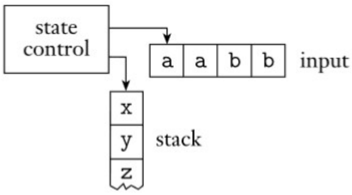
Argomenti trattati durante la lezione:

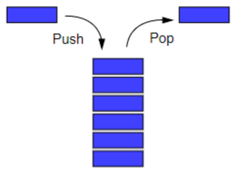
- Chiusura per linguaggi context-free
- Automi a pila. Equivalenza PDA e CFG
- Esercizi grammatiche CF e automi a pila
- Esercizi preparazione primo compitino

- **Input**: stringa di caratteri dell'alfabeto
- **Memoria**: stati + pila
- **Funzione di transizione**: dato lo stato corrente, un simbolo di input ed il simbolo in cima alla pila, stabilisce quali possono essere gli stati successivi e i simboli da scrivere sulla pila



La pila è un dispositivo di memoria last in, first out (LIFO):

- **Push**: scrivi un nuovo simbolo in cima alla pila e "spingi giù" gli altri
- **Pop**: leggi e rimuovi il simbolo in cima alla pila (top)



La pila permette di avere memoria infinita (ad accesso limitato)

Un automa a pila (o Pushdown Automata, PDA) è una sestupla $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$:

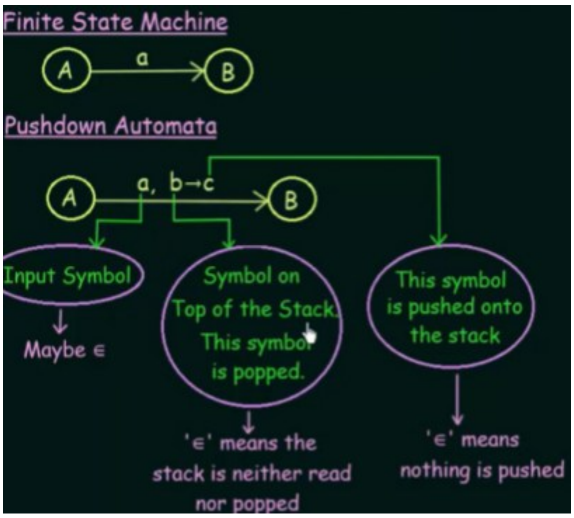
- Q è l'insieme finito di stati
- Σ è l'alfabeto di input
- Γ è l'alfabeto della pila
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \mapsto 2^{Q \times \Gamma_\varepsilon}$ è la funzione di transizione
- $q_0 \in Q$ è lo stato iniziale
- $F \subseteq Q$ è l'insieme di stati accettanti

(dove $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ e $\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$)

Accettazione per pila vuota

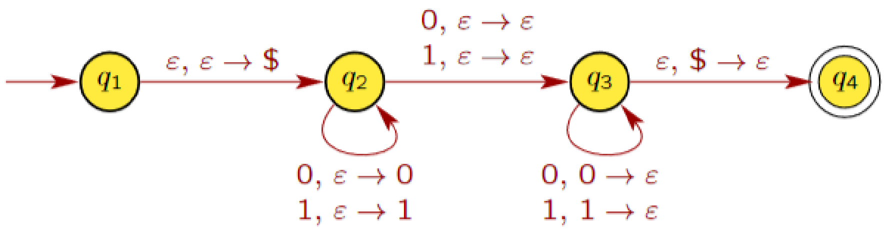
Un PDA accetta la parola w per pila vuota se esiste una computazione che

- consuma tutto l'input
- termina con la pila vuota ($s_m = \varepsilon$)



