

Esempi :

- $(a+b)^* a$ denota $\{a,b\}^* \{a\}$
- Google $\{ \text{Google}, \text{Gooogle}, \text{Goooole}, \dots \}$
- identificatori di variabili
 $(A, B, \dots, z, a, \dots, g) (A, B, \dots, z, a, \dots, z, 0, \dots, g)^*$
", "=" + "
- importi in euro > 0
 $(1+2+\dots+9)(0+1+\dots+9)^*, (0+1+\dots+9)^2$

Teorema di Kleene

L è denotato da una E.R.



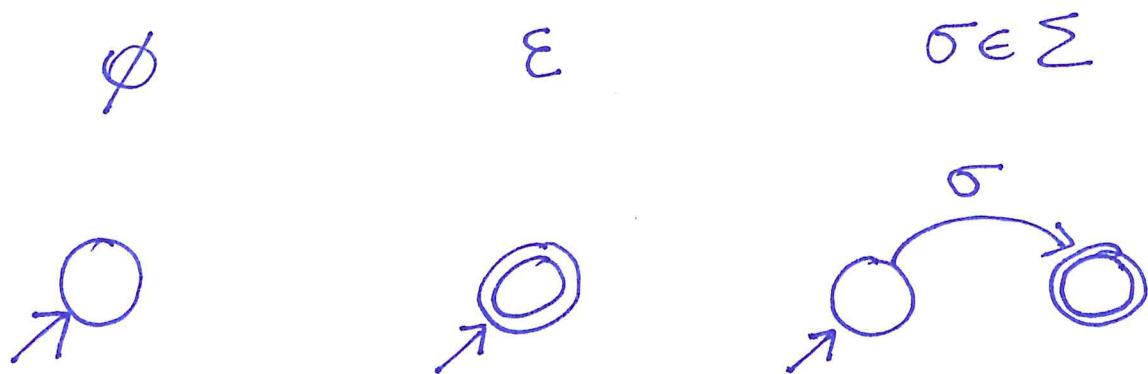
L è riconosciuto da un DFA

ohim :

L definito da E.R. $\Rightarrow L$ ammette un DFA

Tecnica : per induzione.

caso base Esistono DFA per E.R. base:



passo induttivo

Se esistono DFA per E.R. p e q
allora dimostro che esistono DFA

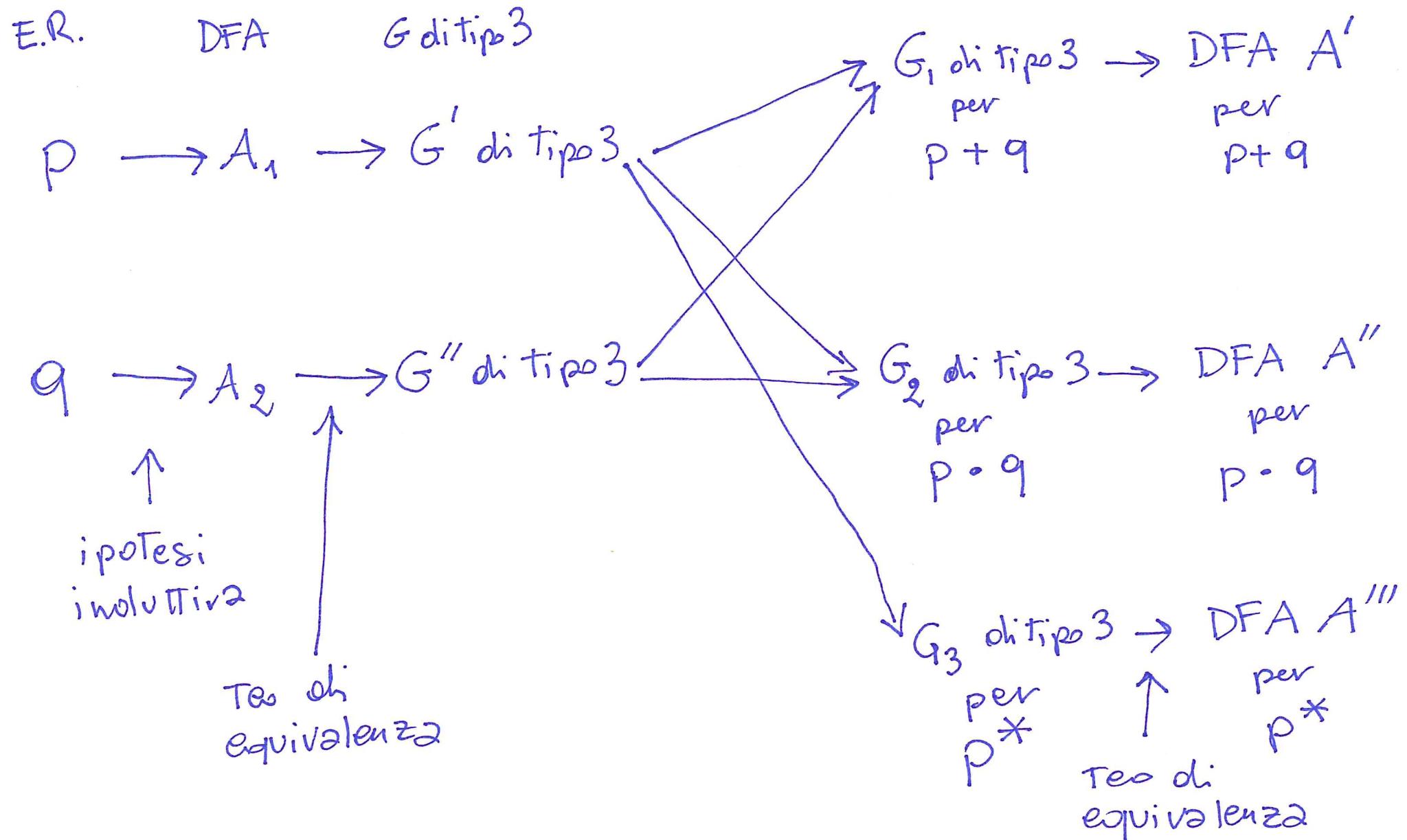
per:

$p + q$

$p \cdot q$

p^*

In realtà useremo le grammatiche di tipo 3:



Abbiamo 2 gramm. di tipo 3:

$$A \rightarrow \sigma B$$

$$\cancel{A \rightarrow \sigma}$$

$$A \rightarrow \epsilon$$

$$G' = (T', V, S', P')$$

$$G'' = (T'', V'', S'', P'')$$

che generano

$$L(G') = \{ x \in T'^* \mid S' \xrightarrow{*} x \}$$

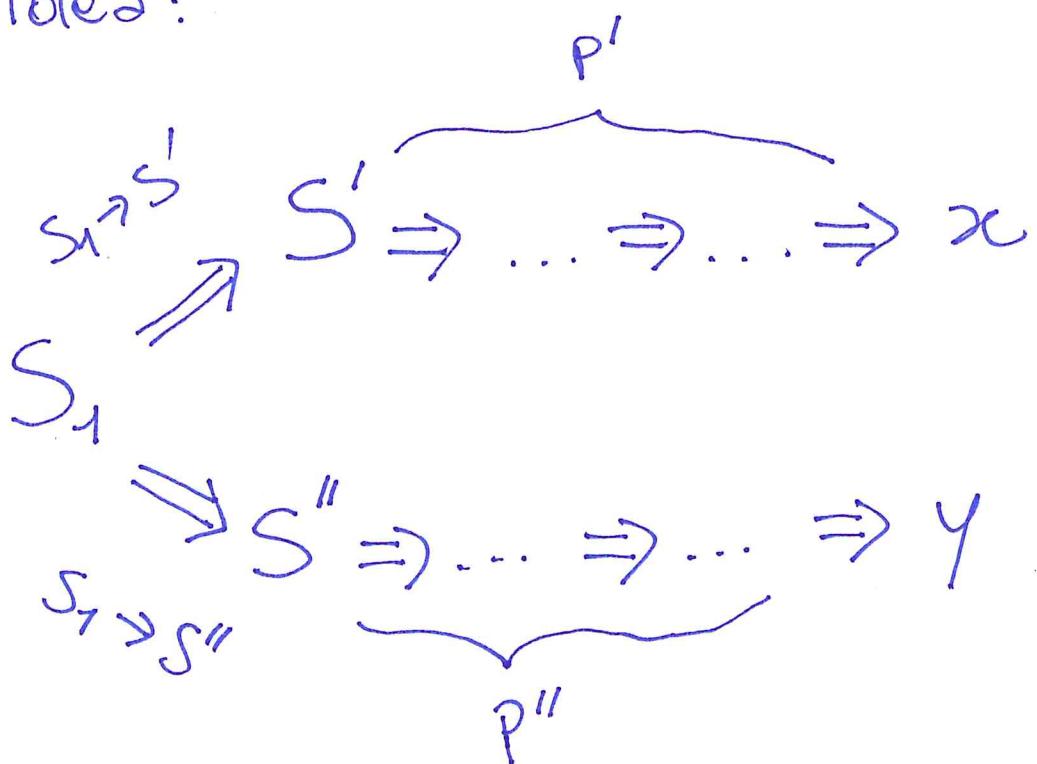
$$L(G'') = \{ y \in T''^* \mid S'' \xrightarrow{*} y \}$$

$$T' = T'' = \Sigma$$

① Costruisco G_1 per $L(G') \cup L(G'')$

$$L(G') \cup L(G'') = \left\{ \omega \in \Sigma^* \mid \begin{array}{l} \omega \in L(G') \\ \text{oppure} \\ \omega \in L(G'') \end{array} \right\}$$

idea:



richiesta $V' \cap V'' = \emptyset$

$$G_1 = (\Sigma, V' \cup V'' \cup \{S_1\}, S_1, P' \cup P'' \cup \{S_1 \rightarrow S', S_1 \rightarrow S''\})$$

G_1 è di tipo 3?

Si, è lineare a destra:

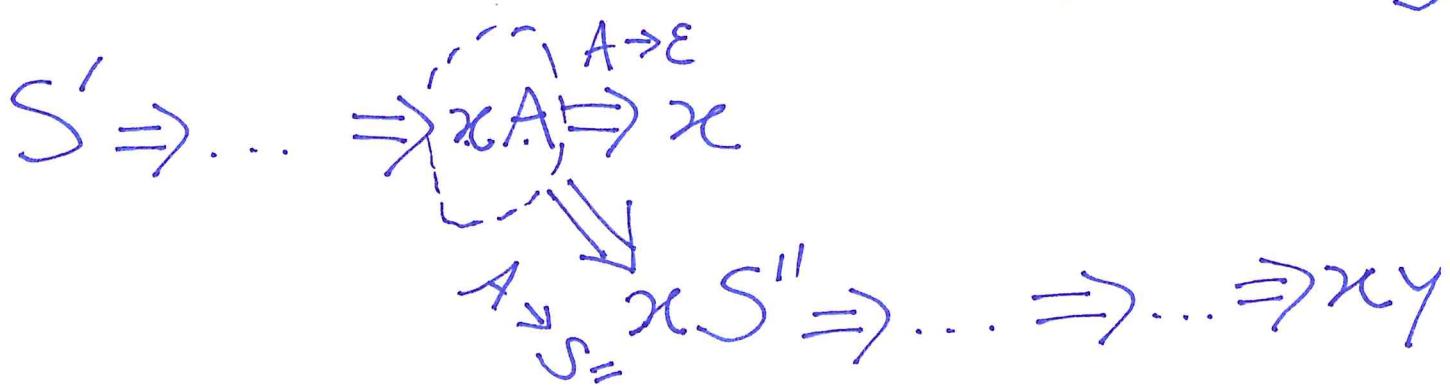
$$A \rightarrow xB$$

$$A \rightarrow y$$

$$x, y \in \Sigma^*$$

② Costruisco G_2 per $L(G') \cdot L(G'')$

$$L(G') \cdot L(G'') = \{xy \mid x \in L(G') \text{ e } y \in L(G'')\}$$



ES: $A \rightarrow aB, B \rightarrow bC, C \rightarrow cD,$
 $D \rightarrow \epsilon$

$\underset{--}{A} \Rightarrow \underset{--}{a} \underset{--}{B} \Rightarrow \underset{--}{a} \underset{--}{b} \underset{--}{C} \Rightarrow \underset{---}{a} \underset{---}{b} \underset{---}{c} \underset{---}{D} \Rightarrow abc$

