Lezione 5

CFG e PDA definiscono gli stessi linguaggi grammatiche ≡ PDA per stack vuoto ≡ PDA per stato finale

la direzione più intuitiva va dalle grammatiche ai PDA per stack vuoto

si costruisce un PDA che simula le derivazioni leftmost

ogni forma sentenziale lm non terminale è:
$xA\alpha$

--x è stringa di terminali

--A è la variabile più a sinistra

-- α è stringa di variabili e di terminali

$A\alpha$ è la <u>coda</u> della forma sentenziale

Intuizione:

se $S \Rightarrow^{*lm} xA\alpha \Rightarrow^{*lm} w$, con $w=xw'$

allora il PDA

$(q,w,Z0) \vdash^{*} (q,w',A\alpha) \vdash (q,\varepsilon,\varepsilon)$

e se $A \rightarrow \beta$, allora

$\delta(q, \varepsilon, A) = \{(q, \beta)\}$

e se β contiene terminali ? Facciamo match con l'input il PDA è non deterministico e può «provare» <u>tutte le derivazioni</u> a partire da S e usa i terminali prodotti per consumare l'input

e quando <u>indovina</u> la derivazione che corrisponde all'input, consuma tutto l'input e contemporaneamente lo stack si svuota

sono tutte derivazioni leftmost.

Costruzione:

sia $G=(V,T,P,S)$, costruiamo $P=(\{q\}, T, V+T, \delta, q, S)$, dove δ è come segue:

1) per ogni variabile A in V,
$\delta(q,\varepsilon,A)=\{(q,\beta) \mid A \rightarrow \beta$ è una produzione in $P\}$

2) per ogni a in T,
$\delta(q,a,a)=\{(q, \varepsilon)\}$ per consumare l'input

esempio:

I -> a | b |Ia | Ib |I0 | I1

E -> I | E*E | E+E | (E)

non determinismo fondamentale

I terminali del PDA sono {a,b,0,1,(,),+,*}, questi assieme a E e I sono i simboli dello stack.

a) $\delta(q, \varepsilon, I)=\{(q,a),(q,b),(q,Ia),(q,Ib),(q,I0),(q,I1)\}$

b) $\delta(q,\varepsilon,E)=\{(q,I),(q,E*E),(q,E+E),(q,(E))\}$

c) $\delta(q,a,a)=\{(q, \varepsilon)\}$, $\delta(q,b,b)=\{(q, \varepsilon)\}$, $\delta(q,0,0)=\{(q, \varepsilon)\}$, $\delta(q,1,1)=\{(q, \varepsilon)\}$, $\delta(q,(,()=\{(q, \varepsilon)\}$, $\delta(q,*,*)=\{(q, \varepsilon)\}$..ecc.

Teorema 6.13

Se P è il PDA costruito da G allora N(P)=L(G)

Dimostrazione: w è in N(P) sse w è in L(G)

(<=) supponiamo che esista $S=\gamma 1 \Rightarrow_{lm} \gamma 2 \Rightarrow_{lm} \gamma 3 \Rightarrow_{lm} \ldots \Rightarrow_{lm} \gamma n=w$

Per induzione su i in [1..n] dimostriamo che per ogni i, $(q,w,S)\vdash^*(q,yi,\alpha i)$, tale che $\gamma i=xi\alpha i$, con $xiyi=w$, quindi xi è quello che è stato consumato dell'input e yi quello che resta da consumare.

Base: $i=1$,

$\gamma 1=S$, quindi $x1=\varepsilon$ e $y1=w$.

Dato che $(q,w,S)|\!-^*(q,w,S)$ in 0 mosse, la base è ok

<u>induzione:</u> assumiamo che sia vera per i >=1, cioè che $(q,w,S)|\!-^*(q,yi,\alpha i)$, con $\gamma i=xi\alpha i$ e $w=xiyi$ e dimostriamo che vale per i+1.

