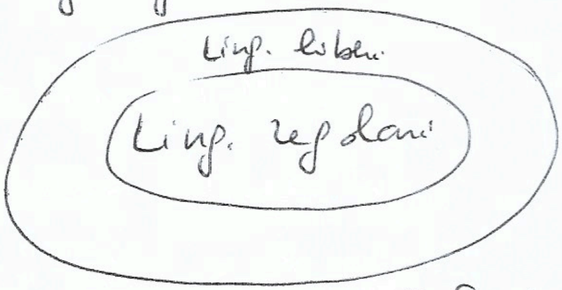
Capitolo 4

Analisi Sintattica: Linguaggi Liberi

- classe di linguaggi più generale di quella dei ling. regolari



- grammatiche libere da contesto più generali di quelle regolari

Regolari		Libere	
$V \to aW$ $V \to a$ $S \to \varepsilon$	$V, W \in NT$ $a \in T$	$V \to \alpha$	$\alpha \in (T \cup NT)^*$

- automi a pila (pushdown automata - PDA) invece di automi finiti (NFA - DFA)

PDA
- nondeterministici $\approx$ Ling. liberi
- deterministici : utili per costruire compilatori
  - top-down - grammatiche LL(K)
  - bottom-up - grammatiche LR(K)

Analisi Sintattica

lista di token (prodotta dall'analizzatore lessicale)

$\downarrow$

prodotto a partire da una grammatica libera $\rightsquigarrow$ **Analizzatore Sintattico PARSER**

$\downarrow$

albero di derivazione
(o anche albero sintattico)
relativo alla grammatica
usata per costruire il parser

Ripasso di definizioni già date qualche lezione fa

Una grammatica libera $G = (NT, T, R, S)$ è tale che
- NT è un insieme finito di simboli nonterminali
- T è un insieme finito di simboli terminali
- $S \in NT$ è il simbolo iniziale
- R è un insieme finito di produzioni (o regole) del tipo

$$V \to \alpha \quad \text{con } \alpha \in (T \cup NT)^* \text{ e } V \in NT$$

$$L(G) = \{ w \in T^* \mid S \Rightarrow^* w \}$$

<u>N.B.</u> Se G è libera, allora $L(G)$ è libero!

La relazione $\Rightarrow$ di derivazione in un passo

$$\frac{v = xAy \qquad (A \to z) \in R \qquad w = xzy}{v \Rightarrow w} \qquad x, y \in (T \cup NT)^*$$

La relazione $\Rightarrow^*$ di derivazione in 0 o più passi

$$\frac{}{v \Rightarrow^* v} \qquad \frac{v \Rightarrow^* w \quad w \Rightarrow z}{v \Rightarrow^* z}$$

(chiusura riflessiva e transitiva di $\Rightarrow$)

Def: <u>Derivazione canonica sinistra (leftmost)</u>:
derivazione in cui, a partire da S, viene <u>sempre</u> riscritto il nonterminale più a <u>sx</u>

Def: <u>Derivazione canonica destra (rightmost)</u>:
derivazione in cui, a partire da S, viene <u>sempre</u> riscritto il nonterminale più a <u>dx</u>

<u>Oss:</u> Esiste una corrispondenza biunivoca tra derivazioni leftmost (/ rightmost) e alberi di derivazione

Come bisogna potenziare gli NFA/DFA per far sì che riconoscano ling. liberi?

Esempio: $L = \{ ww^R \mid w \in \{a,b\}^* \}$

- L è libero perché $L = L(G)$ con $G \left[S \to \varepsilon \mid aSa \mid bSb \right]$ libera.
- L non è regolare (esempio discusso in precedenza)
- $L = \{ \varepsilon, aa, bb, aaaa, abba, baab, bbbb, \dots \}$

- L'automa deve "ricordare" la prima metà dell'input (cioè w) in modo da confrontarla con la seconda metà (cioè w^R) in ordine inverso