$G_2 = (\Sigma, V' \cup V''; S',$

$P' \setminus \{A \rightarrow \epsilon \mid A \in V'\} \cup$

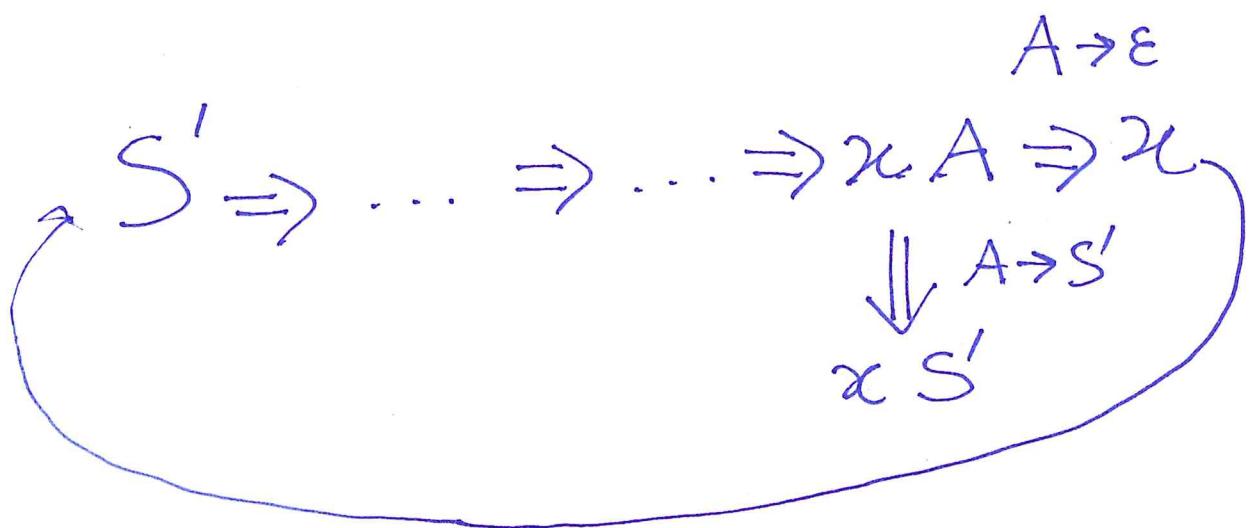
$\cup P'' \cup \{A \rightarrow S'' \mid A \rightarrow \epsilon \in P'\},$

G_2 è di tipo 3?

Si!

③ Costruisco G_3 per $L(G')^*$

$$L(G')^* = \bigcup_{i=0}^{\infty} L(G')^i = \{\epsilon\} \cup L(G')^+$$



$$G_3 = (\Sigma, V', S',$$

$$P' \cup \{ A \rightarrow S' \mid A \rightarrow \epsilon \in P' \}$$

Se $\epsilon \notin L(G')^+$ allora:

$$S_3 \rightarrow \epsilon, S_3 \rightarrow S'$$

Teorema di Kleene

L è denotato da un'espressione regolare



L è riconosciuto da un automa a stati finiti

ohimè: (\Updownarrow) Da un DFA a una ER

Dato il DFA $A = (\Sigma, Q, q_0, \delta, F)$,

sia $Q = \{q_0, q_1, q_2, \dots, q_K\}$,

allora posso associare ad A

i seguenti DFA:

$$A_0 = (\Sigma, Q, q_0, \delta, F),$$
$$A_1 = (\Sigma, Q, q_1, \delta, F),$$

⋮

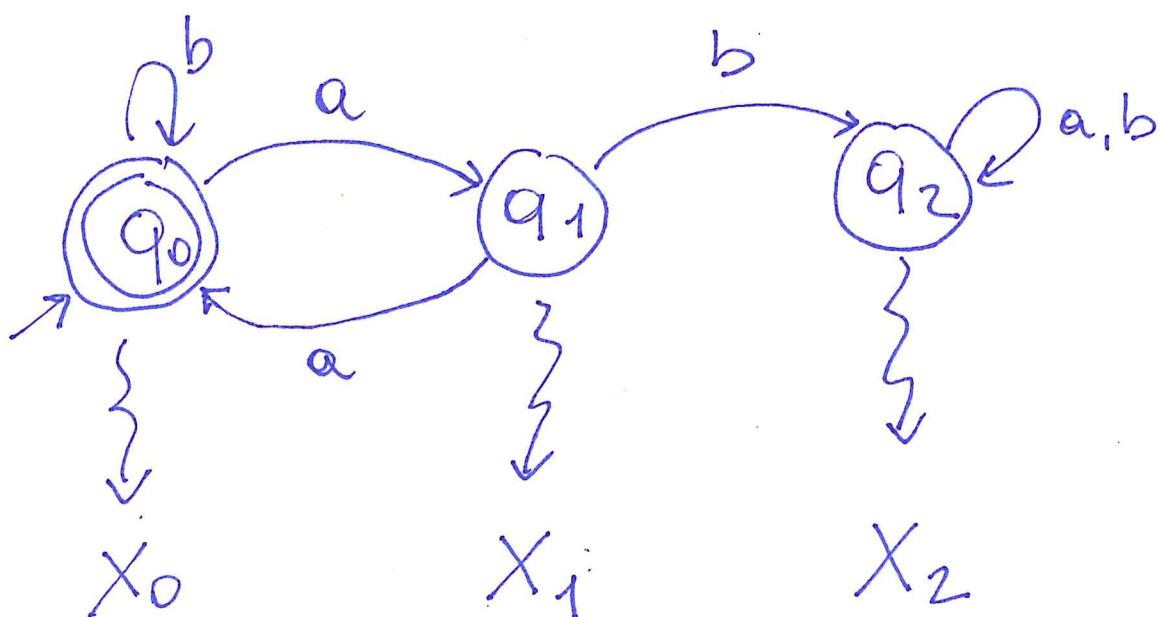
$$A_K = (\Sigma, Q, q_K, \delta, F).$$

-) Ognuno di questi DFA riconosce un linguaggio.

Sia X_i il linguaggio riconosciuto da A_i .

Pertanto: $L(A) = L(A_0) = X_0$.

Esempio:



-) Adesso cerchiamo di esprimere ogni linguaggio in funzione degli altri.

Esempio:

$$X_1 = \{ w \in \Sigma^* \mid q_1 \xrightarrow{w} \textcircled{O} \} =$$

$$= \{ \sigma z \in \Sigma^* \mid q_1 \xrightarrow{\sigma z} \textcircled{O} \} =$$

$$= \{ az \in \Sigma^* \mid q_1 \xrightarrow{a} q_0 \xrightarrow{z} \textcircled{O} \} \cup$$

$$\cup \{ bz \in \Sigma^* \mid q_1 \xrightarrow{b} q_2 \xrightarrow{z} \textcircled{O} \} =$$

$$= a \{ z \in \Sigma^* \mid q_0 \xrightarrow{z} \textcircled{O} \} \cup$$

$$\cup b \{ z \in \Sigma^* \mid q_2 \xrightarrow{z} \textcircled{O} \} =$$

$$X_1 = a \cdot X_0 + b \cdot X_2$$

In generale:

$$X_i = \sum_{\sigma \in \Sigma} \sigma X_j [+\varepsilon]$$

t.c.

$$\delta(q_i, \sigma) = q_j$$

↓
solo se
 $q_i \in F$

Esempio:

$$\left\{ \begin{array}{l} X_0 = aX_1 + bX_0 + \varepsilon \\ X_1 = aX_0 + bX_2 \\ X_2 = aX_2 + bX_2 \end{array} \right.$$

•) Si ricava un sistema con $K+1$ equazioni in $K+1$ incognite X_i
↑
numero di stati di A

•) Per risolvere il sistema ho bisogno di riformi ad un'equazione tipo:

$$X = A X + B$$

la cui soluzione è:

$$X = A^* B$$

se $\epsilon \notin A$ allora la soluzione è unica

se $\epsilon \in A$ allora $A^* B$ è la minima soluzione

Verifica:

$$A^*B \quad A^*B \\ \downarrow \quad \cancel{\downarrow} \\ X = AX + B$$

$$\underbrace{A^*B}_{=} = A(A^*B) + B =$$

$$= A\left(\left(\bigcup_{i=0}^{\infty} A^i\right) B\right) + B =$$

$$= \left(\bigcup_{i=0}^{\infty} A^{i+1}\right) B + B =$$

$$= \left(\bigcup_{i=1}^{\infty} A^i\right) B + \{\varepsilon\} B =$$

$$= \left(\bigcup_{i=1}^{\infty} A^i + \{\varepsilon\}\right) \cdot B$$

$$= \left(\bigcup_{i=0}^{\infty} A^i\right) B = \underbrace{A^*B}_{=}$$

Esempio:

$$\begin{cases} X_0 = aX_1 + bX_0 + \epsilon \\ X_1 = aX_0 + bX_2 \\ X_2 = aX_2 + bX_2 \end{cases}$$

Vediamo $X = Ax + B$

nell'ultima equazione:

$$X_2 = \underbrace{(a+b)}_A X_2 + \underbrace{\emptyset}_B$$

$$X_2 = A^*B = (a+b)^* \cdot \emptyset = \emptyset$$

Elimino una variabile da
sistema sostituendo l'espressione
ricavata:

$$X_2 = \emptyset$$

$$X_1 = aX_0 + bX_2$$

$$X_1 = aX_0 + b \cdot \emptyset = aX_0$$

$$X_1 = A X_1 + B$$
$$\downarrow$$
$$\emptyset \qquad \qquad \qquad aX_0$$

$$X_1 = A^* B = \emptyset^* \cdot aX_0$$
$$= \varepsilon \cdot aX_0 = aX_0$$

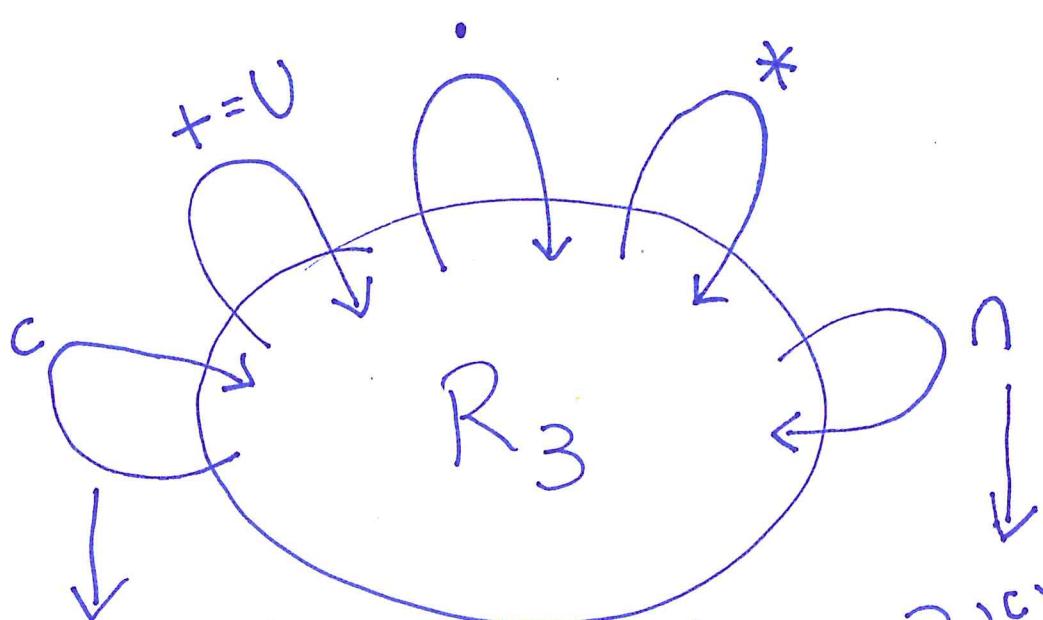
$$X_0 = aX_1 + bX_0 + \varepsilon$$

$$X_0 = aaX_0 + bX_0 + \varepsilon$$

$$X_0 = \underbrace{(aa+b)}_A X_0 + \underbrace{E}_B$$

$$\begin{aligned} X_0 &= A^* B = \\ &= (aa+b)^* \cdot E = (aa+b)^* \end{aligned}$$

Chi usura dei linguaggi regolari



basta scambiare
gli stati finali con
i non finali

$$\begin{aligned} ((A \cap B)^c)^c &= \\ (A^c \cup B^c)^c &= \end{aligned}$$

Linguaggi liberi dal contesto e grammatiche di tipo 2

Problema dell'ambiguità

Esempio in italiano:

Il prof. dice lo studente è un asino.

Definizione di G ambigua:

Una grammatica è ambigua quando genera una parola ambigua cioè quando la parola ammette due alberi di derivazione diversi.

Nota: l'albero di derivazione dà significato alla parola.