$\gamma i=xiA\eta$ e $\alpha i = A\eta$, e supponiamo che
$xiA\,\eta \Rightarrow_{lm} xi\beta\eta$ usando la produzione $A \rightarrow \beta$
per costruzione, c'è la transizione,
$\delta(q,\varepsilon,A)=\{(q, \beta)\}$ e quindi
$(q,yi,A\,\eta)|\!- (q, yi, \beta\eta)$ se in $\beta\eta$ ci sono terminali li elimina tutti consumando yi fino ad ottenere nello stack la coda di $\gamma i+1$

per $i=n$ si arriva ad αn che è la coda di $\gamma n=w$ e quindi $\alpha n=\varepsilon$ e $yn=\varepsilon$, per cui,
$(q,w,S)|\text{-}^*(q, \varepsilon, \varepsilon)$ quindi P accetta w

(=>) se P ha questo calcolo, $(q,x,A)\ |\text{-}^*(q, \varepsilon, \varepsilon)$ allora G ha questa derivazione, $A \Rightarrow^* x$

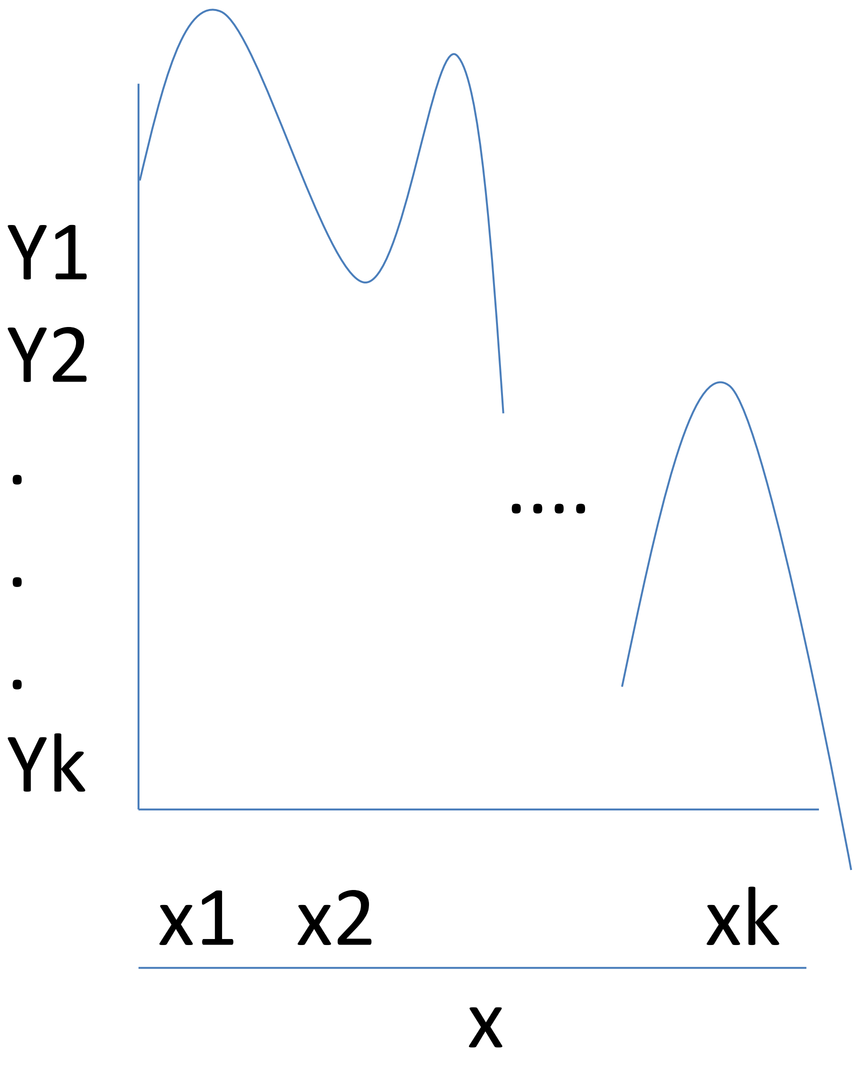
lo dimostriamo per induzione sul numero di mosse di P

Base: una sola mossa, $(q, x, A) \vdash (q, \varepsilon, \varepsilon)$

Dalla costruzione di P, in G deve esserci $A \rightarrow \varepsilon$
quindi $x = \varepsilon$ e $A \Rightarrow \varepsilon$