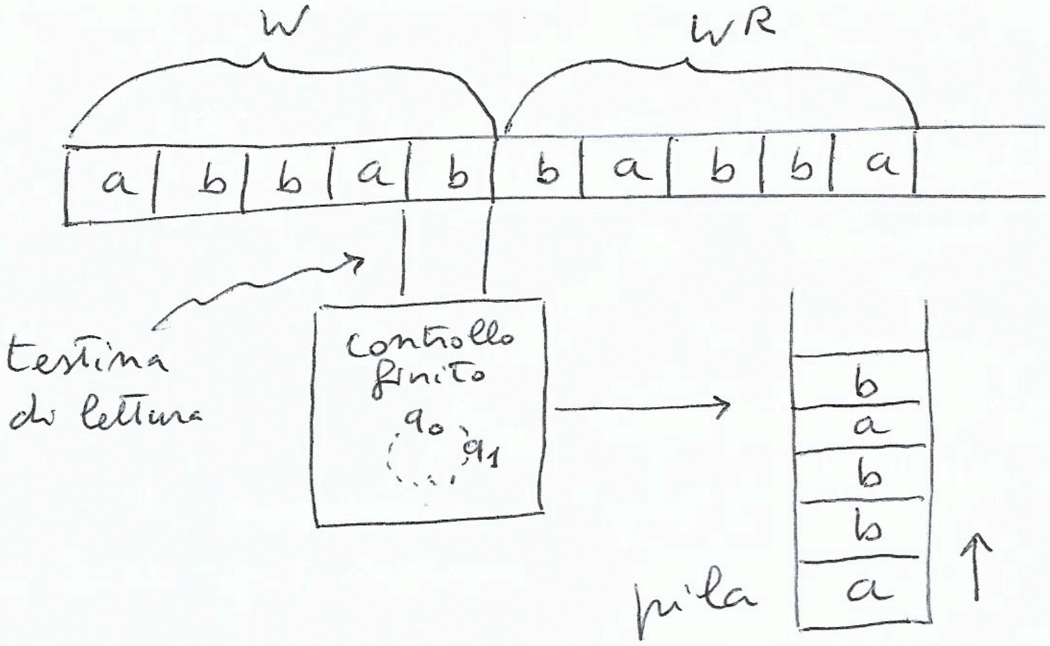
$\Downarrow$

necessità di accumulare (parte dell')l'input!

$\Downarrow$

fornire all'automa una memoria ausiliaria: una pila che può crescere illimitatamente

- Quando si accorge che w è finita?
  In genere, non c'è nessun criterio: nondeterministicamente l'automa assume di essere arrivato al centro/metà ed inizia da lì in poi a confrontare all'indietro!



- la pila può crescere senza limiti (memoria ausiliaria illimitata, perché w può essere arbitrariamente lunga)
- Ma può leggere solo l'elemento top della pila
- Può rimuovere solo l'elemento top della pila
- Può inserire un nuovo elemento solo in testa

<u>Def</u>: Un automa a pila nondeterministico (PDA) è una 7-pla $(\Sigma, Q, \Gamma, \delta, q_0, \perp, F)$ dove
- Σ è un alfabeto finito (simboli in input)
- Q è un insieme finito di stati
- Γ è un insieme finito di simboli della pila
- δ è la funzione di transizione con tipo

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \longrightarrow \mathcal{P}_{fin}(Q \times \Gamma^*)$$

- q_0 è lo stato iniziale
- $\perp \in \Gamma$ è il simbolo iniziale sulla pila
- $F \subseteq Q$ è l'insieme degli stati finali

$$\delta : Q \times \underbrace{(\Sigma \cup \{\varepsilon\})} \times \Gamma \longrightarrow \mathcal{P}_{fin}(Q \times \Gamma^*)$$

consuma
un simbolo dell'input (Σ)
oppure no (ε)

consuma il
simbolo top della pila

sulla pila scrive una
stringa di lunghezza qualsiasi (anche ε)
di simboli in Γ (detto "gamma")

- è <u>nondeterministico</u> perché

(1) $|\delta(q, \sigma, A)|$ può essere > 1

($\text{per } q \in Q,\ \sigma \in \Sigma \cup \{\varepsilon\},\ A \in \Gamma$)

(2) ma se anche richiedessimo che

$|\delta(q, \sigma, A)| \leq 1 \quad \forall q \in Q, \forall \sigma \in \Sigma \cup \{\varepsilon\}, \forall A \in \Gamma$

potrebbe esistere $q' \in Q$, $b \in \Sigma$, $B \in \Gamma$ tali che

$|\delta(q', \varepsilon, B)| = 1 \quad$ e $\quad |\delta(q', b, B)| = 1$

$\Rightarrow$ nondeterminismo tra mossa "ε"
e mossa "b"

Transizioni di un PDA

- descrizione istantanea (o configurazione)

(q, w, β)

- $q \in Q$ (stato corrente)
- $w \in \Sigma^*$ (input ancora non letto)
- $\beta \in \Gamma^*$ (stringa sulla pila) (per convenzione il top è il simbolo più a sinistra)

- mossa

(1)
$$\frac{(q', \alpha) \in \delta(q, a, X) \quad a \in \Sigma}{(q, aw, X\beta) \vdash_N (q', w, \alpha\beta)}$$

(2)
$$\frac{(q', \alpha) \in \delta(q, \varepsilon, X)}{(q, w, X\beta) \vdash_N (q', w, \alpha\beta)}$$

- computazione / cammino

$$\frac{}{(q, w, \beta) \vdash_N^* (q, w, \beta)} \qquad \frac{(q, w, \beta) \vdash_N^* (q', w', \beta') \vdash_N (q'', w'', \beta'')}{(q, w, \beta) \vdash_N^* (q'', w'', \beta'')}$$

(chiusura riflessiva e transitiva di $\vdash_N$)