Esempio di G ambigua:

$P \rightarrow i(x) \vee o(x)$

$C \rightarrow A \mid I \mid E$

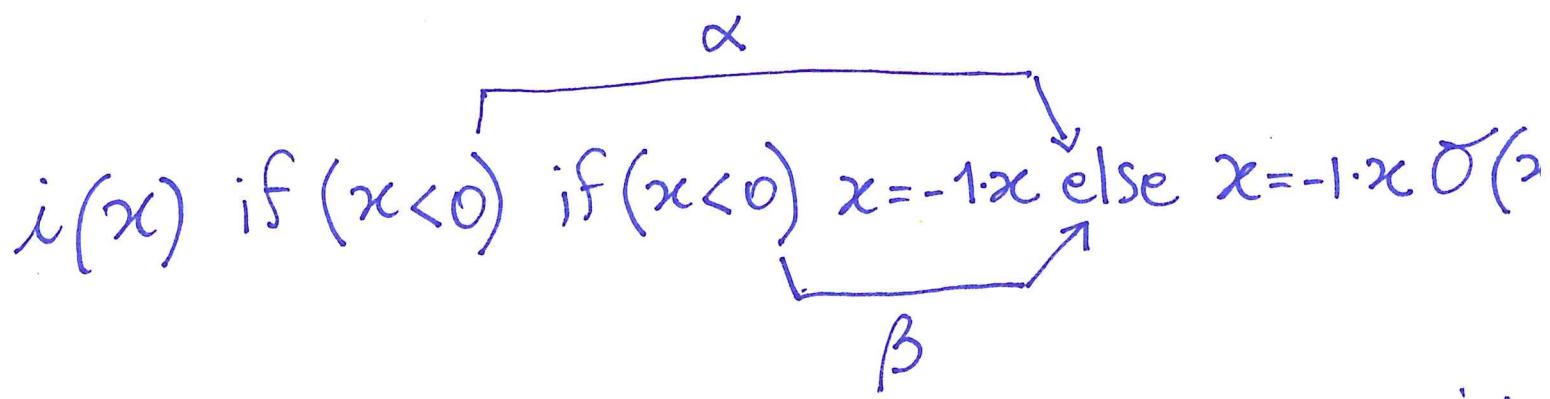
$A \rightarrow x = -1 \cdot x$

$I \rightarrow \text{if } (T) \ C$

$E \rightarrow \text{if } (T) \ C \ \text{else } C$

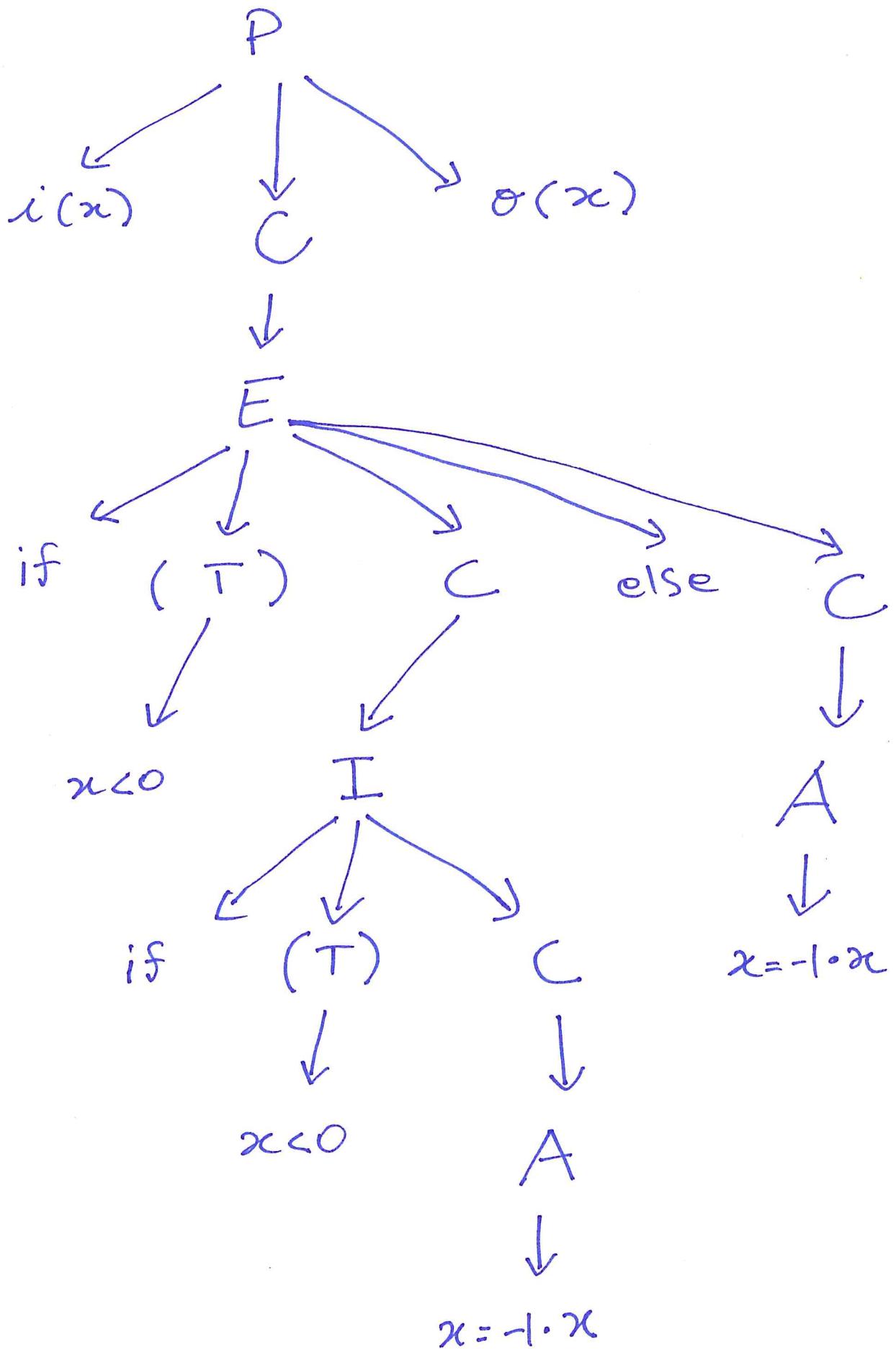
$T \rightarrow x < 0$

G genera la seguente parola:

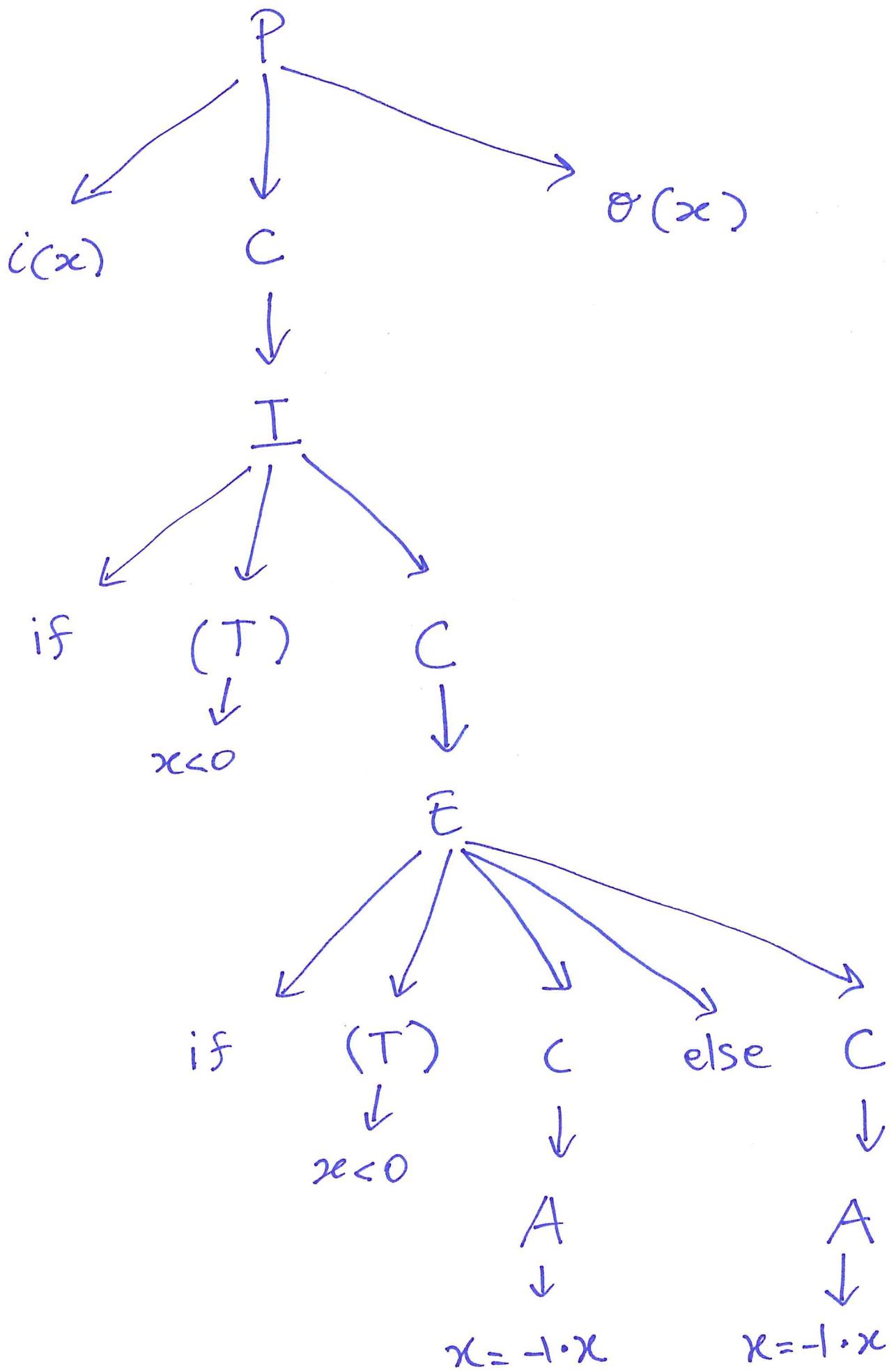


la quale ammette due alberi di derivazione

②



B



L'albero dà significato alla parola
in questo caso associa la funzione
calcolata dal programma

(A)

$$F_A(x) = \begin{cases} -x & x \geq 0 \\ -x & x < 0 \end{cases} \Rightarrow -x$$

(B)

$$F_B(x) = \begin{cases} x & x \geq 0 \\ -x & x < 0 \end{cases} \Rightarrow |x|$$

In conclusione il programma è ambiguo
e G è ambigua

Definizione

Una grammatica G è NON ambigua se ogni parola generata non è ambigua.

Osservazione

A volte è possibile disambiguare una grammatica.

Nel nostro caso si può usare una convenzione:

l'else è associato all'if più vicino

A volte no!

Definizione

Un linguaggio si dice

INERENTEMENTE AMBIGUO

quando ogni G che lo genera
è ambigua.

NOTA:
è di tipo 2



Esempio

$$\{ a^i b^j c^k \mid i=j \text{ o } j=k \}$$

idea:

$$S \xrightarrow{*} a^n b^n c^m \rightarrow i=j$$

$$\xrightarrow{*} a^m b^n c^n \rightarrow j=k$$

$a^n b^n c^n$ ammetterà due
alberi di derivazione \neq

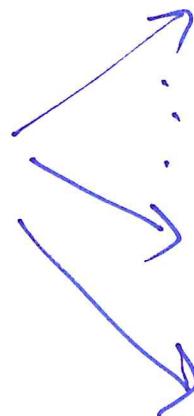
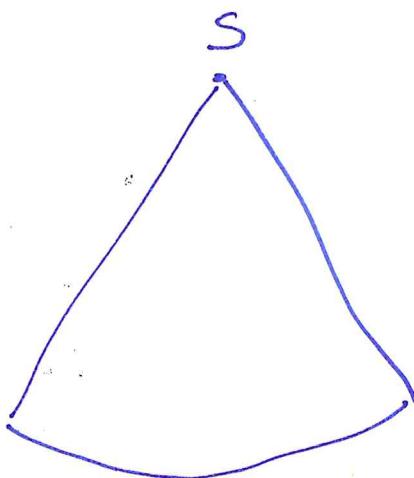
Domanda :

Un linguaggio regolare
può essere inherentemente ambiguo?

Risposta : No

dim.

Ad ogni L regolare corrisponde
un DFA. Da tale DFA si
ricava una Q di tipo 3 che
è NON AMBIGUA.