Induzione: supponiamo che in P,
$(q,x,A) \vdash_n (q,x,Y1\ldots Yk)$, con $n>1$
la prima mossa deve essere del tipo (1),
$(q,x,A) \vdash (q,x,Y1\ldots Yk)$
che corrisponde alla produzione $A \rightarrow Y1\ldots Yk$

dopo di che n-1 mosse eliminano $Y1\ldots\ldots Yk$ e x

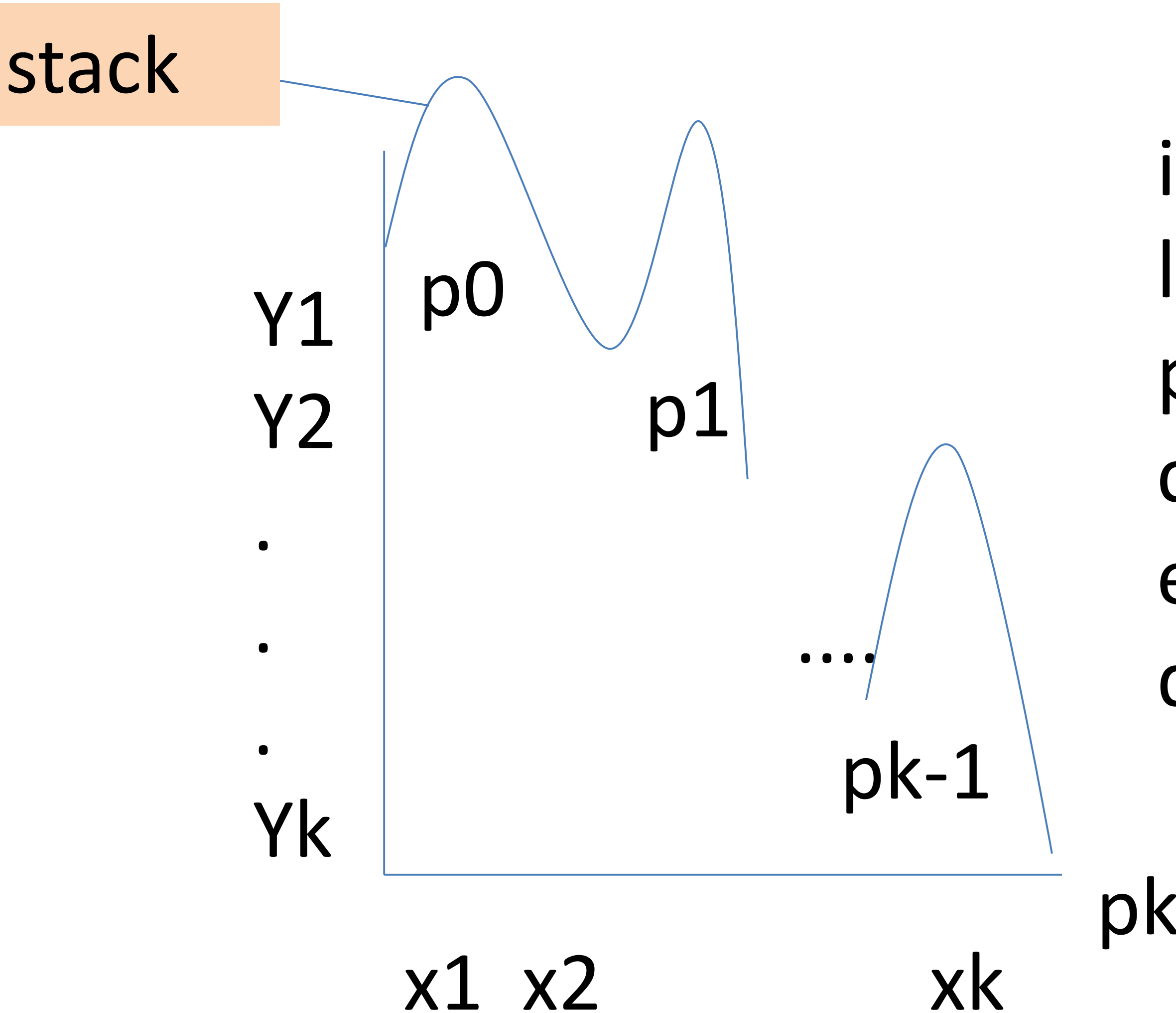

Y1
Y2
.
.
.
Yk
....
x1 x2 xk
X

possiamo scomporre $x=x_1...x_k$ dove x_1 è la parte di input consumata finché si elimina Y_1, x_2 quella consumata finché si elimina Y_2 e così via.
Se Y_i è terminale allora la sua eliminazione avviene attraverso una mossa (2) che consuma $x_i=Y_i$ dall'input quindi $(q, x_i...x_k, Y_i) \vdash^* (q, x_{i+1}...x_k, \varepsilon)$

per ogni $i=1,2,..k$, la corrispondente computazione è più corta di n e quindi, per ipotesi induttiva, $Y_i \Rightarrow^* x_i$ e quindi $A \Rightarrow Y_1...Y_k \Rightarrow^* x_1...x_k = x$

se $A=S$ e $x=w$, da $(q,w,S) \vdash^* (q, \varepsilon, \varepsilon)$ segue che $S \Rightarrow^* w$

ora consideriamo da PDA → CFG



importano
le mosse in cui si fa
pop di Y1...Yk
con stati p0..pk-1
e consumo
dell'input x1...xk

la grammatica ha variabili che rappresentano
--l’eliminazione di un simbolo X dallo stack
-- e il passaggio dallo stato p allo stato q dopo l’eliminazione di X

queste variabili saranno [p X q]

