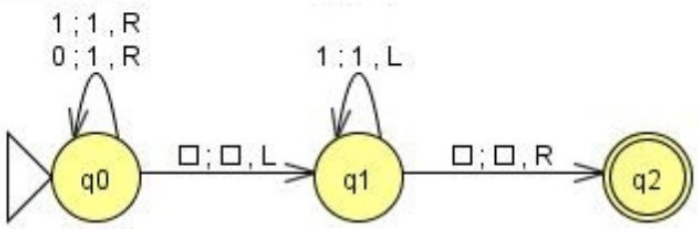
TD06 – Théorie des Langages

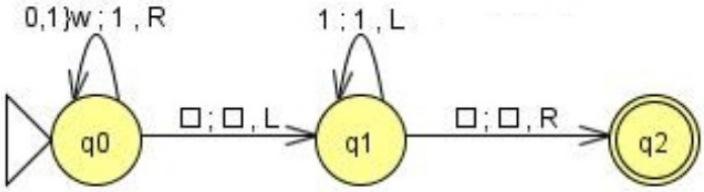
Exercice 1.

Question 1.

Solution.

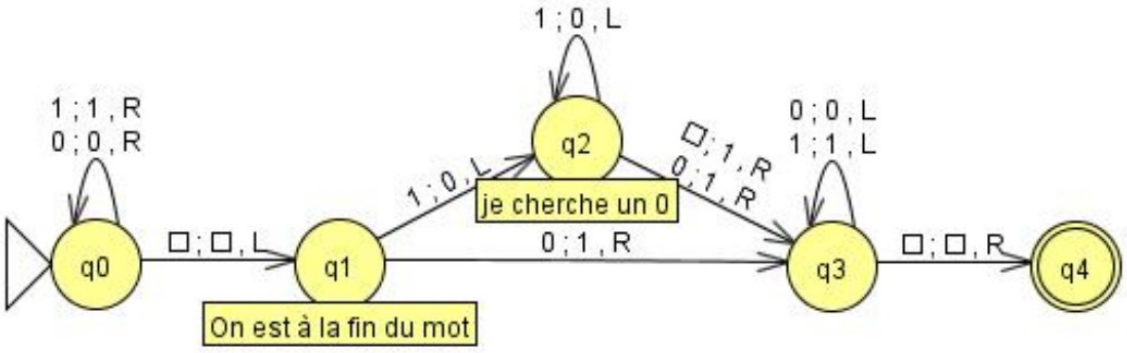


Une autre manière d'écrire cet automate sous JFLAP :



Question 2.

Solution. Pour cet automate, il suffit de chercher le dernier 0 et de le remplacer par un 1. Pour cela, il faut tout d'abord se placer à la fin du mot puis remonter jusqu'à ce que l'on trouve un 0, en remplaçant les 1 par des 0 au fur et à mesure. Si l'on remonte jusqu'au début du mot, on rajoute un 1 devant le mot.

