Esercizi di teoria dei linguaggi

Indice

1. Lezione 01

2. Lezione 02

2.1. Esercizio 01

Considerate l'alfabeto $\Sigma = \{a, b\}$.

- Fornite una grammatica context-free per il linguaggio delle stringhe palindrome di lunghezza pari su Σ, cioè per l'insieme $\mathrm{PAL}_{\text{pari}} = \{ww^R \mid w \in \Sigma^*\}$.

Regole di produzione:
- $S \longrightarrow \varepsilon$;
- $S \longrightarrow aSa$;
- $S \longrightarrow bSb$.

- Modificate la grammatica precedente per generare l'insieme PAL di tutte le stringhe palindrome su Σ.

Regole di produzione:
- $S \longrightarrow \varepsilon$;
- $S \longrightarrow aSa$;
- $S \longrightarrow bSb$;
- $S \longrightarrow L$;
- $L \longrightarrow a$;
- $L \longrightarrow b$.

- Per ogni $k \in [0, 3]$ rispondete alla domanda "il linguaggio PAL é di tipo k?" giustificando la risposta.

- Tipo 0: sì, ogni linguaggio é un linguaggio di tipo 0;
- Tipo 1: sì, per ogni regola di produzione $\alpha \longrightarrow \beta$ vale $|\beta| \geq |\alpha|$;
- Tipo 2: sì, ogni regola di produzione $\alpha \longrightarrow \beta$ vede $\alpha \in V$ e $\beta \in (V \cup \Sigma^*)$;
- Tipo 3: no, la regola $S \longrightarrow aSa$ non é nella forma $A \longrightarrow aB$ oppure $A \longrightarrow a$.

Se sostituiamo l'alfabeto con $\Sigma = \{a, b, c\}$, le risposte al punto precedente cambiano? E se lo sostituiamo con $\Sigma = \{a\}$?

Se $\Sigma = \{a, b, c\}$ le risposte non cambiano visto che vanno aggiunte le regole:
- $S \longrightarrow cSc$;
- $L \longrightarrow c$.

Se $\Sigma = \{a\}$ le regole di produzione diventano:
- $S \longrightarrow \varepsilon$;
- $S \longrightarrow a$;
- $S \longrightarrow aSa$;

ma questo non fa cambiare le risposte.

2.2. Esercizio 02

Non ancora spiegato

2.3. Esercizio 03

Sia $\Sigma = \{(,)\}$ un alfabeto i cui simboli sono la parentesi aperta e la parentesi chiusa.

Scrivete una grammatica context-free che generi il linguaggio formato da tutte le sequenze di parentesi correttamente bilanciate, come ad esempio `(()(()))()`.

Regole di produzione:
- $S \longrightarrow \varepsilon$;
- $S \longrightarrow (S)$;
- $S \longrightarrow SS$.

Risolvete il punto precedente per un alfabeto con due tipi di parentesi, come $\Sigma = \{(,),[,]\}$, nel caso non vi siano vincoli tra i tipi di parentesi (le tonde possono essere contenute tra quadre e viceversa). Esempio `[()([])[]]`, ma non `[[][(])()]`.

Regole di produzione:
- $S \longrightarrow \varepsilon$;
- $S \longrightarrow (S)$;
- $S \longrightarrow [S]$;
- $S \longrightarrow SS$.

Risolvete il punto precedente con $\Sigma = \{(,),[,]\}$, con il vincolo che le parentesi quadre non possano mai apparire all'interno di parentesi tonde. Esempio `[()(())[][]](()())`, ma non `[()([])[]]`.

Regole di produzione:
- $S \longrightarrow \varepsilon$;
- $S \longrightarrow [S]$;
- $S \longrightarrow SS$;
- $S \longrightarrow I$;
- $I \longrightarrow \varepsilon$;
- $I \longrightarrow (I)$;
- $I \longrightarrow II$.

2.4. Esercizio 04

Sia $G = (V, \Sigma, P, S)$ la grammatica con $V = \{S, B, C\}$, $\Sigma = \{a, b, c\}$ e P contenente le seguenti produzioni:
- $S \longrightarrow aSBC \mid aBC$;
- $CB \longrightarrow BC$;
- $aB \longrightarrow ab$;
- $bB \longrightarrow bb$;
- $bC \longrightarrow bc$;
- $cC \longrightarrow cc$.

Dopo avere stabilito di che tipo é G, provate a derivare alcune stringhe. Riuscite a dire da quali stringhe é formato il linguaggio generato da G?

La grammatica G é di tipo 1.

