Linguaggi di Programmazione

Docente: Cataldo Musto

Capitolo 4 – Linguaggi liberi da contesto

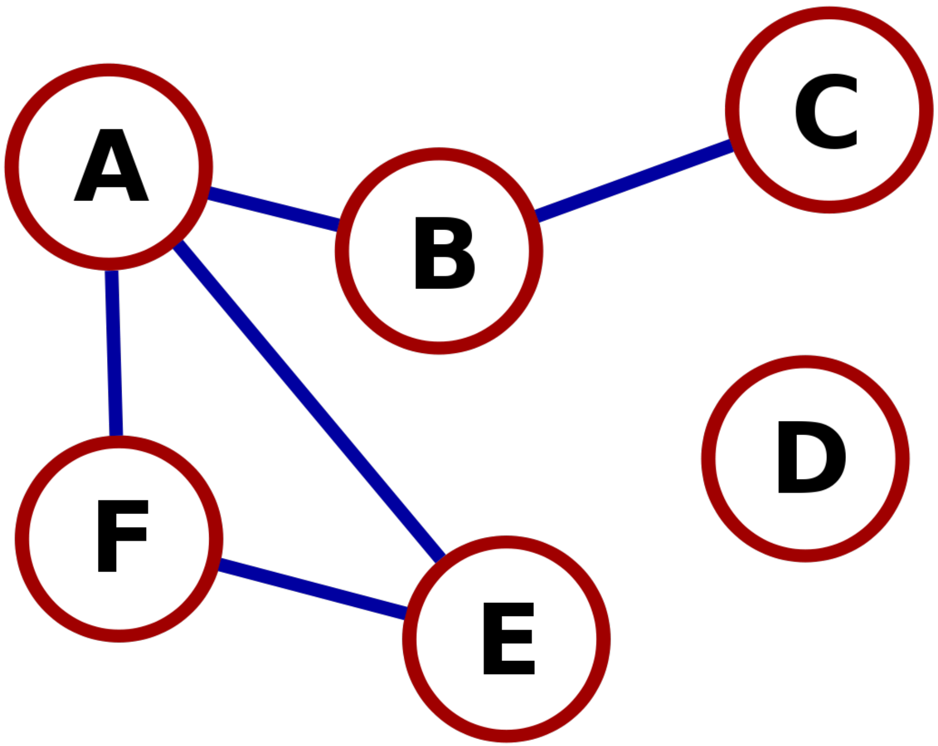
Si ringraziano il Prof. Giovanni Semeraro, il Prof. Marco de Gemmis e il Prof. Pasquale Lops per la concessione del materiale didattico di base

Definizioni preliminari

- Gli insiemi (non vuoti) T_1, T_2, ..., T_m, $m \geq 0$, costituiscono una *partizione* di un insieme (non vuoto) T se:
  - i) $T_i \cap T_j = \varnothing$ per $i \neq j,\ i, j = 1, 2, \ldots, m$
  - ii) $\bigcup_{i=1}^{m} T_i = T$

Definizioni preliminari

- Un Grafo è un insieme di elementi detti nodi, collegati tra di loro mediante degli archi.

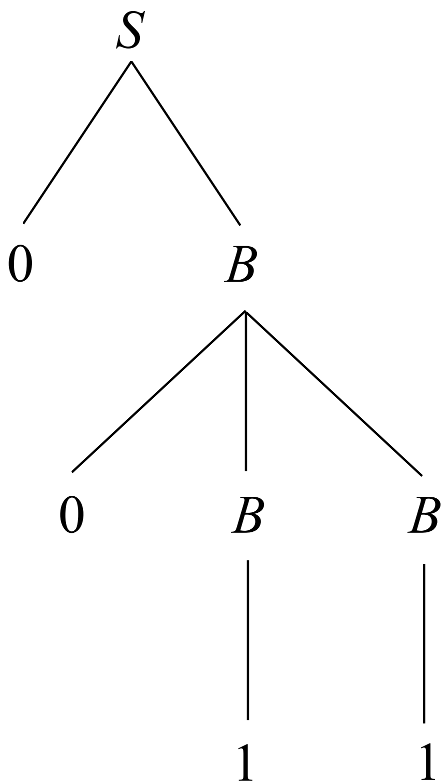


Definizioni preliminari

- Un Albero è un grafo orientato, aciclico, connesso e avente al massimo un arco entrante in ciascun nodo.

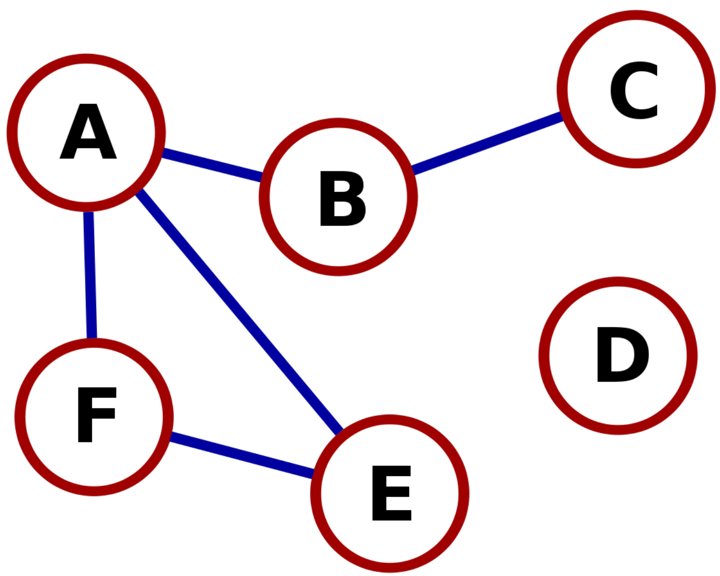
Definizioni preliminari

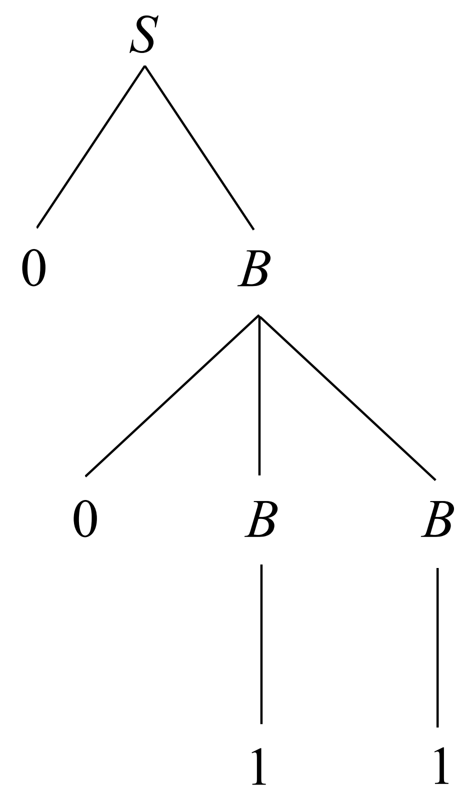
- Un Albero è un grafo orientato, aciclico, connesso e avente al massimo un arco entrante in ciascun nodo.



Definizioni preliminari

- Un Albero è un grafo orientato, aciclico, connesso e avente al massimo un arco entrante in ciascun nodo.

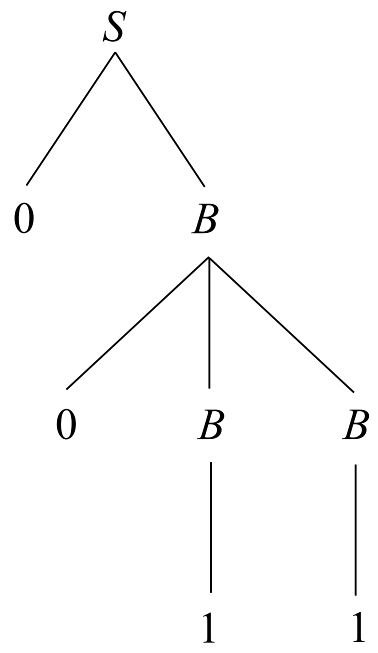




Ogni albero è anche un grafo.
Non tutti i grafi sono alberi!

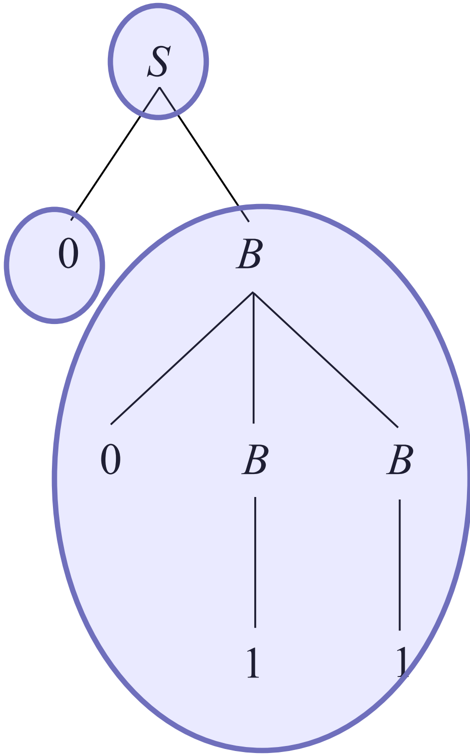
Definizioni preliminari

- 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** $T_1, T_2, ..., T_m$, con $m \geq 0$, degli altri vertici, tale che esista un ordinamento $T_1, T_2, ..., T_m$, e ogni sottoinsieme di vertici T_i, $1 \leq i \leq m$, sia a sua volta un albero.



Definizioni preliminari

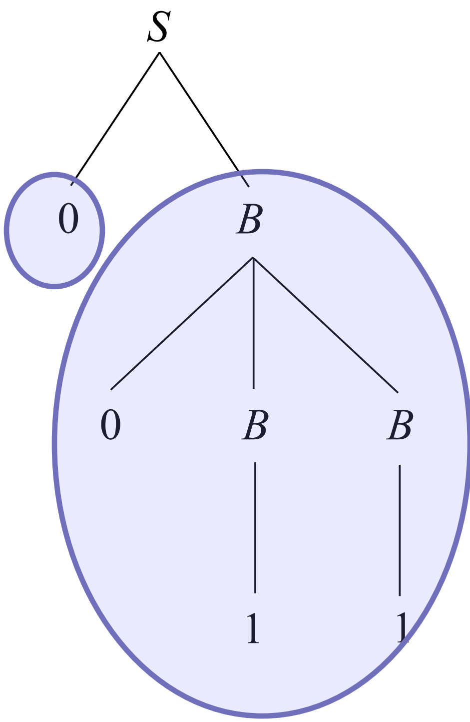
- La **definizione rigorosa** di “albero” è la seguente:
  - Un *albero T* è un insieme finito, non vuoto di vertici (*nodi*) tali che:
    - esiste un vertice speciale, detto radice dell’albero;
    - **esiste una partizione** $T_1, T_2, ..., T_m$, con $m \geq 0$, degli altri vertici, tale che esista un ordinamento $T_1, T_2, ..., T_m$, e ogni sottoinsieme di vertici T_i, $1 \leq i \leq m$, sia a sua volta un albero.



Partizione

Definizioni preliminari

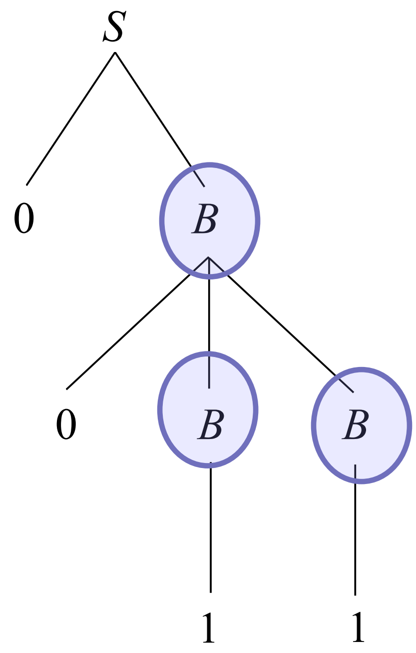
- La **definizione rigorosa** di “albero” è la seguente:
  - Un *albero T* è un insieme finito, non vuoto di vertici (*nodi*) tali che:
    - esiste un vertice speciale, detto radice dell’albero;
    - **esiste una partizione** $T_1, T_2, ..., T_m$, con $m \geq 0$, degli altri vertici, tale che esista un ordinamento $T_1, T_2, ..., T_m$, e ogni sottoinsieme di vertici T_i, $1 \leq i \leq m$, sia a sua volta un albero.
  - $T_1, T_2, ..., T_m$ sono detti **sottoalberi** di T. I vertici privi di discendenti sono le **foglie** dell’albero, gli altri sono i **nodi interni.**



Esempi di sottoalberi

Definizioni preliminari

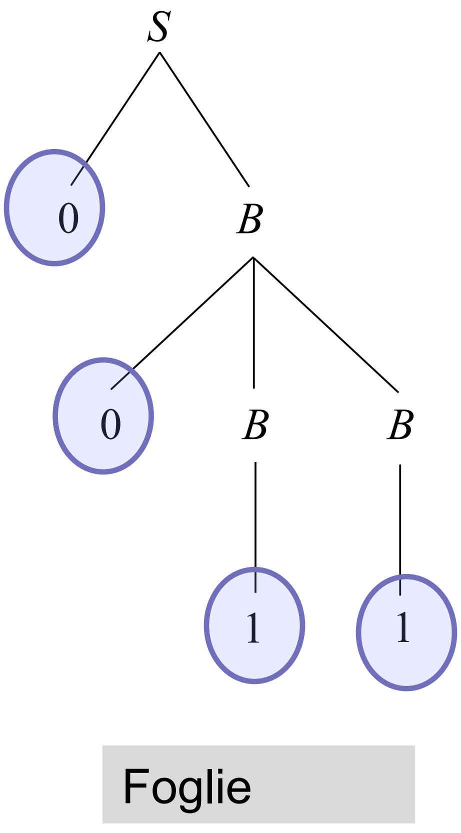
- La **definizione rigorosa** di “albero” è la seguente:
  - Un *albero T* è un insieme finito, non vuoto di vertici (*nodi*) tali che:
    - esiste un vertice speciale, detto radice dell’albero;
    - **esiste una partizione** $T_1, T_2, \ldots, T_m$, con $m \geq 0$, degli altri vertici, tale che esista un ordinamento $T_1, T_2, \ldots, T_m$, e ogni sottoinsieme di vertici T_i, $1 \leq i \leq m$, sia a sua volta un albero.
  - $T_1, T_2, \ldots, T_m$ sono detti **sottoalberi** di T. I vertici privi di discendenti sono le **foglie** dell’albero, gli altri sono i **nodi interni.**



Nodi interni

Definizioni preliminari

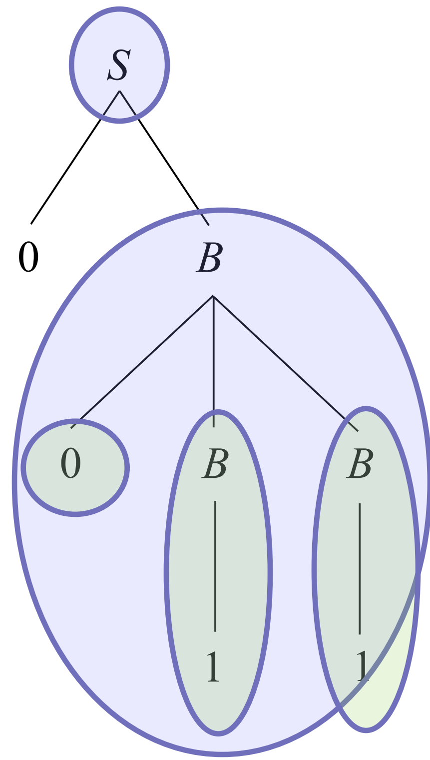
- La **definizione rigorosa** di “albero” è la seguente:
  - Un *albero T* è un insieme finito, non vuoto di vertici (*nodi*) tali che:
    - esiste un vertice speciale, detto radice dell’albero;
    - **esiste una partizione** $T_1, T_2, \ldots, T_m$, con $m \geq 0$, degli altri vertici, tale che esista un ordinamento $T_1, T_2, \ldots, T_m$, e ogni sottoinsieme di vertici T_i, $1 \leq i \leq m$, sia a sua volta un albero.
  - $T_1, T_2, \ldots, T_m$ sono detti **sottoalberi** di T. I vertici privi di discendenti sono le **foglie** dell’albero, gli altri sono i **nodi interni.**



Definizioni preliminari

- 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** $T_1, T_2, ..., T_m$, con $m \geq 0$, degli altri vertici, tale che esista un ordinamento $T_1, T_2, ..., T_m$, e ogni sottoinsieme di vertici T_i, $1 \leq i \leq m$, sia a sua volta un albero.
  - $T_1, T_2, ..., T_m$ sono detti **sottoalberi** di *T*. I vertici privi di discendenti sono le **foglie** dell’albero, gli altri sono i **nodi interni.**

    Questa **definizione è di tipo ricorsivo**, ma non è circolare in quanto ogni sottoalbero T_i contiene meno vertici dell’albero *T*.