Linguaggio Accettato

Due modalità di riconoscimento:

(1) Per stato finale

$$L[N] = \{ w \in \Sigma^* \mid (q_0, w, \perp) \vdash_N^* (q, \varepsilon, \alpha) \text{ con } q \in F \}$$

(2) Per pila vuota

$$P[N] = \{ w \in \Sigma^* \mid (q_0, w, \perp) \vdash_N^* (q, \varepsilon, \varepsilon) \}$$

$$N = (\Sigma, Q, \Gamma, \delta, q_0, \perp, F)$$

Oss: Per un certo PDA N, spesso

$$L[N] \neq P[N]$$

Oss: Dimostreremo però che se $L = L[N]$, allora esiste N' tale che $L = P[N']$, e viceversa, se $L = P[N]$, allora esisterà N'' tale che $L = L[N'']$

cioè non cambia la classe dei linguaggi riconosciuti da PDA per stato finale o per pila vuota

Esempio: PDA per $\{ww^R \mid w \in \{a,b\}^*\}$

$N = (\Sigma, Q, \Gamma, \delta, q_0, \perp, F)$ dove

- $\Sigma = \{a, b\}$
- $Q = \{s_0, s_1, s_2\}$
- $\Gamma = \{a, b, \perp\}$
- $q_0 = s_0$
- $F = \{s_2\}$

$\delta(s_0, a, X) = \{(s_0, aX)\} \quad \forall X \in \Gamma$

$\delta(s_0, b, X) = \{(s_0, bX)\} \quad \forall X \in \Gamma$

$\delta(s_0, \varepsilon, X) = \{(s_1, X)\} \quad \forall X \in \Gamma$

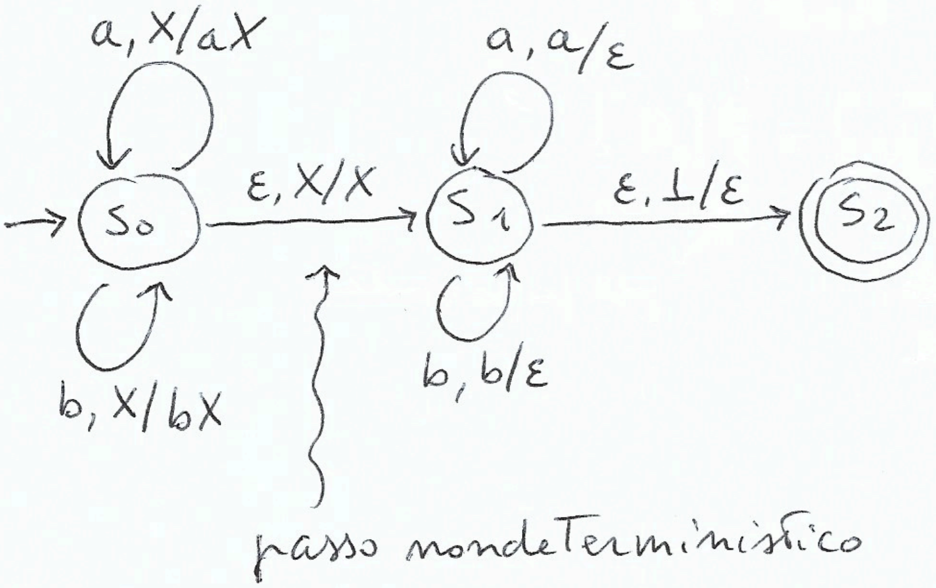
$\delta(s_1, a, a) = \{(s_1, \varepsilon)\}$

