Linguaggi di Programmazione Corso C

Parte n.3

Linguaggi Liberi da Contesto e Linguaggi Contestuali

Nicola Fanizzi (*fanizzi@di.uniba.it*)

Dipartimento di Informatica
Università degli Studi di Bari

Grammatiche e Linguaggi Liberi da Contesto

- $G = (X, V, S, P)$ è una **grammatica libera da contesto** sse: $v \longrightarrow w \in P$ dove $v \in V$.
- Il linguaggio $L(G)$ si dirà allo stesso modo linguaggio libero da contesto.
- Il nome nasce dal fatto che un non terminale può essere sostituito indipendentemente dal contesto della forma di frase dove si trova.
- La sostituzione è sempre valida.
- Appartiene a questa categoria la maggior parte dei linguaggi di programmazione.

Esempio

Consideriamo le produzioni: $\left[\begin{array}{lll} 1. & AB & \longrightarrow & CDEF \\ 2. & CB & \longrightarrow & BC \end{array}\right]$

Possiamo sostituirle con le seguenti produzioni contestuali:

$$AB \longrightarrow CDEF \begin{cases} AB & \longrightarrow & AG \\ AG & \longrightarrow & CG \\ CG & \longrightarrow & CDEF \end{cases} \quad \text{e}$$

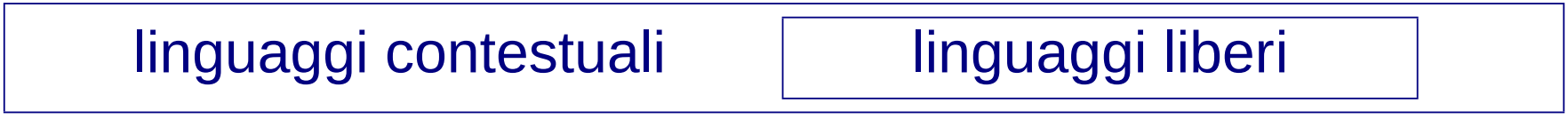
$$CB \longrightarrow BC \begin{cases} CB & \longrightarrow & XB \\ XB & \longrightarrow & XC \\ XC & \longrightarrow & BC \end{cases}$$

Linguaggi Liberi e Linguaggi Contestuali

I linguaggi dipendenti da contesto hanno produzioni del tipo:

1. $yAz \longrightarrow ywz$ con $A \in V, \quad y, z \in (X \cup V)^*$ e $w \in (X \cup V)^+$;
2. $S \longrightarrow \lambda$ purché S non compaia a destra di alcuna produzione;

- La classe dei linguaggi liberi è inclusa in quella dei linguaggi contestuali



- Si può ottenere un linguaggio libero considerando produzioni dove il contesto destro e quello sinistro siano vuoti ($= \lambda$)
- eccezione una grammatica libera ammette produzioni del tipo $A \longrightarrow \lambda$ (λ-regole), con $A \neq S$,
  il che è vietato dai linguaggi contestuali

Linguaggi Monotoni

- Una **grammatica** $G = (X, V, S, P)$ è **monotona** sse ogni produzione è monotona: $\forall v \longrightarrow w \in P\colon\ |v| \leq |w|$.
- Un **linguaggio** è **monotono** sse esiste una grammatica monotona che lo genera.

Teoremi sui Linguaggi Monotoni

Teorema.

Sia G una grammatica monotona, tranne al piú per una produzione $S \longrightarrow \lambda$ se S non appare in una parte destra di produzioni di G. Allora esiste una grammatica contestuale G' che risulta equivalente a G

In maniera equivalente si dimostra:

Teorema.

Un linguaggio L è contestuale sse esiste una grammatica G tale che $L = L(G)$ a produzioni monotone tranne al piú il caso che $\lambda \in L$ per cui puó essere generato direttamente da S se S non appare in alcuna parte destra di produzioni di G

Dim. Supponiamo che le produzioni possano essere ricondotte a:

$$A_1 A_2 \ldots Am \longrightarrow B_1 B_2 \ldots Bn$$

con $m \leq n$ e $A_i \in V, i = 1, \ldots, n$

Usiamo m nuovi simboli non terminali $C_k \notin G,\ k = 1, \ldots, m$

Trasformiamo ogni produzione nella seguente maniera:

$$\begin{aligned}
A_1 A_2 \ldots A_m &\longrightarrow C_1 A_2 \ldots A_m \\
C_1 A_2 \ldots A_m &\longrightarrow C_1 C_2 A_3 \ldots A_m \\
&\ldots \\
C_1 C_2 \ldots C_{m-1} A_m &\longrightarrow C_1 C_2 \ldots C_m B_{m+1} B_{m+2} \ldots B_n \\
C_1 C_2 \ldots C_m B_{m+1} B_{m+2} \ldots B_n &\longrightarrow C_1 C_2 \ldots C_{m-1} B_m B_{m+1} \ldots B_n \\
&\ldots \\
C_1 B_2 \ldots B_n &\longrightarrow B_1 B_2 \ldots B_n
\end{aligned}$$

