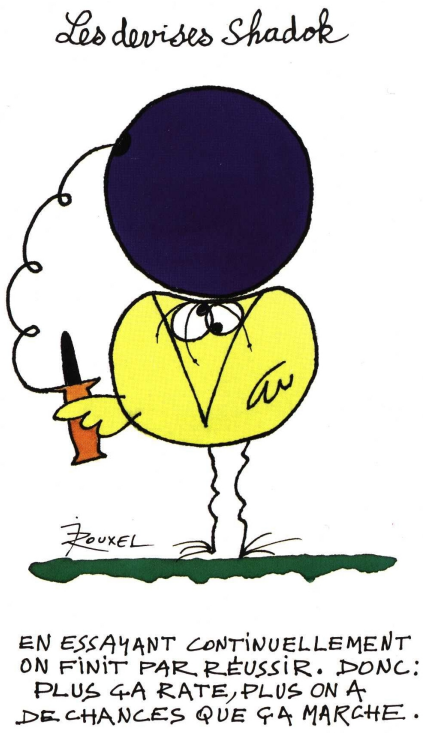
Examen de Théorie des Langages

Modalités

- Durée : **1h 30 minutes**
- Toutes vos affaires (sacs, vestes, trousse, etc.) doivent être placées à l'avant de la salle.
- Aucun document n'est autorisé.
- Aucune machine électronique ne doit se trouver sur vous ou à proximité, même éteinte.
- Aucun déplacement n'est autorisé.
- Aucune question au professeur n'est autorisé. Si vous pensez avoir détecté une erreur d'énoncé, expliquez les hypothèses que vous êtes amené à prendre pour continuer.
- Aucun échange, de quelque nature que ce soit, n'est autorisé.
- Le barème est donné à titre indicatif.
- **La clarté et la précision de la rédaction seront prises en compte dans l'évaluation.**



Rappels et notations

Une anagramme d'un mot w est un mot obtenu par une permutation des lettres de w. Par exemple, Pascal Obispo est une anagramme de Pablo Picasso (et réciproquement).

Exercice 1. Langages réguliers (7 points)

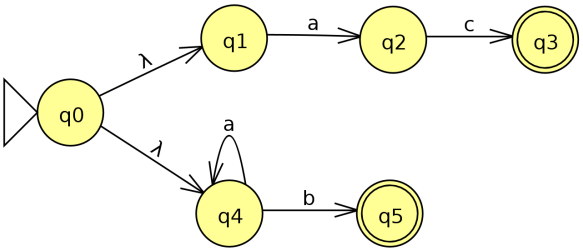
Soit le langage $L = a^*b + ac$ sur l'alphabet $A = \{a, b, c\}$.

Question 1. Lister tous les mots de L dont la taille est inférieure ou égale à 3.

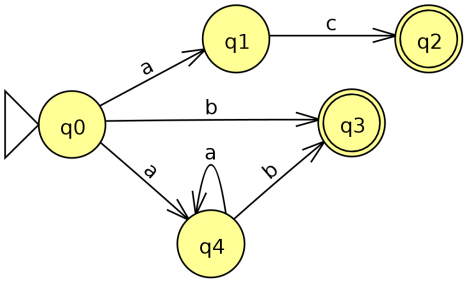
Solution. ac, b, ab, aab.

Question 2. Construire un automate fini non déterministe qui reconnaît les mots du langage L.

Solution. L'automate suivant convient.
Il n'est pas déterministe car il contient des ε-transitions (flèches étiquetées λ) à partir de l'état q0.



On pouvait aussi proposer :



Il n'est pas non plus déterministe car deux flèches étiquetées a partent de l'état q0.

Question 3. Donner la mise en équation de l'automate et résoudre ce système.

Solution. Le système d'équations de l'automate précédent est :

$$\begin{aligned} L_0 &= aL_1 + bL_3 + aL_4 \\ L_1 &= cL_2 \\ L_2 &= \varepsilon \\ L_3 &= \varepsilon \\ L_4 &= aL_4 + bL_3 \end{aligned}$$

On a $L_4 = aL_4 + b$ d'où, par le lemme d'Arden, $L_4 = a^*b$. En substituant dans la première equation, on obtient :

$$L_0 = ac + b + aa^*b = (\varepsilon + aa^*)b + ac = a^*b + ac$$

Question 4. Effectuer la méthode des quotients gauches sur le langage obtenu et conclure.

