

Linguaggi di Programmazione

Corso di Laurea in “I TPS”

Descrizione dei Linguaggi di programmazione

Valeria Carofiglio

(Questo materiale è una rivisitazione del materiale
prodotto da Nicola Fanizzi)


Argomenti

- Introduzione
- Sintassi e Lessico
 - Grammatiche, derivazioni, alberi, ambiguità
- Semantica
- Pragmatica
- Implementazione

Definire un Linguaggio di Programmazione

Linguaggio di programmazione

Una notazione (linguaggio artificiale)
per descrivere algoritmi e strutture dati
destinatario della comunicazione = **calcolatore**

- **Linguistica** => per poter descrivere un linguaggio:
 - Sintassi
 - Lessico
 - (Morfologia)
 - Semantica
 - Pragmatica
- 
- Studio dei
Livelli di descrizione di
un linguaggio

Definire un Linguaggio di Programmazione

Linguaggio di programmazione

Una notazione (linguaggio artificiale)
per descrivere algoritmi e strutture dati
destinatario della comunicazione = **calcolatore**

- **Linguistica** => per poter descrivere un linguaggio:
 - Sintassi
 - Lessico
 - (Morfologia)
 - Semantica
 - Pragmatica
-  Studio dei
Livelli di descrizione di
un linguaggio
- In **informatica** c'è un altro livello:
 - Implementazione

Sintassi: Una relazione tra segni

“quali sono le frasi corrette ?”

Sintassi: Una relazione tra segni

"quali sono le frasi corrette ?"

Dato un alfabeto:

I ° Stadio: Lessico

"quali sequenze di simboli
costituiscono le parole del
linguaggio ?"

•

Esempio: il caso del linguaggio naturale
(Italiano)

Alfabeto latino su 22 lettere: {A..Z }

Livello lessicale

Parole legali in italiano:
sequenze corrette di lettere

Sintassi: Relazione tra segni

"quali sono le frasi corrette ?"

Dato un alfabeto:

I ° Stadio: Lessico

"quali sequenze di simboli
costituiscono le parole del
linguaggio ?"

- Relazione tra simboli
dell'alfabeto

II ° Stadio

"quali sequenze di parole
costituiscono le frasi
del linguaggio ?"

**Esempio: il caso del linguaggio naturale
(Italiano)**

Alfabeto latino su 22 lettere: {A..Z }

Livello lessicale

Parole legali in italiano:
sequenze corrette di **lettere**



Frasi legali in I taliano:
sequenze corrette di **parole**

Livello morfologico

Semantica:

Relazione tra segni e significati

"cosa significa una frase corretta ?"

Difficile in linguaggio naturale/ Più semplice nei linguaggi artificiali

Esempio:

Il significato di un programma potrebbe essere la funzione
matematica che quel programma computa

Pragmatica:

"Come usare una frase corretta e sensata?"

Punto di vista del programmatore:

Fraasi con lo stesso significato possono essere usate da due persone in modo diverso

Contesti linguistici diversi possono richiedere l'uso di frasi diverse

Implementazione: Livello esistente solo per i LdP

“Come eseguire una frase corretta
rispettandone la semantica?”

Punto di vista del progettista (Software o del linguaggio)

Mediante quale processo le *frasi operative* del
linguaggio realizzano lo stato di cose di cui parlano
(descritto dall'implementazione del linguaggio)

Sintassi

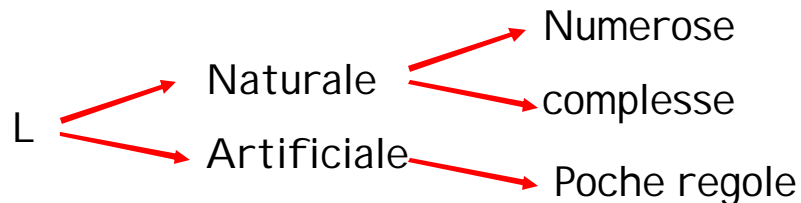
$L = \{ \text{stringhe di caratteri su un certo alfabeto} \}$

Se L è finito: possiamo elencare tutte le parole di L
(es: Il vocabolario di italiano è un volume finito)

I LdP possono essere costituiti anche da un numero infinito di parole
(lessico infinito)

Regole sintattiche

specificano quali stringhe appartengono ad L



Sintassi: gli elementi che ci servono

Elementi lessicali di un LdP (lessemi)

(unità del livello sintattico più basso (lessico) di un linguaggio)

Identificatori costanti operatori punteggiatura

Programma: frase di LdP (visto come <lessemi>)

Token (simboli)

(categorie (astrazioni) di lessemi)

Esempio

`Index := 3 * count + 5;`

Lessemi	Token
index	id
:=	assign
3	num
*	times
count	id
+	plus
5	num
;	semicolon

Sintassi

Definizione formale di un linguaggio

Per riconoscimento

L definito su alfabeto Σ



R è lo strumento per *riconoscere* le frasi di L

Se L è infinito

R non è adatto per enumerare le frasi di L

Usato per verificare la correttezza sintattica di un programma scritto in L

Per generazione



Sintassi

Definizione formale di un linguaggio

Per riconoscimento

L definito su alfabeto Σ



R è lo strumento per riconoscere le frasi di L

Se L è infinito

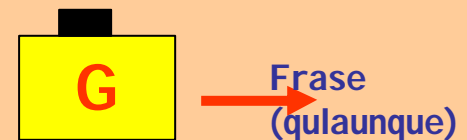
R non è adatto per enumerare le frasi di L

Usato per verificare la correttezza sintattica di un programma scritto in L

Per generazione

L definito su alfabeto Σ

Push



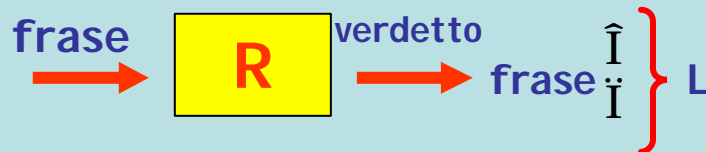
G è lo strumento per generare le frasi di L

Sintassi

Definizione formale di un linguaggio

Per riconoscimento

L definito su alfabeto Σ



R è lo strumento per riconoscere le frasi di L

Se L è infinito

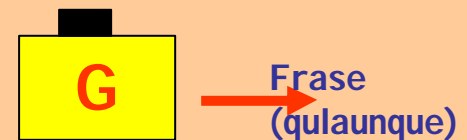
R non è adatto per enumerare le frasi di L

Usato per verificare la correttezza sintattica di un programma scritto in L

Per generazione

L definito su alfabeto Σ

Push



G è lo strumento per generare le frasi di L

R: funziona per tentativi

G: aleatorietà della definizione della frase

Sintassi

Definizione formale di un linguaggio

Per riconoscimento

L definito su alfabeto Σ



R è lo strumento per riconoscere le frasi di L

Se L è infinito

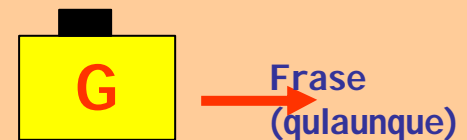
R non è adatto per enumerare le frasi di L

Usato per verificare la correttezza sintattica di un programma scritto in L

Per generazione

L definito su alfabeto Σ

Push



G è lo strumento per generare le frasi di L

R: funziona per tentativi

G: aleatorietà della definizione della frase

Importante scoperta:

Stretta connessione tra R e G:

Il meccanismo di R è basato su quello di G

Metodi formali per la definizione della sintassi

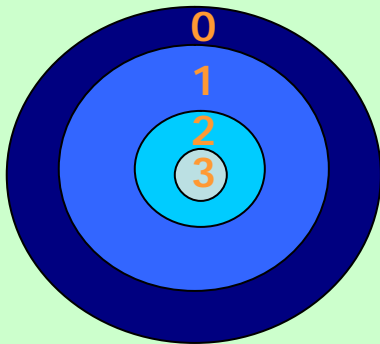
1950: J.Backus } (strumenti generativi)
N.Chomsky } Inventano la stessa notazione anche se in
contesti diversi

Metodi formali per la definizione della sintassi

1950: J.Backus } (strumenti generativi)
N.Chomsky } Inventano la stessa notazione anche se in contesti diversi

(Linguistica) Chomsky:

Specifica 4 classi di strumenti
generativi (grammatiche)



↓
4 classi di
linguaggi

Classe 2 Gram. **non-contestuali** → **sintassi**

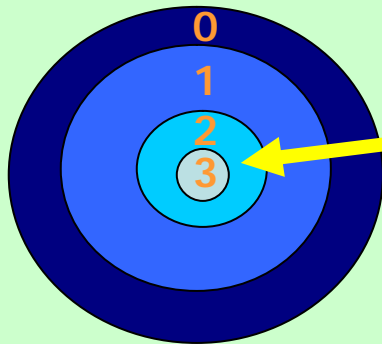
Classe 3 Gram. **Regolari** → **lessico**

Metodi formali per la definizione della sintassi

1950: J.Backus } (strumenti generativi)
N.Chomsky } Inventano la stessa notazione anche se in contesti diversi

(Linguistica) Chomsky:

Specifica 4 classi di strumenti generativi (grammatiche)



4 classi di linguaggi

Classe 2 Gram. **non-contestuali** → **sintassi**

Classe 3 Gram. **Regolari** → **lessico**

(Informatica) Backus:

Presenta Algol 60
(sintassi specificata in BNF)

BNF
(Backus Naur Form)

Quasi equivalente allo
strumento di tipo 2 di chomsky

metalinguaggio

Esempio

Vogliamo descrivere il linguaggio delle stringhe palindrome costruite a partire dai simboli a e b

$A = \{a, b\}$

$L = \{a, b, aa, bb, aaa, \dots, abba, aabaa\}$

Definizione ricorsiva:

Esempio

Vogliamo descrivere il linguaggio delle stringhe palindrome costruite a partire dai simboli a e b

$A = \{a, b\}$ $L = \{a, b, aa, bb, aaa, \dots, abba, aabaa\}$

Definizione ricorsiva:

- **a** e **b** sono palindrome (base)
- Se **s** è palindroma (passo)
 - **asa** e **bsb** sono palindrome

Esempio

Vogliamo descrivere il linguaggio delle stringhe palindrome costruite a partire dai simboli a e b

$A = \{a, b\}$ $L = \{a, b, aa, bb, aaa, \dots, abba, aabaa\}$

Definizione ricorsiva:

- **a** e **b** sono palindrome (base)
- Se **s** è palindroma (passo)
 - **asa** e **bsb** sono palindrome



a, aaa, aba, ababa,
Stringhe di lunghezza
dispari su A

Esempio

Vogliamo descrivere il linguaggio delle stringhe palindrome costruite a partire dai simboli a e b

$A = \{a, b\}$ $L = \{a, b, aa, bb, aaa, \dots, abba, aabaa\}$

Definizione ricorsiva:

- **a** e **b** sono palindrome (base)
- Se **s** è palindroma (passo)
 - **asa** e **bsb** sono palindrome
- Stringa vuota



a, aaa, aba, ababa,
Stringhe di lunghezza
dispari su A

Per stringhe di lunghezza
pari

Esempio

Vogliamo descrivere il linguaggio delle stringhe palindrome costruite a partire dai simboli a e b

$A = \{a, b\}$ $L = \{a, b, aa, bb, aaa, \dots, abba, aabaa\}$

Definizione ricorsiva:

- **a** e **b** sono palindrome (base)
- Se **s** è palindroma (passo)
 - **asa** e **bsb** sono palindrome
- Stringa vuota



a, aaa, aba, ababa,
Stringhe di lunghezza
dispari su A

Per stringhe di lunghezza
pari

Se s è una stringa palindroma
Esiste una
successione di regole sintattiche
che la costruisce

Grammatiche non contestuali (o libere da contesto)

Dato un alfabeto A (finito e non vuoto) si costruisce l'insieme (infinito) A^* di tutte le *stringhe* su A

- A^* contiene anche la *Stringa vuota* (senza simboli di A) denotata con ϵ (un linguaggio formale è un sottoinsieme di A^*)

- **Grammatica** $G = (NT, T, R, S)$

- NT insieme di simboli non terminali o variabili
- T insieme dei simboli terminali
 - (es., A oppure A^*)
- R insieme di regole di produzione $V \rightarrow W$ con
 - $V \in NT$
 - $W \in (T \cup NT)^*$
- $S \in NT$ (scopo, simbolo distintivo, iniziale o di partenza)

Esempio

Grammatica per stringhe di palindromo su
 $A=\{a,b\}$

- $G = (\{P\}, A, R, P)$ con

$R = \{$

$P \rightarrow \varepsilon$

$P \rightarrow a$

$P \rightarrow b$

$P \rightarrow aPa$

$P \rightarrow bPb\}$

Esempio2

Grammatica per descrivere le **espressioni aritmetiche** tra **identificatori** costituiti da sequenze finite di "a" e "b" (su op. unari e binari)

- $G = (\{E, I\}, \{a, b, +, *, -, (,)\}, R, E)$ con

$R = \{$

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow E - E$
5. $E \rightarrow -E$
6. $E \rightarrow (E)$
7. $I \rightarrow a$
8. $I \rightarrow b$
9. $I \rightarrow Ia$
10. $I \rightarrow Ib\}$

Esempio2

Grammatica per descrivere le **espressioni aritmetiche** tra **identificatori** costituiti da sequenze finite di "a" e "b" (su op. unari e binari)

- $G = (\{E, I\}, \{a, b, +, *, -, (,)\}, R, E)$ con

$R = \{$

1. $E \rightarrow I$

2. $E \rightarrow E + E$

3. $E \rightarrow E * E$

4. $E \rightarrow E - E$

5. $E \rightarrow -E$

6. $E \rightarrow (E)$

7. $I \rightarrow a$

8. $I \rightarrow b$

9. $I \rightarrow Ia$

10. $I \rightarrow Ib\}$

Per definire una
espressione aritmetica

Per definire un
Identificatore

Derivazione:

aspetto operativo delle grammatiche

Una **derivazione** consiste nella
ripetuta applicazione di regole di una grammatica

Grammatica per descrivere le
espressioni aritmetiche

$G = (\{E, I\}, \{a, b, +, *, -, (,)\}, R, E)$ con

$R = \{$

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow E - E$
5. $E \rightarrow -E$
6. $E \rightarrow (E)$
7. $I \rightarrow a$
8. $I \rightarrow b$
9. $I \rightarrow Ia$
10. $I \rightarrow Ib\}$

La stringa
 $ab^*(a+b)$

È corretta secondo la
grammatica data

Applicazione ripetuta

regole di produzione

\leftrightarrow

regole di riscrittura

(NOT: \vdash)

Derivazione:

aspetto operativo delle grammatiche

Una **derivazione** consiste nella
ripetuta applicazione di regole di una grammatica

Grammatica per descrivere le
espressioni aritmetiche

$G = (\{E, I\}, \{a, b, +, *, -, (,)\}, R, E)$ con

$R = \{$

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow E - E$
5. $E \rightarrow -E$
6. $E \rightarrow (E)$
7. $I \rightarrow a$
8. $I \rightarrow b$
9. $I \rightarrow Ia$
10. $I \rightarrow Ib$