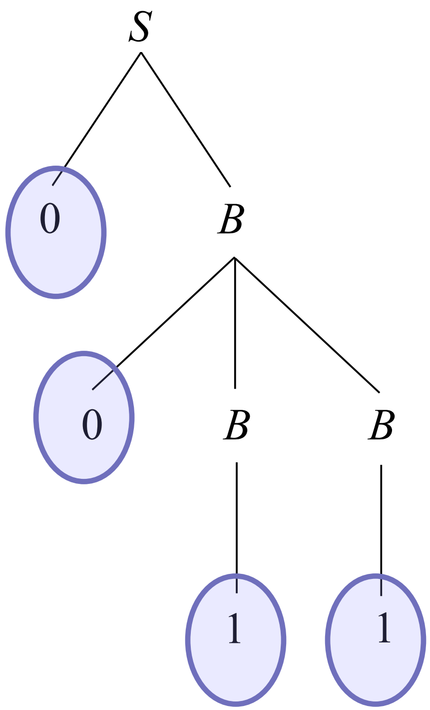


Esempi di sottoalberi

Definizioni preliminari

- 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** $T_1, T_2, \ldots, T_m$, con $m \geq 0$, degli altri vertici, tale che esista un ordinamento $T_1, T_2, \ldots, T_m$, 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.
    $T_1, T_2, \ldots, T_m$ 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.**



Frontiera

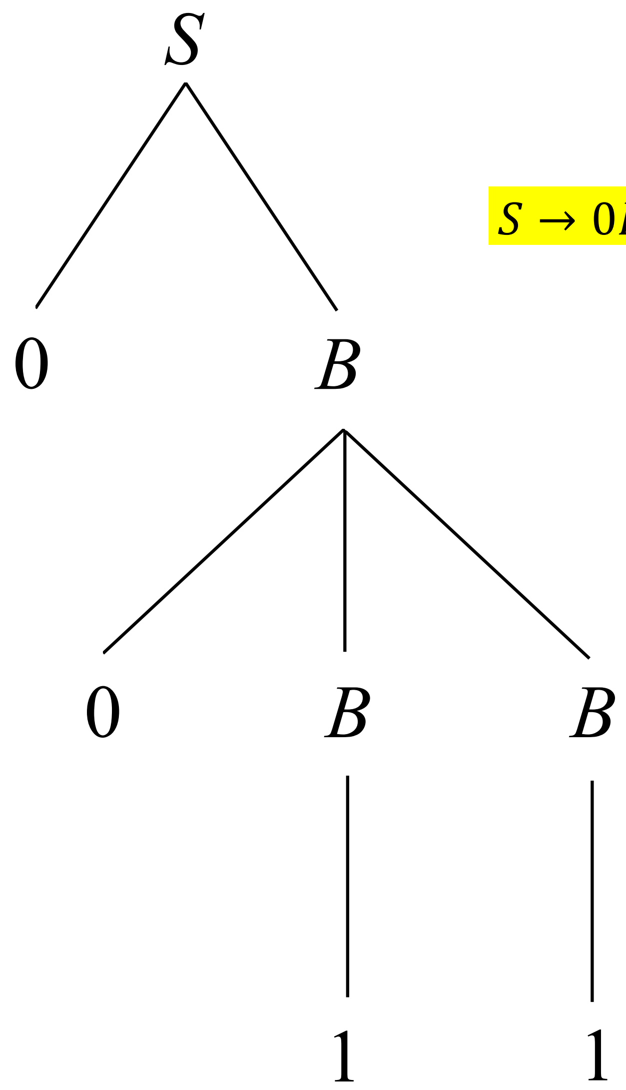
Alberi di derivazione

- Per cosa utilizziamo gli alberi?

Alberi di derivazione

- Per cosa utilizziamo gli alberi?
  - 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 $S \overset{*}{\Rightarrow} w$.
  - Al fine di disporre di una rappresentazione univoca, si preferisce ricorrere agli alberi di derivazione.

Alberi di derivazione



$S \to 0B \to 0BB \to 0011$

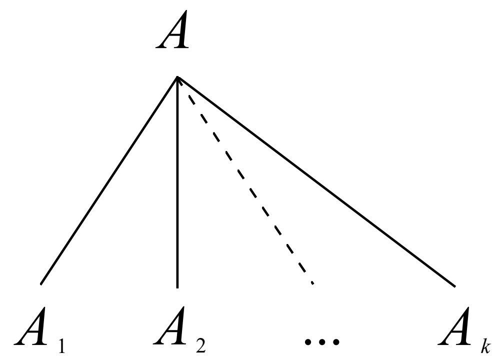
Definizione di albero di derivazione

- Sia $G = (X, V, S, P)$ una grammatica C.F. e $w \in X^*$ una stringa derivabile da S in G, $S \overset{*}{\Rightarrow} w$.
  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 $N_1, N_2, \ldots, N_k$, etichettati con i simboli $A_1, A_2, \ldots, A_k$, rispettivamente, allora la produzione:

$$A \rightarrow A_1 A_2 \ldots A_k$$

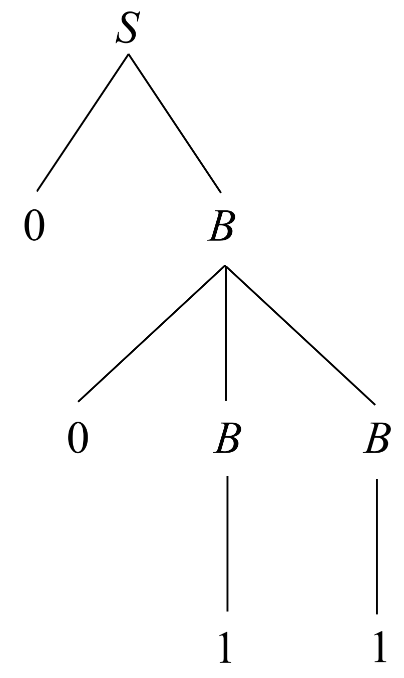


deve appartenere a *P*;

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

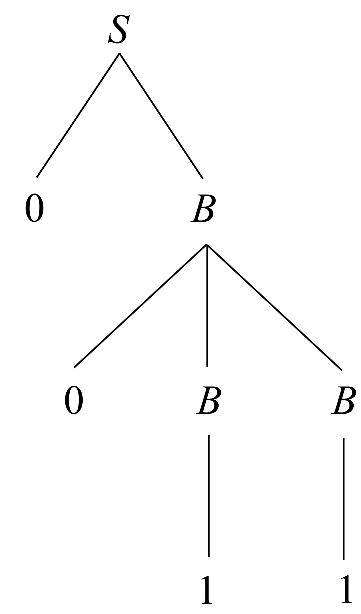
Definizioni

- **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

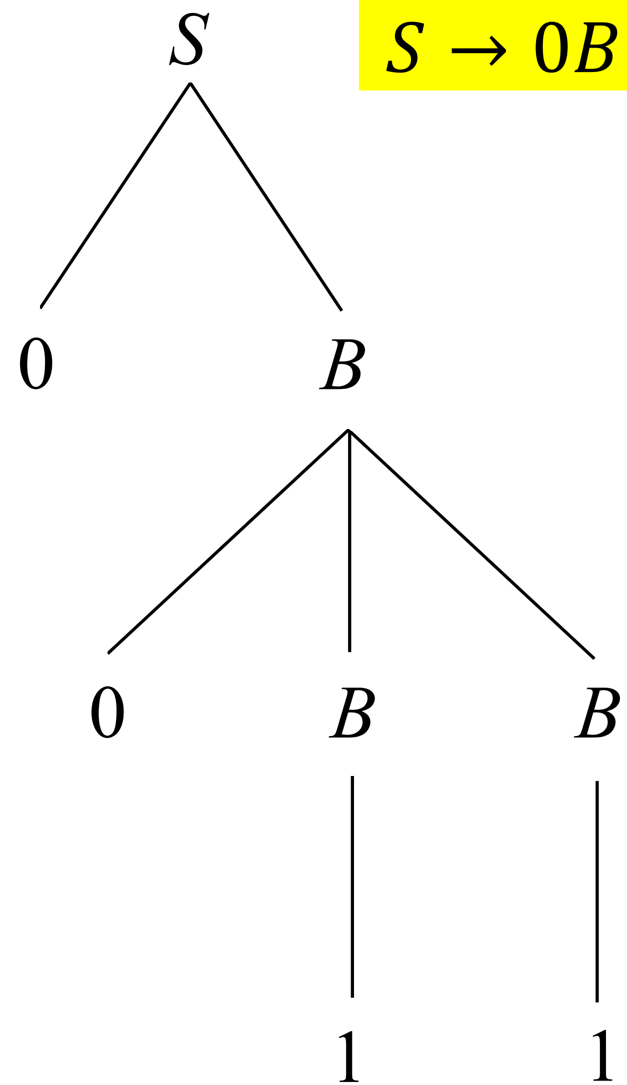


Definizioni

- **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.
  - **Il cammino da S a 1 ha lunghezza 3 (i non terminali sono S, B, B)**
- **Altezza** o profondità
  - L'*altezza* di un albero è data dalla lunghezza del suo cammino più lungo.
  - **L'albero nell'esempio ha altezza 3**



Osservazione



$S \to 0B \to 0BB \to 0011$

Data una derivazione, **quanti alberi** possono rappresentarla?

Dato un albero, **quante derivazioni** può rappresentare?

Esempio 4.1 (cf. Libro)

Esempio 4.1

Si consideri la seguente grammatica libera da contesto:

$$G_1 = (X, V, S, P)$$

dove:

$$X = \{a\} \qquad V = \{S, H\}$$

$$P = \left\{ S \xrightarrow{(1)} Ha,\ H \xrightarrow{(2)} HS,\ H \xrightarrow{(3)} a \right\}$$

La stringa $w = aaaa$ è derivabile da S in G_1 L'albero di derivazione di $w = aaaa$ è rappresentato in Figura 4.2:

Esempio 4.1 (cf. Libro)

$$P = \left\{ S \overset{(1)}{\rightarrow} Ha,\ H \overset{(2)}{\rightarrow} HS,\ H \overset{(3)}{\rightarrow} a \right\}$$

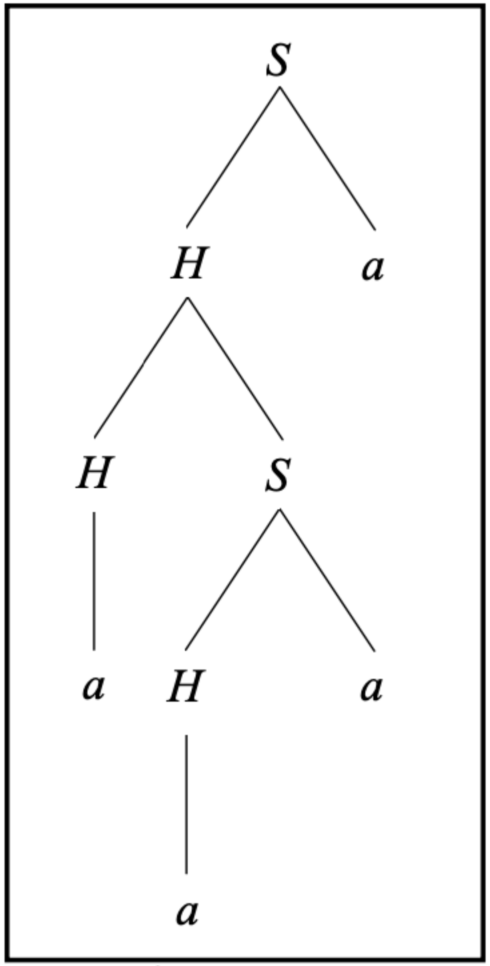
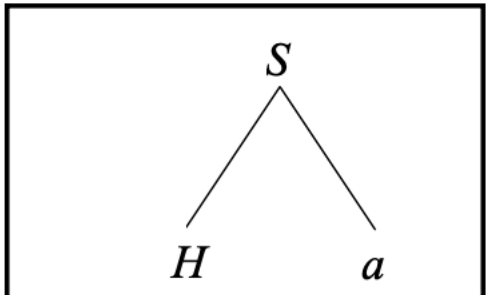


Figura 4.2

Esempio 4.1 (cf. Libro)

$$P = \left\{ S \overset{(1)}{\rightarrow} Ha,\ H \overset{(2)}{\rightarrow} HS,\ H \overset{(3)}{\rightarrow} a \right\}$$



ero corrisponde sia alla *derivazione sinistra*:

$$S \underset{(1)}{\Rightarrow} Ha \underset{(2)}{\Rightarrow} HSa \underset{(3)}{\Rightarrow} aSa \underset{(1)}{\Rightarrow} aHaa \underset{(3)}{\Rightarrow} aaaa$$

lerivazione destra:

$$S \underset{(1)}{\Rightarrow} Ha \underset{(2)}{\Rightarrow} HSa \underset{(1)}{\Rightarrow} HHaa \underset{(3)}{\Rightarrow} Haaa \underset{(3)}{\Rightarrow} aaaa$$

Figura 4.2

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**,
  - Dato un **albero di derivazione, esso rappresenta più derivazioni** (in funzione dell'ordine col quale si espandono i nonterminali).

(Esempio 4.1 sul libro)

Principio di sostituzione di sottoalberi

- Definizione di derivazione destra (sinistra)
  - Data una grammatica *G* = (*X*, *V*, *S*, *P*), diremo che una *derivazione* $S \overset{*}{\Rightarrow} w$, ove:

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

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

è *destra* (*sinistra*) se, per ogni *i*, *i* = 1, 2,..., *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.

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

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

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

  Consideriamo la stringa 0011.
- Come possiamo derivarla?

Esempio

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

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

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

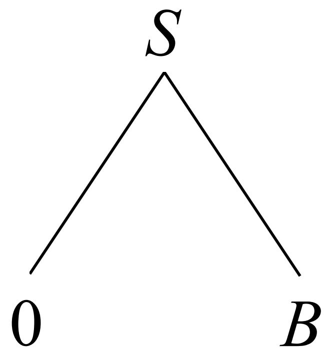
$$B \to 1 \mid 1S \mid 0BB$$

  Consideriamo la stringa 0011. Una possibile derivazione è: $S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 001B \Rightarrow 0011$
  Un'altra possibile è: $S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 00B1 \Rightarrow 0011$
- La prima è una **derivazione sinistra**, la seconda è una **derivazione destra.**

Esempio

$$S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 001B \Rightarrow 0011$$

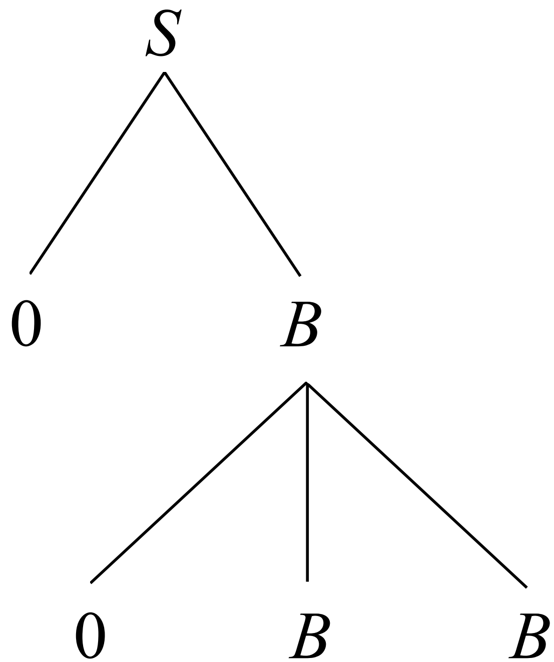
$$S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 00B1 \Rightarrow 0011$$



Esempio

$$S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 001B \Rightarrow 0011$$

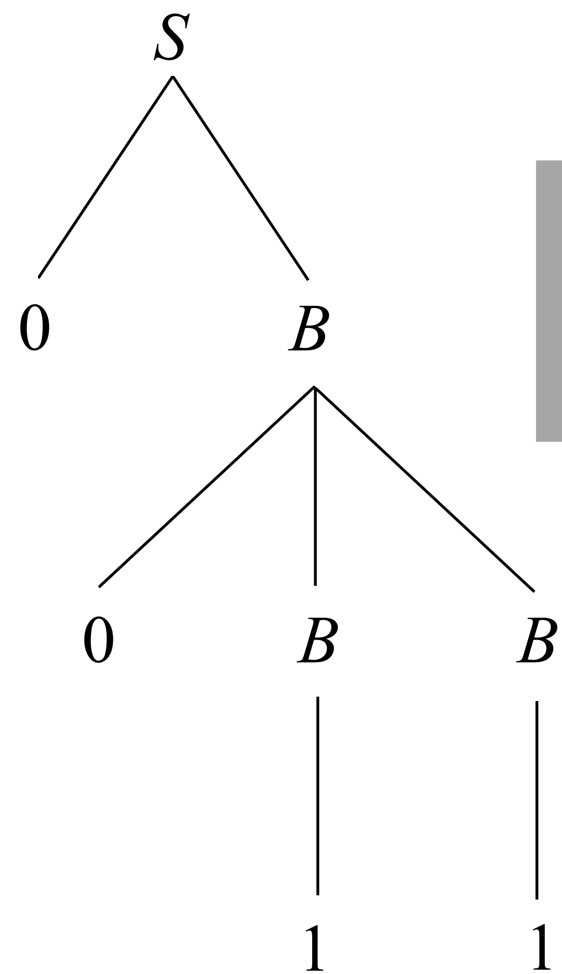
$$S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 00B1 \Rightarrow 0011$$



Esempio

$$S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 001B \Rightarrow 0011$$

$$S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 00B1 \Rightarrow 0011$$



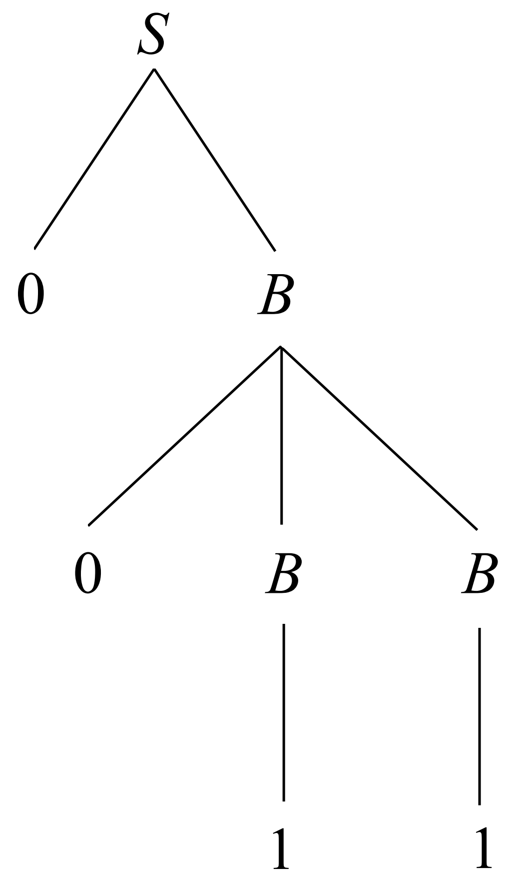
Il non determinismo insito nella derivazione scompare quando si considera il relativo albero di derivazione

Obiettivo

- L’obiettivo di questo modulo è riuscire a individuare **una caratteristica comune a tutti i linguaggi liberi da contesto**
- Perché?
  - Perché se abbiamo una grammatica che genera il linguaggio, possiamo facilmente associare a un linguaggio la sua classe di riferimento
  - Se non abbiamo la grammatica, come facciamo?
  - Si ragiona per assurdo, e se se si verifica che il linguaggio che vogliamo studiare non ha questa «caratteristica», si dimostra che un linguaggio non è libero da contesto

