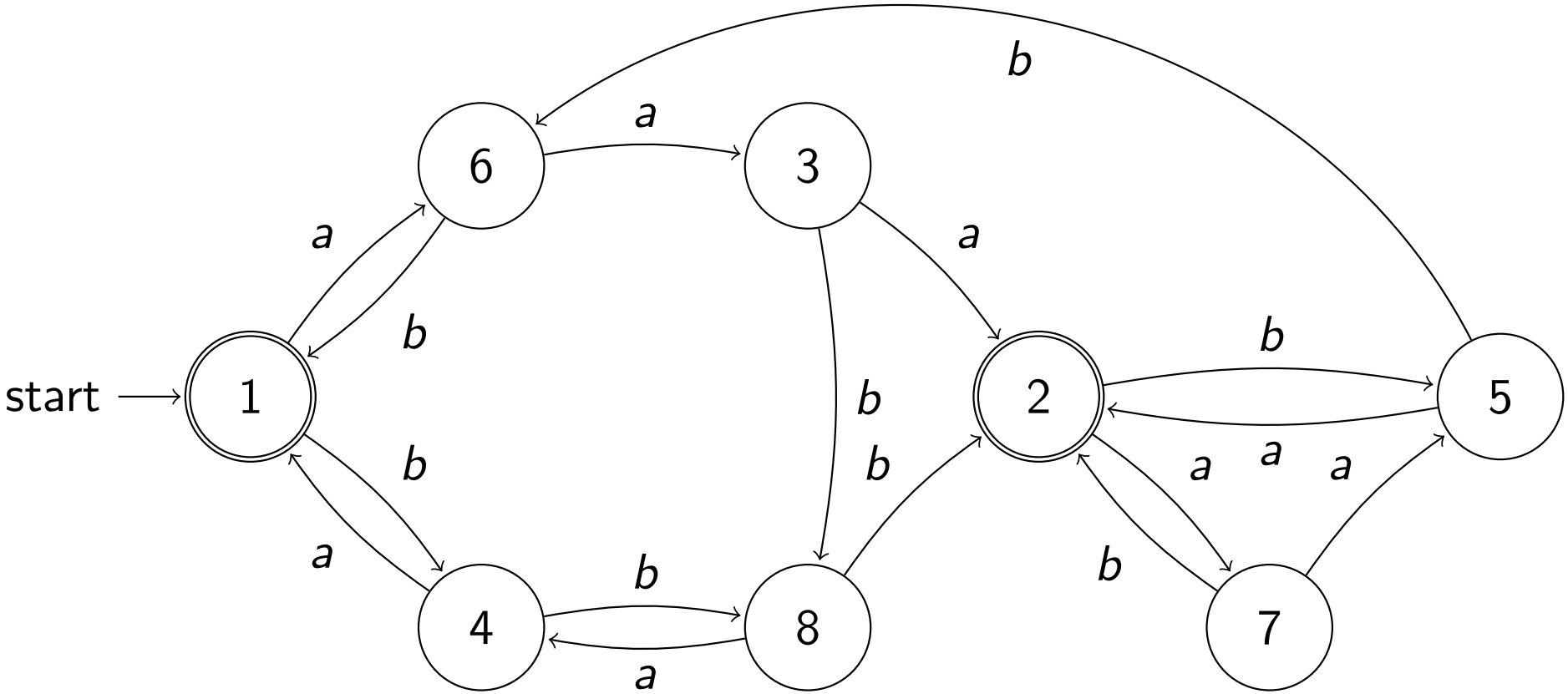
CS 302 Recitation 5

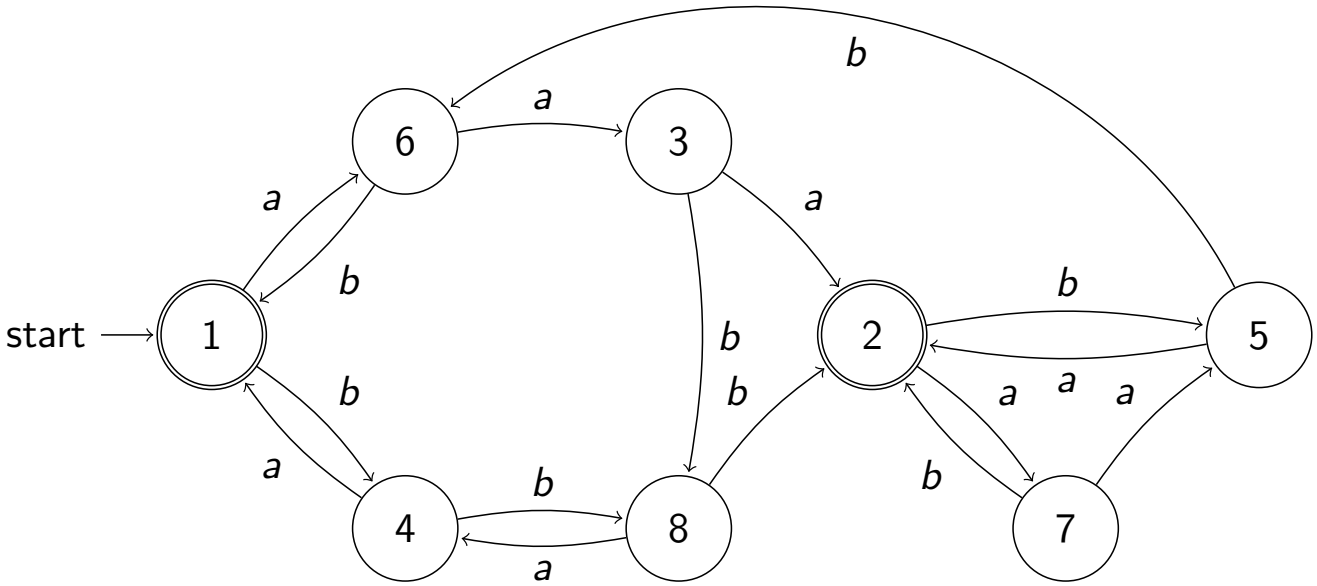
November 9, 2020

Problem 1

Minimize given DFA.

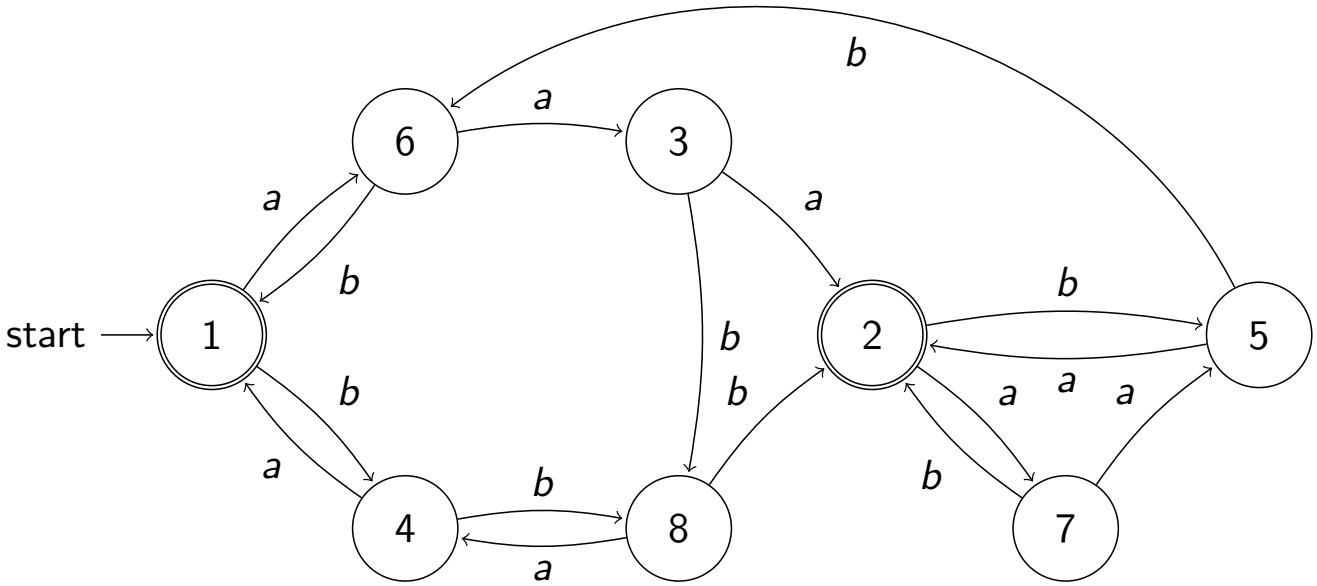


Problem 1 - Step 0



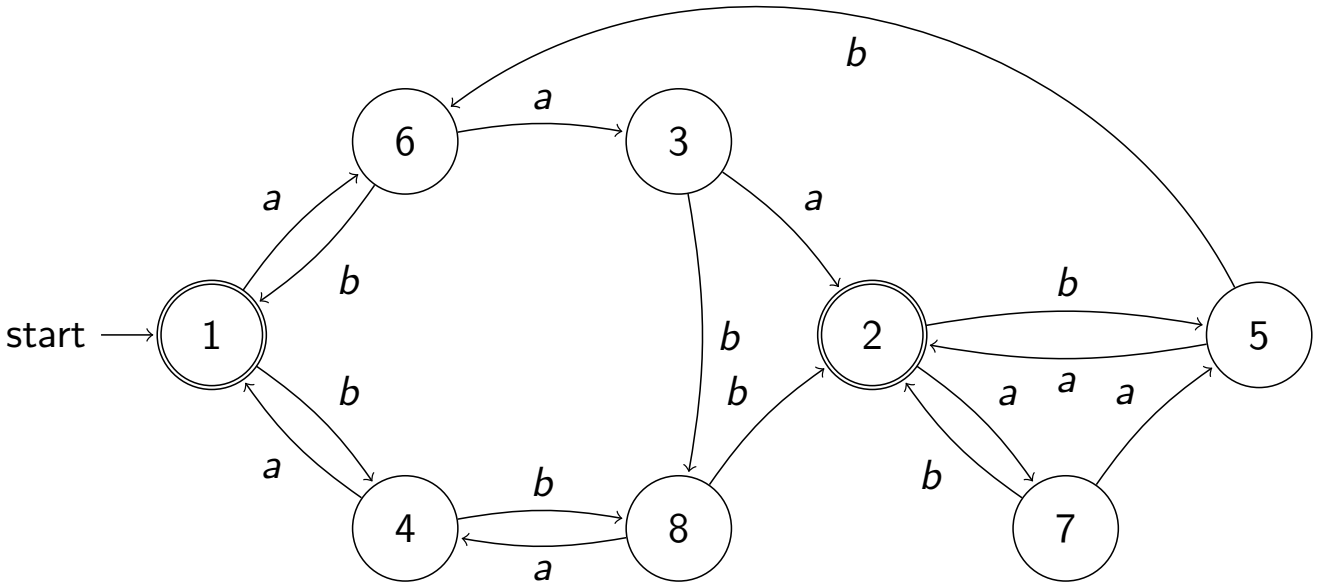
	3	4	5	6	7	8	1	2
3							0	0
4							0	0
5							0	0
6							0	0
7							0	0
8							0	0
1								
2								

Problem 1 - Step 1



	3	4	5	6	7	8	1	2
3				1	1	1	0	0
4				1	1	1	0	0
5				1	1	1	0	0
6							0	0
7							0	0
8							0	0
1								
2								

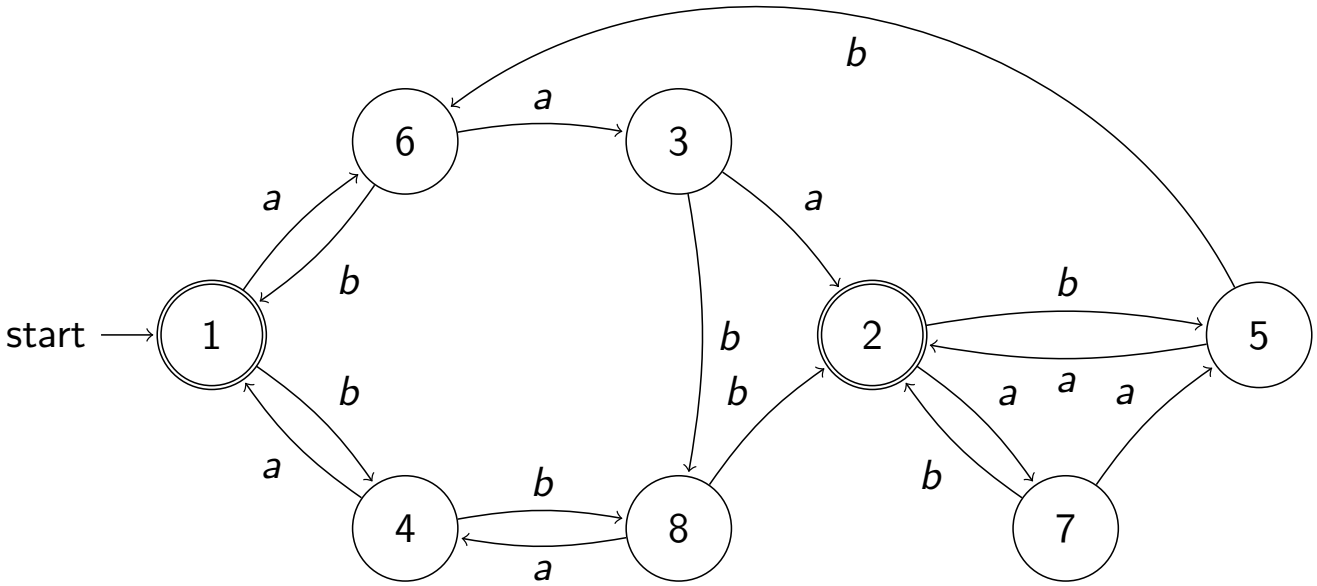
Problem 1 - Step 2



	3	4	5	6	7	8	1	2
3				1	1	1	0	0
4				1	1	1	0	0
5				1	1	1	0	0
6							0	0
7							0	0
8							0	0
1								
2								

No Change!

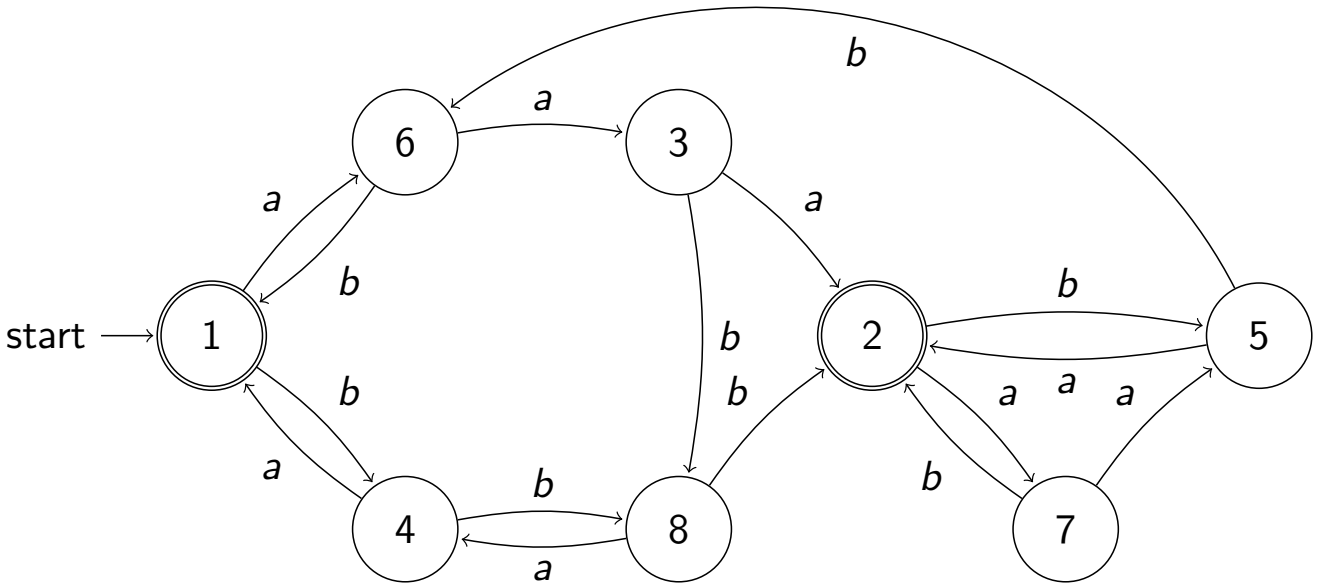
Problem 1 - End of Algorithm

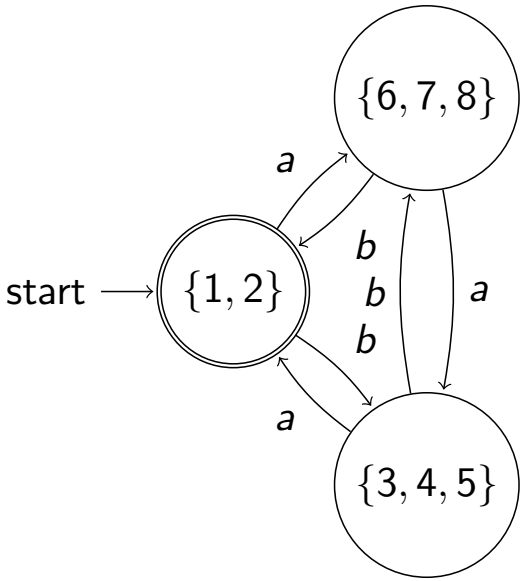


	3	4	5	6	7	8	1	2
3				1	1	1	0	0
4				1	1	1	0	0
5				1	1	1	0	0
6							0	0
7							0	0
8							0	0
1								
2								