La stringa $ab^*(a+b)$

È corretta secondo la grammatica

Applicazione ripetuta

$$\begin{aligned}
 & (E \xrightarrow{P_3} E * E \\
 & \quad \xrightarrow{P_1} I * E \\
 & \quad \xrightarrow{P_{10}} I b * E \\
 & \quad \xrightarrow{P_7} ab * E \\
 & \quad \xrightarrow{P_6} ab^*(E) \\
 & \quad \xrightarrow{P_2} ab^*(E + E) \\
 & \quad \xrightarrow{P_1} ab^*(I + E) \\
 & \quad \xrightarrow{P_7} ab^*(a + E) \\
 & \quad \xrightarrow{P_1} ab^*(a + I) \\
 & \quad \xrightarrow{P_8} ab^*(a + b)
 \end{aligned}$$

Derivazione:

aspetto operativo delle grammatiche

Una **derivazione** consiste nella
ripetuta applicazione di regole di una grammatica

Grammatica per descrivere le
espressioni aritmetiche

$G = (\{E, I\}, \{a, b, +, *, -, (,)\}, R, E)$ con

$R = \{$

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow E - E$
5. $E \rightarrow -E$
6. $E \rightarrow (E)$
7. $I \rightarrow a$
8. $I \rightarrow b$
9. $I \rightarrow Ia$
10. $I \rightarrow Ib$

La stringa $ab^*(a+b)$

È corretta secondo la grammatica

Applicazione ripetuta

$(E \xrightarrow{P_3} E * E$
 $\quad \xrightarrow{P_1} I * E$
 $\quad \xrightarrow{P_{10}} I b * E$
 $\quad \xrightarrow{P_7} ab * E$
 $\quad \xrightarrow{P_6} ab^*(E)$
 $\quad \xrightarrow{P_2} ab^*(E + E)$
 $\quad \xrightarrow{P_1} ab^*(I + E)$
 $\quad \xrightarrow{P_7} ab^*(a + E)$
 $\quad \xrightarrow{P_1} ab^*(a + I)$
 $\quad \xrightarrow{P_8} ab^*(a + b)$

Notazione

$E \xrightarrow{*}$
 $(ab^*(a+b))$

Derivazione

Fissata una grammatica

$$G = (NT, T, R, S)$$

Assegnate due stringhe

"v" e "w" su NT È T

- *Da v si deriva direttamente w ($v \bar{P} w$)*
 - Se w si ottiene da v sostituendo ad un simbolo non terminale V di v il corpo di una regola di produzione di R la cui testa sia V
- *Da v si deriva w ($v \bar{P}^* w$)*
 - Se esiste una sequenza finita (anche vuota) di derivazioni dirette: $v \bar{P} w_0 \bar{P} w_1 \bar{P} \dots \bar{P} w$

Linguaggio generato da una grammatica

Il linguaggio generato da una grammatica

$$G = (NT, T, R, S)$$

è costituito

dall'insieme delle stringhe derivabili dal simbolo iniziale di G

Formalmente:

$$L(G) = \{ \omega \in T^* \mid S \Rightarrow^* \omega \}$$

Definisce un linguaggio su T^*

Backus-Naur Form (BNF)

Idea di fondo:

astrazione per definire le strutture sintattiche
(Token-Simboli)

Es (C): Istruzione di assegnazione

Assign → **var** = **Expr**

L'astrazione **assign** è definita come una istanza dell'astrazione **var** seguita dal lessema = seguita da una istanza dell'astrazione **Expr**

Backus-Naur Form (BNF)

Idea di fondo:

astrazione per definire le strutture sintattiche
(Token-Simboli)

Es (C): Istruzione di assegnazione

Assign → **var** = **Expr**

L'astrazione **assign** è definita come una istanza dell'astrazione **var** seguita dal lessema = seguita da una istanza dell'astrazione **Expr**

Astrazione ≡ simbolo non terminale

Struttura sintattica/Token/lessema ≡
simbolo terminale

Grammatica ≡ insieme di regole

Regole BNF

- Una regola ha una **parte sinistra** (left-hand side: LHS) e una **parte destra** (right-hand side: RHS), che consistono di simboli ***terminali*** e *non-terminali*

LHS ::= **RHS**

Es: Istruzione di assegnazione

<Assign> ::= **<var>** = **<Expr>**

- Una regola per un simbolo non-terminale può avere più d'una parte destra, separate dal simbolo " | "

Regole BNF

- La freccia " \rightarrow " è sostituita da " $::=$ "
- I simboli non terminali sono scritti tra parentesi angolari $\langle \dots \rangle$
 - (maiuscoli nelle grammatiche di Chomsky)

Esempio3

Grammatica per descrivere le espressioni aritmetiche tra identificatori costituiti da sequenze finite di "a" e "b" (su op. + * unario e binario)

- $G = (\{E, I\}, \{a, b, +, *, -, (,)\}, R, E)$ con

$R = \{$

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow E - E$
5. $E \rightarrow -E$
6. $E \rightarrow (E)$
7. $I \rightarrow a$
8. $I \rightarrow b$
9. $I \rightarrow Ia$
10. $I \rightarrow Ib$

BNF

```
<E> ::= <I> | <E> + <E> |  
        <E>*<E>|  
        <E>-<E> |  
        -<E>|  
        (<E>)
```

Esempio4

$\langle \text{program} \rangle ::= \langle \text{stmts} \rangle$
 $\langle \text{stmts} \rangle ::= \langle \text{stmt} \rangle \mid \langle \text{stmt} \rangle ; \langle \text{stmts} \rangle$
 $\langle \text{stmt} \rangle ::= \langle \text{var} \rangle = \langle \text{expr} \rangle$
 $\langle \text{expr} \rangle ::= \langle \text{term} \rangle + \langle \text{term} \rangle \mid \langle \text{term} \rangle - \langle \text{term} \rangle$
 $\langle \text{term} \rangle ::= \langle \text{var} \rangle \mid \underline{\text{const}}$
 $\langle \text{var} \rangle ::= a \mid b \mid c \mid d$

Osserviamo che..

La derivazione di una stringa
è un processo
rigidamente sequenziale

Alcuni passi devono
essere effettuati
in un certo ordine

Altri passi possono
essere scambiati
d'ordine

La stringa $ab^*(a+b)$

È corretta secondo la grammatica

Applicazione ripetuta

$(E \Rightarrow_3 E^*E$
 $\Rightarrow_1 I^*E$
 $\Rightarrow_{10} I b^*E$
 $\Rightarrow_7 ab^*E$
 $\Rightarrow_6 ab^*(E)$
 $\Rightarrow_2 ab^*(E+E)$
 $\Rightarrow_1 ab^*(I+E)$
 $\Rightarrow_7 ab^*(a+E)$
 $\Rightarrow_1 ab^*(a+I)$
 $\Rightarrow_8 ab^*(a+b)$

Derivazioni canoniche

- Una **derivazione sinistra** (*leftmost derivation*) è una derivazione nella quale si espande il non-terminale più a sinistra in ogni forma di frase intermedia
- Analogamente si definisce la **derivazione destra** (*rightmost derivation*)

NB:

Ci sono derivazioni che non sono né destre né sinistre

Alcuni passi della derivazione possono essere scambiati

Grammatica per descrivere le
espressioni aritmetiche

$G = (\{E, I\}, \{a, b, +, *, -, (,)\}, R, E)$ con
 $R = \{$

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow E - E$
5. $E \rightarrow -E$
6. $E \rightarrow (E)$
7. $I \rightarrow a$
8. $I \rightarrow b$
9. $I \rightarrow Ia$
10. $I \rightarrow Ib\}$

App.ripetuta

$(E \rightarrow_3 E * E$
 $\rightarrow_6 E * (E)$
 $\rightarrow_2 E * (E + E)$
 $\rightarrow_1 E * (E + I)$
 $\rightarrow_8 E * (E + b)$
 $\rightarrow_1 E * (I + b)$
 $\rightarrow_7 E * (a + b)$
 $\rightarrow_1 I * (a + b)$
 $\rightarrow_{10} Ib * (a + b)$
 $\rightarrow_7 ab * (a + b)$

App.ripetuta

$(E \rightarrow_3 E * E$
 $\rightarrow_1 I * E$
 $\rightarrow_{10} Ib * E$
 $\rightarrow_7 ab * E$
 $\rightarrow_6 ab * (E)$
 $\rightarrow_2 ab * (E + E)$
 $\rightarrow_1 ab * (I + E)$
 $\rightarrow_7 ab * (a + E)$
 $\rightarrow_1 ab * (a + I)$
 $\rightarrow_8 ab * (a + b)$

Alcuni passi della derivazione possono essere scambiati

Grammatica per descrivere le espressioni aritmetiche

$G = (\{E, I\}, \{a, b, +, *, -, (,)\}, R, E)$ con

$R = \{$

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow E - E$
5. $E \rightarrow -E$
6. $E \rightarrow (E)$
7. $I \rightarrow a$
8. $I \rightarrow b$
9. $I \rightarrow Ia$
10. $I \rightarrow Ib$

Deriv. destra

App.ripetuta

$(E \xrightarrow{P_3} E * E$
 $\xrightarrow{P_6} E * (E)$
 $\xrightarrow{P_2} E * (E + E)$
 $\xrightarrow{P_1} E * (E + I)$
 $\xrightarrow{P_8} E * (E + b)$
 $\xrightarrow{P_1} E * (I + b)$
 $\xrightarrow{P_7} E * (a + b)$
 $\xrightarrow{P_1} I * (a + b)$
 $\xrightarrow{P_{10}} Ib * (a + b)$
 $\xrightarrow{P_7} ab * (a + b)$

Deriv. sinistra

App.ripetuta

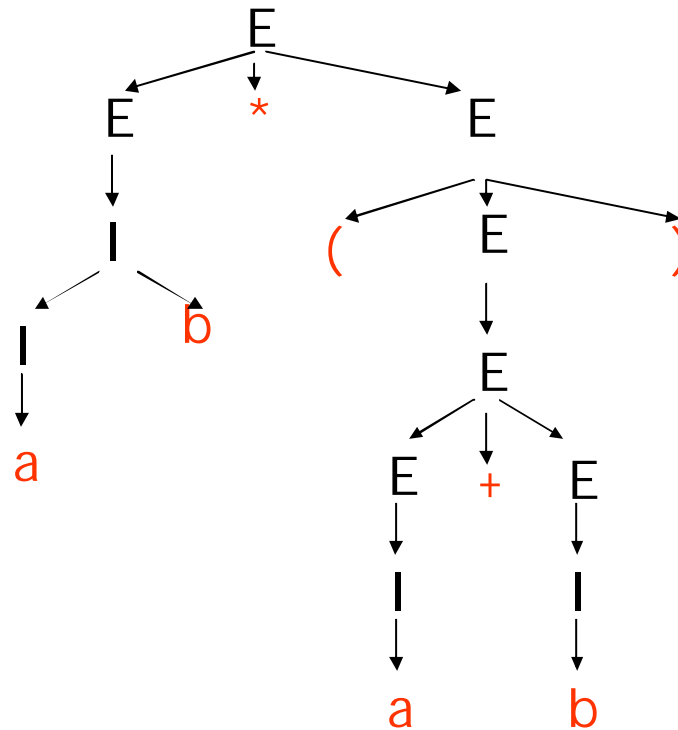
$(E \xrightarrow{P_3} E * E$
 $\xrightarrow{P_1} I * E$
 $\xrightarrow{P_{10}} Ib * E$
 $\xrightarrow{P_7} ab * E$
 $\xrightarrow{P_6} ab * (E)$
 $\xrightarrow{P_2} ab * (E + E)$
 $\xrightarrow{P_1} ab * (I + E)$
 $\xrightarrow{P_7} ab * (a + E)$
 $\xrightarrow{P_1} ab * (a + I)$
 $\xrightarrow{P_8} ab * (a + b)$

Utilità della
derivazione in forma
di albero

Derivazione in Forma di albero (un esempio)

App.ripetuta

$(E \rightarrow E * E$
 $\rightarrow E * (E)$
 $\rightarrow E * (E + E)$
 $\rightarrow E * (E + I)$
 $\rightarrow E * (E + b)$
 $\rightarrow E * (I + b)$
 $\rightarrow E * (a + b)$
 $\rightarrow I * (a + b)$
 $\rightarrow I b * (a + b)$
 $\rightarrow a b * (a + b)$

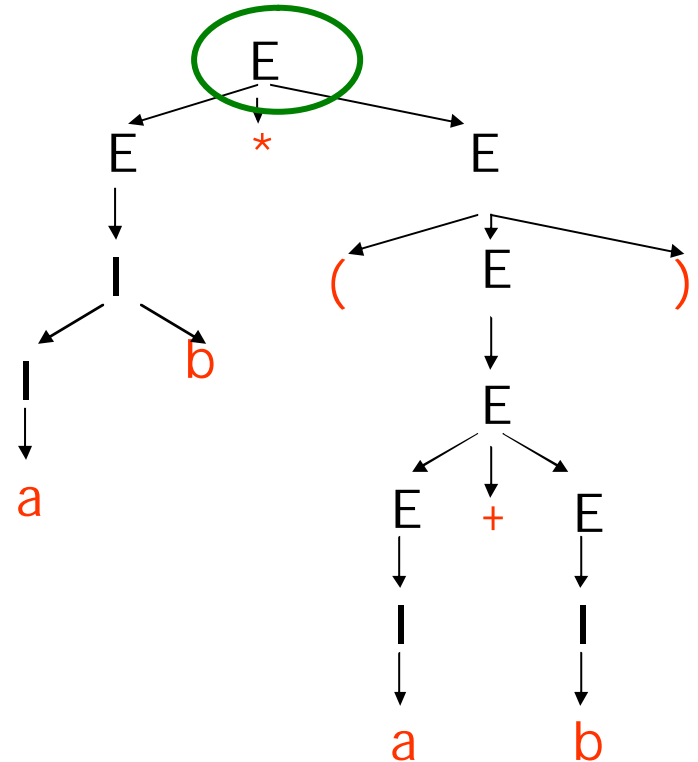


Alberi

Struttura costituita da un insieme di nodi N

Se non vuota si distinguono:

- La radice r
- Un partizionamento (S_1, S_2, \dots, S_n) dei nodi di $N \setminus \{r\}$ dove ogni S_i è strutturato ad albero

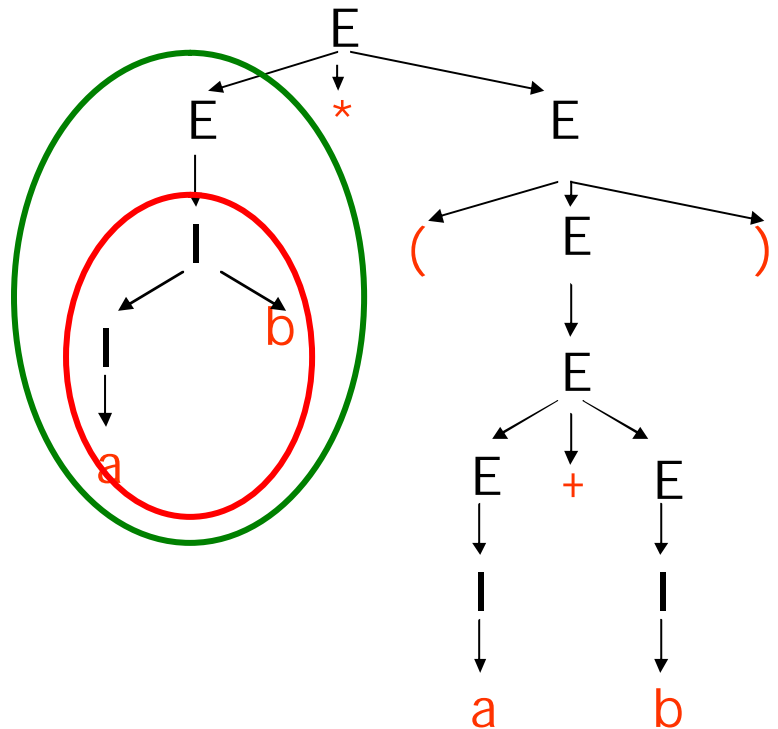


Alberi

Struttura costituita da un insieme di nodi N

Se non vuota si distinguono:

- La radice r
- Un partizionamento (S_1, S_2, \dots, S_n) dei nodi di $N \setminus \{r\}$ dove ogni S_i è strutturato ad albero



Alberi

Struttura costituita da un insieme di nodi N

Se non vuota si distinguono:

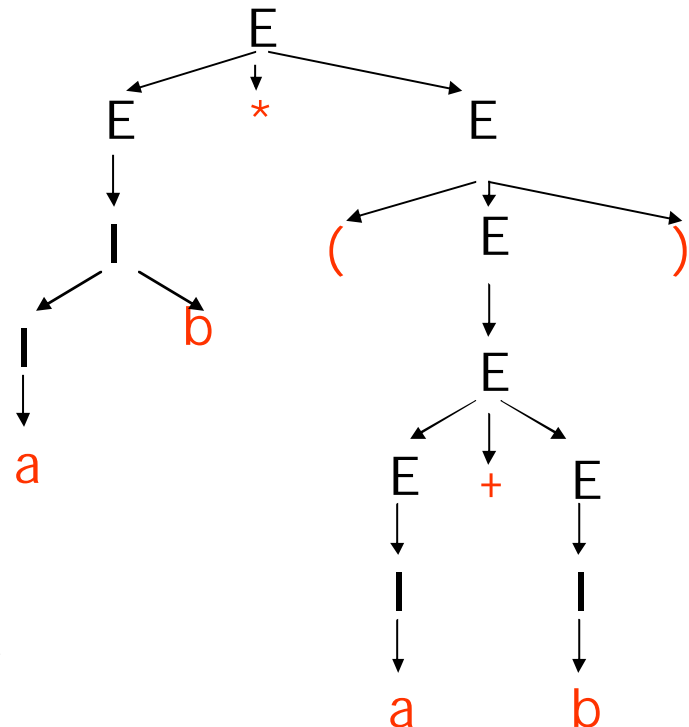
- La **radice** r
- Un partizionamento (S_1, S_2, \dots, S_n) dei nodi di $N \setminus \{r\}$ dove ogni S_i è **strutturato ad albero**

Grafo connesso $T = (N, A)$

N = insieme dei **nodi**;

A = insieme degli **archi** (n_i, n_j)

- $|N| = |A| + 1$
- un Nodo n **padre** di tutti i nodi **figli** (ordinati) m_1, \dots, m_p , tali che $(n, m_j) \in A$
- **radice** r , $\text{livello}(r) = 0$
- Se $\text{livello}(n) = i$ allora tale che $(n, m) \in A$: $\text{livello}(m) = i + 1$

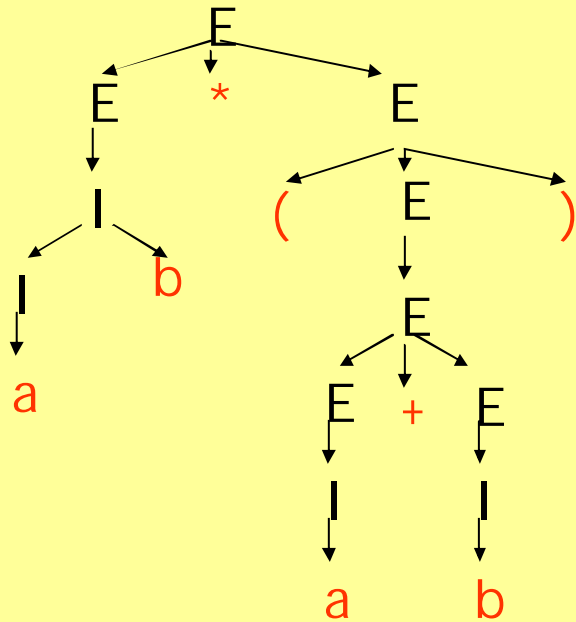


Albero di derivazione

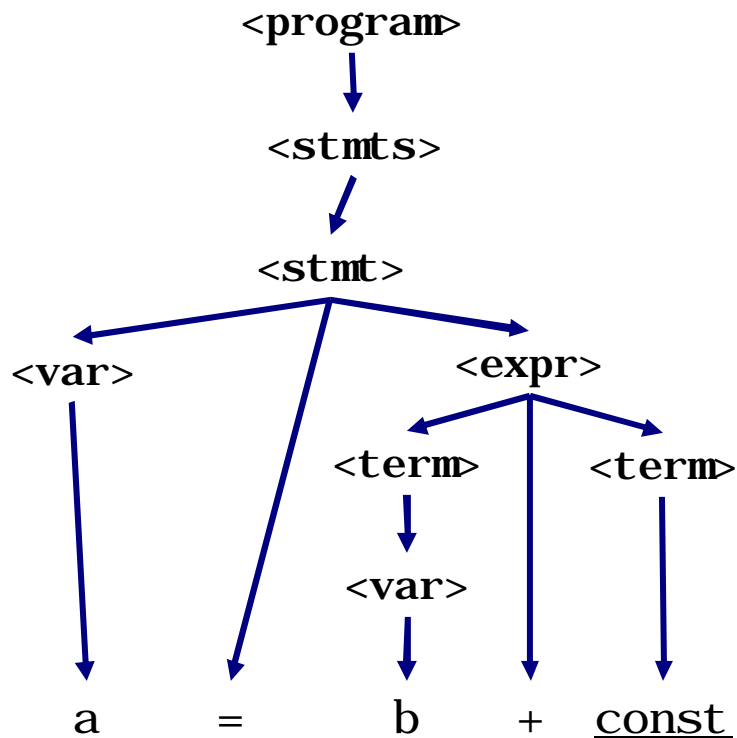
Data una grammatica $G = (NT, T, R, S)$

Rappresentazione gerarchica di una derivazione:

- I nodi sono etichettati con elementi di $T \cup NT \cup \{\epsilon\}$
 - Nodo **radice** = S
 - Nodo **interno** (con figli) è un $A \in NT$:
 - Se i figli di A sono etichettati $X_1 \dots X_n$ dove $\forall X_i \in T \cup NT$, allora deve esistere in R la regola
$$A \rightarrow X_1 \dots X_n$$
 - Se ϵ è l'unico figlio di A, allora deve esistere in R la regola
$$A \rightarrow \epsilon$$
 - Nodi **foglia** (senza figli) sono simboli di $T \cup \{\epsilon\}$



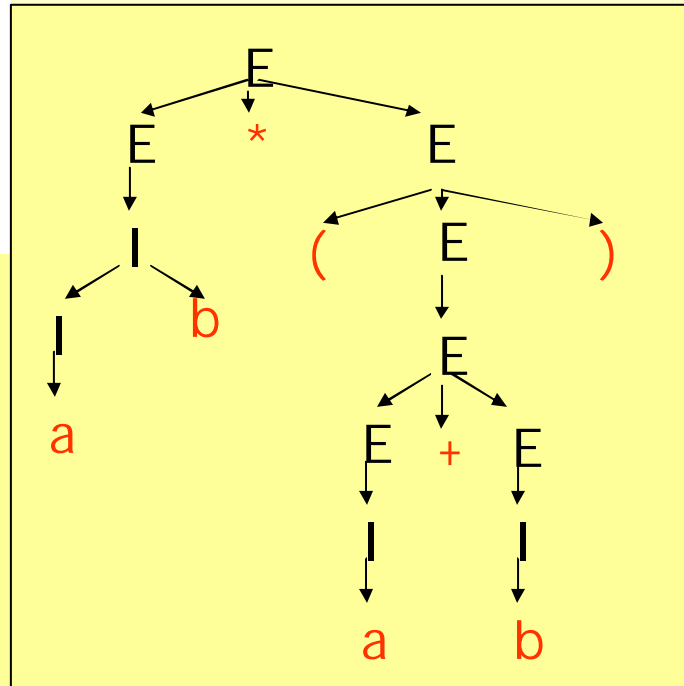
Albero di derivazione: Altro esempio



```
<program> ::= <stmts>
<stmts> ::= <stmt>
           | <stmt> ; <stmts>
<stmt> ::= <var> = <expr>
<var> ::= a | b | c | d
<expr> ::= <term> + <term>
           | <term> - <term>
<term> ::= <var> | const
```

Osserviamo che...

Albero della derivazione uguale



Deriv. destra

App.ripetuta

$(E \vdash_3 E * E$
 $\vdash_6 E * (E)$
 $\vdash_2 E * (E + E)$
 $\vdash_1 E * (E + I)$
 $\vdash_8 E * (E + b)$
 $\vdash_1 E * (I + b)$
 $\vdash_7 E * (a + b)$
 $\vdash_1 I * (a + b)$
 $\vdash_{10} I b * (a + b)$
 $\vdash_7 a b * (a + b)$

Deriv. sinistra

App.ripetuta

$(E \vdash_3 E * E$
 $\vdash_1 I * E$
 $\vdash_{10} I b * E$
 $\vdash_7 a b * E$
 $\vdash_6 a b * (E)$
 $\vdash_2 a b * (E + E)$
 $\vdash_1 a b * (I + E)$
 $\vdash_7 a b * (a + E)$
 $\vdash_1 a b * (a + I)$
 $\vdash_8 a b * (a + b)$

Osserviamo che...

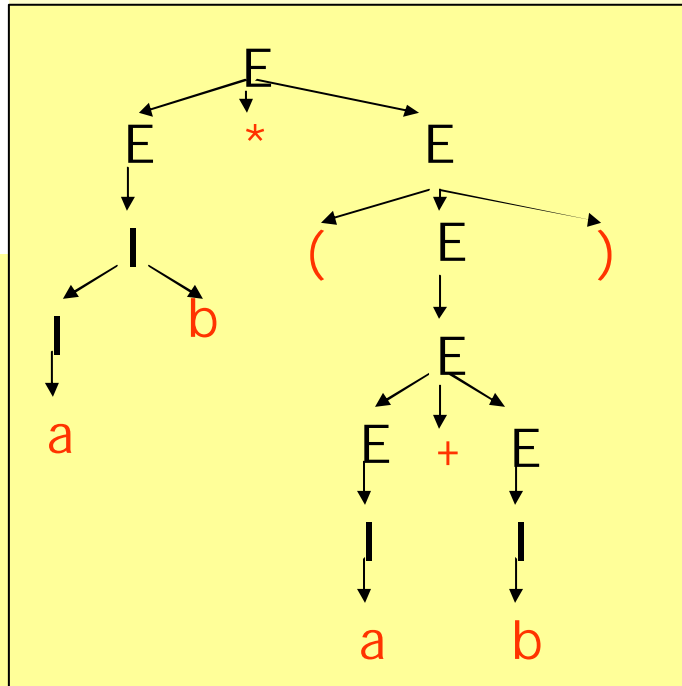
Albero della derivazione uguale

Deriv. destra

Deriv. sinistra

App.ripetuta

$(E \vdash_3 E^*E$
 $\vdash_6 E^*(E)$
 $\vdash_2 E^*(E+E)$
 $\vdash_1 E^*(E+I)$
 $\vdash_8 E^*(E+b)$
 $\vdash_1 E^*(I+b)$
 $\vdash_1 E^*(a+b)$



App.ripetuta

$(E \vdash_3 E^*E$
 $\vdash_1 I^*E$
 $\vdash_{10} Ib^*E$
 $\vdash_7 ab^*E$
 $\vdash_6 ab^*(E)$
 $\vdash_2 ab^*(E+E)$
 $\vdash_1 ab^*(a+b)$

IMPORTANTE PER L'ANALISI SINTATTICA DI UN LdP
 La struttura dell'albero (attraverso i sottoalberi) esprime la struttura logica che la grammatica assegna alla stringa

Osserviamo che...

IMPORTANTE PER L'ANALISI SINTATTICA DI UN LdP
La struttura dell'albero (attraverso i sottoalberi) esprime la struttura logica che la grammatica assegna alla stringa

Grammatica per descrivere le
espressioni aritmetiche

$G = (\{E, I\}, \{a, b, +, *, -, (,)\}, R, E)$ con

$R = \{$

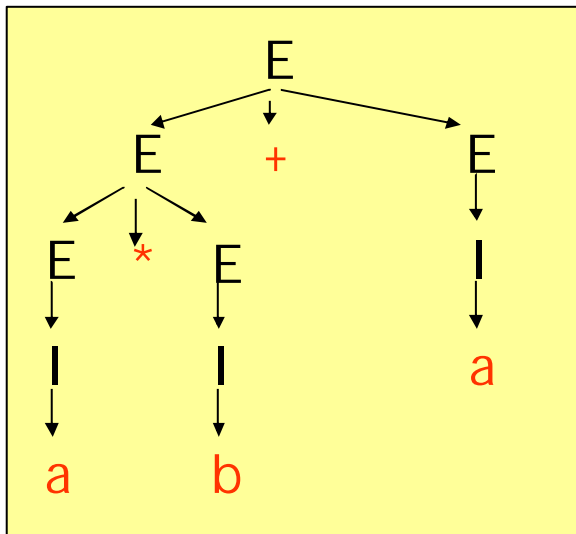
1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow E - E$
5. $E \rightarrow -E$
6. $E \rightarrow (E)$
7. $I \rightarrow a$
8. $I \rightarrow b$
9. $I \rightarrow Ia$
10. $I \rightarrow Ib\}$

App.ripetuta

$(E \rightarrow_2 E + E$
 $\rightarrow_3 E * E + E$
 $\rightarrow_1 I * E + E$
 $\rightarrow_7 a * E + E$
 $\rightarrow_1 a * I + E$
 $\rightarrow_8 a * b + E$
 $\rightarrow_1 a * b + I$
 $\rightarrow_7 a * b + a$

Osserviamo che...

IMPORTANTE PER L'ANALISI SINTATTICA DI UN LdP
La struttura dell'albero (attraverso i sottoalberi) esprime la struttura logica che la grammatica assegna alla stringa

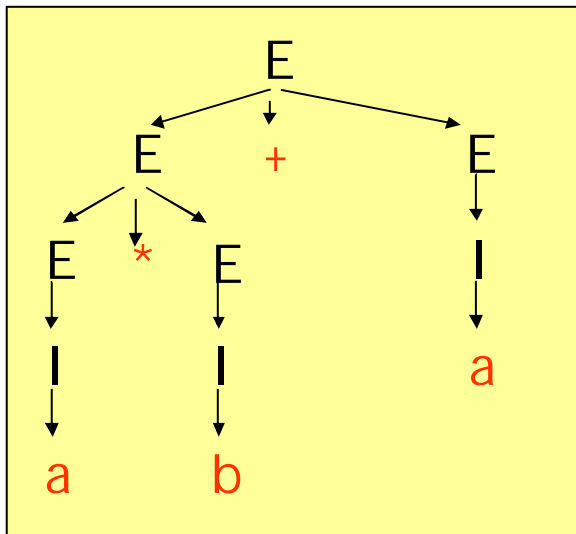


App.ripetuta

$(E \text{ } P_2 E + E$
 $P_3 E * E + E$
 $P_1 I * E + E$
 $P_7 a * E + E$
 $P_1 a * I + E$
 $P_8 a * b + E$
 $P_1 a * b + I$
 $P_7 a * b + a$

Osserviamo che...