Principio di sostituzione di sottoalberi

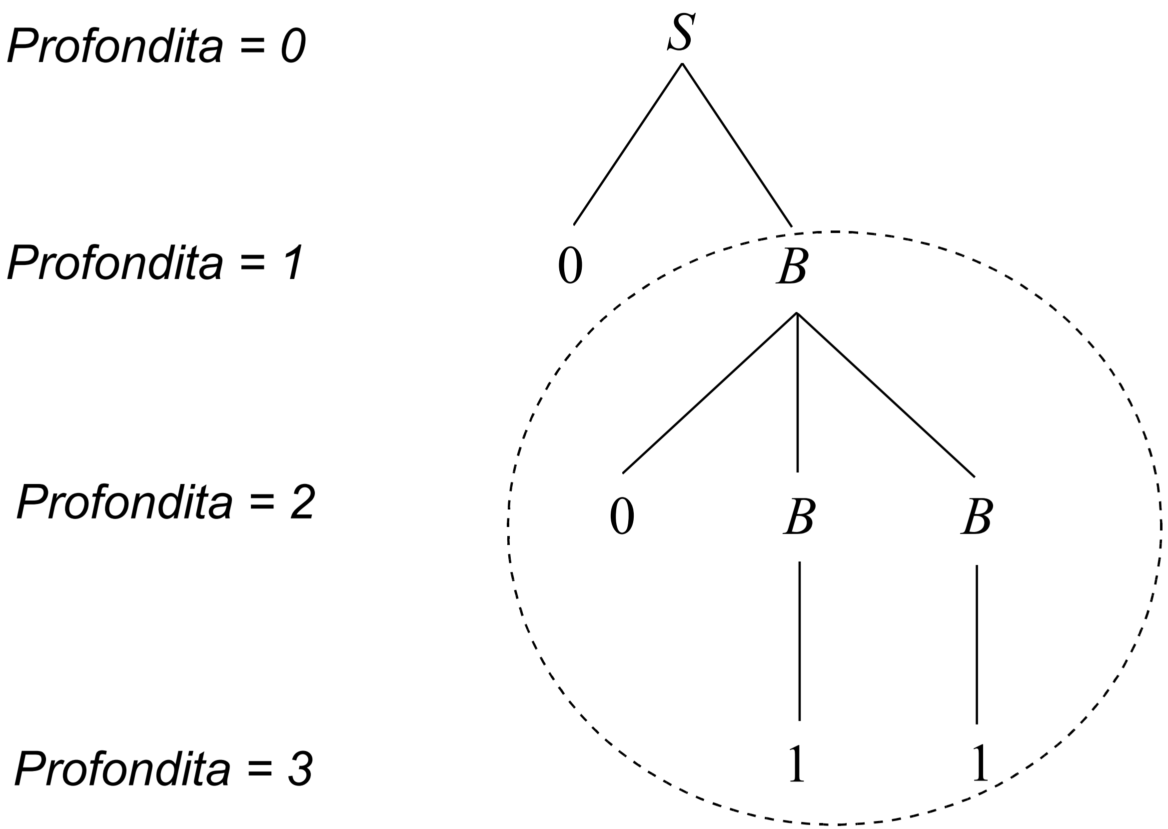
- Consideriamo ora il sottoalbero con radice nel nodo di profondità minore etichettato con una *B*.



Quanti sottoalberi etichettati con B?

Principio di sostituzione di sottoalberi

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

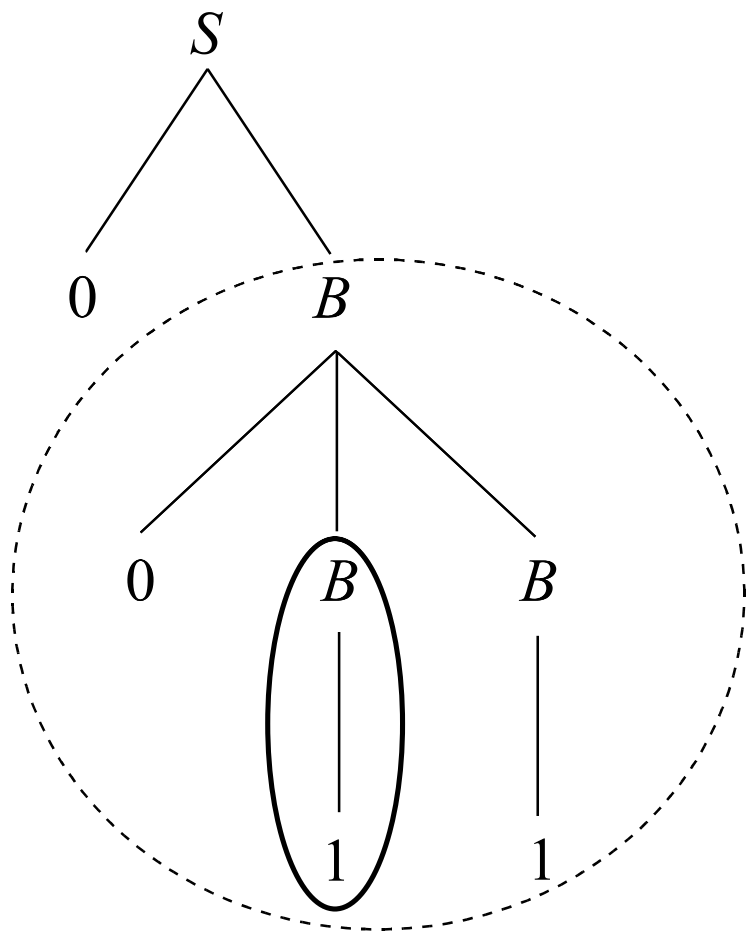


3 sottoalberi, uno a profondità 1 e due a profondità

Consideriamo quello a profondità 1 (minore)

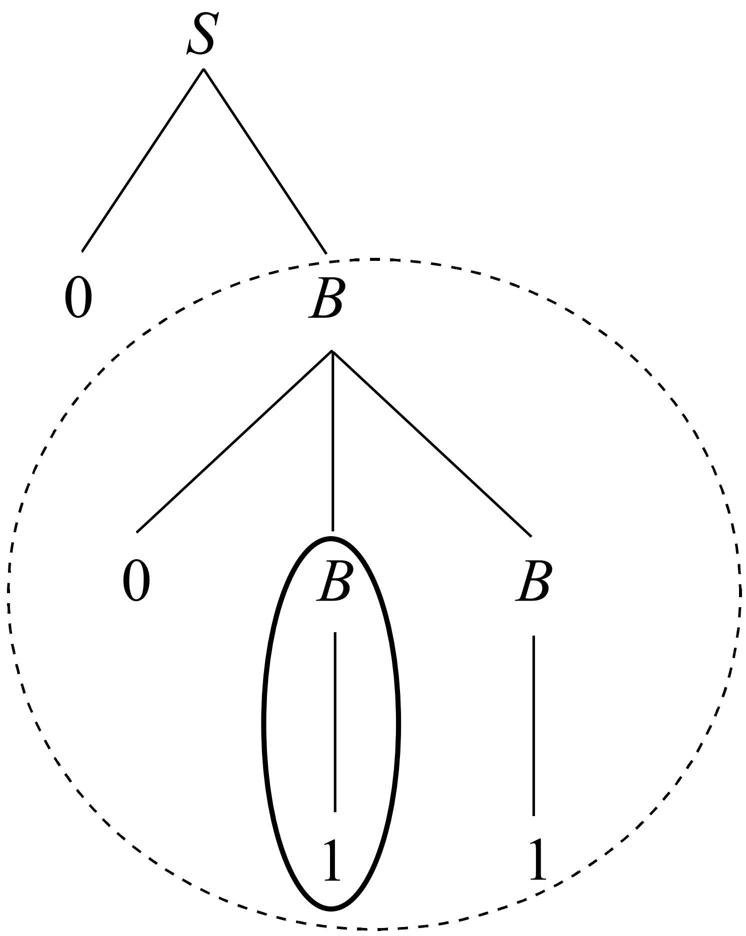
Principio di sostituzione di sottoalberi

- Cosa accade se un qualsiasi sottoalbero con radice in un nodo **etichettato con una *B*** viene sostituito con il sottoalbero ?

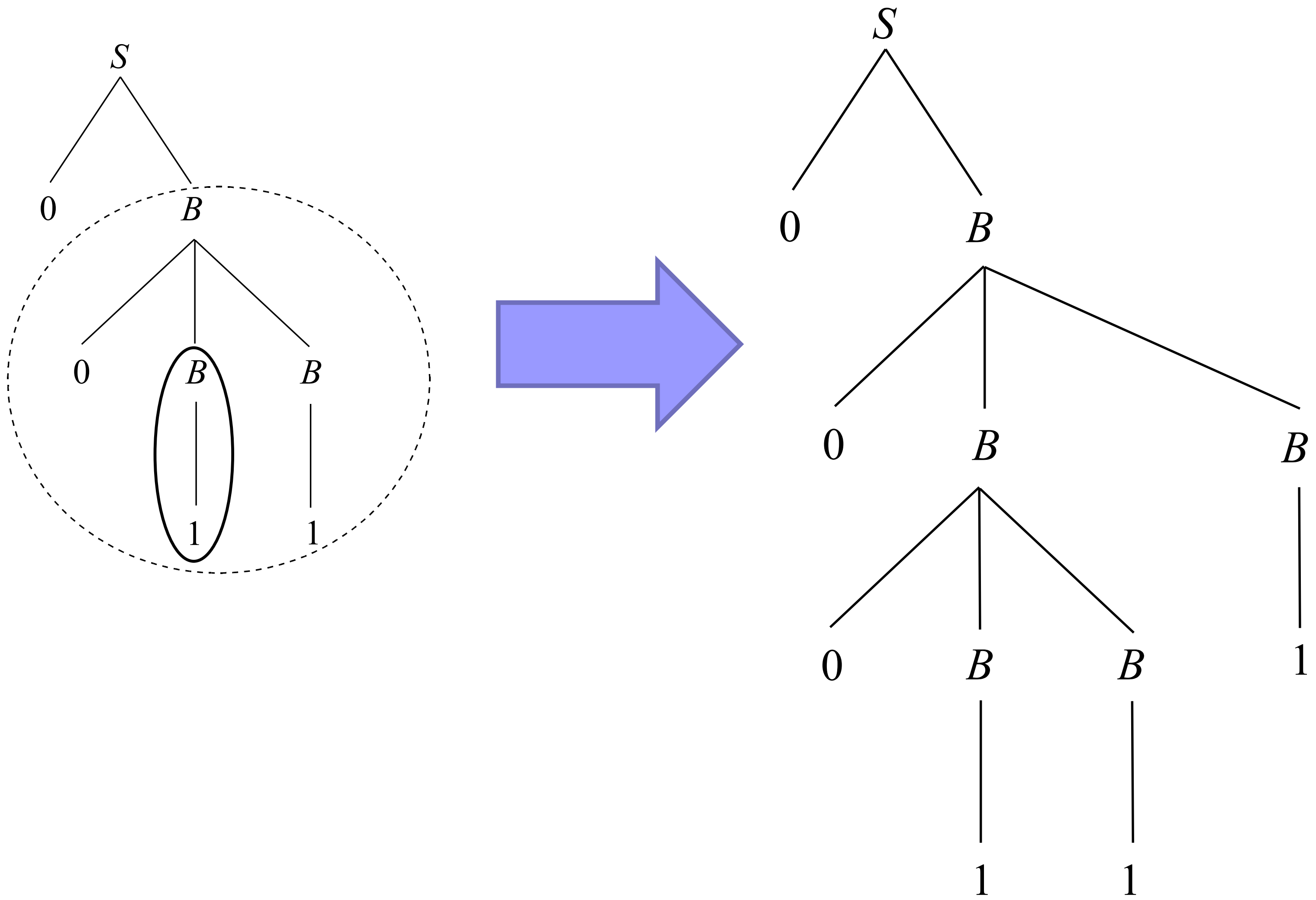


Principio di sostituzione di sottoalberi

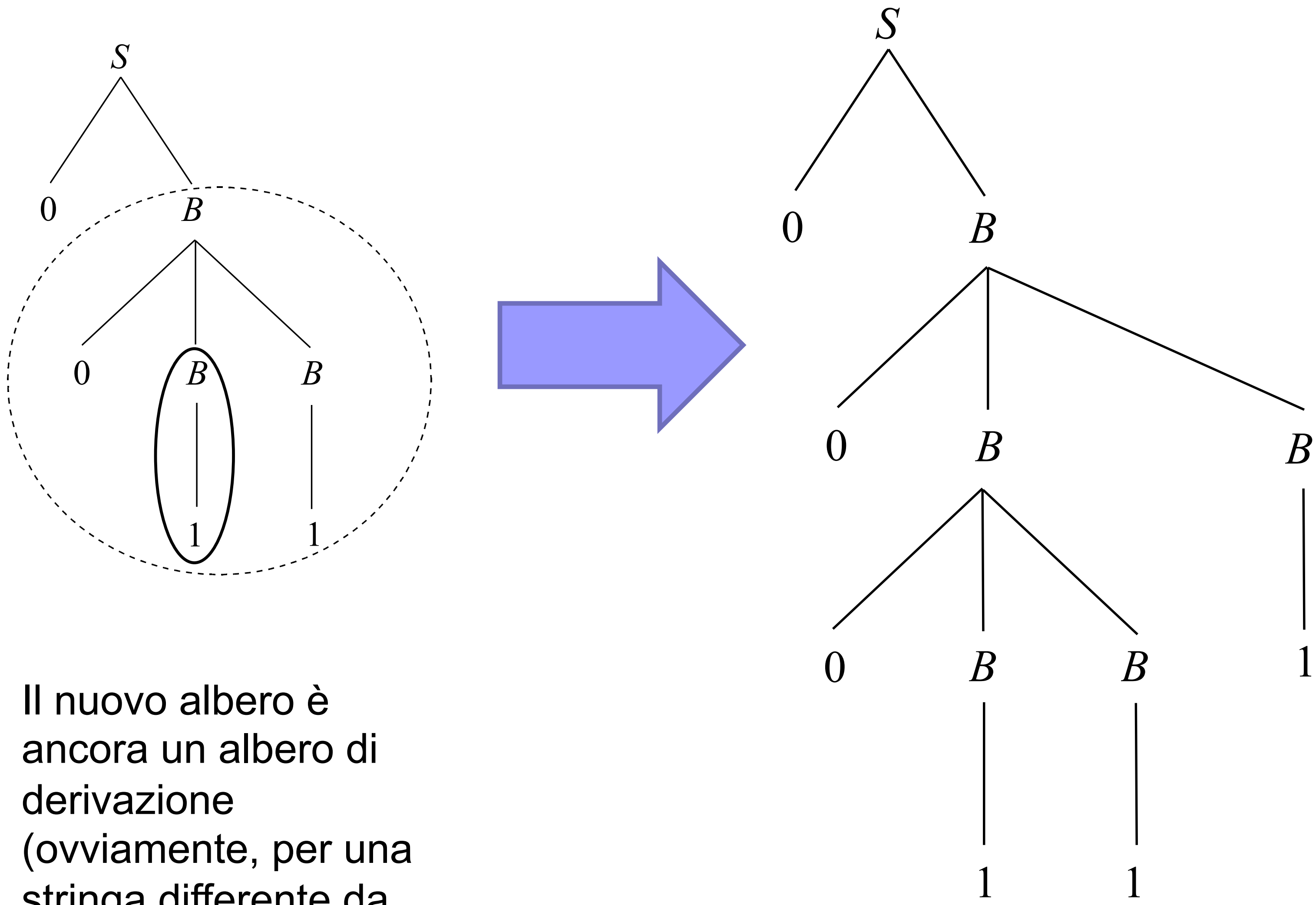
- Cosa accade se un qualsiasi sottoalbero con radice in un nodo **etichettato con una *B*** viene sostituito con il sottoalbero ?
- Consideriamo, ad esempio, il sottoalbero individuato dal cerchio pieno nella figura. **Lo sostituisco con il sottoalbero denotato dalle linee tratteggiate.**
- **Cosa ottengo?**



Principio di sostituzione di sottoalberi

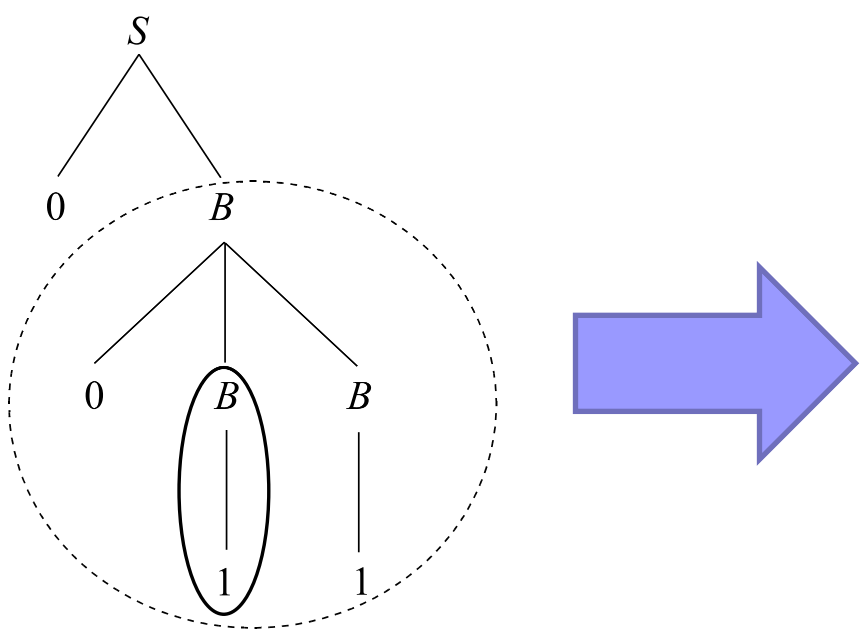


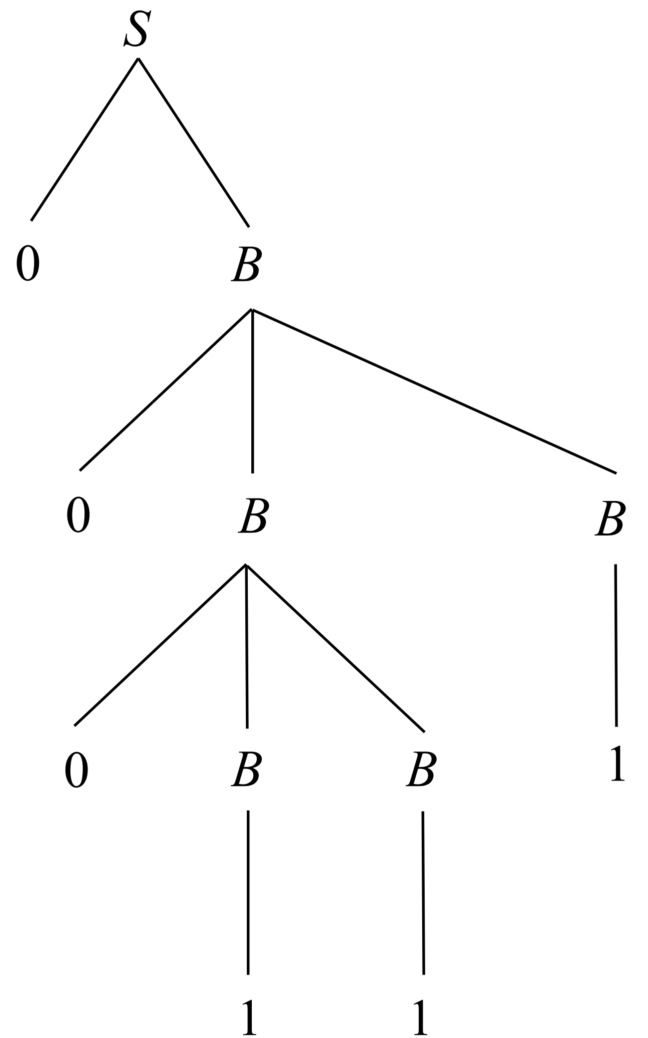
Principio di sostituzione di sottoalberi



Il nuovo albero è ancora un albero di derivazione (ovviamente, per una stringa differente da 0011).

