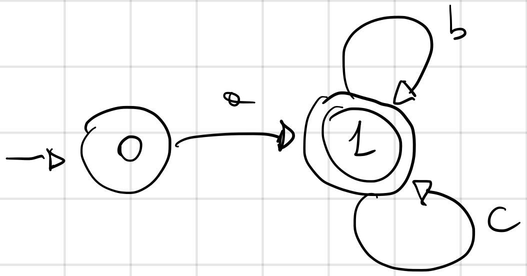
$a(b|c)^*$ find a minimal DFA

CANONICAL STRATEGY

1) $a(b|c)^* \longrightarrow$ NFA Thompson's algorithm
2) NFA $\rightarrow$ DFA Subset construction algorithm
3) DFA $\rightarrow$ minimal DFA Partition Refinement Alg.

To solve this particular problem we can skip 1) and 2) easily and give directly a DFA

$$\mathcal{L}(a(b|c)^*) = \{ax \mid x \in \{b,c\}^*\}$$



$\Pi^{(1)} = \{\{0\}, \{1\}\}$

This partition cannot be refined so this automaton is also minimal for the language.

Ex: Given r_1, r_2 regexps, are they equivalent?

$r_1 \equiv r_2$ iff $\mathcal{L}(r_1) = \mathcal{L}(r_2)$

Strategy:

1) $r_1 \longrightarrow NFA_1$

$r_2 \longrightarrow NFA_2$

2) $NFA_1 \longrightarrow DFA_1$

$NFA_2 \longrightarrow DFA_2$

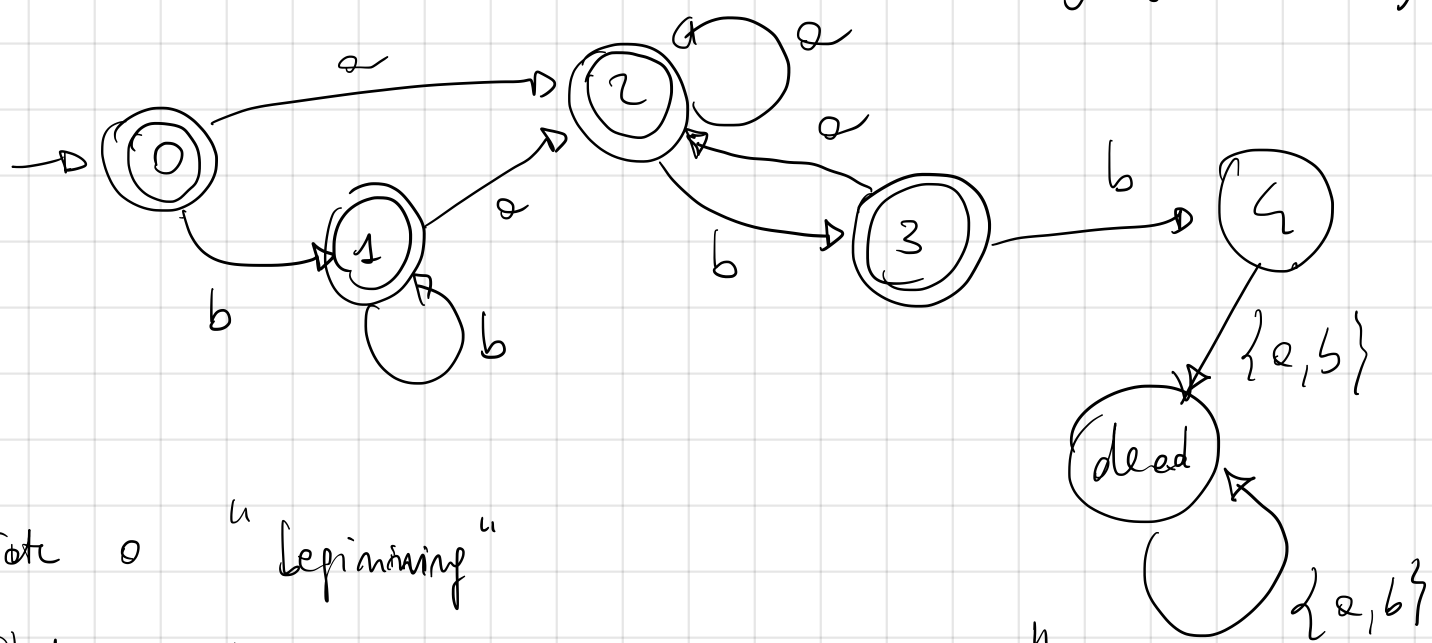
3) $DFA_1 \longrightarrow DFA_{1\_min}$

$DFA_2 \longrightarrow DFA_{2\_min}$

isomorphic

4) if ($DFA_{1\_min} \approx DFA_{2\_min}$)
then return YES
else return NO

$\mathcal{L} = \{ x \in \{a,b\}^* \mid x \neq y\,abb\,z \text{ for any } y,z \in \{a,b\}^* \}$



State 0 "beginning"

State 1 "I have not seen an a yet"

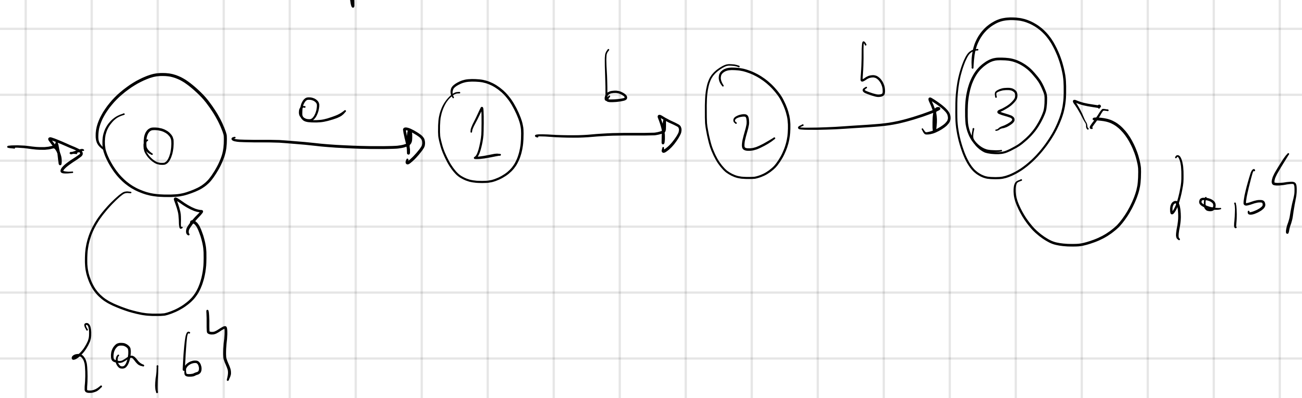
State 2 "Last character was a"

State 3 "Last sequence was ab"

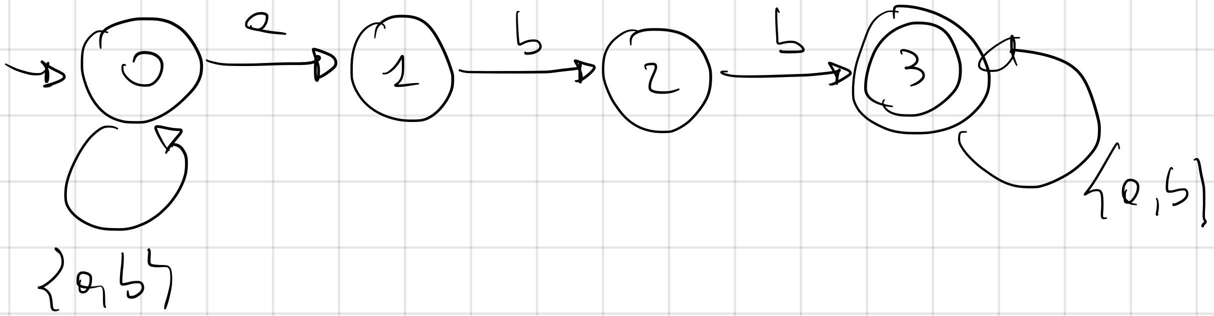
State 4 "The string contains abb"