Equivalances : $\{3, 4, 5\}, \{6, 7, 8\}, \{1, 2\}$

Problem 1 - Resulting DFA



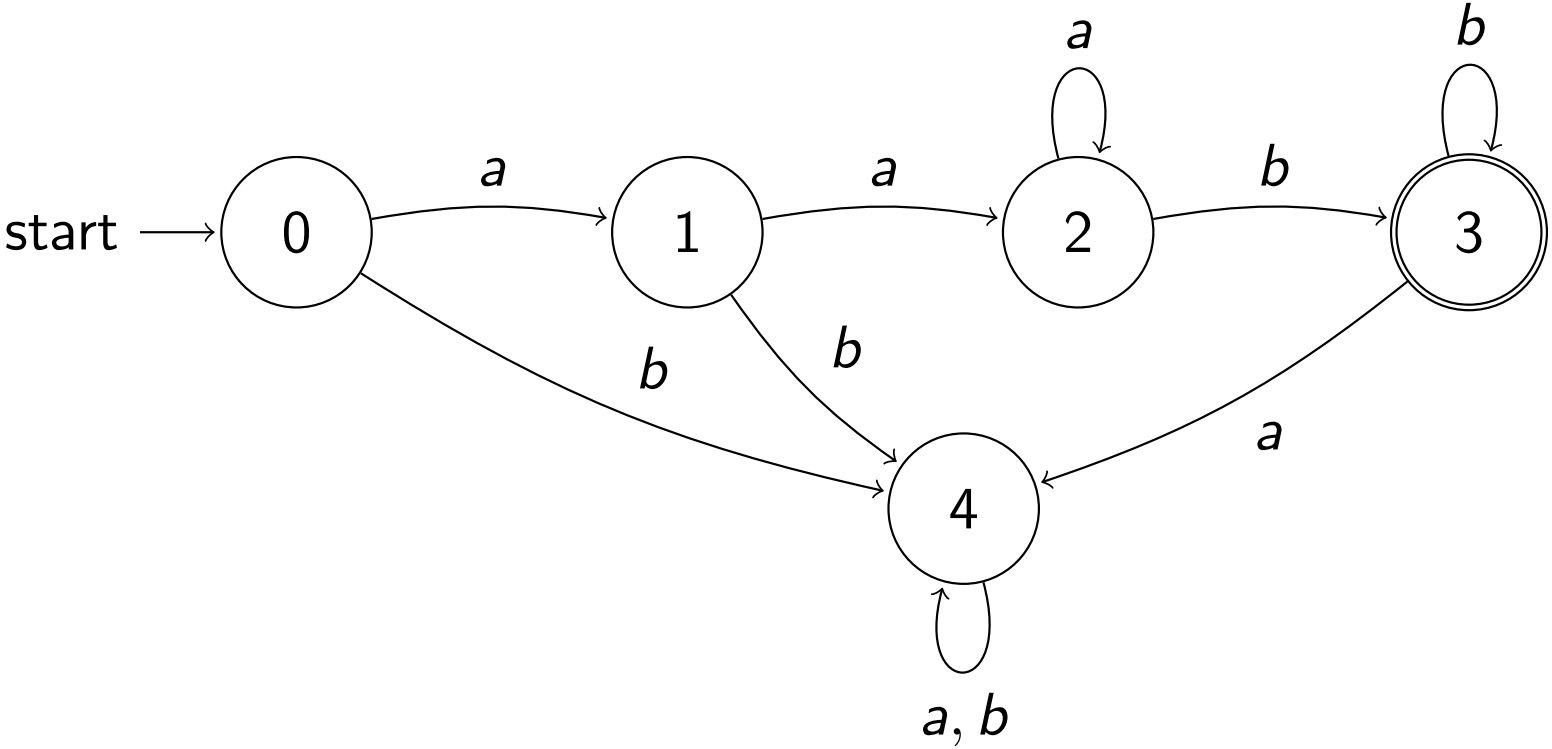


Problem 2

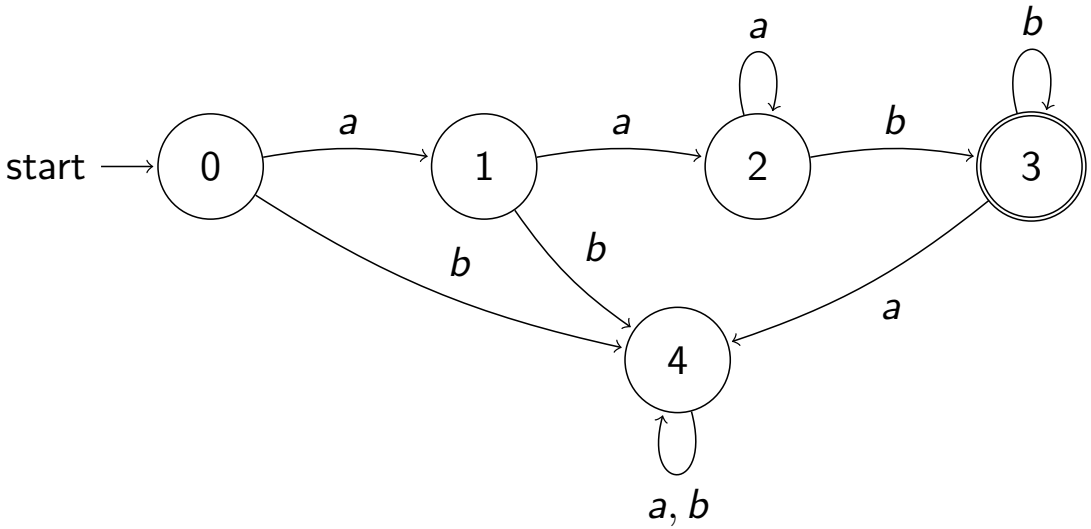
- Draw a DFA for the language $L = \{a^n b^m : n \geq 2, m \geq 1\}$

Problem 2

- Draw a DFA for the language $L = \{a^n b^m : n \geq 2, m \geq 1\}$

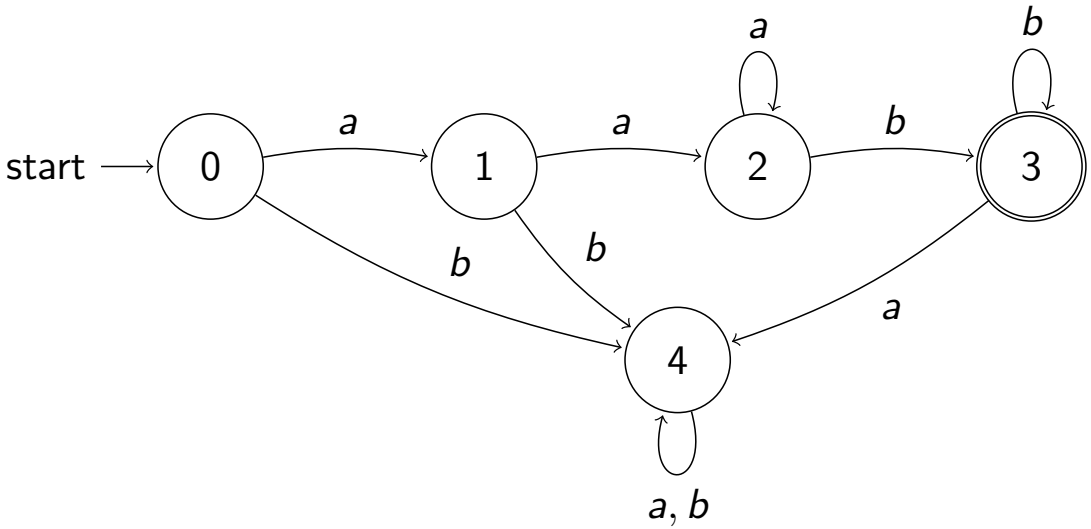


Problem 2 - Minimization - Step 0



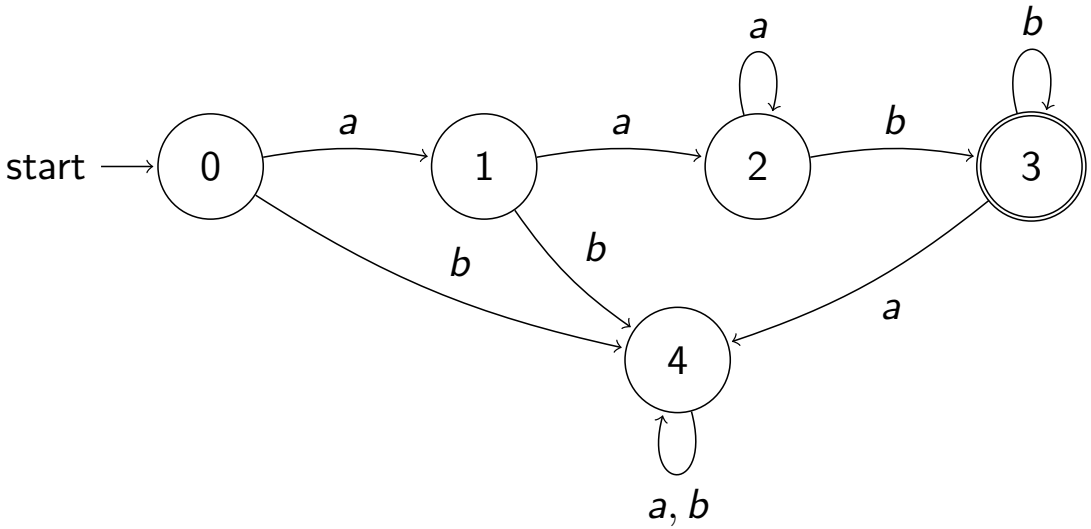
	0	1	2	4	3
0					0
1					0
2					0
4					0
3					

Problem 2 - Minimization - Step 1



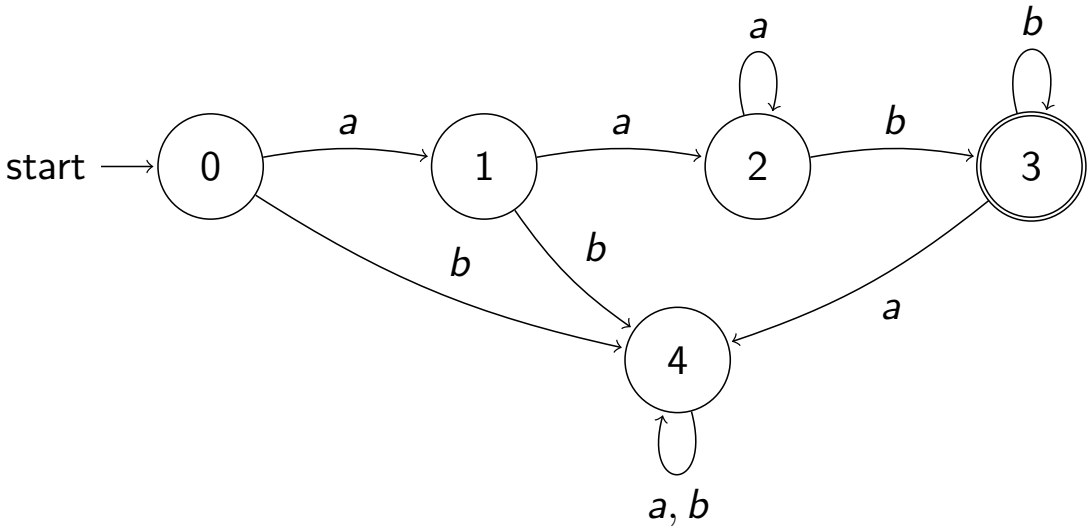
	0	1	2	4	3
0			1		0
1			1		0
2				1	0
4					0
3					

Problem 2 - Minimization - Step 2



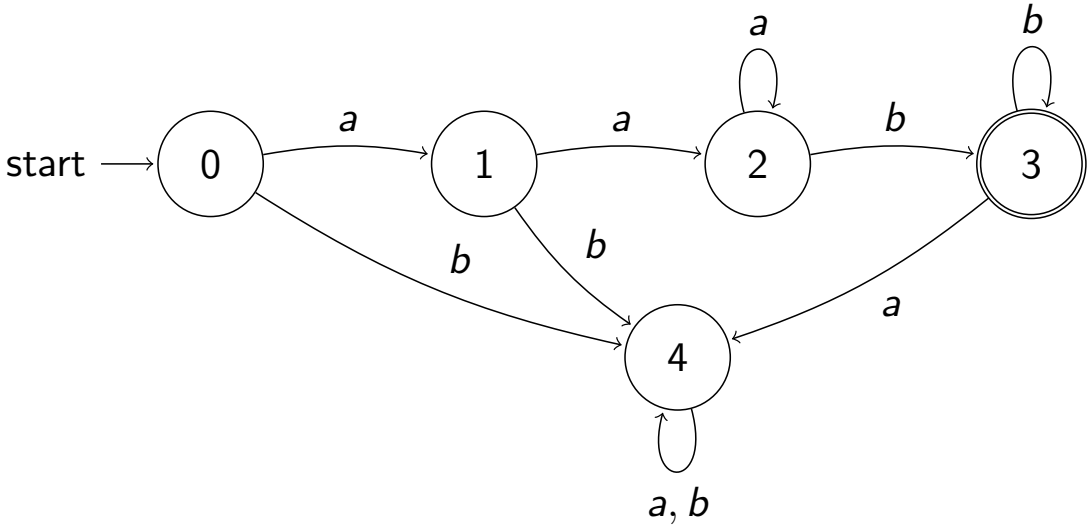
	0	1	2	4	3
0		2	1		0
1			1	2	0
2				1	0
4					0
3					

Problem 2 - Minimization - Step 3



	0	1	2	4	3
0		2	1	3	0
1			1	2	0
2				1	0
4					0
3					

Problem 2 - Minimization - Result

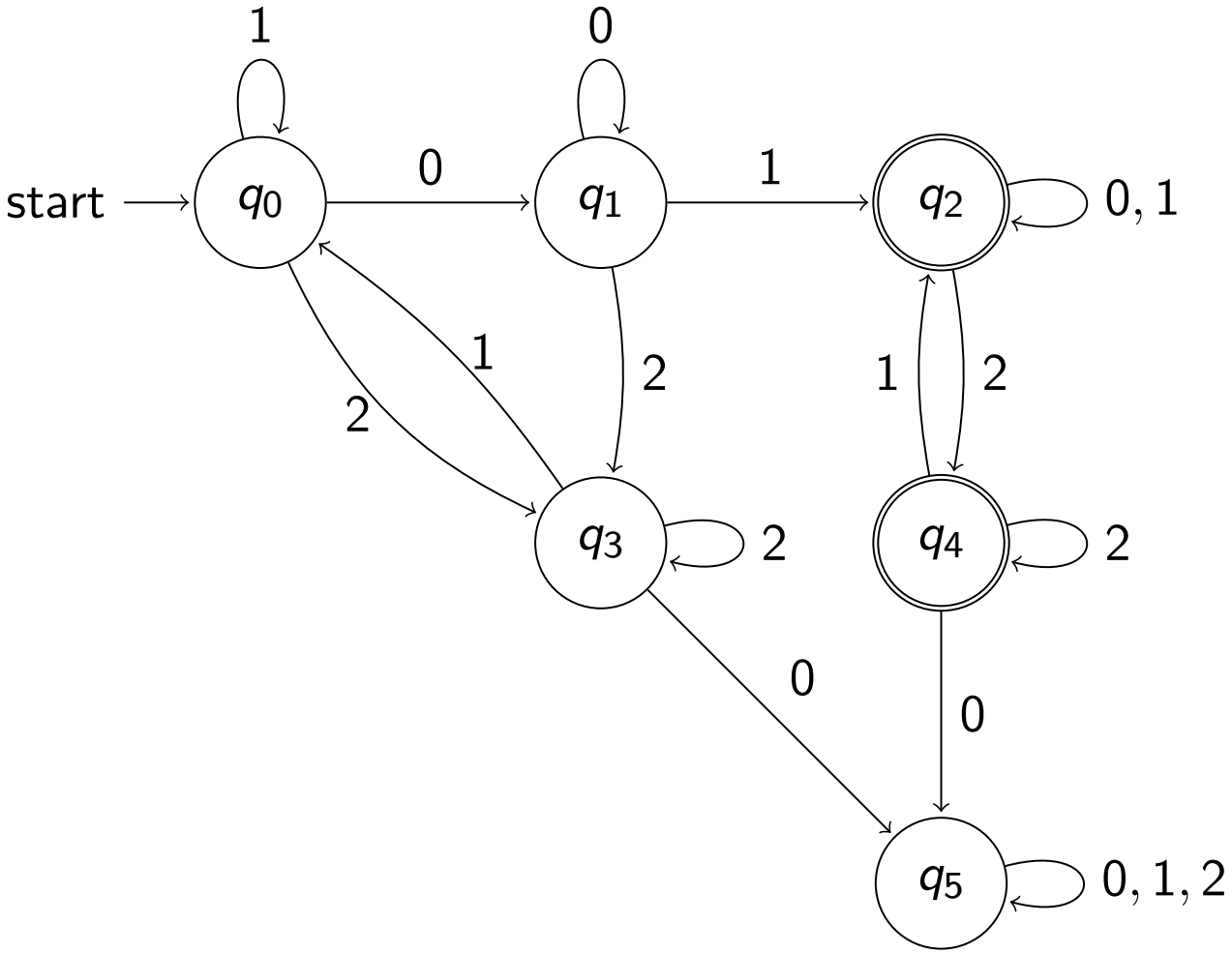


	0	1	2	4	3
0		2	1	3	0
1			1	2	0
2				1	0
4					0
3					

It is already a minimum DFA!

