Es Costruire un AP equivalente alla grammatica

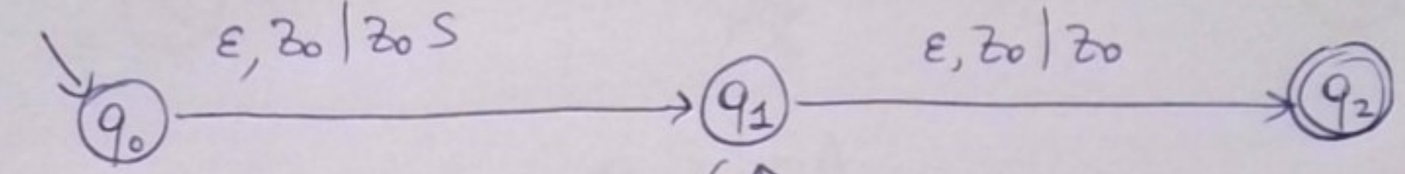
$G = (V_N = \{S, A, B\}, V_T = \{a, b, c\}, P, S)$ con

P:
$$S \longrightarrow aAB$$
$$A \longrightarrow aAA \mid bAB \mid c$$
$$B \longrightarrow bBB \mid aBA \mid c$$

È possibile trovare un AP A t.c. $\mathcal{L}(A) = \mathcal{L}(G)$?

Sì, G è una grammatica context-free (monotona) quindi esiste un APND equivalente.

C'è un modo semplice per trovare un APND equivalente



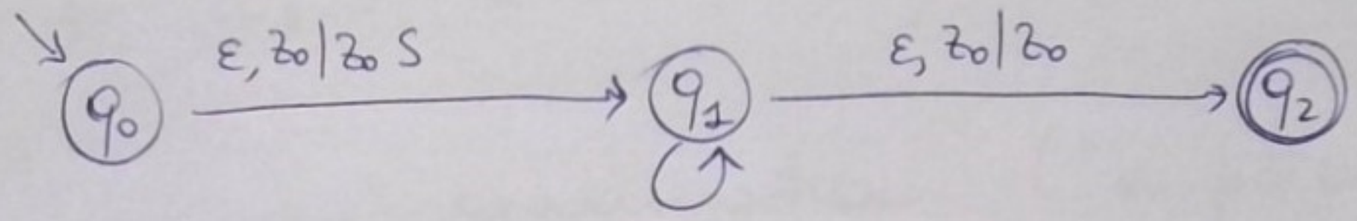
non-determinismo →

ε, S | aBAa	ε, A | c
ε, A | bBAb	a, a | ε
ε, B | bBBb	b, b | ε
ε, B | aABa	c, c | ε

queste transizioni simulano le produzioni di G.

non è deterministico, però in questo caso particolare possiamo osservare che nelle produzioni c'è sempre un solo terminale a dx e questo potrebbe eliminare il non determinismo.

L'automa è equivalente a questo



a, S | BA
a, A | AA
a, B | ~~AB~~ ← "fondo" la transizione ε, B | ABa con a, a | ε
b, A | BA
b, B | BB
c, A | ε
c, B | ε

↑

casualmente è diventato un APD ⇒,

NON capita sempre

Es Sia $L_1 = \{ a^n b^{2m} \mid n, m \geq 1 \}$

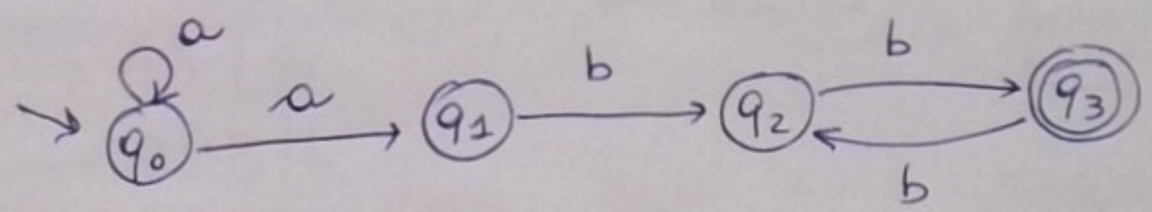
e $L_2 = \{ a^n b^{3n} \mid n \geq 0 \}$.