G' è contestuale e si dimostra: $L(G) = L(G')$

Esempio

Consideriamo il linguaggio:$\{a^n b^n C^n | n > 0\}$

Una possibile grammatica contiene le seguenti produzioni:

$$\begin{cases} S & \longrightarrow & aSBC|aBC \\ CB & \longrightarrow & BC \\ aB & \longrightarrow & ab \\ bB & \longrightarrow & bb \\ bC & \longrightarrow & bc \\ cC & \longrightarrow & cc \end{cases}$$

Monotona ma non contestuale.

Trasformandola in grammatica contestuale:

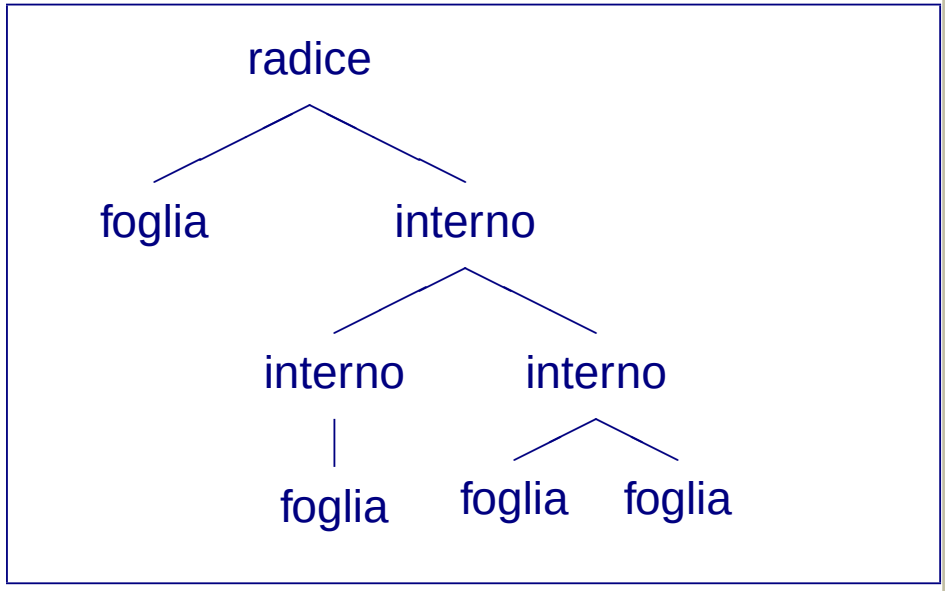
$$\begin{cases} S & \longrightarrow & aSBC|aBC \\ CB & \longrightarrow & XB \\ XB & \longrightarrow & XC \\ XC & \longrightarrow & BC \\ aB & \longrightarrow & ab \\ bB & \longrightarrow & bb \\ bC & \longrightarrow & bc \\ cC & \longrightarrow & cc \end{cases}$$

Alberi

Albero Grafo orientato, aciclico, connesso con al più un arco entrante per ogni nodo

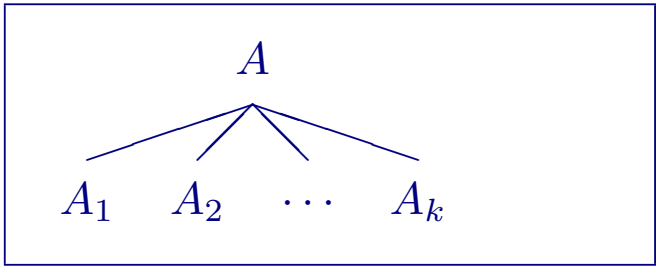
Le derivazioni di una grammatica libera possono essere rappresentate graficamente da alberi

La frontiera dell'albero è rappresentata dalle foglie lette da sinistra verso destra
La lunghezza di un cammino dalla radice ad una foglia è data dal numero di non terminali incontrati
L'altezza dell'albero è data dalla lunghezza del cammino più lungo.



