alle parti del programma $P = D_3 ; C_2 \qquad D_3 = D_1 ; D_2$
2. Costruiamo albero di sintassi astratta 
3. Riscriviamo il codice ma con le BNF
4. Sostituiamo man mano i simboli terminali
5. Sostituiamo man mano i simboli non terminali

Ricerca identificatori liberi

1. Diamo nomi alle parti del programma $P = D_3 ; C_2 \qquad D_3 = D_1 ; D_2$
2. Costruiamo albero di sintassi astratta 
3. Applichiamo le regole e verifichiamo se restano variabili

Indirizzi di memoria Interi che contano in byte a partire dalla posizione 0x000000 e possono manipolare aritmeticamente le locazioni

Puntatori Variabili che memorizzano indirizzi di memoria



Operatori su puntatori

- & Operatore di indirizzo restituisce l'indirizzo di memoria
- * Operatore di indirezione restituisce il valore dell'oggetto a cui punta

Strutture dati Serve ad organizzare e memorizzare dati così da essere facili da manipolare

Omogenee Contiene dati dello stesso tipo

Disomogenee non contiene dati dello stesso tipo

Statica La sua dimensione non varia

Dinamica La sua dimensione varia

Lineare I dati sono organizzati come sequenza di valori

Non lineare I dati non sono organizzati come sequenza di valori

Array Strutture dati Omogenee, Statica, Lineare. Sono implementati tramite celle contigue di memoria dello stesso tipo (cioè le stesse dimensione) Si accede agli elementi tramite indice

Il nome dell'array è una variabile che contiene la locazione di memoria in cui è memorizzata la prima celle

$\rho(a, loc)$ $\sigma = (Loc, Loc_1)$

Locazione dove è memorizzato a

Locazione dove è memorizzata la prima celle di a

Per accedere ad un elemento si utilizza la coppia (Indirizzo iniziale, offset)

- Indirizzo iniziale Indirizzo della cella 0
- Offset È dato da size(celle) · Indice
  - Size(celle) determinato dal tipo degli elementi

$$a[i] = \sigma(\rho(a) + size(type(a)) \cdot i)$$

Costo di accesso costante Accedere a tutti gli elementi con una operazione (Indirizzo iniziale, offset)

Funzioni

I parametri delle funzioni possono essere considerate come variabili legate

Scoping Statico le variabili libere nel corpo delle funzione vengono legate a tempo di compilazione costruendo delle chiusure

Le dichiarazione di funzione genera nell'ambiente dinamico un legame tra il nome della funzione e una λ-astrazione che contiene tutte le informazioni necessarie ad eseguire la chiamata della funzione

CalcoloIVA(imioCosto) ⟶ let costo = imioCosto : { [(aliquota, Iz)]: return costo * aliquota/100 } ⟶ v

Associazione tra parametri formali e attuali

Chiusura: blocco che lega le variabili libere

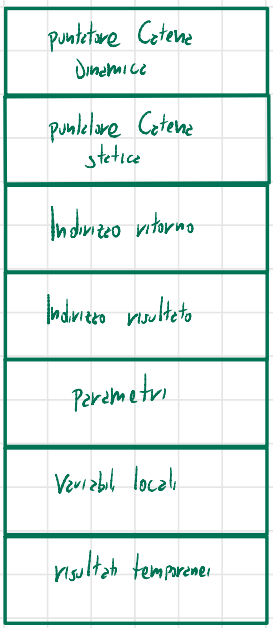
Valore restituito dalla funzione

Chiamata Funzione essendo un espressione, rappresenta un valore del tipo dichiarato nella funzione

Record di attivazione Contiene tutte le informazioni necessarie all'esecuzione del blocco o delle funzione

Catene dinamice rappresente le sequenze di chiamate e serve a garantire il corretto ordine di esecuzione

Catene statice Implementa lo scoping statico e garantisce che i nomi siano referenziati rispettando la visibilità di variabile e funzioni



Indirizzo del record di attivazione delle funzione chiamante

Implementazione scoping statico

Indirizzo dell'istruzione da eseguire al termine delle funzione/blocco corrente

Indirizzo del record di attivazione per memorizzare il risultato

Spazza riservato all'allocazione parametri formali - parametri attuali

Spazza riservato all'allocazione delle variabili locali al blocco

Spazza riservato all'allocazione delle variabili temporanee generate dal compilatore