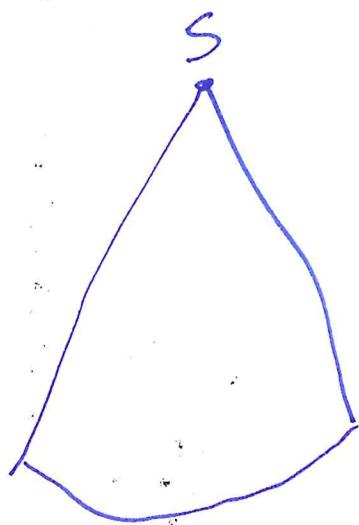


ho
più derivazioni
associate

Derivazione leftmost:

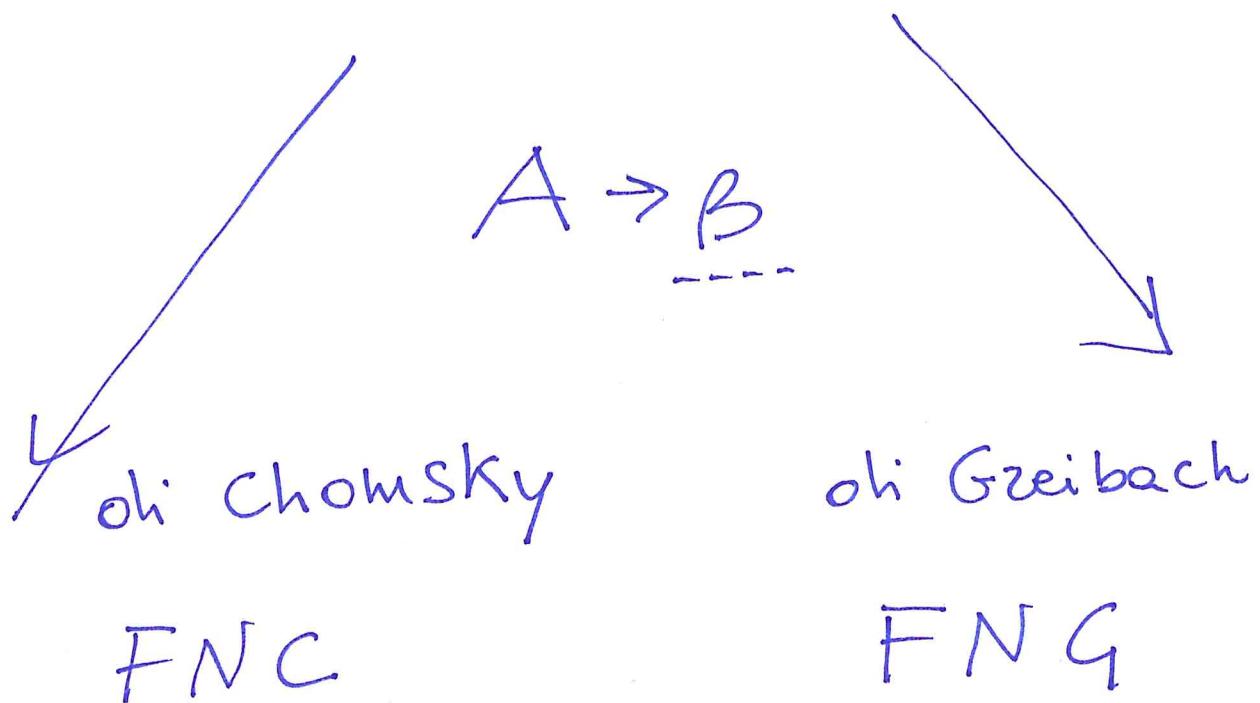
Espandi per primo il metasimbolo
più a sinistra.

es: abb A a BC
 ↑



è associato
una unica
derivazione
leftmost

FORME NORMALI PER Grafi Tip 2 (senza ε)



$$A \rightarrow BC$$

$$A \rightarrow \sigma W$$

$$A \rightarrow \sigma$$

$$A \in V$$

$$A, B, C \in V$$

$\sigma \in T$

$$\sigma \in T$$
$$W \in V^*$$

Esempio

$$G = \langle \{x, y, +, (,)\}, \{E\}, P, E \rangle$$

oltre $P = \{ E \rightarrow (E + E) \mid x \mid y \}$

Ricaviamo FNC e FNG:

FNG:

$$E \rightarrow x \quad \text{o.k.}$$

$$E \rightarrow y \quad \text{o.k.}$$

$$\cancel{E \rightarrow (E + E)}$$

↑ ↑ NO, la trasform

$$E \rightarrow (E P E D) \quad \text{o.k.}$$

$$P \rightarrow + \quad \text{o.k.}$$

$$D \rightarrow) \quad \text{o.k.}$$

FNC :

$E \rightarrow x$ o.k.

$E \rightarrow y$ o.k.

$E \rightarrow (E + E)$ NO, 12 Trasformo

~~$E \rightarrow$~~ $C \rightarrow ($ o.k.

$P \rightarrow +$ o.k.

$D \rightarrow)$ o.k.

$E \rightarrow C E P E D$

$\underbrace{}_{X \quad Y}$

$X \rightarrow C \bar{E}$ o.k.

$y \rightarrow P \bar{E}$ o.k.

$E \rightarrow X Y D$ no

$\underbrace{}_Z$

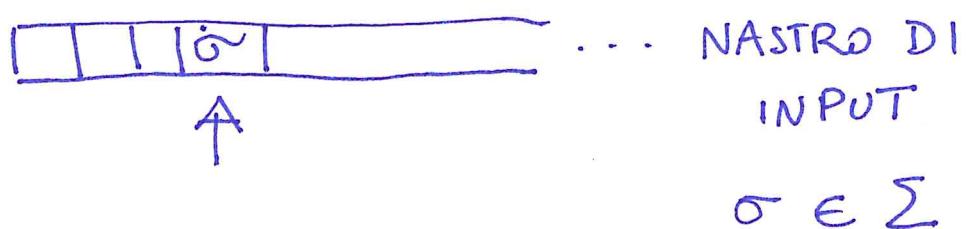
$Z \rightarrow X Y$ o.k.

$E \rightarrow Z D$ o.k.

Riconoscitori a Pila

per i linguaggi di tipo 2

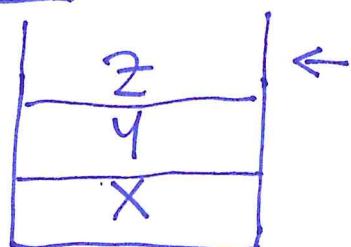
Hardware:



Def di PILA

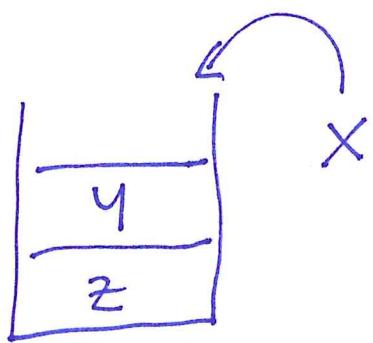
Memoria ad accesso limitato,
politica di accesso: LIFO
(last in first out)

Notazione

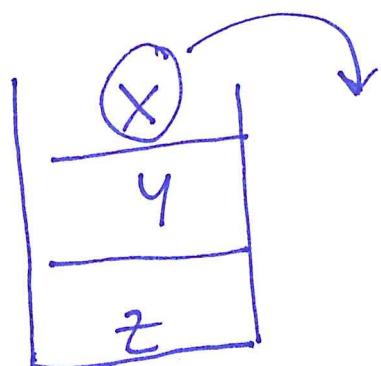


TOP
↓
z y x
↑
FONDO

Operazioni di modifica della pila:



posso inserire X
in cima alla pila
PUSH



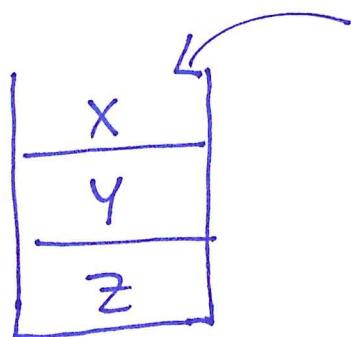
posso estrarre X
dalla cima della pila
POP

Formalizzo:

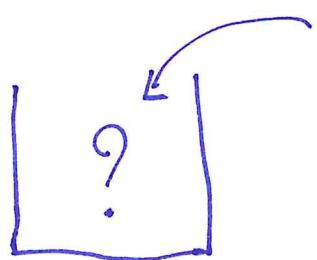
$$\text{PUSH}(P, X) = XP$$

$$\text{POP}(P) = \begin{cases} W & \text{se } P = XW \\ \perp & \text{se } P = \epsilon \end{cases}$$

Operazioni di interrogazione della pila :



posso leggere l'elemento
in cima alla pila : X
TOP



posso chiedere se la
pila è vuota
ISEMPTY

Formalizzo:

$$\text{TOP}(P) = \begin{cases} X & \text{se } P=X \\ \perp & \text{se } P=\emptyset \end{cases}$$

$$\text{ISEMPTY}(P) = \begin{cases} 1 & \text{se } P=\emptyset \\ 0 & \text{se } P \neq \emptyset \end{cases}$$

Def di riconoscitore a pila:

Un riconoscitore a pila è una tupla

$$A = (\Sigma, K, S, \delta) \text{ dove:}$$

- Σ l'alfabeto di input
- K l'alfabeto della pila

$$\Sigma \cap K = \emptyset$$

- S simbolo iniziale della pila
- δ funzione di evoluzione della pila

$$\delta: K \times \Sigma \rightarrow 2^{K^*}$$

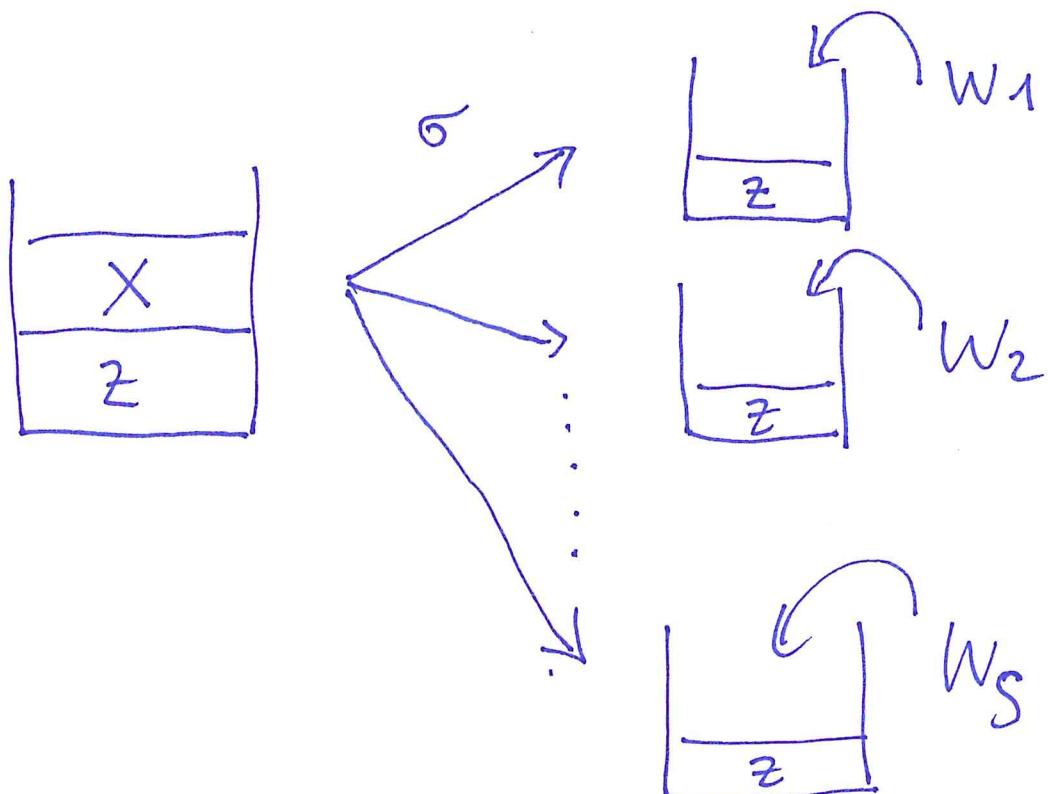
con la scrittura:

$$\delta(x, \sigma) = \{w_1, w_2, \dots, w_s\}$$

$$\text{con } w_i \in K^*$$

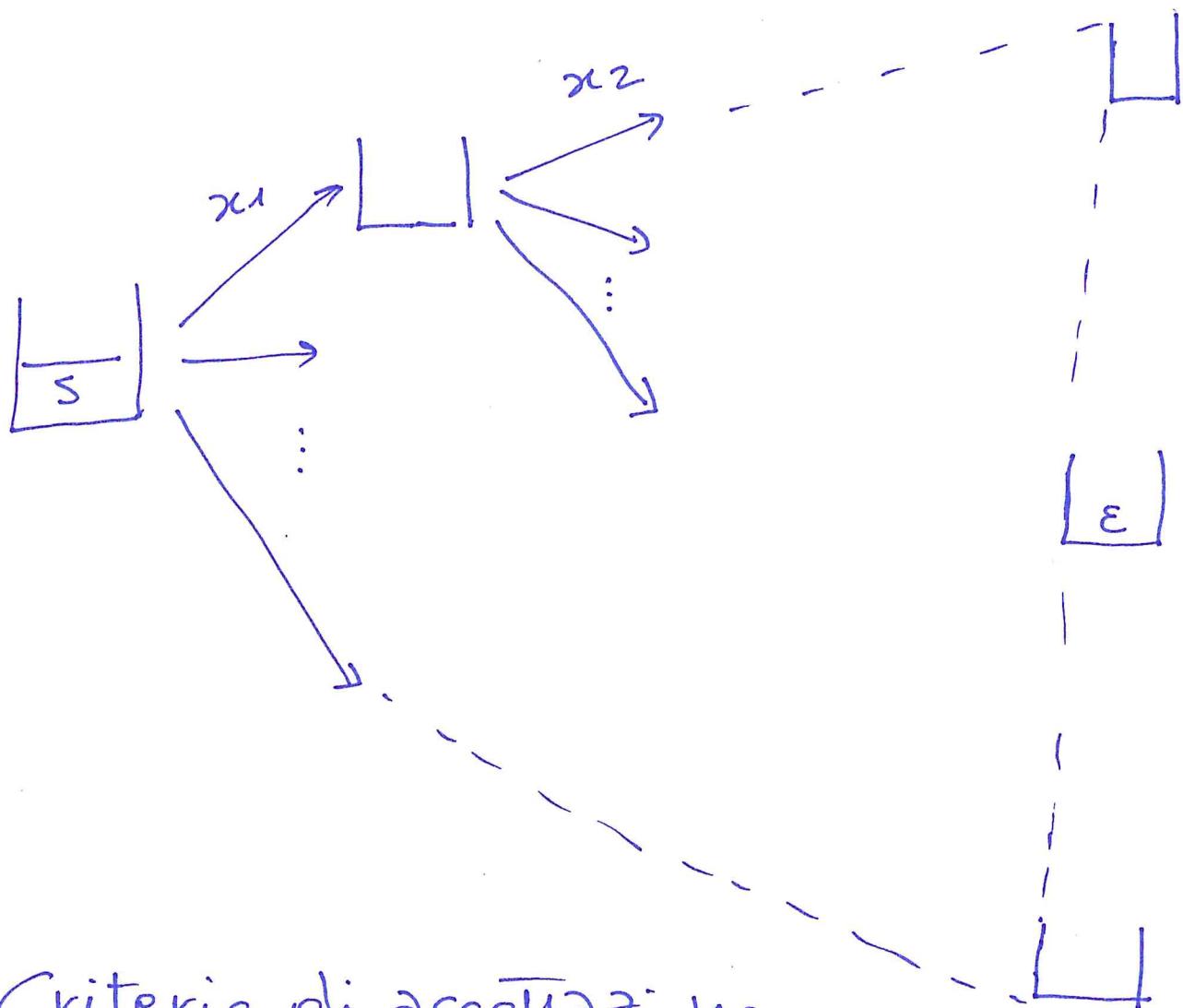
si indica che:

- X è letto in cima alla pila
TOP
- σ è letto sul nastro di input
- X viene cancellato dalla pila
POP
- viene scelto un W_i
NON DETERMINISTICA MENTE
da inserire nella pila
PUSH



Grafo di computazione dovuto a

$x_e = x_1 x_2 \dots x_n \leftarrow \text{INPUT}$



Criterio di accettazione

la parola x_e si dirà accettata
se nel grafo di computazione di x_e
esiste un cammino che partendo
da S mi porta nella pila vuota

ISEMPTY

Formalizzo:

Configurazione

PILA . INPUT
↑
parte di input
ancora da leggere

Esempi

S · x configurazione
iniziale

ε · ε configurazione
finale
accettante

Passo di computazione

configurazione +

configurazione
successiva

Scriveremo:

$$X \cdot w \cdot \sigma w \vdash \gamma w \cdot w$$



$$\gamma \in \delta(X, \sigma)$$

Linguaggio riconosciuto da A

$$L(A) = \left\{ x \in \Sigma^* \mid S \cdot x \xrightarrow{*} E \cdot E \right\}$$

dove

\vdash^* è: zero o più passi
di computazione

Esempio:

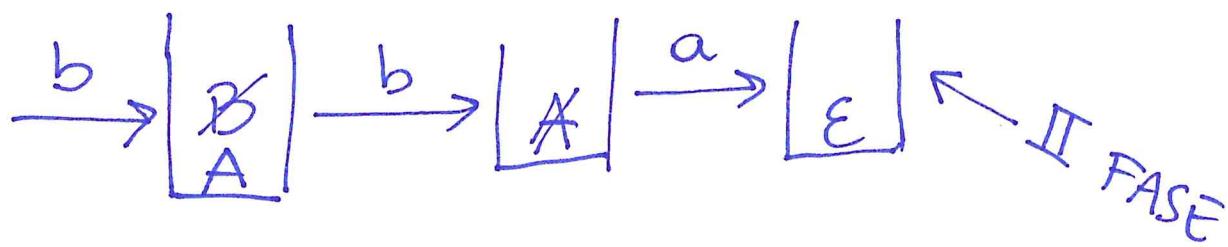
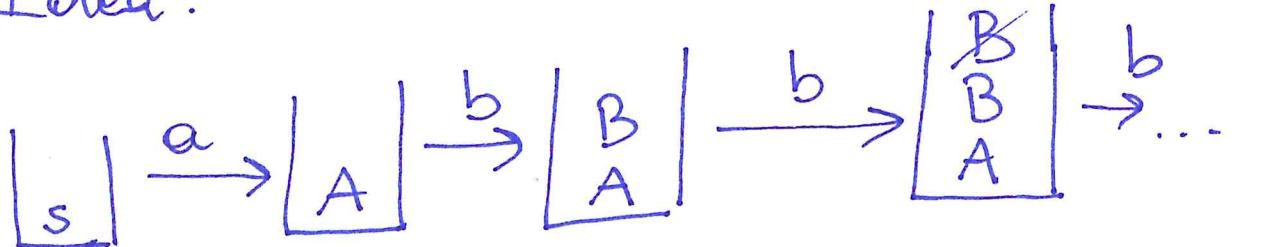
linguaggio: $w w^R \leftarrow$ PALINDROME

$w = w_1 w_2 \dots w_n$

$w^R = w_n w_{n-1} \dots w_1$

es: abb bba

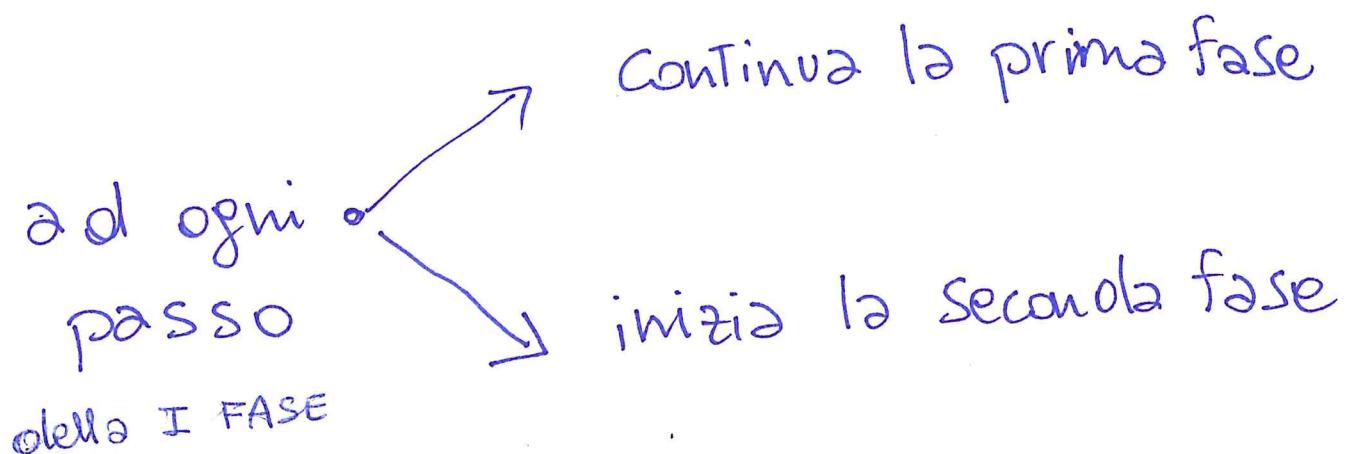
Idea:



Problema:

Come riconoscere il centro
della parola di input?