Trovare una grammatica ("a potenza minima")
che generi L_1, una per L_2 e una
per $L = L_1 \cap L_2$. Compito: trovate voi ASF/AP/MT.

L_1: è regolare, $L_1 = a^+(bb)^+$

Basta una gr. regolare e un ASF

Siamo abituati all'ASF:



non serve sia det.! possibilmente
anzi, così avrò meno stati e la grammatica viene più semplice

È facile trasformare l'automa in grammatica

$V_N = \{q_0, q_1, q_2, q_3\}$, assiamo q_0

$V_T = \{a, b\}$

P: $q_0 \longrightarrow a q_0 \mid a q_1$

$q_1 \longrightarrow b q_2$

$q_2 \longrightarrow b q_3 \mid b$

$q_3 \longrightarrow b q_2$

Compito: determinizzare l'automa (è molto facile, non c'è bisogno dell'algoritmo generale)

Una grammatica che genera L_1 è

$S \longrightarrow ABB$

$A \longrightarrow AA \mid a$

$B \longrightarrow BBB \mid b$

o anche

$S \longrightarrow AS \mid BB$

$A \longrightarrow a$

$B \longrightarrow b$

ma non sono regolari

L_2: non è regolare, è riconosciuto da un APD;

però il fatto che basti un AP deterministico

non influenza il fatto che L_2 sia context-free.

Una grammatica CF per L_2 è

$$S \longrightarrow aSbbb \mid \varepsilon$$

In questo caso è più semplice la grammatica dell'automa.

$L = L_1 \cap L_2$ può essere regolare o context-free*.

~~(i ling. c.f. non sono chiusi per intersezione)~~ Studiamolo meglio.

Le stringhe di L stanno in L_1 e in L_2,

sono del tipo $a^* b^*$ con

. il numero delle 'b' che è almeno 2, pari (sta in L_1)

e multiplo di 3, quindi b^{6n} con $n > 0$. (sta in L_2)

. il numero delle 'a' è almeno 1 ed è un terzo delle 'b' (sta in L_1, sta in L_2)

$L_2 = \{ a^{n} b^{3n} \mid n \geq 0 \}$

quindi $L = \{ a^{2n} b^{6n} \mid n > 0 \}$, che è context-free.

(basta un APD)

* NB: l'intersezione tra un linguaggio CF e uno regolare è sempre al più context-free

Una grammatica per L è

$$S \longrightarrow aa\,S\,bbbbbb \mid aabbbbbb$$

Es Trovare una grammatica ("a potenza minima") che generi

$$L = \{ a^n (bc)^m \mid n, m \geq 1,\ m < \tfrac{n}{2} \}$$

↙ provate a usare il Pumping Lemma

L non è regolare, L è riconosciuto da APD: per ogni due 'a' che leggo faccio un segno in pila, poi quando non ho più 'a', per ogni segno in pila leggo 'bc'; provate a farlo.

Come è fatto L?

$L = \{aaabc, aaaaabc, a^5bc, a^5(bc)^2, \dots\}$

$n=1$: $1 \le m < \frac{1}{2}$ imp.

$n=2$: $1 \le m < 1$ imp

$n=3$: $1 \le m < \frac{3}{2} \to m = 1$

$n=4$: $1 \le m < 2 \to m = 1$

$n=5$: $1 \le m < \frac{5}{2} \to m \in \{1, 2\}$

...

L è "strano", ma la grammatica no.

$S \longrightarrow ASB \mid aaabc \mid aaaaabc$

è C-F.

$A \longrightarrow aa$

$B \longrightarrow bc \mid \varepsilon$

Un'altra soluzione è

$S \longrightarrow aaabc \mid aaaXbc$

$X \longrightarrow aaXbc \mid aabc \mid a$ così no, manca a^5bc, ...

$S \longrightarrow aaaXbc$ così sì

$X \longrightarrow aaXbc \mid aX \mid \varepsilon$

Es Trovare una grammatica che generi

$$L = \{ a^n b^{3n} \mid n \geq 0 \} \cup \{ a^n (bbbc)^n \mid n \geq 0 \}$$

Si descriva un macchina a potenza minima che riconosca L.