IMPORTANTE PER L'ANALISI SINTATTICA DI UN LdP
La struttura dell'albero (attraverso i sottoalberi) esprime la struttura logica che la grammatica assegna alla stringa



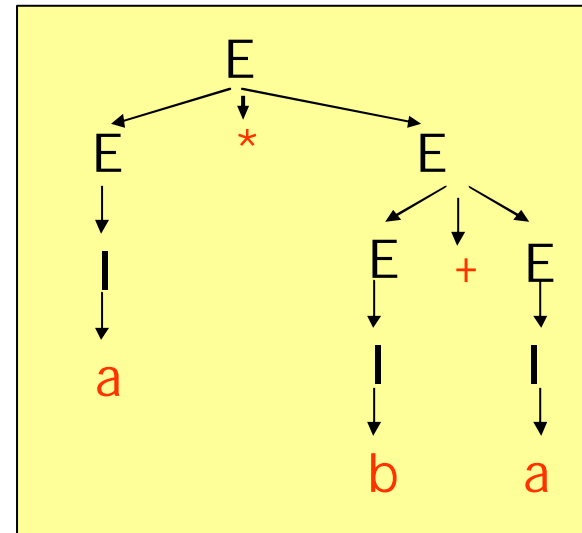
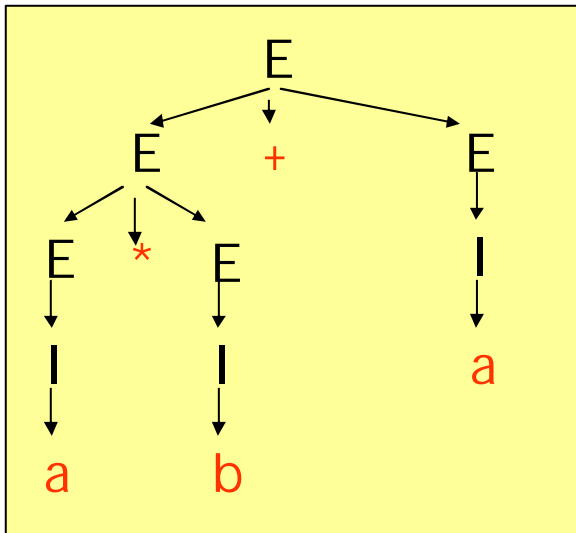
Moltiplica a per b poi somma a

App.ripetuta

$(E \ P_2 E + E$
 $P_3 E * E + E$
 $P_1 I * E + E$
 $P_7 a * E + E$
 $P_1 a * I + E$
 $P_8 a * b + E$
 $P_1 a * b + I$
 $P_7 a * b + a$

Ambiguità nelle grammatiche (esempio)

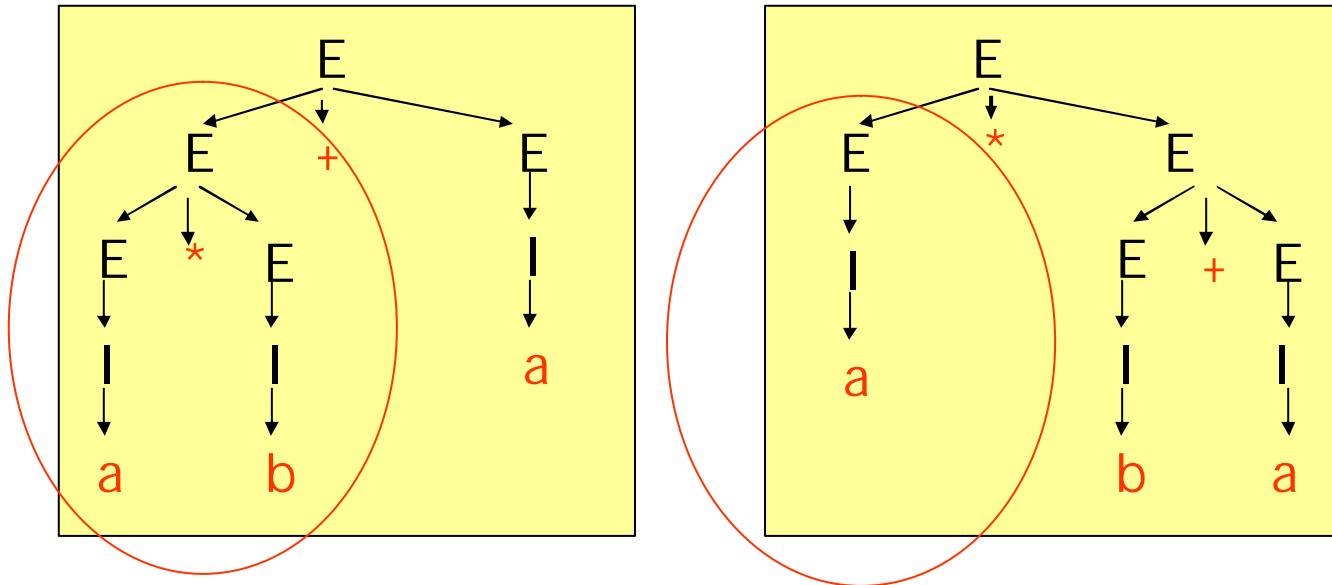
Alberi di derivazione per la stringa $a * b + a$



Due alberi diversi producono la stessa stringa

Ambiguità nelle grammatiche (esempio)

Alberi di derivazione per la stringa $a * b + a$



Due alberi diversi producono la stessa stringa

A seconda di come la derivazione è costruita
cambia la precedenza tra i due operatori aritmetici

Ambiguità nelle grammatiche

Una **grammatica** è **ambigua** se e solo se essa genera (almeno) una stringa (forma di frase) che abbia due o più alberi di derivazione distinti

NB:

ma una stringa può però avere più derivazioni distinte senza che la grammatica sia ambigua

Disambiguità nelle grammatiche (esempio)

Grammatica **AMBIGUA** per descrivere
le espressioni aritmetiche

$G = (\{E, I\}, \{a, b, +, *, -, (,)\}, R, E)$ con
 $R = \{E ::= I \mid E + E \mid E * E \mid E - E \mid$
 $\quad -E \mid (E)$
 $\quad I ::= a \mid b \mid Ia \mid Ib\}$

Grammatica **NON AMBIGUA** per
descrivere le espressioni
aritmetiche

$G = (\{E, T, A, I\}, \{a, b, +, *, -, (,)\}, R', E)$
con
 $R = \{$
 $\quad E ::= T \mid T + E \mid T - E$
 $\quad T ::= A \mid A * T \mid$
 $\quad A ::= I \mid -A \mid (E)$
 $\quad I ::= a \mid b \mid Ia \mid Ib$

Disambiguità nelle grammatiche (esempio)

Grammatica **AMBIGUA** per descrivere
le espressioni aritmetiche

$G = (\{E, I\}, \{a, b, +, *, -, (,)\}, R, E)$ con
 $R = \{E ::= I \mid E + E \mid E * E \mid E - E \mid$
 $\quad -E \mid (E)$
 $\quad I ::= a \mid b \mid Ia \mid Ib\}$

Rappresenta la nozione di
precedenza degli operatori

Grammatica **NON AMBIGUA** per
descrivere le espressioni
aritmetiche

$G = (\{E, T, A, I\}, \{a, b, +, *, -, (,)\}, R', E)$
con
 $R = \{$
 $\quad E ::= T \mid T + E \mid T - E$
 $\quad T ::= A \mid A * T \mid$
 $\quad A ::= I \mid -A \mid (E)$
 $\quad I ::= a \mid b \mid Ia \mid Ib$



Disambiguità nelle grammatiche (esempio)

Grammatica **AMBIGUA** per descrivere
le espressioni aritmetiche

$G = (\{E, I\}, \{a, b, +, *, -, (,)\}, R, E)$ con
 $R = \{E ::= I \mid E + E \mid E * E \mid E - E \mid$
 $-E \mid (E)$
 $I ::= a \mid b \mid Ia \mid Ib\}$

Rappresenta la nozione di
precedenza degli operatori

Gli operatori dello stesso
livello di precedenza vengono
associati a destra:

$$a + b + a \leftrightarrow a + (b + a)$$

Grammatica **NON AMBIGUA** per
descrivere le espressioni
aritmetiche

$G = (\{E, T, A, I\}, \{a, b, +, *, -, (,)\}, R', E)$
con
 $R = \{$
 $E ::= T \mid T + E \mid T - E$
 $T ::= A \mid A * T \mid$
 $A ::= I \mid -A \mid (E)$
 $I ::= a \mid b \mid Ia \mid Ib$



Disambiguità nelle grammatiche (esempio)

Grammatica **AMBIGUA** per descrivere
le espressioni aritmetiche

$G = (\{E, I\}, \{a, b, +, *, -, (,)\}, R, E)$ con
 $R = \{E ::= I \mid E + E \mid E * E \mid E - E \mid$
 $-E \mid (E)$
 $I ::= a \mid b \mid Ia \mid Ib\}$

Rappresenta la nozione di
precedenza degli operatori

Gli operatori dello stesso
livello di precedenza vengono
associati a destra:

$a + b + a \leftrightarrow a + (b + a)$

Grammatica **NON AMBIGUA** per
descrivere le espressioni
aritmetiche

$G = (\{E, T, A, I\}, \{a, b, +, *, -, (,)\}, R', E)$
con
 $R = \{$
 $E ::= T \mid T + E \mid T - E$
 $T ::= A \mid A * T \mid$
 $A ::= I \mid -A \mid (E)$
 $I ::= a \mid b \mid Ia \mid Ib$

Osserviamo che:

La disambiguazione della
grammatica è pagata in termini di
complessità



LdP a volte molto contorti

Caso dell'istruzione condizionale

Grammatica **ambigua** per il costrutto
"if" in Java

$\langle \text{istruzione} \rangle ::= \dots \langle \text{if-else} \rangle \mid \langle \text{altra} \rangle$

$\langle \text{altra} \rangle ::= \dots \langle \text{blocco} \rangle \mid \langle \text{vuota} \rangle \mid \langle \text{return} \rangle$

$\langle \text{if-else} \rangle ::= \underline{\text{if}} (\langle \text{espressione} \rangle) \langle \text{istruzione} \rangle \mid$
 $\quad \underline{\text{if}} (\langle \text{espressione} \rangle) \langle \text{istruzione} \rangle \underline{\text{else}} \langle \text{istruzione} \rangle$

Caso dell'istruzione condizionale

Grammatica **ambigua** per il costrutto
"if" in Java

$\langle \text{istruzione} \rangle ::= \dots \langle \text{if-else} \rangle \mid \langle \text{altra} \rangle$

$\langle \text{altra} \rangle ::= \dots \langle \text{blocco} \rangle \mid \langle \text{vuota} \rangle \mid \langle \text{return} \rangle$

$\langle \text{if-else} \rangle ::= \underline{\text{if}} (\langle \text{espressione} \rangle) \langle \text{istruzione} \rangle \mid$
 $\quad \underline{\text{if}} (\langle \text{espressione} \rangle) \langle \text{istruzione} \rangle \underline{\text{else}} \langle \text{istruzione} \rangle$

Quanti alberi per la stringa seguente ?

`if (<espressione>) if(<espressione>) <istruzione> else <istruzione>`

Caso dell'istruzione condizionale

Grammatica **ambigua** per il costrutto
"if" in Java

$\langle \text{istruzione} \rangle ::= \dots \langle \text{if-else} \rangle \mid \langle \text{altra} \rangle$

$\langle \text{altra} \rangle ::= \dots \langle \text{blocco} \rangle \mid \langle \text{vuota} \rangle \mid \langle \text{return} \rangle$

$\langle \text{if-else} \rangle ::= \underline{\text{if}} (\langle \text{espressione} \rangle) \langle \text{istruzione} \rangle \mid$
 $\quad \underline{\text{if}} (\langle \text{espressione} \rangle) \langle \text{istruzione} \rangle \underline{\text{else}} \langle \text{istruzione} \rangle$

Quanti alberi per la stringa seguente ?

if (<espressione>) if(<espressione>) <istruzione> else <istruzione>

if (<espressione>)
 if(<espressione>)
 <istruzione>
↓
else <istruzione>

if (<espressione>)
 if(<espressione>)
 <istruzione>
 ↓
 else <istruzione>

Caso dell'istruzione condizionale

Grammatica **non ambigua** per il costrutto "if" in Java

<istruzione> ::= ... <if> | <if-else> | <altra>

<altra> ::= ... <blocco> | <vuota> | <return>

<no-if-breve> ::= <altra> | <if-else2>

<if> ::= if (<espressione>) <istruzione>

<if-else> ::= if (<espressione>) <no-if-breve> else <istruzione>

<if-else2> ::= if (<espressione>) <no-if-breve> else <no-if-breve>

Il ramo else appartiene all'if più vicino

Metodi Formali per Descrivere la Sintassi

- Grammatiche e riconoscitori
- Backus-Naur Form e Grammatiche libere da contesto
 - Metodi più diffusi per descrivere la sintassi dei linguaggi di programmazione
- Extended BNF
 - Migliora la leggibilità e la scrivibilità della BNF

Extended BNF (EBNF)

Estensioni della BNF

- Opzionalità

- Le parti opzionali sono poste tra parentesi quadre []
- `<proc_call> ::= ident [(<expr_list>)]`

- Disgiunzione

- Le parti destre alternative sono poste tra parentesi tonde () e separate con barre verticali |
- `<term> ::= <term> (+|-) const`

- Ripetizione

- Ripetizioni (di 0 o più elementi) sono poste tra parentesi graffe { }
- `<ident> ::= letter {letter | digit}`

BNF e EBNF

- **BNF**

$\langle \text{expr} \rangle ::= \langle \text{expr} \rangle + \langle \text{term} \rangle$
 | $\langle \text{expr} \rangle - \langle \text{term} \rangle$
 | $\langle \text{term} \rangle$
 $\langle \text{term} \rangle ::= \langle \text{term} \rangle * \langle \text{factor} \rangle$
 | $\langle \text{term} \rangle / \langle \text{factor} \rangle$
 | $\langle \text{factor} \rangle$

- **EBNF**

$\langle \text{expr} \rangle ::= \langle \text{term} \rangle \{ (+ \mid -) \langle \text{term} \rangle \}$
 $\langle \text{term} \rangle ::= \langle \text{factor} \rangle \{ (* \mid /) \langle \text{factor} \rangle \}$

Grammatica **NON AMBIGUA** per
descrivere le espressioni
aritmetiche

$G = (\{E, T, A, I\}, \{a, b, +, *, -, (,)\}, R', E)$
con
 $R = \{$
 $E ::= T \mid T + E \mid T - E$
 $T ::= A \mid A * T \mid$
 $A ::= I \mid -A \mid (E)$
 $I ::= a \mid b \mid Ia \mid Ib$
 $\}$

Vincoli contestuali

Correttezza di una frase può dipendere dal contesto nel quale la frase si trova:

- Identificatore da dichiarare prima dell'uso
- Numero dei parametri effettivi pari al numero di parametri formali
- Espressione nella parte destra di un assegnazione di tipo compatibile con quello della variabile nella parte sinistra
- Non modificabilità della variabile che controlla l'istruzione for
- Inizializzare ogni variabile prima dell'uso
- Metodi sovrascritti da metodi con firma uguale (o compatibile)

Vincoli contestuali

Correttezza di una frase può dipendere dal contesto in cui la frase si trova:

- Identificatore da distinguere dall'altro
- Numero di parametri **NON** numero di
- Esplicitazione di una parte
- Tipo di operazione di
- Insieme di parametri
- Istogramma
- Metodo di scrittura da metodi con firma uguale (o compatibile)

si possono esprimere
con una
grammatica
Non Contestuale

Grammatiche contestuali e Vincoli contestuali

Grammatiche contestuali per descrivere i vincoli contestuali

Dovere di cronaca

Le grammatiche contestuali hanno regole di produzione del tipo:

$$uAv \rightarrow uwv \quad \text{con } u, v, w \in T^+ \text{ e } A \in N$$

Il simbolo non terminale A può essere riscritto come w solo se appare in un certo contesto (definito da u e v)

Grammatiche contestuali e Vincoli contestuali

Grammatiche contestuali per descrivere i vincoli contestuali

Dovere di cronaca

Le grammatiche contestuali hanno regole di produzione del tipo:

$$uAv \rightarrow uvw \quad \text{con } u, v, w \in T^+ \quad T \text{ È NT}$$

Il simbolo non terminale A può essere riscritto come w solo se appare in un certo contesto (definito da u e v)

Perché tali grammatiche non vengono di fatto utilizzate

- Grammatiche molto pesanti
- Per le quali non esistono tecniche automatiche di generazione di traduttori efficienti (che invece esistono per le grammatiche libere da contesto)

Vincoli contestuali e **semantica statica**

La necessità di gestire i vincoli della
sintassi contestuale
porta alla nozione di **semantica statica**

- ossia verificabile sul testo del programma sorgente
- Diversa dalla **semantica dinamica**
(che riguarda il significato del programma mentre gira)

Es. grammatiche con attributi

Grammatiche libere + vincoli ed azioni

Sintassi o semantica?