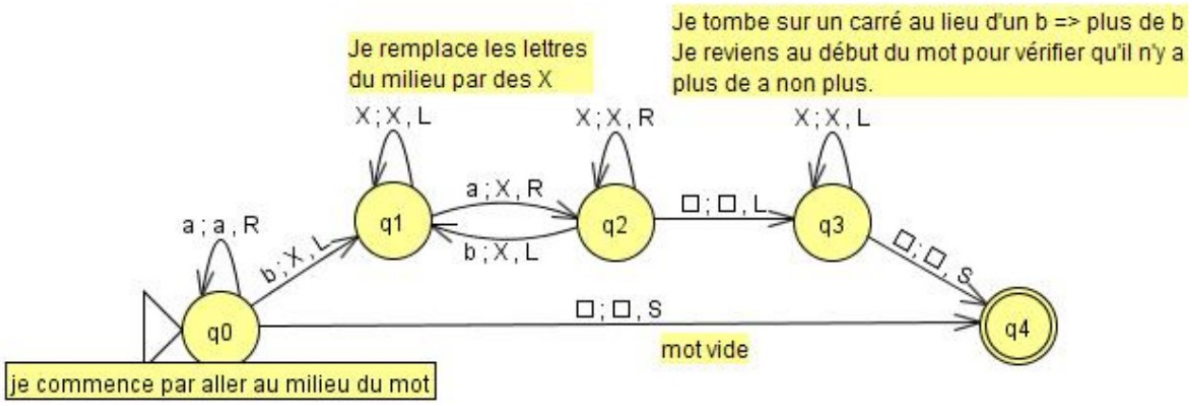


Exercice 2.

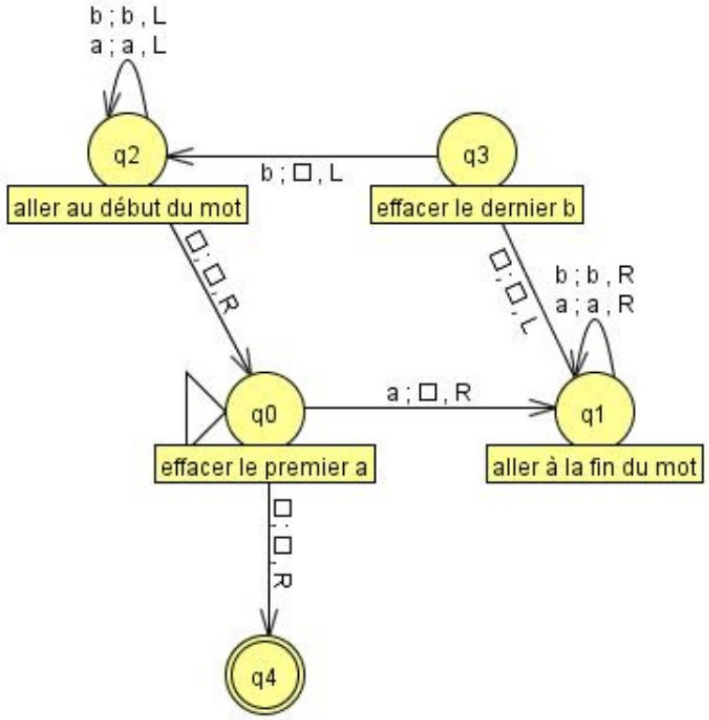
Question 1.

Question 1. Soit l'alphabet $A = \{a, b\}$.
Définir une machine de Turing permettant de reconnaître le langage $L = \{a^n b^n / n \in N\}$.
On rappelle que ce langage est de type 2 et qu'il peut donc être aussi reconnu par un automate à pile.

