Ex. 1

Soit l'automate représenté par la table de transition suivante:

	a	b
$\rightarrow$ 1	2	3
$\leftarrow$ 2	2	2
$\leftarrow$ 3	4	3
$\leftarrow$ 4	4	3

Proposez un automate reconnaissant le même langage mais minimal en nombre d'états.

.... Answer ....

correction given in class

Ex. 2

Let's consider the FSA given by the following transition table

	a	b	c
$\rightarrow$ 1	3,5	2	1
2	5	2,3	
3		5	
4	3	2	4,5
$\leftarrow$ 5		5	

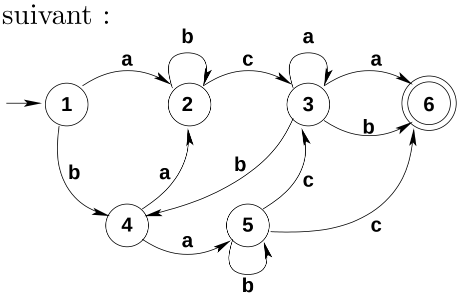
Propose a deterministic FSA recognizing the same language.
Would it be better to start with a complete FSA ?

.... Answer ....

correction given in class

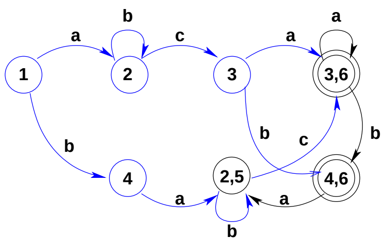
Ex. 3

Déterminiser l'automate suivant :



.... Answer ....

En appliquant l'algorithme de déterminisation, qui consiste, en partant de l'état initial, à créer au fur et à mesure du parcours des états correspondant à l'union des chemins possibles, on aboutit à l'automate représenté graphiquement par la figure suivante :



	a	b	c
$\rightarrow$ 1	2	4	/
2	/	2	3
4	(2,5)	/	/
3	(3,6)	(4,6)	/
(2,5)	/	(2,5)	(3,6)
$\leftarrow$ (3,6)	(3,6)	(4,6)	/
$\leftarrow$ (4,6)	(2,5)	/	/

Ex. 4

Proposer un automate sans ε-transition qui reconnaît le même langage que l'automate ci-dessous (on ne demande pas un automate déterministe).

	a	b	c	ε
$\rightarrow$ 1	1	3	5	5
2	3	2	1	
$\leftarrow$ 3			4	5 4
4	3		6	2
5	5	4,6	6	
$\leftarrow$ 6				

.... Answer ....

Automate de départ ... on calcule $\varepsilon+$

	a	b	c	ε	ε
$\rightarrow$ 1	1	3	5	5	5
2	3	2	1		
$\leftarrow$ 3			4	5 4	2,4,5
4	3		6	2	2
5	5	4,6	6		
$\leftarrow$ 6					

... et on « court-circuite »:

	a	b	c
$\rightarrow$ 1	1,5	3,4,6	5,6
2	3	2	1
$\leftarrow$ 3	3,5	2,4,6	1,4,6
4	3	2	1,6
5	5	4,6	6
$\leftarrow$ 6			

Il reste à vérifier le statut terminal/non terminal de chaque état: tous les états ayant un état terminal dans leur $\varepsilon+$ deviennent terminaux. Ici, pas de changement.

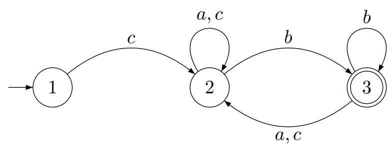
Ex. 5

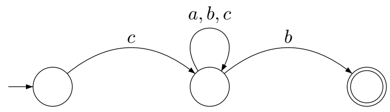
Soit $X = \{a, b, c\}$.

1. Proposer un automate déterministe (pas nécessairement complet) qui reconnaît le langage sur X^* de tous les mots qui commencent par c et se terminent par b.
2. Proposer un automate déterministe (pas nécessairement complet) qui reconnaît tous les mots de X^* qui comprennent le motif $b(a|c)b$.
3. Proposer un automate déterministe (pas nécessairement complet) qui reconnaît tous les mots de X^* qui comprennent le motif $b(a|c)b$ **et** commencent par c **et** se terminent par b.

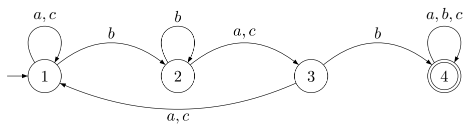
. . . . Answer . . . .

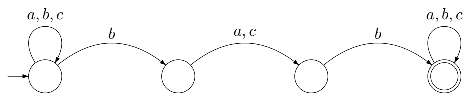
1. Il est assez facile de concevoir l'automate ≪ à main levée ≫, cela donne l'automate de gauche. Il était aussi possible de partir d'une version non déterministe (automate de droite) et de la déterminiser, le résultat est le même.



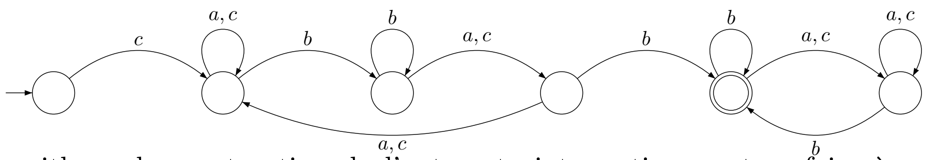


2. De la même façon, on peut tenter de proposer un automate ≪ à main levée ≫, ou passer par une version non déterministe à déterminiser. Dans ce cas, les résultat n'est pas tout à fait le même, l'algorithme de déterminisation ayant l'inconvénient d'ajouter des états (d'acceptation) redondants. Noter que l'on ne demandait pas de reconnaître les mots de la forme $b(a|c)b$, mais tous les mots dont bab ou bcb sont des facteurs. A gauche l'automate fait directement, à droite l'automate non déterministe qui peut servir de point de départ à la déterminisation.





3. Il est facile de voir que le langage reconnu dans cette question est l'**intersection** des langages des deux questions précédentes. Dès lors plusieurs options se présentent: soit, en négligeant les réponses aux questions précédentes, proposer un automate correspondant à la définition — on pouvait de nouveau le faire ≪ à main levée ≫ (automate ci-dessous) ou en passant par un automate non déterministe et une déterminisation [laissé en exercice], ou bien appliquer l'algorithme d'intersection à partir des deux automates précédents.



L'algorithme de construction de l'automate intersection peut se faire à partir d'automates non complets, à condition de considérer qu'une transition qui n'est pas définie dans au moins un des automates initiaux est non définie dans l'automate intersection.

Ici, si l'on part des deux automates ci-dessus (versions de gauche), on aboutit à la table de transition ci-dessous, qui correspond à l'automate précédent. En partant d'une version complétée des automates (il n'est nécessaire de compléter que le premier automate en ajoutant un état puits), deux états puits sont créés par l'algorithme.

	a	b	c
→ (1,1)			(2,1)
(2,1)	(2,1)	(3,2)	(2,1)
(3,2)	(2,3)	(3,2)	(2,3)
(2,3)	(2,1)	(3,4)	(2,1)
← (3,4)	(2,4)	(3,4)	(2,4)
(2,4)	(2,4)	(3,4)	(2,4)