



Linguaggi di Programmazione

Capitolo 4 – Linguaggi liberi da contesto

Alberi di derivazione

- Le derivazioni in una grammatica libera da contesto possono essere rappresentate da *alberi* (detti *alberi di derivazione*).

La sequenza delle regole sintattiche utilizzate per generare una stringa w da una grammatica G di simbolo iniziale S definisce la *struttura* di w , che dunque potrebbe essere rappresentata da una delle derivazioni $S \xRightarrow{*} w$. Al fine di disporre di una rappresentazione univoca, si preferisce ricorrere agli alberi di derivazione.

Definizioni preliminari

- Un **Albero** è un grafo orientato, aciclico, connesso e avente al massimo un arco entrante in ciascun nodo.
- La definizione rigorosa di “albero” è la seguente:
 - Un *albero* T è un insieme finito, non vuoto di vertici (*nodi*) tali che:
 - esiste un vertice speciale, detto *radice* dell'albero;
 - esiste una partizione T_1, T_2, \dots, T_m , con $m \geq 0$, degli altri vertici, tale che esista un ordinamento T_1, T_2, \dots, T_m , e ogni sottoinsieme di vertici T_i , $1 \leq i \leq m$, sia a sua volta un albero.
 - Questa definizione è di tipo ricorsivo, ma non è circolare in quanto ogni sottoalbero T_i contiene meno vertici dell'albero T .
 T_1, T_2, \dots, T_m sono detti *sottoalberi* di T . I vertici privi di discendenti sono le *foglie* dell'albero, gli altri sono i *nodi interni*.
La stringa dei simboli che etichettano le foglie di un albero T , letti nell'ordine da sinistra a destra, prende il nome di **frontiera** di T .

Definizione di partizione

- Gli insiemi (non vuoti) T_1, T_2, \dots, T_m , $m \geq 0$, costituiscono una *partizione* di un insieme (non vuoto) T se:

- i) $T_i \cap T_j = \emptyset$ per $i \neq j$, $i, j = 1, 2, \dots, m$

- ii) $\bigcup_{i=1}^m T_i = T$

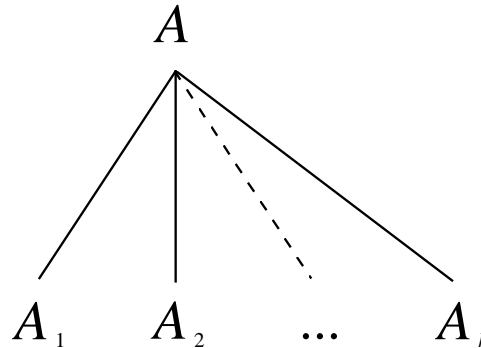
Definizione di albero di derivazione

- Sia $G = (X, V, S, P)$ una grammatica C.F. e $w \in X^*$ una stringa derivabile da S in G , $S \Rightarrow_w^* w$.
Dicesi *albero di derivazione* l'albero T avente le seguenti proprietà:
 - (1) la radice è etichettata con il simbolo iniziale S ;
 - (2) ogni nodo interno (nodo non foglia) è etichettato con un simbolo di V (un nonterminale);
 - (3) ogni nodo foglia è etichettato con un simbolo di X (un terminale) o con λ ;

Definizione di albero di derivazione

- (4) se un nodo N è etichettato con A , ed N ha k discendenti diretti N_1, N_2, \dots, N_k etichettati con i simboli A_1, A_2, \dots, A_k , rispettivamente, allora la produzione:

$$A \rightarrow A_1 A_2 \dots A_k$$



deve appartenere a P ;

- (5) la stringa w può essere ottenuta leggendo (e concatenando) le foglie dell'albero da sinistra a destra.

Definizioni

■ Lunghezza di un cammino

- Dato un albero di derivazione, la *lunghezza di un cammino* dalla radice ad una foglia è data dal numero di nonterminali su quel cammino.

■ Altezza o profondità

- L'*altezza* di un albero è data dalla lunghezza del suo cammino più lungo.

Osservazione

- Un albero di derivazione non impone alcun ordine sull'applicazione delle produzioni in una derivazione. In altri termini, *data una derivazione, esiste uno ed un solo albero di derivazione* che la rappresenta, mentre *un albero di derivazione rappresenta in generale più derivazioni* (in funzione dell'ordine col quale si espandono i nonterminali).

Esempio

Principio di sostituzione di sottoalberi

■ Definizione di derivazione destra (sinistra)

- Data una grammatica $G = (X, V, S, P)$, diremo che una *derivazione* $S \Rightarrow^* w$, ove:

$$S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w_n = w$$

$$w_i = y_i A z_i, \quad w_{i+1} = y_i w_i z_i, \quad i = 1, 2, \dots, n-1$$

è *destra* (*sinistra*) se, per ogni i , $i = 1, 2, \dots, n-1$, risulta:

$$z_i \in X^* \quad (y_i \in X^*)$$

In altre parole in una *derivazione destra* (*sinistra*) ad ogni passo si espande il nonterminale posto più a destra (sinistra).

Esempio

- Riconsideriamo la grammatica che genera tutte e sole le stringhe che hanno un ugual numero di 1 e di 0.

$$S \rightarrow 0B \mid 1A$$

$$A \rightarrow 0 \mid 0S \mid 1AA$$

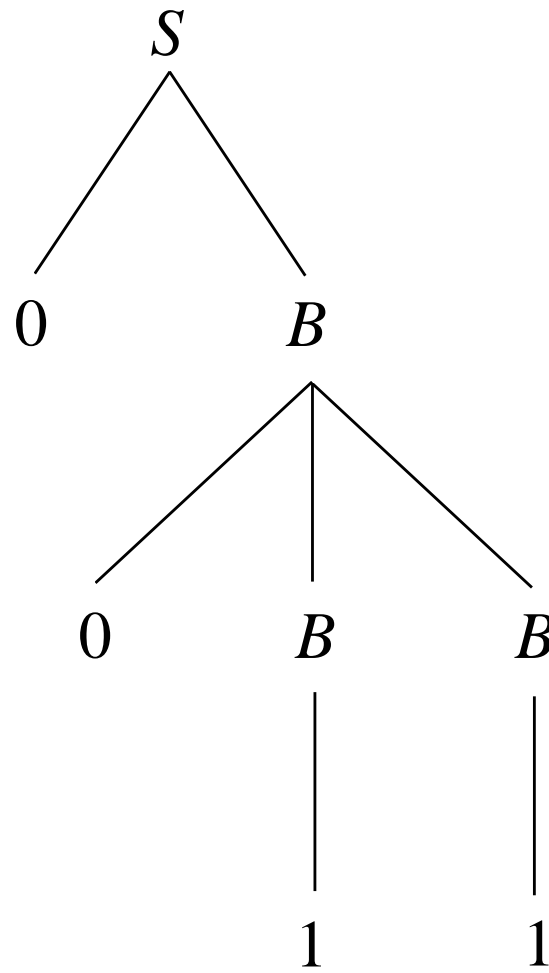
$$B \rightarrow 1 \mid 1S \mid 0BB$$

Consideriamo la stringa 0011. Una possibile derivazione è: $S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 001B \Rightarrow 0011$