$\mathcal{L}' = \{ x \in \{a,b\}^* \mid x = y\,abb\,z \text{ for some } y, z \in \{a,b\}^* \}$

$\mathcal{L}'$ is the complement of $\mathcal{L}$. NFA for $\mathcal{L}'$:

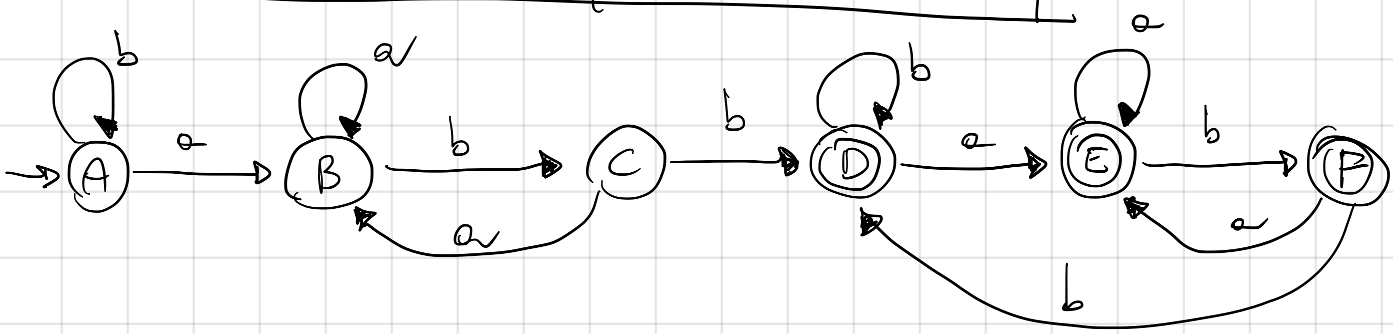


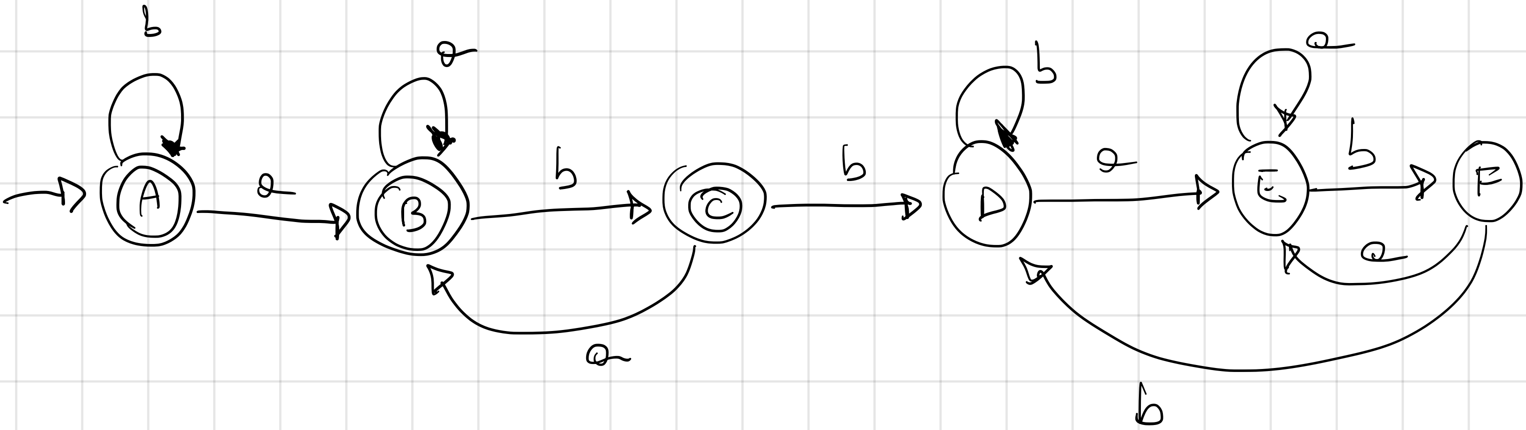
Transform



into a DFA

	a	b
A={0}	{0,1}=B	{0}=A
B={0,1}	{0,1}=B	{0,2}=C
C={0,2}	{0,1}=B	{0,3}=D
D={0,3}	{0,1,3}=E	{0,3}=D
E={0,1,3}	{0,1,3}=E	{0,2,3}=F
F={0,2,3}	{0,1,3}=E	{0,3}=D





Theorem: If $\mathcal{L}$ is a regular language then $\mathcal{L}^c = \Sigma^* - \mathcal{L}$ is a regular language.

Proof: 1) $\mathcal{L}$ is regular then there is a regular expression $r_{\mathcal{L}}$ such that $L(r_{\mathcal{L}}) = \mathcal{L}$

the language denoted by $r_{\mathcal{L}}$.

2) $r_{\mathcal{L}} \longrightarrow NFA_{\mathcal{L}}$ 3) $NFA_{\mathcal{L}} \rightarrow DFA_{\mathcal{L}}$

4) if $DFA_{\mathcal{L}}$ is blocking, then add the dead state

5) $DFA'_{\mathcal{L}}$ is $DFA_{\mathcal{L}}$ s.t. the final and non-final states are exchanged