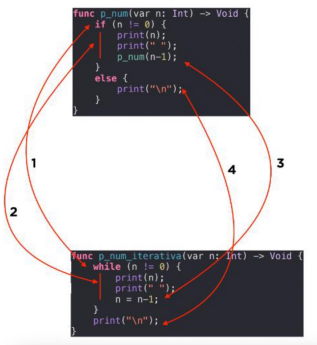
LIFO: l'ultimo elemento inserito è il primo ad essere cacciato
FIFO: il primo elemento inserito è il primo ad essere cacciato

Ricorsione Lineare: Quando c'è solo una chiamata all'interno del blocco
Ricorsione non Lineare: Quando ci sono più chiamate all'interno del blocco

Ricorsione Annidata: Quando ho una funzione come parametro all'interno della chiamata ? DA EVITARE

Accumulatori: Invertire una funzione non ricorsive in coda in una in coda. Tiene quindi conto del risultato

? È sempre possibile trasformare una ricorsione in coda in una iterazione



Esecuzione Funzione

- Catena dinamica: Garantisce il corretto ordine di esecuzione
- Catena statica: Implementa lo scoping statico e le regole di visibilità delle variabili associate

- Indirizzo di ritorno: Indirizzo dell'istruzione da eseguire per terminare il blocco corrente
- Indirizzo risultato: Indirizzo nel record di attivazione del chiamante dove porre il risultato

- Parametri e variabili locali: Sono aree di memoria riservati a questi ultimi
- Risultati temporanei: Aree di memoria per variabili temporanee

ricorsione e iterazione

	Ricorsione	Iteazione
Controllo di terminazione	condizione di ricorsione	condizione di controllo del loop
Ripetizione	ripetute chiamate della stessa funzione (chiamate ricorsive)	esecuzione ripetuta del corpo del costrutto iterativo
Convergono gradualmente alla terminazione	i passi ricorsivi riducono il problema e tendono al caso base	nel caso controllato dal contatore, ad ogni loop il contatore si avvicina al valore di terminazione
Ripetizione infinita (errori logici non fatali)	il passo ricorsivo non riduce il problema e non avvicina al caso base	la condizione di controllo del ciclo non diventa mai falsa

$$(FS1)\frac{\Delta \vdash_E E : \tau}{\Delta \vdash_C \text{return } E}$$

$$\begin{cases} \mathcal{T}(nil) = nil \\ \mathcal{T}(\text{let Id} : \tau, \text{form}) = \tau, \mathcal{T}(\text{form}) \\ \mathcal{T}(\text{var Id} : \tau, \text{form}) = \tau, \mathcal{T}(\text{form}) \end{cases}$$

$$(FS2)\frac{\text{form} : \Delta_0, \quad \Delta[\Delta_0] \vdash_C \text{var } res : \tau = E; C; \text{return } res, \quad res \notin BI(C)}{\Delta \vdash_D \text{func Id(form)} \ \to \tau \ \{C; \text{return } E\} : \ [(\text{Id}, \mathcal{T}(\text{form}) \to \tau)]}$$

$$(FS3) \quad nil : \varnothing, \quad \frac{\text{form} : \Delta_0, \text{Id} \notin \Delta_0}{\text{let Id} : \tau, \text{form} : \Delta_0[(\text{Id}, \tau)]} \quad \frac{\text{form} : \Delta_0, \text{Id} \notin \Delta_0}{\text{var Id} : \tau, \text{form} : \Delta_0[(\text{Id}, \tau loc)]}$$

$$(FS4)\frac{\Delta \vdash_{ae} ae : aet, \Delta(\text{Id}) = aet \to \tau}{\Delta \vdash_E \text{Id}(ae) : \tau}$$

$$\begin{cases} \Delta \vdash_{ae} nil : \varnothing \\ \dfrac{\Delta \vdash_E E{:}\tau, \Delta \vdash_{ae} ae{:}aet}{\Delta \vdash_{ae} E, ae{:}\tau, aet} \end{cases}$$

$$(F1)\frac{\Delta \vdash_E E : \tau}{\Delta \vdash_C \text{return } E}$$

se da delta posso associare il tipo tau ad E,
allora il comando return E è ben formato