Problem 3

For alphabet $\Sigma = \{0, 1, 2\}$, design a DFA/NFA accepts all strings that includes "01", but does not include "20", and minimize it. Solution from the first recitation:



Problem 3 - Alternative Analysis

- $L = \{w \mid w \in$
  $\Sigma^* \wedge$ *contains* "01" *as substring* $\wedge$ *does not contain* "20" *as substring*$\}$

Problem 3 - Alternative Analysis

- $L = \{w \mid w \in$
  $\Sigma^* \wedge$ *contains* "01" *as substring* $\wedge$ *does not contain* "20" *as substring*$\}$
- $L_A = \{w \mid w \in \Sigma^* \ \wedge \ \textit{contains}$ "01" *as substring*$\}$

Problem 3 - Alternative Analysis

- $L = \{w \mid w \in \Sigma^* \wedge \textit{contains "01" as substring} \wedge \textit{does not contain "20" as substring}\}$
- $L_A = \{w \mid w \in \Sigma^* \wedge \textit{contains "01" as substring}\}$
- $L_B = \{w \mid w \in \Sigma^* \wedge \textit{contains "20" as substring}\}$

Problem 3 - Alternative Analysis

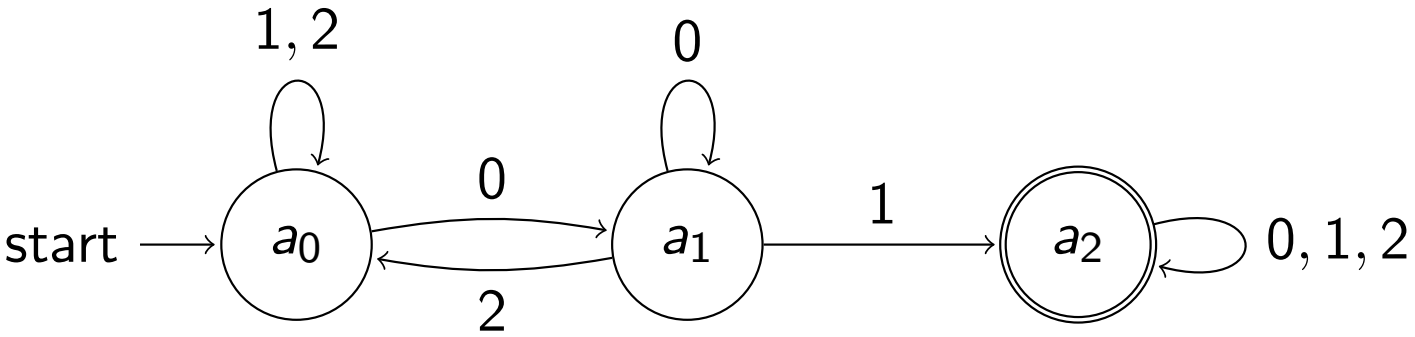
- $L = \{w \mid w \in \Sigma^* \wedge \textit{contains "01" as substring} \wedge \textit{does not contain "20" as substring}\}$
- $L_A = \{w \mid w \in \Sigma^* \wedge \textit{contains "01" as substring}\}$
- $L_B = \{w \mid w \in \Sigma^* \wedge \textit{contains "20" as substring}\}$
- $L = L_A \backslash L_B$

Problem 3 - Alternative Analysis

- $L = \{w \mid w \in \Sigma^* \wedge \textit{contains "01" as substring} \wedge \textit{does not contain "20" as substring}\}$
- $L_A = \{w \mid w \in \Sigma^* \wedge \textit{contains "01" as substring}\}$
- $L_B = \{w \mid w \in \Sigma^* \wedge \textit{contains "20" as substring}\}$
- $L = L_A \backslash L_B$
- $L = L_A \backslash L_B = L_A \cap L_B^c$

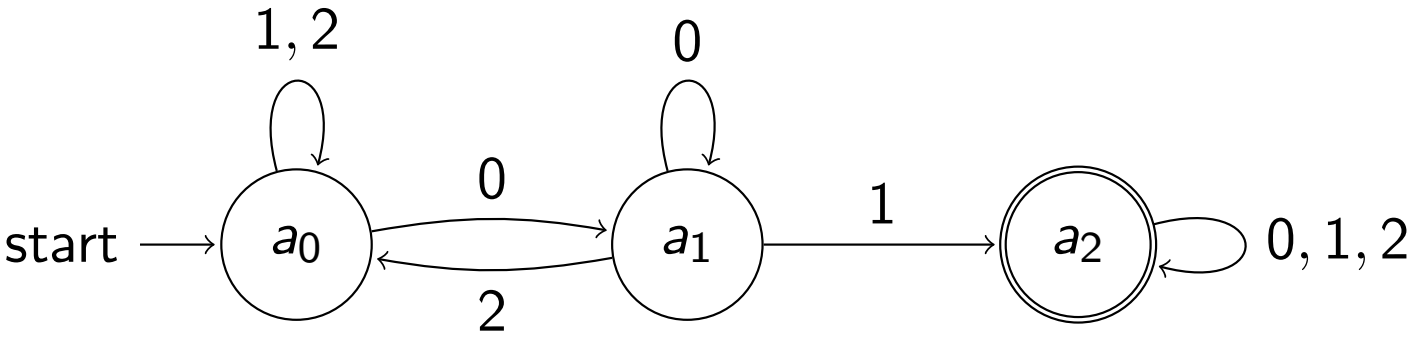
Problem 3 - DFA for L_A and L_B

- L_A:

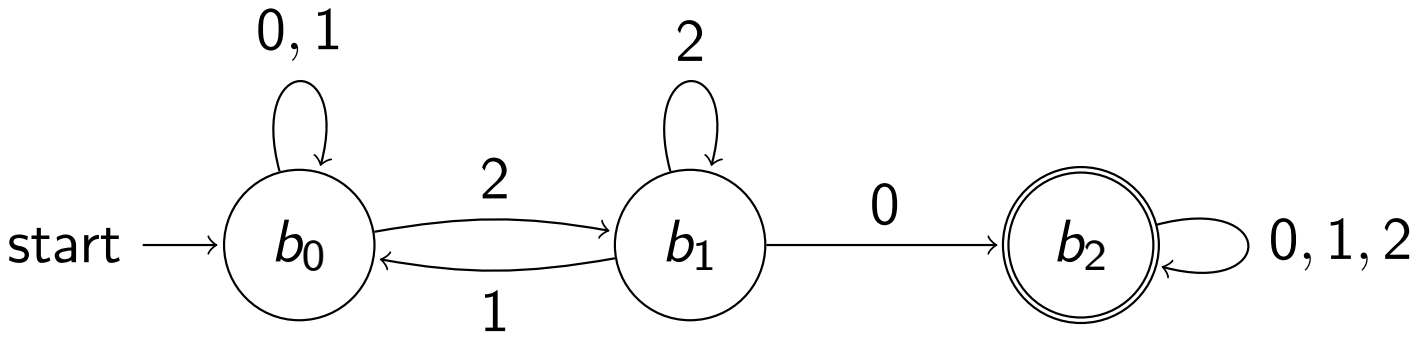


Problem 3 - DFA for L_A and L_B

- L_A:

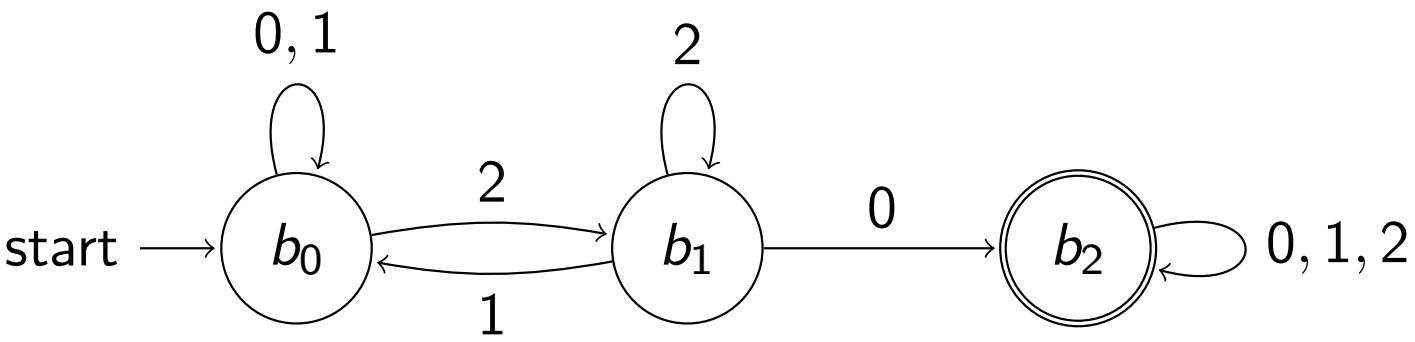


- L_B:



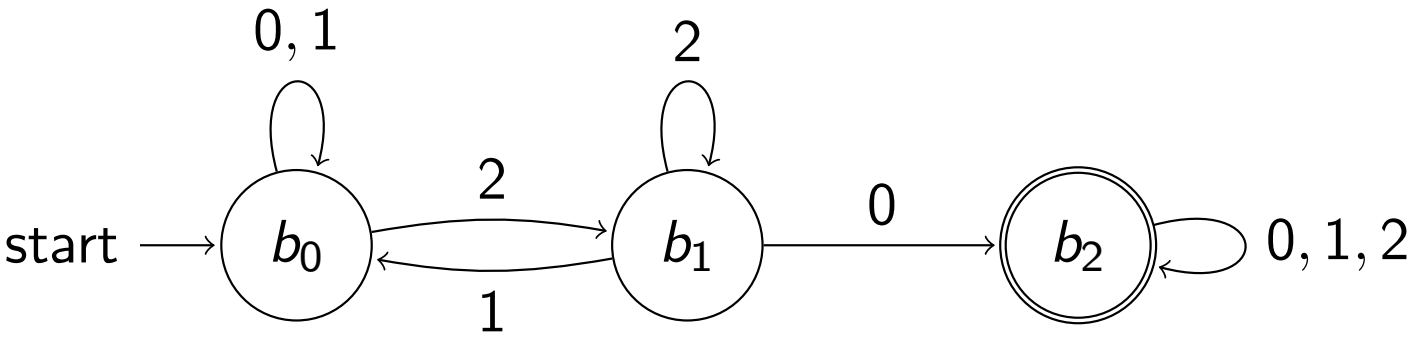
Problem 3 - Complement of L_B

- L_B:

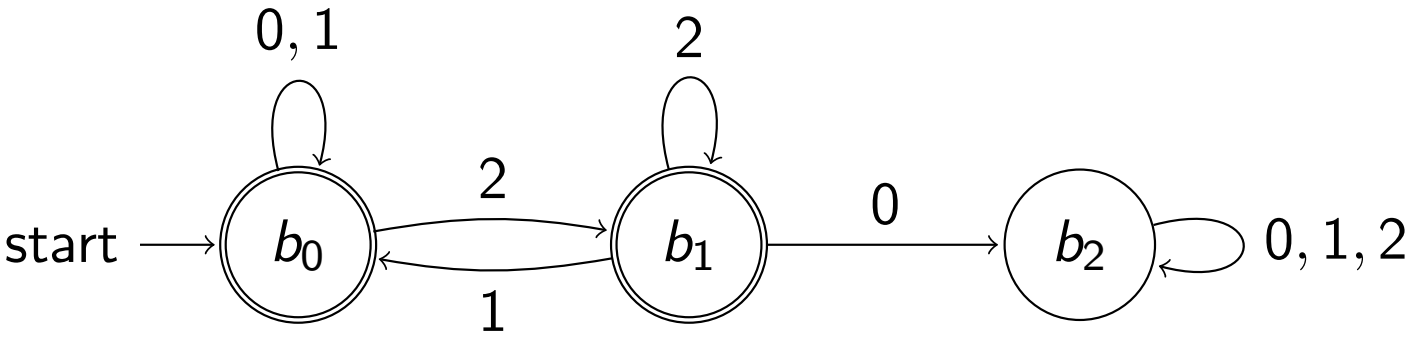


Problem 3 - Complement of L_B

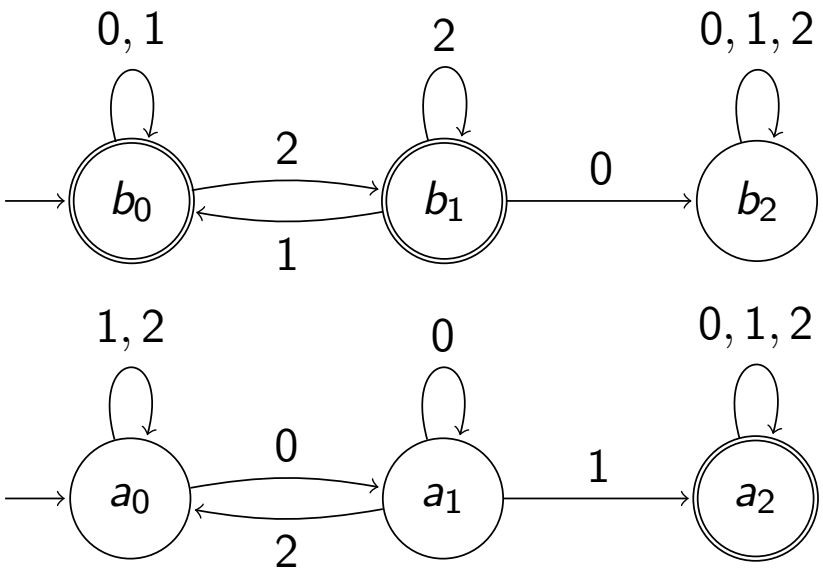
- L_B:

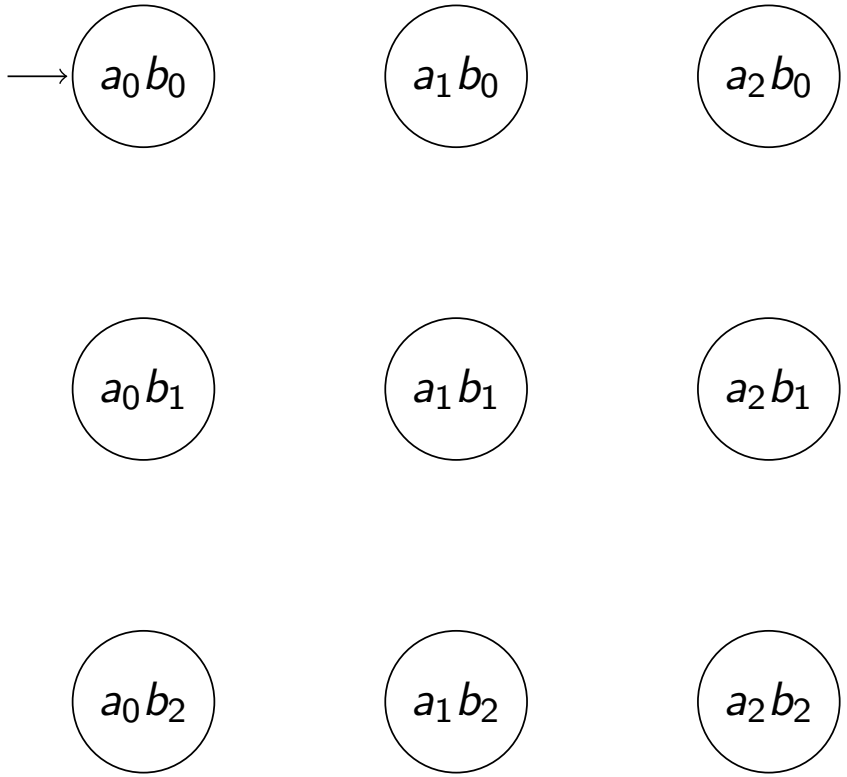


- L_B^c:

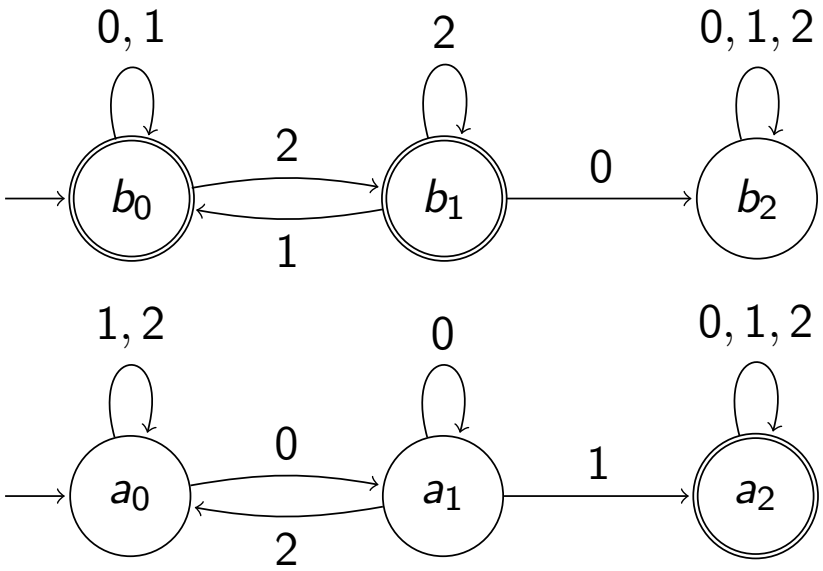


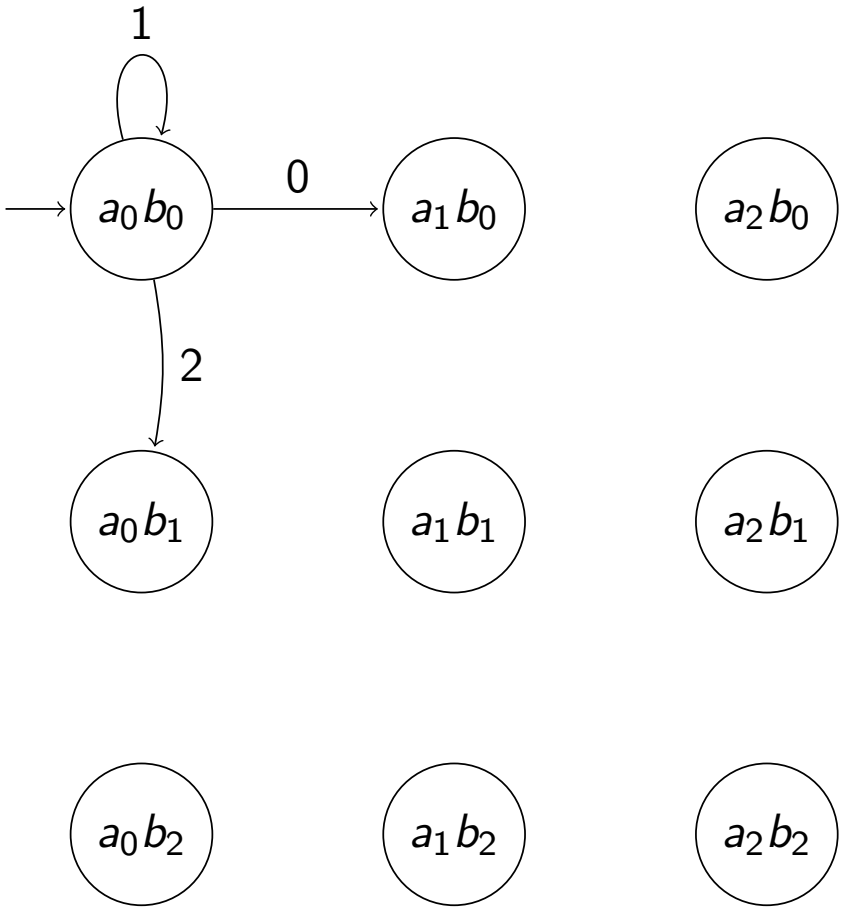
Problem 3 - Product of L_A and L_B^c



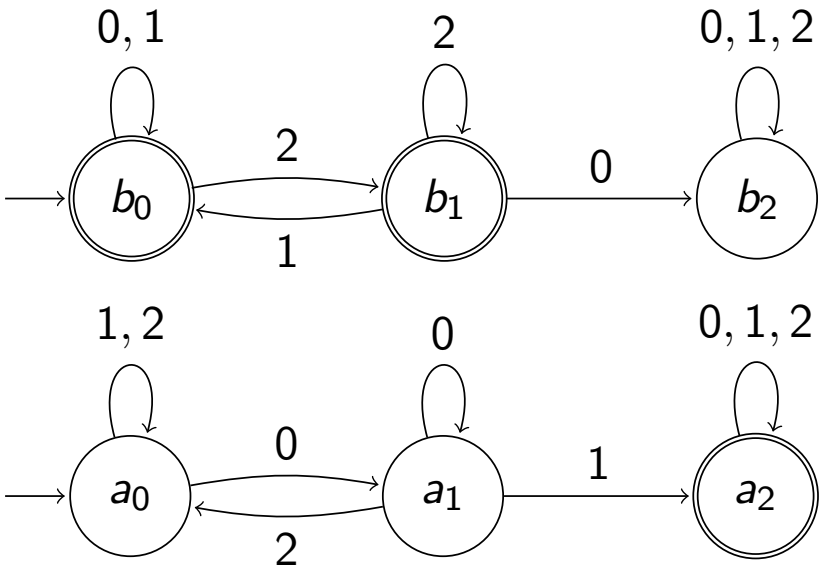


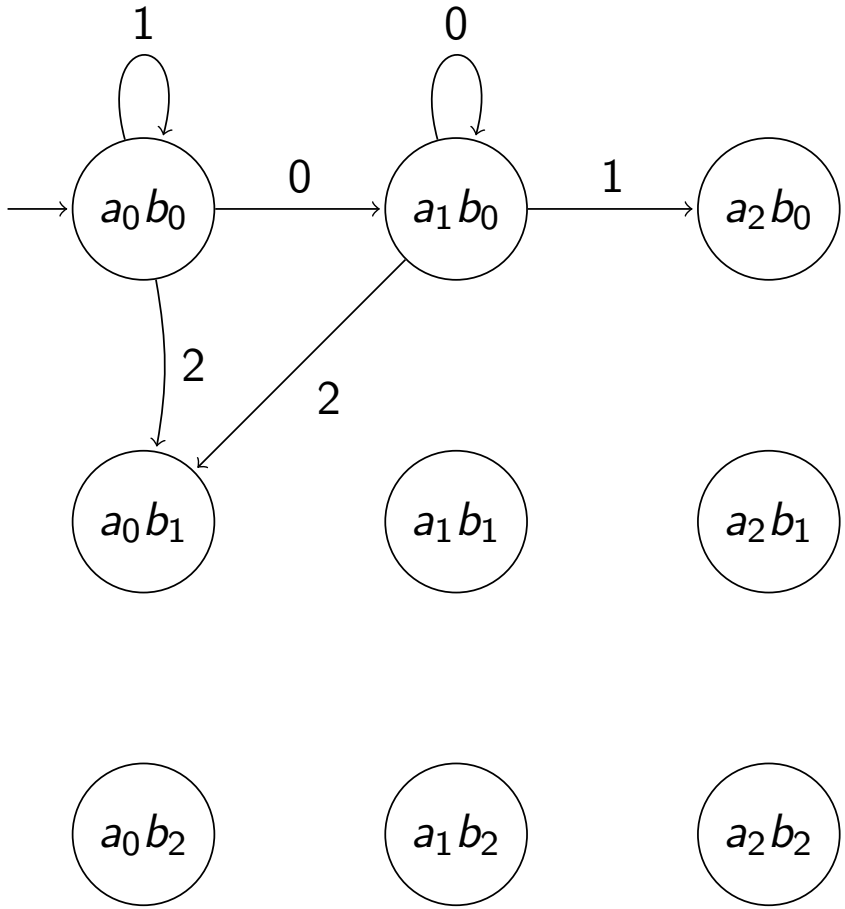
Problem 3 - Product of L_A and L_B^c



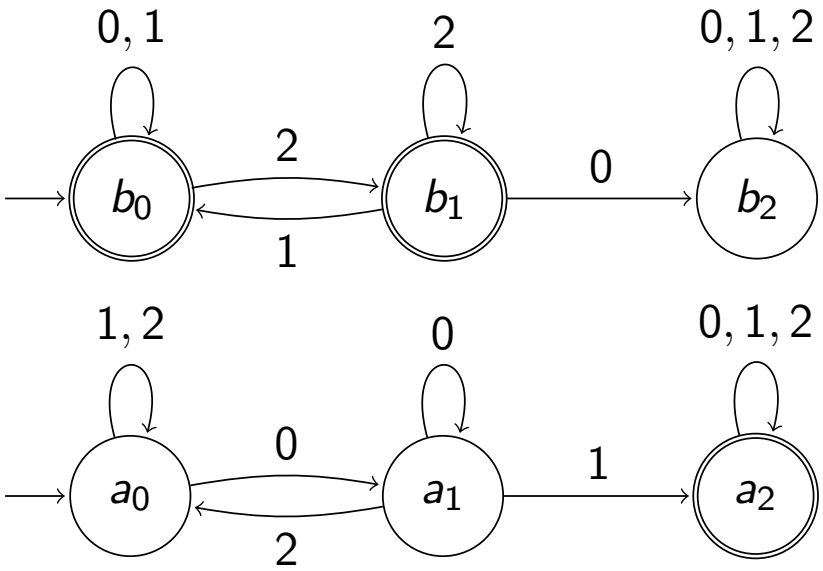


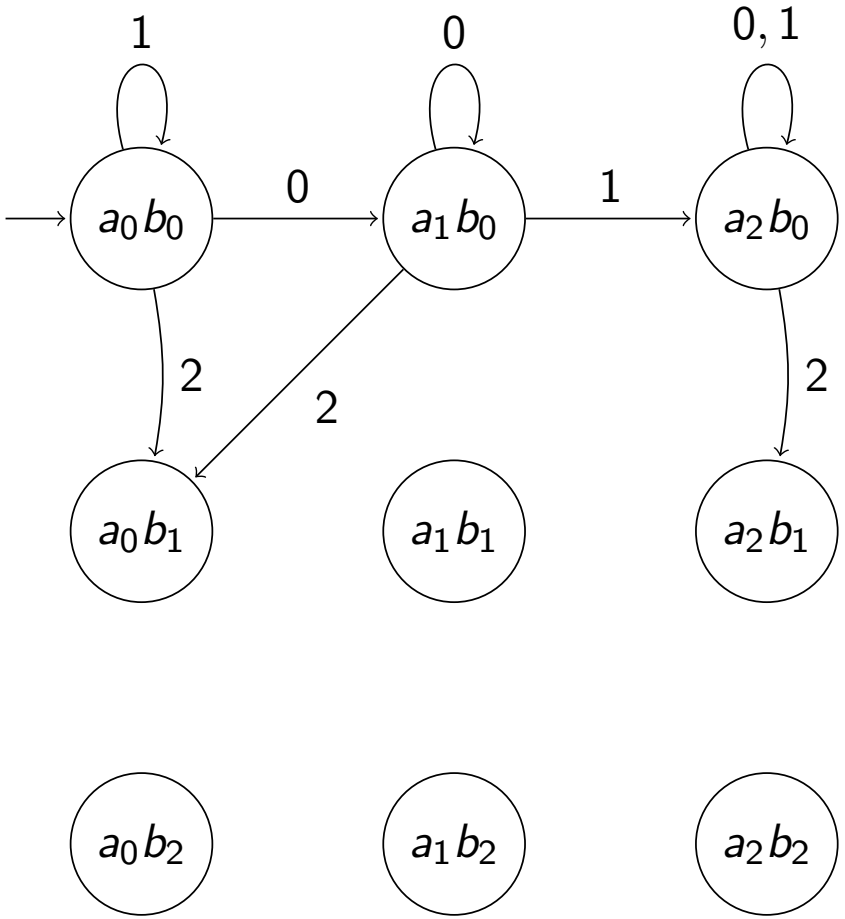
Problem 3 - Product of L_A and L_B^c



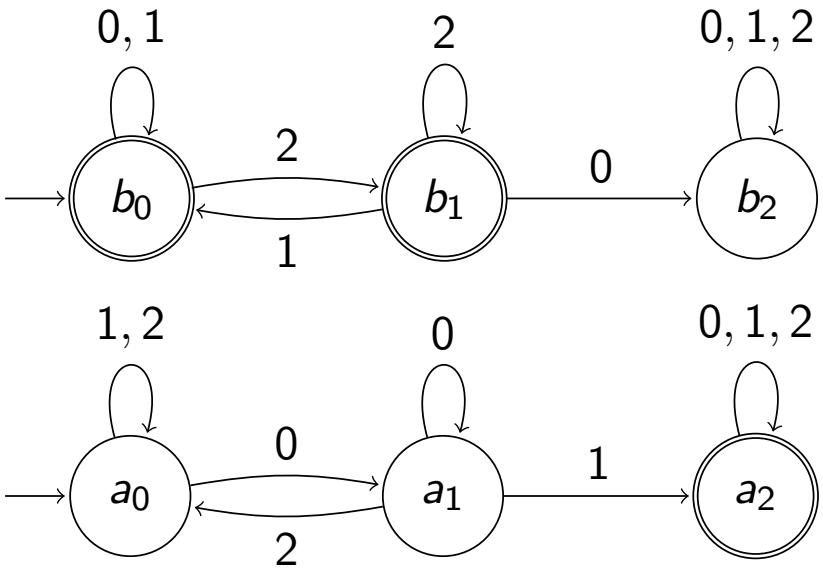


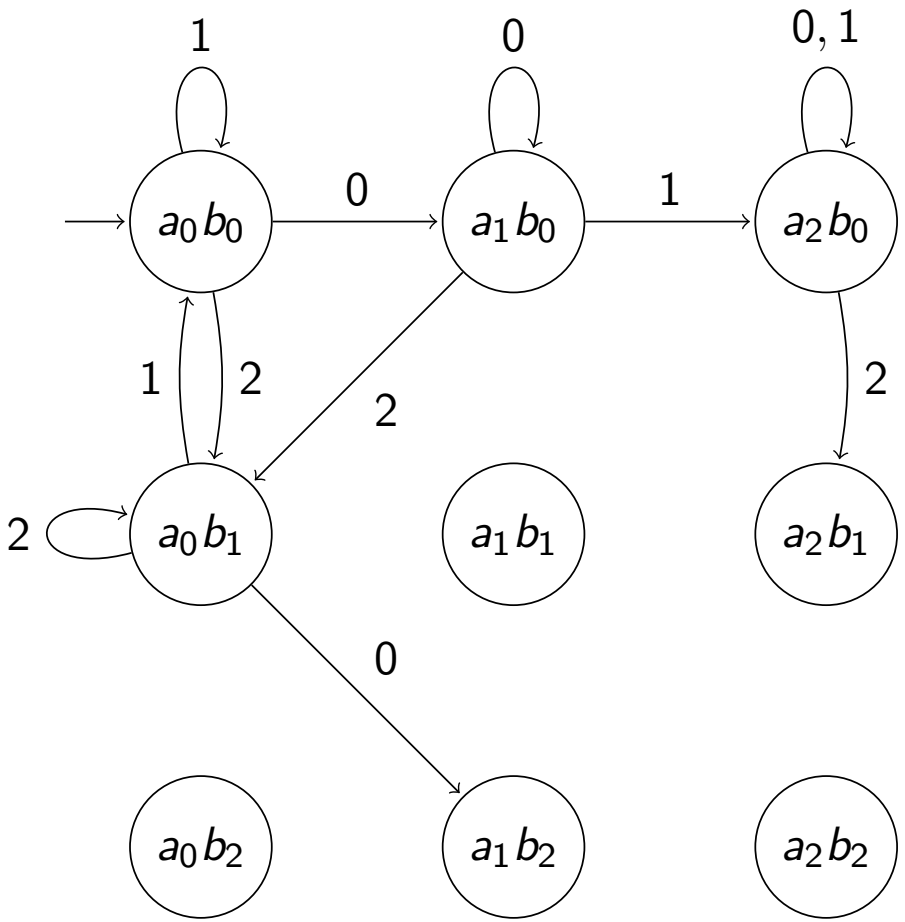
Problem 3 - Product of L_A and L_B^c