Alberi di Derivazione

Data una grammatica libera $G = (X, V, S, P)$ e $w \in X^*$ tale che $S \stackrel{*}{\Longrightarrow} w$ un albero di derivazione di w ha le seguenti proprietà:

1. radice = S
2. nodi interni = V
3. foglie = $X \cup \{\lambda\}$
4. 

se il nodo interno è A e $A_1, A_2, \ldots, A_k$ sono i figli del nodo A
allora $\exists A \longrightarrow A_1 A_2 \cdots A_k \in P$
5. w si ottiene leggendo e concatenando le foglie da sinistra a destra

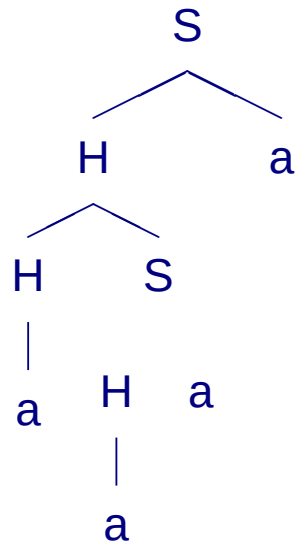
Osservazioni

Un albero di derivazione non impone alcun ordine nell'applicazione delle produzioni in sequenza per ottenere una derivazione

- data una derivazione
  esiste uno ed un solo albero che la rappresenta
- dato un albero di derivazione
  esistono più derivazioni possibili a seconda dell'ordine scelto per l'applicazione delle produzioni

Esempio Data una grammatica libera $G = (X, V, S, P)$ con $X = \{a\}$, $V = \{S, H\}$ e $P = \{S \xrightarrow{1} Ha, \quad H \xrightarrow{2} HS, \quad H \xrightarrow{3} a\}$

La stringa $aaaa$ è in $L(G)$, come dimostra l'albero:



Da cui la stringa si ricava sia tramite:

$$S \Longrightarrow_1 Ha \Longrightarrow_2 HSa \Longrightarrow_3 aSa \Longrightarrow_1 aHaa \Longrightarrow_3 aaaa$$

sia con:

$$S \Longrightarrow_1 Ha \Longrightarrow_2 HSa \Longrightarrow_1 HHaa \Longrightarrow_3 Haaa \Longrightarrow_3 aaaa$$

Derivazioni

Data una grammatica $G = (X, V, S, P)$ diremo che una derivazione

$$S \Longrightarrow w_1 \Longrightarrow w_2 \Longrightarrow \cdots \Longrightarrow w_n = w$$

con $w_i = y_i A z_i$ e $w_{i+1} = y_i w_i z_i, \qquad i = 1, \ldots, n-1$
è una **derivazione destra** (risp. **sinistra**) sse per ogni $i = 1, \ldots, n-1$ risulta:

$$z_i \in X^* \qquad \text{(risp. } y_i \in X^*\text{)}$$

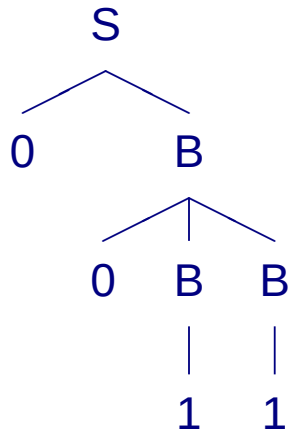
Esempio Considero la grammatica con produzioni: $P = \left\{ \begin{array}{lcl} S & \longrightarrow & 0B|1A \\ A & \longrightarrow & 0|0S|1AA \\ B & \longrightarrow & 1|1S|0BB \end{array} \right\}$

Derivazione sinistra di 0011: $S \Longrightarrow 0B \Longrightarrow 00BB \Longrightarrow 001B \Longrightarrow 0011$

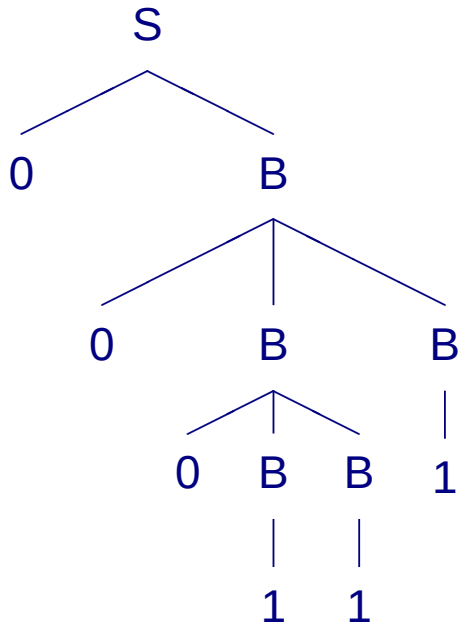
Derivazione destra di 0011: $S \Longrightarrow 0B \Longrightarrow 00BB \Longrightarrow 00B1 \Longrightarrow 0011$