Un'altra possibile è: $S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 00B1 \Rightarrow 0011$

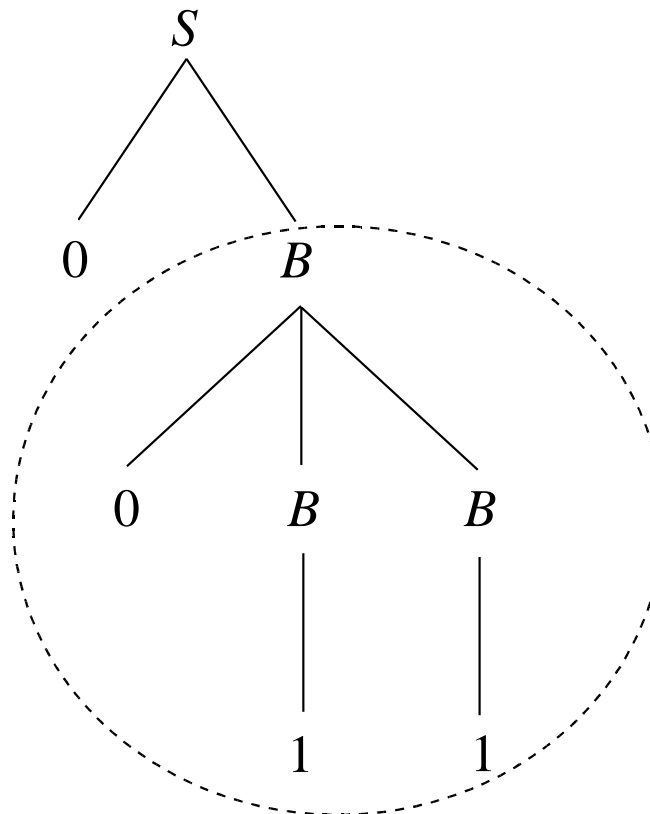
- Il non determinismo insito nella derivazione scompare quando si considera il relativo albero di derivazione (vedi figura seguente).

Esempio

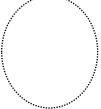
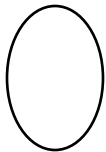


Principio di sostituzione di sottoalberi

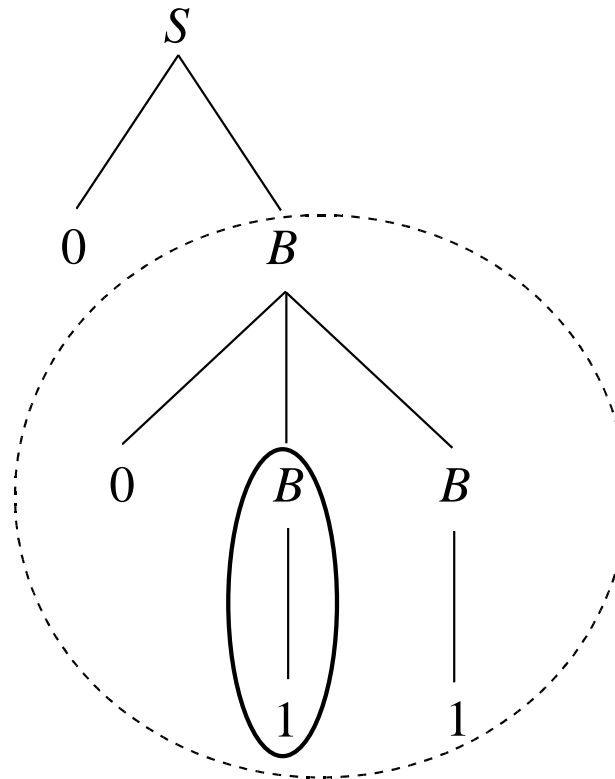
- Consideriamo ora il sottoalbero con radice nel nodo di profondità minore etichettato con una B .



Principio di sostituzione di sottoalberi

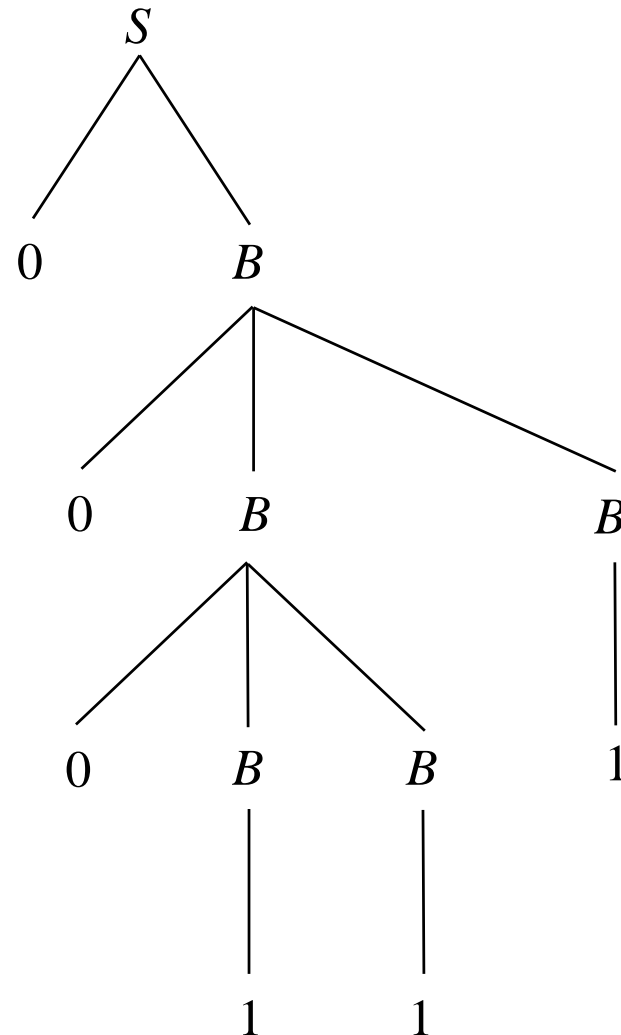
- Cosa accade se un qualsiasi sottoalbero con radice in un nodo etichettato con una B viene sostituito con il sottoalbero  ?
- Il nuovo albero è ancora un albero di derivazione (ovviamente, per una stringa differente da 0011).
- Consideriamo, ad esempio, il sottoalbero individuato dal cerchio pieno  nella figura seguente