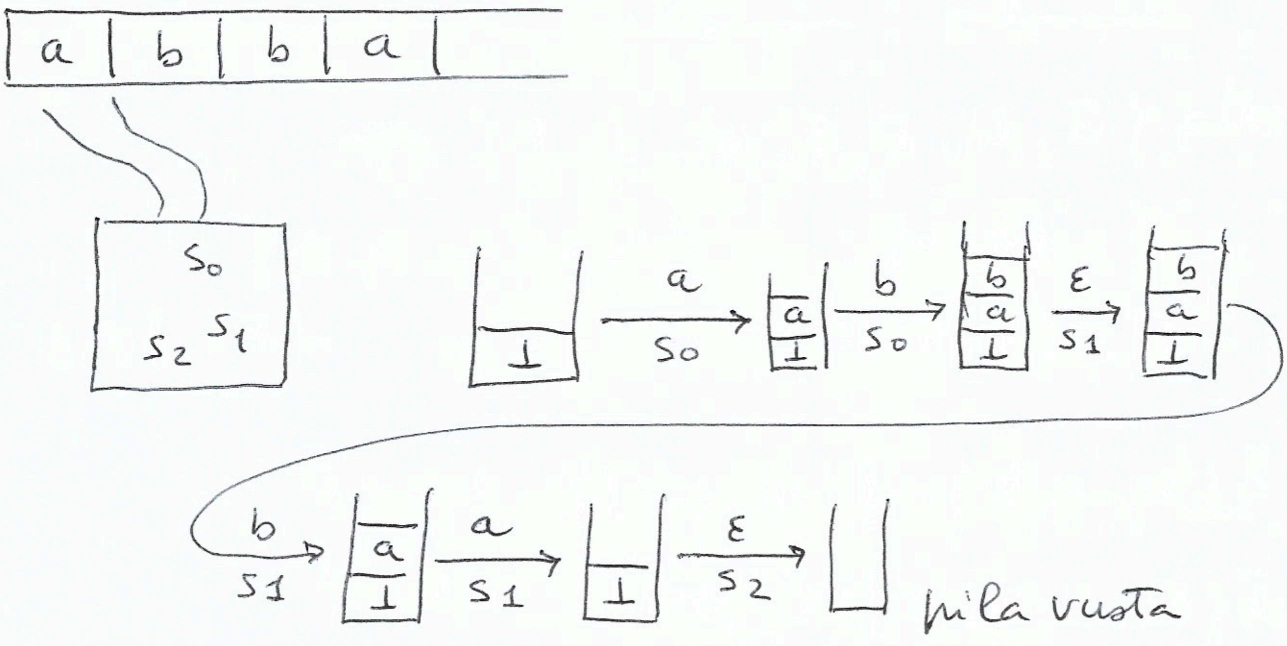
$\delta(s_1, b, b) = \{(s_1, \varepsilon)\}$

$\delta(s_1, \varepsilon, \perp) = \{(s_2, \varepsilon)\}$

$\delta(s_2, \alpha, X) = \emptyset \quad \forall \alpha \in \Sigma \cup \{\varepsilon\}, \forall X \in \Gamma$

Diagramma di Transizione





- <u>esiste</u> una "computazione" per abba che porta l'automa N allo stato finale s_2 (così come a svuotare la pila)

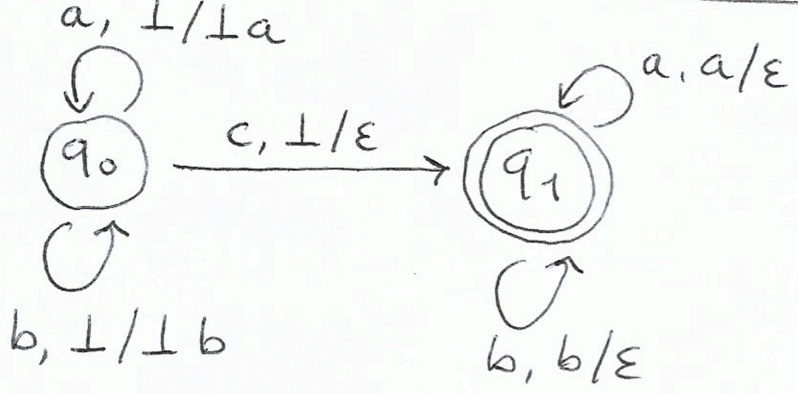
  $\Rightarrow$ abba è riconosciuta (sia per stato finale, sia per pila vuota)

- si può vedere che "svuoto la pila sse vado sullo stato s_2"

$$\Rightarrow L[N] = P[N]$$

per stato finale — per pila vuota

<u>Esempio</u>: PDA per $L = \{ w c w^R \mid w \in \{a,b\}^* \}$



- N è <u>deterministico</u>:
  - $|\delta(q,a,Z)| \leq 1 \quad \forall q \in Q, \ \forall a \in \Sigma \cup \{\varepsilon\}, \ \forall Z \in \Gamma$
  - se $\delta(q,\varepsilon,Z) \neq \emptyset$, allora $\delta(q,a,Z) = \emptyset \quad \forall a \in \Sigma$

  <u>Oss</u>: diversamente dai DFA, un PDA deterministico non garantisce di leggere tutto l'input. <u>Ad es</u>: N si blocca con input acc