Deriviamo qualche stringa:
- $S \longrightarrow aBC \longrightarrow abC \longrightarrow abc$;
- $S \longrightarrow aSBC \longrightarrow aaBCBC \longrightarrow aabCBC \longrightarrow aabBCC \longrightarrow aabbCC \longrightarrow aabbcC \longrightarrow aabbcc$.

Il linguaggio $L(G)$ è l'insieme $\{a^n b^n c^n \mid n \geq 1\}$.

2.5. Esercizio 05

Sia $G = (V, \Sigma, P, S)$ la grammatica con $V = \{S, B, C\}$, $\Sigma = \{a, b, c\}$ e P contenente le seguenti produzioni:
- $S \longrightarrow aBSc \mid abc$;
- $Ba \longrightarrow aB$;
- $Bb \longrightarrow bb$.

Dopo avere stabilito di che tipo é G, provate a derivare alcune stringhe. Riuscite a dire da quali stringhe é formato il linguaggio generato da G?

La grammatica G é di tipo 1.

Deriviamo qualche stringa:
- $S \longrightarrow abc$;
- $S \longrightarrow aBSc \longrightarrow aBabcc \longrightarrow aaBbcc \longrightarrow aabbcc$.

Il linguaggio $L(G)$ è l'insieme $\{a^n b^n c^n \mid n \geq 1\}$.

2.6. Esercizio 06

Sia $G = (V, \Sigma, P, S)$ la grammatica con $V = \{S, A, B, C, D, E\}$, $\Sigma = \{a, b\}$ e P contenente le seguenti produzioni:
- $S \longrightarrow ABC$;
- $AB \longrightarrow aAD \mid bAE \mid \varepsilon$;
- $DC \longrightarrow BaC$;
- $EC \longrightarrow BbC$;
- $Da \longrightarrow aD$;
- $Db \longrightarrow bD$;
- $Ea \longrightarrow aE$;
- $Eb \longrightarrow bE$;
- $C \longrightarrow \varepsilon$;
- $aB \longrightarrow Ba$;
- $bB \longrightarrow bB$.

Dopo avere stabilito di che tipo é G, provate a derivare alcune stringhe. Riuscite a dire da quali stringhe é formato il linguaggio generato da G?

La grammatica G é di tipo 1.

Deriviamo qualche stringa:
- $S \longrightarrow ABC \overset{*}{\longrightarrow} \varepsilon$;
- $S \longrightarrow ABC \longrightarrow aADC \longrightarrow aABaC \overset{*}{\longrightarrow} aa$;

- $S \xrightarrow{*} aABaC \longrightarrow aaADaC \longrightarrow aaAaDC \longrightarrow aaAaBaC \longrightarrow aaABaaC \xrightarrow{*} aaaa$;
- $S \xrightarrow{*} aABaC \longrightarrow abAEaC \longrightarrow abAaEC \longrightarrow abAaBbC \longrightarrow abABabC \xrightarrow{*} abab$;
- $S \longrightarrow ABC \longrightarrow bAEC \longrightarrow bABbC \xrightarrow{*} bb$;
- $S \xrightarrow{*} bABbC \longrightarrow bbAEbC \longrightarrow bbAbEC \longrightarrow bbAbBbC \longrightarrow bbABbbC \xrightarrow{*} bbbb$;
- $S \xrightarrow{*} bABbC \longrightarrow baADbC \longrightarrow baAbDC \longrightarrow baAbBaC \longrightarrow baABbaC \xrightarrow{*} baba$.

Il linguaggio $L(G)$ è l'insieme $\left\{a^{2n} \cup b^{2n} \cup (ab)^{2n} \cup (ba)^{2n} \mid n \geq 0\right\}$.

2.7. Esercizio 07

Sia $G = (V, \Sigma, P, S)$ la grammatica con $V = \{S, A, B, C, X, Y, L, R\}$, $\Sigma = \{a\}$ e P contenente le seguenti produzioni:
- $S \longrightarrow LXR$;
- $LX \longrightarrow LYYA \mid aC$;
- $AX \longrightarrow YYA$;
- $AR \longrightarrow BR$;
- $YB \longrightarrow BX$;
- $LB \longrightarrow L$;
- $CX \longrightarrow aC$;
- $CR \longrightarrow \varepsilon$.

Riuscite a stabilire da quali stringhe é formato il linguaggio generato da G?