Solution. On calcule les quotients gauche de L :

$$\begin{aligned}
a^{-1}L &= a^{-1}(a^*b + ac) = a^{-1}(a^*b) + a^{-1}(ac) \\
&= a^{-1}(a^*)b + a^{-1}(b) + a^{-1}(a)c \\
&= a^*b + \varepsilon + \varepsilon c \\
&= a^*b + c \\
b^{-1}L &= b^{-1}(a^*b + ac) = b^{-1}(a^*b) + b^{-1}(ac) \\
&= b^{-1}(a^*)b + b^{-1}(b) + b^{-1}(a)c \\
&= \emptyset + \varepsilon + \emptyset c \\
&= \varepsilon \\
c^{-1}L &= c^{-1}(a^*b + ac) = c^{-1}(a^*b) + c^{-1}(ac) \\
&= c^{-1}(a^*)b + c^{-1}(b) + c^{-1}(a)c \qquad = \emptyset
\end{aligned}$$

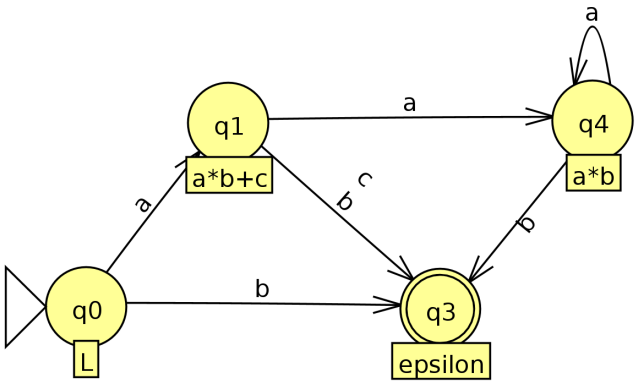
L'état correspondant à $\emptyset$ est un état puit, on peut simplement ignorer ces transitions.

L'état correspondant à ϵ est un état n'acceptant que le mot vide. Il est donc terminal mais n'a aucune transition sortante.

On génère ces nouveaux états et on recalcule les quotients gauches correspondants.

$$\begin{aligned}
a^{-1}(a^*b + c) &= a^{-1}(a^*b) + a^{-1}(c) = a^{-1}(a^*)b + a^{-1}(b) + a^{-1}(c) = a^*b + \emptyset + \emptyset = a^*b \\
b^{-1}(a^*b + c) &= b^{-1}(a^*b) + b^{-1}(c) = b^{-1}(a^*)b + b^{-1}(b) + b^{-1}(c) = \emptyset + \varepsilon + \emptyset = \varepsilon \\
c^{-1}(a^*b + c) &= c^{-1}(a^*b) + c^{-1}(c) = c^{-1}(a^*)b + c^{-1}(b) + c^{-1}(c) = \emptyset + \emptyset + \varepsilon = \varepsilon
\end{aligned}$$

Le reste de la résolution fournit l'automate suivant :



Exercice 2. Langages algébriques (7 points)

Soit le langage $L = a^n b^* c^n$ sur l'alphabet $A = \{a, b, c\}$.

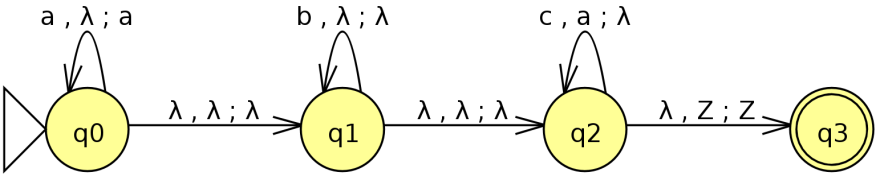
$$L = \{a^n\ w\ c^n\ /\ n \in \mathbb{N} \text{ et } w \in b^*\} = \{a^n\ b^p\ c^n\ /\ n \in \mathbb{N} \text{ et } p \in \mathbb{N}\}$$

Question 1. Lister tous les mots de L dont la taille est inférieure ou égale à 3.

Solution. ε, b, bb, bbb (pour $n = 0$), ac, abc (pour $n = 1$).

Question 2. Construire un automate à pile qui reconnaît les mots du langage L.