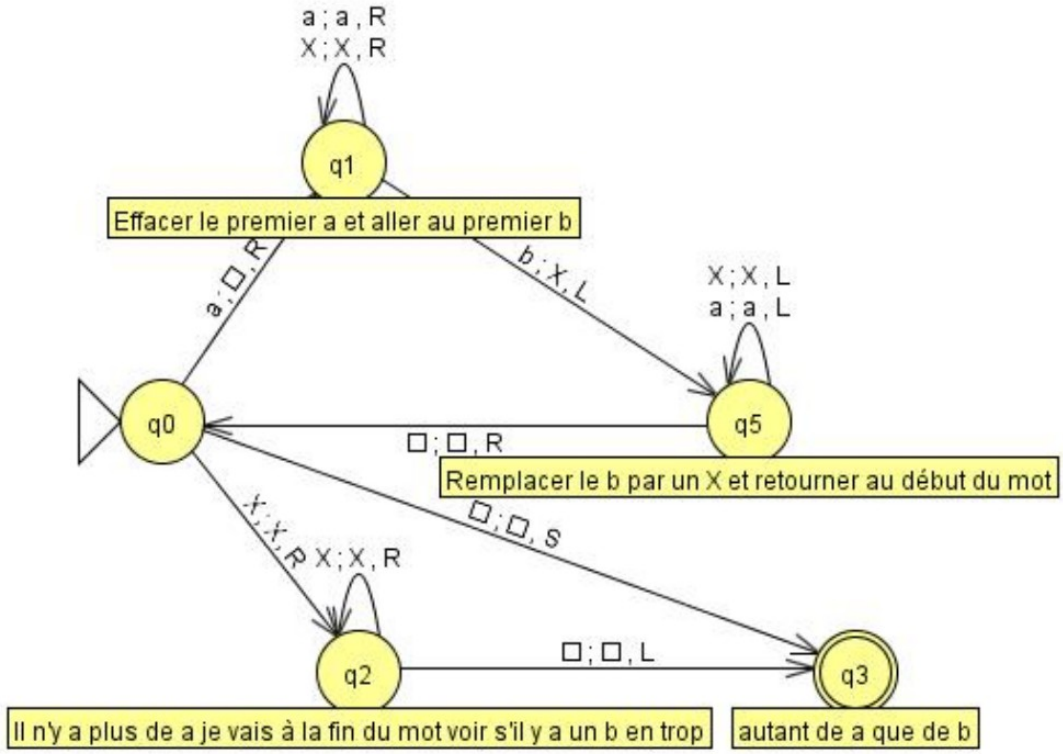
Solution. Dans cet automate, je pars du milieu du mot et je remplace le premier b par X, puis le dernier a par X, etc...



Une autre possibilité est de supprimer le premier a et le dernier b du mot, jusqu'à n'avoir plus que des □ :



Une autre possibilité : je supprime les a en début de mot et je remplace les premiers b trouvés par X :



Question 2.

Question 2. Soit l'alphabet $A = \{a, b, c\}$. Définir une machine de Turing permettant de reconnaître le langage $L = \{a^n b^n c^n / n \in N\}$.

Note : Cet exemple est l'un des plus "simples" qui soit un "vrai" langage de type 1 (i.e : non reconnaissable par un automate à pile).

Pour rappel, une grammaire possible de ce langage (contextuelle) est $G =< T, N, S, P >$ avec :