Deriviamo qualche stringa:
- $S \longrightarrow LXR \longrightarrow aCR \longrightarrow a$;
- $S \longrightarrow LXR \longrightarrow LYYAR \xrightarrow{*} LXXR \longrightarrow aCXR \longrightarrow aaCR \longrightarrow aa$;
- $S \longrightarrow LXR \xrightarrow{*} LXXR \longrightarrow LYYAXR \longrightarrow LYYYYAR \xrightarrow{*} LXXXXR \xrightarrow{*} aaaa$.
- $S \longrightarrow LXR \xrightarrow{*} LXXXXR \xrightarrow{*} LYYYYYYYYAR \xrightarrow{*} LXXXXXXXXR \xrightarrow{*} aaaaaaaa$.

Il linguaggio $L(G)$ è l'insieme $\left\{a^{2^n} \mid n \geq 0\right\}$.

2.8. Esercizio 08

Modificate la grammatica dell'esercizio 07 in modo da ottenere una grammatica di tipo 1 che generi lo stesso linguaggio.

Modificando la regola $LB \longrightarrow L$ in $LB \longrightarrow CRL$ la grammatica diventa di tipo 1.

2.9. Esercizio 09

Dimostrate che la grammatica $G = (\{A, B, S\}, \{a, b\}, P, S)$, con l'insieme delle produzioni P elencate sotto, genera il linguaggio $\left\{w \in \{a, b\}^* \mid \forall x \in \{a, b\}^* w \neq xx\right\}$:
- $S \longrightarrow AB \mid BA \mid A \mid B$
- $A \longrightarrow aAa \mid aAb \mid bAa \mid bAb \mid a$
- $B \longrightarrow aBa \mid aBb \mid bBa \mid bBb \mid b$

Consideriamo in primo luogo i “casi base”:

- $S \longrightarrow A \longrightarrow a$ va bene perché di lunghezza dispari;
- $S \longrightarrow B \longrightarrow b$ va bene perché di lunghezza dispari;
- $S \longrightarrow AB \overset{*}{\longrightarrow} ab$ va bene perché $a \neq b$;
- $S \longrightarrow BA \overset{*}{\longrightarrow} ba$ va bene perché $b \neq a$.

Consideriamo poi $S \longrightarrow A \mid B$:

$$\begin{aligned} S \longrightarrow A \longrightarrow aAa &\overset{*}{\longrightarrow} a^n A a^n \longrightarrow a^n a a^n; \\ aAa &\overset{*}{\longrightarrow} ab^n A b^n a \longrightarrow ab^n a b^n a; \\ aAa &\overset{*}{\longrightarrow} a\{a,b\}^n A\{a,b\}^n a \longrightarrow a\{a,b\}^n a\{a,b\}^n a; \\ aAb &\overset{*}{\longrightarrow} \dots . \end{aligned}$$

$$\begin{aligned} S \longrightarrow B \longrightarrow aBa &\overset{*}{\longrightarrow} a^n B a^n \longrightarrow a^n b a^n; \\ aBa &\overset{*}{\longrightarrow} ab^n B b^n a \longrightarrow ab^n b b^n a; \\ aBa &\overset{*}{\longrightarrow} a\{a,b\}^n B\{a,b\}^n a \longrightarrow a\{a,b\}^n b\{a,b\}^n a; \\ aBb &\overset{*}{\longrightarrow} \dots . \end{aligned}$$

Tutte le stringhe che vengono generate vanno bene perché sono di lunghezza dispari.

Consideriamo infine $S \longrightarrow AB \mid BA$ in due casi:

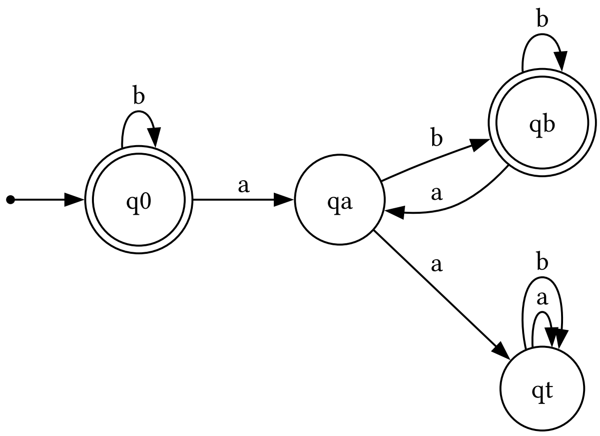
- se eseguiamo su A e B lo stesso numero di passi di derivazione abbiamo altri due casi:
  - usiamo regole con lo "stesso contesto", ma alla fine avremo un carattere diverso nella posizione dove sono presenti A e B;
  - usiamo regole con "diverso contesto", ma la prima regola che rispecchia questa casistica ha almeno un carattere diverso (oltre ad avere il carattere in A e B diverso alla fine della derivazione);
- se eseguiamo su A e B un numero diverso di passi di derivazione, abbiamo due punti di partenza:
  - partiamo da AB e indichiamo con n la lunghezza della stringa derivata da A e con k la lunghezza della stringa derivata da B, con $k > n$. Per ottenere due stringhe della stessa lunghezza devo rimuovere da k un numero $\frac{n-k}{2}$ di caratteri e appenderli a n, ottenendo due stringhe di lunghezza t. Prima dell'ultimo passo di derivazione di A la variabile A era in posizione $\frac{n-1}{2}$, mentre ora si trova in posizione $\frac{t-1}{2} - \frac{n-k}{4}$ perché prima mi devo prima posizionare nel "nuovo centro" e poi mi devo spostare di una posizione indietro ogni due caratteri che avevo aggiunto. Facciamo lo stesso ragionamento per trovare l'indice dell'ultima B di B. Le due posizioni trovate sono le stesse, ma prima dell'ultima derivazione in A si aveva una A e in B si aveva una B, che però generano rispettivamente a e b, quindi otteniamo due stringhe che sono sempre diverse.
  - partiamo da BA e facciamo lo stesso discorso, basta invertire l'ordine delle stringhe.