Solution. L'automate suivant convient.



On empile le symbole a sur la pile pour chaque symbole a lu dans le mot (état q0), puis on lit autant de symbole b que nécessaire sans toucher à la pile (état q1) et enfin on lit les symboles c en dépilant les symboles a de la pile. On passe dans l'état terminant quand on atteint le fond de la pile (au moins autant de c que de a), le mot est alors accepté si et seulement si il ne reste plus de caractères à lire (pas plus de c que de a).

Question 3. Écrire une grammaire sous forme normale de Chomsky qui reconnaît ce langage.

Solution. La grammaire $G =< T, N, S, P >$ convient avec :

$$\begin{aligned} T &= \{a, b, c\} \\ N &= \{S, T\} \\ S &= S \\ P &= \left\{ \begin{array}{l} S \to aSc \mid T \\ T \to \varepsilon \mid bT \end{array} \right\} \end{aligned}$$

Comme celle-ci n'est pas sous forme normale de Chomsky, on applique l'algorithme de normalisation :

$$\begin{aligned} N &= \{S, T, U, A, B, C\} \\ P &= \left\{ \begin{array}{l} S \to AU \mid BT \mid b \mid \varepsilon \\ U \to SC \\ T \to BT \mid b \\ A \to a \\ B \to b \\ C \to c \end{array} \right\} \end{aligned}$$

Cette grammaire est bien sous forme normalisée de Chomsky car toutes les règles sont de la forme

$$\begin{aligned} X &\to YZ \\ X &\to x \\ S &\to \varepsilon \end{aligned}$$

Question 4. Appliquer l'algorithme CKY pour vérifier l'appartenance du mot *aabcc* à ce langage.

Solution. On peut construire le tableau suivant en utilisant l'algorithme CKY.

5	S				
4	$\emptyset$	U			
3	$\emptyset$	S	$\emptyset$		
2	$\emptyset$	$\emptyset$	U	$\emptyset$	
1	A	A	T, S, B	C	C
input	a	a	b	c	c

On peut produire le mot *aabcc* à partir des symboles de la dernière case du tableau. Comme cette case contient S, alors le mot considéré est produit par l'axiome et appartient donc au langage L.

Exercice 3. Langages récursivement énumérables (6 points)

Soit le langage L des mots $w = m_1 \square m_2$ sur l'alphabet $A = \{0, 1\}$ tels que m_1 est une anagramme de m_2.

$$L = \{m_1 \square m_2 \ / \ m_1 \in A^*, \ m_2 \in A^* \text{ et } m_1 \text{ est une anagramme de } m_2\}$$

Question 1. Construire une machine de Turing qui reconnaît les mots du langage L.

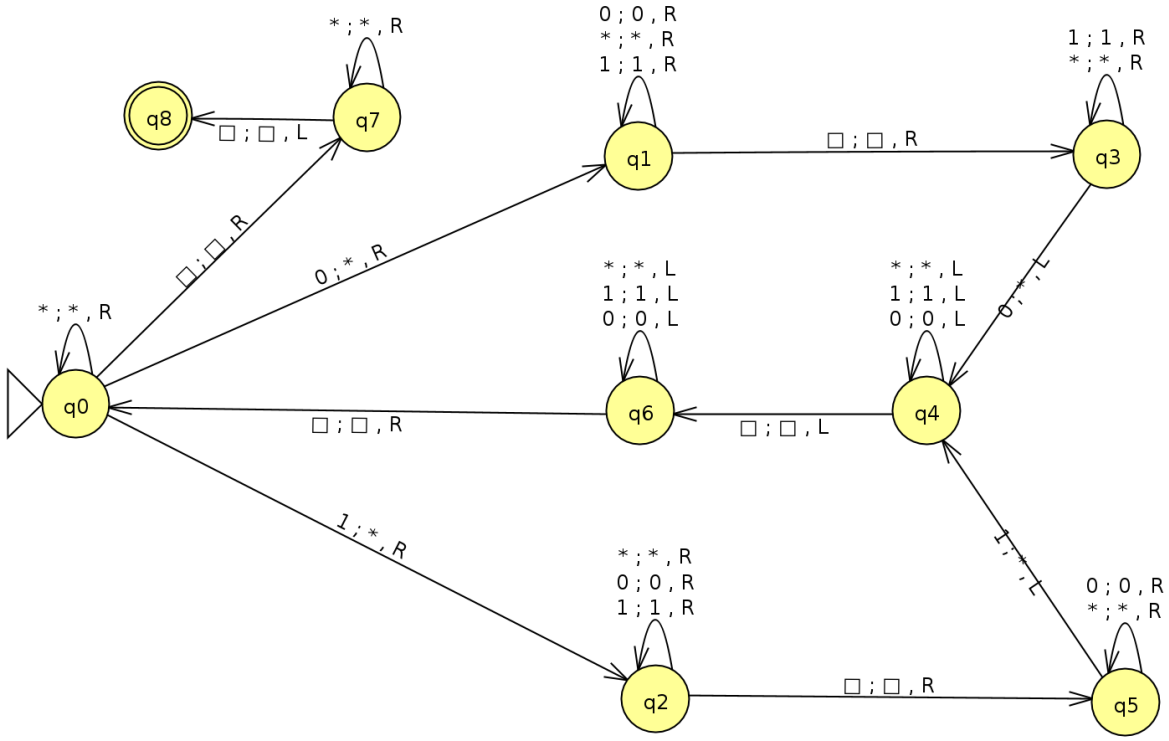
Solution. Cette machine de Turing est utilisée comme un accepteur, elle peut donc utiliser le ruban à sa guise pour vérifier que les deux mots m_1 et m_2, séparés par une unique case blanche, $\square$ sont bien des anagrammes l'un de l'autre.