La grammatica è facile: ne basta una CF.
(e serve CF visto che L non è regolare)

$S \longrightarrow X \mid Y$

$X \longrightarrow aXbbb \mid \varepsilon$

$Y \longrightarrow aYbbbc \mid \varepsilon$

è CF.

L'automa è più difficile: un APND si può fare di sicuro, ma ne basta uno deterministico

- Inizialmente, per ogni 'a' letta faccio un segno in pila.
- Finite le 'a', ci saranno le 'b'. Leggo le prime 3 e non faccio nulla in pila.

Se il quarto carattere è 'c', so che devo accettare stringhe del tipo $a^n(bbbc)^n$: faccio la pop del simbolo in pila e proseguo su una parte di automa adatta a stringhe di quel tipo.

Se c'è 'b', faccio la pop e proseguo riconoscendo stringhe del tipo $a^n b^{3n}$.

Es Trovare una grammatica (a potenza minima) per

$$L = \{ a^n w \mid w \in \{b,c\}^*, \#_b(w) = \#_c(w) = n \geq 1 \}$$

$$L' = \{ a^n w \mid w \in \{b,c\}^*, \#_b(w) + \#_c(w) = n \geq 1 \}$$

$$L'' = \{ a^n w \mid w \in \{b,c\}^*, \#_b(w) + \#_c(w) = 2n, n \geq 1 \}$$

L' e L'' sono linguaggi più semplici di L, per loro basta un APD: per ogni 'a' faccio uno (per L') o due (per L'') segni in pila, finite le 'a', per ogni 'b' o 'c' tolgo un segno dalla pila finché non trovo Z_0, poi mi fermo.

Per L' e L'' bastano grammatiche context-free

Per L': $S \to aSb \mid aSc \mid ab \mid ac$

ma va bene anche: $S \to aSX \mid aX$; $X \to b \mid c$

Per L'', ispirandosi alla seconda soluzione per L':

$S \to aSXX \mid aXX$; $X \to b \mid c$

L è un linguaggio più complesso, di sicuro non è CF (infatti, sapendo che l'intersezione tra un CF e un regolare è ancora CF, osserviamo che se L fosse CF, allora anche $L \cap a^*b^*c^* = \{a^n b^n c^n \mid n \geq 1\}$ dovrebbe essere CF, cosa che invece non è).

È facile osservare che L è riconosciuto da un MT a nastro singolo che usa solo le celle occupate dall'input, o da una MT a un nastro che occupa una porzione del

nastro di memoria non superiore alla lunghezza dell'input (basta scorrere l'input, fare un segno per ogni 'a', controllare che le 'b' e le 'c' siano tante quante le 'a' e che dopo una 'b' o una 'c' non ci sia 'a').

Allora L è context-sensitive e basta una grammatica di tipo 1, o monotona.

$S \to aSBC \mid aBC$

$BC \to CB$ | c'è bisogno di entrambe

$CB \to BC$ |

$B \to b$

$C \to c$

Es Scrivere un grammatica per

$$L = \{ a^n b^n c^n \mid n \geq 1 \}$$

$S \longrightarrow aSBC \mid aBC$ ← così il numero di 'a', 'b' e 'c' è bilanciato, e le 'a' sono al posto giusto

così le 'C' posso superare le 'B' → $CB \longrightarrow BC$

$@B \longrightarrow ab$

$bB \longrightarrow bb$

$bC \longrightarrow bc$

$cC \longrightarrow cc$

'B' diventa 'b' solo se è preceduta da 'a' oppure 'b', così le 'b' sono tutte consecutive

le 'c' dopo le 'b' e tutte consecutive

problema: NON sono produzioni lecite poiché a sx si possono avere solo non terminali

Idea: introduco nuovi non-terminali

$S \to ASBC \mid ABC$

$CB \to BC$

$A \to a$

$AB \to aB'$

$B'B \to bB'$

$B'C \to bC'$

$C'C \to cC'$

$C' \to c$

esempio: per generare $a^3b^3c^3$

$S \to^3 A^3BCBCBC \to^2 A^3BBCCBC$

$\to A^3BBCBCC \to A^3B^3C^3 \to^2 aaAB^3C^3$

$\to a^3B'BBCCC \to a^3bB'BCCC \to a^3bbB'CCC$

$\to a^3b^3C'CC \to a^3b^3cC'C \to a^3b^3ccC'$

$\to a^3b^3c^3$.