Principio di sostituzione di sottoalberi



- Il nuovo albero di derivazione è rappresentato nella seguente figura:

Principio di sostituzione di sottoalberi



■ per cui: $S \xRightarrow{*} 000111$

Principio di sostituzione di sottoalberi

- Possiamo ripetere indefinitamente questo processo di sostituzione di sottoalberi, ottenendo parole del linguaggio di lunghezza crescente:

$$\begin{array}{ll} 0011 & |w| = 4 \\ 000111 & |w| = 6 \\ w = 00001111 & |w| = 8 \\ M & M \\ 00^n 11^n & |w| = 2n + 2 \quad n = 1, 2, K \end{array}$$

La lunghezza cresce in maniera costante.

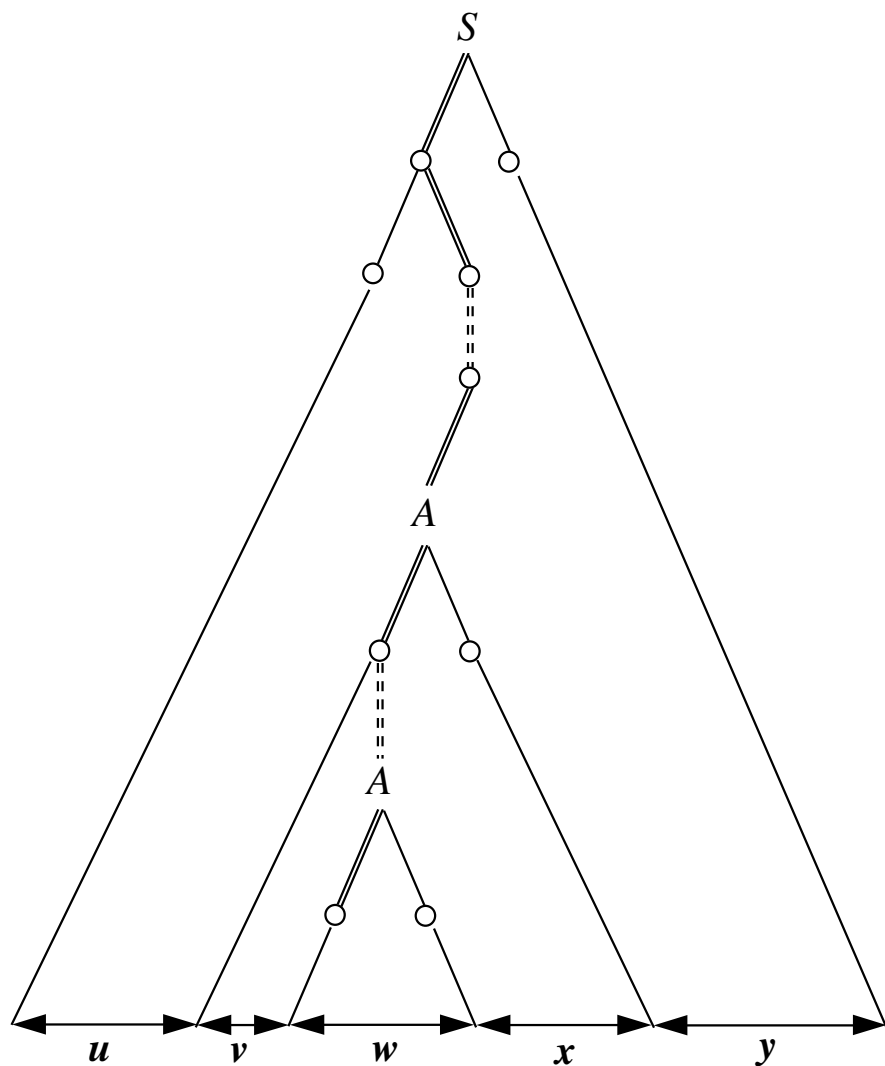
Principio di sostituzione di sottoalberi

- Nelle grammatiche C.F., dunque, possiamo sostituire alberi più piccoli con alberi di dimensioni maggiori, purché abbiano la stessa radice - più precisamente, purché i nodi radice siano etichettati con lo stesso NT - ottenendo ancora alberi di derivazione validi.
- Una caratteristica dei linguaggi C.F., che discende direttamente dal processo descritto in precedenza, è che la lunghezza delle parole cresce in maniera costante.
- Dunque, se una grammatica genera parole la cui lunghezza cresce in maniera esponenziale, allora il linguaggio generato non è libero.
- Controesempio $L = \{a^n b^n c^n \mid n > 0\}$