Principio di sostituzione

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



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

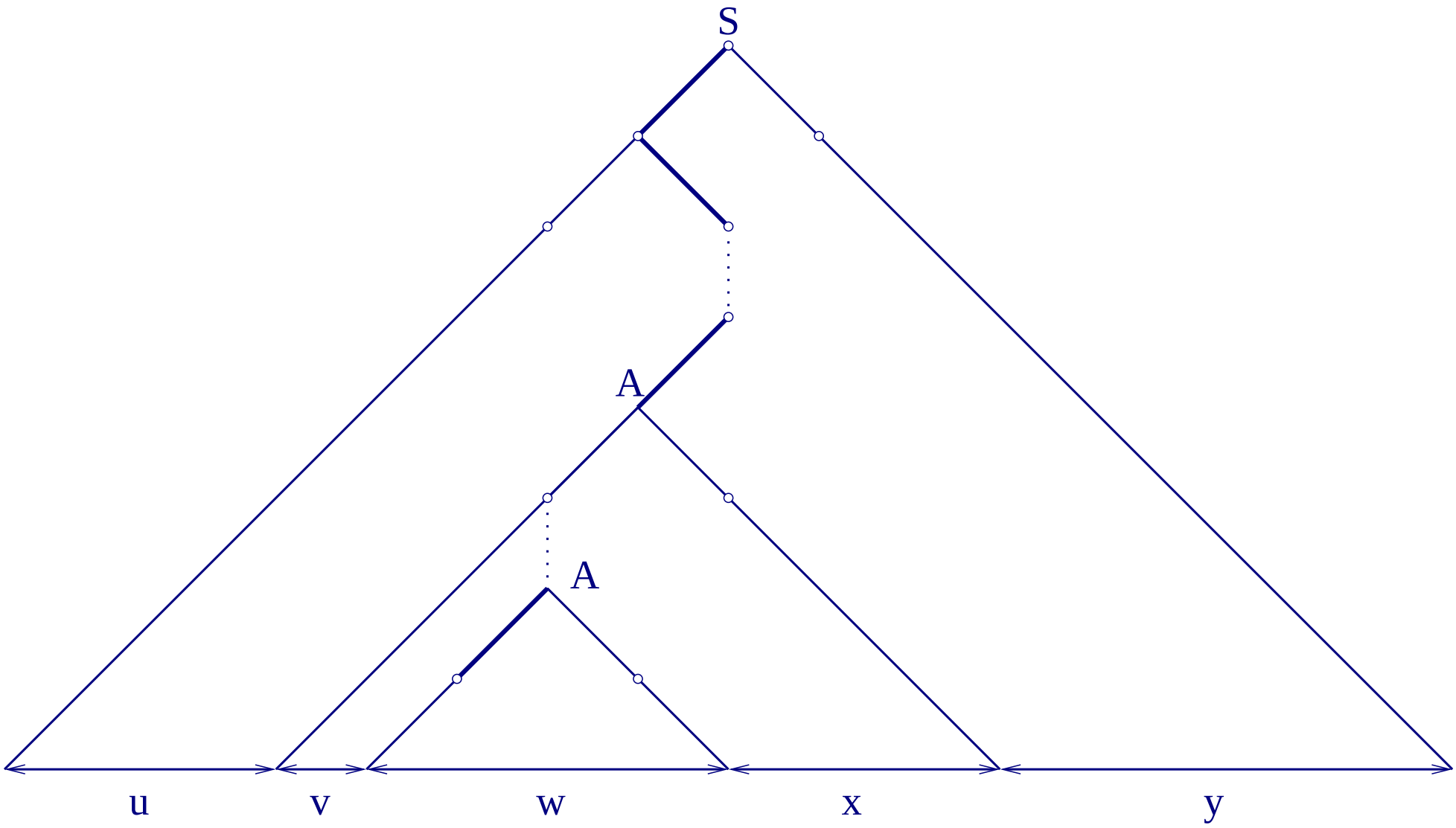


Iterando:

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

Principio di sostituzione (II)

- Nelle grammatiche libere alberi piccoli si possono sostituire con alberi maggiori di pari radice (non terminale)
- La lunghezza delle parole così ottenute cresce in maniera costante
- Una grammatica che genera parole che crescono via via in modo esponenziale non è libera da contesto
- Generalizzazione: supposto di incontrare 2 volte il non terminale A nell'albero di derivazione di z.
  - il sottoalbero più basso con radice A genera w
  - quello più alto genera vwx
  - per il principio di sostituzione, sostituendo quello più alto con quello più basso si ottiene una derivazione valida della stringa uwy
  - invece, sostituendo quello più basso con quello più alto si ottiene una derivazione della stringa $uvvwxxy$ cioè uv^2wx^2y
  - iterando questa sostituzione ottengo l'insieme di stringhe $\{uv^nwx^ny | n \geq 0\}$


S
A
A
u
v
w
x
y

Principio di sostituzione (III)

Proposizione.

Ogni linguaggio libero infinito deve contenere almeno un sottinsieme infinito di stringhe della forma

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

Lemma.

Data una grammatica $G = (X, V, S, P)$ libera, supponiamo che

$$m = \max\{|w| \in \mathbb{N} \mid A \longrightarrow w \in P\}$$

Sia T_w un albero di derivazione per una stringa w di $L(G)$.