Dato $P=(Q,\Sigma,\Gamma,\delta, q0, Z0)$ costruiamo $G=(V, \Sigma,R,S)$, dove V contiene:

1. il simbolo iniziale S
2. [p X q] per ogni X in Γ e p e q in Q

R contiene

a) S ->[q0 Z0 p] per ogni p in Q,

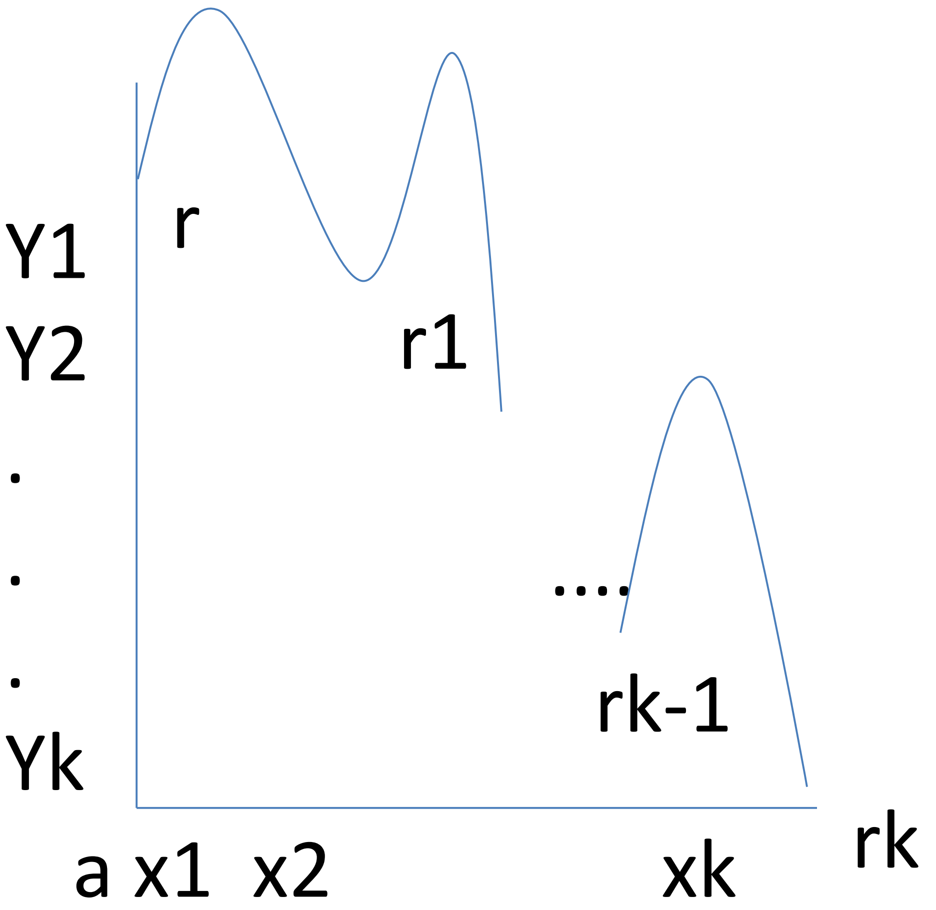
[q0 Z0 p] di G deve generare tutte le stringhe w che sono accettate da P con stack vuoto in stato p, cioè le stringhe consumate fino a che Z0 viene eliminato (svuotando lo stack)

(b) supponiamo che $\delta(q, a, X)$ contenga $(r, Y1\ldots Yk)$, dove a in $\Sigma \cup \{\varepsilon\}$ e $k>=0$

allora, per ogni r1…rk in Q, R contiene
[q X rk] -> a [r Y1 r1][r1 Y2 r2]…[rk-1 Yk rk]

questa produzione di G corrisponde alla computazione di P che arriva ad avere lo stack corrente meno X e deve generare la stringa terminale che P consuma in questa computazione

ci deve essere un tale calcolo e r, r1, ...rk
la grammatica li prova tutti



funziona !!