Principio di sostituzione di sottoalberi

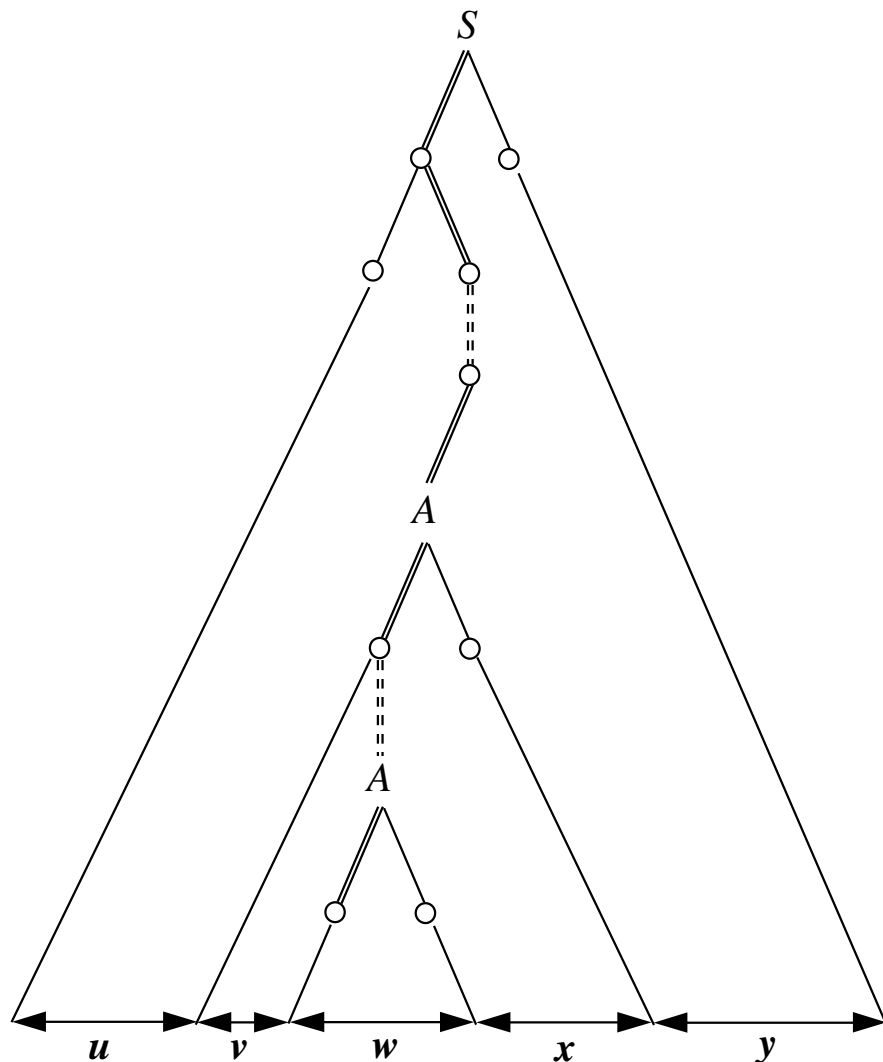
- Generalizziamo ora il discorso:
Supponiamo di avere un albero di derivazione T_z per una stringa z di terminali generata da una grammatica C.F. G , e supponiamo inoltre che il simbolo $NT A$ compaia due volte su uno stesso cammino.
La situazione è rappresentata graficamente in Figura.

Principio di sostituzione di sottoalberi



Il sottoalbero più in basso con radice nel nodo etichettato con una A genera la sottostringa w , mentre quello più in alto genera la sottostringa vwx . Poiché la G è C.F., sostituendo il sottoalbero più in alto con quello più in basso, si ottiene ancora una derivazione valida. Il *nuovo albero* genera la *stringa* uwy .

Principio di sostituzione di sottoalberi



Se effettuiamo la sostituzione inversa (il sottoalbero più in basso viene rimpiazzato da quello più in alto), otteniamo un albero di derivazione lecito per la stringa $uvvwxxxy$, ossia uv^2wx^2y . Ripetendo questa sostituzione un numero finito di volte, si ottiene l'insieme di stringhe:

$$\{uv^nwx^n y \mid n \geq 0\}$$

Proposizione

- Ogni linguaggio C.F. infinito deve contenere almeno un sottoinsieme infinito di stringhe della forma:

$$uv^nw x^n y \quad n \geq 0$$

Formalizziamo ora alcuni risultati connessi con il processo di sostituzione di sottoalberi.

Lemma

- Sia $G = (X, V, S, P)$ una grammatica C.F. e supponiamo che:

$$m = \max \left\{ |v| \mid A \rightarrow v \in P \right\}$$

Sia T_w un albero di derivazione per una stringa w di $L(G)$. Se l'altezza di T_w è al più uguale ad un intero j , allora: $|w| \leq m^j$

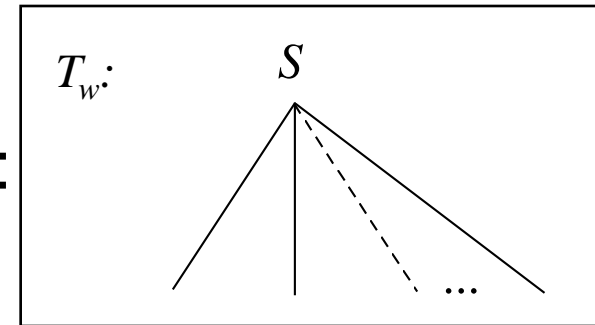
In formule: $height(T_w) \leq j \Rightarrow |w| \leq m^j$

Dimostrazione

- La dimostrazione vale per un albero di derivazione che abbia come radice un qualunque simbolo NT , non necessariamente S .
Procediamo per induzione su j .

Passo base $j = 1$

T_w rappresenta un'unica produzione:



Dunque la parola generata è composta al più da m caratteri terminali e si ha: $|w| \leq m$

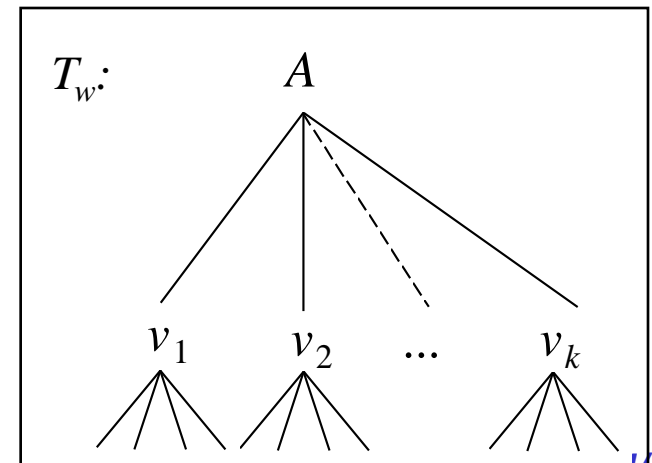
Dimostrazione

■ Passo induttivo

Supponiamo che il lemma valga per ogni albero di altezza al più uguale a j e la cui radice sia un simbolo NT e dimostriamolo per un albero di altezza $j+1$.

Supponiamo che il livello più alto dell'albero rappresenti la produzione $A \rightarrow v$, dove

$v = v_1 v_2 \dots v_k$, $|v| = k$, $k \leq m$ (il sottoalbero di profondità 1 ha al più m figli).



Dimostrazione

- Ogni simbolo v_i , $i = 1, 2, \dots, k$ di v può essere radice di un sottoalbero di altezza al più uguale a j , poiché T_w ha altezza uguale a $j+1$ (un v_i potrebbe anche essere un terminale).

Dunque, per ipotesi di induzione, ciascuno di questi alberi ha al più m^j foglie. Poiché $|v| = k \leq m$, la stringa w , frontiera dell'albero T_w , ha lunghezza:

$$|w| \leq \underbrace{m^j + m^j + \dots + m^j}_{|v|=k \text{ volte}} = |v| \cdot m^j = k \cdot m^j \leq m \cdot m^j = m^{j+1}$$

c.v.d.