Abbiamo quindi dimostrato che $L(G) = \{w \in \{a,b\}^* \mid \forall x \in \{a,b\}^* w \neq xx\}$.

3. Lezione 03

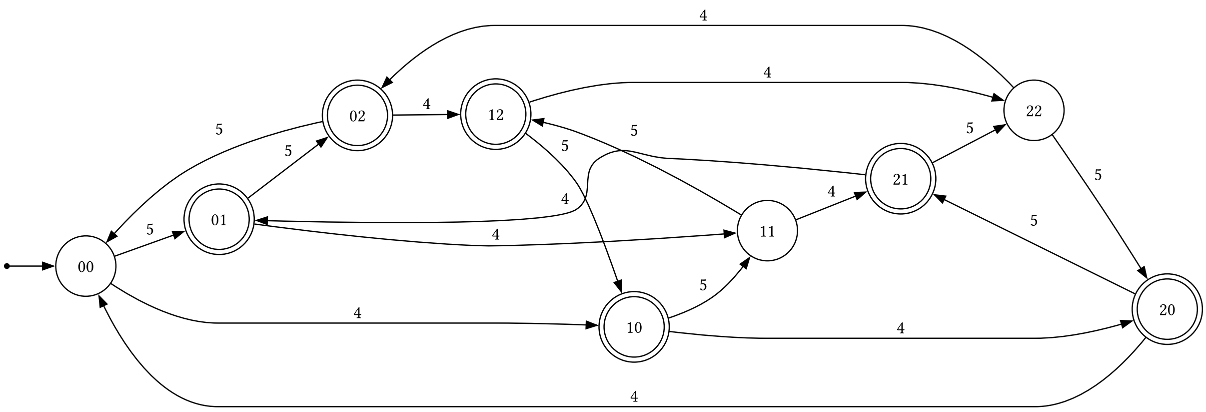
3.1. Esercizio 01

Costruite un automa a stati finiti che riconosca il linguaggio formato da tutte le stringhe sull'alfabeto $\{a, b\}$ nelle quali ogni a é seguita immediatamente da una b.



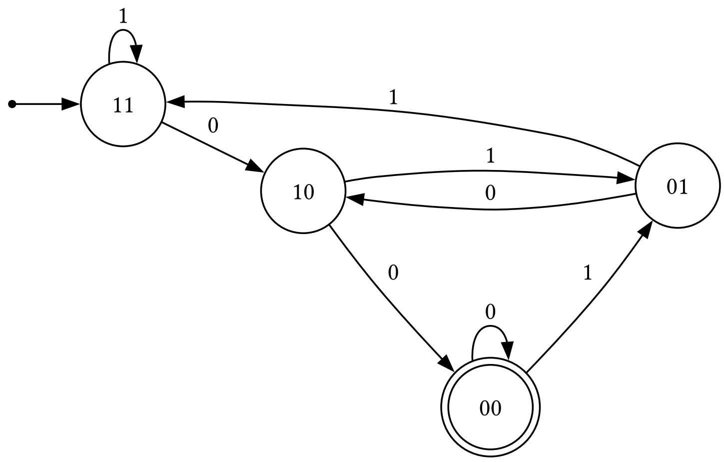
3.2. Esercizio 02

Costruite un automa a stati finiti che riconosca il linguaggio formato da tutte le stringhe sull'alfabeto $\{4, 5\}$ che, interpretate come numeri in base 10, rappresentano numeri interi che *non sono* divisibili per 3.