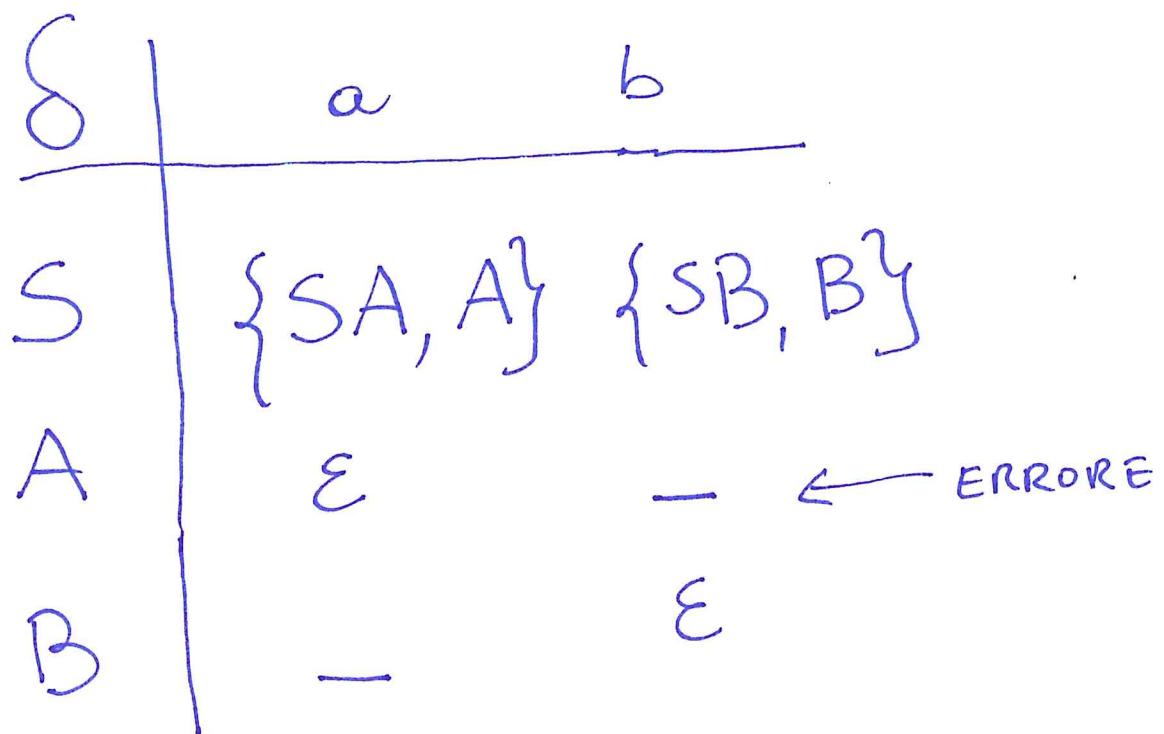
Risposta: Si usa il nondeterminismo



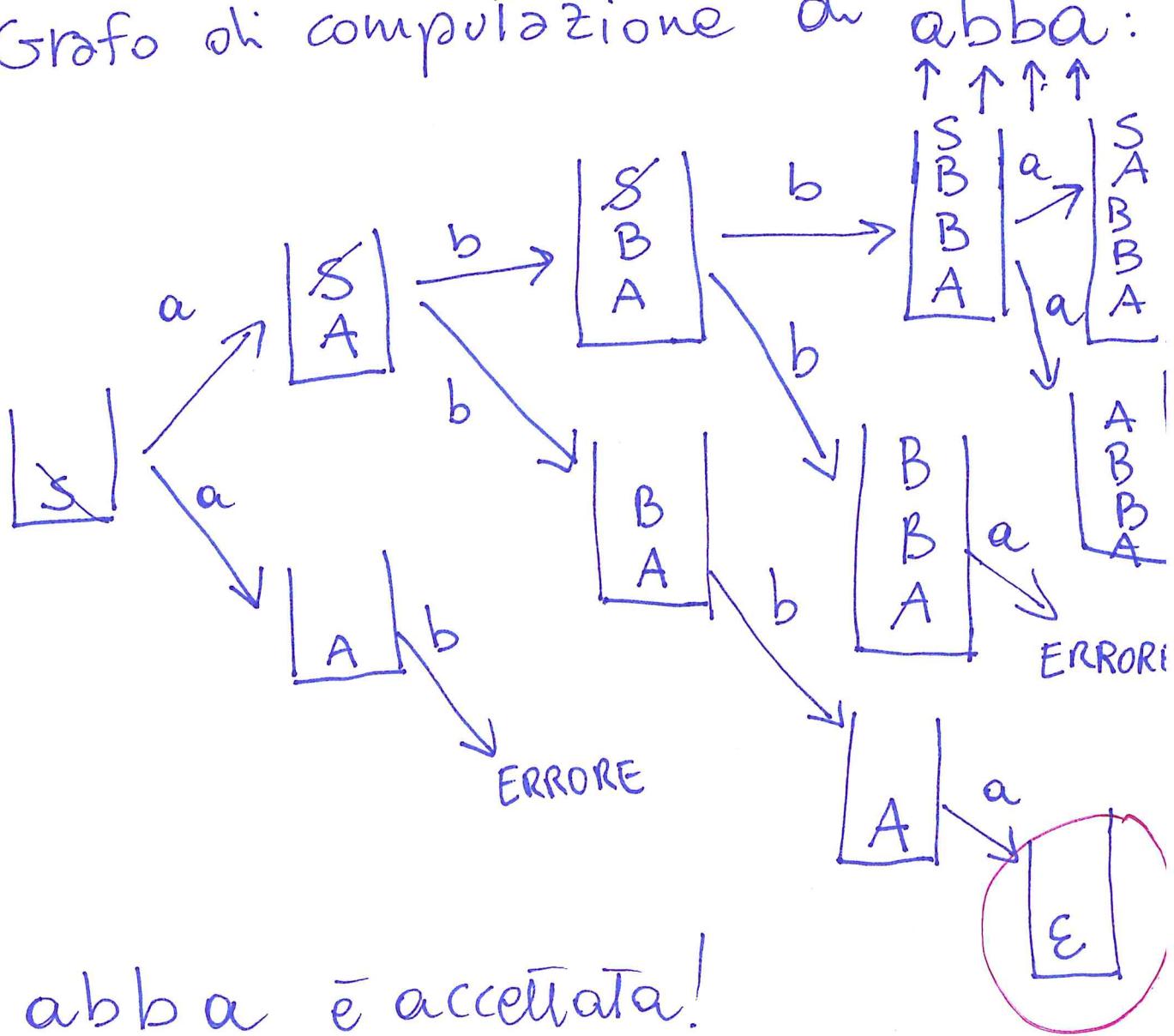
CIMA della PILA

S : siamo nella prima fase
ovvero inserimento dei simboli nella pila

A, B : siamo nella seconda fase
ovvero confronto dei simboli di input con il contenuto della pila



Grafo di computazione di abba:



DOMANDA:

Qual è la classe dei linguaggi riconosciuti dai riconoscitori a pila?

RISPOSTA: I linguaggi di tipo 2.

Attenzione: Il modello dei riconoscitori a pila deve essere NON DETERMINISTICO:

$$S: K \times \Sigma \rightarrow 2^{K^*} \text{ e non } K^*$$

Infatti:

Teorema

L è generato da G di tipo 2



L è riconosciuto da un riconoscitore a pila

dim: Da A r.p. a G di tipo 2.

Sia $A = (\Sigma, K, S_A, \delta)$ posso

costruire G di tipo 2:

$$G = (T = \Sigma$$

$$V = K$$

$$S = S_A$$

P contiene la regola:

$$) \quad X \rightarrow^\sigma W \Leftrightarrow W \in \delta(X, \sigma)$$

Esempio:

$$L = \omega\omega^R \text{ su } \Sigma = \{a, b\}$$

$$A = (\{a, b\}, \{A, B, S\}, S, \delta)$$

dove

S	a	b
S	{SA, A}	{SB, B}
A	ϵ	-
B	-	ϵ

$$G = (\quad T = \{a, b\})$$

$$V = \{A, B, S\}$$

S è l'assiooma

$$\begin{aligned} P = \{ & S \rightarrow aSA, S \rightarrow aA \\ & S \rightarrow bSB, S \rightarrow bB \\ & A \rightarrow a, B \rightarrow b \} \end{aligned})$$

la G ottenuta è la F.N. di Greibach

oh: $S \rightarrow aSa, S \rightarrow aa$

$S \rightarrow bSb, S \rightarrow bb$

ohim: da G di tipo 2 ad A r.p.

Abbiamo G di tipo 2 e per prima cosa
la trasformo in F.N. di Greibach:

$G \rightsquigarrow G' = (\Sigma, V, S_{G'}, P)$. Costruisco

$$A = (\Sigma = \Gamma$$

$$K = V$$

$$S = S_{G'}$$

δ così definita:

$$\delta(X, \sigma) = \{W \mid X \xrightarrow{\sigma} W \in P\}$$

Esempio

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

G per L: $S \rightarrow aSb, S \rightarrow ab$

trasforno G in G' F.N. di Greibach:

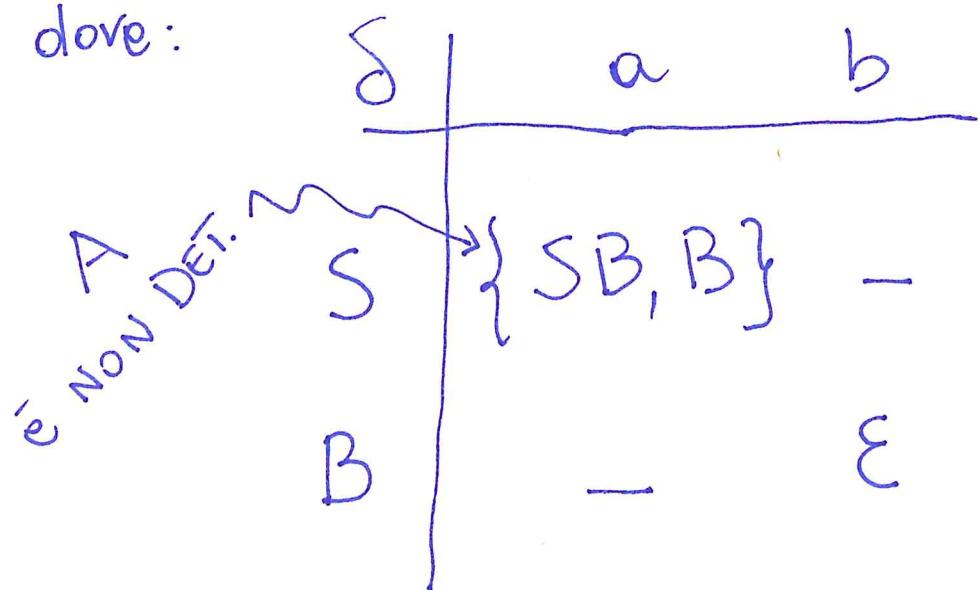
$$\begin{aligned} S \rightarrow aSb &\rightsquigarrow S \rightarrow aSB \\ &\qquad B \rightarrow b \end{aligned}$$

$$S \rightarrow ab \rightsquigarrow S \rightarrow aB$$

definisco

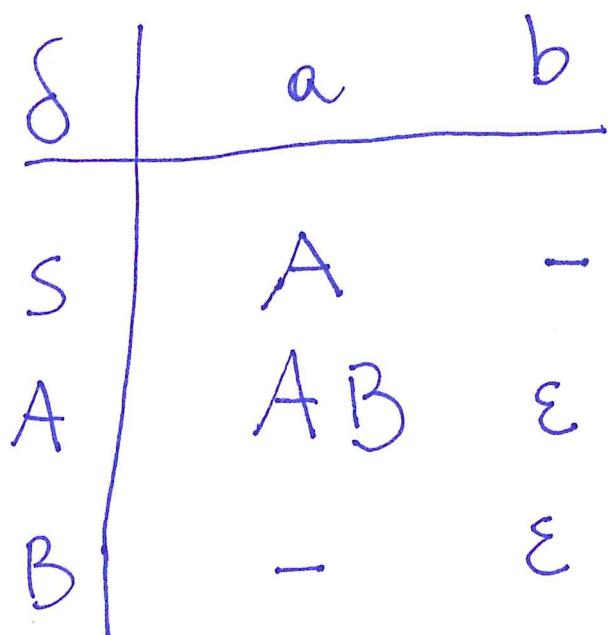
$$A = (\{a, b\}, \{S, B\}, S, \delta)$$

dove:



Attenzione: non è il miglior riconoscitore per $a^n b^m$, infatti ne esiste uno DETERMINISTICO:

$A' = (\{a, b\}, \{\$, A, B\}, S, \delta)$ dove



note:

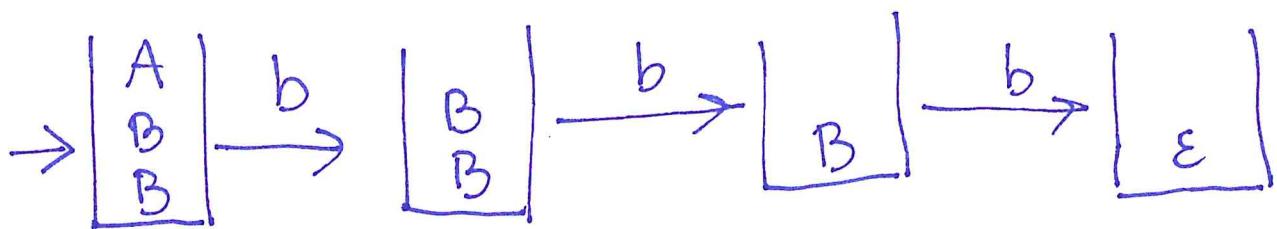
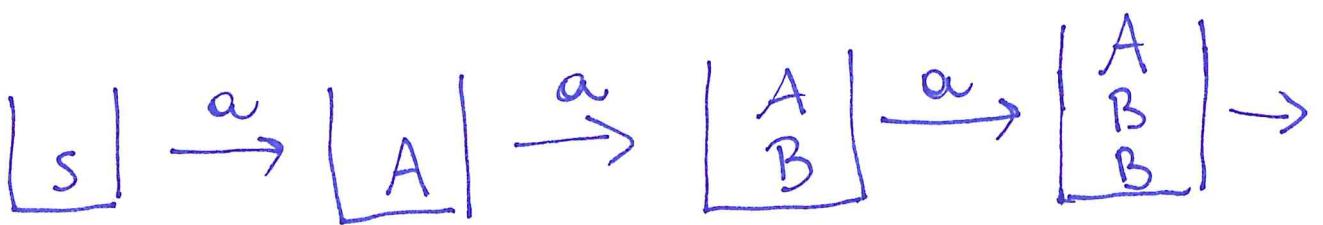
A in cima:

sono nella prima metà e inserisco nella pila una B

B in cima:

solo nella seconda metà e cancello le B della pila

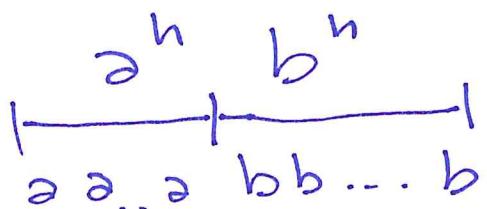
Esempio : aaa bbb



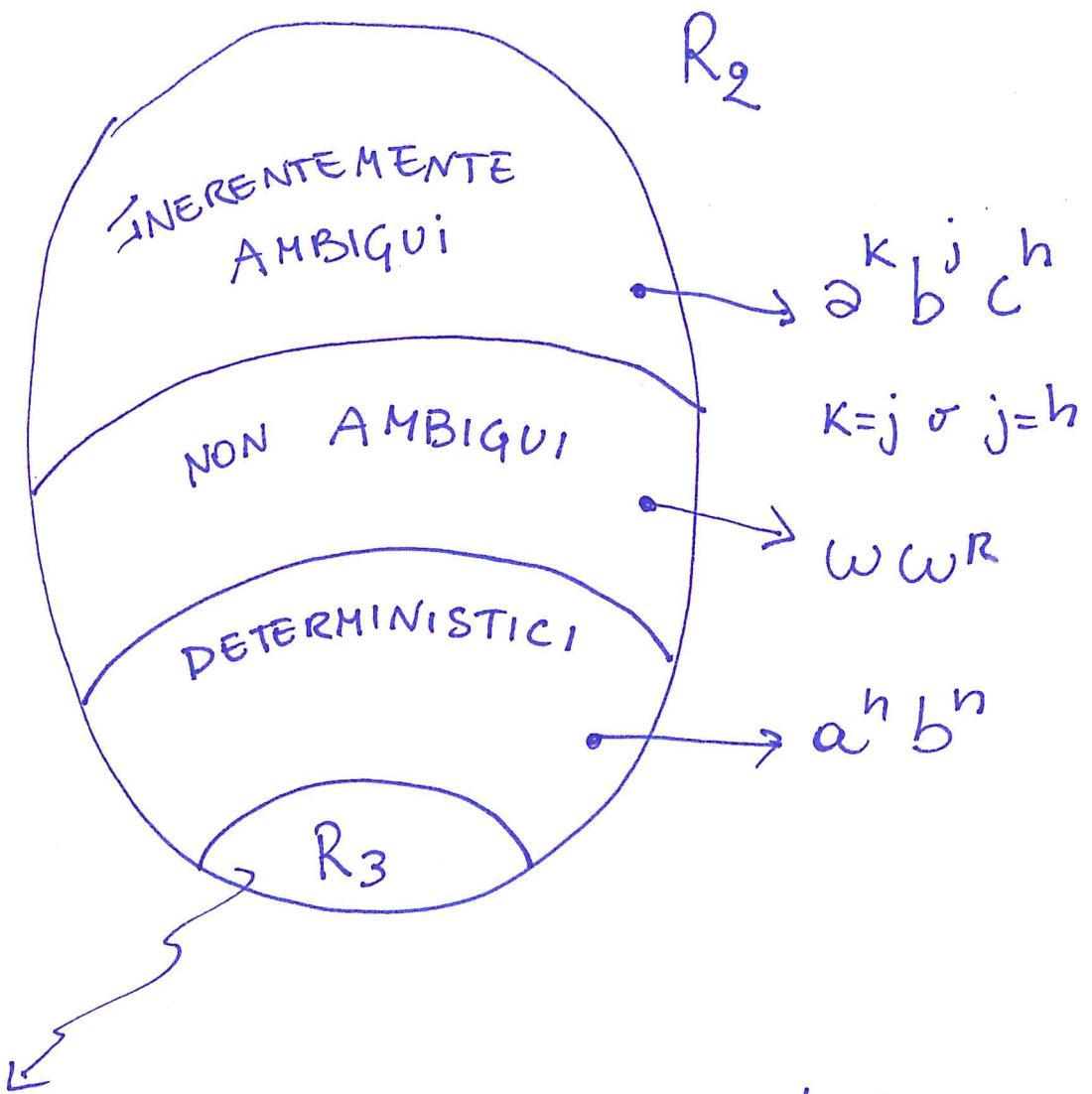
OSSERVAZIONE: senza il simbolo A

aaa bbabb verrebbe accettata!

$\#_a = \#_b$ ma non è nel formato



Riassumendo:



FATTO: Un linguaggio regolare ammette un riconoscitore a pila DETERMINIS.

dim: IDEA

$L \in R_3 \rightarrow$ esiste A DFA \rightarrow simulo A usando la pila:
metto lo stato nella pila *

* "empty" viene sostituito da "is final state"!

Alberi binari

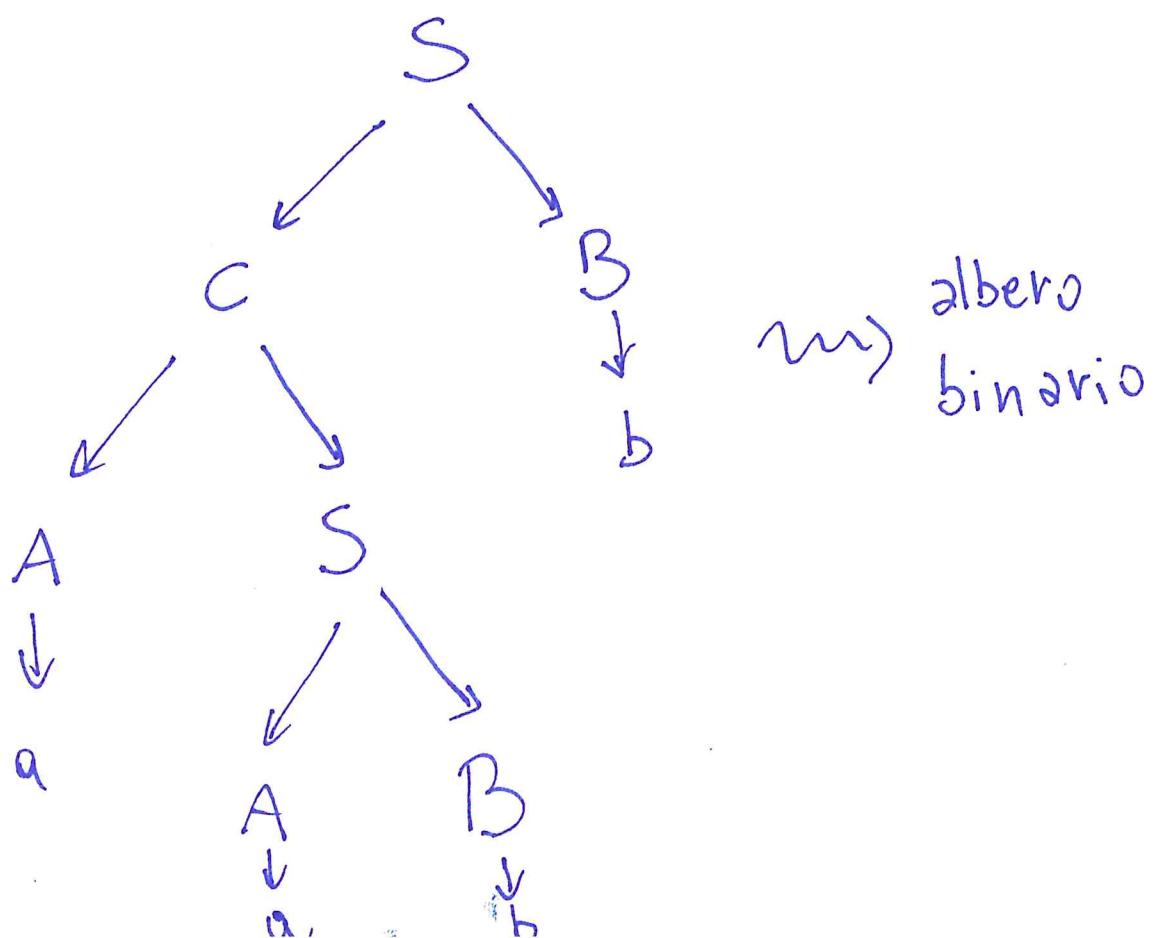
F. N. Chomsky

$$\begin{array}{l} A \rightarrow BC \\ A \rightarrow \sigma \end{array}$$

si hanno alberi di derivazione binari

es: $S \rightarrow AB$ $S \rightarrow CB$ $C \rightarrow AS$

$$\begin{array}{l} A \rightarrow a \\ B \rightarrow b \end{array}$$

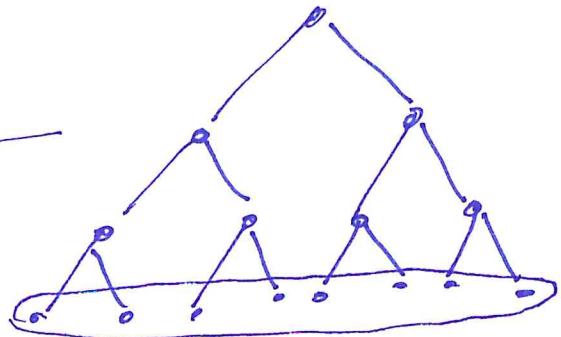


Relazione:

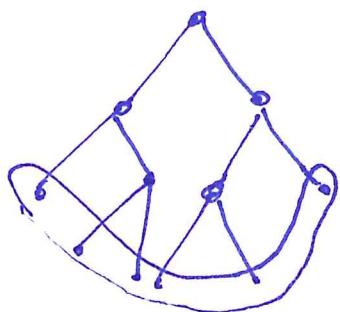
$m =$ numero di foglie

$h =$ altezza

albero
bilanciato



$$m = 2^h$$



$$m \leq 2^h$$

In generale ho:

$$m \leq 2^h \quad \text{da cui:}$$

$$\log m \leq h$$

PUMPING LEMMA

Esprime una condizione necessaria
per i linguaggi di Tipo 2

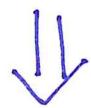
Nota:

- L non soddisfa il lemma



L non è di tipo 2

- L soddisfa il lemma



L può essere di tipo 2

Enunciato:

Per ogni L di tipo 2 esiste una costante $H > 0$ tale che per ogni $z \in L$ con $|z| > H$ esiste una scomposizione in

$$uvwx^ky = z$$

che soddisfa:

$$1) |vwx| > 1$$

$$2) |vwx| \leq H$$

$$3) \forall K \geq 0 \quad u v^K w x^K y \in L$$

dim:

- "Per L di tipo 2 esiste $H > 0$ "

Se L è di tipo 2 allora ammette una

G in F.N. di Chomsky

fisso $H = 2^{h+1}$ ove

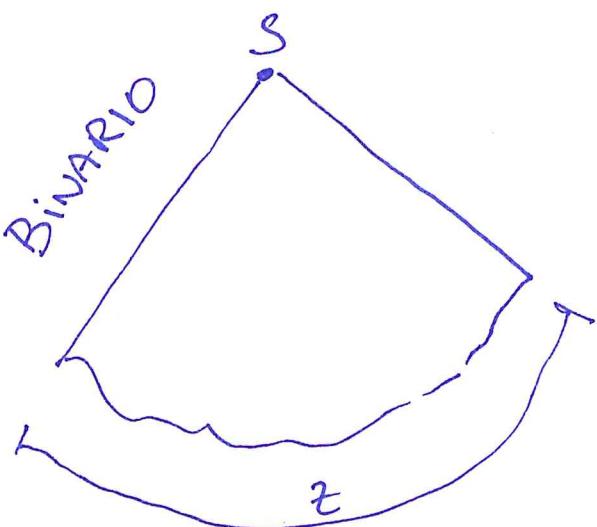
$h = \text{numero di variabili in } G$.

- "per ogni $z \in L$ t.c. $|z| > H$ "