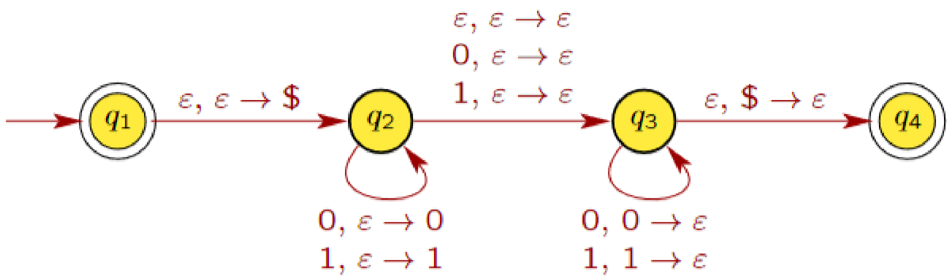
(b) $B = \{ w \in \{0, 1\}^* \mid w = w^{\mathcal{R}} \text{ and the length of } w \text{ is odd} \}$

Answer:



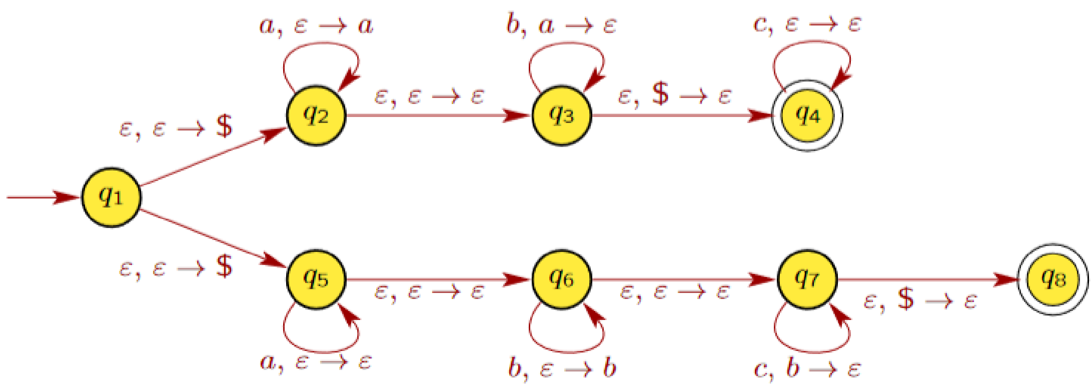
(c) $C = \{ w \in \{0, 1\}^* \mid w = w^{\mathcal{R}} \}$

Answer:



(d) $D = \{ a^i b^j c^k \mid i, j, k \geq 0, \text{ and } i = j \text{ or } j = k \}$

Answer:



Lemma

Se un linguaggio è context free, allora esiste un PDA che lo riconosce

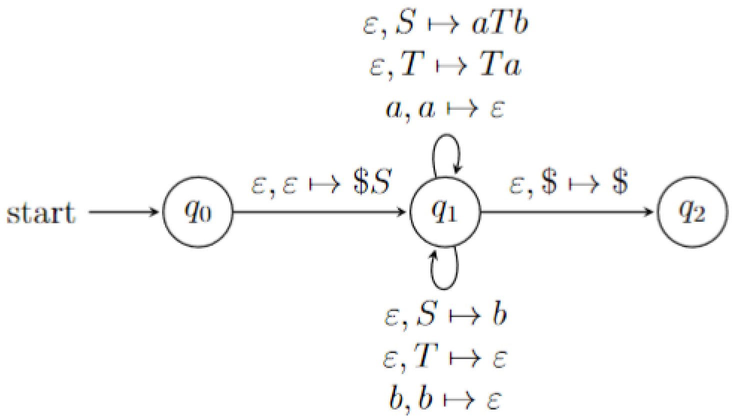
Trasformiamo la seguente CFG in PDA:

$$S \to aTb \mid b$$
$$T \to Ta \mid \varepsilon$$

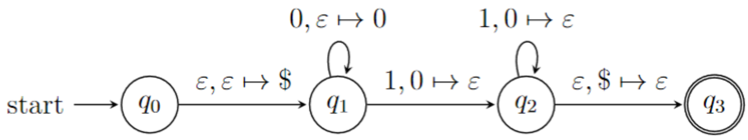
L'idea quindi è di seguire la leftmost derivation, quindi avremo:

- $\varepsilon, S \to aTb$ (prima cosa fatta)
- l'altra transizione possibile di S
- le due transizioni possibili di T
- a e b (in pop) perché simboli terminali