Principio di sostituzione di sottoalberi





- per cui:

$$S \overset{*}{\Rightarrow} 0011$$

Ma anche

$$S \overset{*}{\Rightarrow} 000111$$

Principio di sostituzione di sottoalberi

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

$$
\begin{array}{rll}
 & 0011 & |w| = 4 \\
 & 000111 & |w| = 6 \\
w = & 00001111 & |w| = 8 \\
 & \vdots & \vdots \\
 & 00^n11^n & |w| = 2n+2 \quad n = 1, 2, \ldots
\end{array}
$$

Come cresce la lunghezza delle parole?

Principio di sostituzione di sottoalberi

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

$$
\begin{array}{rll}
 & 0011 & |w| = 4 \\
 & 000111 & |w| = 6 \\
w = & 00001111 & |w| = 8 \\
 & \vdots & \vdots \\
 & 00^n11^n & |w| = 2n+2 \quad n = 1, 2, \ldots
\end{array}
$$

La lunghezza cresce in **maniera costante**.

- **Questa è la «caratteristica» che stiamo cercando!**

Principio di sostituzione di sottoalberi

- Nelle grammatiche C.F., dunque, possiamo sostituire alberi **più piccoli con alberi di dimensioni maggiori, e ottenere ancora parole derivabili del linguaggio**
- **A che condizione?**
  - **Purché abbiano la stessa radice** - più precisamente, purché i nodi radice dei due sotto-alberi siano etichettati con lo stesso *NT*

Principio di sostituzione di sottoalberi

- Nelle grammatiche C.F., dunque, possiamo sostituire alberi **più piccoli con alberi di dimensioni maggiori, e ottenere ancora parole derivabili del linguaggio**
- **A che condizione?**
  - **Purché abbiano la stessa radice** - più precisamente, purché i nodi radice dei due sotto-alberi siano etichettati con lo stesso *NT*
- Una caratteristica dei linguaggi C.F., che discende direttamente dal processo descritto in precedenza, **è che sono sempre caratterizzati da un sottoinsieme di parole la cui lunghezza cresce in maniera costante.**

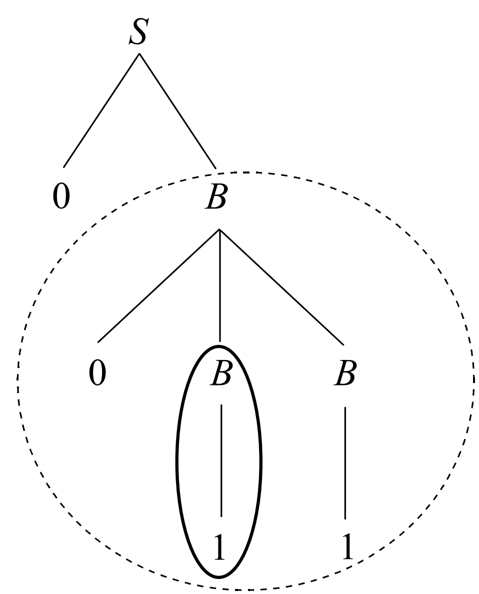
Controesempio

Non sempre la crescita costante è sinonimo di linguaggio C.F.!

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

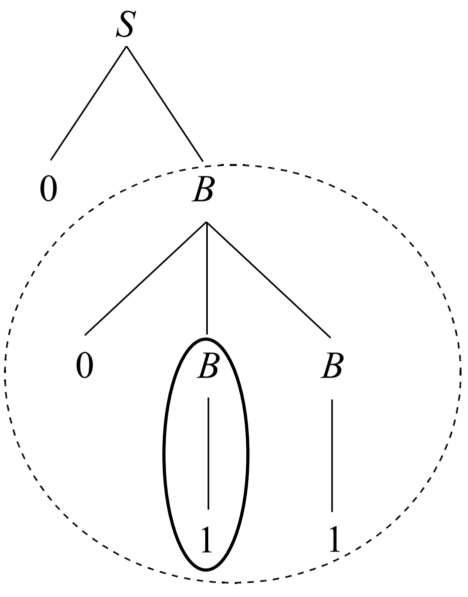
Principio di sostituzione di sottoalberi

- Generalizziamo ora il discorso
- Quale **caratteristica dell'albero** permette di applicare la sostituzione?



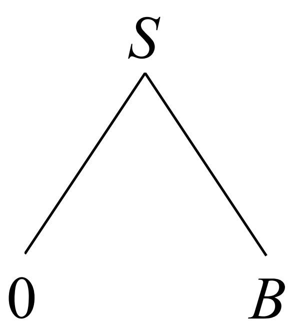
Principio di sostituzione di sottoalberi

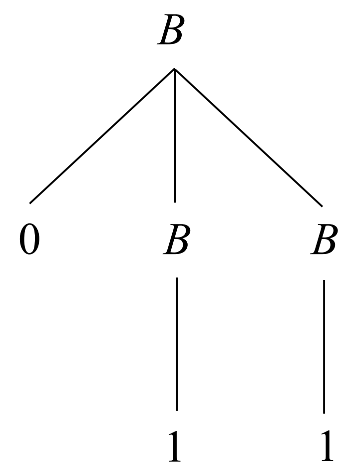
- Generalizziamo ora il discorso
- Quale **caratteristica dell'albero** permette di applicare la sostituzione?
  - La presenza di due NT uguali sullo **stesso cammino**
    - **Importante: sullo stesso cammino, non necessariamente connessi in modo diretto**

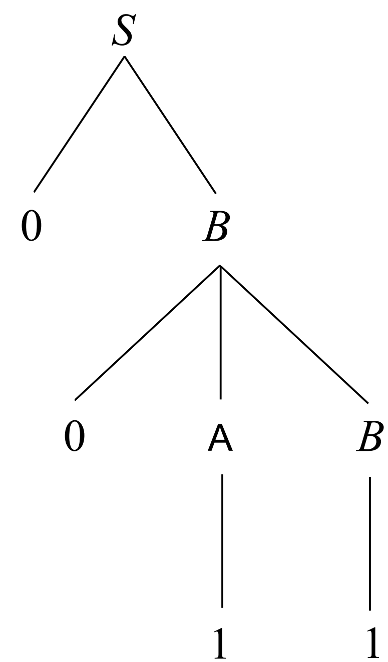


Principio di sostituzione di sottoalberi

- Generalizziamo ora il discorso
- Quale **caratteristica dell'albero** permette di applicare la sostituzione?
  - La presenza di due NT uguali sullo **stesso cammino**
    - **Importante: sullo stesso cammino, non necessariamente connessi in modo diretto**

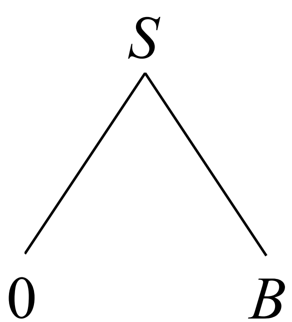




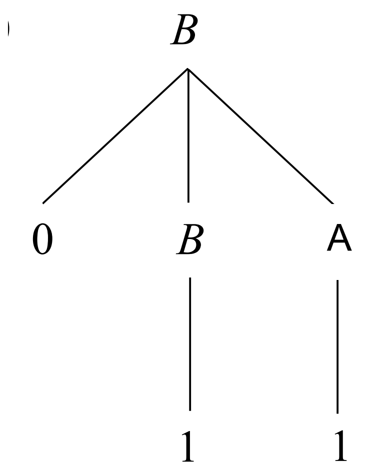


Principio di sostituzione di sottoalberi

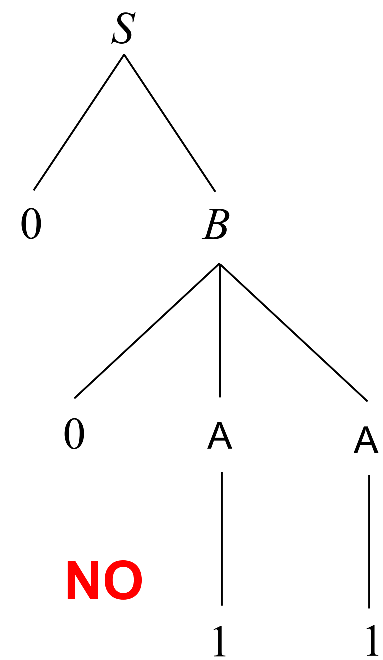
- Generalizziamo ora il discorso
- Quale **caratteristica dell'albero** permette di applicare la sostituzione?
  - La presenza di due NT uguali sullo **stesso cammino**
    - **Importante: sullo stesso cammino, non necessariamente connessi in modo diretto**



NO



SI



NO

Principio di sostituzione di sottoalberi

- Generalizziamo ora il discorso
- Quale **caratteristica dell'albero** permette di applicare la sostituzione?
  - La presenza di due NT uguali sullo **stesso cammino**

- Dunque
  - 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.
    - **Domanda:** quale è l'altezza minima del sotto-albero che garantisce questa condizione?

Principio di sostituzione di sottoalberi

- Generalizziamo ora il discorso
- Quale **caratteristica dell'albero** permette di applicare la sostituzione?
  - La presenza di due NT uguali sullo **stesso cammino**
- Dunque
  - 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.
    - **Domanda:** quale è l'altezza minima del sotto-albero che garantisce questa condizione? → $|V| + 1$
    - **Se L'altezza è pari al numero dei non terminali + 1, abbiamo la garanzia di una ripetizione**

Principio di sostituzione di sottoalberi

- Generalizziamo ora il discorso
- Quale **caratteristica dell'albero** permette di applicare la sostituzione?
  - La presenza di due NT uguali sullo **stesso cammino**

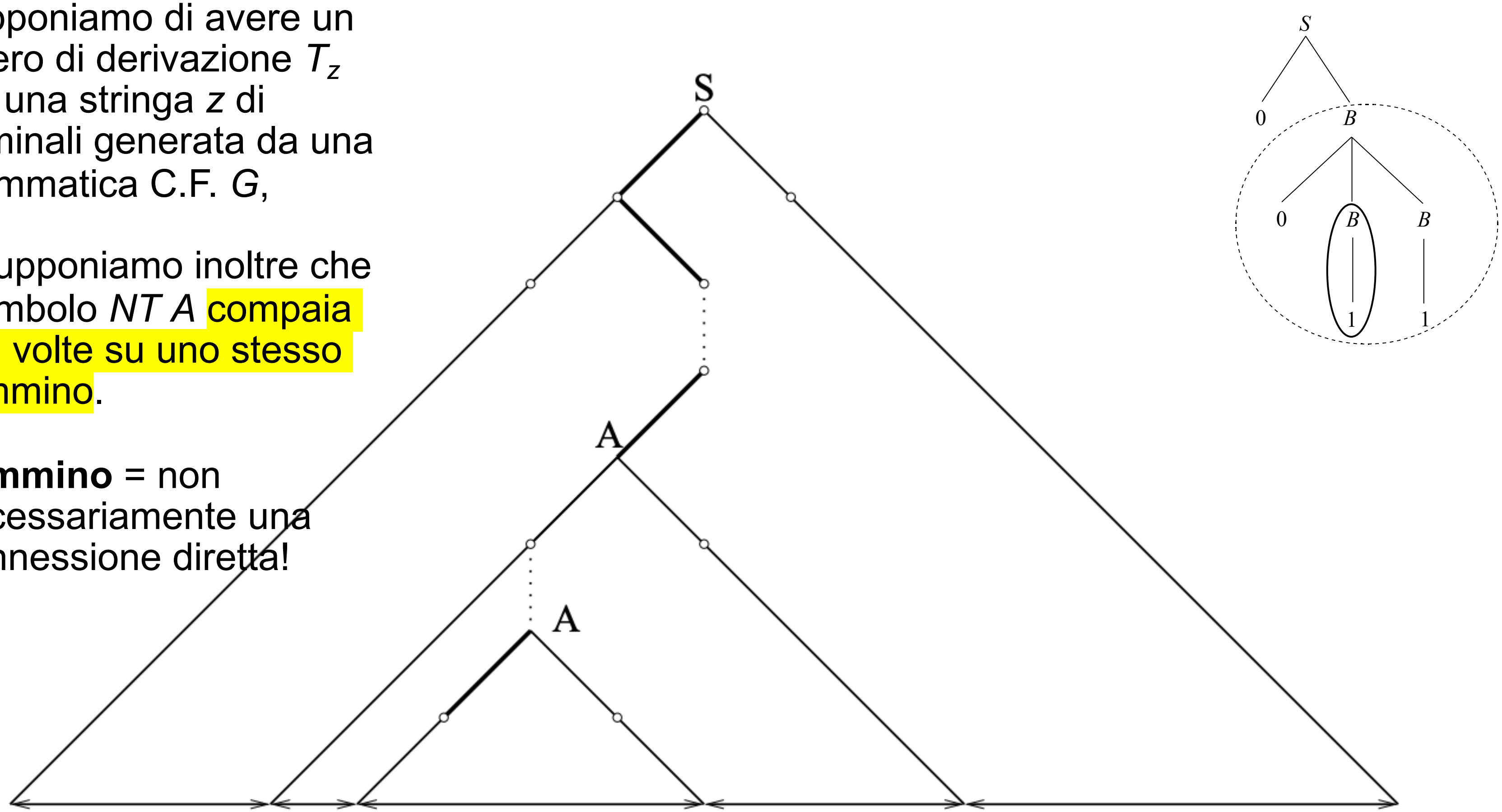
- Dunque
  - Supponiamo di avere un albero di derivazione T_z per una stringa *z* di terminali generata da una grammatica C.F. *G*,
  - E supponiamo inoltre che il simbolo *NT A* compaia due volte su uno stesso cammino.
    La situazione è rappresentata graficamente in Figura.

Principio di sostituzione dei sottoalberi

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.

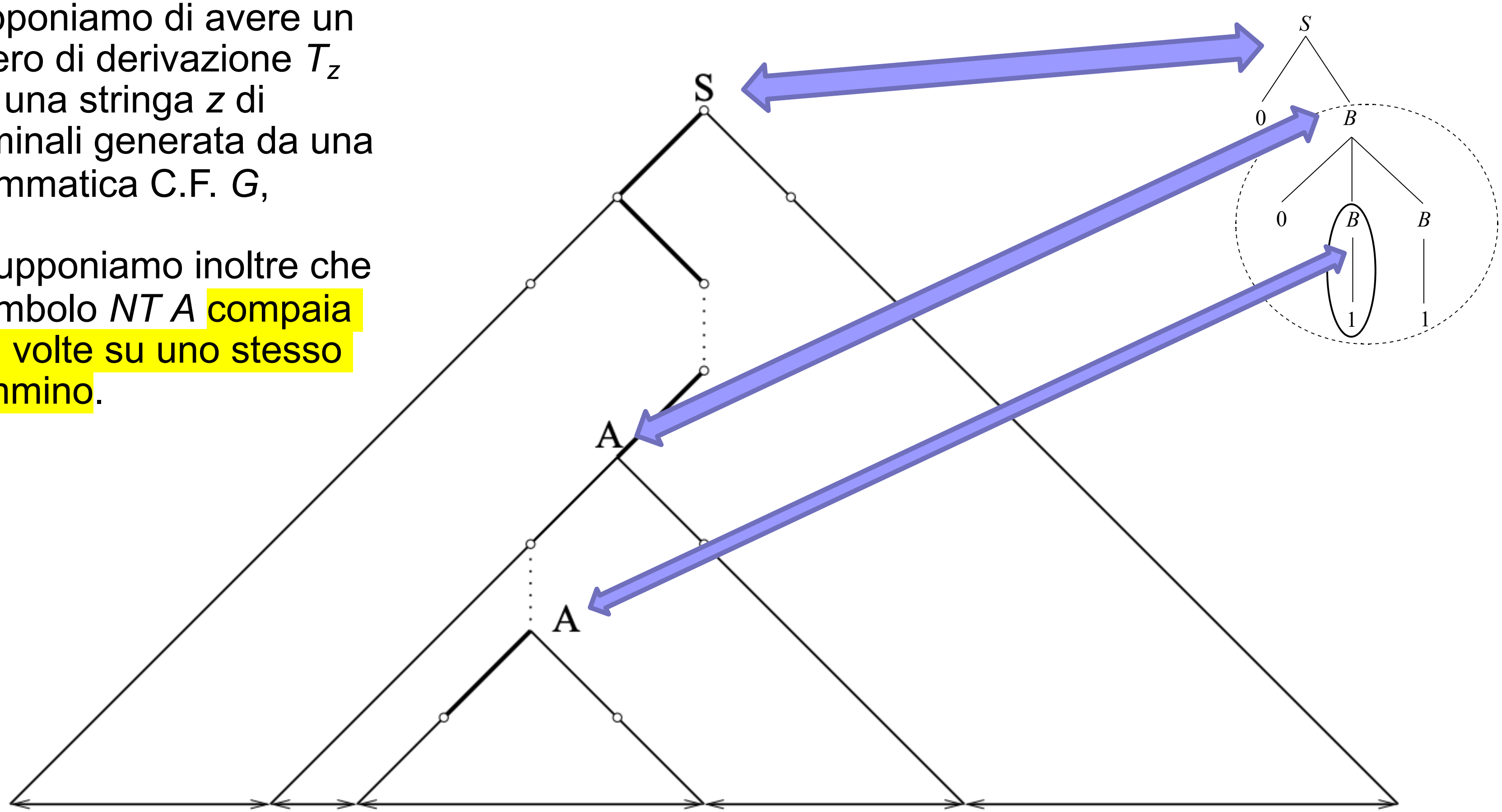
Cammino = non Necessariamente una Connessione diretta!