Pour cela une façon de faire est la suivante :

1. On cherche un symbole binaire (0 ou 1) dans le mot de gauche, m_1 (état q0).
2. On le remplace par un symbole spécial (*) pour être sur de ne pas le relire (transitions q0 $\rightarrow$ q1 et q0 $\rightarrow$ q2).
3. On déplace la tête de lecture sur le mot de droite, m_2 (états q1 et q2).
4. On cherche le même symbole binaire (états q3 et q5).
5. On le remplace par le symbole spécial (transitions vers q4).
6. On retourne sur le mot de gauche, m_1 (état q4).
7. On se place au début (état q6) et on recommence (transition vers q0).

Si l'étape 1 échoue à trouver un symbole binaire, alors c'est que le mot de gauche à été entièrement lu, on vérifie alors que le mot de droite a aussi été entièrement lu et ne contient plus que des symboles spéciaux (état q7). Si c'est le cas on termine et on accepte (état q8).

Si l'étape 4 échoue alors c'est que le mot de droite, m_2, ne contient pas autant du symbole cherché que le mot de gauche, m_1. Donc on échoue : absence de transition étiquetée $\square$ à partir des états q3 et q5.



Question 2. Écrire une grammaire de type 0 qui reconnaît ce langage.

Solution. La grammaire $G =< T, N, S, P >$ convient avec :

$$\begin{aligned} T &= \{0, 1, \square\} \\ N &= \{S\} \\ S &= S \\ P &= \left\{ \begin{array}{l} S \to 0S0 \mid 1S1 \mid \square \\ 01 \to 10 \\ 10 \to 01 \end{array} \right\} \end{aligned}$$

Les règles pour S garantissent qu'il y a autant de 0 et de 1 de chaque côté du séparateur $\square$.

Les deux dernières règles permettent de réarranger ces symboles n'importe comment à droite et à gauche.

Question 3. Donner une condition pour que votre grammaire soit de type 1 ?

Solution. Pour être de type 1, il faudrait que toutes les règles de notre grammaire soient de la forme $uXv \to uwv$.

Remarque : cependant notre grammaire ne contient que des règles croissantes, elle pourrait donc être convertie en une grammaire de type 1, $G' =< T, N', S, P' >$, avec :

$$\begin{aligned} N' &= \{S, U, T\} \\ P' &= \left\{ \begin{array}{l} S \to 0S0 \mid 1S1 \mid \square \\ 01 \to U1 \\ U1 \to U0 \\ U0 \to 10 \\ 10 \to 1T \\ 1T \to 0T \\ 0T \to 01 \end{array} \right\} \end{aligned}$$

Exercice 4. Bonus

Quelle anagramme de "logarithme" est issue du célèbre mathématicien Al-Khwârizmî ?

Solution. algorithme, évidemment.