I vincoli contestuali appartengono alla sintassi

- Identificatore da dichiarare prima dell'uso
- Numero dei parametri effettivi pari al numero di parametri formali
- Espressione nella parte destra di un'assegnazione di tipo compatibile con quello della variabile nella parte sinistra
-

Tradizionalmente nei LdP si intende

- **sintassi**: tutto ciò che si descrive mediante BNF
 - **Semantica**: tutto il resto

I vincoli contestuali sono dunque vincoli semantici....

Controlli di tipo semantico

Semantica statica

vincoli contestuali determinabili al momento della compilazione

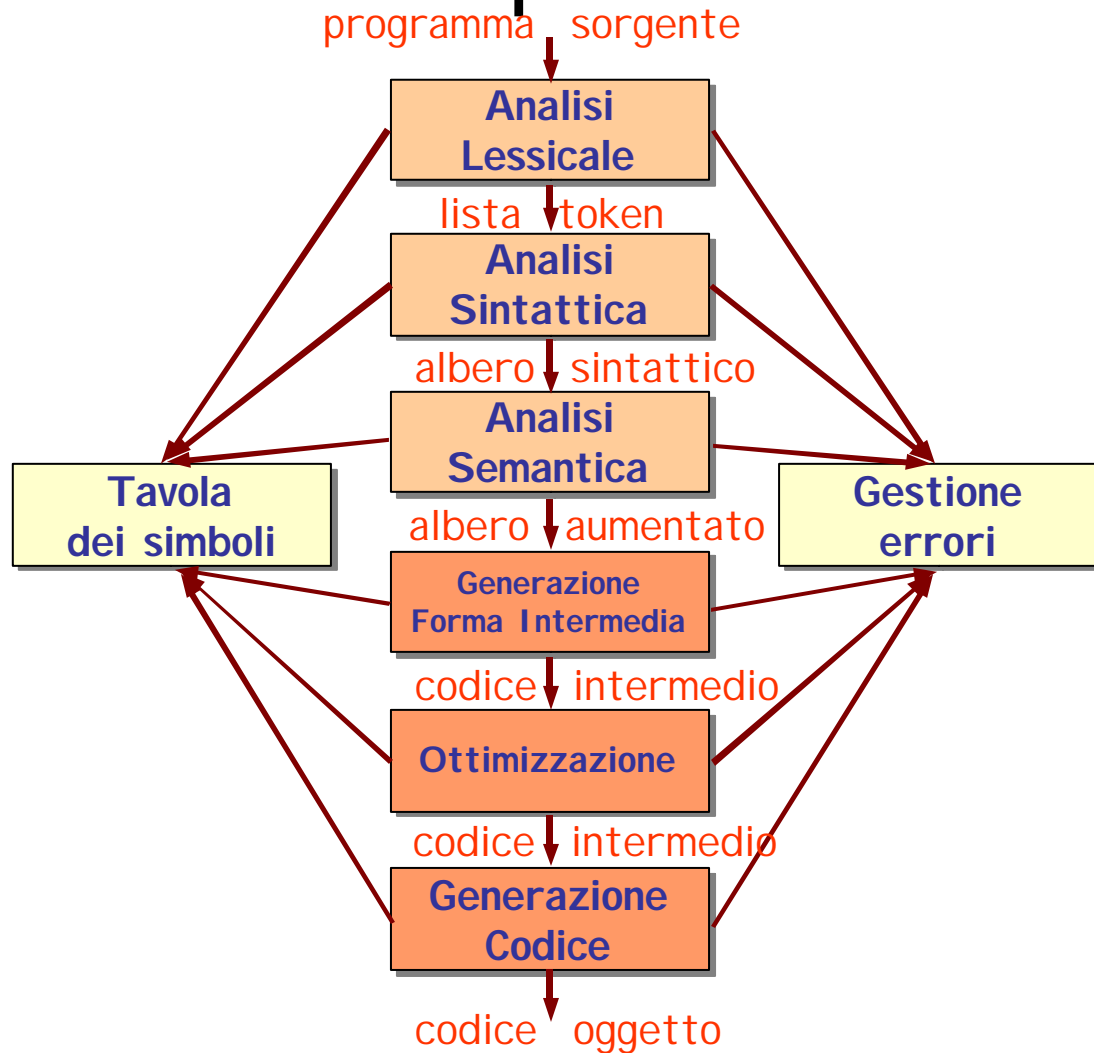
- Var A: integer tipo e allocazione di memoria per A
- int B[10] tipo e allocazione di memoria per il vettore B
- Float myProgram(float x, integer y) {...} parametri
- X=3.5 che tipo è x?

Semantica dinamica

vincoli determinabili in fase di esecuzione del programma

- $z=x/y$ errore se $y=0$
-

Compilatori



Compilazione: analisi lessicale

Detta anche *scanning o lexing*

Da cui **scanner** (o lexer) è il (sotto)programma che la implementa

Lettura dei caratteri dal sorgente
(da sinistra a destra in una passata)
e costruzione di una lista di **token**
(le parole del linguaggio)

es. la stringa `a = 12 * indice++;` genera 7 token:

- `id`, `assign`, `const`, `star`, `id`, `incremento`, `semicolon`

Compilazione: analisi lessicale

Detta anche *scanning o lexing*

Da cui **scanner** (o *lexer*) è il (sotto)programma che la implementa

Lettura dei caratteri dal sorgente
(da sinistra a destra in una passata)
e costruzione di una **lista di token**
(le parole del linguaggio)

es. la stringa `a = 12 * indice++;` genera 7 token:

– `id`, `assign`, `const`, `star`, `id`, `incremento`, `semicolon`

Per descrivere gli **elementi del lessico** basta una sottoclasse delle grammatiche libere detta delle **grammatiche regolari** (o *lineari*)

Compilazione: analisi lessicale

Detta a

Da cui scanner (o lex

Lettura de

(da sinistra

e costruzi

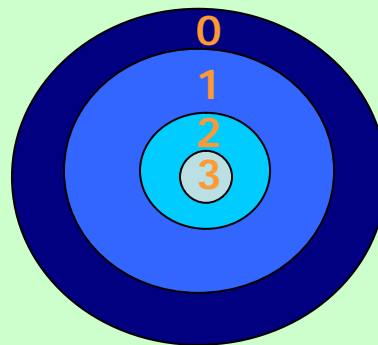
(le pa

es. la stringa a

• a, =, 12, *, indice,

(Linguistica) **Chomsky:**

Specifica 4 classi di strumenti
generativi (grammatiche)



↓
4 classi di
linguaggi

Classe 2 Gram. **non-contestuali** → **sintassi**

Classe 3 Gram. **Regolari** → **lessico**

Per descrivere gli **elementi del lessico** basta una
sottoclasse delle grammatiche libere detta delle
grammatiche regolari (o *lineari*)

Analisi lessicale e Grammatiche regolari

Grammatiche regolari per descrivere il lessico di un LdP

Dovere di cronaca

Le grammatiche regolari hanno regole di produzione del tipo:

$$A \rightarrow uB \quad A \rightarrow Bu$$

con $u \in T^*$ e $A, B \in NT$

Analisi lessicale e Grammatiche regolari

Grammatiche regolari per descrivere il lessico di un LdP

Dovere di cronaca

Le grammatiche regolari hanno regole di produzione del tipo:

$$A \rightarrow uB \quad A \rightarrow Bu$$

con $u \in T^*$ e $A, B \in NT$

Grammatica per descrivere le
espressioni aritmetiche

$G = (\{E, I\}, \{a, b, +, *, -, (,)\}, R, E)$ con

$R = \{$

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow E - E$
5. $E \rightarrow -E$
6. $E \rightarrow (E)$

7. $I \rightarrow a$
8. $I \rightarrow b$
9. $I \rightarrow Ia$
10. $I \rightarrow Ib$

Analisi lessicale e Grammatiche regolari

Grammatiche regolari per descrivere il lessico di un LdP

Dovere di cronaca

Le grammatiche regolari hanno regole di produzione del tipo:

$$A \rightarrow uB \quad A \rightarrow Bu$$

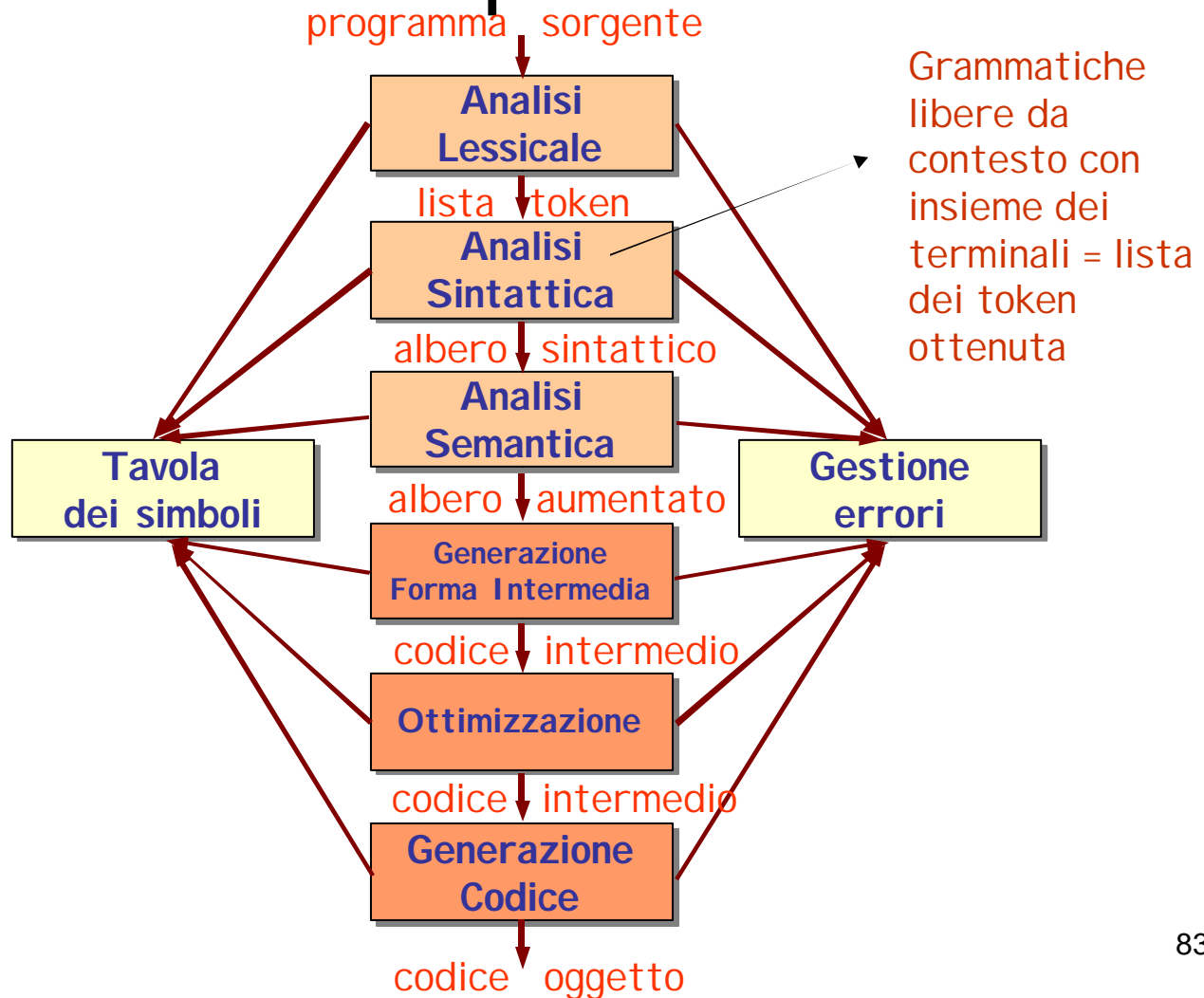
con $u \in T^*$ e $A, B \in NT$

Potere espressivo limitato

Es: non è possibile contare un numero arbitrario di caratteri
(bilanciamento parentesi???)