Principio di sostituzione dei sottoalberi

Supponiamo di avere un albero di derivazione T_z per una stringa z di terminali generata da una grammatica C.F. G,

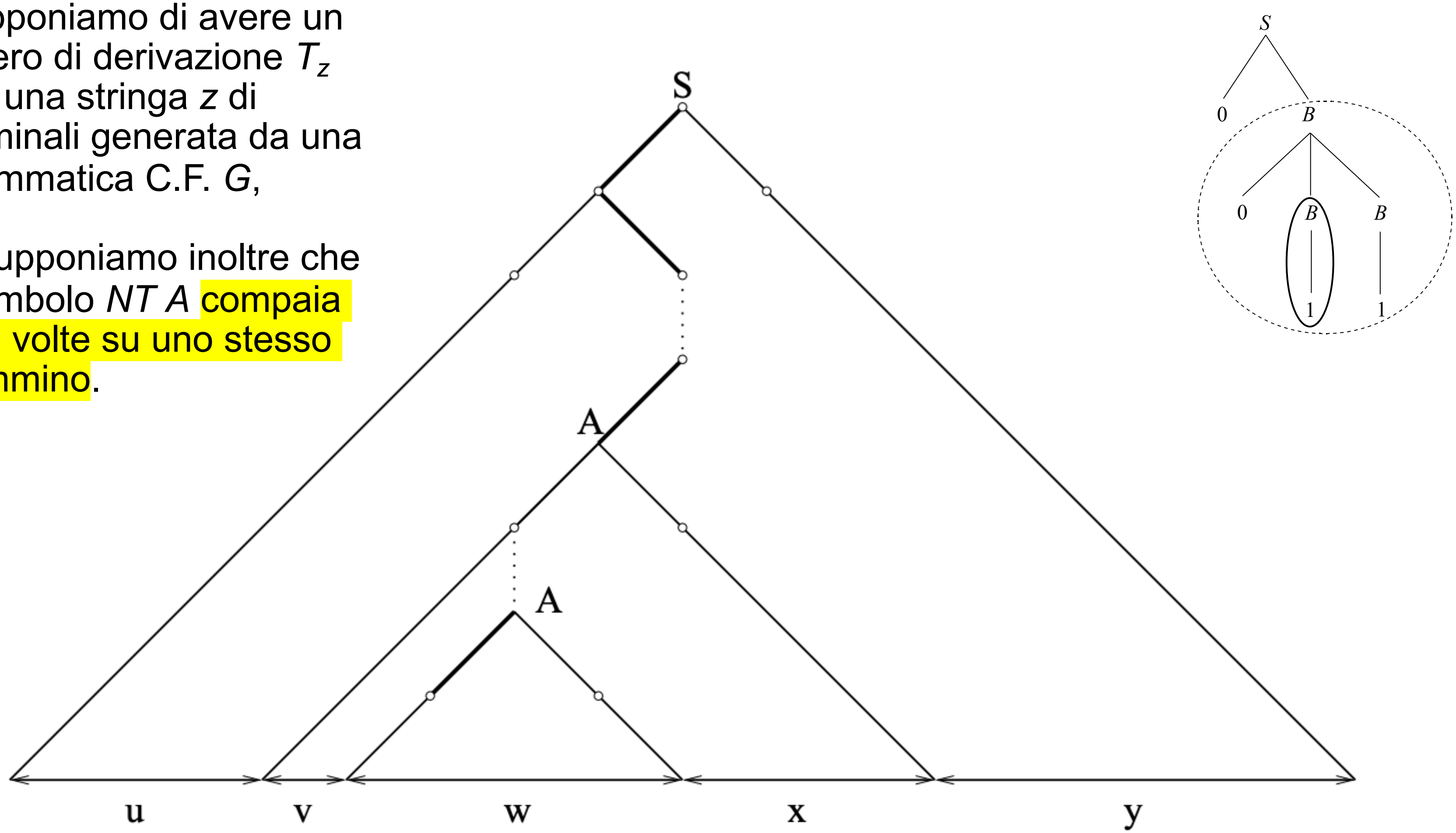
E supponiamo inoltre che il simbolo *NT A* compaia due volte su uno stesso cammino.



Principio di sostituzione dei sottoalberi

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.

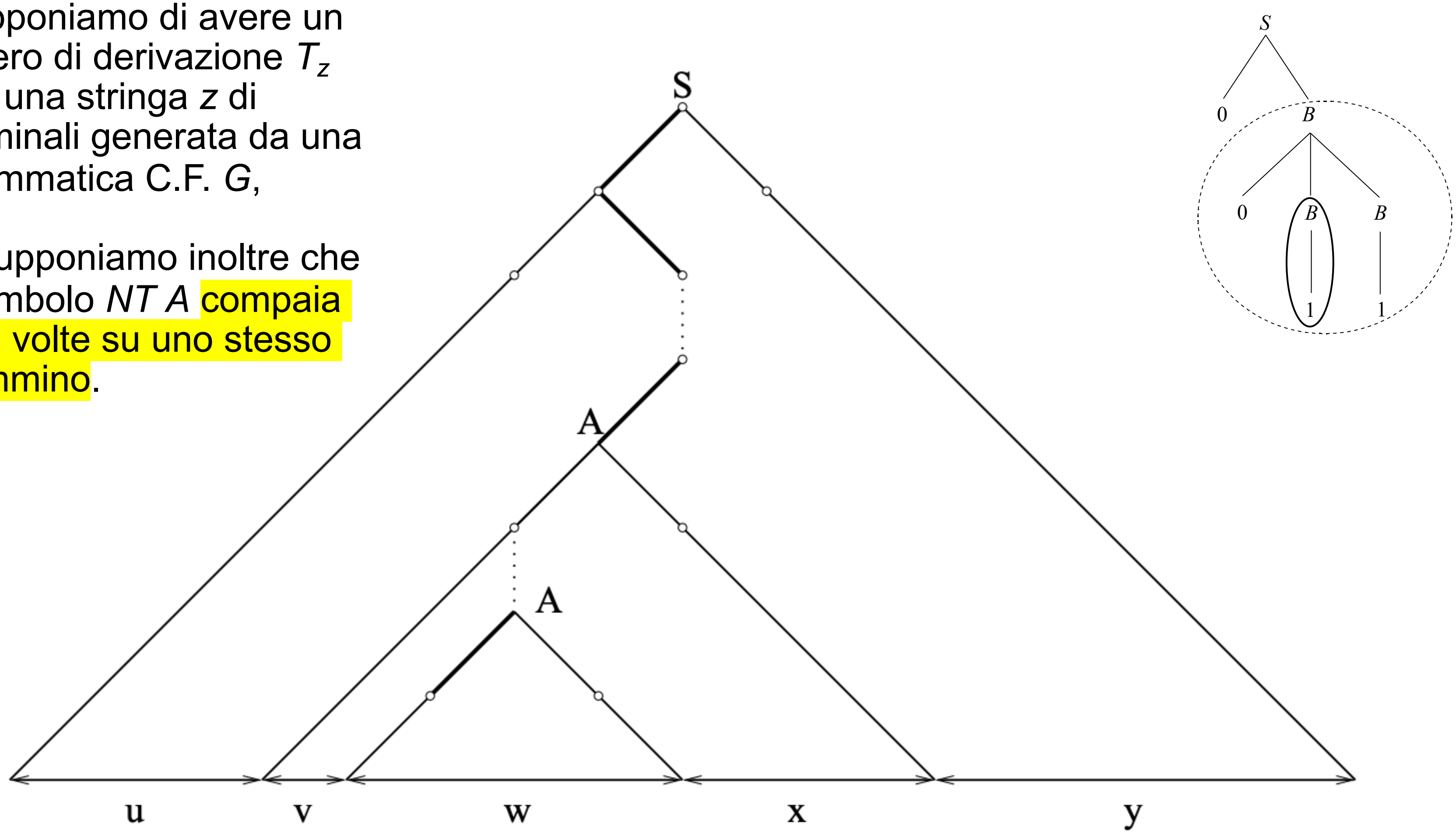


La stringa ottenuta mediante il processo può essere vista ha una struttura fissa. Un pezzo della stringa è ricavata più in basso (chiamato 'w'), che a sua volta è inclusa in quella ottenuta dal sottalbero più in alto ('v' e 'x') e altri due a partire dal simbolo di derivazione iniziale ('u' e 'y') (e non sono inclusi nel cammino)

Principio di sostituzione dei sottoalberi

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.

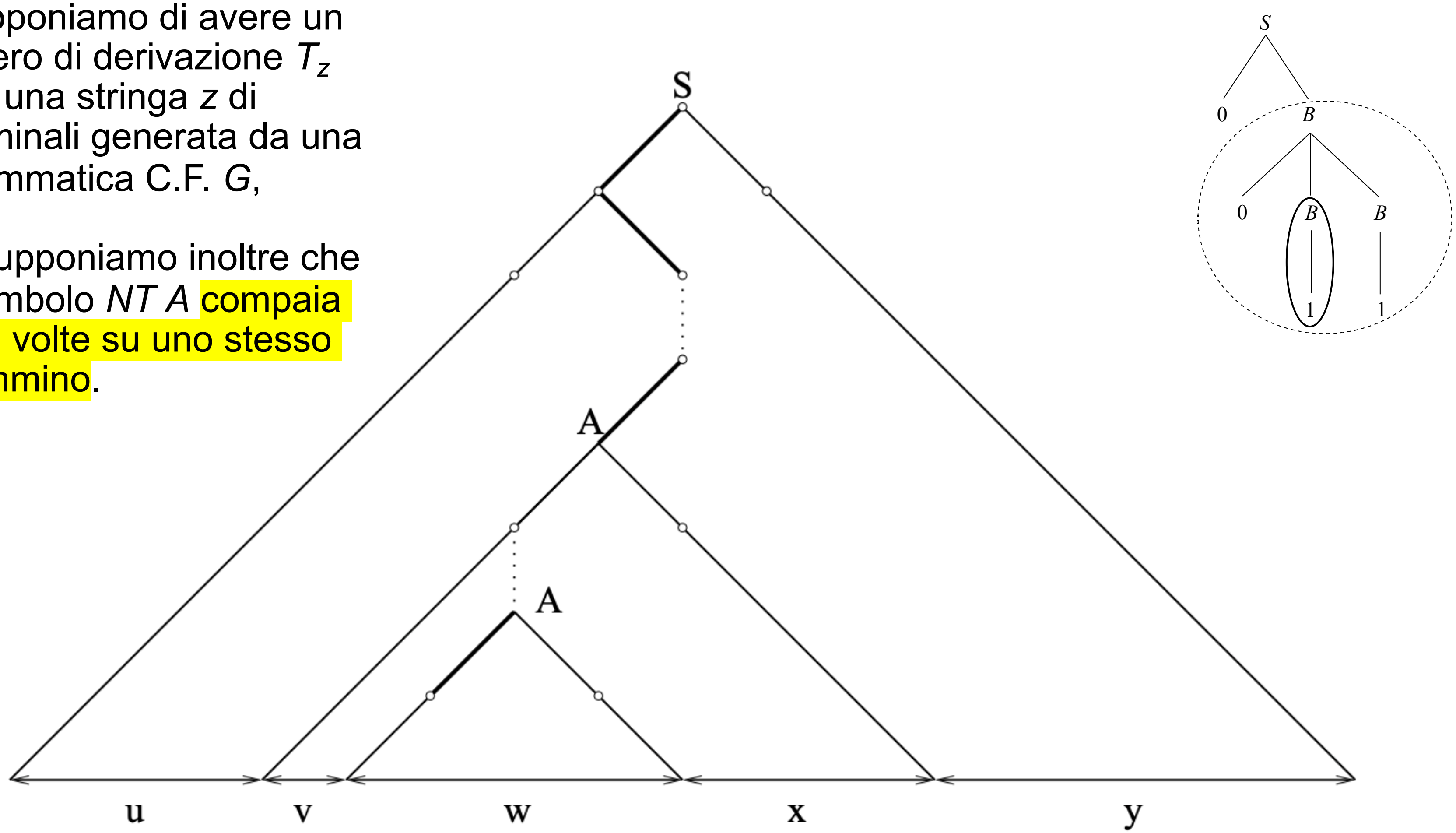


0 0 1 1
(in riferimento all'esempio precedente)

Principio di sostituzione dei sottoalberi

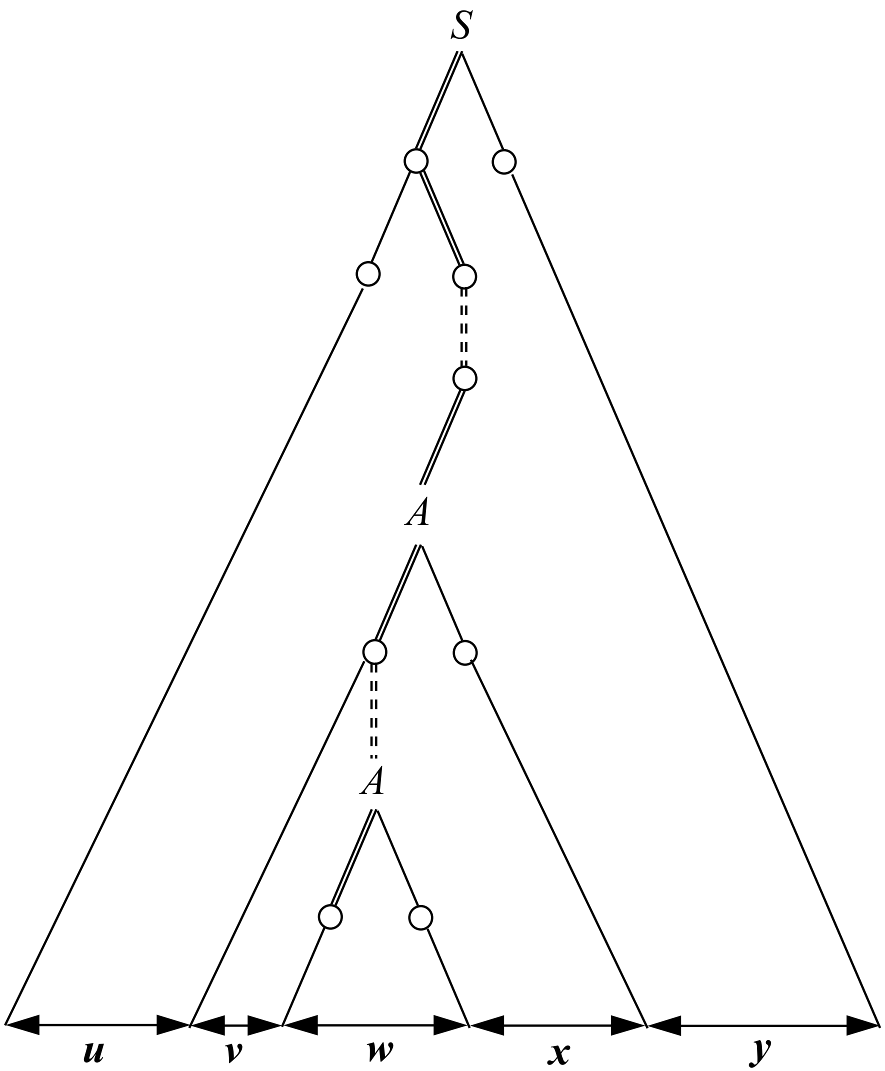
Supponiamo di avere un albero di derivazione T_z per una stringa *z* di terminali generata da una grammatica C.F. *G*,

E supponiamo inoltre che il simbolo *NT A* compaia due volte su uno stesso cammino.



Facendo riferimento all'esempio precedente, la w corrisponde a 0 e vwx corrisponde a 011. Lo zero rimanente finisce nella 'u'.

