INFO-F-302
Informatique Fondamentale
Logique du premier ordre

Prof. Emmanuel Filiot

Exercice 1 Construire un automate qui reconnaît le langage :

1. $L = \{\varepsilon\}$.
2. $L_k = \{w \in \{a, b\}^* : |w| \in k\mathbb{N}\}$, i.e. la taille de w est un multiple de k.
3. $L = \{w \in \{a, b\}^* : n_a(w)$ est impair $\}$, i.e. w a un nombre impair de a.
4. $L = \{w \in \{a, b, c\}^* : abc$ est un facteur de $w\}$.
5. $L = \{w \in \{a, b, c\}^* : abc$ n'est pas un sous-mot de $w\}$.
6. $L = \{w \in (\{0, 1\}^3)^* : \pi_1(w) + \pi_2(w) = \pi_3(w)\}$, i.e. les séquences de vecteurs binaires de dimension 3 où la somme binaire de la première et deuxième dimensions est égale a la troisième (des poids faibles aux poids forts, puis même question des forts aux faibles).

$$w = \begin{pmatrix}0\\1\\1\end{pmatrix}_1 \begin{pmatrix}1\\1\\0\end{pmatrix}_2 \begin{pmatrix}0\\0\\1\end{pmatrix}_4 \begin{pmatrix}1\\0\\1\end{pmatrix}_8 \quad 1010 + 0011 = 1101$$

Exercice 2 Construire un automate minimal équivalent aux automates suivants :

1. l'automate a,
2. l'automate qui reconnaît le langage $L = \{bananas, ananas, nanas\}$,
3. l'automate c,
4. l'automate d,
5. l'automate e,
6. l'automate f.

Exercice 3 Construire un automate pour chaque expression régulière :

1. $(a + ab)$,
2. $(a + ab)^*$.

Exercice 4 Donner l'expression régulière et l'automate pour les langages ($\Sigma = \{0, 1\}$) suivants :

1. $\{w : w$ a exactement deux 0 $\}$,
2. $\{w : w$ a au moins deux 0 $\}$,
3. $\{w : w$ a un nombre pair de 0 $\}$,
4. $\{w : w$ n'a pas de 0 $\}$,
5. $\{w : w$ est un identifiant valide dans le langage de programmation `C` $\}$. Ici Σ contient toutes les lettres et symboles sur votre clavier.

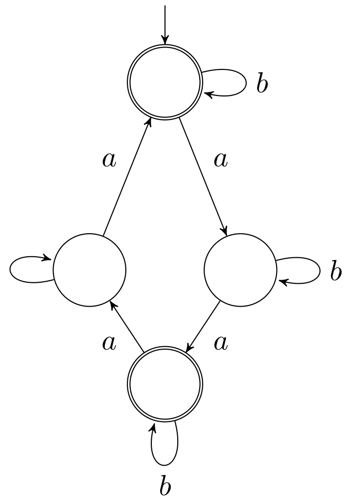


FIGURE 1 – Automate a

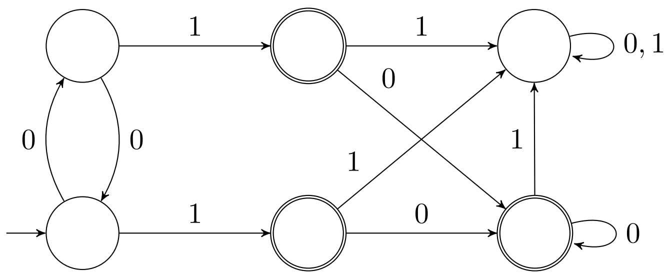


FIGURE 2 – Automate c

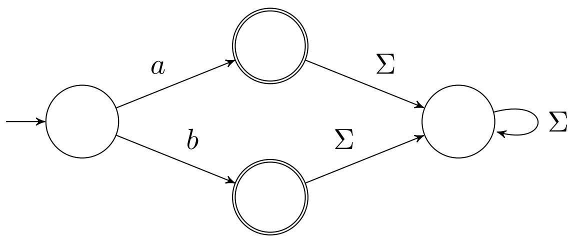


FIGURE 3 – Automate d

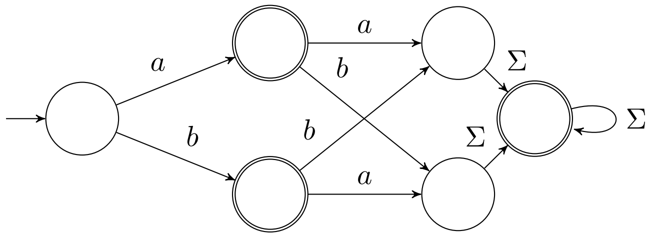


FIGURE 4 – Automate e

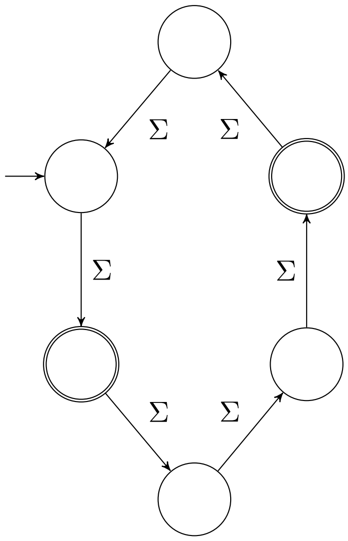


FIGURE 5 – Automate f

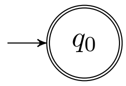
Exercice 5 Construire l'expression régulière pour $L(e_1) \cap L(e_2)$ pour chaque pair e_1, e_2 ci dessous :

