pumping lemma per CFL

La forma normale di Chomsky (CNF) per le CFG prevede che tutte le produzioni siano:

A -> a, con a in T

A-> BC, con B e C variabili

non c'è ε !!

Teorema. Per ogni CFG G, esiste una CFG G' in CNF tale che L(G')=L(G)-{ε} ripuliamo la CFG

--dai simboli inutili

--dalle produzioni unitarie, i.e. A->B

--dalla ε

un simbolo X è <u>utile</u> per una CFG se
$S \Rightarrow^* \alpha X \beta \Rightarrow^* w$

un simbolo è generatore se
$X \Rightarrow^* w$

è raggiungibile se
$S \Rightarrow^* \alpha X \beta$

prima si eliminano i non generatori e poi i non raggiungibili
restano solo i simboli utili

esempio:

S-> AB | a

A -> b

non generatori : B

resta S -> a e A -> b

ma A è diventato non raggiungibile, quindi resta

S -> a

Attenzione all’ordine

Teorema 7.2 data la CFG G con L(G) non vuoto, costruiamo G1 in 2 passi:

1) Otteniamo G' da G eliminando da G le variabili non generatrici e le produzioni in cui queste variabili compaiono
2) Otteniamo G1 da G' eliminando da G' i simboli non raggiungibili

G1 contiene tutti e soli i simboli utili di G e quindi L(G1)=L(G)

Dimostrazione: in 2 passi

Passo 1)
ogni derivazione X =>* w in G contiene solo simboli generatori e produzioi con solo tali simboli,
quindi eliminando i simboli non generatori e le produzioni che ne contengono qualcuno, non tocchiamo queste derivazioni che saranno anche in G’
in particolare L(G’)=L(G)
G’ ha solo simboli generatori

2) Ogni derivazione S => * w di G’ non contiene simboli non raggiungibili.
Quindi eliminando i simboli non raggiungibili e le produzioni con tali simboli a sinistra, non tocchiamo queste derivazioni che saranno anche in G1, in particolare L(G1)=L(G’)=L(G).
G1 ha solo simboli raggiungibili e generatori

Osservazione: se X è raggiungibile in G1, X è ancora generatore in G1 perché qualsiasi X =>* w in G’ usa solo simboli raggiungibili

calcolo dei simboli generatori ∏ :
-base: i simboli terminali sono generatori, ∏ =T
-induzione: aggiungiamo a ∏ tutte le variabili X
t.c. esiste X -> α in cui α ha solo simboli in ∏

fino a quando ∏ cresce

Calcolo dei simboli raggiungibili

Base: R contiene S

Induzione: Aggiungere ad R i simboli in β tale che $A \rightarrow \beta$ per A in R

In modo simile:
calcolo dell'insieme Z dei simboli che producono ε

base: $X \to \varepsilon$ mettiamo X in Z
induzione: aggiungiamo a Z ogni Y tale che $Y \to \alpha$ con i simboli di α tutti in Z

Per una qualsiasi CFG G possiamo costruire G' t.c. $L(G') = L(G) - \{\varepsilon\}$ e senza ε-produzioni

Idea della costruzione di G’:

Prendiamo una qualsiasi produzione p di G, A->X1…Xk, se m dei k Xi sono in Z, allora G’ ha 2^m produzioni ottenute da p annullando tutti i sottoinsiemi delle m Xi

non si considera m=k, infatti A è in Z

Le produzioni A -> ε sono eliminate

Esempio:

S-> AB

A-> a AA | ε

B-> bBB | ε

Z={A,B,S} quindi G’ ha

S-> AB | A | B

A->aAA | aA | aA | a

B-> bBB | bB | bB | b

<u>Eliminazione delle produzioni unitarie:</u>

A -> B

esempio:

I -> a | b | Ia | Ib

F -> I | (E)

T -> F | T*F

E -> T | E + T

<u>idea</u>: al posto di E -> T, E -> F | T*F e al posto di E ->F, E -> I | (E) e finalmente E -> a | b | Ia | Ib

Costruiamo tutte le coppie (A,B) t.c. A =>* B con solo produzioni unitarie:
base (A,A) per ogni variabile A
induzione se (A,B) e B -> C allora aggiungiamo (A,C)

(E,E) e E ->T danno (E,T)
(E,T) e T -> F danno (E,F)
(E,F) e F -> I danno (E,I)
(T,T) e T -> F danno (T,F)
(T,F) e F -> I danno (T,I)
(F,F) e F -> I danno (F,I)

I -> a | b | Ia | Ib

F -> I | (E)

T -> F | T*F

E -> T | E + T

(E,E) E -> E + T

(E,T) E -> T * F

(E,F) E -> (E)

(E,I) E -> a | b | Ia | Ib

(T,T) T -> T * F

(T,F) T -> (E)

(T,I) T -> a | b | Ia | Ib

(F,F) F -> (E)

(F,I) F -> a | b | Ia | Ib

(I,I) I -> a | b | Ia | Ib

La forma normale di Chomsky (CNF) per le CFG prevede che tutte le produzioni siano:

A -> a, con a in T

A-> BC, con B e C variabili

1) eliminiamo simboli inutili
2) eliminiamo la empty word
3) eliminiamo le produzioni unitarie

restano produzioni A -> a   ok e anche
A -> aBcDF -> A -> KBK’DF con K -> a e K’ -> c

quindi avremo A -> KBK’DF
A ->K<BK’DF> e <BK’DF> -> B<K’DF> e <K’DF>
-> K’<DF> e finalmente <DF> -> DF

Teorema 7.17.