Se l'altezza di T_w è al più pari a $j \in \mathbb{N}$, allora $|w| \leq m^j$

Dim.

Per induzione su j:

base $j = 1 : |w| \leq m = m^1$

passo supponiamo che il lemma valga per un'altezza pari al più a j e la cui radice sia un non terminale e dimostriamo la tesi per $j + 1$:

Supponiamo che il livello più alto dell'albero sia determinato da $A \longrightarrow v$, dove $v = v_1 v_2 \cdots v_k, \quad |v| = k, \quad k \leq m$

Ogni simbolo $v_i \in v,\ i = 1, \ldots, k$ può avere altezza al più uguale a j, essendo T_w in questo caso di altezza $j + 1$

Per ipotesi di induzione, ciascuno di questi alberi ha al più m^j foglie.

Poiché $|v| = k \leq m$ la stringa w frontiera di T_w avrà lunghezza:

$$|w| \leq \overbrace{m^j + m^j + \cdots + m^j}^{k \text{ volte}} = |v| \cdot m^j = k \cdot m^j \leq m \cdot m^j = m^{j+1}$$

Pumping Lemma per linguaggi Liberi

Teorema uvwxy.

Sia L un linguaggio libero da contesto. Allora esiste una costante p dipendente 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 che:

1. $|vwx| \leq p$
2. al più uno tra v e x è la parola vuota ($vx \neq \lambda$)
3. $\forall i \in \mathbb{N}, i \geq 0 : \ uv^i wx^i y \in L$

Dim. Pumping Lemma Sia G una grammatica che genera L
$m = \max\{|v||A \longrightarrow v \in P\}$ e $k = |V|$

Poniamo $p = m^{k+1}$ e consideriamo $z \in L$ tale che $|z| > p$

Per il lemma: $|z| > p = m^{k+1}$ allora ogni albero di derivazione per z ha un'altezza maggiore di $k+1$, cioè esiste un cammino di lunghezza maggiore o uguale a $k+2$.

Ma $k = |V|$ quindi sul cammino ci sono almeno due NT ripetuti o un NT che compare 3 volte. Chiamiamo A questo NT.

Siccome A è l'NT ripetuto più in alto non vi sono altri NT ripetuti almeno due volte sotto la A più in alto, quindi il cammino dalla A superiore ad una foglia ha lunghezza al più $k+1$.

Chiamiamo vwx la stringa derivata dal sottoalbero radicato nella A superiore, dove w è la sottostringa derivata dall'A inferiore

1. Dal Lemma risulta: $|vwx| \leq m^{k+1} = p$

2. Per assurdo se fosse $v = \lambda = x$ la sostituzione dell'albero radicato nell'A superiore con quello inferiore non provoca nessun cambiamento.
   Ma in tal caso esiste un cammino di lunghezza inferiore. Si ottiene un albero di derivazione di z di altezza al più pari a $k + 1$. Assurdo.

3. Applicando il principio di sostituzione a $z = uvwxy$
   sostituiamo il sottoalbero radicato nell'A inferiore con quello dell'A superiore ottenendo: $uwy = uv^0wx^0y$
   Con la sostituzione inversa: uv^2wx^2y e ripetendo $i - 1$ volte: uv^iwx^iy

Esercizi

Dimostrare che i seguenti linguaggi non sono liberi da contesto:

- $L = \{a^t | t \text{ primo}\}$
- $L = \{a^n b^n c^n | n > 0\}$
- $L = \{a^{n^2} | n \geq 0\}$
- $L = \{a^i b^j | i = 2^j,\ i, j \geq 0\}$
- $L = \{a^k b^r | k > 0,\ r > k^2\}$

Esercizio Dimostrare che il linguaggio $\{a^n b^n c^n | n > 0\}$ non è libero.

Supponiamo che L sia libero. Vale il Pumping Lemma per un certo $p \in \mathbb{N}$.

Considero allora $z = uvwxy = a^p b^p c^p \in L$ tale che $|z| = 3p > p$ ma $|vwx| \leq p$

Per vwx si hanno le seguenti possibilità.

In tutti questi casi si dimostra che $uv^2wx^2y \notin L$ quindi L non può essere libero.