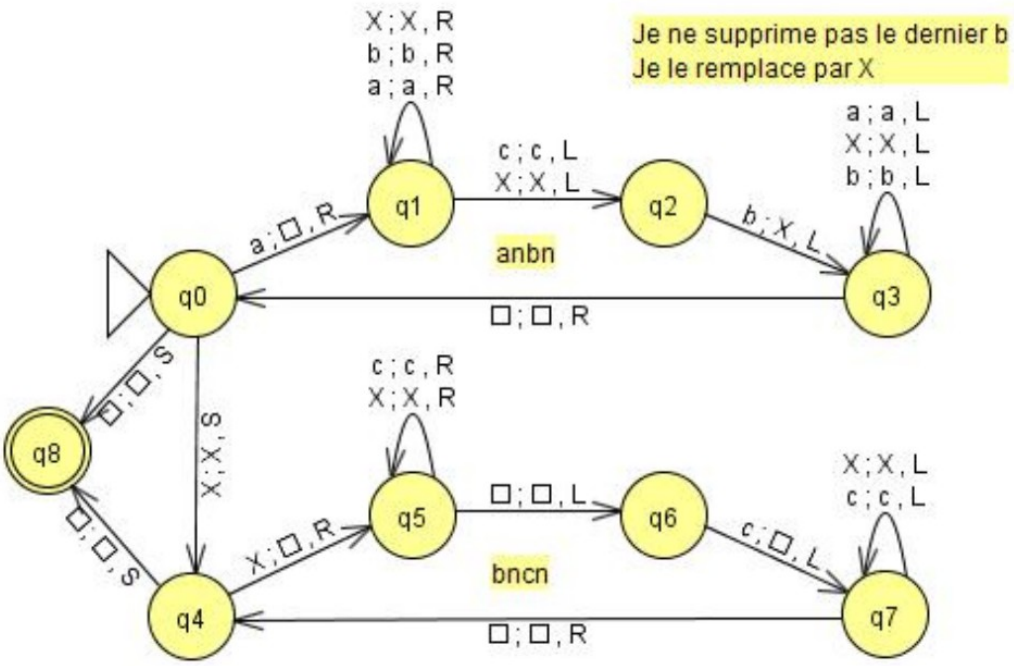
$T = \{a, b, c\}$

$N = \{S, T, Z, C\}$ les états de l'automate

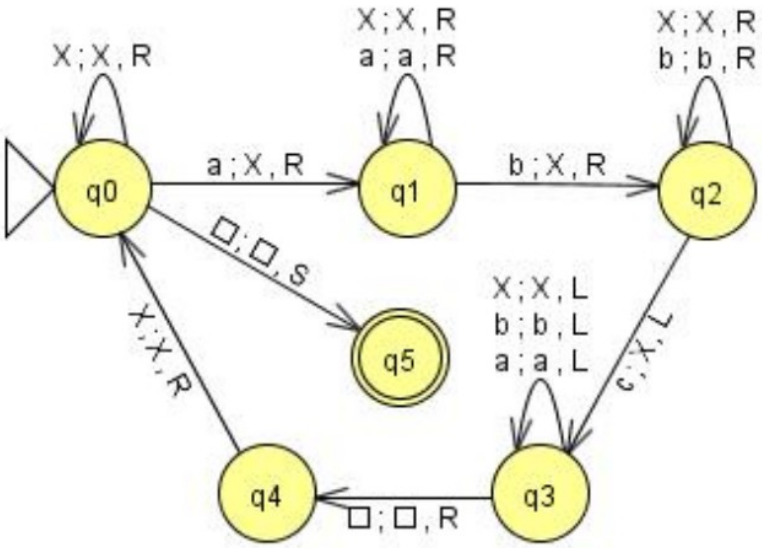
$S = S$

$$P = \begin{cases} S \rightarrow & T\ Z & // \ Z \text{ est utilisé pour repérer la fin du mot (puis à droite de } Z \text{ on n'aura que des } c) \\ T \rightarrow & a\ T\ b\ C \mid \varepsilon & // \text{ à chaque fois que l'on ajoute un } a, \text{ on ajoute un } b \text{ et un } C \\ & & \quad \text{(Problème : les } b \text{ et les } C \text{ seront alternés)} \\ C\ b \rightarrow & b\ C & // \text{ On désentrelace les } b \text{ et les } C \text{ en faisant passer les } C \text{ à droite des } b \\ C\ z \rightarrow & Z\ c & // \text{ si on a un } C \text{ à la fin du mot, on le remplace par un } c \text{ et on décale} \\ & & \quad \text{le caractère de fin de mot : } Z \\ b\ Z \rightarrow & b & // \text{ si le caractère de fin de mot atteint les } b, \text{ on n'a plus de } C \text{ à transformer,} \\ & & \quad \text{on supprime donc } Z \end{cases}$$

Solution. Pour cet exercice, il suffit de reprendre l'automate précédent et d'exécuter 2 fois la procédure pour tester $a^n b^n$ puis $b^n c^n$.