1.2 Rappresentazione grafica del PDA



Trasformiamo il PDA per il linguaggio $\{0^n1^n \mid n \geq 0\}$ in grammatica:



1.1 Analisi del PDA

Prima di procedere con la trasformazione, analizziamo brevemente il PDA dato:

- L'automa a pila riconosce il linguaggio $\{0^n1^n \mid n \geq 0\}$
- Il PDA ha 4 stati: q_0 (iniziale), q_1, q_2 e q_3 (finale)
- In q_1, per ogni '0' letto, viene inserito un '0' nello stack
- In q_2, per ogni '1' letto, viene rimosso un '0' dallo stack
- Il PDA accetta se, dopo aver letto tutti i simboli di input, raggiunge q_3 svuotando lo stack

$$
\begin{aligned}
&A^{\$}_{03} \to A^{\$}_{13} \quad \text{(dalla transizione } q_0 \to q_1 \text{ con } \varepsilon, \varepsilon \mapsto \$) \\
&A^{\$}_{13} \to A^{\$}_{12} A^{\$}_{23} \quad \text{(decomposizione in sottoproblemi)} \\
&A^{\$}_{13} \to \varepsilon \quad \text{(caso } \varepsilon \text{, se } n = 0) \\
&A^{\$}_{12} \to 0 A^{0}_{11} \quad \text{(dalla transizione } q_1 \to q_1 \text{ con } 0, \varepsilon \mapsto 0) \\
&A^{0}_{11} \to 0 A^{0}_{11} A^{0}_{11} \mid \varepsilon \quad \text{(per concatenare più '0')} \\
&A^{\$}_{23} \to \varepsilon \quad \text{(dalla transizione } q_2 \to q_3 \text{ con } \varepsilon, \$ \mapsto \varepsilon) \\
&A^{0}_{12} \to 1 \quad \text{(dalla transizione } q_1 \to q_2 \text{ con } 1, 0 \mapsto \varepsilon) \\
&A^{0}_{22} \to 1 \quad \text{(dalla transizione } q_2 \to q_2 \text{ con } 1, 0 \mapsto \varepsilon)
\end{aligned}
$$

Possiamo semplificare questa grammatica attraverso sostituzioni e eliminazioni di produzioni ridondanti:

$$
\begin{aligned}
&S \to A \\
&A \to 0A1 \mid \varepsilon
\end{aligned}
$$

Questa grammatica genera esattamente il linguaggio $\{0^n 1^n \mid n \geq 0\}$.

Dopo aver dimostrato questi due versi, discutiamo una serie di esercizi per il primo parziale.

1. **(12 punti)** Diciamo che una stringa x è un *prefisso* della stringa y se esiste una stringa z tale che $xz = y$, e che è un *prefisso proprio* di y se vale anche $x \neq y$. Dimostra che se $L \subseteq \Sigma^*$ è un linguaggio regolare allora anche il linguaggio

$$NOPREFIX(L) = \{w \in L \mid \text{ nessun prefisso proprio di } w \text{ appartiene ad } L\}$$

è un linguaggio regolare.

Soluzione: Se L è un linguaggio regolare, allora sappiamo che esiste un DFA $A = (Q, \Sigma, \delta, q_0, F)$ che riconosce L. Costruiamo un DFA A' che accetta il linguaggio $NOPREFIX(L)$, aggiungendo uno stato "pozzo" agli stati di A, ossia uno stato non finale q_s tale che la funzione di transizione obbliga l'automa a rimanere per sempre in q_s una volta che lo si raggiunge. Lo stato iniziale e gli stati finali rimangono invariati. La funzione di transizione di A' si comporta come quella di A per gli stati non finali, mentre va verso lo stato pozzo per qualsiasi simbolo dell'alfabeto a partire dagli stati finali. In questo modo le computazioni accettanti di A' sono sempre sequenze di stati dove solo l'ultimo stato è finale, mentre tutti quelli intermedi sono non finali. Di conseguenza le parole che A' accetta sono accettate anche da A, mentre tutti i prefissi propri sono parole rifiutate da A, come richiesto dalla definizione del linguaggio $NOPREFIX(L)$.