Analisi lessicale: Verifica che la stringa in ingresso possa
essere decomposta in token (ognuno descritto da una
grammatica regolare)

Compilatori



Compilazione: analisi sintattica

Detta anche parsing

Da cui parser è il (sotto)programma che la implementa

Lettura dei token dalla lista
e costruzione **dell'albero di derivazione**
(i token sono le foglie)

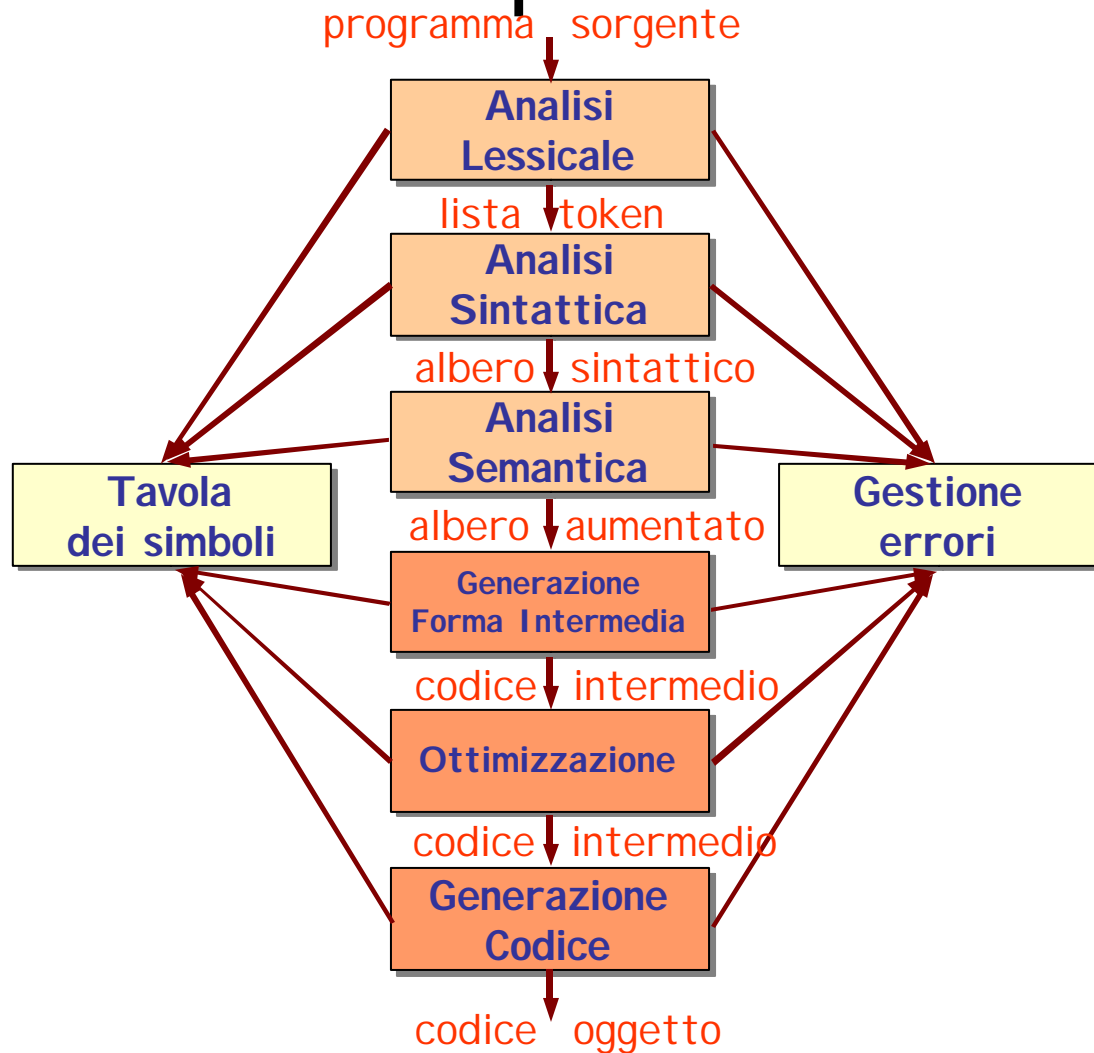
Stringa non corretta nel linguaggio:

se non si è in grado di costruire l'albero

Spesso lo scanner è un sottoprogramma
chiamato dal parser

(lo scanner restituisce un token ad ogni invocazione del parser)

Compilatori



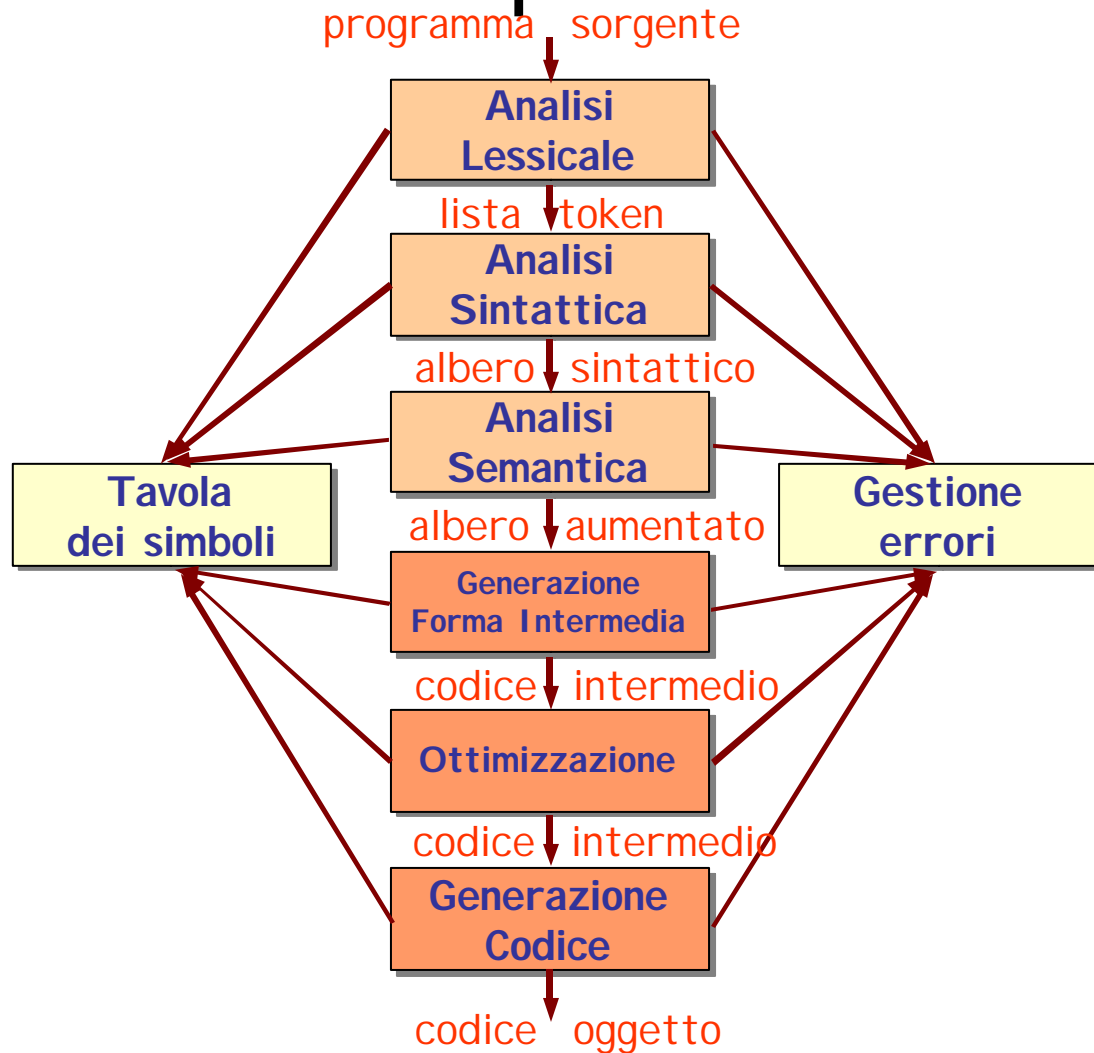
Compilazione: analisi semantica

Vengono effettuati i controlli dei
vincoli contestuali
(ossia di semantica statica)

L'albero viene aumentato
via via con queste nuove informazioni
es. variabile: tipo, punto di dichiarazione, scope, ...

Per gli identificatori esse sono anche registrate
nella **tabella dei simboli**

Compilatori



Compilazione: fasi finali

- **Generazione della forma intermedia**
 - Visita dell'albero per generare codice
 - Forma intermedia
 - Ottimizzabile
 - Portabile
- **Ottimizzazione**
 - Rimozione codice inutile, espansione inline delle chiamate di sottoprogrammi, fattorizzazione espressioni, ottimizzazione dei cicli
- **Generazione del codice oggetto**
 - Segue spesso un'altra fase di ottimizzazione
 - es. assegnamento di variabili molto usate a registri

Sintassi e Semantica

	<i>Complessità</i>	<i>Standardizzazione</i>
<i>Sintassi</i>	Piccola	SI
<i>Semantica</i>	Grande	NO

Separate nella descrizione ma intimamente collegate

La sintassi dovrebbe *suggerire* la semantica

La semantica è guidata dalla sintassi

Semantica (dinamica)

Per descrivere il significato dei programmi.

E' complessa:

- Specifica del linguaggio in modo da bilanciare
 - Esattezza
 - (specifica non ambigua, a priori, di cosa ci si debba aspettare da costrutti sintatticamente corretti, indipendentemente dall'architettura della macchina che esegue il programma)
 - Flessibilità
 - (le molteplici implementazioni del LdP non devono essere impedito.)



Semantica (dinamica)

Per descrivere il significato dei programmi.

E' complessa:

- Specifica del linguaggio in modo da bilanciare
 - Esattezza
 - (specifica non ambigua, a priori, di cosa ci si debba aspettare da costrutti sintatticamente corretti, indipendentemente dall'architettura della macchina che esegue il programma)
 - Flessibilità
 - (le molteplici implementazioni del LdP non devono essere impediti.)



Uso di metodi formali
per descrivere

il significato dei programmi

(derivati e adattati dai linguaggi artificiali della logica matematica)

Semantica (dinamica)

I metodi formali: due grandi famiglie (ma ce ne sono altre)

Semantica denotazionale

(applicazione ai linguaggi di programmazione di tecniche sviluppate per la semantica dei linguaggi logico-matematici)

Significato di un programma \leftrightarrow Uso di funzione: descrive I/O del programma
Dominio e codominio della funzione sono strutture matematiche (I/O, ambienti, memoria)

Semantica (dinamica)

I metodi formali: due grandi famiglie (ma ce ne sono altre)

Semantica denotazionale

(applicazione ai linguaggi di programmazione di tecniche sviluppate per la semantica dei linguaggi logico-matematici)

Significato di un programma \leftrightarrow Uso di funzione: descrive I/O del programma
Dominio e codominio della funzione sono strutture matematiche (I/O, ambienti, memoria)

Semantica operativa

(specifica del comportamento della macchina astratta, tramite un formalismo a basso livello: Automi Assiomi algebrico-logici Stati e transizioni (SOS))

Significato di un programma \leftrightarrow Uso di formalismo di basso livello: descrive il comportamento dell'interprete della macchina astratta

Semantica operativa

Ottenuta definendo un interprete del linguaggio L
su di una macchina ospite i cui componenti sono descritti in modo matematico

L'idea alla base:

Dare la semantica di un linguaggio L
mediante
la definizione del comportamento dell'interprete
(della macchina astratta che riconosce L)
in corrispondenza di programmi scritti in linguaggio L

Ricordiamo il nostro obiettivo:

Descrivere il significato di un programma scritto in linguaggio L
in termini
di cosa ci si debba aspettare dall'uso dei costrutti corretti
del linguaggio L

COSA FA IL PROGRAMMA?

Semantica operativa

Ottenuta definendo un interprete del linguaggio L
su di una macchina ospite i cui componenti sono descritti in modo matematico

L'idea alla base:

Dare la semantica di un linguaggio L
mediante
la definizione del comportamento dell'interprete
(della macchina astratta che riconosce L)
in corrispondenza di programmi scritti in linguaggio L

Utile perché fornisce direttamente un modello di
implementazione

Definibile in modo formale a partire dalla sintassi
(Semantica Operazionale Strutturata)

Sintassi astratta per definire la semantica

Una descrizione sintattica
che evidenzia la struttura sintattica essenziale dei vari costrutti del linguaggio

Alberi di derivazione
che rispettano i vincoli contestuali

Strumento utile per chi pensa a come sia
possibile manipolare un linguaggio

Partiamo dalla sintassi

Un semplice linguaggio di programmazione

$\langle \text{Num} \rangle ::= 1 | 2 | \dots$

$\langle \text{Var} \rangle ::= X_1, X_2, \dots$

$\langle \text{AExpr} \rangle ::= \text{Num} | \text{Var} | \langle \text{AExpr} \rangle + \langle \text{AExpr} \rangle | \langle \text{AExpr} \rangle - \langle \text{AExpr} \rangle$

$\langle \text{BExpr} \rangle ::= \text{tt} | \text{ff} | \langle \text{AExpr} \rangle == \langle \text{AExpr} \rangle | \emptyset \langle \text{BExpr} \rangle |$
 $\langle \text{BExpr} \rangle \neg \langle \text{BExpr} \rangle$

$\langle \text{Com} \rangle ::= \text{skip} | \langle \text{Var} \rangle := \langle \text{AExpr} \rangle | \langle \text{Com} \rangle ; \langle \text{Com} \rangle |$
 $\text{if } \langle \text{BExpr} \rangle \text{ then } \langle \text{Com} \rangle \text{ else } \langle \text{Com} \rangle |$
 $\text{while } \langle \text{BExpr} \rangle \text{ do } \langle \text{Com} \rangle$

La grammatica è ambigua

Ma non importa in questo contesto

Semantica Operazionale Strutturata (SOS)

Definizione della semantica in modo guidato dalla sintassi

Si definiscono delle *regole di transizione*
che specificano
i passi della computazione di
costrutti complessi del linguaggio
(es: $\text{Exp} + \text{Exp}'$)
in termini dei passi della computazione di
costrutti componenti
(es: Exp e Exp')

Semantica Operazionale Strutturata (SOS)

I costrutti cui siamo interessati modificano una qualche
nozione di stato
(in cui la macchina astratta si trova durante la computazione)

Regole definite su configurazioni del tipo

<Comando, Stato>

Not. $\alpha C, s$

Configurazioni \leftrightarrow Stati in cui la macchina si trova durante il suo
funzionamento (mentre l'interprete del linguaggio computa)

Semantica Operazionale Strutturata (SOS)

Regole sono definite su configurazioni del tipo...



Semantica Operazionale Strutturata (SOS)

Tipi di transizioni tra configurazioni

Forma semplice di una transizione: in un solo passo

$$\langle \text{Comando}, \text{Stato} \rangle \rightarrow \text{Stato}'$$
$$c, s \rightarrow t$$

s: stato di partenza

t: stato d'arrivo

es. comando nullo:
 $\langle \text{skip}, \sigma \rangle \rightarrow \sigma$

Semantica Operazionale Strutturata (SOS)

Tipi di regole per la specifica di transizioni tra configurazioni

Forma composta di una transizione: in tanti piccoli passi

$$\langle \text{Comando}, \text{Stato} \rangle \textcircled{R} \langle \text{Comando}', \text{Stato}' \rangle$$
$$\acute{a}c, s\tilde{n} \textcircled{R} \acute{a}c', t\tilde{n}$$

s: stato di partenza

t: stato di arrivo

es. Comando "if then else":
 $\langle \text{if } tt \text{ then } C1 \text{ else } C2, \sigma \rangle \rightarrow \langle C1, \sigma \rangle$

Semantica Operazionale Strutturata (SOS)

forma condizionale

premessa

conseguenza

Le premesse sono i prerequisiti della transizione (conseguenza)

$$\frac{\langle c1, \sigma1 \rangle \rightarrow \langle c'1, \sigma'1 \rangle \quad \langle c2, \sigma2 \rangle \rightarrow \langle c'2, \sigma'2 \rangle}{\langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle}$$

Se $c1$, partendo dallo stato $s1$,
può fare un passo trasformandosi in $c'1$, nello stato $s'1$
e $c2$ partendo da $s2$
può fare un passo trasformandosi in $c'2$, nello stato $s'2$

Allora

il comando c partendo da s può fare un passo trasformandosi in c' nello
stato s'

Semantica delle espressioni aritmetiche

Le configurazioni

$\langle \text{Comando}, \text{Stato} \rangle$

$\hat{a}C, s\hat{n}$

Configurazioni \leftrightarrow Stati in cui la macchina si trova durante il suo
funzionamento

rappresentano gli stati in cui il sistema si trova ad operare
mentre esegue il calcolo di una espressione

Semantica delle espressioni aritmetiche

Le transizioni

$\langle \text{Comando}, \text{Stato} \rangle \rightarrow \langle \text{Comando}', \text{Stato}' \rangle$

$\langle C, s \rangle \rightarrow \langle C', s' \rangle$

" \rightarrow " (relazione di transizione),
ovvero come si passa da una configurazione all'altra

rappresentano il procedere del calcolo stesso
dell'espressione

Una configurazione finale rappresenta il risultato del
calcolo

Semantica delle espressioni aritmetiche (semplificate)

$\langle \text{AExpr} \rangle ::= \langle \text{AExpr} \rangle + \langle \text{AExpr} \rangle \mid \langle \text{Num} \rangle$

Come si calcola il valore di una espressione in questo linguaggio?