- $L = \mathbf{P}[N]$ cioè il linguaggio L è riconosciuto <u>per pila vuota</u>

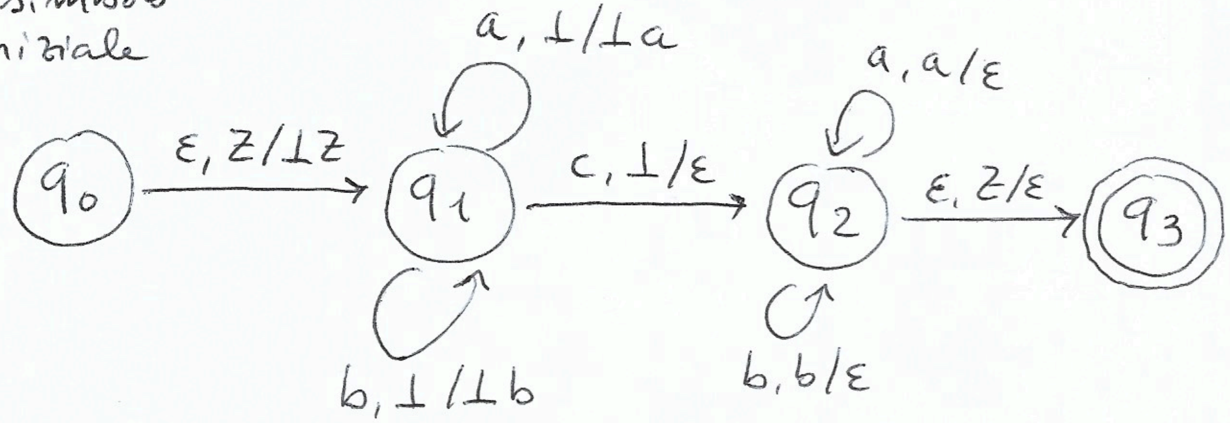
- $L[N] = \{ wcy \mid w \in \{a,b\}^* \text{ e } y \text{ è prefisso di } w^R \}$

  <u>Ad es</u>: $ac \in L[N]$ $\quad w = a$, c, $y = \varepsilon$

- Se l'input è <u>acab</u>, questo non viene letto tutto: N si blocca dopo aver letto <u>aca</u> sullo stato finale cioè riconosce <u>aca</u> ma non <u>acab</u>

$$L = \{ w c w^R \mid w \in \{a, b\}^* \}$$

Z simbolo iniziale



$$L = L[N] = P[N]$$

N è deterministico e riconosce L sia per stato finale, sia per pila vuota

---

La <u>classe</u> dei linguaggi riconosciuti per pila vuota o per stato finale <u>non cambia</u>

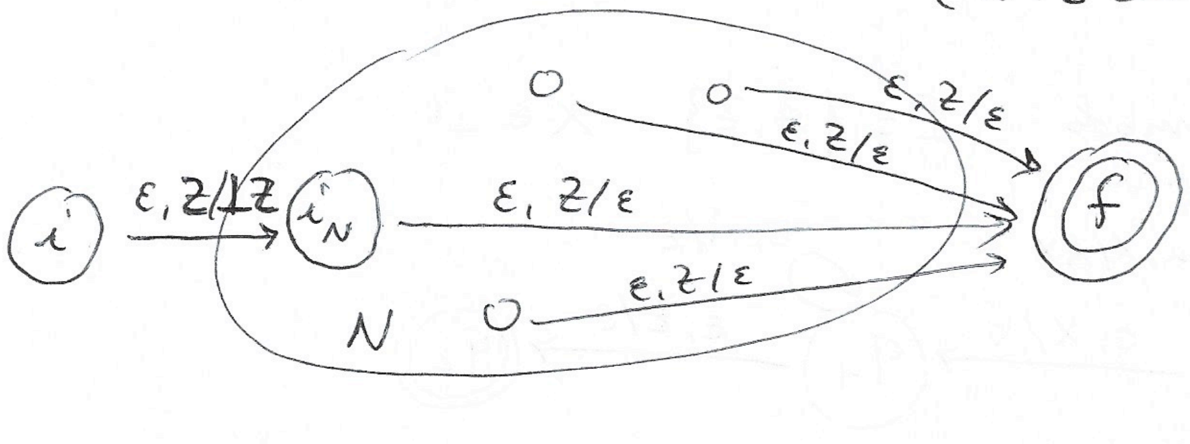
<u>Teorema</u>

(i) Se $L = P[N]$ (per pila vuota), possiamo costruire N' tale che $L = L[N']$

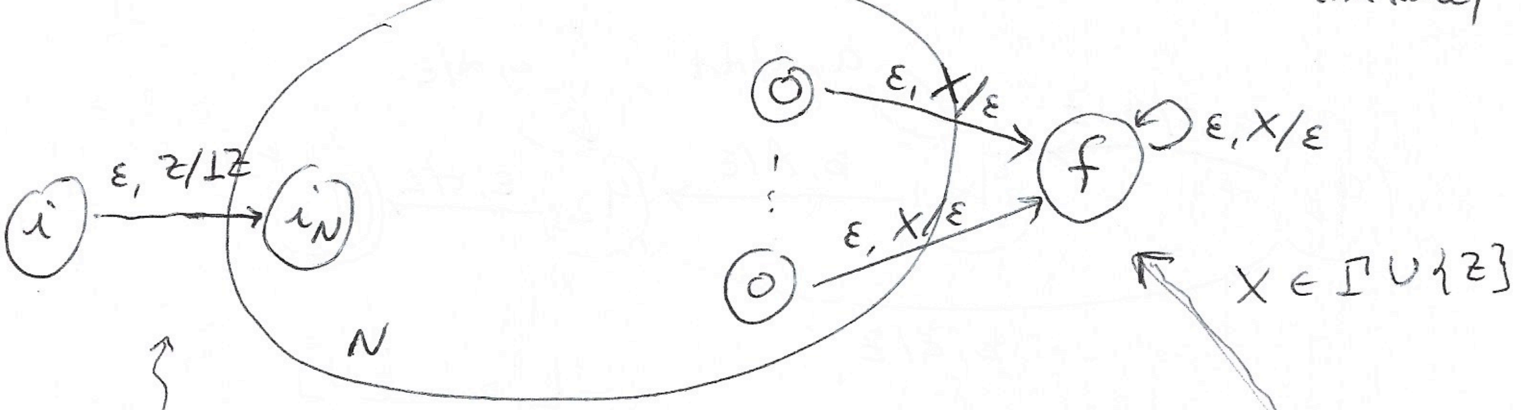
(ii) Se $L = L[N]$ (per stato finale), possiamo costruire N' tale che $L = P[N']$.