Formalmente, $A' = (Q', \Sigma, \delta', q'_0, F')$ è definito come segue.

- $Q' = Q \cup \{q_s\}$, con $q_s \notin Q$.
- L'alfabeto Σ rimane lo stesso.
- $\delta'(q, a) = \begin{cases} \delta(q, a) & \text{se } q \notin F \\ q_s & \text{altrimenti} \end{cases}$
- $q'_0 = q_0$. Lo stato iniziale non cambia.
- $F' = F$. Gli stati finali rimangono invariati.

Per dimostrare che A' riconosce il linguaggio $NOPREFIX(L)$, dobbiamo considerare due casi.

- Se $w \in NOPREFIX(L)$, allora sappiamo che $w \in L$, mentre nessun prefisso proprio di w appartiene ad L. Di conseguenza esiste una computazione di A che accetta la parola:
$$s_0 \xrightarrow{w_1} s_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} s_n$$
con $s_0 = q_0$ e $s_n \in F$. Siccome tutti i prefissi propri di w sono rifiutati da A, allora gli stati $s_0, \dots, s_{n-1}$ sono tutti non finali. Per la definizione di A', la computazione che abbiamo considerato è anche una computazione accettante per A', e di conseguenza, $w \in L(A')$.
- Viceversa, se w è accettata dal nuovo automa A', allora esiste una computazione accettante che ha la forma
$$s_0 \xrightarrow{w_1} s_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} s_n$$

con $s_0 = q_0$, $s_n \in F$ e dove tutti gli stati intermedi $s_0, \dots, s_{n-1}$ sono non finali. Di conseguenza, la computazione è una computazione accettante anche per A, quindi $w \in L$. Siccome tutti gli stati intermedi della computazione sono non finali, allora A rifiuta tutti i prefissi propri di w, e quindi $w \in NOPREFIX(L)$.

2. (12 punti) Considera il linguaggio
$$L_2 = \{uvvu \mid u, v \in \{0,1\}^*\}.$$
Dimostra che L_2 non è regolare.

Soluzione: Usiamo il Pumping Lemma per dimostrare che il linguaggio non è regolare. Supponiamo per assurdo che L_2 sia regolare:

- sia k la lunghezza data dal Pumping Lemma;
- consideriamo la parola $w = 0^k110^k$, che è di lunghezza maggiore di k ed appartiene ad L_2 perché la possiamo scrivere come $uvvu$ ponendo $u = 0^k$ e $v = 1$;
- sia $w = xyz$ una suddivisione di w tale che $y \neq \varepsilon$ e $|xy| \leq k$;
- poiché $|xy| \leq k$, allora x e y sono entrambe contenute nella sequenza iniziale di 0. Inoltre, siccome $y \neq \varepsilon$, abbiamo che $x = 0^q$ e $y = 0^p$ per qualche $q \geq 0$ e $p > 0$. z contiene la parte rimanente della stringa: $z = 0^{k-q-p}110^k$. Consideriamo l'esponente $i = 2$: la parola xy^2z ha la forma
$$xy^2z = 0^q0^{2p}0^{k-q-p}110^k = 0^{k+p}110^k$$
La parola iterata xy^2z non appartiene ad L_2 perché non si può scrivere nella forma $uvvu$. Visto che la parte iniziale deve essere uguale a quella finale, si deve porre $u = 0^k$, ma in questo caso la parte centrale della parola è 0^p11 che non si può dividere in due metà uguali. Viceversa, se si pone $v = 1$ per avere la parte centrale della parola composta da due metà uguali, allora si ottiene una sequenza iniziale di 0 che è più lunga della sequenza finale di 0.

3. (12 punti) Una grammatica context-free è *lineare* se ogni regola in R è nella forma $A \to aBc$ o $A \to a$ per qualche $a, c \in \Sigma \cup \{\varepsilon\}$ e $A, B \in V$. I linguaggi generati dalle grammatiche lineari sono detti *linguaggi lineari*. Dimostra che i linguaggi regolari sono un sottoinsieme proprio dei linguaggi lineari.

