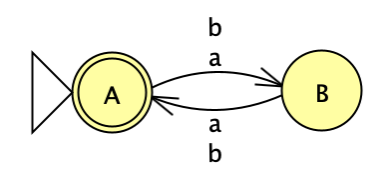
Esercizio 1

Se due linguaggi sono regolari allora la loro unione è regolare, basta prendere i due DFA che riconoscono i linguaggi e creare un nuovo stato iniziale che si collega agli stati iniziali dei due DFA tramite una ϵ-transizione quindi la risposta è **VERO**.

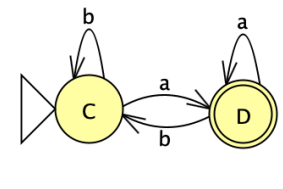
Nota ~~inutile ai fini dell'esercizio~~:

Se prendessimo ad esempio:

- $\mathcal{L}_1 = \{w \in \{a,b\}^* | \ w \ ha \ una \ lunghezza \ pari\}$
- $\mathcal{L}_2 = \{w \in \{a,b\}^* | \ w \ termina \ con \ a\}$

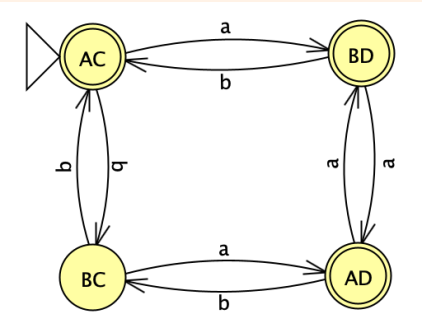


DFA per $\mathcal{L}_1$

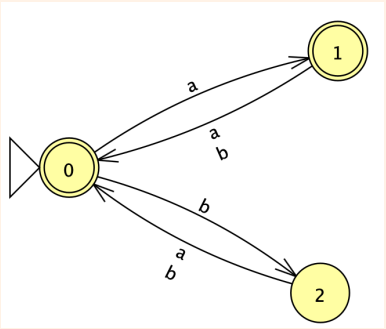


DFA per $\mathcal{L}_2$

L'unione di $\mathcal{L}_1$ con $\mathcal{L}_2$ sta a significare che la parola su $\{a,b\}$ deve essere pari oppure finire con la lettera a. Per costruire l'automa che verifichi l'unione potremo leggere la parola contemporaneamente in entrambi i DFA e se al termine della parola mi trovo in uno stato finale di uno dei due DFA allora la parola appartiene all'unione.
Possiamo costruire il DFA finale quindi facendo il prodotto cartesiano degli stati, quello iniziale sarà AC mentre quelli finali saranno AC, AD e BD ottenendo il seguente DFA, confermando che l'unione è regolare.



Andando ad 'occhio' si poteva comunque costruire un DFA che riconosceva $\mathcal{L}_1 \cup \mathcal{L}_2$:



Esercizio 2

Sulle slide o dispensa studenti.

Esercizio 3

Per la costruzione di un DFA da un NFA partiamo dallo stato iniziale A e calcoliamo la sua ϵ-chiusura:

$$closure(A) = \{A\ , B,\ C,\ E\} = T_0$$

Controllando le transazioni da T_0:

- a-transizione: $closure(D,\ E) = \{D,\ E\} = T_1$
- b-transizione: $closure(A,\ E) = \{A\ , B,\ C,\ E\} = T_0$

T_1:

- a-transizione: $closure(E) = \{E\} = T_2$
- b-transizione: $closure(A,\ B) = \{A\ , B,\ C,\ E\} = T_0$

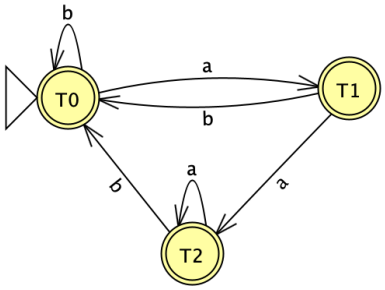
T_2:

- a-transizione: $closure(E) = \{E\} = T_2$
- b-transizione: $closure(A) = \{A\ , B,\ C,\ E\} = T_0$

La funzione di transazione del DFA sarà:

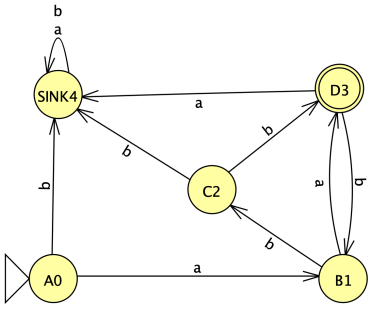
	a	**b**
T_0	T_1	T_0
T_1	T_2	T_0
T_2	T_2	T_0

Quindi se Q è lo stato iniziale del DFA allora: $Q[ab] = T_0 = A, B, C, E$



Esercizio 4

D ha una funzione di transizione parziale. Per renderla totale, aggiungiamo uno stato 'sink' che assorbe tutte le transizioni non definite:



	a	**b**
A	B	$Sink$
B	D	C
C	D	$Sink$
D	$Sink$	B
$Sink$	$Sink$	$Sink$

Funzione di Transizione

Per minimizzare l'automa, analizziamo gli stati di equivalenza usando il *partition refinment*:

0-equivalenti	$\{A,\ B,\ C,\ Sink\}$	$\{D\}$			
1-equivalenti	$\{A,\ Sink\}$	$\{B,\ C\}$	$\{D\}$		
2-equivalenti	$\{A\}$	$\{Sink\}$	$\{B\}$	$\{C\}$	$\{D\}$

Spiegazione

- **Stati 0-equivalenti**: si distinguono gli stati finali e non finali
- **Stati 1-equivalenti**:
  - $\{A, Sink\}$ transitano entrambi verso lo stesso insieme $\{A,\ B,\ C,\ Sink\}$ sia con *a-transazione* che con *b-transazione*
  - $\{B, C\}$ transitano entrambi verso lo stesso insieme $\{D\}$ con *a-transazione* e in $\{A,\ B,\ C,\ Sink\}$ con *b-transazione*.
- **Stati 2-equivalenti**:
  - A e $Sink$ vanno divisi perché la *a-transazione* li porta a due insiemi diversi (A va a $\{B, C\}$ mentre Sink va a $\{A, Sink\}$)
  - in modo analogo anche B e C vanno separati perché la *b-transazione* li porta a due insiemi diversi

Il DFA è già minimo poiché nessun raggruppamento è possibile.
Quindi lo stato a cui corrisponde $P[abab]$ è B.