Dimostrazione

(i) N riconosce per pila vuota (con $\perp$ come iniziale) $\Rightarrow$ costruiamo N' che riconosce per stato finale (con Z come iniziale)



(ii) N riconosce per stato finale (con $\perp$ come iniziale) $\Rightarrow$ costruiamo N' che riconosce per pila vuota (con Z come iniziale)



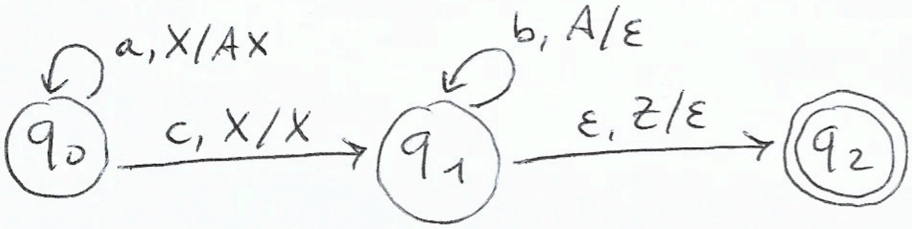
serve per evitare che uno stato di N possa vuotare lo stack ($Z \notin \Gamma$)

Allora l'unico stato che può vuotare lo stack è f

Esercizi (da ling. a PDA)

1) $L = \{ a^n c b^n \mid n \geq 0 \}$

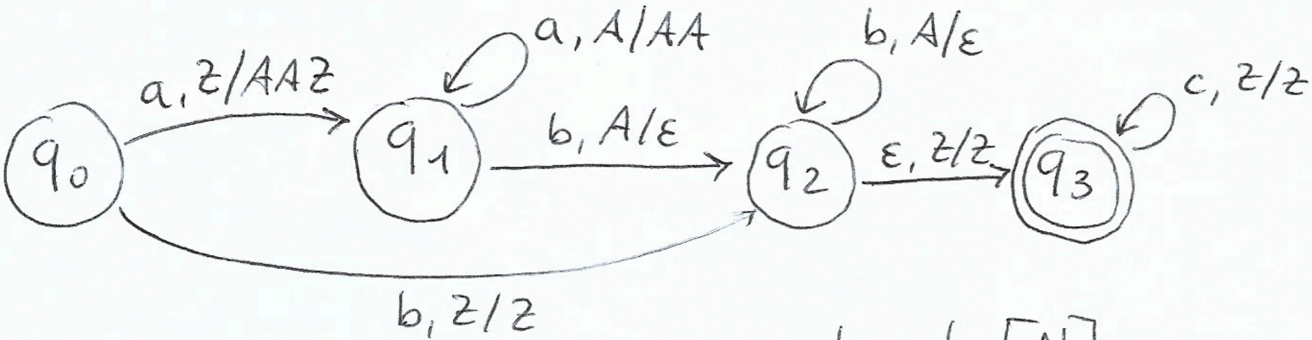
Z simbolo iniziale $\quad \Gamma = \{A, Z\} \quad X \in \Gamma$



