MACCHINA ASTRATTA

Sintassi

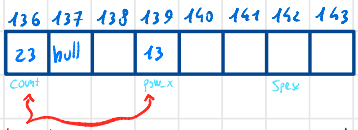
Fornisce le regole con cui si possono comporre parole accettate da un determinato linguaggio

Semantica

Assegna un significato ai simboli del linguaggio

Memoria Astratta

È un insieme di celle di memoria caratterizzate da un indirizzo univoco (locazione) e da un contenuto (valore)



locazione 136: 23 locazione 137: null

Identificatore Una stringa che contiene lettere, cifre, "_" e sono case sensitive

Ambiente È una funzione che associa ogni Identificatore ad una locazione locazione 136 = Count

Ambiente ρ: Id ⟶ Loc $\rho(count) = 136$

Memoria σ: Loc ⟶ Val $\sigma(136) = 23$ Insieme di locazioni ⟶ Insieme di celle 1 + 2 + ...

Assegnamento Scrivere un valore in una Locazione di memoria $\sigma(new\ \rho(x)) = 4$

Spese = Cont + Pow_x

$$\sigma(\rho(Spese)) = \sigma(\rho(cont)) + \sigma(\rho(pow\_x))$$

Variabili Identificatore di una cella di memoria dove il valore contenuto può variare

Costanti Identificatore di una cella di memoria dove il valore contenuto non può variare ρ: Id ⟶ Loc ∪ Val

Dichiarazioni Definiscono gli identificatori per la prima volta

Comandi Modificano la memoria

Espressioni Operazioni che non modificano la memoria

Tipi di dato Insieme di valori omogenei Int: n Double: d bool: b String: s

Tipo di dato Astratto valore + operatore

Casting Converte il tipo di dato al completare

Ambiente Statico Id ∪ Val ⟶ T ∪ TLoc (τLoc)

Overloading degli operatori effettuare operazioni tra diversi tipi

Errori Possono verificarsi a tempo di compilazione o esecuzione

Semantica Statica (Tempo di compilazione) Effettua tutti i controlli sul programma senza eseguirlo

Semantica Dinamica (Tempo di esecuzione) Definisce i passi che deve compiere la macchina astratta per eseguire il programma

Nella semantica dinamica, le espressioni non modificano mai ambiente e memoria

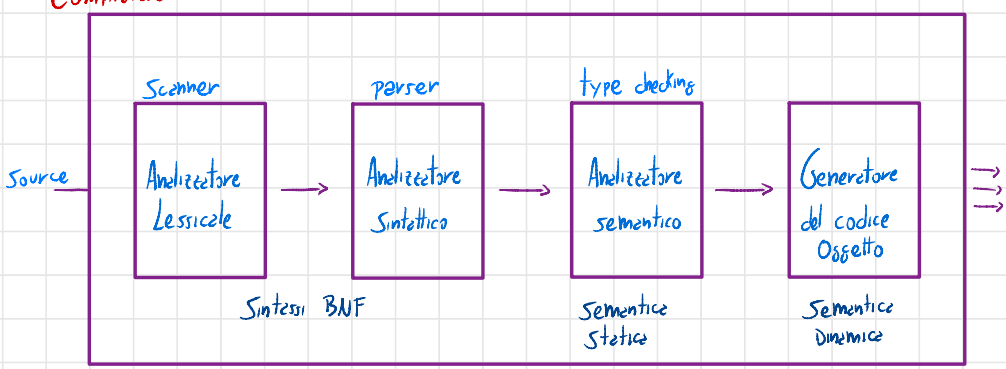
Regole di inferenza $\frac{A_1, \ldots, A_n}{B}$ Se le premesse (A_n) sono vere allora è vera la conclusione (B)

Assioma regola di inferenza sempre vera

Dimostrazione Albero con la quale si verifica se un comando è corretto

$$\frac{C_1 \quad \frac{D_n}{C_n} \quad \frac{E_{n-1}}{A_1, \ldots, A_n}}{B}$$

Compilatore



Sistema di transizioni Definisce i passi che la macchina deve eseguire e il codice oggetto che il compilatore deve generare

È una quadrupla (S, T, F, S_0) dove

- S = Insieme degli stati
- T = Insieme delle transizioni
- F = Insieme degli stati finali
- $S_0 \subseteq S$ = Stato iniziale

Visibilità degli identificatori Il campo di azione (scope) è la porzione di programma dove l'identificatore può essere utilizzato
Block-Scope parte di blocco dove è definita la variabile (anche nei blocchi annidati)

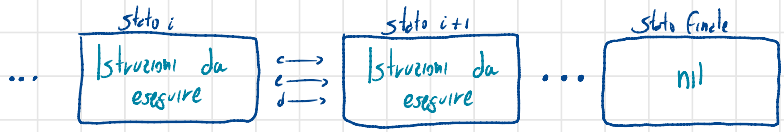
Occorrenza di legame Quando l'identificatore è in posizione di definizione e può essere utilizzato da altre stesse occorrenze nel proprio scope.

Identificatore legato Quando viene utilizzato nello stesso scope in cui è definito. Le occorrenze di legame sono identificatori legati

Identificatore libero Quando viene utilizzato al di fuori dello scope in cui è definito. Se in un programma ci sono identificatori liberi, non è corretto staticamente

Controllo sintattico del codice: Partendo dalla grammatica, ricostruire il codice grazie alle produzioni
Controllo statico del codice: Effettuare un albero di derivazione sintattico e controllare le occorrenze di legame
Trovare le occorrenze di legame: Si trova l'insieme degli identificatori liberi o legati e si effettua una sottrazione tra essi
Controllo dinamico del codice:

Macchine Astratte



esecuzione **C**: $\langle C,\rho,\sigma\rangle \longrightarrow_c \langle C',\rho',\sigma'\rangle$, $\mathbf{Exec}(C,\rho,\sigma)=\sigma' \iff \langle C,\rho,\sigma\rangle \longrightarrow_c^* \sigma'$
valutazione **E**: $\langle E,\rho,\sigma\rangle \longrightarrow_e \langle E',\rho,\sigma\rangle$, $\mathbf{Eval}(E,\rho,\sigma)=v\in Val \iff \langle E,\rho,\sigma\rangle \longrightarrow_e^* v$
elaborazione **D**: $\langle D,\rho,\sigma\rangle \longrightarrow_d \langle D',\rho',\sigma'\rangle$, $\mathbf{Elab}(D,\rho,\sigma)=\langle\rho',\sigma'\rangle \iff \langle D,\rho,\sigma\rangle \longrightarrow_d^* \langle\rho',\sigma'\rangle$