Negli alberi di derivazione di una CFG in CNF è vero che se il cammino più lungo è di lunghezza n, allora il prodotto w è t.c. $|w| <= 2^{n-1}$

Dimostrazione: per induzione su n

Base: n=1 l'albero consiste della radice e di una foglia etichettata con un terminale

Prodotto w = foglia quindi $|w|=1 = 2^{1-1}$

Induzione: altezza n. La radice usa A->BC
B e C sono radici di alberi di altezza al massimo n-1,

quindi i loro prodotti, per ipotesi induttiva, sono di lunghezza al più 2^{n-2}, e $2 * 2^{n-2} = 2^{n-1}$

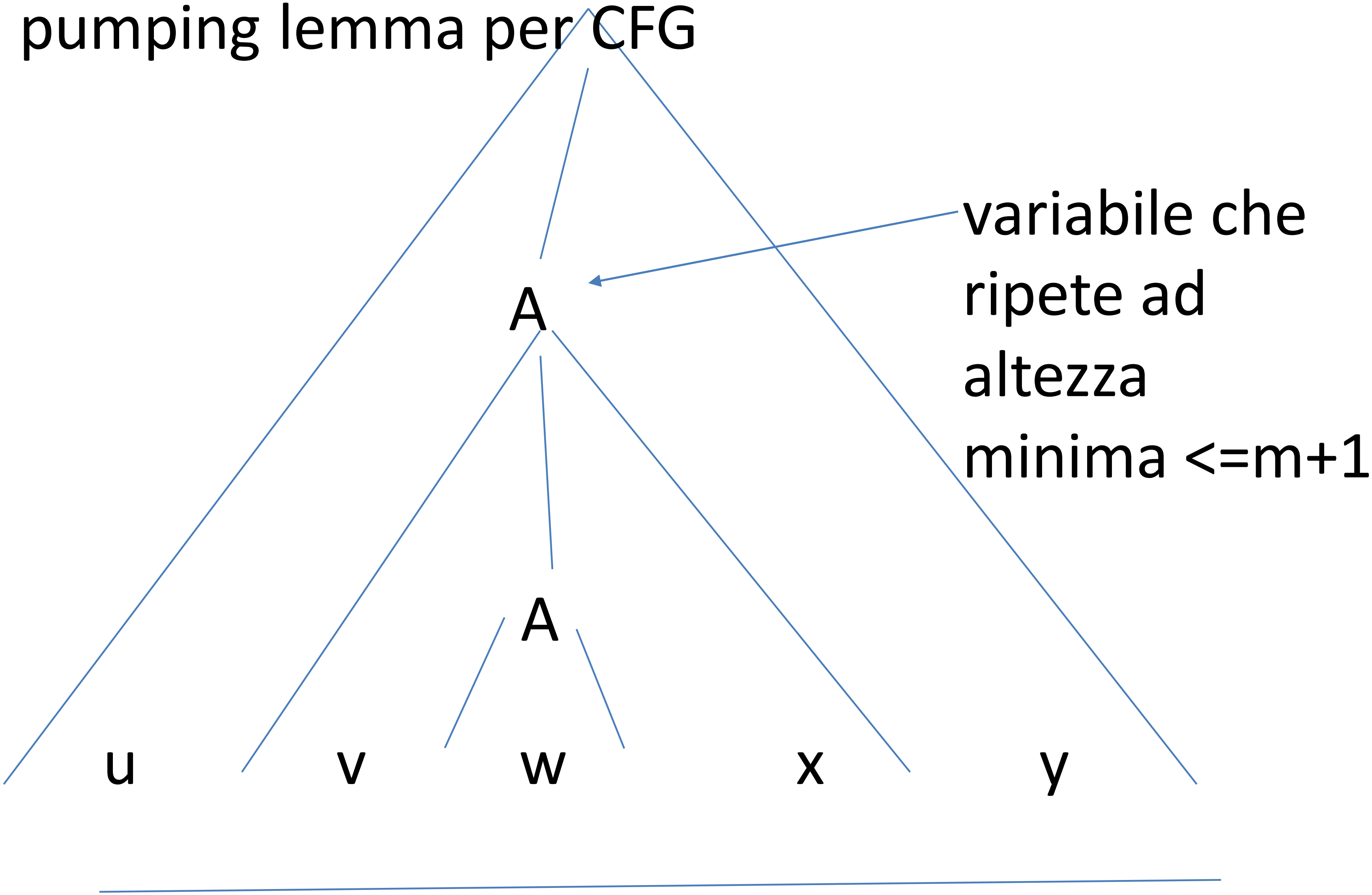
Conseguenza:

se m è il numero delle variabili della grammatica in CNF:
un albero con prodotto w t.c. $|w|>=2^m$ deve avere un cammino di lunghezza $>=m+1$

e allora su quel cammino almeno una variabile ripete

prendiamo $n = 2^m$

pumping lemma per CFG



z = uvwxy

con |vwx| <=n e vx != ε

$z = uvwxy$

ma anche uv^nwx^ny, con $n>=0$, è generabile e quindi è nel linguaggio

Serve a dimostrare che certi linguaggi non sono CF perché se lo fossero avremmo una contraddizione

Esempio: $\{0^n1^n2^n \mid n > 0\}$ non è CF

Per il pumping lemma, esiste n tale che ogni z più lunga di n è z=uvwxy. Scegliamo
$z = 0^n1^n2^n$ sappiamo che $|vwx| <= n$ e quindi o contiene degli 0 o contiene dei 2, ma non entrambi.
Supponiamo che contengano 0 e non 2.
Quindi ripetendo v e x (non entrambi vuoti) aumento il n. di 0 e 1 mantenendo inalterato il n. di 2 => otterrei una stringa che non è più nel linguaggio