1. $vwx = a^k, \quad 0 < k \leq p$ aggiungendo almeno a ed al più a^p si ottiene:

$$uv^2wx^2y = a^{p+k}b^p c^p$$

2. $vwx = b^k, \quad 0 < k \leq p$ analogamente

3. $vwx = c^k, \quad 0 < k \leq p$ analogamente

4. $vwx = a^k b^r, \quad 0 < k + r \leq p$ per la 2. del Pumping Lemma:

   a) $v \neq \lambda,\ x \neq \lambda$:
   se $v \neq \lambda$ allora $v = a^{k'}$ perchè se fosse $v = a^k b^{r'}$ allora

   $$uv^2wx^2y = a^{p-k}a^k b^{r'} a^k b^{r'} b^s c^p \notin L$$

   con $p \leq s \leq 2(r - r') + p - r$
   Analogamente $x \neq \lambda$ implica che $x = b^{r'}$ per cui:

   $$uv^2wx^2y = a^{p+k'} b^{p+r'} c^p \notin L$$

   per $k', r' > 0$

   b) $v \neq \lambda,\ x = \lambda$: per le considerazioni fatte: $v = a^{k'},\ 0 < k' \leq k$ e

   $$uv^2wx^2y = a^{p+k'} b^p c^p \notin L$$

   c) $v = \lambda,\ x \neq \lambda$: analogamente

5. $vwx = b^k c^r, \quad 0 < k + r \leq p$ analogamente al caso precedente

Esercizio Dimostrare che il linguaggio $L = \{a^{n^2} | n \geq 0\}$ non è libero.

Consideriamo

$$L = \{\lambda, a, aaaa, a^9, a^{16}, \ldots\}$$

e supponiamo che sia libero.

Vale il Pumping Lemma per un certo $p \in \mathbb{N}$.

Considero allora $z = uvwxy = a^{p^2} \in L$ tale che $|z| = p^2 > p$

Anche $uv^2wx^2y \in L$ (per la 3. del Lemma)

Ma: $|uv^2wx^2y| = |uvwxy| + |vx| = |z| + |vx| \leq p^2 + p < p^2 + 2p + 1 = (p+1)^2$

$|uv^2wx^2y| < (p+1)^2$ implica che $uv^2wx^2y \notin L$

Assurdo, quindi L non è libero

Osservazioni

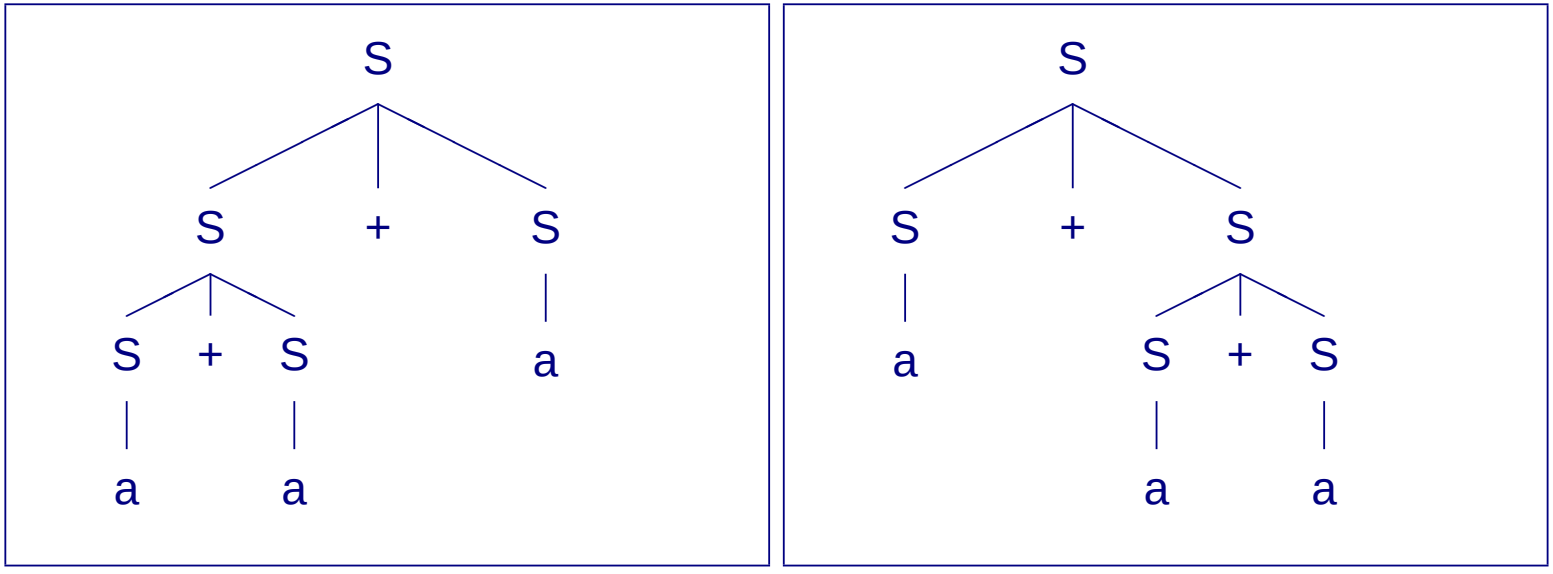
- Dato un linguaggio generato da una grammatica non libera non si può escludere che esista una grammatica libera che lo generi
- se un linguaggio infinito non rispetta il Pumping Lemma dei linguaggi liberi non potrà essere generato da una grammatica libera
- quindi questo teorema fornisce una condizione *necessaria* (ma non sufficiente) perché un linguaggio sia libero
- Si può utilizzare per dimostrare (per assurdo) che il linguaggio generato da una data grammatica non sia libero