6) Thus, $DFA'_{\mathcal{L}}$ acceps $\mathcal{L}^c$

7) By Kleene theorem, since there is a DFA accepting $\mathcal{L}^c$ then $\mathcal{L}^c$ is REGULAR $\square$

Kleene Theorem

L is regular

iff $\exists$ regexp r s.t. $\mathcal{L}(r) = L$

iff $\exists$ NFA accepting L

iff $\exists$ DFA accepting L

$\mathcal{L}_1$ regular
$\mathcal{L}_2$ regular

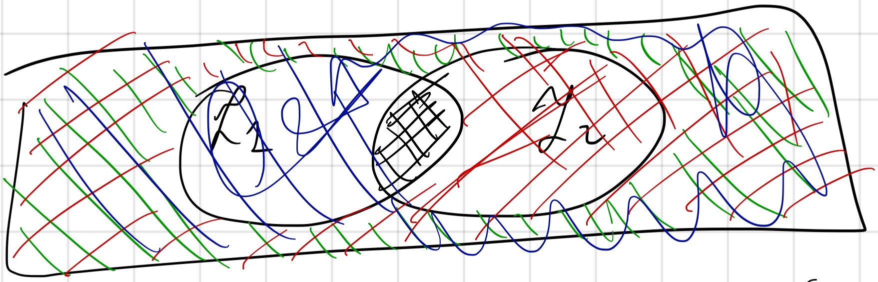
? $\mathcal{L}_1 \cap \mathcal{L}_2$ is regular?

YES

In Lex ^ operator

if r is a regexp ^r is a regexp

$$\mathcal{L}(\hat{}r) = \Sigma^* - \mathcal{L}(r)$$



$(\mathcal{L}_1 \cup \mathcal{L}_2)^c$

$\mathcal{L}_1^c$ $\mathcal{L}_2^c$ $(\mathcal{L}_1^c \cup \mathcal{L}_2^c)^c$

Proof: $\mathcal{L}_1 \rightarrow r_1$ regexp because $\mathcal{L}_1$ and $\mathcal{L}_2$ are regular

$\mathcal{L}_2 \rightarrow r_2$ regexp

Then ^(^r_1 | ^r_2) is a regular expression denoting $(\mathcal{L}_1^c \cup \mathcal{L}_2^c)^c = \mathcal{L}_1 \cap \mathcal{L}_2$.