[q X p]=>* w sse (q, w, X) |-* (p, ε, ε)

(<=) per induzione sulla computazione,

Base: 1 mossa, w deve essere un terminale o ε, per cui [q X p] -> w è in R e [q X p] => w

Induzione: supponiamo che (q, w, X) |-* (p, ε, ε) in n passi con n>1, la prima mossa deve avere la forma,

(q, w, X) |-* (r0, x, Y1...Yk) |-* (p, ε, ε) dove w=a x, per a in $\Sigma \cup \{\varepsilon\}$

dalla fig precedente: siano r1...rk gli stati quando Y1...Yk sono eliminati e x1...xk= x i pezzi di x consumati,

quindi (ri-1, xi, Yi) |-* (ri, ε, ε) e per ipotesi induttiva [ri-1 Yi ri] =>* wi

ed esiste in G:
[q X rk] -> a [r0 Y1 r1][r1 Y2 r2]....[rk-1 Yk rk]

quindi [q X rk]=>* a x1 x2 ...xk = w

(=>) per induzione sul numero di passi della derivazione.

<u>Base</u>: 1 passo, [q X p] => a allora $\delta(q, a, X)$ contiene (p, ε) e quindi (q,a,X) |- $(p, \varepsilon, \varepsilon)$

<u>Induzione</u>: assumiamo che [q X rk] =>* w in n>1 passi, il primo passo è del tipo:

[q X rk] => a [r0 Y1 r1][r1 Y2 r2]...[rk-1 Yk rk] =>* ax = w

e quindi per ogni i=1,2,...k

[ri-1 Yi ri]=>*xi con x1 x2...xk=x

per ipotesi induttiva, per ogni i=1,2,..k
(ri-1, xi, Yi) |-* (ri, ε, ε)

e quindi per Teorema 6.5 vale anche
(ri-1, xixi+1..xk YiYi+1...Yk) |-* (ri, xi+1..xk, Yi+1...Yk)

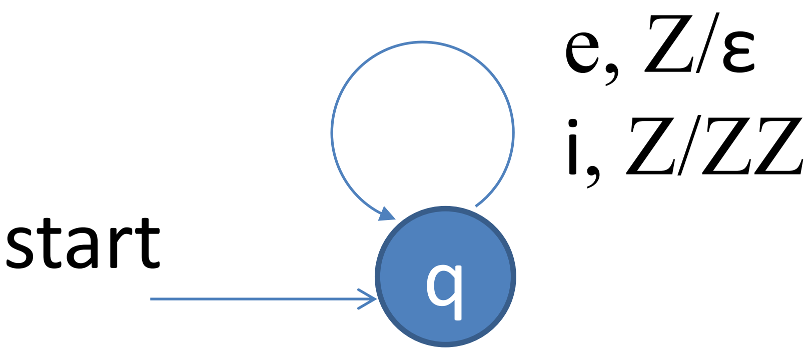
ma la produzione [q X rk] => a [r0 Y1 r1][r1 Y2 r2]...[rk-1 Yk rk] deriva dal fatto che (r0 ,Y1 Y2 ...Yk) è in δ(q,a,X) per cui
(q, ax1...xk,X) |- (r0,x1...xk, Y1...Yk) |-*

$(r1, x2..xk, Y2..Yk) \vdash^* (r2, x3...xk, Y3..Yk) \vdash^* (rk, \varepsilon, \varepsilon)$

per finire basta osservare, per il punto (a) della costruzione, che $S \Rightarrow^* w$ sse per qualche p, $[q\, Z0\, p] \Rightarrow^* w$

da questo, la prova precedente dimostra che $S \Rightarrow^* w$ sse $(q,w,Z0) \vdash^* (p,\varepsilon,\varepsilon)$

esempio: $P=(\{q\}, \{i,e\}, \{Z\},\delta,\theta,Z)$



riconosce $L = \{w \mid w$ in $\{i,e\}^*$ di lunghezza minima con più e di i$\}$

la grammatica ha solo 2 variabili: S e [q Z q]

S -> [q Z q]

[q Z q] -> e

[q Z q] ->i [q Z q][q Z q]

S -> [q Z q]
[q Z q] ->i [q Z q][q Z q]
[q Z q] -> e

S => [q Z q] => i [q Z q] [q Z q]=>i e [q Z q] => i e e

genera linguaggio delle stringhe di lunghezza minima che hanno più e di i