Ambiguità dei Linguaggi

Una grammatica G libera da contesto si dice **ambigua** sse esiste una stringa x in $L(G)$ che ha due alberi di derivazione differenti ovvero, sse x ha due derivazioni sinistre (o destre)

Esempio. La grammatica libera $G = (X, V, S, P)$ con
$X = \{a, +\}, \qquad V = \{S\}$ e $P = \{S \longrightarrow S + S, \quad S \longrightarrow a\}$
è una grammatica ambigua.

$w = a + a + a$ ottenibile con



Linguaggi Inerentemente Ambigui

Un linguaggio G si dice **inerentemente ambiguo** sse ogni grammatica che lo genera è ambigua

Esempio.

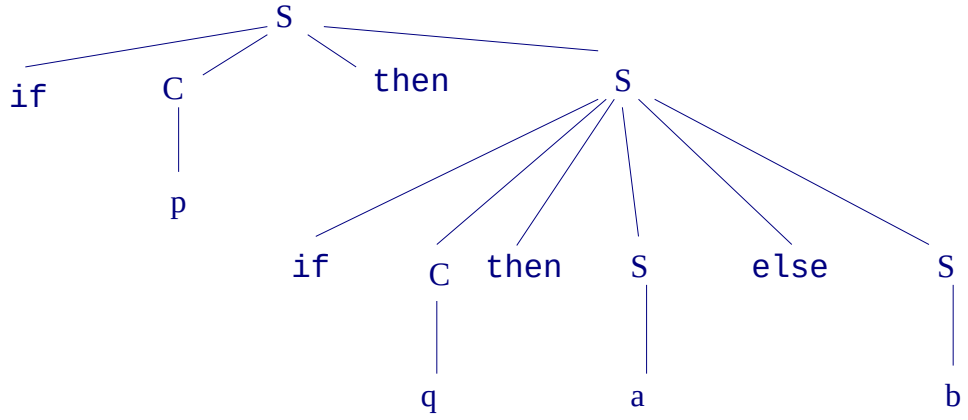
$$L = \{ab^j c^k | i, j, k > 0,\ (i = j) \vee (j = k)\}$$

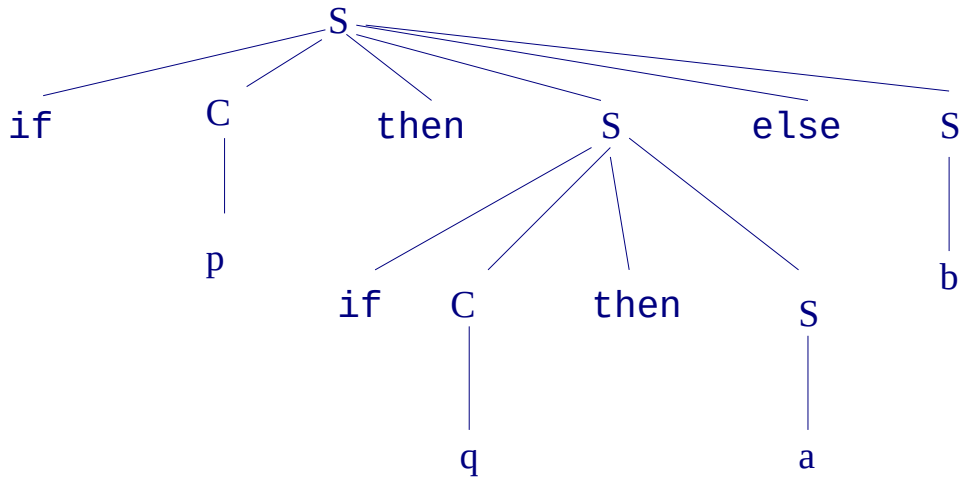
Esempio. $G = (X, V, S, P)$

con $X = \{\texttt{if}, \texttt{then}, \texttt{else}, \texttt{a}, \texttt{b}, \texttt{p}, \texttt{q}\}$, $V = \{S, C\}$

$P = \{S \longrightarrow \texttt{if}\ C\ \texttt{then}\ S\ \texttt{else}\ S | \texttt{if}\ C\ \texttt{then}\ S | a | b, \quad C \longrightarrow p | q\}$