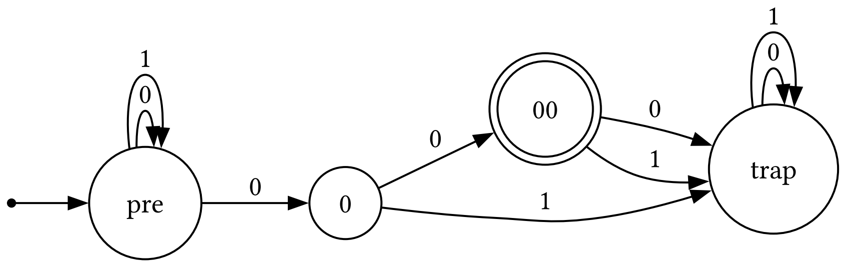
3.3. Esercizio 03

Costruite un automa a stati finiti deterministico che riconosca il linguaggio formato da tutte le stringhe sull'alfabeto $\{0, 1\}$ che, interpretate come numeri in notazione binaria, denotano multipli di 4.



Utilizzando il non determinismo si riesce a costruire un automa con meno stati? Generalizzate l'esercizio a multipli di $2k$, dove $k > 0$ é un intero fissato.

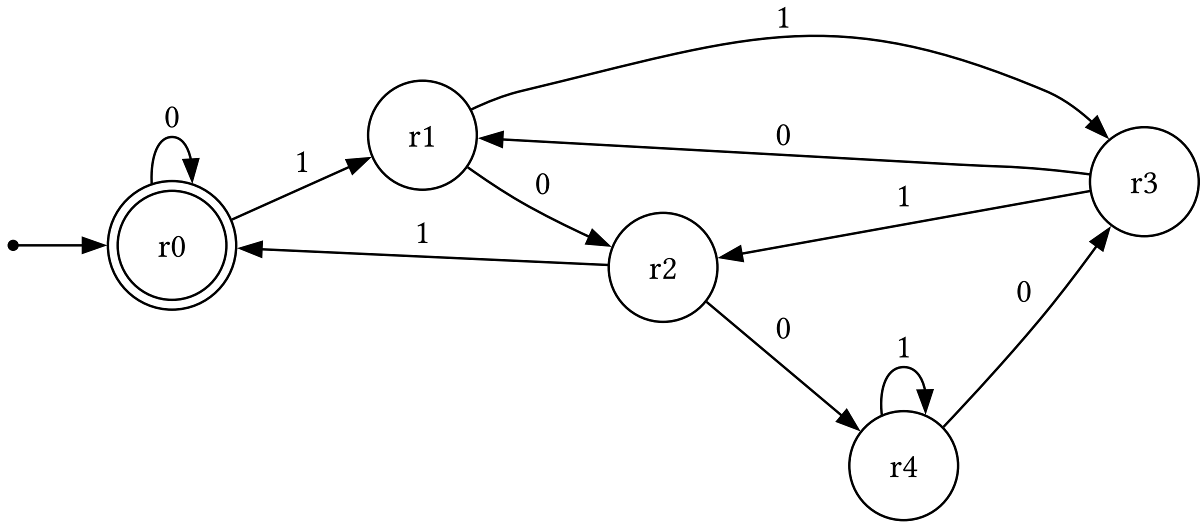
Utilizzando il non determinismo si usano ancora 4 stati.



Generalizzando a multipli di $2k$, con $k > 0$, abbiamo:
- per il DFA 2^k stati;
- per il NFA $k + 2$ stati.

3.4. Esercizio 04

Costruite un automa a stati finiti che riconosca il linguaggio formato da tutte le stringhe sull'alfabeto $\{0, 1\}$ che, interpretate come numeri in notazione binaria, rappresentano multipli di 5.