Bonus: l'intersezione tra un CF e un regolare è CF.

Sia L ling. CF e R un ling. regolare:
sia P un AP che riconosce L e A un ASF che riconosce R.
Si può costruire un AP P' che riconosce $L \cap R$ così:

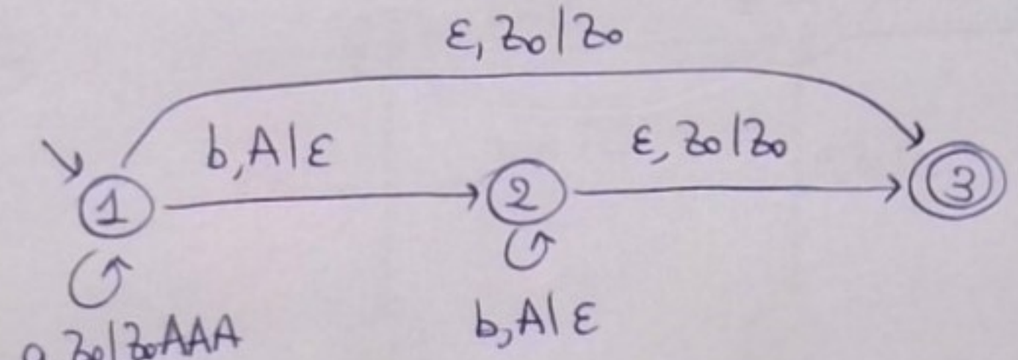
- la pila di P' funziona come quella di P
- gli stati di P' sono coppie formate da uno stato di P e uno di A
- le transizioni di P' simulano in parallelo quelle di P ed A

. lo stato iniziale di P' è formato dalla coppia degli stati iniziali di P ed A

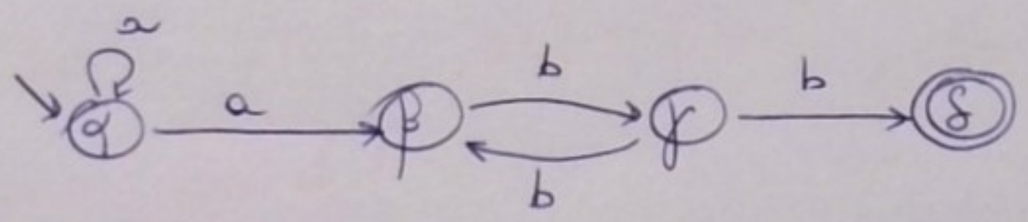
. idem per gli stati finali

esempio: $L = \{ a^n b^{3n} \mid n \geq 0 \}$, $R = a^+ (bb)^+$

P:



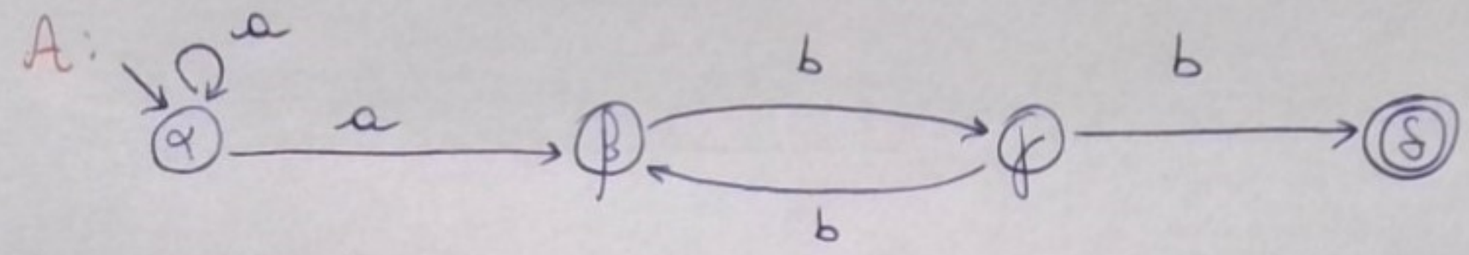
A:



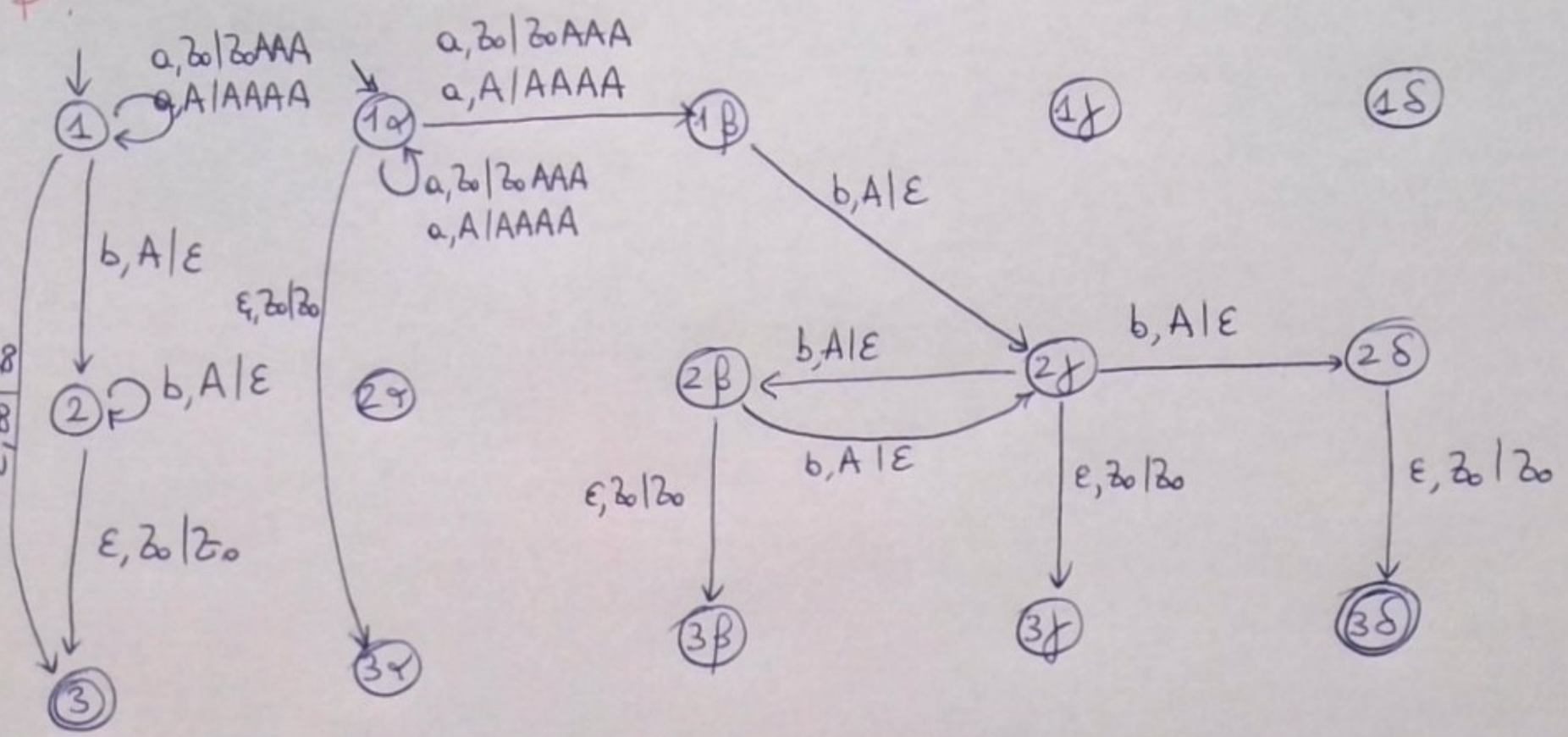
da questo il non-det. si elimina facilmente...

li prendo non-det così hanno meno stati: il numero di stati di P' è il prodotto tra il numero di quelli di P e di A.

A:



P:



ci sono stati superflui, poiché A e P sono non deterministici, anche P' è non-det.