Thus, $\mathcal{L}_1 \cap \mathcal{L}_2$ is regular $\square$

Another way:

$$\mathcal{L}_1 \to r_1 \to NFA_1 \to DFA_1 = \langle S_1, \Sigma, s_0^1, \delta_1, F_1 \rangle$$

$$\mathcal{L}_2 \to r_2 \to NFA_2 \to DFA_2 = \langle S_2, \Sigma, s_0^2, \delta_2, F_2 \rangle$$

Create an automaton that accepts $\mathcal{L}_1 \cap \mathcal{L}_2$

$$\langle S_1 \times S_2, \Sigma, (s_0^1, s_0^2), \delta, F_1 \times F_2 \rangle$$

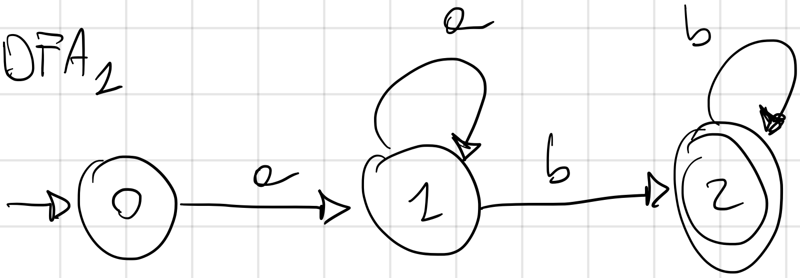
where δ is defined s.t. $\forall s \in S_1, t \in S_2, c \in \Sigma$

if $\Big(\delta_1(s, c) = s'$ and $\delta_2(t, c) = t' \Big)$ then $\delta((s,t), c) = (s', t')$

Example $\to$

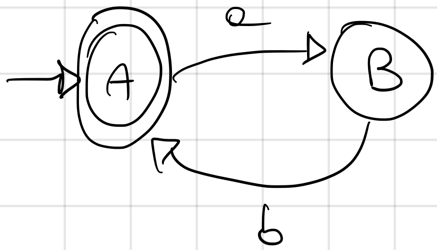
$\mathcal{L}_1 = \{a^n b^m \mid n > 0, m > 0\}$

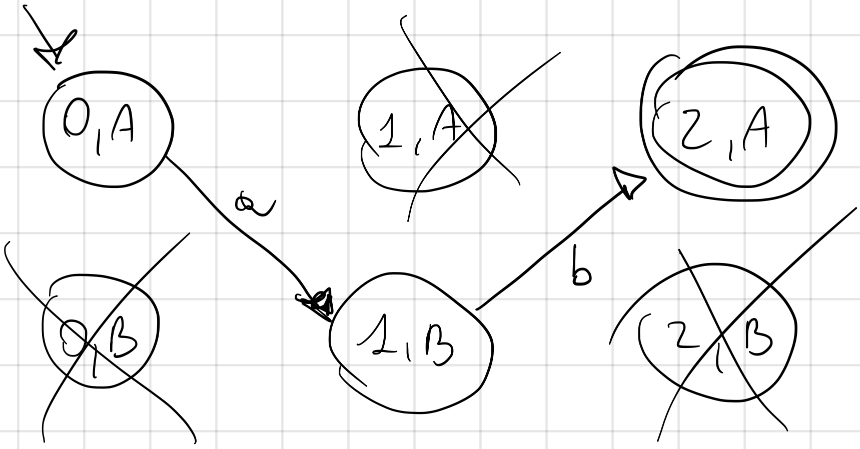
DFA$_1$



$\mathcal{L}_2 = \{(ab)^n \mid n \geq 0\}$

DFA$_2$





$$\frac{s \xrightarrow{c} s' \text{ and } t \xrightarrow{c} t'}{(s,t) \xrightarrow{c} (s',t')}$$

This automata accepts $\{ab\} = \mathcal{L}_1 \cap \mathcal{L}_2$