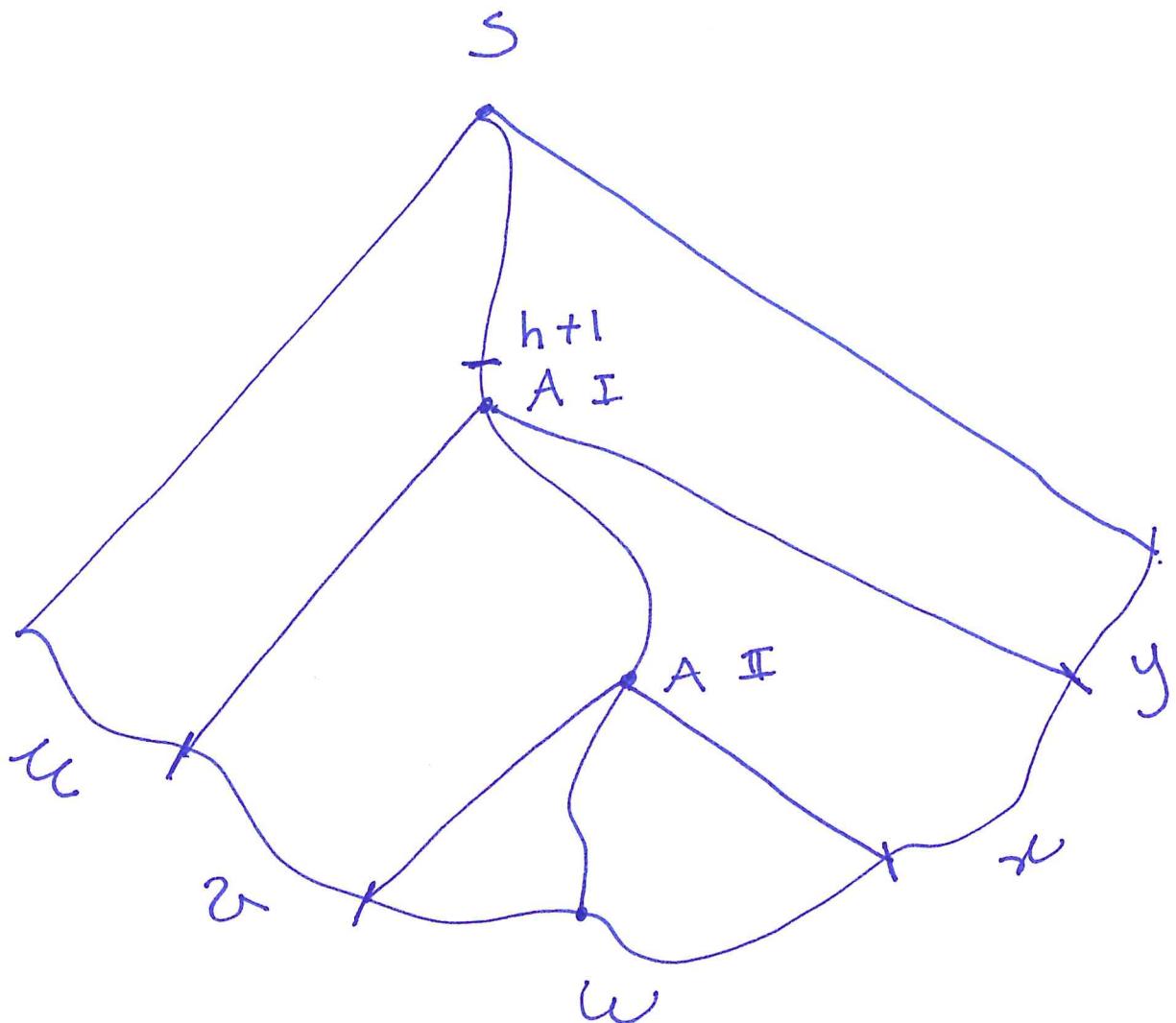
altezza $\geq \log |z| >$

$> \log H = h+1$



- "esiste una scomposizione di z in $u v w x y$ "

Si ricava dall'albero di derivazione per z :



- considero il ramo più lungo dell'albero
- risalgo dal fondo di $h+1$ nodi
- per il principio della piccinaia due nodi sono etichettati con la stessa variabile : A
- la scomposizione si ricava da questi due nodi : A I e A II

- "La scomposizione soddisfa 1) 2) 3)":

condizione 1)

$|vwx| \geq 1$ v e x non possono essere contemporaneamente ε perché A_I e A_{II} sono distinti.

condizione 2)

$$|vwx| \leq H$$

si ha:

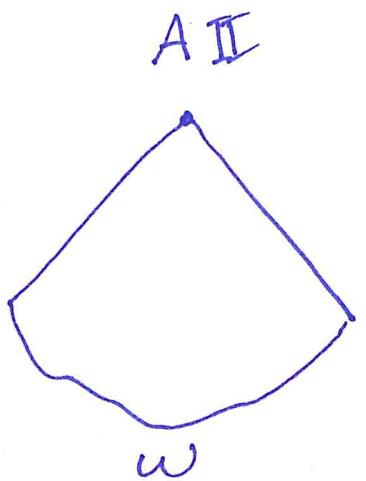
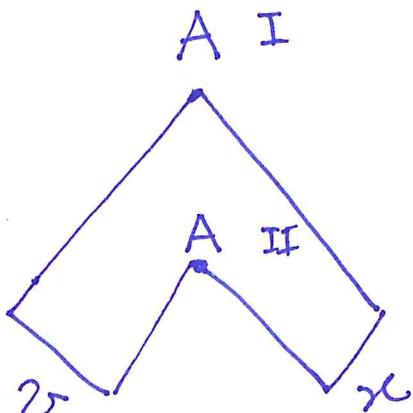
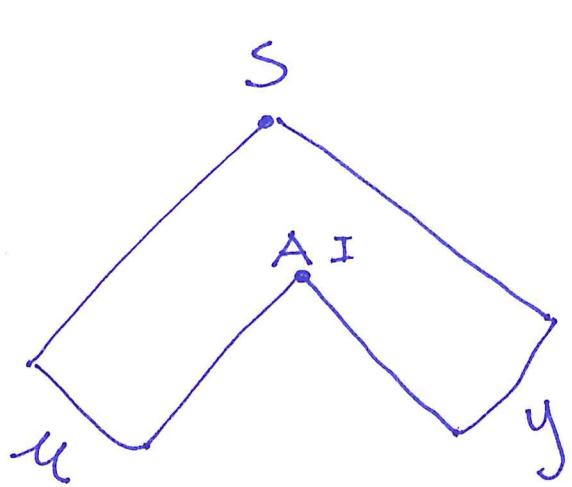
$$|vwx| \leq 2^{h'} \leq 2^{h+1} = H$$

dove h' è l'altezza dell'albero con radice A_I

condizione 3)

$$\forall K > 0 \quad \exists v^K w^K x^k y \in L$$

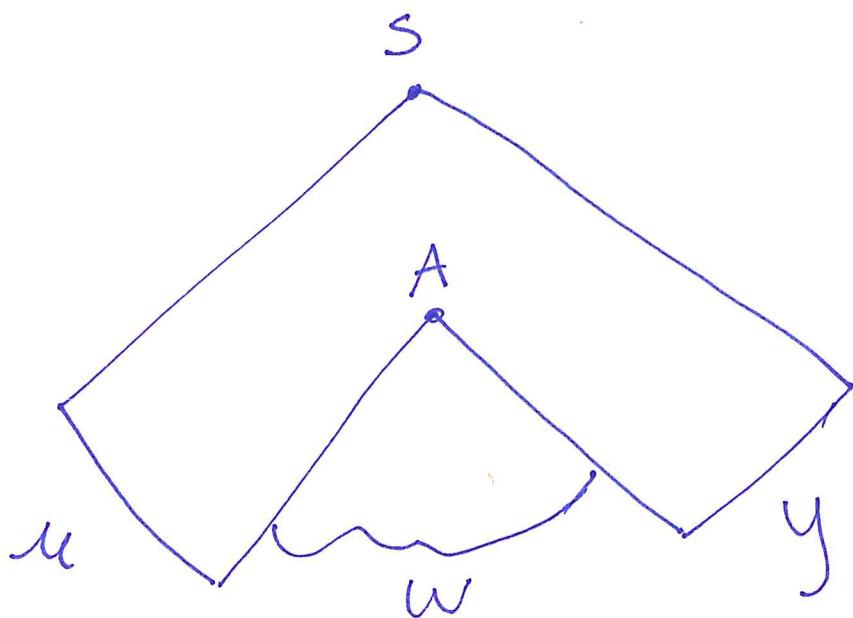
Ritaglio l'albero



$$S \Rightarrow^* u A y \quad A \Rightarrow^* v A x \quad A \Rightarrow^* w$$

[CASO $k=0$]:

dovò mostrare che $u w y \in L$

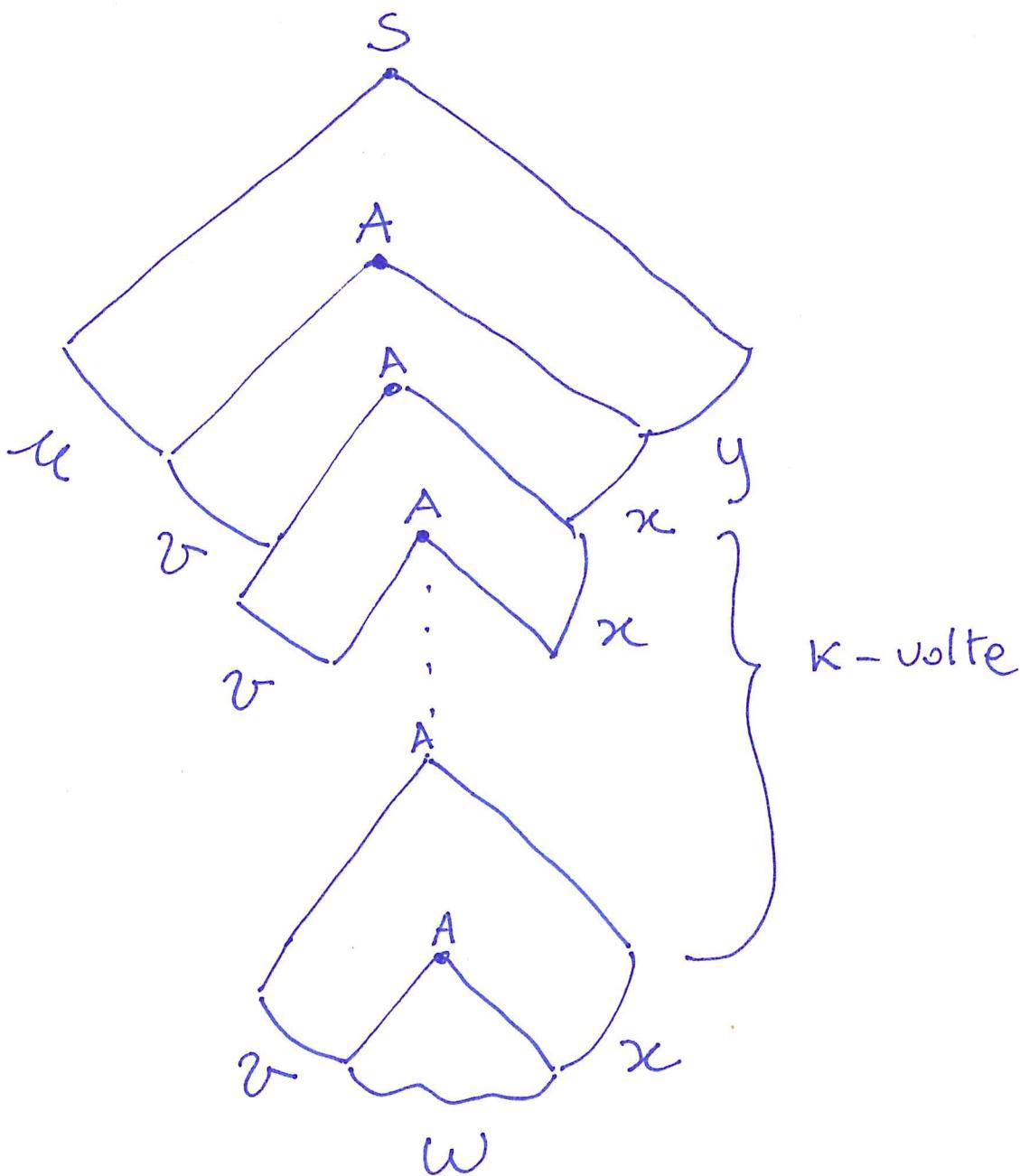


$$S \Rightarrow^* u A y \Rightarrow^* u w y$$

CASO $K > 0$:

dovs mostrare che

$$\underbrace{\mu v v \dots v}_{k\text{-volte}} w \underbrace{xx \dots x y}_{k\text{-volte}} \in L$$



$$S \Rightarrow^* \mu A y \Rightarrow^* \mu v A x y \Rightarrow^* \dots \\ \dots \Rightarrow^* \mu v^K A x^K y \Rightarrow^* \mu v^K w x^K y$$

Applicazione del pumping lemma.

Fatto: $a^h b^n c^n$ non è di tipo 2.

dim: Considero la parola:

$$z = a^H b^H c^H, \quad |z| = 3H > H$$

e faccio vedere che qualunque sua scomposizione in $uvwxy$ non soddisfa contemporaneamente i), ii), iii)

$$z = a a \dots a \overbrace{bb \dots b}^w c c \dots c$$

$\xrightarrow{u \quad v \quad w \quad x \quad y}$

Per vwx si hanno i seguenti casi:

- i. vwx contiene sia a, b, c
- ii. oppure contiene $\circlearrowleft a, b \circlearrowright$ o b, c
- iii. oppure contiene $\circlearrowleft a \circlearrowright b \circlearrowleft c$

Studiamo i casi:

i.

$$vwxe = \overbrace{abb\dots b}^H c$$



$$|vwxe| > H \quad \text{la condizione}$$

2) è falsa

ii.

Ricavo che $vwxe$ non contiene le "c"

Considera:

$$\mu v^K w x^K y \in L$$

(cioè supponi che vale la condizione 3))

Allora

$$\mu v w x y = \overbrace{abb}^H c^H$$

$$|\mu v^K w x^K y| = 3H \stackrel{\downarrow}{=} |\mu v w x y|$$



perché il numero
di "c" è H

dai cui si ottiene che non vale

la condizione 1)

cioè vera solo se
 $v = x = \epsilon$