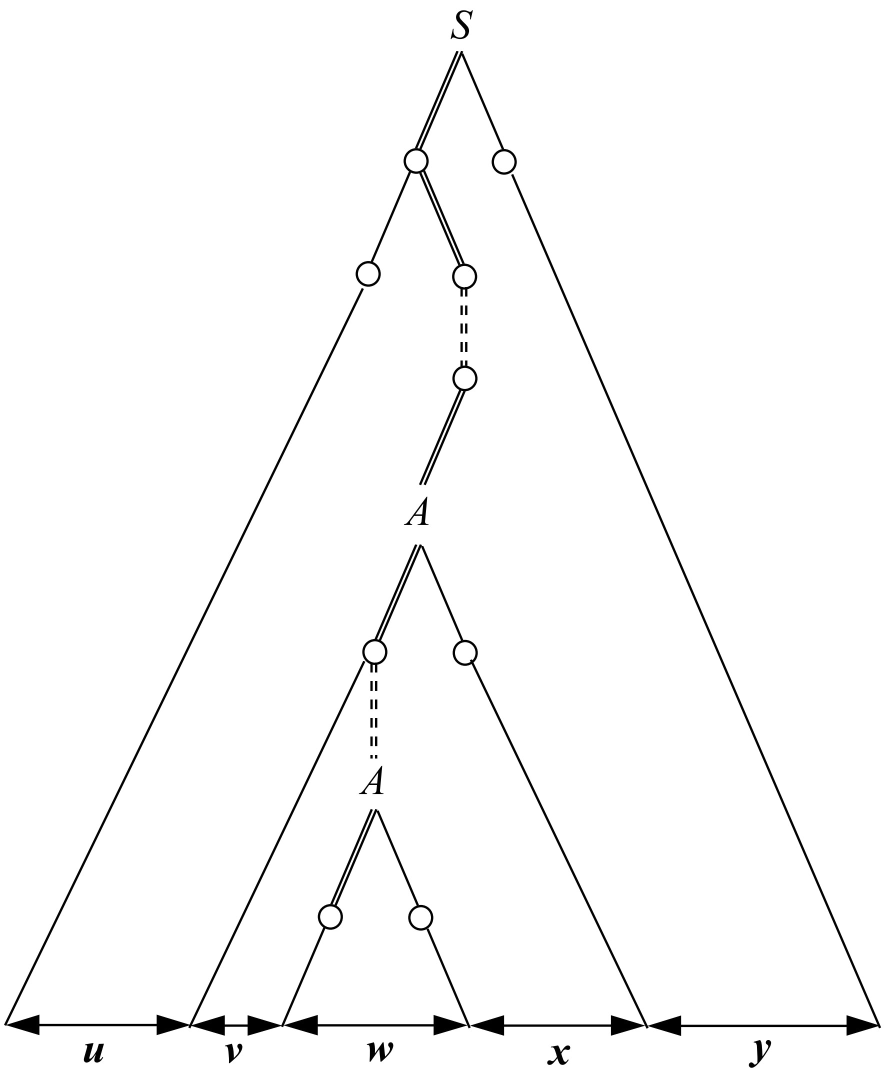
Principio di sostituzione di sottoalberi



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.

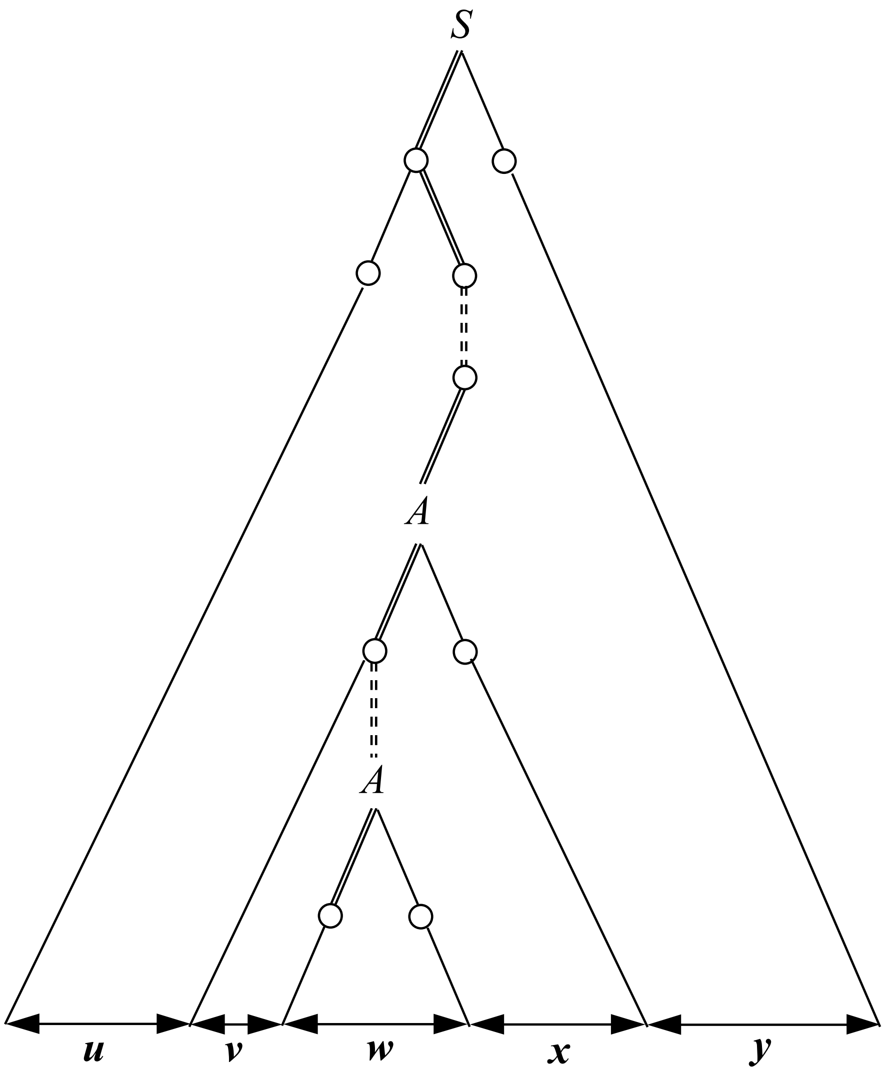
Che stringa?

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 *uvvwxxy*, ossia uv^2wx^2y. **Ripetendo questa sostituzione un numero finito di volte**, si ottiene l'insieme di stringhe:

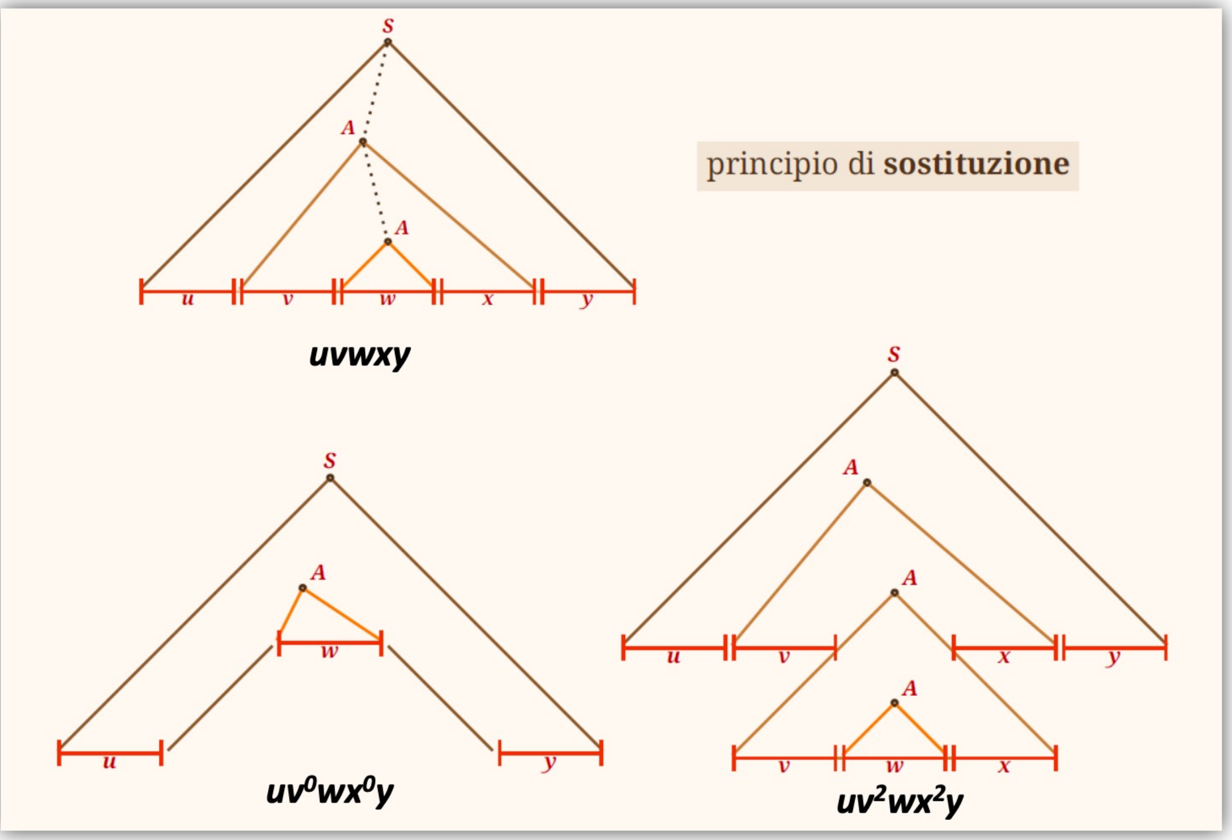
$$\{uv^n wx^n y \mid n \geq 0\}$$

Proposizione

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

$$uv^nwx^ny \qquad n\geq 0$$

.



Proposizione

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

$$uv^n wx^n y \qquad n \geq 0$$

In altri termini, ci deve essere almeno un sottoinsieme di stringhe la cui dimensione cresce in maniera costante

Un linguaggio che non ha questa caratteristica non è libero da contesto!

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:

$$m = \max\left\{ |v| \;\middle|\; A \to v \in P \right\}$$

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:

$$|w| \le m^j$$

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

Esempio

$$S \to ASA \mid a$$
$$A \to bSb \mid a$$

m = 3

Supponiamo `height`**(T_w)=2**

$|w| \leq 9$

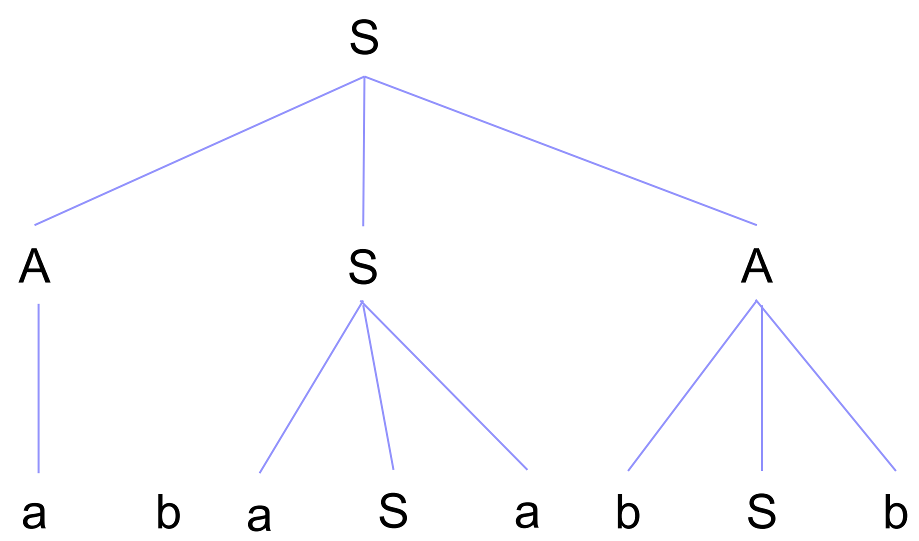
Esempio

$$S \rightarrow ASA \mid a$$
$$A \rightarrow bSb \mid a$$

m = 3

Supponiamo `height(`T_w`)=2`

$|w| \leq 9$

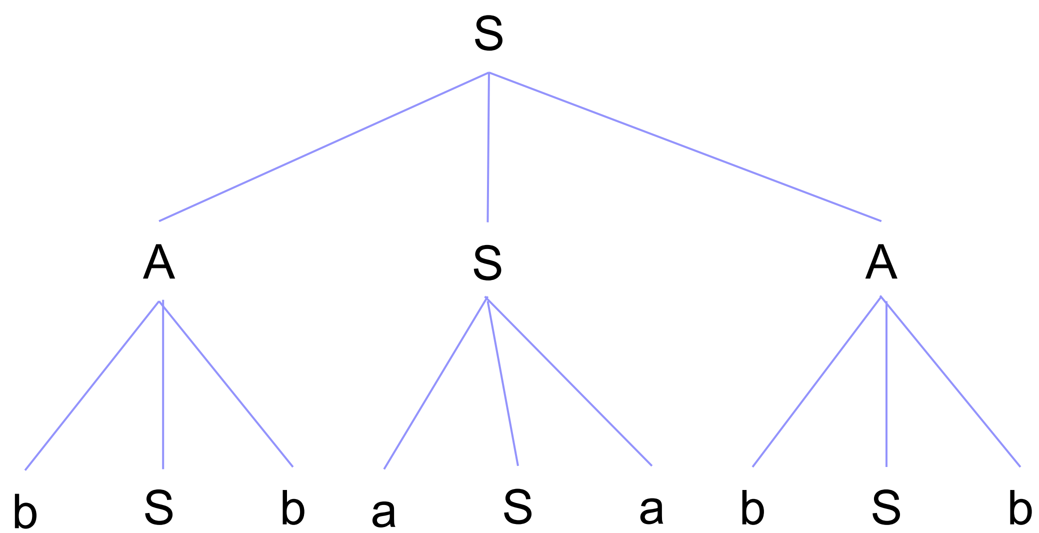


Il risultato è intuitivo, perché ad ogni passaggio posso derivare al massimo m simboli

Esempio

$S \rightarrow ASA \mid a$
$A \rightarrow bSb \mid a$

m = 3
Supponiamo `height(Tw)=2` ($height(T_w)=2$)
$|w| \leq 9$



Il risultato è intuitivo, perché ad ogni passaggio posso derivare al massimo m simboli

Dimostrazione

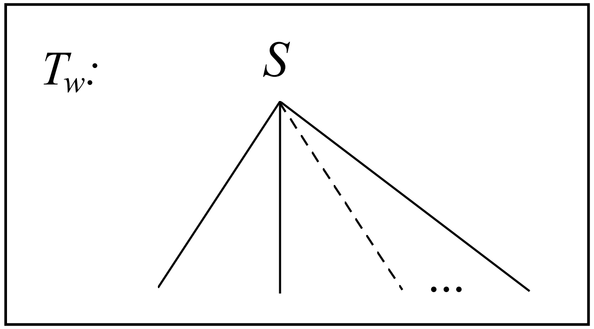
$$m = \max\left\{ |v| \;\middle|\; A \rightarrow v \in P \right\}$$

Se l'altezza è pari a j, $|w| \leq m^j$

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

  <u>Passo base</u> $j = 1$

  T_w rappresenta un'unica produzione:



Sapendo che $m = \max\left\{ |v| \;\middle|\; A \rightarrow v \in P \right\}$

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

Dimostrazione

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

Se l'altezza è pari a j, $|w| \le m^j$

- <u>Passo induttivo</u>

  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.

Dimostrazione

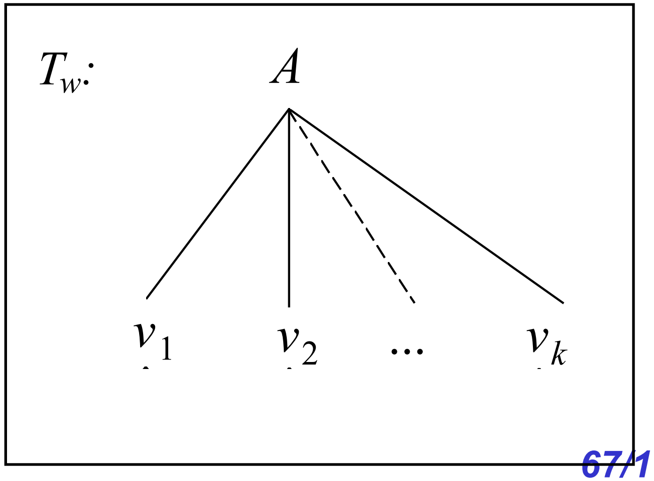
$$m = \max\left\{ |v| \;\middle|\; A \rightarrow v \in P \right\}$$

Se l’altezza è pari a j, $|w| \leq m^j$

- <u>Passo induttivo</u>

  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.

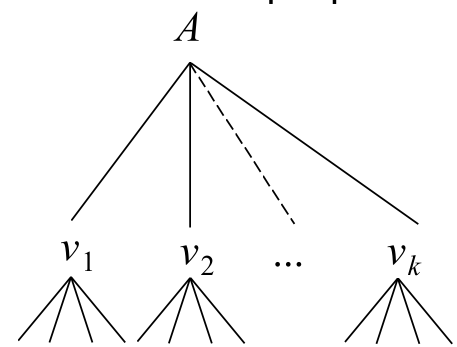
  Supponiamo che il livello più alto dell’albero rappresenti la produzione $A \rightarrow v$, dove $v = v_1 v_2 \ldots v_k$, $|v| = k,\; k \leq m$ (il sottoalbero di profondità 1 ha al più *m* figli).



Dimostrazione

$$m = \max\left\{ |v| \;\middle|\; A \to v \in P \right\}$$

Se l'altezza è pari a j, $|w| \le m^j$

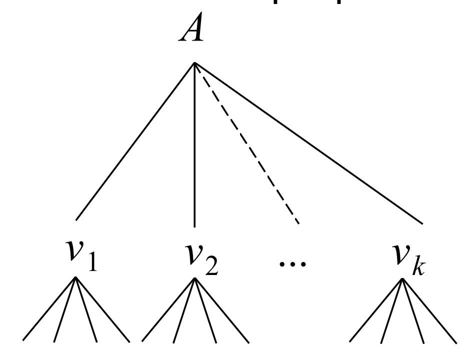


- Ogni simbolo v_i, $i = 1, 2, \ldots, k$ di v può essere radice di un sottoalbero di altezza al più uguale a j, poiché T_w ha altezza uguale a j+1 (un v_i potrebbe anche essere un terminale).

Dimostrazione

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

Se l'altezza è pari a j, $|w| \le m^j$



- Ogni simbolo v_i, i = 1, 2,..., 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 $|v| = k \le m$, la stringa w, frontiera dell'albero T_w, ha lunghezza:

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

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) $\forall\, i,\ i \geq 0 : uv^i wx^i y \in L$

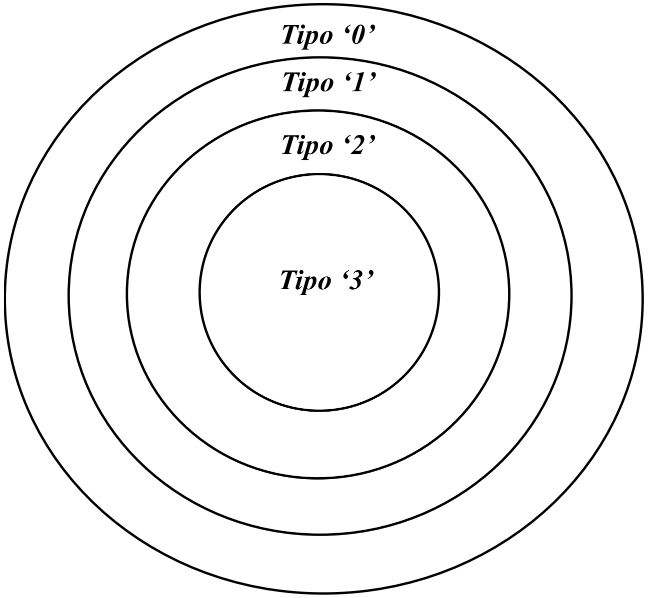
Pumping Lemma per i linguaggi liberi da contesto o teorema *uvwxy*

- 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) $\forall\, i,\ i \geq 0 : uv^i wx^i y \in L$

A cosa serve?