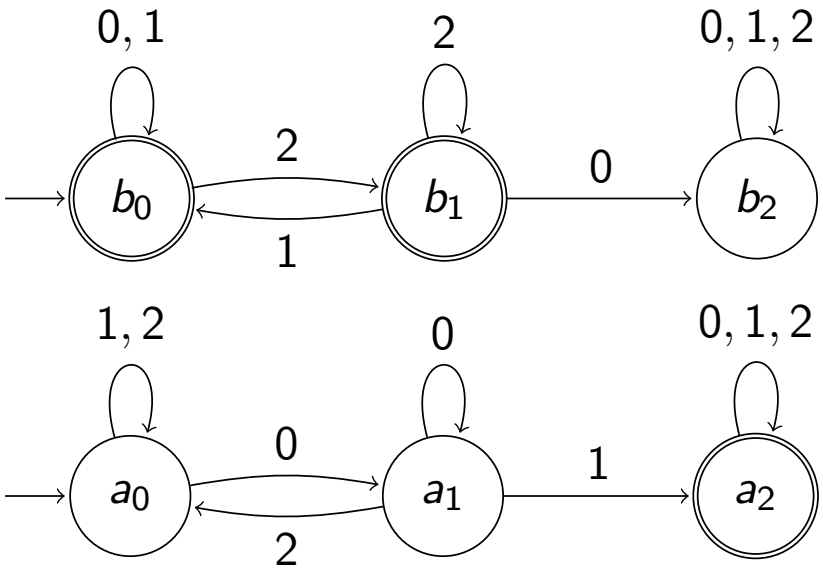


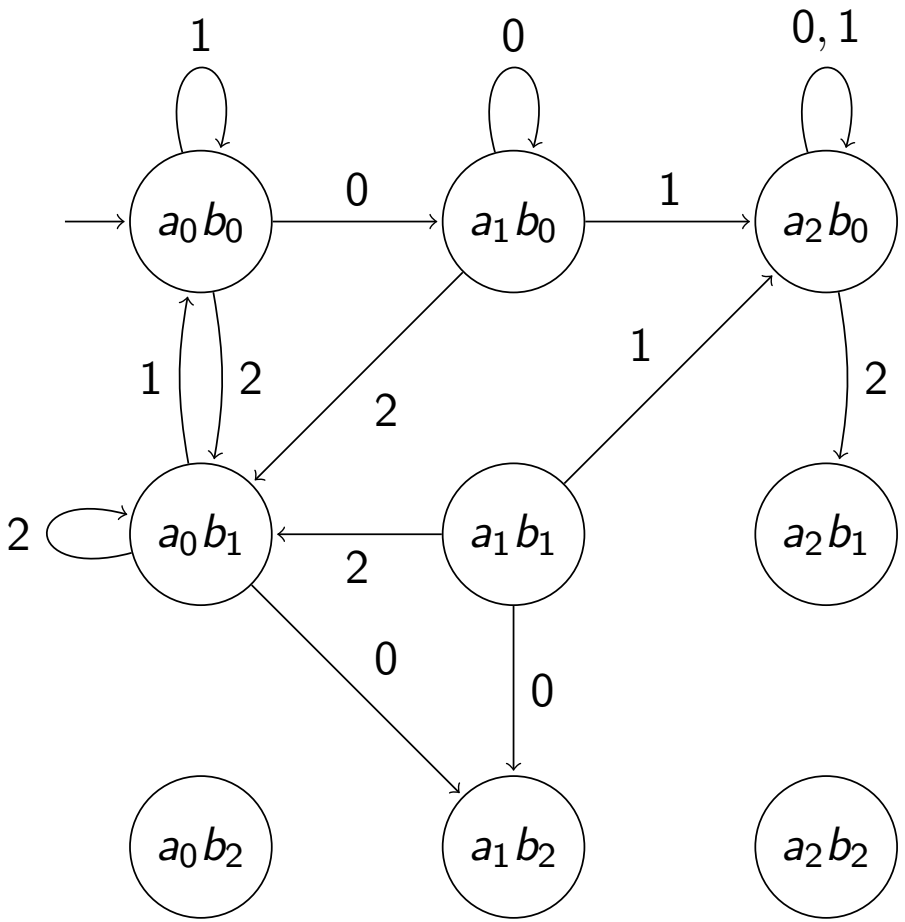
Problem 3 - Product of L_A and L_B^c



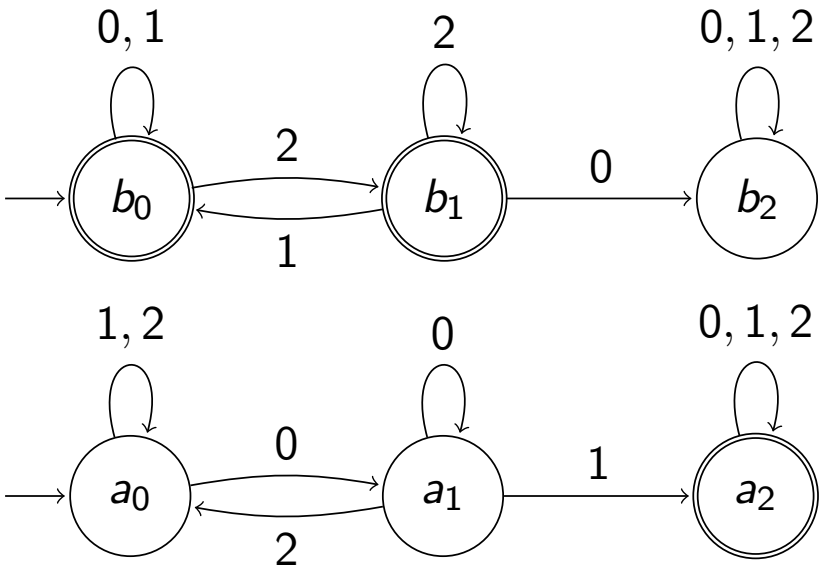


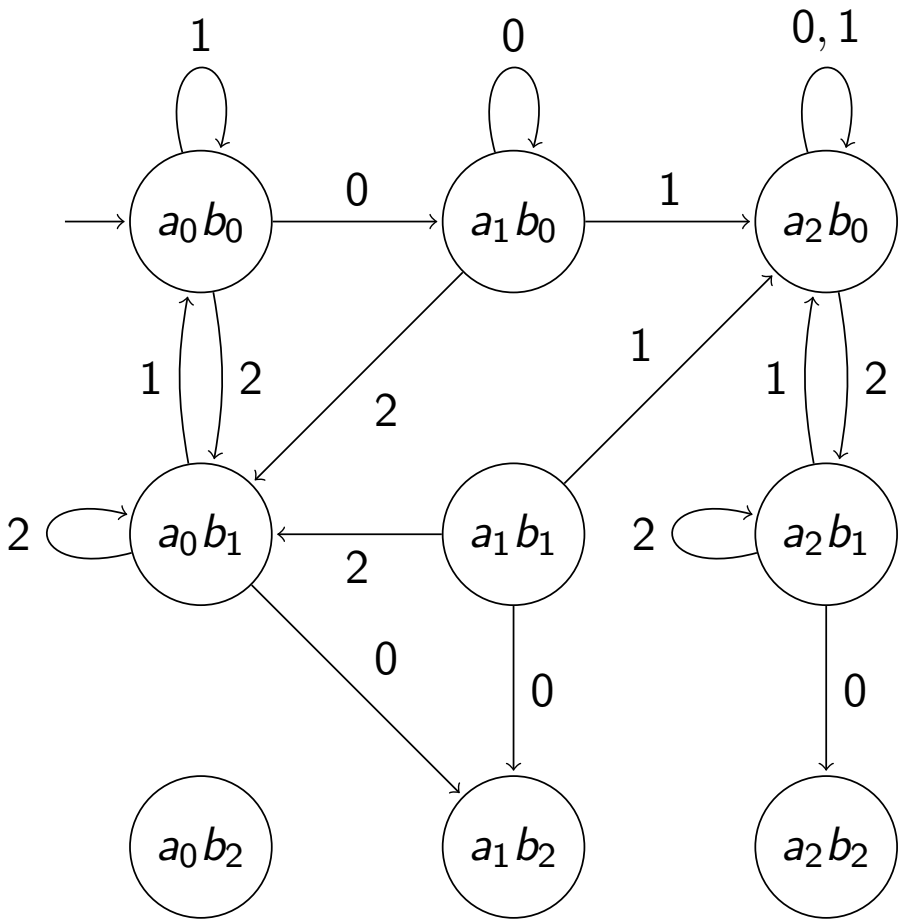
Problem 3 - Product of L_A and L_B^c



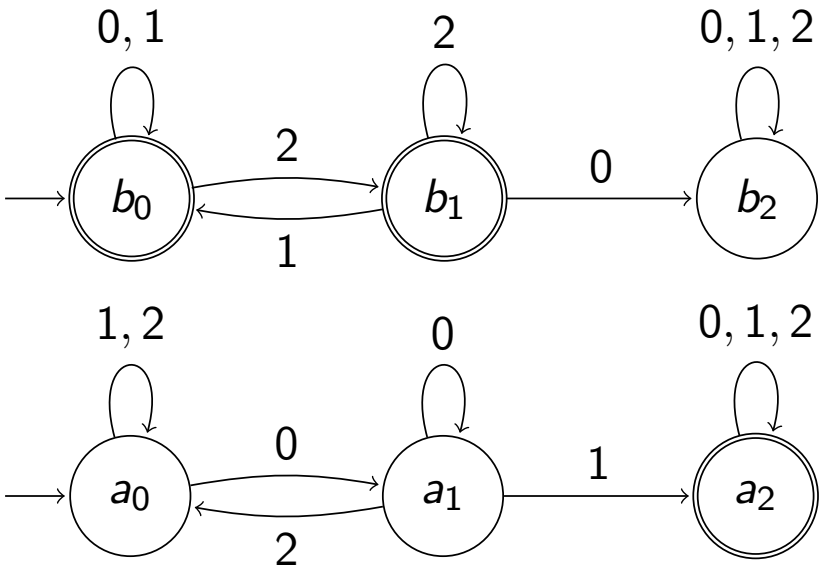


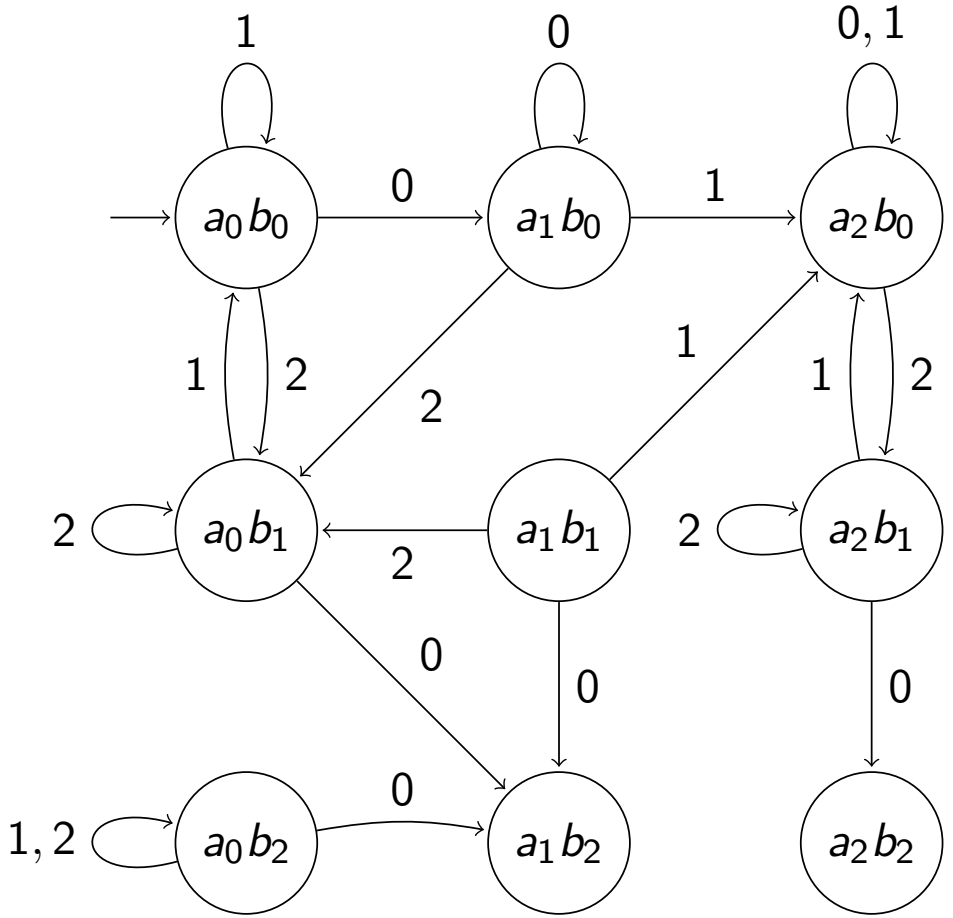
Problem 3 - Product of L_A and L_B^c



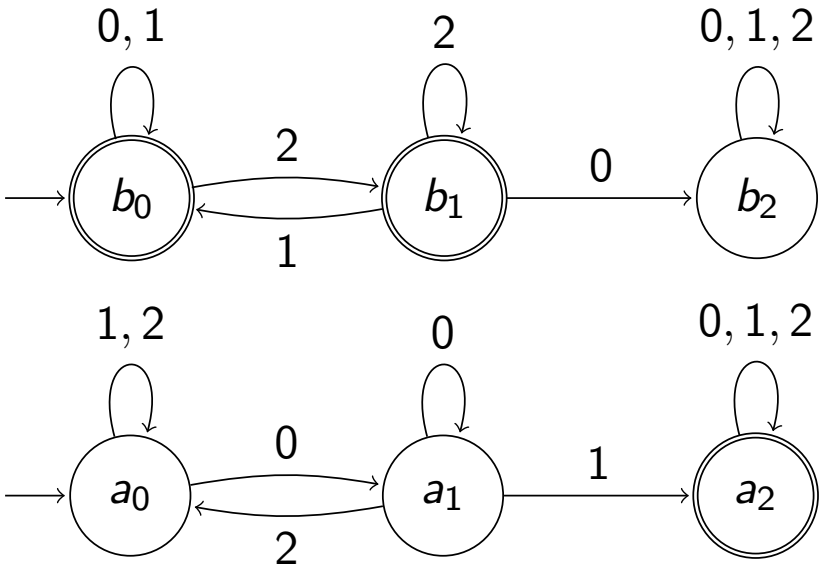


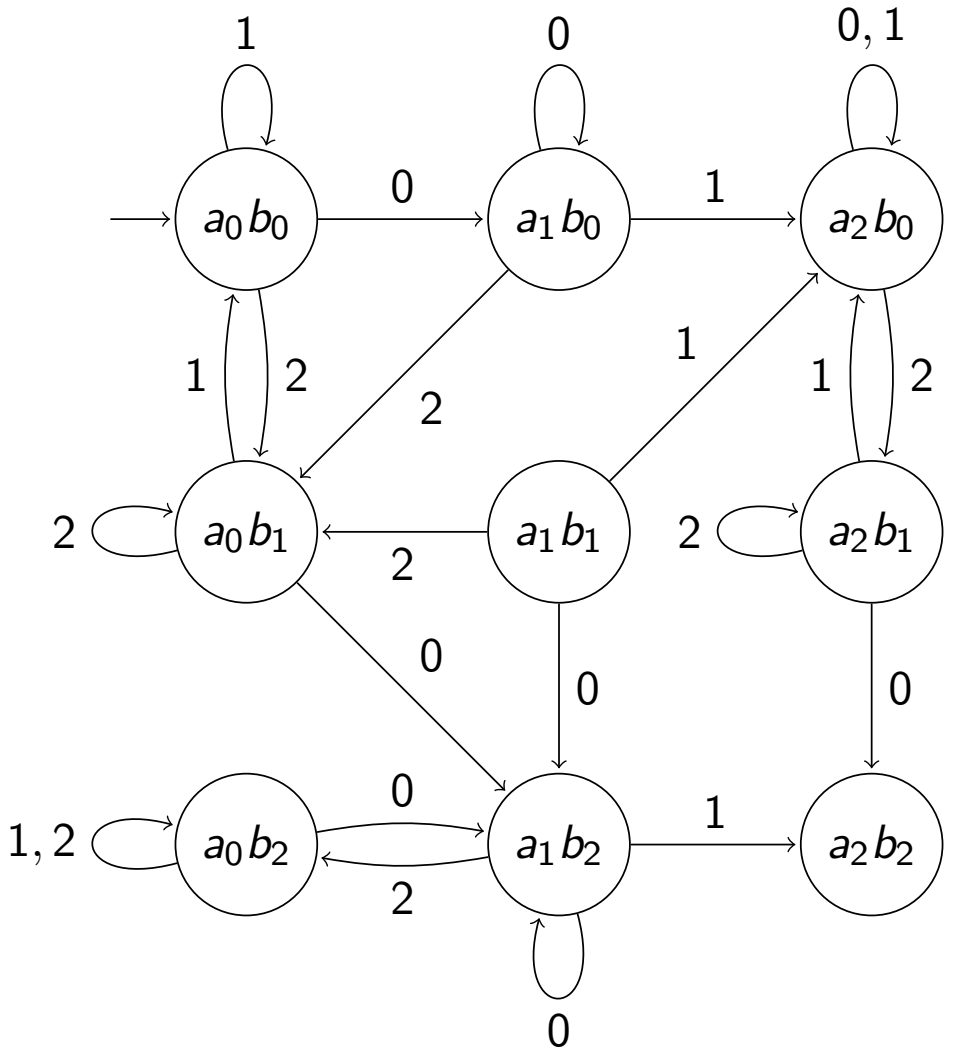
Problem 3 - Product of L_A and L_B^c