0
1
r0
r1
1
0
1
0
r2
r3
1
0
0
1
r4

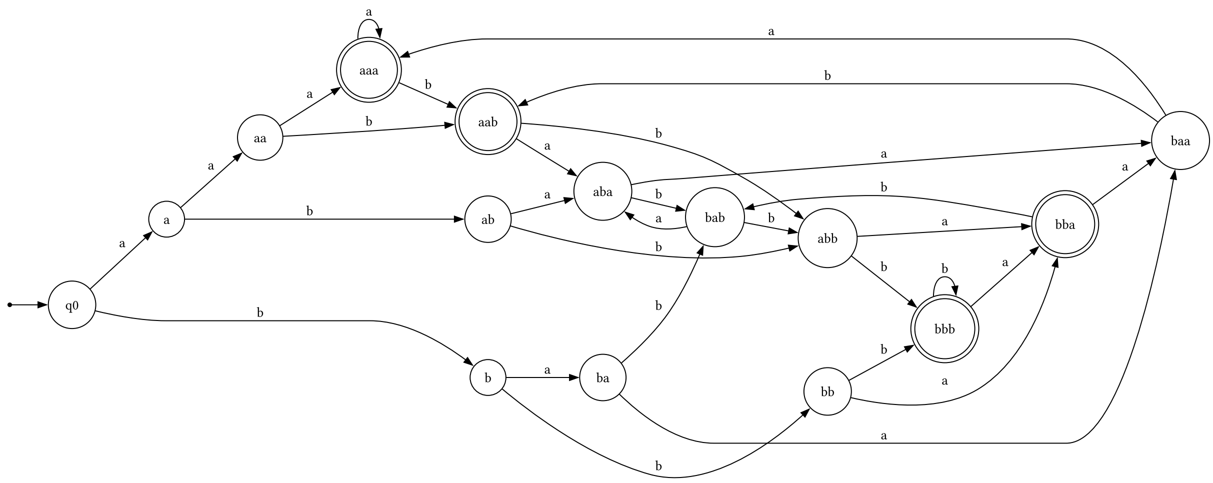
4. Lezione 04

4.1. Esercizio 01

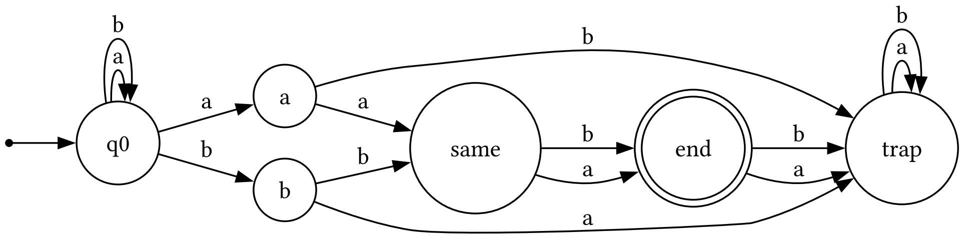
Considerate il linguaggio

$$L = \{w \in \{a, b\}^* \mid \text{il penultimo e il terzultimo simbolo di } w \text{ sono uguali}\}.$$

Costruite un automa a stati finiti deterministico che accetta L.



Costruite un automa a stati finiti non deterministico che accetta L.



Dimostrate che per il linguaggio L tutte le stringhe di lunghezza 3 sono distinguibili tra loro.

	aaa	aab	aba	abb	baa	bab	bba	bbb
aaa	-	a	ε	ε	ε	ε	a	aa
aab	-	-	ε	ε	ε	ε	bb	b
aba	-	-	-	b	a	aa	ε	ε

$\boldsymbol{abb}$	-	-	-	-	aa	b	ε	ε
$\boldsymbol{baa}$	-	-	-	-	-	a	ε	ε
$\boldsymbol{bab}$	-	-	-	-	-	-	ε	ε
$\boldsymbol{bba}$	-	-	-	-	-	-	-	a
$\boldsymbol{bbb}$	-	-	-	-	-	-	-	-

Dimostrate che per il linguaggio L la parola vuota é distinguibile da tutte le stringhe di lunghezza 3.

	$\boldsymbol{aaa}$	$\boldsymbol{aab}$	$\boldsymbol{aba}$	$\boldsymbol{abb}$	$\boldsymbol{baa}$	$\boldsymbol{bab}$	$\boldsymbol{bba}$	$\boldsymbol{bbb}$
ε	ε	ε	ab	a	a	ba	ε	ε

Utilizzando i risultati precedenti, ricavate un limite inferiore per il numero di stati di ogni automa deterministico che accetta L.

L'insieme $X = \left\{w \in \{a, b\}^+ \mid |w| = 3\right\}$ é un insieme di parole tutte distinguibili tra loro rispetto al linguaggio L, come dimostrato nei punti precedenti, quindi ogni DFA per L deve avere almeno $|X|$ stati, ovvero almeno 8 stati.

4.2. Esercizio 02

Costruite un insieme di stringhe distinguibili tra loro per ognuno dei seguenti linguaggi:
- $L_1 = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$,
- $L_2 = \{a^n b^n \mid n \geq 0\}$,
- $L_3 = \{ww^R \mid w \in \{a, b\}^*\}$ dove, per ogni stringa w, w^R indica la stringa w scritta al contrario.