In linguaggio naturale:

Semantica delle espressioni aritmetiche (semplificate)

$\langle \text{AExpr} \rangle ::= \langle \text{AExpr} \rangle + \langle \text{AExpr} \rangle \mid \langle \text{Num} \rangle$

Come si calcola il valore di una espressione in questo linguaggio?

In linguaggio naturale:

R1: Il valore di una espressione costituita da un numero n è il valore rappresentato da n (not. \underline{n})

es: il valore di 10 è $\underline{10}$

Semantica delle espressioni aritmetiche

(semplificate)

$\langle \text{AExpr} \rangle ::= \langle \text{AExpr} \rangle + \langle \text{AExpr} \rangle \mid \langle \text{Num} \rangle$

Come si calcola il valore di una espressione in questo linguaggio?

In linguaggio naturale:

R1: Il valore di una espressione costituita da un numero n è il valore rappresentato da n (not. \underline{n})

es: il valore di 10 è $\underline{10}$

R2: Il valore di una espressione $E + E'$ si ottiene sommando n ed m (dove n è il valore di E ed m è il valore di E')

Semantica delle espressioni aritmetiche (semplificate)

$\langle \text{AExpr} \rangle ::= \langle \text{AExpr} \rangle + \langle \text{AExpr} \rangle \mid \langle \text{Num} \rangle$

Come si calcola il valore di una espressione in questo linguaggio?

In linguaggio naturale:

R1: Il valore di una espressione costituita da un numero n è il valore rappresentato da n (not. \underline{n})

es: il valore di 10 è $\underline{10}$

R2: Il valore di una espressione $E + E'$ si ottiene sommando n ed m (dove n è il valore di E ed m è il valore di E')

OSSERVIAMO CHE:

R1 ed R2 corrispondono alle due alternative nella definizione sintattica di $\langle \text{AExpr} \rangle$

R2 suggerisce il modo per calcolare una espressione complessa:
calcolare il valore delle due espressioni di cui si compone e sommare i valori

Semantica delle espressioni aritmetiche (semplificate)

$\langle \text{AExpr} \rangle ::= \langle \text{AExpr} \rangle + \langle \text{AExpr} \rangle \mid \langle \text{Num} \rangle$

Come si calcola il valore di una espressione in questo linguaggio?

CALCOLIAMO $10+11$ applicando le regole R1 ed R2:

R2: Il valore di $10+11$ è $\underline{n} + \underline{m}$, dove \underline{n} è il valore di 10 ed \underline{m} è il valore di 11

ABBIAMO RIDOTTO IL PROBLEMA ALLA RICERCA DEI VALORI \underline{n} ed \underline{m}



R1: il valore di 10 è $\underline{2}$

R1: il valore di 11 è $\underline{3}$

Sappiamo che la somma di $\underline{2}$ e $\underline{3}$ è $\underline{5}$

Il valore di $10+11$ è $\underline{5}$

Semantica delle espressioni aritmetiche (semplificate)

$\langle \text{AExpr} \rangle ::= \langle \text{AExpr} \rangle + \langle \text{AExpr} \rangle \mid \langle \text{Num} \rangle$

Come si calcola il valore di una espressione in questo linguaggio?

Riprendiamo le regole in linguaggio naturale:

R2: Il valore di una espressione $E + E'$ si ottiene sommando n ed m
(dove n è il valore di E ed m è il valore di E')

Le transizioni rappresentano il procedere del calcolo

(scrivere $E \rightarrow n$ significa dire che il valore di E è il numero naturale rappresentato da n)

Semantica delle espressioni aritmetiche (semplificate)

$\langle AExpr \rangle ::= \langle AExpr \rangle + \langle AExpr \rangle \mid \langle Num \rangle$

Come si calcola il valore di una espressione in questo linguaggio?

Riprendiamo le regole in linguaggio naturale:

R2: Il valore di una espressione $E + E'$ si ottiene sommando \underline{n} ed \underline{m}
(dove n è il valore di E ed m è il valore di E')

Le transizioni rappresentano il procedere del calcolo

(scrivere $E \rightarrow \underline{n}$ significa dire che il valore di E è il numero naturale rappresentato da n)

R2 è vista come
regola per determinare la transizione
dalla configurazione del tipo $E + E'$
alla configurazione che rappresenta il valore di $E + E'$

transizioni: se $E \rightarrow \underline{n}$ e $E' \rightarrow \underline{m}$ allora $E + E' \rightarrow \underline{k}$ con $\underline{k} = \underline{n} + \underline{m}$

Semantica delle espressioni aritmetiche (semplificate)

$\langle \text{AExpr} \rangle ::= \langle \text{AExpr} \rangle + \langle \text{AExpr} \rangle \mid \langle \text{Num} \rangle$

Come si calcola il valore di una espressione in questo linguaggio?

Riprendiamo le regole in linguaggio naturale:

R1: Il valore di una espressione costituita da un numero n è il valore rappresentato da n (not. \underline{n})

Le transizioni rappresentano il procedere del calcolo

(scrivere $E \rightarrow \underline{n}$ significa dire che il valore di E è il numero naturale rappresentato da n)

R1 è vista come
regola per determinare il valore \underline{n} di una espressione semplice n

transizioni: $n \rightarrow \underline{n}$

Semantica delle espressioni aritmetiche (semplificate)

$\langle \text{AExpr} \rangle ::= \langle \text{AExpr} \rangle + \langle \text{AExpr} \rangle \mid \langle \text{Num} \rangle$

Regole di transizioni

$n \rightarrow \underline{n}$

$E \rightarrow \underline{n} \quad E' \rightarrow \underline{m} \quad \underline{k} = \underline{n+m}$

$E + E' \rightarrow \underline{k}$

Le regole sono guidate dalla sintassi

La prima regola definisce le transizioni per configurazioni costituite da un elemento della struttura sintattica $\langle \text{Num} \rangle$

La seconda regola definisce transizioni per espressioni in cui compare +

$K=n+m \leftrightarrow$ ipotesi di saper calcolare la somma tra numeri naturali (valori)

Grammatica delle espressioni aritmetiche a valori naturale (semplificata)

$$\langle \text{AExpr} \rangle ::= \langle \text{AExpr} \rangle + \langle \text{AExpr} \rangle \mid \langle \text{AExpr} \rangle - \langle \text{AExpr} \rangle \mid (\text{AExpr}) \mid \langle \text{Num} \rangle$$
$$\langle \text{Num} \rangle ::= 1 \mid 2 \mid \dots$$

La semantica di questo linguaggio è una estensione
Di quello appena visto

Semantica delle espressioni aritmetiche sui numeri naturali

(estensione delle regole precedenti)

$$n \rightarrow \underline{n}$$

$$\frac{E \rightarrow \underline{n} \quad E' \rightarrow \underline{m} \quad k = \underline{n+m}}{E + E' \rightarrow \underline{k}}$$

$$\frac{E \rightarrow \underline{n} \quad E' \rightarrow \underline{m} \quad \underline{n} \geq \underline{m} \quad k = \underline{n-m}}{E - E' \rightarrow \underline{k}}$$

$$\frac{E \rightarrow \underline{n}}{(E) \rightarrow \underline{n}}$$

In che senso semantica del linguaggio delle espressioni aritmetiche?

Vediamo le configurazioni finali come i **valori** delle espressioni di partenza

Osserviamo che

Siano a questo punto abbiamo fatto una semplificazione:

le transizioni erano determinate come funzioni che associavano ad una espressione (in termini di struttura astratta) il valore di questa espressione:

$$E \rightarrow \underline{k}$$

Che fine ha fatto il concetto di STATO?

Grammatica delle espressioni aritmetiche a valori naturali con identificatori

$\langle \text{AExpr} \rangle ::= \langle \text{Num} \rangle \mid \langle \text{Var} \rangle \mid \langle \text{AExpr} \rangle + \langle \text{AExpr} \rangle \mid$
 $\quad \quad \quad \langle \text{AExpr} \rangle * \langle \text{AExpr} \rangle$
 $\langle \text{Num} \rangle ::= 1 \mid 2 \mid \dots$
 $\langle \text{Var} \rangle ::= X_1, \dots$

$\langle \text{AExpr} \rangle ::= \langle \text{Var} \rangle$

consente di scrivere espressioni in cui compaiono nomi
(identificatori)

Semantica delle espressioni aritmetiche a valori naturali con identificatori

Il significato di una espressione

$x+2$

dipende dal significato di x (valore associato ad x)

**In questo contesto serve una qualche nozione di
STATO**

Semantica delle espressioni aritmetiche a valori naturale con identificatori

Il significato di una espressione

$x+2$

dipende dal significato di x (valore associato ad x)

**In questo contesto serve una qualche nozione di
STATO**

Le transizioni sono del tipo

Scrivo:

$\langle E, s \rangle \rightarrow \underline{n}$

Leggo: Il valore di E date le associazioni di s è \underline{n}

Stato

E' definito con riferimento al
modello di memoria della nostra macchina astratta
(mantiene i valori delle variabili)

Stato σ :

sequenza finita di coppie **(X, n)**

Nello stato σ la variabile **X** ha il valore **n**
(not. $n = \sigma(X)$)

Semantica delle espressioni aritmetiche

$$s(\mathbf{X}) = \underline{n}$$

$$\langle n, s \rangle \rightarrow \underline{n}$$

$$\frac{\langle E, s \rangle \rightarrow \underline{n} \quad \langle E', s \rangle \rightarrow \underline{m} \quad \underline{k} = \underline{n} + \underline{m}}{\langle E + E', s \rangle \rightarrow \underline{k}}$$

$$\frac{\langle E, s \rangle \rightarrow \underline{n} \quad \langle E', s \rangle \rightarrow \underline{m} \quad \underline{n} > \underline{m} \quad \underline{k} = \underline{n} - \underline{m}}{\langle E - E', s \rangle \rightarrow \underline{k}}$$

$$\frac{\langle E, s \rangle \rightarrow \underline{n}}{\langle (E), s \rangle \rightarrow \underline{n}}$$

Semantica delle espressioni aritmetiche

$\langle \text{AExpr} \rangle ::= \langle \text{Var} \rangle$

$s(\mathbf{X}) = \underline{n}$

$\langle n, s \rangle \rightarrow \underline{n}$

$$\frac{\langle E, s \rangle \rightarrow \underline{n} \quad \langle E', s \rangle \rightarrow \underline{m} \quad \underline{k} = \underline{n} + \underline{m}}{\langle E + E', s \rangle \rightarrow \underline{k}}$$

$$\frac{\langle E, s \rangle \rightarrow \underline{n} \quad \langle E', s \rangle \rightarrow \underline{m} \quad \underline{n} > \underline{m} \quad \underline{k} = \underline{n} - \underline{m}}{\langle E - E', s \rangle \rightarrow \underline{k}}$$

$$\frac{\langle E, s \rangle \rightarrow \underline{n}}{\langle (E), s \rangle \rightarrow \underline{n}}$$

Esempio Completo

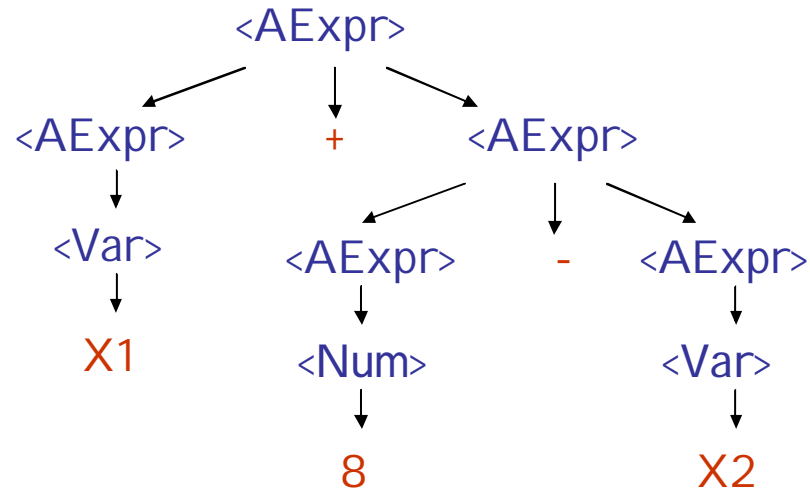
Tramite la seguente grammatica ...

$\langle \text{AExpr} \rangle ::= \langle \text{Num} \rangle | \langle \text{Var} \rangle | \langle \text{AExpr} \rangle + \langle \text{AExpr} \rangle | \langle \text{AExpr} \rangle - \langle \text{AExpr} \rangle$

$\langle \text{Num} \rangle ::= 1 | 2 | \dots$

$\langle \text{Var} \rangle ::= X1, X2, \dots$

Generiamo la stringa sintatticamente corretta
X1+8 - X2 (con struttura astratta)



Esempio Completo

Valutiamo la semantica della struttura astratta $X1 + 8 - X2$ a partire da uno stato s che associa a $X1$ il valore 5 e ad $X2$ il valore 3

Ricordate che la stringa, in realtà è l'albero di derivazione

$$\langle X1 + 8 - X2, s \rangle \rightarrow \{ \dots \} \underline{10}$$

$$\langle X1 + 8 - X2, s \rangle \rightarrow \underline{k}$$

Esempio Completo

Valutiamo la semantica della struttura astratta $X1 + 8 - X2$ a partire da uno stato s che associa a $X1$ il valore 5 e ad $X2$ il valore 3

Ricordate che la stringa, in realtà è l'albero di derivazione

$$3: \frac{\langle E, s \rangle \rightarrow \underline{n} \quad \langle E', s \rangle \rightarrow \underline{m} \quad \underline{k} = \underline{n+m}}{\langle E + E', s \rangle \rightarrow \underline{k}} \quad \langle X1 + 8 - X2, s \rangle \rightarrow \{\dots\} \underline{10}$$

$$\frac{\langle X1, s \rangle \rightarrow \underline{n} \quad \langle 8 - X2, s \rangle \rightarrow \underline{m} \quad \underline{k} = \underline{n+m}}{\langle X1 + 8 - X2, s \rangle \rightarrow \underline{k}} \quad 3$$

Esempio Completo

Valutiamo la semantica della struttura astratta $X1 + 8 - X2$ a partire da uno stato s che associa a $X1$ il valore 5 e ad $X2$ il valore 3

Ricordate che la stringa, in realtà è l'albero di derivazione

$$\langle X1 + 8 - X2, s \rangle \rightarrow \{ \dots \} \underline{10}$$

1: $s(X) = \underline{n}$

$\frac{}{\langle X1, s \rangle \rightarrow \underline{5}}$	$\langle 8 - X2, s \rangle \rightarrow \underline{m}$	$\underline{k} = \underline{5} + \underline{m}$
$\frac{}{\langle X1 + 8 - X2, s \rangle \rightarrow \underline{k}}$		