Pumping Lemma per i linguaggi liberi da contesto o teorema $uvwxy$

- Sia L un linguaggio libero da contesto. Allora esiste una costante p , che dipende solo da L , tale che se z è una parola di L di lunghezza maggiore di p ($|z| > p$), allora z può essere scritta come $uvwxy$ in modo tale che:
 - (1) $|vwx| \leq p$
 - (2) al più uno tra v e x è la parola vuota ($vx \neq \lambda$);
 - (3) $\forall i, i \geq 0 : uv^iwx^iy \in L$

Osservazione

- Dato un linguaggio generato da una grammatica che non è C.F., non possiamo escludere immediatamente che non esista una grammatica C.F. che generi lo stesso linguaggio.
- Se un linguaggio infinito non obbedisce al Pumping Lemma, non può essere generato da una grammatica C.F.
- Il Pumping Lemma per i linguaggi liberi fornisce una *condizione necessaria* affinché un linguaggio sia libero.

$$L \text{ libero} \Rightarrow \exists p, \dots$$

- Dunque può essere utilizzato per dimostrare che un linguaggio non è libero (con una dimostrazione per assurdo - modus tollens).

$$\nexists p, \dots \Rightarrow L \text{ non è libero}$$

Dimostrazione Pumping Lemma

- Sia $G = (X, V, S, P)$ una grammatica C.F. che genera L . Denotiamo con m il numero di simboli della più lunga parte destra di una produzione di G :

$$m = \max \left\{ |v| \mid A \rightarrow v \in P \right\}$$

Denotiamo con k la cardinalità dell'alfabeto nonterminale di G :

$$k = |V|$$

Poniamo $p = m^{k+1}$ e sia $z \in L$, con $|z| > p$

Per il lemma precedente:

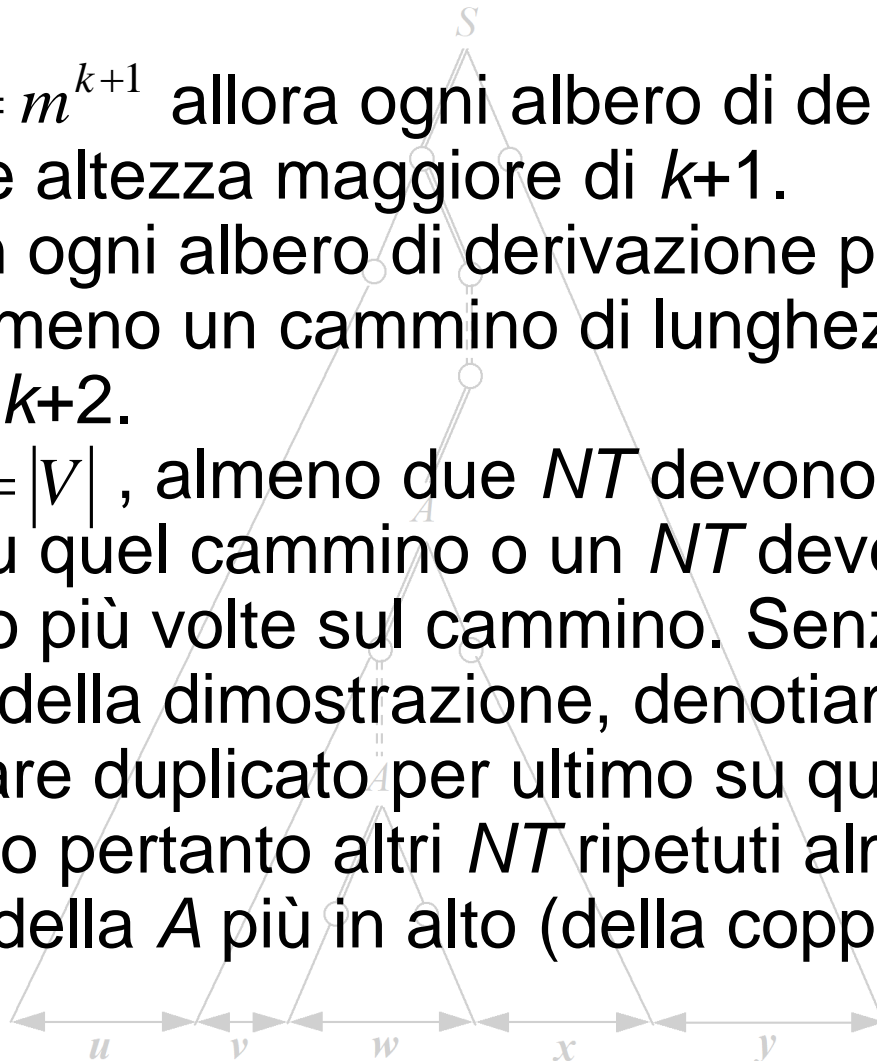
$$\left[\left(\text{height}(T_z) \leq j \Rightarrow |z| \leq m^j \right) \Leftrightarrow \left(|z| > m^j \Rightarrow \text{height}(T_z) > j \right) \right]$$

Dimostrazione Pumping Lemma

- se $|z| > p = m^{k+1}$ allora ogni albero di derivazione per z deve avere altezza maggiore di $k+1$.

Dunque, in ogni albero di derivazione per z deve esistere almeno un cammino di lunghezza non inferiore a $k+2$.

Poiché $k = |V|$, almeno due NT devono comparire duplicati su quel cammino o un NT deve essere ripetuto 3 o più volte sul cammino. Senza ledere la generalità della dimostrazione, denotiamo con A il NT che compare duplicato per ultimo su quel cammino. Non vi sono pertanto altri NT ripetuti almeno due volte al di sotto della A più in alto (della coppia di A).