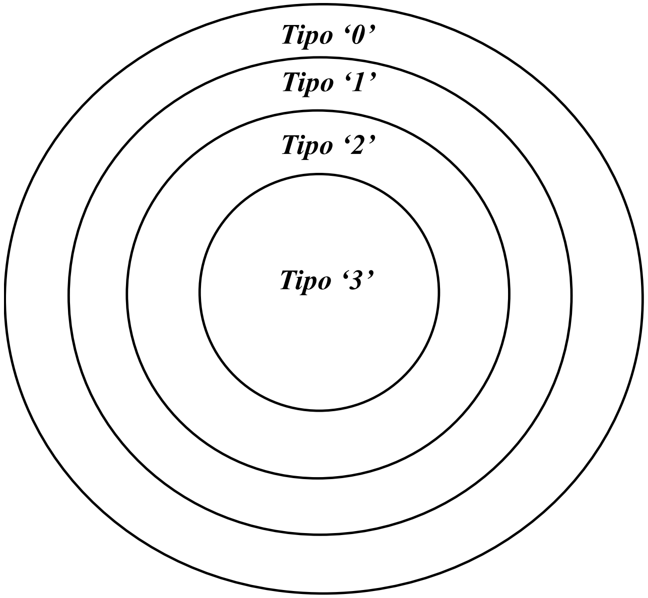
Osservazione

- Dato un linguaggio generato da una grammatica che non è C.F., (es. $a^n b^n c^n$) **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.



Osservazione

- Dato un linguaggio generato da una grammatica che non è C.F., (es. $a^n b^n c^n$) **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.
- **Un linguaggio C.F. (qualsiasi linguaggio C.F.!) deve rispondere alle regole del pumping lemma!**



Osservazione

- Dato un linguaggio generato da una grammatica che non è C.F., (es. $a^n b^n c^n$) **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.

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

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

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

Intuitivamente, riassumendo

- Se un Linguaggio L è libero, è possibile applicare il principio di sostituzione dei sotto-alberi (a patto di avere due non-terminali sullo stesso cammino)
- Il principio di sostituzione ci mostra che un sottoinsieme del linguaggio può essere «pompato» (agganciando un sottoalbero più grande a uno più piccole), ottenendo sempre parole del linguaggio
- Se questa operazione di pompaggio non si può applicare (o se applicandola ci da parole che non appartengono al linguaggio), **il linguaggio 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:

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

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

$$k = |V|$$

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:

  $$m = \max\left\{ |v| \;\middle|\; A \to v \in P \right\}$$

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

  $$k = |V|$$

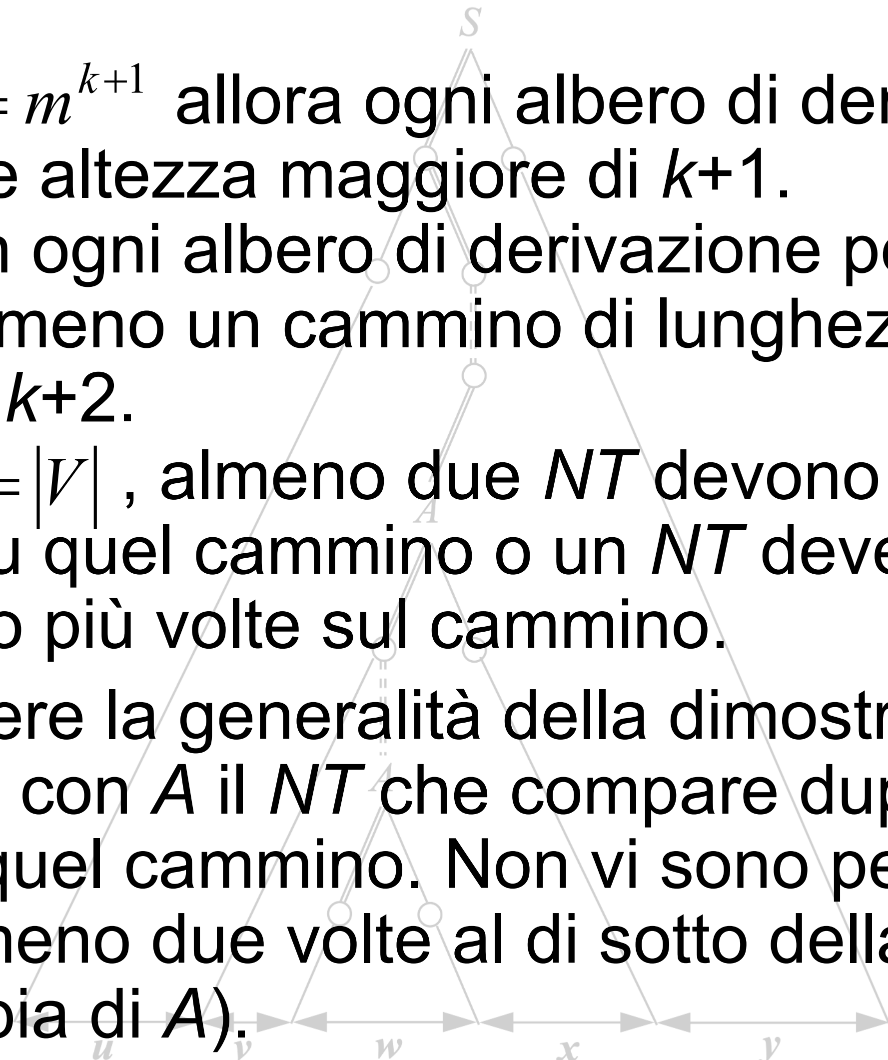
  Poniamo $p = m^{k+1}$ e sia $z \in L$, con $|z| > p$

  Per il lemma precedente:

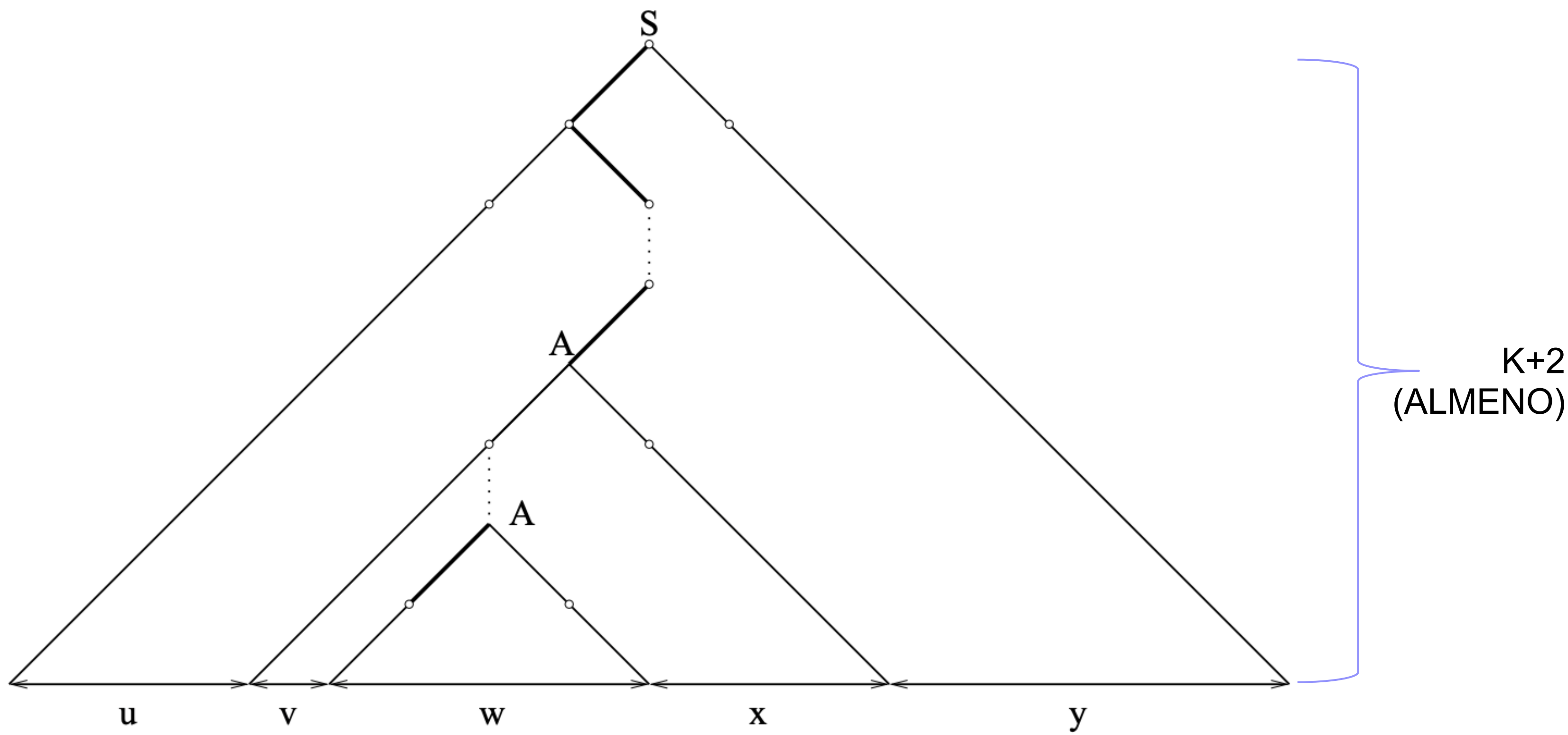
$$\left[\left(height(T_z) \le j \Rightarrow |z| \le m^j \right) \Leftrightarrow \left(|z| > m^j \Rightarrow height(T_z) > j \right) \right]$$

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*).

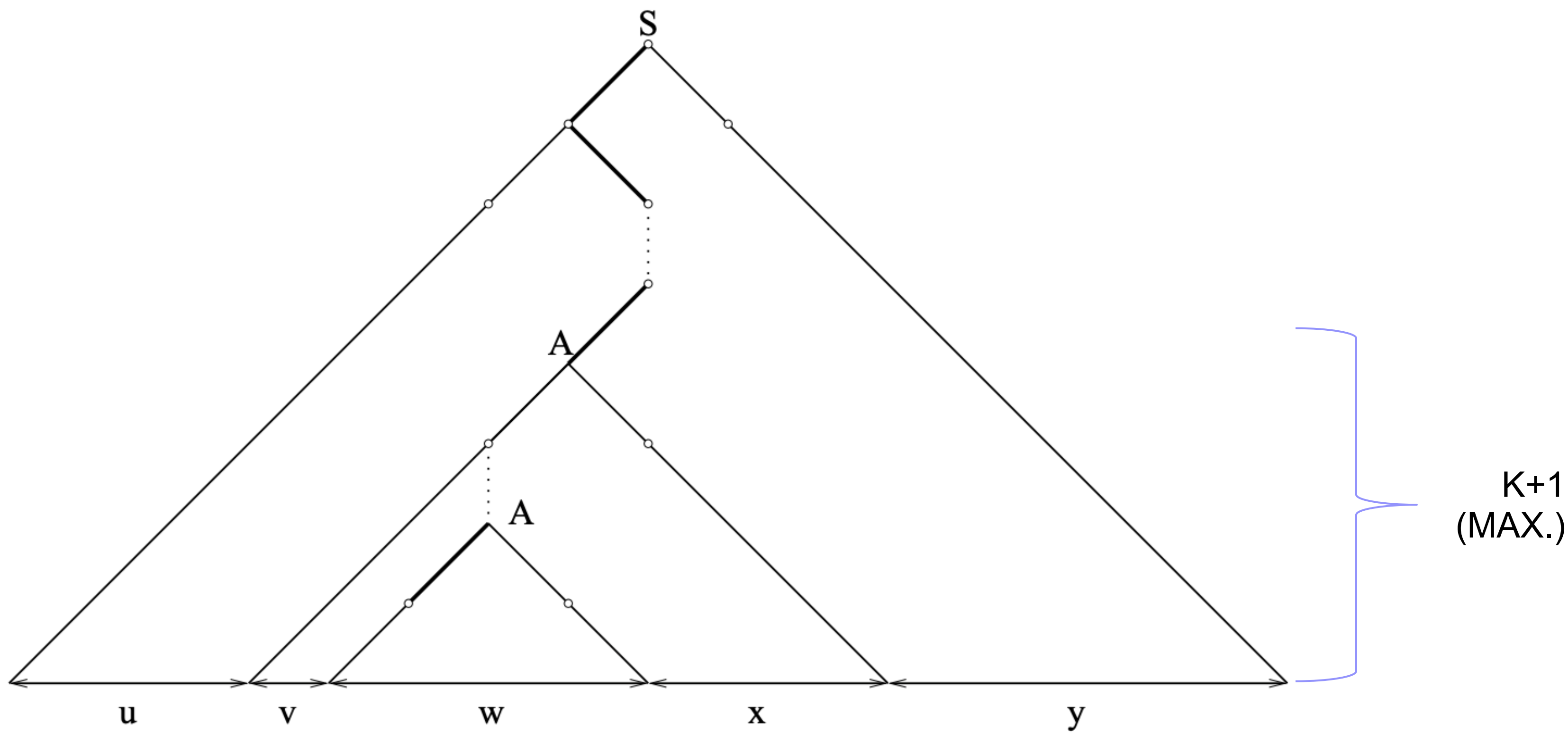


Dimostrazione Pumping Lemma



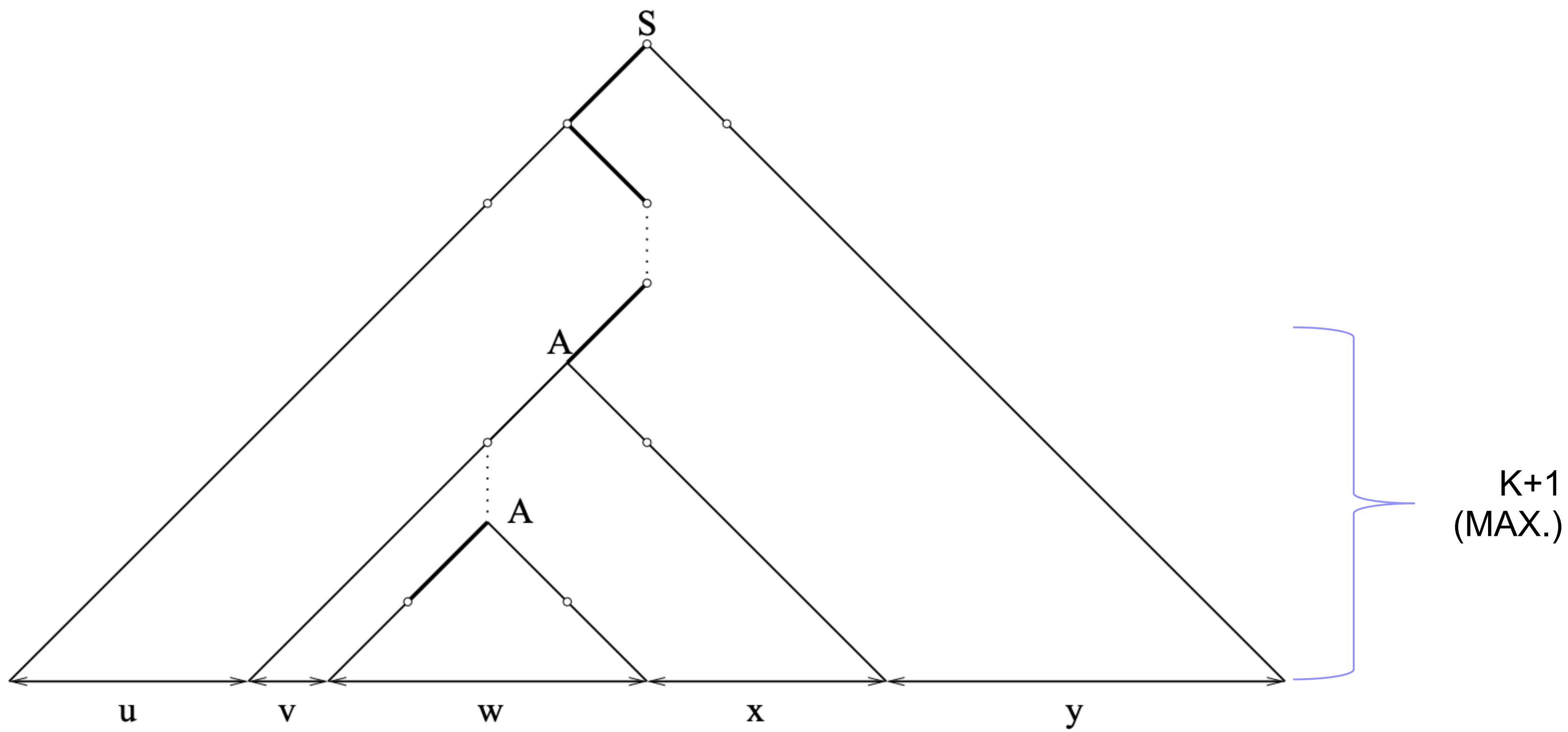
La lunghezza dell'intero albero deve essere > K+1 (almeno K+2)

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.
(supponendo che il cammino da S ad A sia pari a 1)