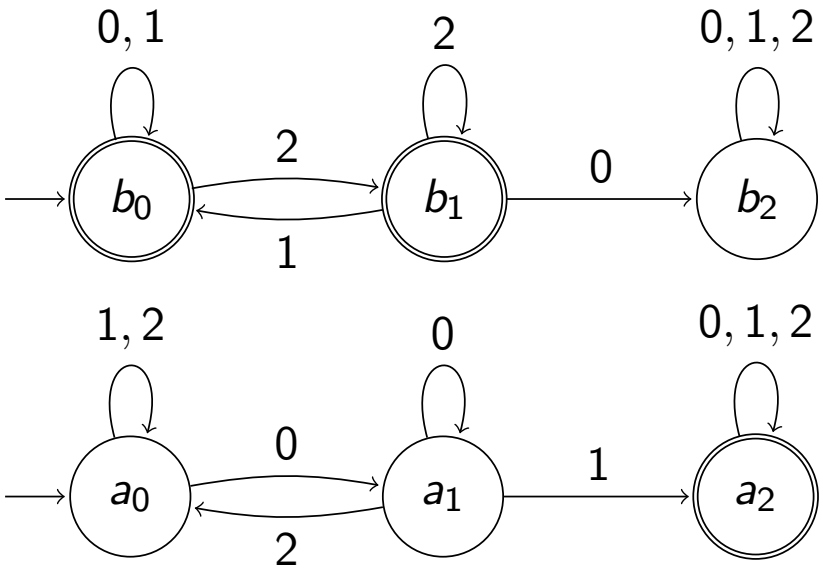


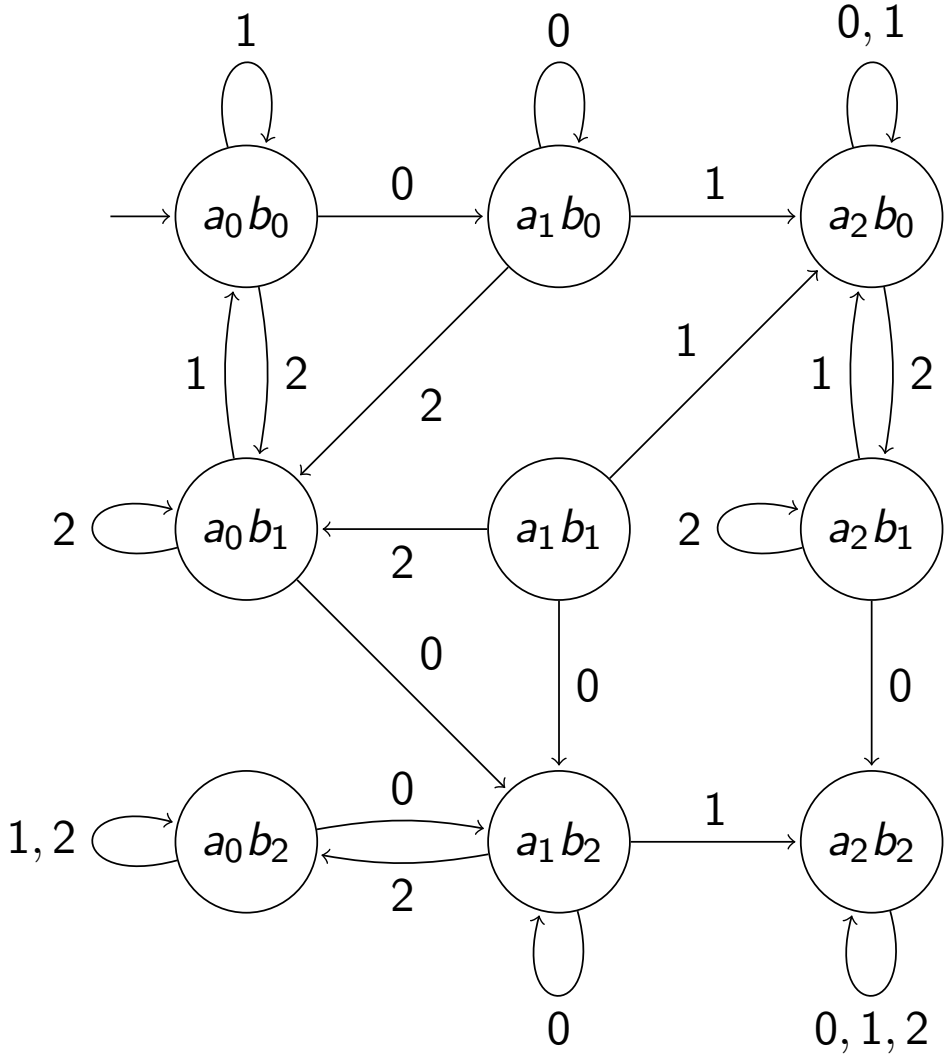
Problem 3 - Product of L_A and L_B^c





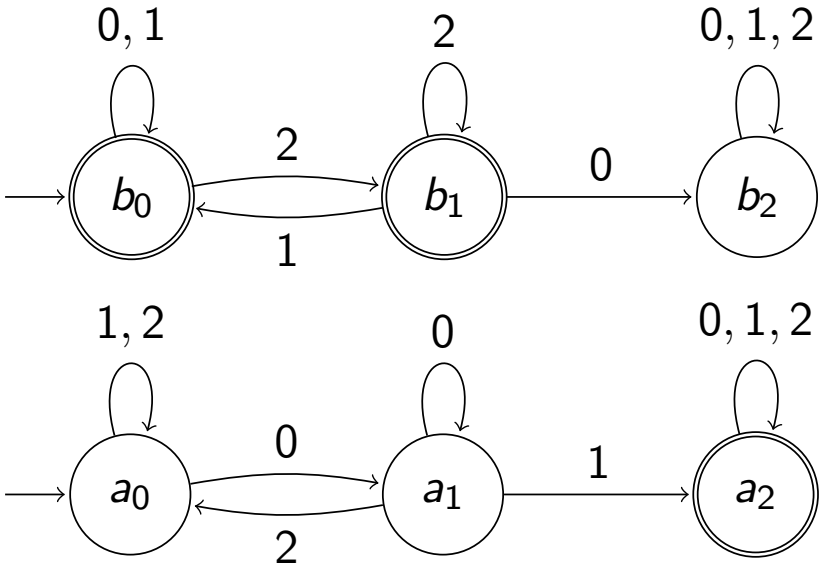
Problem 3 - Product of L_A and L_B^c

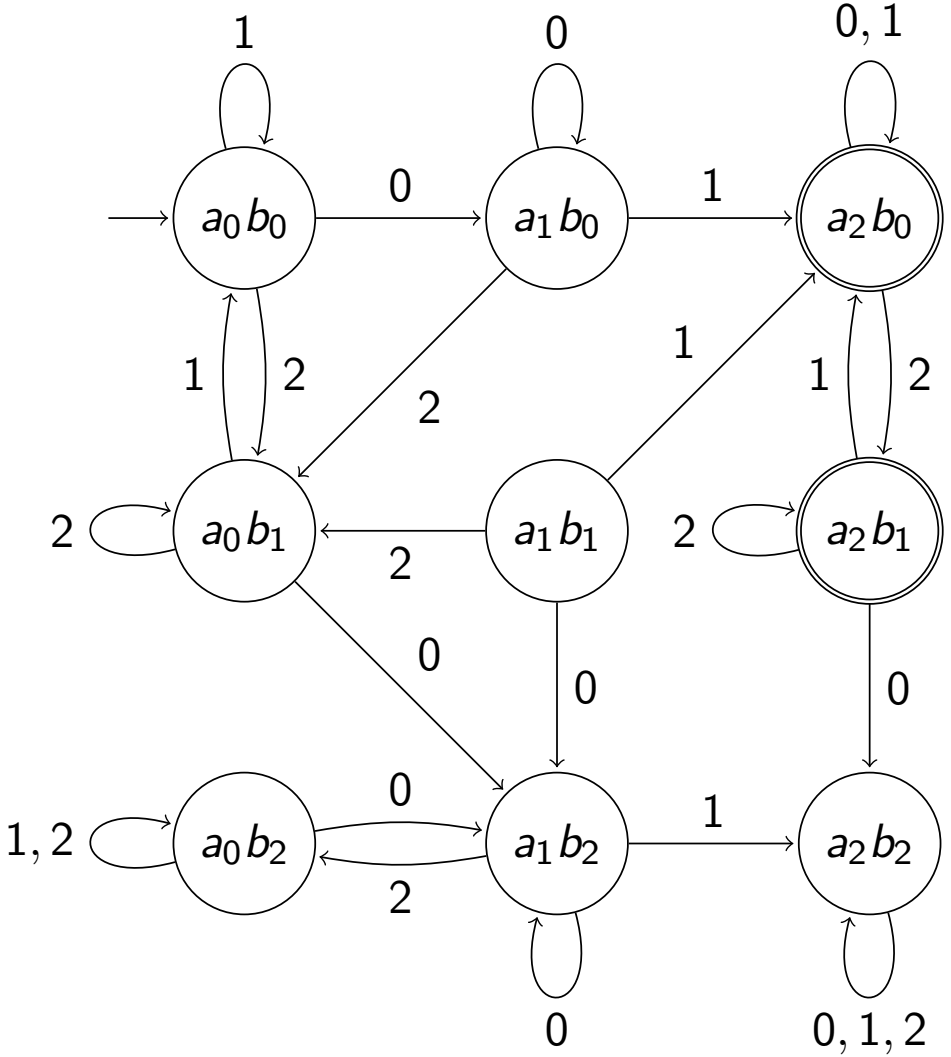




Problem 3 - Product of L_A and L_B^c

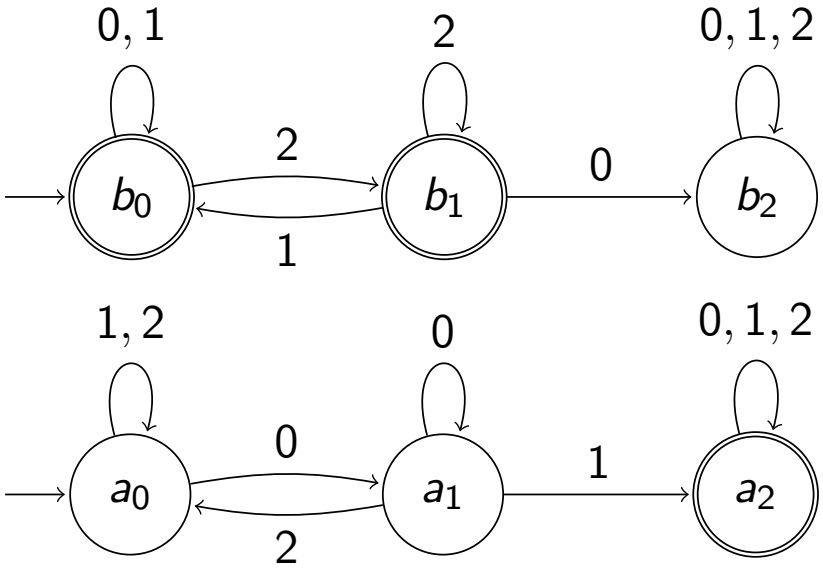
Add final states...

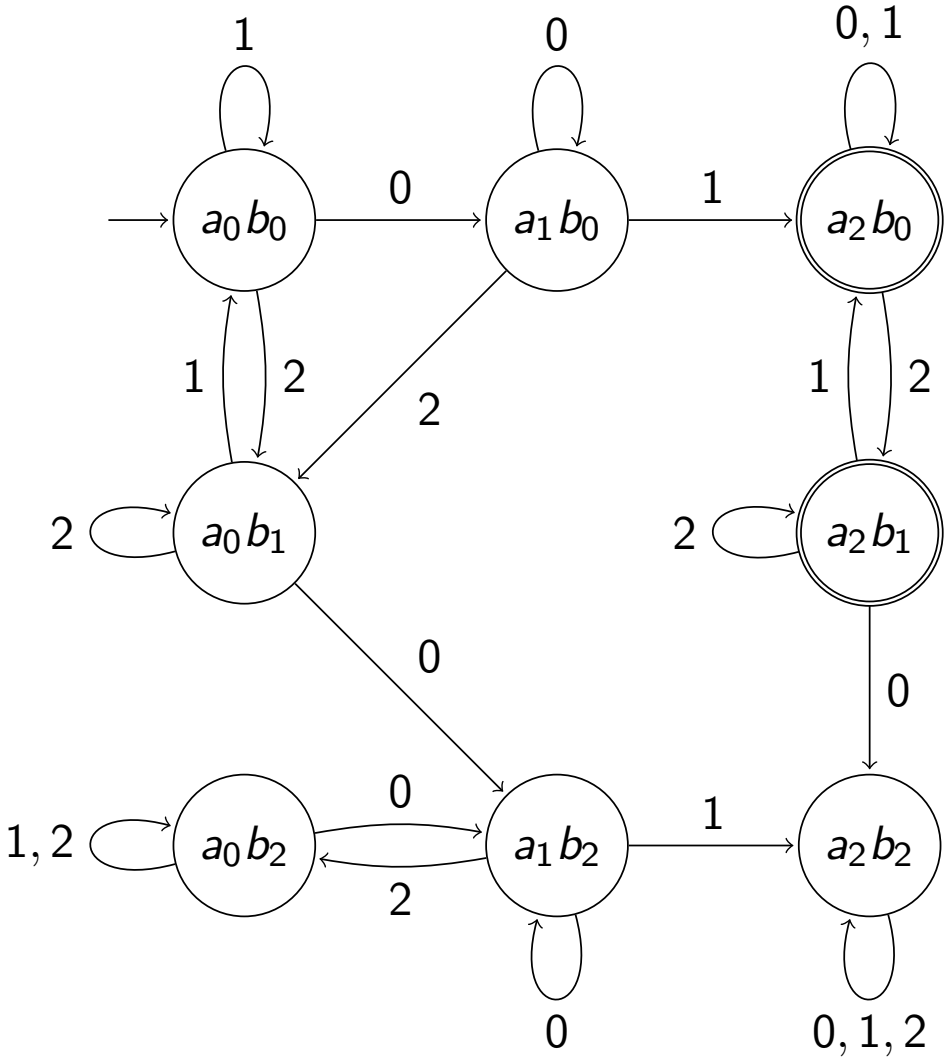




Problem 3 - Product of L_A and L_B^c

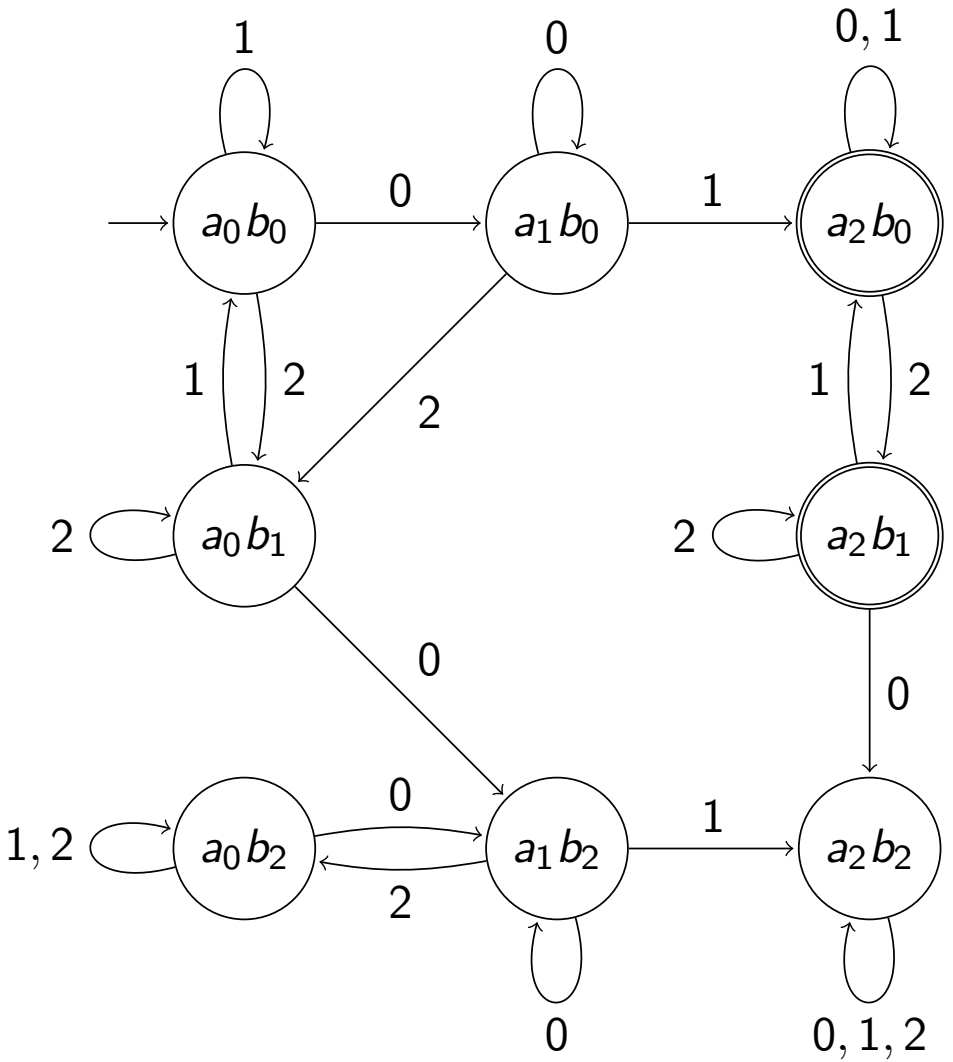
Eliminate unreachable states...