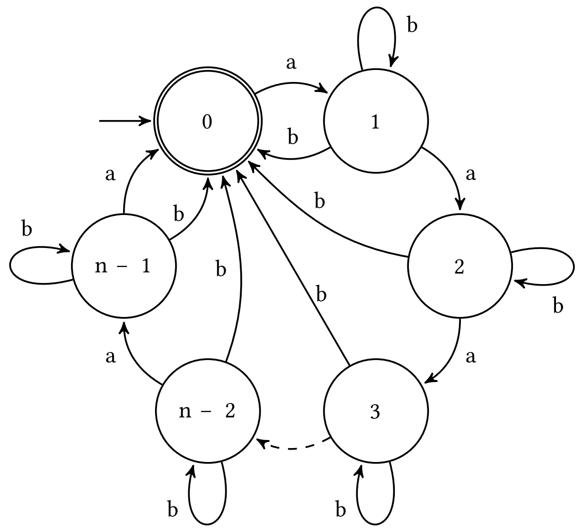
Costruiamo i seguenti insiemi:
- $X_1 = \left\{a^i \mid i \geq 1\right\}$ di cardinalità infinita;
- $X_2 = \left\{a^i \mid i \geq 1\right\}$ di cardinalità infinita;
- $X_3 = \left\{(ab)^i \mid i \geq 1\right\}$ di cardinalità infinita.

Per alcuni di questi linguaggi riuscite ad ottenere insiemi di stringhe distinguibili di cardinalità infinita? Cosa significa ciò?

I linguaggi che hanno insiemi di stringhe distinguibili di cardinalità infinita sono linguaggi non di tipo 3.

4.3. Esercizio 03

Considerate l'automa di Meyer e Fischer M_n presentato nella Lezione 4 (caso peggiore della costruzione per sottoinsiemi) e mostrato nella seguente figura:



Descrivete a parole la proprietà che deve soddisfare una stringa per essere accettata da M_n. Riuscite a costruire un automa non deterministico, diverso da M_n, per lo stesso linguaggio, basandovi su tale proprietà?

Non lo so fare.

5. Lezione 05

5.1. Esercizio 01

Considerate il linguaggio

$$\text{DOUBLE}_k = \left\{ ww \mid w \in \{a,b\}^k \right\},$$

dove $k > 0$ é un numero intero fissato.

É abbastanza facile trovare un fooling set di cardinalità 2^k per questo linguaggio. Riuscite a trovare un fooling set o un extended fooling set di cardinalità maggiore?

Considero l'insieme

$$P = \left\{ (x,x) \mid x \in \{a,b\}^k \right\}.$$

Questo é un extended fooling set per DOUBLE_k perché:
1. $xx \in \text{DOUBLE}_k$;
2. $xy \notin \text{DOUBLE}_k$.

La cardinalità di questo insieme é 2^k, non penso di riuscire a fare meglio.

5.2. Esercizio 02

Considerate il linguaggio

$$\text{PAL}_k = \left\{ w \in \{a,b\}^k \mid w = w^R \right\},$$

dove k é un intero fissato.

Qual é l'extended fooling set per PAL_k di cardinalità maggiore che riuscite a trovare?

Considero l'insieme

$$P = \left\{ (x, x^R) \mid x \in \{a,b\}^k \right\}.$$

Questo é un extended fooling set per PAL_k perché:
1. $xx^R \in \text{PAL}_k$;
2. $xy^R \notin \text{PAL}_k$.

La cardinalità di questo insieme é 2^k.

5.3. Esercizio 03

Considerate il linguaggio

$$K_k = \left\{ w \mid w = x_1 \cdot \ldots \cdot x_m \cdot x \mid m > 0, x_1, \ldots, x_m, x \in \{a,b\}^k, \exists i \in [1,m] \mid x_i = x \right\},$$

dov k é un intero fissato. Si può osservare che ogni stringa w di questo linguaggio é la concatenazione di blocchi di lunghezza k, in cui l'ultimo blocco coincide con uno dei blocchi precedenti.

Riuscite a costruire un (extended) fooling set di cardinalità 2^k o maggiore per il linguaggio K_k?

Considero l'insieme

$$P = \left\{(x^m, x) \mid x \in \{a, b\}^k \land m > 0\right\}.$$

Questo é un extended fooling set per K_k perché:
1. $x^m x \in K_k$;
2. $x^m y \notin K_k$.

La cardinalità di questo insieme é 2^k.

Quale é l'informazione principale che un automa non deterministico può scegliere di ricordare nel proprio controllo a stati finiti durante la lettura di una stringa per riuscire a riconoscere K_k?