$$(FS2)\frac{\text{form} : \Delta_0, \Delta[\Delta_0] \vdash_C \text{var } res : \tau = E; C; \text{return } res, \Delta[\Delta_0][(res, \tau)] \vdash_E E : \tau}{\Delta \vdash_D \text{func Id(form)} \ \to \tau \ \{C; \text{return } E\} : \ [(\text{Id}, \mathcal{T}(\text{form}) \to \tau)]}$$

se posso associare un ambiente statico Delta0 a `form`,
dall'ambiente corrente Delta esteso con Delta0 posso dimostrare che `var res: tau = E; C; return res` è ben formato, e da Delta esteso con Delta0 esteso con il legame (res,tau) associo ad `E` lo stesso tipo `tau` della dichiarazione della funzione
allora alla dichiarazione della funzione associo l'ambiente statico che lega il nome `Id` al tipo funzionale `T(form) -> tau`

$$(FS3)\quad nil : \emptyset, \quad \frac{\mathsf{form} : \Delta_0, \mathsf{Id} \notin \Delta_0}{\mathsf{let\ Id} : \tau, \mathsf{form} : \Delta_0[(\mathsf{Id}, \tau)]} \qquad \frac{\mathsf{form} : \Delta_0, \mathsf{Id} \notin \Delta_0}{\mathsf{var\ Id} : \tau, \mathsf{form} : \Delta_0[(\mathsf{Id}, \tau loc)]}$$

se posso associare Delta0 a `form`, e l'identificatore `Id` non compare nel dominio di Delta0
allora estendo Delta0 con il legame tra `Id` e `tau` nel caso `let`
e tra `Id` e `tau loc` nel caso `var`

$$(FS4)\frac{\Delta \vdash_{ae} ae : aet, \Delta(\mathsf{Id}) = aet \to \tau}{\Delta \vdash_{E} \mathsf{Id}(ae) : \tau}$$

se aet è il tipo dei parametri attuali e coincide con quello dei formali,
allora il tipo associato alla chiamata di funzione è `tau`

$$(FD1)\langle \mathsf{func\ Id(form)} \to T\{C; \mathsf{return}\ E\}, \rho, \sigma\rangle \to_D \langle (Id, \lambda\ \mathsf{form}\ .\ \{\rho'; C; \mathsf{return}\ E\}), \sigma\rangle$$

$$(FD2)\frac{\rho(Id) = \lambda\ \mathsf{form}\ .\ C}{\langle Id(ae), \rho, \sigma\rangle \to_e \langle \{form = ae; C\}, \rho, \sigma\rangle}$$

$$\begin{cases} \rho' = \rho_{|FI(C)-BI(form)} & scoping\ statico \\ \rho' = nil & scoping\ dinamico \end{cases}$$

$$(FD3)\frac{\langle E, \rho, \sigma\rangle \to_e \langle E', \rho, \sigma\rangle}{\langle E, ae, \rho, \sigma\rangle \to_{ae} \langle E', ae, \rho, \sigma\rangle} \qquad (FD4)\frac{\langle ae, \rho, \sigma\rangle \to_{ae} \langle ae', \rho, \sigma\rangle}{\langle v, ae, \rho, \sigma\rangle \to_{ae} \langle v, ae', \rho, \sigma\rangle}$$

$$(FD5)\frac{\langle ae, \rho, \sigma\rangle \to_{ae} \langle ae', \rho, \sigma\rangle}{\langle \mathsf{form} = ae, \rho, \sigma\rangle \to_d \langle \mathsf{form} = ae', \rho, \sigma\rangle} \qquad (FD6)\frac{av \vdash \mathsf{form} : \rho_0, \sigma_0}{\langle \mathsf{form} = av, \rho, \sigma\rangle \to_d \langle \rho_0, \sigma_0\rangle}$$

$$nil \vdash nil : \emptyset, \emptyset \qquad \frac{av \vdash \mathsf{form} : \rho, \sigma}{v, av \vdash \mathsf{let\ Id} : \tau, \mathsf{form} : \rho[(\mathsf{Id}, v)], \sigma} \qquad \frac{av \vdash \mathsf{form} : \rho, \sigma}{v, av \vdash \mathsf{var\ Id} : \tau, \mathsf{form} : \rho[(\mathsf{Id}, l_{(new)})], \sigma[(l_{(new)}, v)]}$$