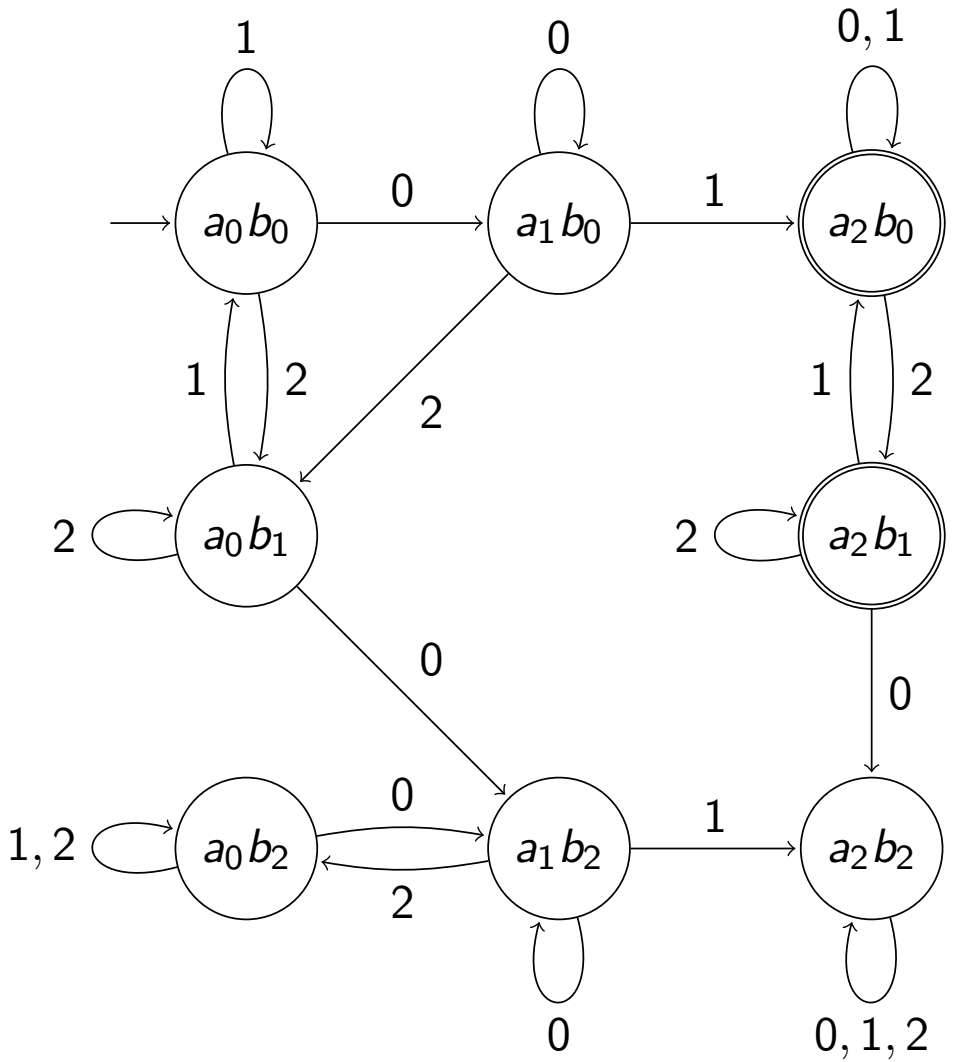
Problem 3 - Minimize the Product

	a_0b_0	a_1b_0	a_0b_1	a_0b_2	a_1b_2	a_2b_2	a_2b_0	a_2b_1
a_0b_0							0	0
a_1b_0							0	0
a_0b_1							0	0
a_0b_2							0	0
a_1b_2							0	0
a_2b_2							0	0
a_2b_0								
a_2b_1								



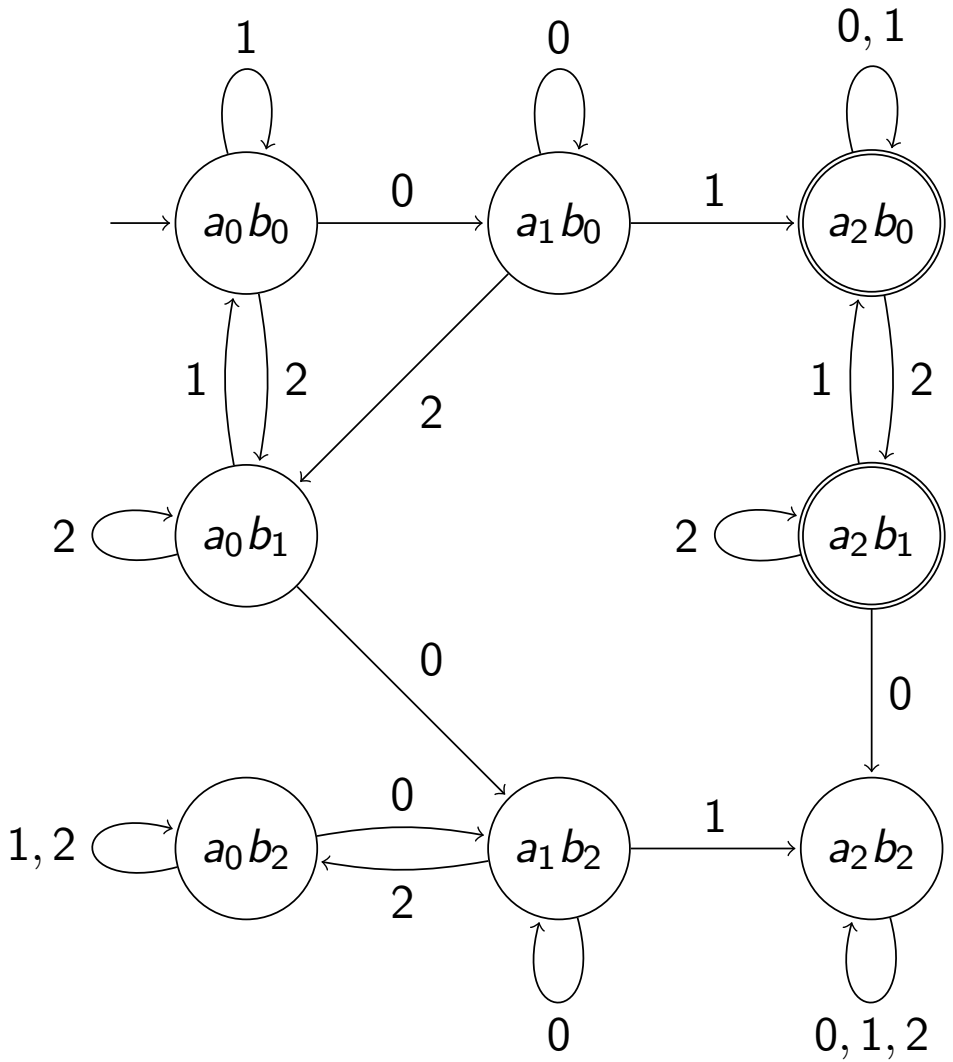
Problem 3 - Minimize the Product - Step 1

	a_0b_0	a_1b_0	a_0b_1	a_0b_2	a_1b_2	a_2b_2	a_2b_0	a_2b_1
a_0b_0		1					0	0
a_1b_0			1	1	1	1	0	0
a_0b_1							0	0
a_0b_2							0	0
a_1b_2							0	0
a_2b_2							0	0
a_2b_0								1
a_2b_1								



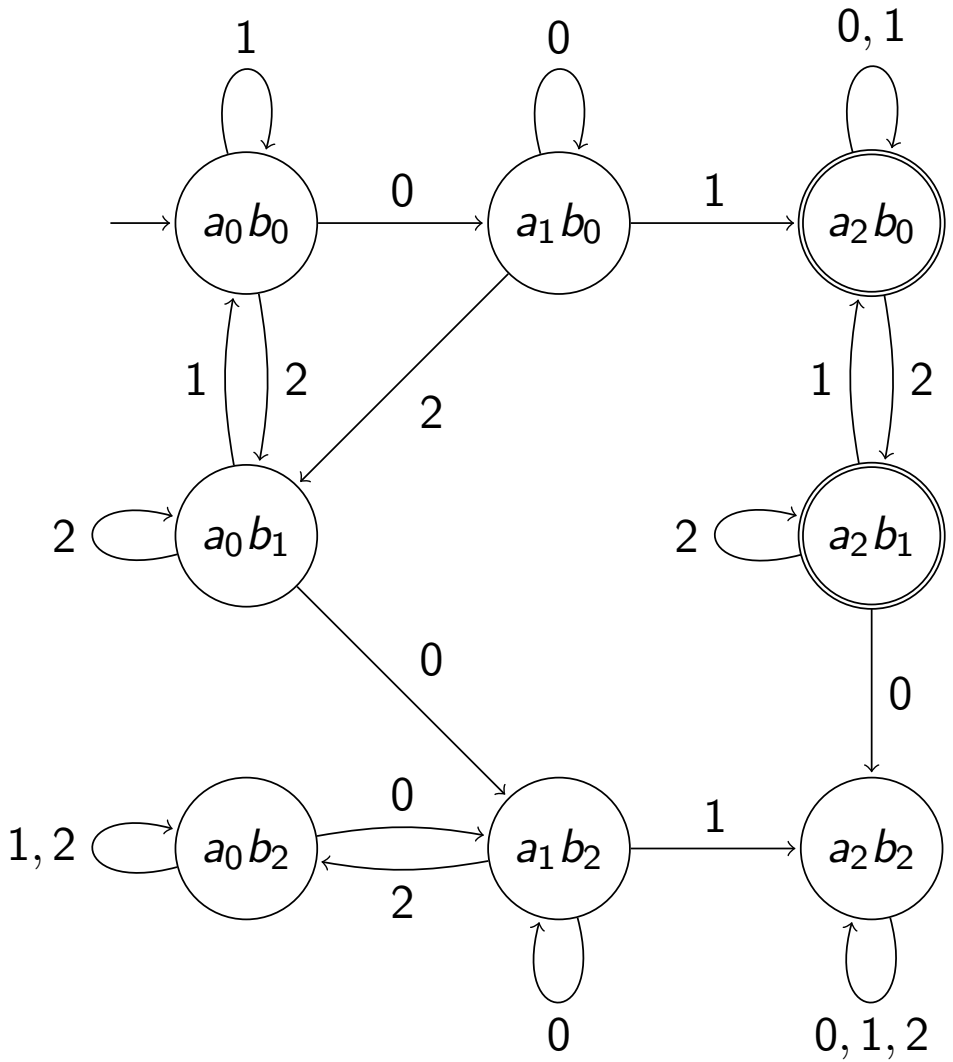
Problem 3 - Minimize the Product - Step 2

	a_0b_0	a_1b_0	a_0b_1	a_0b_2	a_1b_2	a_2b_2	a_2b_0	a_2b_1
a_0b_0		1	2	2	2	2	0	0
a_1b_0			1	1	1	1	0	0
a_0b_1							0	0
a_0b_2							0	0
a_1b_2							0	0
a_2b_2							0	0
a_2b_0								1
a_2b_1								



Problem 3 - Minimize the Product - Step 3

	a_0b_0	a_1b_0	a_0b_1	a_0b_2	a_1b_2	a_2b_2	a_2b_0	a_2b_1
a_0b_0		1	2	2	2	2	0	0
a_1b_0			1	1	1	1	0	0
a_0b_1				3	3	3	0	0
a_0b_2							0	0
a_1b_2							0	0
a_2b_2							0	0
a_2b_0								1
a_2b_1								

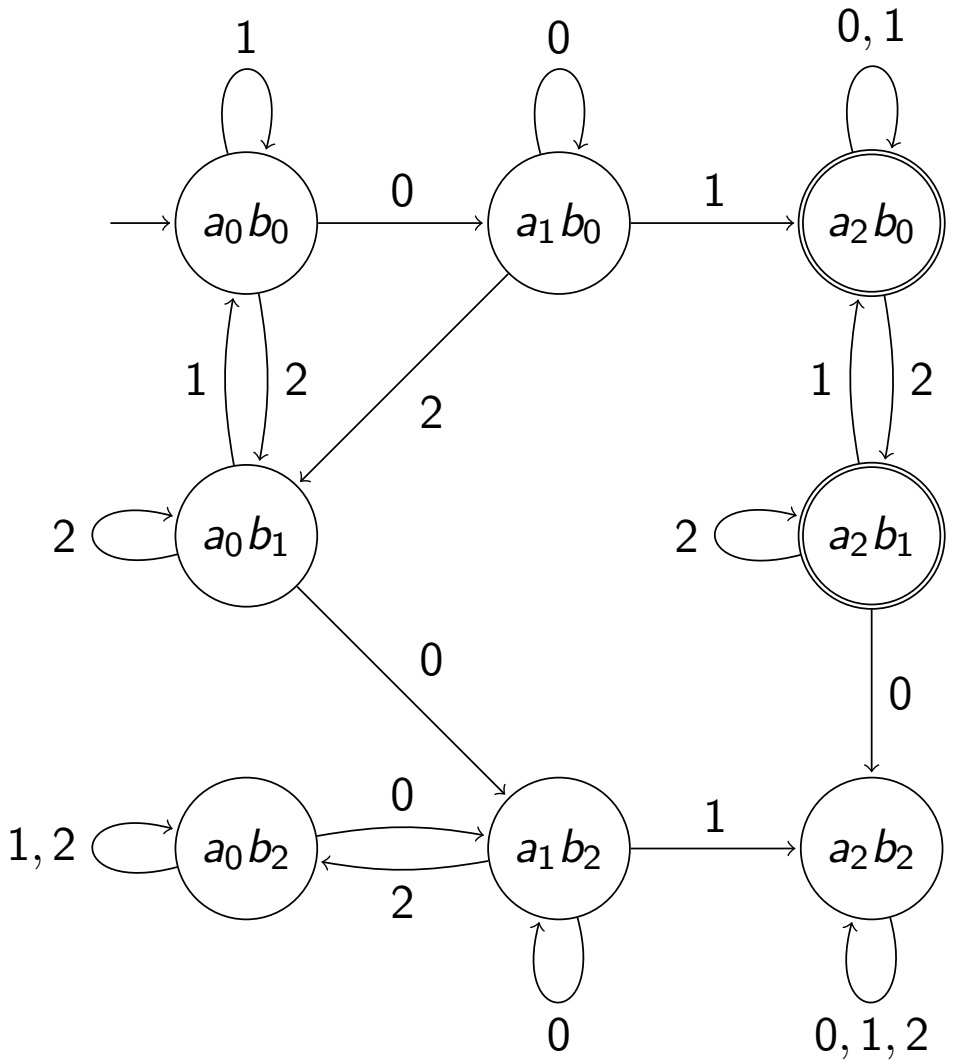


Problem 3 - Minimize the Product - Step 4

	a_0b_0	a_1b_0	a_0b_1	a_0b_2	a_1b_2	a_2b_2	a_2b_0	a_2b_1
a_0b_0		1	2	2	2	2	0	0
a_1b_0			1	1	1	1	0	0
a_0b_1				3	3	3	0	0
a_0b_2							0	0
a_1b_2							0	0
a_2b_2							0	0
a_2b_0								1
a_2b_1								

Equivalence Classes:

$\{a_0b_2, a_1b_2, a_2b_2\}$

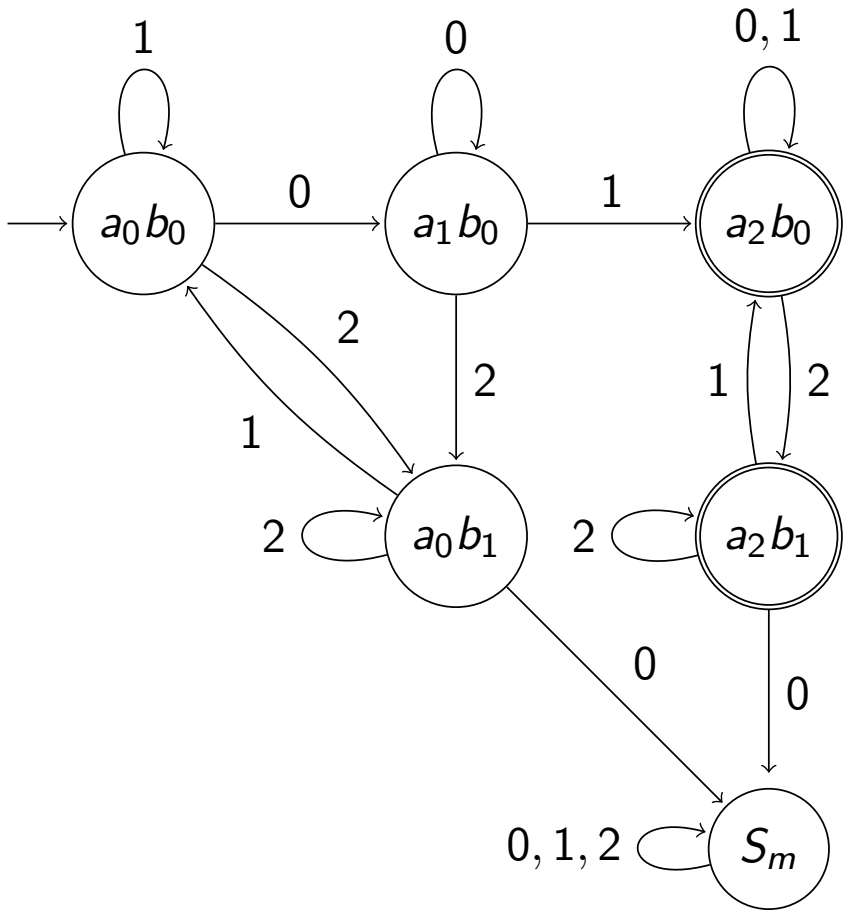


Problem 3 - Minimize the Product - Minimum DFA

	a_0b_0	a_1b_0	a_0b_1	a_0b_2	a_1b_2	a_2b_2	a_2b_0	a_2b_1
a_0b_0		1	2	2	2	2	0	0
a_1b_0			1	1	1	1	0	0
a_0b_1				3	3	3	0	0
a_0b_2							0	0
a_1b_2							0	0
a_2b_2							0	0
a_2b_0								1
a_2b_1								