Dimostrazione Pumping Lemma

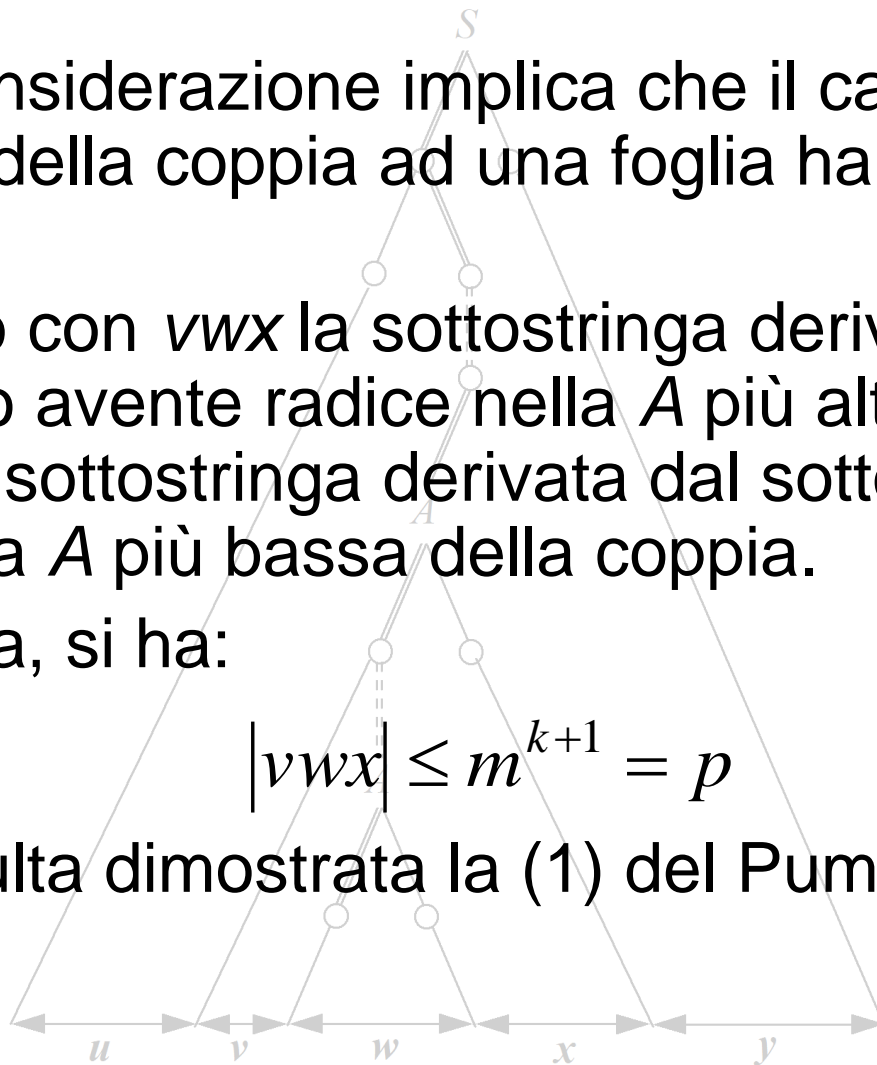
- Questa considerazione implica che il cammino dalla A più in alto della coppia ad una foglia ha lunghezza al più $k+1$.

Indichiamo con vwx la sottostringa derivata dal sottoalbero avente radice nella A più alta della coppia, ove w è la sottostringa derivata dal sottoalbero avente radice nella A più bassa della coppia.

Dal Lemma, si ha:

$$|vwx| \leq m^{k+1} = p$$

per cui risulta dimostrata la (1) del Pumping Lemma.



Dimostrazione Pumping Lemma

- Scriviamo z nella forma $uvwxy$, ed applichiamo il principio di sostituzione di sottoalberi. Se sostituiamo al sottoalbero avente radice nella A più alta della coppia quello avente radice nella A più bassa, otteniamo un albero di derivazione per la stringa:
 $uwy = uv^0wx^0y$ che è la (3) per $i = 0$.
Se operiamo la sostituzione inversa, otteniamo un albero di derivazione per la stringa: $uvvwxxxy = uv^2wx^2y$ che è la (3) per $i = 2$.
Se ripetiamo la suddetta sostituzione $i-1$ volte, la stringa derivata è: $u \underbrace{vv \dots v}_{i \text{ volte}} w \underbrace{xx \dots x}_{i \text{ volte}} y = uv^iwx^iy$
per cui la (3) risulta dimostrata.

Dimostrazione Pumping Lemma

- La (2) può essere dimostrata per assurdo.

Sia: $v = \lambda = x$

La sostituzione del sottoalbero avente radice nella A più alta della coppia con quello avente radice nella A più bassa non provoca alcun cambiamento nella stringa z derivata dall'intero albero. Ma tale sostituzione provoca la diminuzione della lunghezza del cammino che dalla A (in origine quella più in alto) porta ad una foglia. Dunque, anche il cammino di lunghezza non inferiore a $k+2$ (su cui compariva la coppia di A) nell'albero di derivazione per z risulta accorciato. In questo modo abbiamo ottenuto un albero di derivazione per z con altezza almeno uguale a $k+1$. Ma questo è assurdo per il Lemma.

c.v.d.

Definizione di grammatica ambigua

- Una grammatica G libera da contesto è **ambigua** se esiste una stringa x in $L(G)$ che ha due alberi di derivazione differenti.
- In modo alternativo, G è ambigua se esiste una stringa x in $L(G)$ che ha due derivazioni sinistre (o destre) distinte.

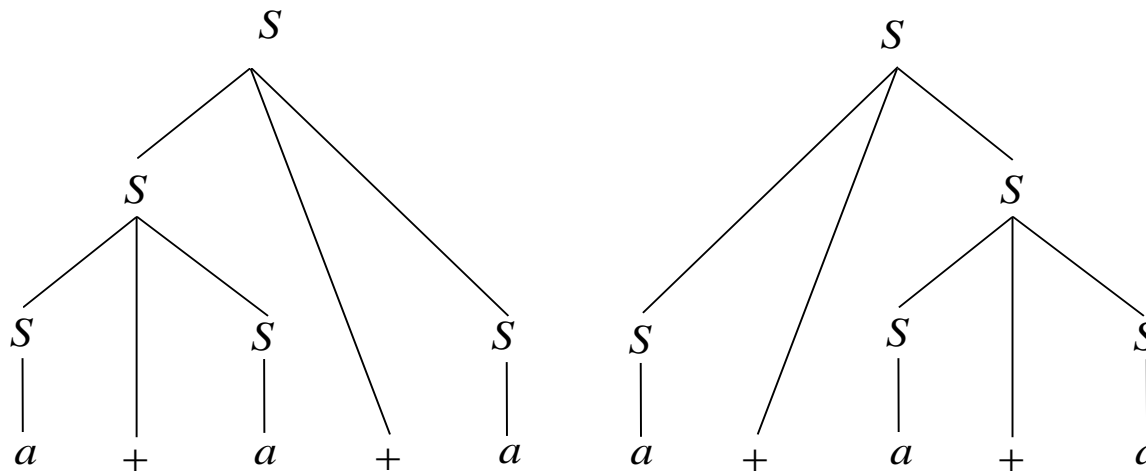
Esempio di grammatica ambigua

- La seguente grammatica libera da contesto:

$$G_2 = (X, V, S, P)$$

$$X = \{a, +\}, \quad V = \{S\}, \quad P = \{S \rightarrow S + S, S \rightarrow a\}$$