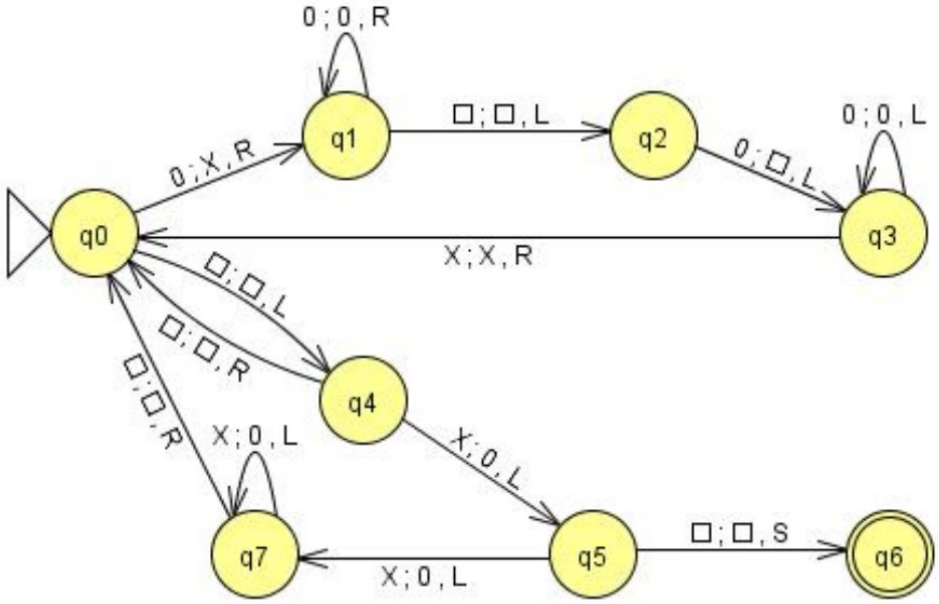
Une autre possibilité est de faire les 3 en un seul passage, en remplaçant tout par X :



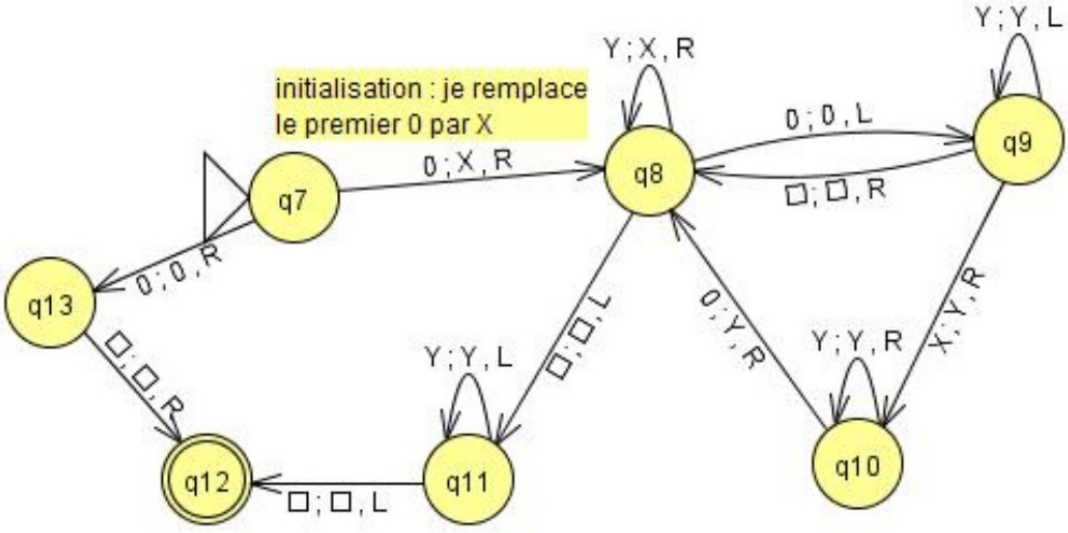
Question 3.

Solution. Pour cet exercice assez difficile, une fois l'astuce trouvée, l'automate est très simple. Il suffit en effet de diviser par 2 le nombre de 0 à chaque étape. Si on finit par n'avoir plus qu'un seul 0 alors le mot est accepté.

Pour diviser les 0 par 2, je remplace le premier par un X et je supprime le dernier.



Une autre possibilité est de raisonner par multiplication plutot que par division. On part du mot 0 qu'on transforme en X, puis on crée autant de Y à la place des 0 restant qu'il y a de X. Si on ne peut pas en créer assez, c'est qu'il n'y a pas suffisamment de 0.



Question 4.

Solution. A l'instar de la reconnaissance du langage $a^n b^n$, on vérifie si la première lettre du mot correspond bien à la derniere, puis on les supprime et on réitère jusqu'à n'avoir plus de lettres.