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

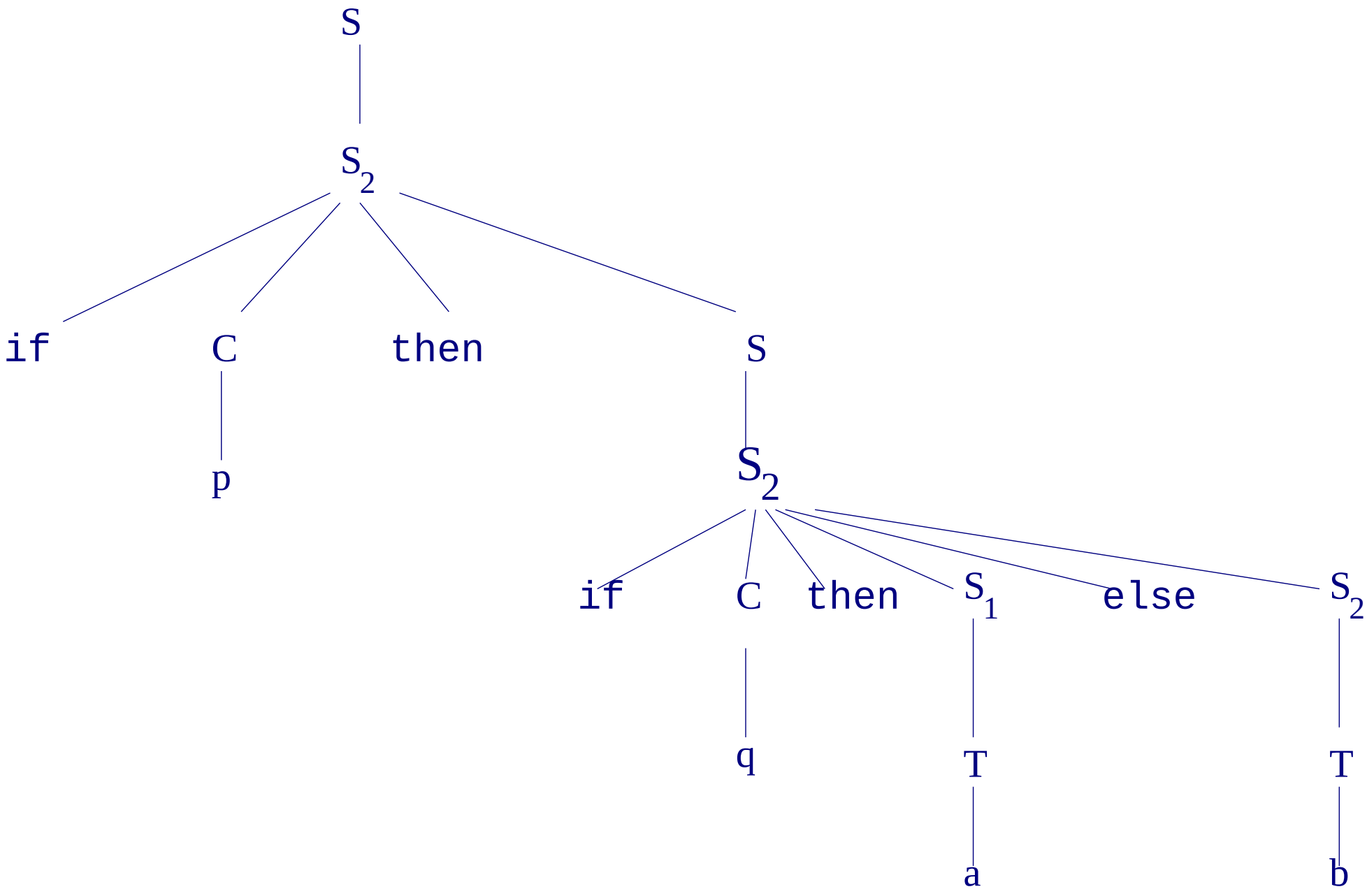




Per ottenere $G' = (X, V', S, P')$ non ambigua usiamo la convenzione di associare ogni `else` alla `if` più vicina:

$$V' = V \cup \{S_1, S_2, T\}$$

$$P' = \left\{ \begin{array}{lcl} S & \longrightarrow & S_1 \mid S_2 \\ S_1 & \longrightarrow & \texttt{if}\ C\ \texttt{then}\ S_1\ \texttt{else}\ S_2 \mid T \\ S_2 & \longrightarrow & \texttt{if}\ C\ \texttt{then}\ S \mid \texttt{if}\ C\ \texttt{then}\ S_1\ \texttt{else}\ S_2 \mid T \\ C & \longrightarrow & p \mid q \\ T & \longrightarrow & a \mid b \end{array} \right\}$$


S
S2
if
C
then
S
p
S2
if
C
then
S1
else
S2
q
T
T
a
b