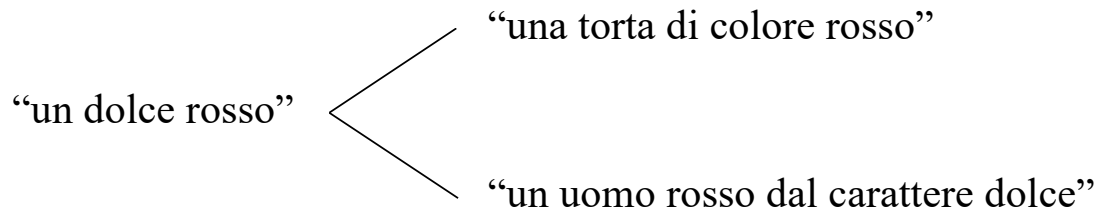
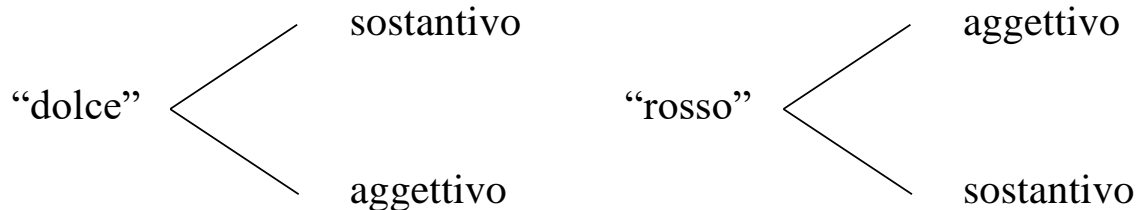
è ambigua. Infatti la stringa $w = a + a + a$ in $L(G_2)$ ha due differenti alberi di derivazione.



Nel linguaggio naturale, l'ambiguità (sintattica) è un fenomeno intrinseco, che si verifica frequentemente.

Esempio

“un dolce rosso”



- Agli effetti delle applicazioni ai linguaggi di programmazione, *l'ambiguità è una proprietà negativa*. Infatti, il *significato di una frase* può essere definito come una *funzione del suo albero di derivazione*.

Esempio

- Sicché, se esiste più di un albero di derivazione per una stessa frase, essa può avere un significato non univoco.

Si preferisce perciò cercare una grammatica differente che, pur generando lo stesso linguaggio, non sia ambigua.

L'unico vantaggio delle grammatiche ambigue risulta essere, in generale, il minore numero di regole che possono avere rispetto ad una grammatica non ambigua.

Esempio

- Il linguaggio *precedente* può essere generato anche dalla seguente grammatica:

$$G_3 = (X, V, S, P)$$

$$X = \{a, +\}, \quad V = \{S\}, \quad P = \{S \rightarrow S + a, S \rightarrow a\}$$

G_3 non è ambigua.

Inoltre: $L(G_2) = L(G_3) = \{aw \mid w \in \{+a\}^*\} = a \cdot \{+a\}^*$

- Esistono però dei linguaggi, detti *inerentemente ambigui*, per i quali tutte le grammatiche che li generano risultano ambigue.

Definizione di linguaggio inerentemente ambiguo

- Un linguaggio L è *inerentemente ambiguo* se ogni grammatica che lo genera è ambigua.

$$(\forall G) L = L(G) : G \text{ ambigua}$$

Esempio di linguaggio inerentemente ambiguo

- Un linguaggio inerentemente ambiguo è:

$$L = \{a^i b^j c^k \mid a^i b^j c^k, (i = j \vee j = k)\}$$

Intuitivamente, ci si rende conto di ciò pensando che in ogni grammatica che genera L devono esistere due diversi insiemi di regole per produrre le stringhe $a^i b^j c^k$ e le stringhe $a^i b^i c^i$. Le stringhe $a^i b^i c^i$ saranno necessariamente prodotte in due modi distinti, con distinti alberi di derivazione e conseguente ambiguità.

Esempio di linguaggio ambiguo

■ Linguaggi di programmazione

□ Si consideri la seguente grammatica $G = (X, V, S, P)$:

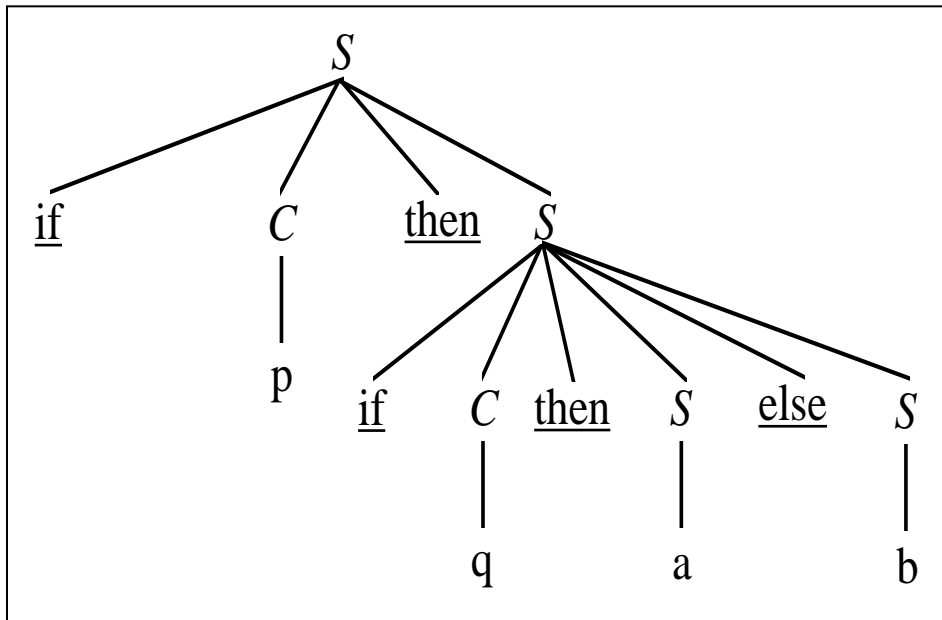
$$X = \{\underline{\text{if}}, \underline{\text{then}}, \underline{\text{else}}, a, b, p, q\}$$
$$V = \{S, C\}$$
$$P = \{S \rightarrow \underline{\text{if}} \ C \ \underline{\text{then}} \ S \ \underline{\text{else}} \ S \mid \underline{\text{if}} \ C \ \underline{\text{then}} \ S \mid a \mid b, C \rightarrow p \mid q\}$$