$L = L[N] = P[N]$

---

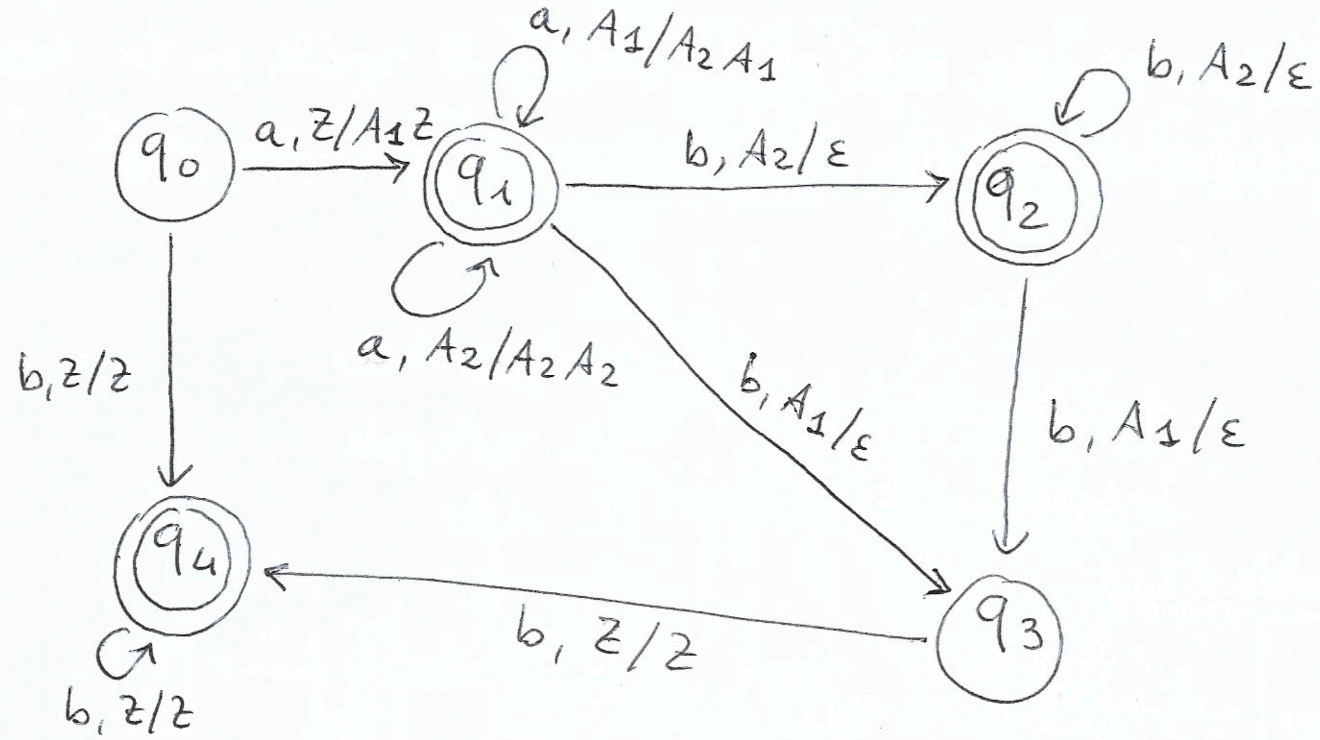
2) $L = \{ a^n b^{n+1} c^m \mid n, m \geq 0 \}$



$L = L[N]$ ← per stato finale

$P[N] = \emptyset$

3) $L = \{ a^n b^m \mid n \neq m,\ n, m \geq 0 \}$



$\Gamma = \{ A_1, A_2, Z \}$

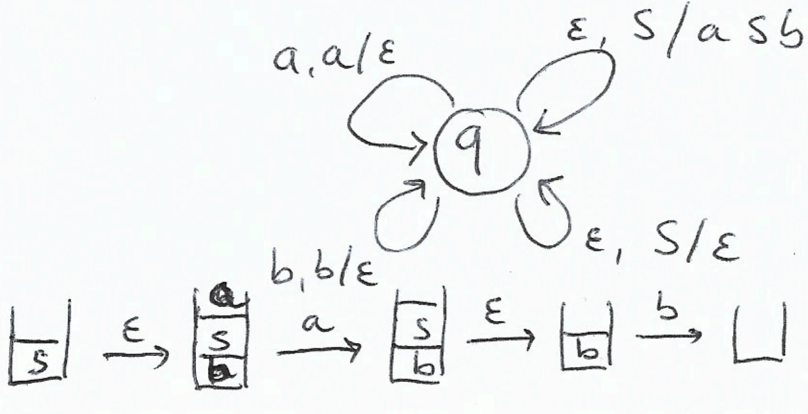
$L = L[N]$

per stato finale

All'incontrario, come ottenere un PDA a partire da una grammatica libera?