Soluzione. Per risolvere l'esercizio dobbiamo dimostrare che ogni linguaggio regolare è anche un linguaggio lineare (i linguaggi regolari sono un sottoinsieme dei linguaggi lineari), e che esistono linguaggi lineari che non sono regolari (l'inclusione è propria).

- Dato un linguaggio regolare L, sappiamo che esiste un DFA $A = (Q, \Sigma, \delta, q_0, F)$ che riconosce L. Inoltre, sappiamo che ogni DFA può essere convertito in una grammatica context-free dove le regole sono del tipo $R_i \to aR_j$ per ogni transizione $\delta(q_i, a) = q_j$ del DFA, e del tipo $R_i \to \varepsilon$ per ogni stato finale del q_i del DFA. Entrambi i tipi di regola rispettano le condizioni di linearità, quindi la grammatica equivalente al DFA è lineare, e questo implica che L è un linguaggio lineare.

- Consideriamo il linguaggio non regolare $L = \{0^n 1^n \mid n \geq 0\}$. La seguente grammatica lineare genera L:
$$S \to 0S1 \mid \varepsilon$$
Quindi, esiste un linguaggio lineare che non è regolare.

3. (12 punti) Dimostra che se $L \subseteq \Sigma^*$ è un linguaggio context-free allora anche il seguente linguaggio è context-free:

$$dehash(L) = \{dehash(w) \mid w \in L\},$$

dove $dehash(w)$ è la stringa che si ottiene cancellando ogni # da w.

Soluzione: Se L è un linguaggio context-free, allora esiste una grammatica $G = (V, \Sigma, R, S)$ che lo genera. Possiamo assumere che questa grammatica sia in forma normale di Chomsky. Per dimostrare che $dehash(L)$ è context-free, dobbiamo essere in grado di definire una grammatica che possa generarlo. Questa grammatica è una quadrupla $G' = (V', \Sigma', R', S')$ definita come segue.

- L'alfabeto tutti i simboli di Σ tranne #: $\Sigma' = \Sigma \setminus \{\#\}$.
- L'insieme di variabili è lo stesso della grammatica G: $V' = V$.

- Il nuovo insieme di regole R' è ottenuto rimpiazzando ogni regola nella forma $A \to \#$ con la regola $A \to \varepsilon$, e lasciando invariate le regole nella forma $A \to BC$, le regole nella forma $A \to b$ quando $b \neq \#$, e la regola $S \to \varepsilon$ (se presente).
- La variabile iniziale rimane la stessa: $S' = S$.

Data una derivazione $S \Rightarrow^* w$ della grammatica G possiamo costruire una derivazione nella nuova grammatica G' che applica le stesse regole nello stesso ordine, e che deriva una parola dove ogni $\#$ è rimpiazzato dalla parola vuota ε. Quindi, G' permette di derivare tutte le parole in $dehash(L)$.

Viceversa, data una derivazione $S \Rightarrow^* w$ della nuova grammatica G' possiamo costruire una derivazione nella grammatica G che applica le stesse regole nello stesso ordine. Di conseguenza, in ogni punto in cui la derivazione per G' applica la regola modificata $A \to \varepsilon$, la derivazione per G applicherà la regola $A \to \#$ inserendo un $\#$ in qualche punto della parola w. Al termine della derivazione si ottiene una parola w' tale che $dehash(w') = w$. Quindi, G' permette di derivare solo parole che appartengono a $dehash(L)$.

1. **(12 punti)** Data una stringa w di 0 e 1, il *flip* di w si ottiene cambiando tutti gli 0 in w con 1 e tutti gli 1 in w con 0. Dato un linguaggio L, il flip di L è il linguaggio

$$flip(L) = \{w \in \{0,1\}^* \mid \text{ il flip di } w \text{ appartiene ad } L\}.$$

Dimostra che la classe dei linguaggi regolari è chiusa rispetto all'operazione di flip.