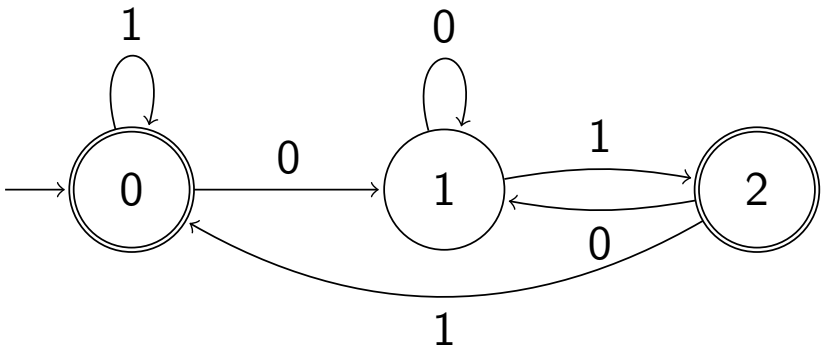
Equivalence Classes:

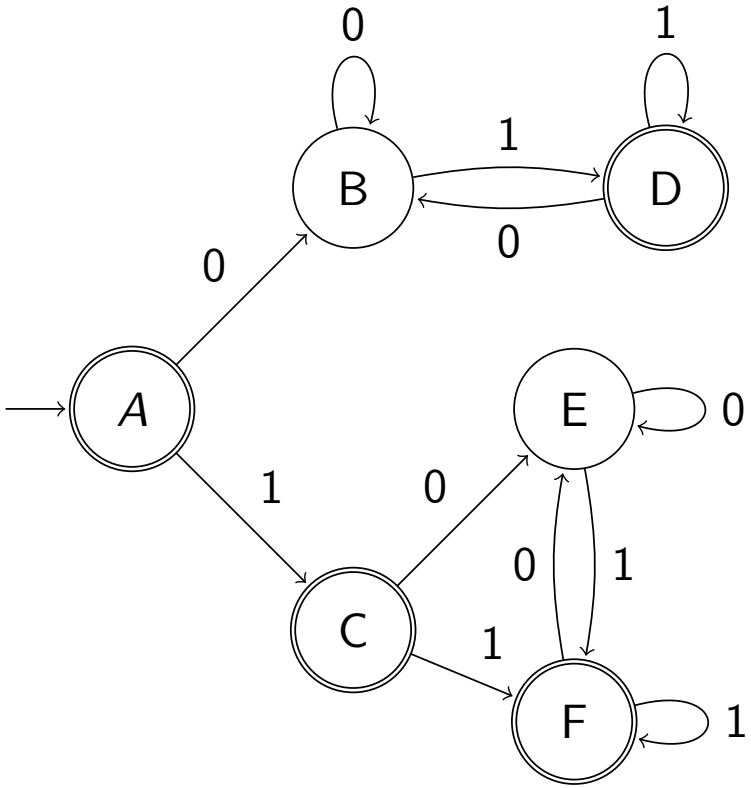
$\{a_0b_2, a_1b_2, a_2b_2\} = S_m$



Problem 4

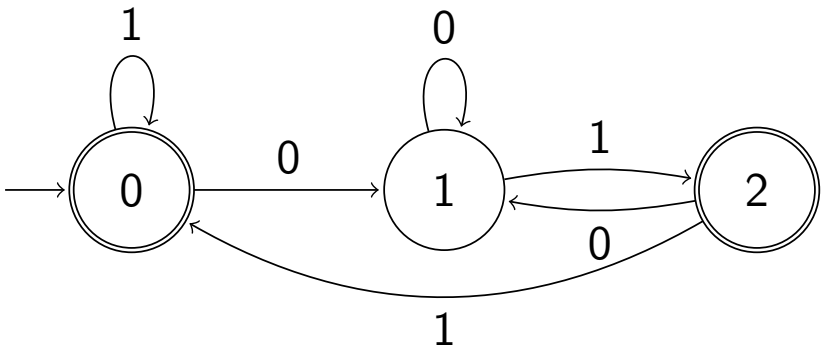
Determine if the given two DFA is equivalent or not.

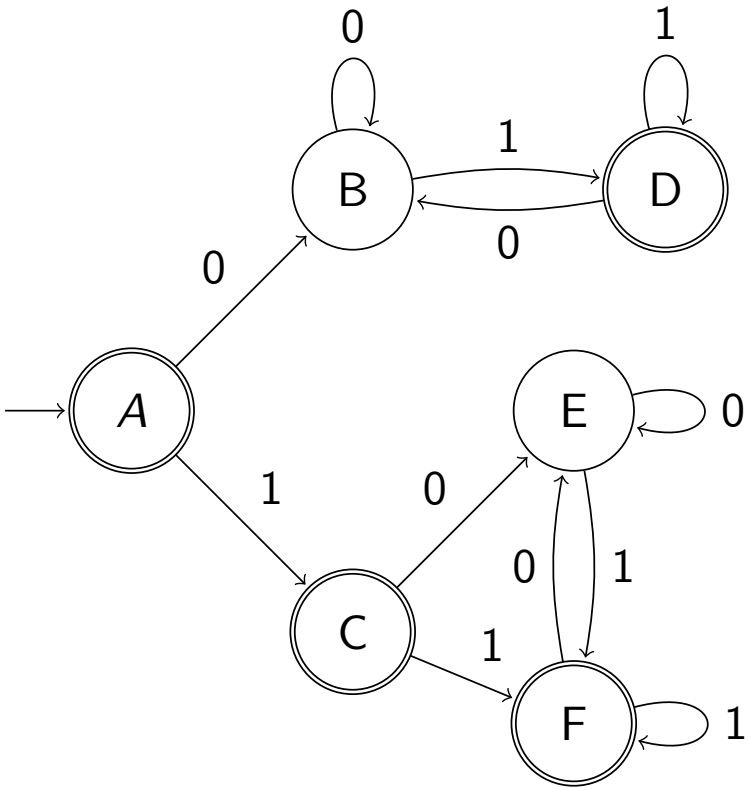




Problem 4 - Solution

Determine if the given two DFA is equivalent or not.

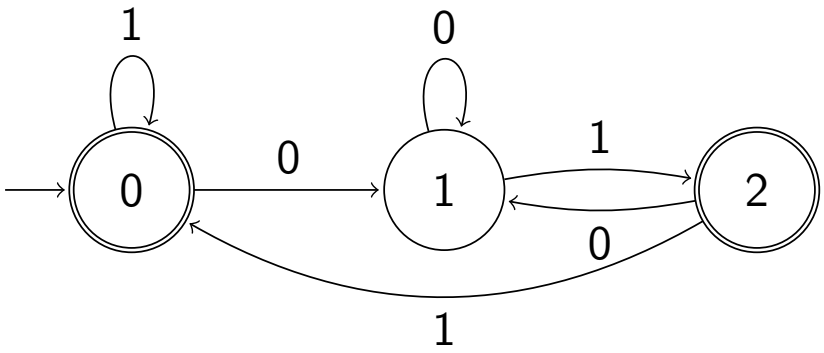


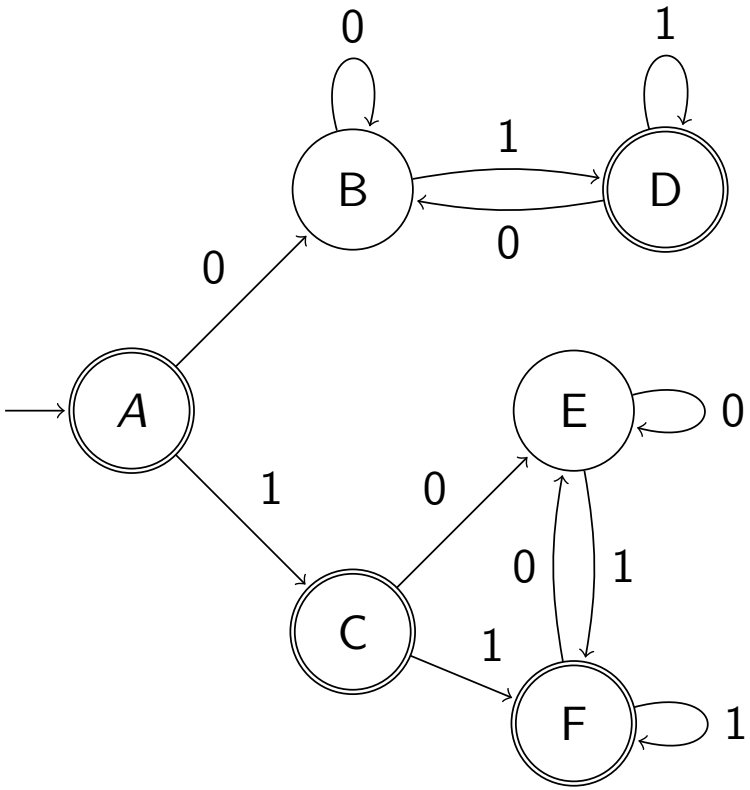


	B	E	1	A	C	D	E	F	0	2
B				0	0	0	0	0	0	0
E				0	0	0	0	0	0	0
1				0	0	0	0	0	0	0
A										
C										
D										
E										
F										
0										
2										

Problem 4 - Solution

Determine if the given two DFA is equivalent or not.

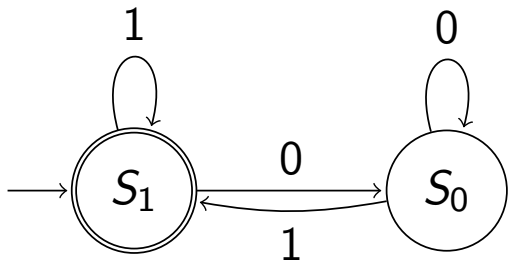




	B	E	1	A	C	D	E	F	0	2
B				0	0	0	0	0	0	0
E				0	0	0	0	0	0	0
1				0	0	0	0	0	0	0
A										
C										
D										
E										
F										
0										
2										

$$S_0 = \{B, E, 1\}$$

$$S_1 = \{A, C, D, E, F, 0, 2\}$$



Problem 4 - Guide to alternative solution

- In order to prove two DFA is equivalent, we can prove that two DFA accepts the same language by using the corresponding regular expressions.

Problem 4 - Guide to alternative solution

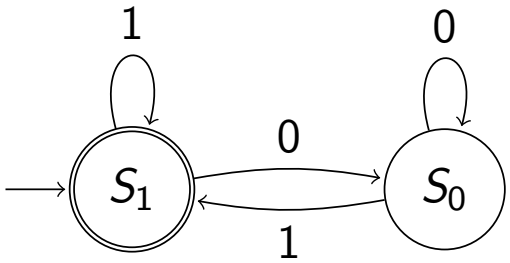
- In order to prove two DFA is equivalent, we can prove that two DFA accepts the same language by using the corresponding regular expressions.
- Get corresponding regular expressions for DFA using state elimination method, and prove that two expression is equivalent.

Problem 4 - Additional tutorial for altn. solution

- Both DFA's have initial states that are also final states. How can we perform state elimination in such case?

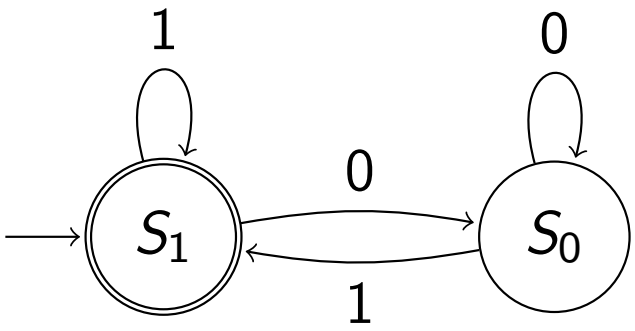
Problem 4 - Additional tutorial for altn. solution

- Both DFA's have initial states that are also final states. How can we perform state elimination in such case?
- Example: Give a regular expression for given FA using state elimination.



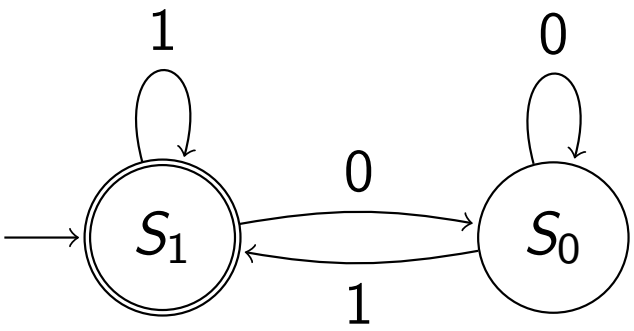
Problem 4 - Additional tutorial for altn. solution

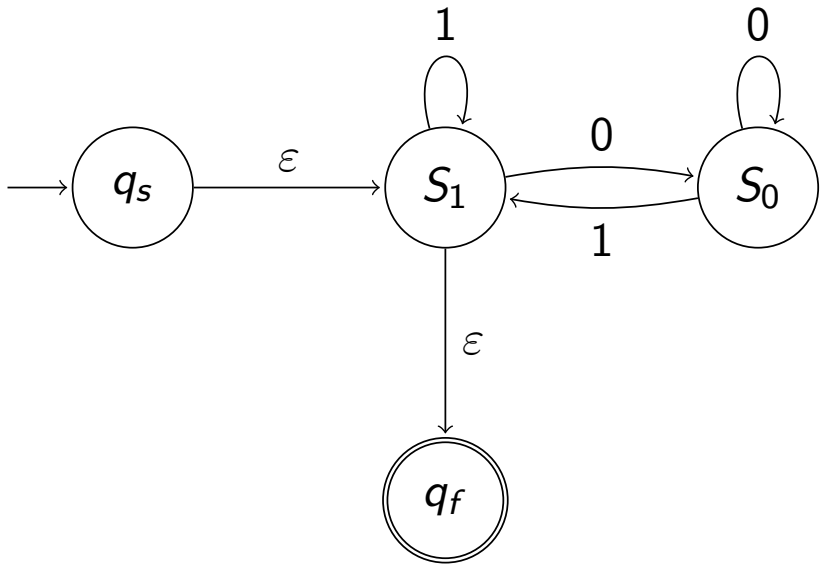
Example: Give a regular expression for given FA using state elimination.



Problem 4 - Additional tutorial for altn. solution

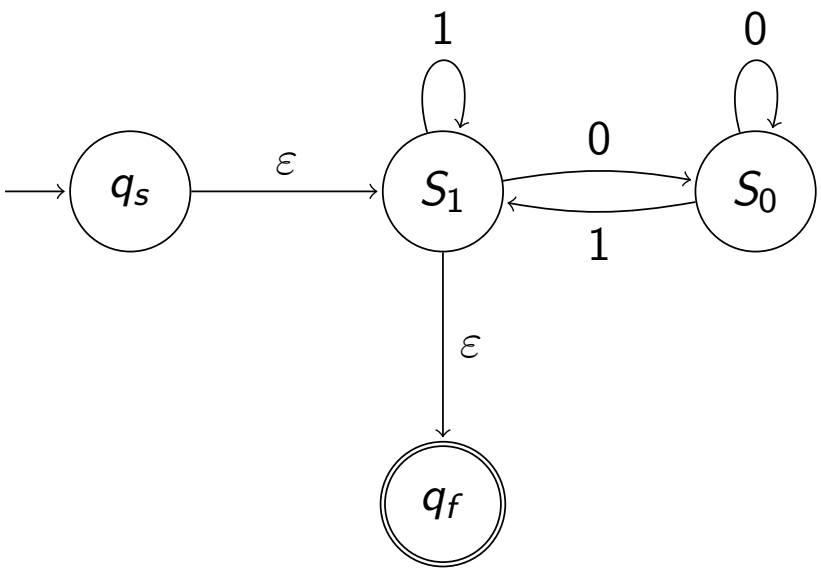
Example: Give a regular expression for given FA using state elimination.

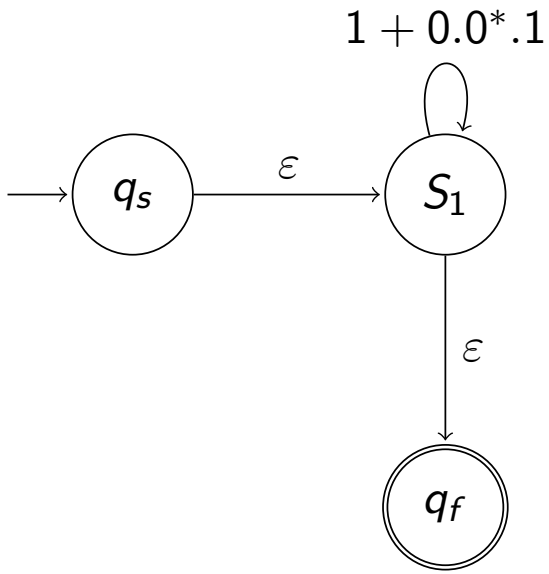




Problem 4 - Additional tutorial for altn. solution

Example: Give a regular expression for given FA using state elimination.
Eliminate S_0.





Problem 4 - Additional tutorial for altn. solution

Example: Give a regular expression for given FA using state elimination.
Eliminate S_1.