Dimostrazione Pumping Lemma

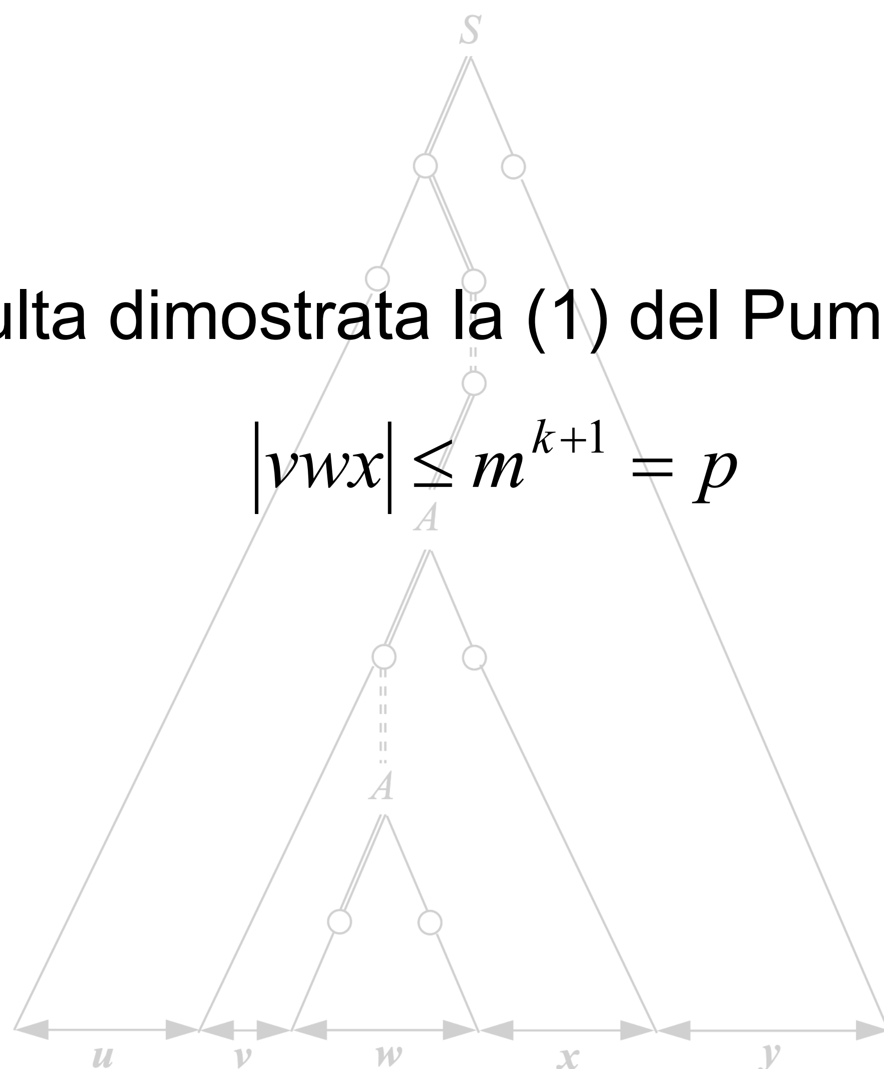


La stringa generata a partire da quel non-terminale, per il lemma precedente, avrà una lunghezza massima di m^K+1

Dimostrazione Pumping Lemma

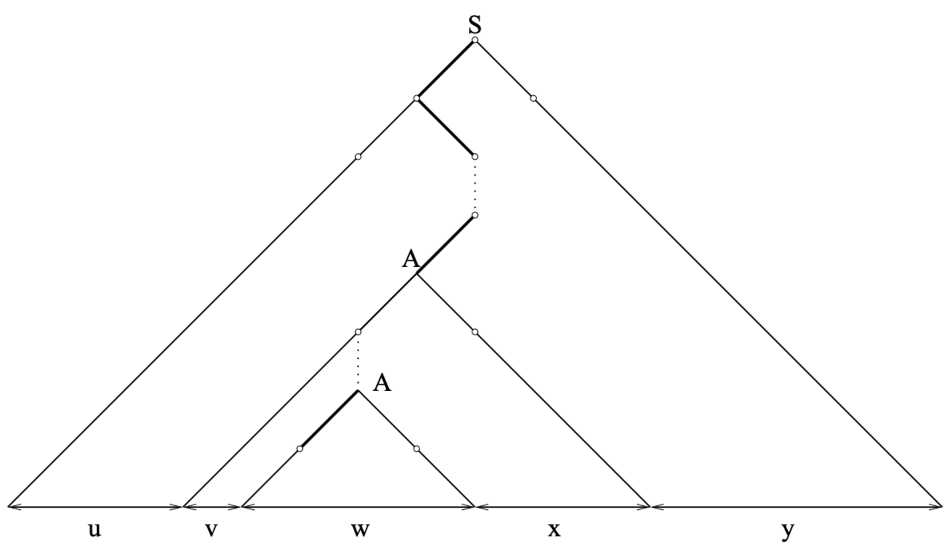
per cui risulta dimostrata la (1) del Pumping Lemma.

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



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.

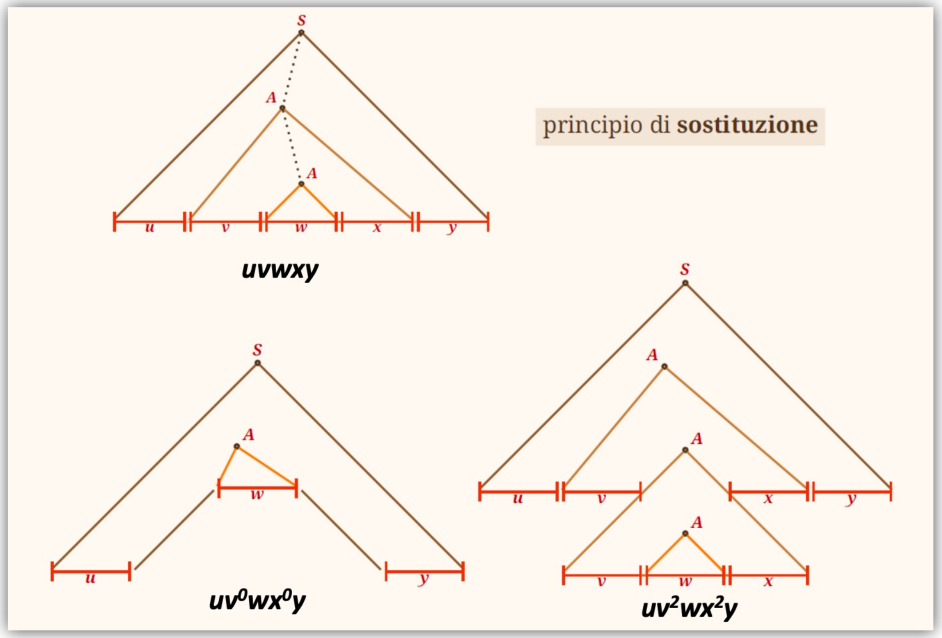


Dimostrazione Pumping Lemma

- 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: $uvvwxxy = uv^2wx^2y$ che è la (3) per i = 2.
  Se ripetiamo la suddetta sostituzione i-1 volte, la stringa derivata è: $u\underbrace{vv \ldots v}_{i \text{ volte}}w\underbrace{xx \ldots x}_{i \text{ volte}}y = uv^iwx^iy$
  per cui la (3) risulta dimostrata.

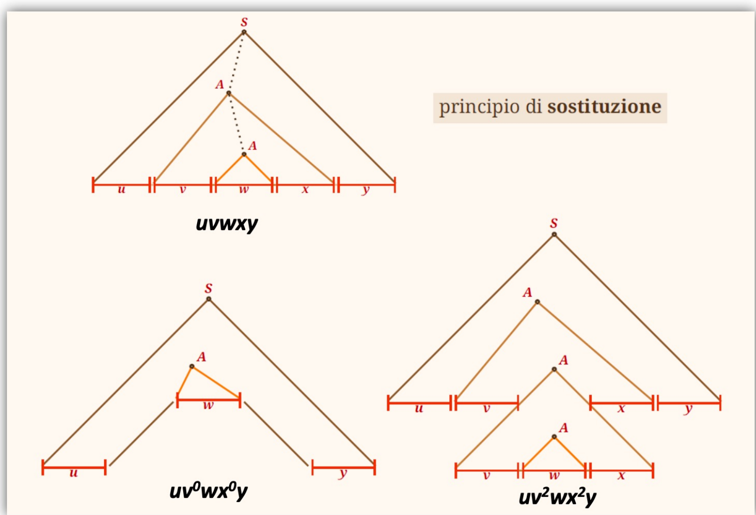
Dimostrazione Pumping Lemma

- 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.



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.



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:

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

$$X = \{a, +\}, \qquad V = \{S\}, \qquad P = \{S \to S + S,\ S \to a\}$$

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

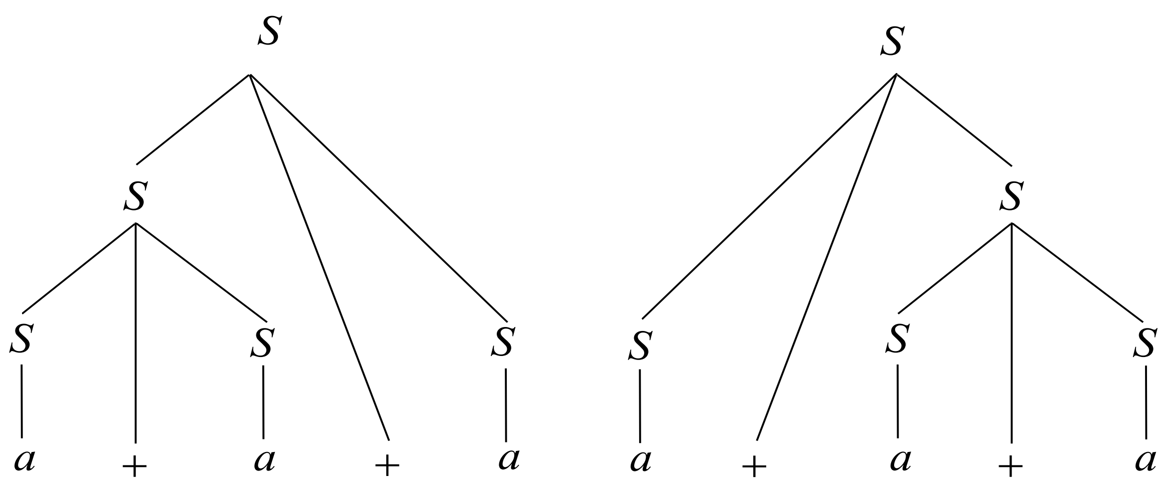
Esempio di grammatica ambigua

- La seguente grammatica libera da contesto:

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

$$X = \{a, +\}, \qquad V = \{S\}, \qquad P = \{S \to S + S,\ S \to a\}$$

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”

“dolce” — sostantivo / aggettivo

“rosso” — aggettivo / sostantivo

“un dolce rosso” — “una torta di colore rosso” / “un uomo rosso dal carattere dolce”

- 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

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

Esempio

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

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

$$X = \{a, +\}, \qquad V = \{S\}, \qquad P = \{S \to S + a,\ S \to a\}$$

  G_3 non è ambigua.
  Inoltre: $L(G_2) = L(G_3) = \left\{ aw \mid w \in \{+a\}^* \right\} = a \cdot \{+a\}^*$

- 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 $a^i b^i c^k$ e le stringhe $a^i b^j c^j$.

- 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 $G = (X, V, S, P)$:

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

$$V = \{S, C\}$$

$$P = \{S \to \underline{\text{ if }} C \ \underline{\text{then}} \ S \ \underline{\text{else}} \ S \mid$$

$$\underline{\text{if}} \ C \ \underline{\text{then}} \ S \mid$$

$$\text{a} \mid \text{b},$$

$$C \to \text{p} \mid \text{q}\}$$

Esempio di linguaggio ambiguo

- Linguaggi di programmazione
  - Si consideri la seguente grammatica $G = (X, V, S, P)$:

X = {if, then, else, a, b, p, q}

$V = \{S, C\}$

$P = \{S \rightarrow$ if C then S else S |

if C then S |

a | b,

$C \rightarrow$ p | q}

***G* è ambigua.** Infatti la stringa:

w = if p then if q then a else b

può essere generata in due modi.

Esempio di linguaggio ambiguo

- Linguaggi di programmazione
  - Si consideri la seguente grammatica $G = (X, V, S, P)$:

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

$$V = \{S, C\}$$

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

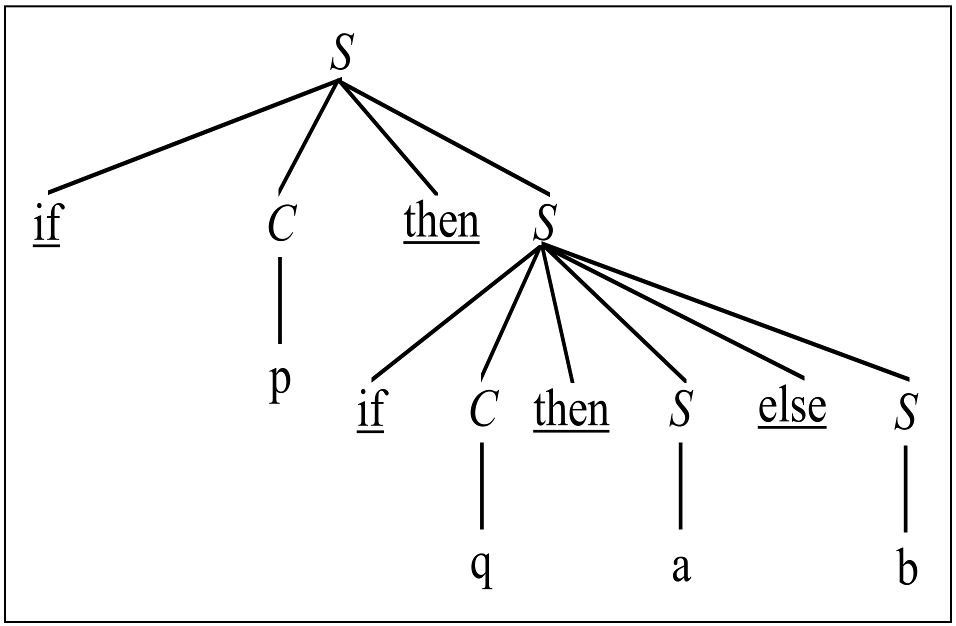
G è ambigua. Infatti la stringa:

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

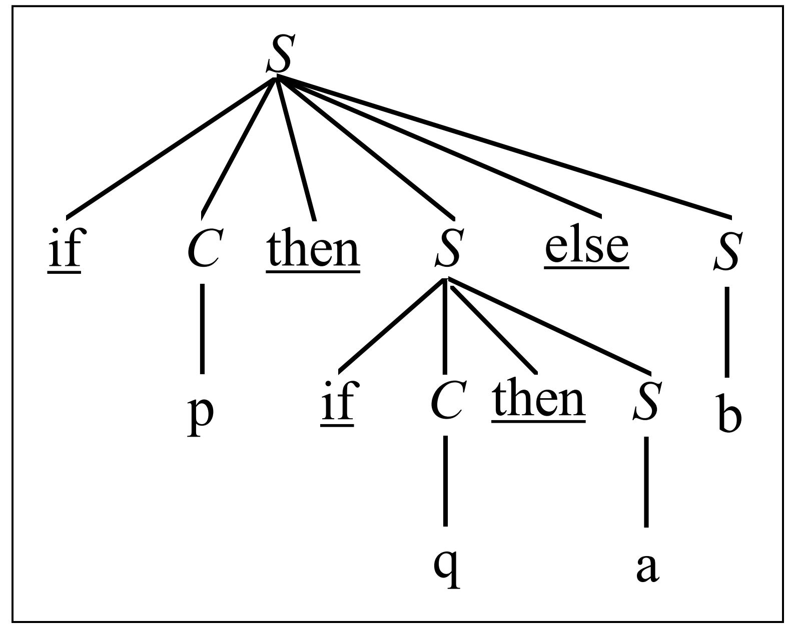
può essere generata in due modi.

A seconda dell'ordine con cui le prime due regole di produzione si applicano

Esempio di linguaggio ambiguo



<u>if</u> p <u>then</u> (<u>if</u> q <u>then</u> a <u>else</u> b)



<u>if</u> p <u>then</u> (<u>if</u> q <u>then</u> a) <u>else</u> 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:

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

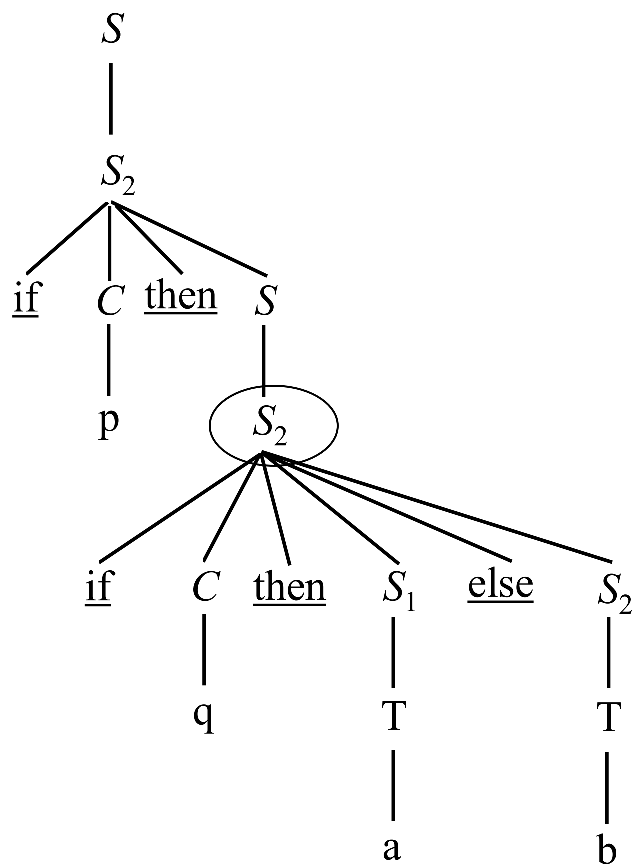
$$X = \{\underline{\text{if}}, \underline{\text{then}}, \underline{\text{else}}, \text{a}, \text{b}, \text{p}, \text{q}\}$$

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

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

Esempio di linguaggio ambiguo

- In *G'* la stringa *w* = <u>if </u>p <u>then if </u>q<u> then </u>a<u> else</u> 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.

Domande?


-It gets easier.
-Huh?
Every day, it gets a little easier.
-Yeah?
-But you gotta do it every day.
That's the hard part.
But it does get easier.