3

Esempio Completo

Valutiamo la semantica della struttura astratta $X1 + 8 - X2$ a partire da uno stato s che associa a $X1$ il valore 5 e ad $X2$ il valore 3

Ricordate che la stringa, in realtà è l'albero di derivazione

$$4: \frac{\langle E, s \rangle \rightarrow \underline{n} \quad \langle E', s \rangle \rightarrow \underline{m} \quad \underline{n} > \underline{m} \quad \underline{k} = \underline{n} - \underline{m}}{\langle E - E', s \rangle \rightarrow \underline{k}} \quad \langle X1 + 8 - X2, s \rangle \rightarrow \{ \dots \} \underline{10}$$

$$\begin{array}{c} \frac{\frac{\frac{\underline{\langle X1, s \rangle \rightarrow 5}}{1} \quad \underline{\langle 8, s \rangle \rightarrow n1} \quad \underline{\langle X2, s \rangle \rightarrow m2} \quad \underline{n1} > \underline{m2} \quad \underline{m} = \underline{n1} - \underline{m2}}{4:} \quad \underline{\langle 8 - X2, s \rangle \rightarrow m} \quad \underline{k = 5 + m} \quad 3}{\underline{\langle X1 + 8 - X2, s \rangle \rightarrow k}} \end{array}$$

Esempio Completo

Valutiamo la semantica della struttura astratta $X1 + 8 - X2$ a partire da uno stato s che associa a $X1$ il valore 5 e ad $X2$ il valore 3

Ricordate che la stringa, in realtà è l'albero di derivazione

$$\langle X1 + 8 - X2, s \rangle \rightarrow \{ \dots \} \underline{10}$$

$$2: \langle n, s \rangle \rightarrow \underline{n}$$

$$\begin{array}{c}
 \frac{\frac{\frac{\frac{\frac{\frac{\langle X1, s \rangle \rightarrow \underline{5}}{1} \quad \frac{\frac{\frac{\frac{\langle 8, s \rangle \rightarrow \underline{8}}{2} \quad \langle X2, s \rangle \rightarrow \underline{m2} \quad \underline{8} > \underline{m2} \quad \underline{m} = \underline{8} - \underline{m2}}{4:} \quad \underline{k} = \underline{5} + \underline{m}}{3}}{\langle X1 + 8 - X2, s \rangle \rightarrow \underline{k}}
 \end{array}$$

Esempio Completo

Valutiamo la semantica della struttura astratta $X1 + 8 - X2$ a partire da uno stato s che associa a $X1$ il valore 5 e ad $X2$ il valore 3

Ricordate che la stringa, in realtà è l'albero di derivazione

$$\langle X1 + 8 - X2, s \rangle \rightarrow \{...\} \underline{10}$$

1: $s(X) = \underline{n}$

$$\begin{array}{c}
 \frac{}{\langle X1, s \rangle \rightarrow \underline{5}} \quad \frac{\frac{}{\langle 8, s \rangle \rightarrow \underline{8}} \quad \frac{}{\langle X2, s \rangle \rightarrow \underline{3}}}{\frac{\underline{8} > \underline{3} \quad \underline{5} = \underline{8} - \underline{3}}{\langle 8 - X2, s \rangle \rightarrow \underline{5}}} \quad 4: \\
 \hline
 \frac{\langle X1, s \rangle \rightarrow \underline{5} \quad \langle 8 - X2, s \rangle \rightarrow \underline{5}}{\langle X1 + 8 - X2, s \rangle \rightarrow \underline{10}} \quad 3
 \end{array}$$

Un semplice L

La sintassi

**<Exp> ::= <Const> | <Var> | (<Exp>) | <Exp> <Op>
 <Exp> | <!Exp>**

<Op> ::= + | - | * | / | % | == | != | < | <= | > | >= | && | ||

<Const> ::= <Num> | <Bool>

<Bool> ::= true | false

<Num> ::= ...

Semantica delle espressioni logiche

$\text{true}, \text{false} \in \{tt, ff\}$
 $\text{true}, \text{false} \in \{tt, ff\}$

Op. elementari

$$\frac{\langle E, s \rangle \rightarrow bv \quad \langle E', s \rangle \rightarrow bv' \quad bv'' = bv \cup bv'}{\langle E \parallel E', s \rangle \rightarrow bv''}$$

$$\frac{\langle E, s \rangle \rightarrow bv \quad \langle E', s \rangle \rightarrow bv' \quad bv'' = bv \wedge bv'}{\langle E \& E', s \rangle \rightarrow bv''}$$

$$\frac{\langle E, s \rangle \rightarrow bv \quad bv' = \neg bv}{\langle !E, s \rangle \rightarrow bv'}$$

secondo la tavola di verità

bv indica un valore booleano tt o ff

Semantica delle espressioni logiche

$$\frac{\langle E, s \rangle \rightarrow \underline{n} \quad \langle E', s \rangle \rightarrow \underline{m} \quad bv = \underline{n} \text{ rel } \underline{m}}{\langle E \text{ rel } E', s \rangle \rightarrow bv}$$

Rel indica uno qualunque degli operatori di confronto

I comandi

Stato σ :

sequenza finita di coppie (X, n)

Nello stato σ la variabile X ha il valore n

La soluzione ad un problema di programmazione consiste nella individuazione di una sequenza di azioni che modificano lo stato iniziale fino a trasformarlo nello stato finale desiderato

Modifica di uno stato

$s[X \leftarrow v]$: nuovo stato simile a σ dove la variabile X prende il nuovo valore v

La sintassi dei comandi

$\langle \text{Com} \rangle ::= \text{skip} \mid \langle \text{Var} \rangle := \langle \text{AExpr} \rangle \mid \langle \text{Com} \rangle ; \langle \text{Com} \rangle \mid$
 $\text{if } \langle \text{BExpr} \rangle \text{ then } \langle \text{Com} \rangle \text{ else } \langle \text{Com} \rangle \mid$
 $\text{while } \langle \text{BExpr} \rangle \text{ do } \langle \text{Com} \rangle$

$\langle \text{BExpr} \rangle ::= \text{tt} \mid \text{ff} \mid \langle \text{AExpr} \rangle == \langle \text{AExpr} \rangle \mid \emptyset \mid \langle \text{BExpr} \rangle$
 $\langle \text{BExpr} \rangle \hat{\cup} \langle \text{BExpr} \rangle$

Semantica dei comandi

Le transizioni sono del tipo

Scrivo:

$C, s \rightarrow t$

Leggo: Il comando C , date le associazioni di s , porta nello stato s'

Semantica dei comandi

$$\langle \mathbf{skip}, \sigma \rangle \rightarrow \sigma$$

Com nullo

$$\frac{\langle E, s \rangle \rightarrow \underline{n}}{\langle \mathbf{X} := E, s \rangle \rightarrow s[\mathbf{X} \leftarrow n]}$$

Com assegnazione

$$\frac{\langle c1, s \rangle \rightarrow s' \quad \langle c2, s' \rangle \rightarrow t}{\langle c1 ; c2, s \rangle \rightarrow t}$$

Com blocco di istruzioni da eseguire in ordine

Semantica dei comandi

$$\frac{\langle E, s \rangle \rightarrow tt \quad \langle c1, s \rangle \rightarrow t}{\langle \text{i f } E \text{ then } c1 \text{ el se } c2, s \rangle \rightarrow t}$$

$$\langle \text{i f } E \text{ then } c1 \text{ el se } c2, s \rangle \rightarrow t$$

$$\frac{\langle E, s \rangle \rightarrow ff \quad \langle c2, s \rangle \rightarrow t}{\langle \text{i f } E \text{ then } c1 \text{ el se } c2, s \rangle \rightarrow t}$$

$$\langle \text{i f } E \text{ then } c1 \text{ el se } c2, s \rangle \rightarrow t$$

Semantica dei comandi

$$\frac{\langle E, \sigma \rangle \rightarrow ff}{\langle \text{while } E \text{ do } C, \sigma \rangle \rightarrow \sigma}$$

Se la condizione è falsa lo stato non viene modificato (eq. Com nullo)

$$\frac{\langle E, \sigma \rangle \rightarrow tt \quad \langle C, \sigma \rangle \quad \langle \text{while } E \text{ do } C, \sigma' \rangle \rightarrow \tau}{\langle \text{while } E \text{ do } C, \sigma \rangle \rightarrow \tau}$$

Se la condizione è vera C viene ripetutamente eseguito

Esempio

Valutiamo la semantica della struttura astratta $x=2; y=3; x=x-1$ a partire da uno stato s_0 che associa a x e y il valore 0

Ricordate che la stringa, in realtà è l'albero di derivazione

$\langle x=2; y=3; x=x-1, s_0 \rangle \rightarrow \{...\} s_0[x \leftarrow 1, y \leftarrow 3]$

$\langle x=2; y=3; x=x-1, s_0 \rangle \rightarrow \underline{s_f}$

Esempio

Valutiamo la semantica della struttura astratta $x=2; y=3; x=x-1$ a partire da uno stato s_0 che associa a x e y il valore 0

Ricordate che la stringa, in realtà è l'albero di derivazione

$$\frac{\langle c1, s \rangle \rightarrow s' \quad \langle c2, s' \rangle \rightarrow t}{\langle c1 ; c2, s \rangle \rightarrow t} \quad \langle x=2; y=3; x=x-1, s_0 \rangle \rightarrow \{...\} s_0[x \leftarrow 1, y \leftarrow 3]$$

$$\frac{\langle x=2; , s_0 \rangle \rightarrow s_1 \quad \langle y=3; x=x-1; , s_1 \rangle \rightarrow s_f}{\langle x=2; y=3; x=x-1; , s_0 \rangle \rightarrow s_f}$$

Esempio

Valutiamo la semantica della struttura astratta $x=2; y=3; x=x-1$ a partire da uno stato s_0 che associa a x e y il valore 0

Ricordate che la stringa, in realtà è l'albero di derivazione

$$\frac{\langle E, s \rangle \rightarrow \underline{n}}{\langle x := E, s \rangle \rightarrow s[x \leftarrow \underline{n}]} \quad \langle x=2; y=3; x=x-1, s_0 \rangle \rightarrow \{...\} s_0[x \leftarrow 1, y \leftarrow 3]$$

$$\langle 2, s_0 \rangle \rightarrow \underline{2}$$

$$\langle x=2; y=3, s_0 \rangle \rightarrow s_1 = s_0[x \leftarrow \underline{2}]$$

$$\langle y=3; x=x-1; s_1 = s_0[x \leftarrow \underline{2}] \rangle \rightarrow s_f$$

$$\langle x=2; y=3; x=x-1; s_0 \rangle \rightarrow \underline{s_f}$$

Esempio

Valutiamo la semantica della struttura astratta $x=2; y=3; x=x-1$ a partire da uno stato s_0 che associa a x e y il valore 0

Ricordate che la stringa, in realtà è l'albero di derivazione

$$\frac{\langle c1, s \rangle \rightarrow s' \quad \langle c2, s' \rangle \rightarrow t}{\langle c1 ; c2, s \rangle \rightarrow t} \quad \langle x=2; y=3; x=x-1, s_0 \rangle \rightarrow \{...\} s_0[x \leftarrow 1, y \leftarrow 3]$$

$$\frac{\frac{\langle 2, s_0 \rangle \rightarrow \underline{2}}{\langle x=2; s_0 \rangle \rightarrow s_1 = s_0[x \leftarrow \underline{2}]} \quad \frac{\langle y=3; s_1 = s_0[x \leftarrow 2] \rangle \rightarrow s_2 \quad \langle x=x-1; s_2 \rangle \rightarrow s_f}{\langle y=3; x=x-1; s_1 = s_0[x \leftarrow \underline{2}] \rangle \rightarrow s_f}}{\langle x=2; y=3; x=x-1; s_0 \rangle \rightarrow \underline{s_f}}$$

Esempio

Valutiamo la semantica della struttura astratta $x=2; y=3; x=x-1$ a partire da uno stato s_0 che associa a x e y il valore 0

Ricordate che la stringa, in realtà è l'albero di derivazione

$$\frac{\langle E, s \rangle \rightarrow \underline{n}}{\langle x := E, s \rangle \rightarrow s[x \leftarrow \underline{n}]} \quad \langle x=2; y=3; x=x-1, s_0 \rangle \rightarrow \{...\} s_0[x \leftarrow 1, y \leftarrow 3]$$

$$\frac{\frac{\langle 2, s_0 \rangle \rightarrow \underline{2}}{\langle x=2; s_0 \rangle \rightarrow s_1 = s_0[x \leftarrow 2]} \quad \frac{\frac{\langle 3, s_1 \rangle \rightarrow \underline{3}}{\langle y=3; s_1 = s_0[x \leftarrow 2] \rangle \rightarrow s_2 = s_0[x \leftarrow 2, y \leftarrow 3]} \quad \langle x=x-1; s_2 \rangle \rightarrow s_f}{\langle y=3; x=x-1; s_1 = s_0[x \leftarrow 2] \rangle \rightarrow s_f} \quad \frac{\langle x=2; y=3; x=x-1; s_0 \rangle \rightarrow \underline{s_f}}$$

Esempio

Valutiamo la semantica della struttura astratta $x=2; y=3; x=x-1$ a partire da uno stato s_0 che associa a x e y il valore 0

Ricordate che la stringa, in realtà è l'albero di derivazione

$$\frac{\langle E, s \rangle \rightarrow \underline{n}}{\langle x := E, s \rangle \rightarrow s[x \leftarrow \underline{n}]} \quad \langle x=2; y=3; x=x-1, s_0 \rangle \rightarrow \{...\} s_0[x \leftarrow 1, y \leftarrow 3]$$

$$s_f = s_2[x \leftarrow 1], = s_1[y \leftarrow 3] \rightarrow s_0[x \leftarrow 1, y \leftarrow 3]$$

$$\begin{array}{c} \frac{\langle 2, s_0 \rangle \rightarrow \underline{2}}{\langle x=2; s_0 \rangle \rightarrow s_1 = s_0[x \leftarrow 2]} \quad \frac{\frac{\langle 3, s_1 \rangle \rightarrow \underline{3}}{\langle y=3; s_1 = s_0[x \leftarrow 2] \rangle \rightarrow s_2 = s_0[x=2, y \leftarrow 3]} \quad \frac{\langle x-1, s_2 \rangle \rightarrow \underline{1}}{\langle x=x-1; s_2 \rangle \rightarrow s_f}}{\langle y=3; x=x-1; s_1 = s_0[x \leftarrow 2] \rangle \rightarrow s_f} \\ \hline \langle x=2; y=3; x=x-1; s_0 \rangle \rightarrow \underline{s_f} \end{array}$$

Esercizio

Valutiamo la semantica della struttura astratta `while x>0 do x=x-1` **a**
partire da uno stato s che associa a x il valore 2

Computazione

- Sequenza di transizioni concatenate non ulteriormente estendibile in cui ogni transizione è premessa di qualche regola
- Due tipi di computazioni
terminanti: finite
divergenti: infinite (loop)

Pragmatica e Implementazione

- "a che serve / come usare un certo un certo costrutto linguistico ?"
- Obiettivo: migliorare il SW
- Fattori in evoluzione
 - Convenzioni
 - Stile
 - Usare il goto ?
 - Modificare le variabili dei cicli for ?
 - Modalità di passaggio di parametri
 - Scelta iterazioni
- Connesso alla realizzazione pratica dei compilatori
 - "come vengono implementati ?"
 - "a quale costo ?"
- Domande di tipo ingegneristico