Un possibile metodo (top-down) è il seguente

$S \to aSb \mid \varepsilon$

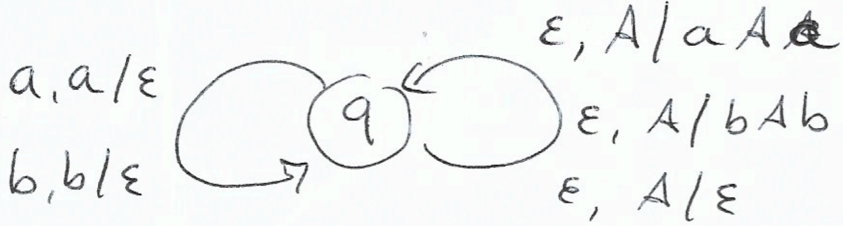


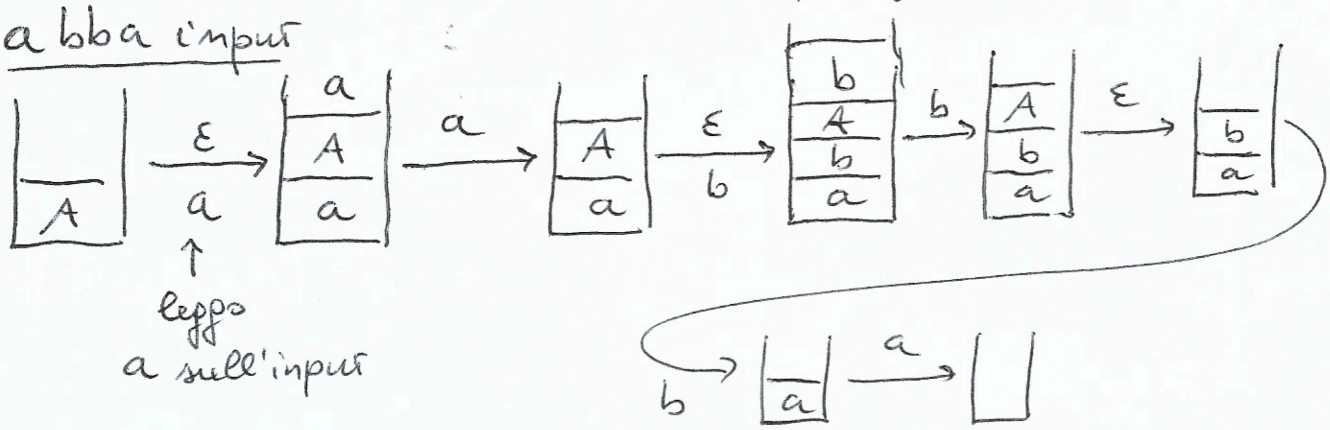
— S simbolo iniziale sulla pila

— riconosce per pila vuota

$A \to aAa \mid bAb \mid \varepsilon$

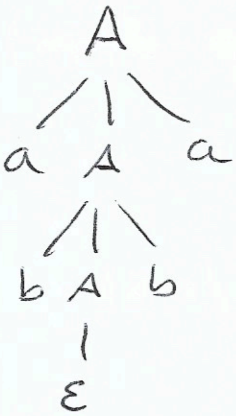
PDA con un solo stato, con A sulla pila come simbolo iniziale, che riconosce per pila vuota





Costruisca una derivazione leftmost a partire da A

$A \Rightarrow aAa \Rightarrow abAba \Rightarrow abba$



<u>Teorema</u> Un linguaggio L è libero da contesto sse è accettato da un PDA

<u>Dim)</u>

$\Rightarrow$) Cioè se L è libero, allora $\exists$ PDA N tale che $L = P[N]$.

Se L è libero, allora $\exists\, G = (NT, T, R, S)$ libera tale che $L = L(G)$. Costruiamo il PDA $N = (\underset{\Sigma}{T}, \{q\}, \underset{\Gamma}{T \cup NT}, \delta, q, \underset{\perp}{S}, \emptyset)$, dove la funzione di transizione δ simula la costruzione di una derivazione canonica sinistra (riscrivo sempre il nonterminale sul top, che è quello più a sx nella derivazione leftmost)

$$\delta(q, \varepsilon, A) = \{(q, \beta) \mid A \to \beta \in R\} \qquad \forall A \in NT$$

$$\delta(q, a, a) = \{(q, \varepsilon)\} \qquad \forall a \in T$$

Ogni volta che N ha un nonterminale A in cima alla pila, sceglie nondeterministicamente una produzione per A, senza consumare l'input. Se invece sulla pila c'è un terminale a e se l'input presenta proprio a, questi vengono consumati. Se invece l'input è $b \neq a$, ci si blocca, e si fa backtracking provando con un'altra produzione

Si può dimostrare, per induzione sulla lunghezza di w, che

$$S \Rightarrow^* w \quad \text{sse} \quad (q, w, S) \vdash_N^* (q, \varepsilon, \varepsilon)$$

con derivazione leftmost

$\Leftarrow$) Cioè se $L = P[N]$, allora esiste G libera tale che $L = L(G)$.

Molto più difficile: prova solo schematizzata

Lemma 1: Ogni PDA N può essere simulato da un PDA N' con un solo stato (aumentando opportunamente i simboli della pila)

Lemma 2: Ogni PDA con un solo stato, ha una equivalente grammatica libera.

Idea della prova:

Se $(q, B_1 B_2 \dots B_k) \in \delta(q, a, A)$ con $a \in \Sigma \cup \{\varepsilon\}$ nel PDA con un solo stato, allora la grammatica G contiene la produzione

$$A \to a B_1 \dots B_k$$

Si può dimostrare che $S \Rightarrow_G^* w$ sse $(q, w, S) \vdash_N^* (q, \varepsilon, \varepsilon)$

Riassumendo:

Un linguaggio L è libero

sse

è accettato da un PDA (nondeterministico)