Soluzione: Se L è un linguaggio regolare, allora sappiamo che esiste un DFA $A = (Q, \Sigma, \delta, q_0, F)$ che riconosce L. Costruiamo un DFA $A' = (Q', \Sigma, \delta', q'_0, F')$ che accetta il linguaggio $flip(L)$ come segue.

- $Q' = Q$. L'insieme degli stati rimane lo stesso.
- L'alfabeto Σ rimane lo stesso.
- Per ogni stato $q \in Q$, $\delta'(q, 0) = \delta(q, 1)$ e $\delta'(q, 1) = \delta(q, 0)$. La funzione di transizione scambia gli 0 con 1 e gli 1 con 0.
- $q'_0 = q_0$. Lo stato iniziale non cambia.
- $F' = F$. Gli stati finali rimangono invariati.

Per dimostrare che A' riconosce il linguaggio $flip(L)$, dobbiamo considerare due casi.

- Se $w \in flip(L)$, allora sappiamo che il flip di w appartiene ad L. Chiamiamo $\overline{w}$ il flip di w. Siccome A riconosce L, allora esiste una computazione di A che accetta $\overline{w}$:

$$s_0 \xrightarrow{\overline{w}_1} s_1 \xrightarrow{\overline{w}_2} \dots \xrightarrow{\overline{w}_n} s_n$$

con $s_0 = q_0$ e $s_n \in F$. Se scambiamo gli zeri e gli uni nella computazione, otteniamo una computazione accettante per A' sulla parola w. Di conseguenza, abbiamo dimostrato che $w \in L(A')$.

- Viceversa, se w è accettata dal nuovo automa A', allora esiste una computazione accettante che ha la forma

$$s_0 \xrightarrow{w_1} s_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} s_n$$

con $s_0 = q_0$, $s_n \in F'$. Se scambiamo gli zeri e gli uni nella computazione, otteniamo una computazione accettante per A sul flip di w. Di conseguenza, il flip di w appartiene ad L e abbiamo dimostrato che $w \in \mathit{flip}(L)$.

Problema (9 punti): Considera la seguente funzione da $\{0,1\}^* \to \{0,1\}^*$:

$$\text{stutter}(w) = \begin{cases} \varepsilon & \text{se } w = \varepsilon \\ aa \cdot \text{stutter}(x) & \text{se } w = ax \text{ per qualche simbolo } a \text{ e parola } x \end{cases} \tag{1}$$

Dimostra che se L è un linguaggio context-free sull'alfabeto $\{0,1\}$, allora anche il seguente linguaggio è context-free:

$$\text{stutter}(L) = \{\text{stutter}(w) \mid w \in L\}. \tag{2}$$

(Riferimento: File – Stutter – Soluzioni)

Automi e Linguaggi Formali – 8/7/2022
Secondo Appello – Prima Parte

1. **(12 punti)** Se L è un linguaggio sull'alfabeto $\{0,1\}$, la *rotazione a sinistra* di L è l'insieme delle stringhe

$$\text{ROL}(L) = \{wa \mid aw \in L, w \in \{0,1\}^*, a \in \{0,1\}\}.$$

Per esempio, se $L = \{0, 01, 010, 10100\}$, allora $\text{ROL}(L) = \{0, 10, 100, 01001\}$. Dimostra che se L è regolare allora anche $\text{ROL}(L)$ è regolare.

2. **(12 punti)** Considera l'alfabeto $\Sigma = \{0,1\}$, e sia L_2 l'insieme di tutte le stringhe che contengono almeno un 1 nella loro prima metà:

$$L_2 = \{uv \mid u \in \Sigma^*1\Sigma^*, v \in \Sigma^* \text{ e } |u| \leq |v|\}.$$

Dimostra che L_2 non è regolare.

3. **(12 punti)** Mostra che per ogni PDA P esiste un PDA P_2 con due soli stati tale che $L(P_2) = L(P)$. *Suggerimento:* usate la pila per tenere traccia dello stato di P.

(Riferimento: Appello 08/07/22 – Soluzioni)

(Andiamo ad altri esercizi… - Tutti inseriti in "Raccolta esercizi risolti")