Esercizio 5

Iniziamo calcolando i $first$ e i $follow$.

First:

- B può derivare ϵ. Pertanto, $first(B) = \{\epsilon\}$
- A può derivare $BcBaA$ quindi aggiungiamo i $first(B)$ ai $first(A)$, $first(B)$ contiene ϵ quindi consideriamo $first(cBaA)$ e aggiungiamo c ai $first(A)$, A può anche derivare ϵ quindi aggiungiamola ai $first(A)$
- S può derivare AaB quindi aggiungiamo i $first(A)$ ai $first(S)$, essendoci ϵ nei $first(A)$ allora dobbiamo aggiungere $first(aB)$ ai $first(S)$, infine S può derivare b che sarà da aggiungere ai suoi $first$

	first	follow
S	b a c	\$
A	ϵ c	a
B	ϵ	c a \$

Follow:

- $follow(S)$: S è lo start symbol, quindi $follow(S) = \{\$\}$, S non compare in nessun body quindi ci possiamo fermare
- $follow(A)$: nella produzione $S \to AaB$, A è seguito da 'aB', quindi $first(aB) = \{a\}$ è in $follow(A)$
- $follow(B)$:
  - nella produzione $S \to AaB$, B è l'ultimo simbolo della produzione dobbiamo aggiungere i $follow(S)$ ai $follow(B)$
  - nella produzione $A \to BcBaA$, B è seguito da 'cBaA', $first(cBaA) = \{c\}$ è in $follow(B)$

Le regole per popolare la tabella di parsing LL(1) sono:

- per ogni produzione $A \to \alpha$ nella grammatica:
  - per ogni terminale 'b' in $first(\alpha)$ (escludendo ϵ) bisogna aggiungere $A \to \alpha$ alla voce della tabella $M[A, b]$.
  - se ϵ è in $first(\alpha)$ allora per ogni terminale 'x' in $follow(A)$ bisogna aggiungere $A \to \alpha$ alla voce della tabella $M[A, x]$.
- imposta tutte le voci vuote nella tabella a 'errore'

La tabella di parsing $LL(1)$ è la seguente e la risposta è evidenziata:

	a	**b**	**c**	**$**
S	$S \to AaB$	$S \to b$	$S \to AaB$	
A	$A \to \epsilon$		$A \to BcBaA$	
B	$B \to \epsilon$		$B \to \epsilon$	$B \to \epsilon$

Esercizio 6

$$\mathcal{L} = \{ww | w \in \mathcal{L}((a|b)^*)\}$$

Questo linguaggio rappresenta l'insieme delle stringhe formate da due copie consecutive di una parola del linguaggio denotato da $(a|b)^*$.
Per dimostrare che non è regolare utilizzando il pumping lemma:

1. supponiamo che $\mathcal{L}$ sia regolare
2. prendiamo una *pumping constant* p arbitraria
3. scegliamo una parola opportuna $z = a^p\, b^p\, a^p\, b^p$
4. guardiamo a tutte le possibili decomposizioni di z in uvw con:
   (a) $|uv| \leq p$
   (b) $|v| \geq 1$

   Quindi abbiamo che $u = a^\alpha$, $v = a^\beta$ e $w = a^{p-\alpha-\beta}\, b^p\, a^p\, b^p$
5. scelgo una i opportuna tale che $uv^iw \notin \mathcal{L}$

   $$a^\alpha\, a^{i\cdot\beta}\, a^{p-\alpha-\beta}\, b^p\, a^p\, b^p = a^{p+\beta(i-1)}\, b^p\, a^p\, b^p$$

   Sapendo che $\beta \neq 0$ perché $|v| \geq 1$ ci basta scegliere $i \neq 1$, quindi scegliamo 2 e otteniamo

   $$a^{p+\beta}\, b^p\, a^p\, b^p \notin \mathcal{L}$$

Andando così in contraddizione con l'enunciato del pumping lemma, quindi $\mathcal{L}$ non è regolare.

Esercizio 7

Un metodo per risolvere questo esercizio sarebbe quello di fare la *costruzione di Thompson* per ottenere un NFA, dopo di che fare il *subset construction* per tradurlo in un DFA ed infine il *partition refinment* per minimizzarlo. Solo che all'esame, non si ha tutto questo tempo, quindi possiamo analizzare la regex:

- b^* denota o la parola vuota oppure una serie arbitraria di b
- $b^*\, a\, (\epsilon \mid a \mid b)^*$ denota una serie arbitraria di b (anche zero) seguita da una a seguita da $(a|b)^*$

Si riescie a notare che possiamo formare qualsiasi parola su $\{a, b\}$, quindi l'automa caratteristico può essere il seguente:

Che è già minimo. Quindi la risposta è che $\mathcal{D}$ ha uno stato che è anche finale.

Esercizi 8, 9 e 10

Guardare **5.4 Proprietà di chiusura nei linguaggi regolari** della dispensa studenti.