G è ambigua. Infatti la stringa:

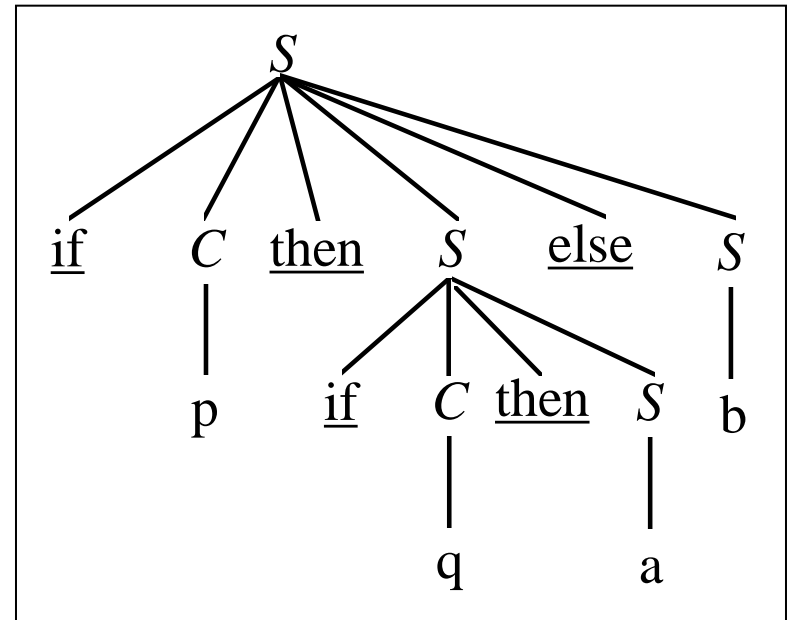
$$w = \underline{\text{if}} \ p \ \underline{\text{then}} \ \underline{\text{if}} \ q \ \underline{\text{then}} \ a \ \underline{\text{else}} \ b$$

può essere generata in due modi.

Esempio di linguaggio ambiguo



if *p* then (if *q* then *a* else *b*)



if *p* then (if *q* then *a*) else *b*

Esempio di linguaggio ambiguo

- Per rendere non ambigua G si usa solitamente la convenzione di associare ogni istruzione else con l'istruzione if più vicina.

La grammatica che riflette questa convenzione è la seguente:

$$G' = (X, V, S, P)$$

$$X = \{\underline{\text{if}}, \underline{\text{then}}, \underline{\text{else}}, a, b, p, q\}$$

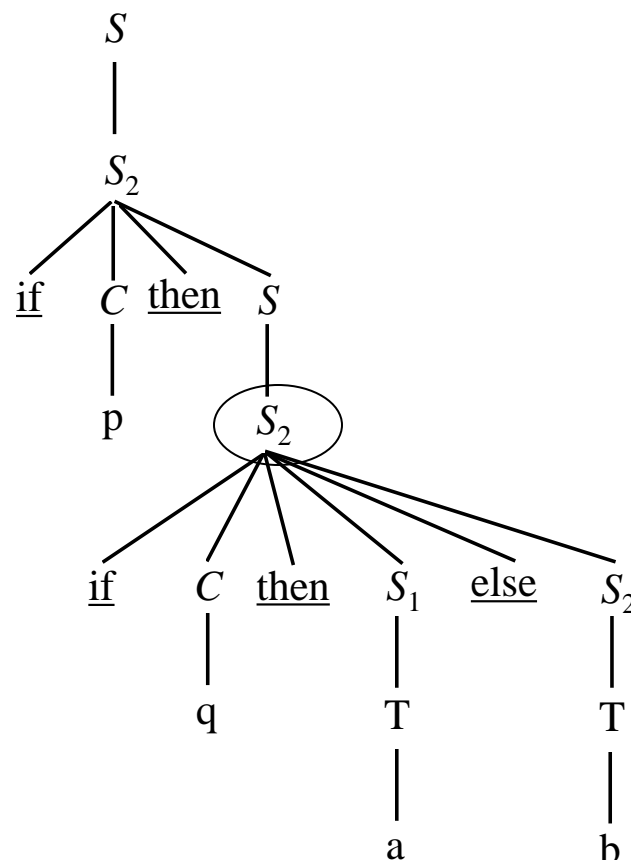
$$V = \{S, S1, S2, C, T\}$$

$$P = \{S \rightarrow S1 \mid S2, S1 \rightarrow \underline{\text{if}} \ C \ \underline{\text{then}} \ S1 \ \underline{\text{else}} \ S2 \mid T, S2 \rightarrow \underline{\text{if}} \ C \ \underline{\text{then}} \ S \mid \underline{\text{if}} \ C \ \underline{\text{then}} \ S1 \ \underline{\text{else}} \ S2 \mid T, C \rightarrow p \mid q, T \rightarrow a \mid b\}$$

Esempio di linguaggio ambiguo

- In G' la stringa $w = \underline{\text{if}}\ p\ \underline{\text{then}}\ \underline{\text{if}}\ q\ \underline{\text{then}}\ a\ \underline{\text{else}}\ b$ ha un unico albero di derivazione:

- Ma è proprio vero che l'albero di derivazione per w è unico? Si tenga conto che l'ambiguità di un linguaggio prescinde dal nome delle variabili ma dipende dalla struttura.



Esercizi

- Determinare la grammatica che genera il seguente linguaggio:

$$L = \{a^n b^n c^n \mid n > 0\}$$

e dimostrare la correttezza di tale grammatica.

- Di che tipo è la grammatica che genera L ?
- Applicare il Pumping Lemma per dimostrare che L non è libero.

Soluzione esercizio

Esercizi

- Applicare il Pumping Lemma per dimostrare che il seguente linguaggio L non è libero da contesto:

$$L = \left\{ a^{n^2} \mid n \geq 0 \right\}$$

Soluzione esercizio

Esercizi

- Applicare il Pumping Lemma per dimostrare che il seguente linguaggio L non è libero da contesto:

$$L = \left\{ a^i b^j \mid i = 2^j, i, j \geq 0 \right\}$$

Soluzione esercizio

Esercizi

- Dimostrare che il seguente linguaggio non è libero da contesto:

$$L = \{ a^k b^r \mid k > 0, r > k^2 \}$$

Soluzione esercizio