Un NFA dovrebbe formare prima l'albero di tutte le possibili stringhe di lunghezza k, inserendo la scommessa nei nodi ad altezza $k - 1$. Questa scommessa fa tornare indietro alla radice, e si "vince" la scommessa quando si finisce nel nodo ad altezza k, ovvero la stringa di lunghezza k letta ora é quella che sarà presente anche alla fine. Il numero di stati per questa parte é $2^{k+1} - 1$.

Il controllo viene poi fatto con $k2^k$ stati, dove solo l'ultimo é finale. Vanno aggiunti $(k - 1)2^k$ stati che cancellano gruppi di lunghezza k prima dell'ultimo gruppo.

Il numero totale di stati é quindi $k2^{k+1} + 2^k - 1$.

Supponete di costruire un automa deterministico per riconoscere K_k. Cosa ha necessità di ricordare l'automa nel proprio controllo a stati finiti mentre legge la stringa in input?

Un DFA deve ricordarsi le sequenze lunghe k che ha trovato nella stringa.

Utilizzando il concetto di distinguibilità, dimostrate che ogni automa deterministico che riconosce K_k deve avere almeno 2^{2^k} stati.

Costruisco l'insieme

$$X = \left\{S \subseteq \{a, b\}^k\right\} = \mathbb{P}\left(\{a, b\}^k\right)$$

insieme delle parti di $\{a, b\}^k$.

Questo é un insieme di parole distinguibili tra loro perché

$$\forall X_1, X_2 \in X \quad \exists x \in X_1 - X_2 \quad \mid \quad \prod_{x_1 \in X_1} x_1 \cdot x \in K_k \land \prod_{x_2 \in X_2} x_2 \cdot x \notin K_k.$$

La cardinalità di questo insieme é $2^{|X|} = 2^{2^k}$.

5.4. Esercizio 04

Considerate il linguaggio

$$J_k = \left\{w \mid w = x \cdot x_1 \cdot \ldots \cdot x_m \mid m > 0, x_1, \ldots, x_m, x \in \{a, b\}^k, \exists i \in [1, m] \mid x_i = x\right\},$$

dove k é un intero fissato. Si può osservare che ogni stringa w di questo linguaggio é la concatenazione di blocchi di lunghezza k, in cui il primo blocco coincide con uno dei blocchi successivi; ogni stringa di J_k si ottiene "rovesciando" una stringa del linguaggio K_n dell'esercizio 3.

Supponete di costruire automi a stati finiti per J_k. Valgono ancora gli stessi limiti inferiori ottenuti per K_n o si riescono a costruire automi più piccoli? Rispondete sia nel caso di automi deterministici sia in quello di automi non deterministici.

Un DFA deve prima costruire l'albero di altezza k che contiene tutte le possibili stringhe di lunghezza k e poi, dopo ogni foglia, deve costruire un ciclo di k stati che riconosce la sequenza definita dal cammino a quella foglia. Vanno aggiunti $(k-1)2^k$ stati che cancellino le sequenze lunghe k che non sono uguali alla prima letta.

Il numero totale di stati é $k2^{k+1} + 2^k - 1$.

Un NFA deve fare la stessa cosa del DFA ma mettendo la scommessa nelle foglie.

Il numero totale di stati é ancora $k2^{k+1} + 2^k - 1$.

5.5. Esercizio 05

Ispirandovi all'esercizio 3, fornite limiti inferiori per il numero di stati degli automi che riconoscono il seguente linguaggio:

$$E_k = \left\{ w \mid w = x_1 \cdot \ldots \cdot x_m \mid m > 0, x_1, \ldots, x_m \in \{a, b\}^k, \exists i, j \in [1, m] \mid x_i = x_j \right\},$$

dove k é un intero fissato. Considerate sia il caso deterministico che quello non deterministico.

Un NFA dovrebbe formare prima l'albero di tutte le possibili stringhe di lunghezza k, inserendo la scommessa nei nodi ad altezza $k-1$. Questa scommessa fa tornare indietro alla radice, e si "vince" la scommessa quando si finisce nel nodo ad altezza k, ovvero la stringa di lunghezza k letta ora é quella che sarà presente successivamente. Il numero di stati per questa parte é $2^{k+1} - 1$.

Il controllo viene poi fatto con $k2^k$ stati, dove solo l'ultimo é finale. Vanno aggiunti $(k-1)2^k$ stati che cancellano gruppi di lunghezza k che sono sono uguali alla sequenza scelta.

Il numero totale di stati é quindi $k2^{k+1} + 2^k - 1$.

Un DFA deve ricordarsi le sequenze lunghe k che ha trovato nella stringa. Costruendo l'insieme X dell'esercizio 3 si può concludere che ogni DFA deve avere almeno 2^{2^k} stati.