1. $e_1 = a(a+b)^*$, $e_2 = (a+b)^*b$;
2. $e_1 = (b^*ab^*ab^*)^*$, $e_2 = a(a+b)^*$;
3. $e_1 = (b^*ab^*ab^*)^*$, $e_2 = (b^*ab^*ab^*ab^*)^*$.

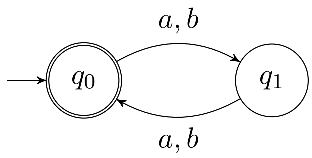
Correction

Exercice 1

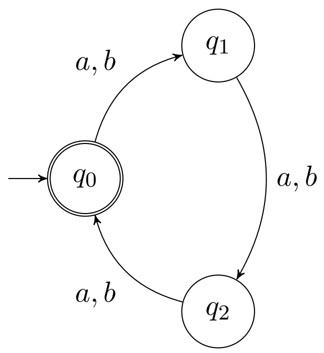
1. $L = \{\varepsilon\}$.



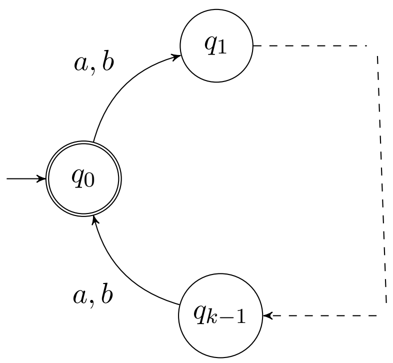
2. $L_k = \{w \in \{a,b\}^* : |w| \in k\mathbb{N}\}$, i.e. la taille de w est un multiple de k.
   Pour $k = 2$



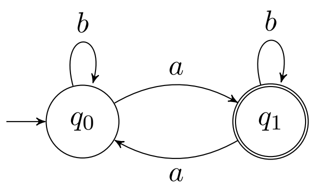
   Pour $k = 3$



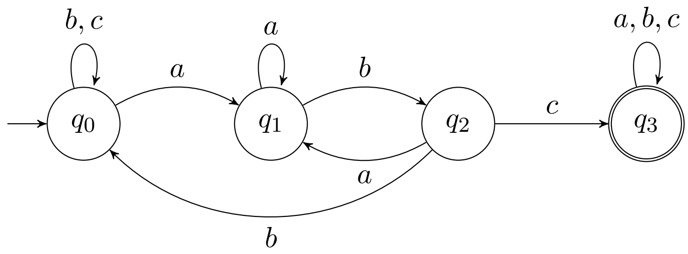
   Cas général (k états)



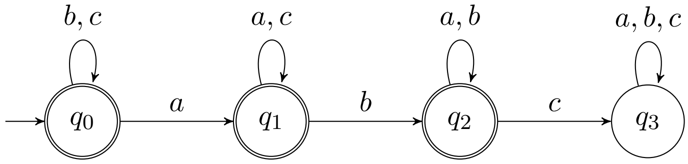
3. $L = \{w \in \{a,b\}^* : n_a(w)$ est impair $\}$, i.e. w a un nombre impair de a.



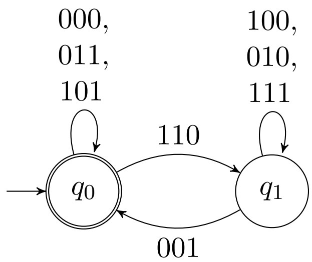
4. $L = \{w \in \{a,b,c\}^* : abc$ est un facteur de $w\}$.



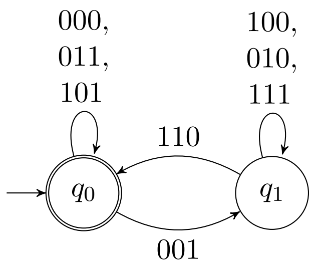
5. $L = \{w \in \{a, b, c\}^* : abc$ n'est pas un sous-mot de $w\}$.



6. $L = \{w \in (\{0,1\}^3)^* : \pi_1(w) + \pi_2(w) = \pi_3(w)\}$, i.e. les séquences de vecteurs binaires de dimension 3 où la somme binaire de la première et deuxième dimensions est égale a la troisième Poids faibles aux poids forts : on se rappelle de la retenue si nécessaire (en q_1).

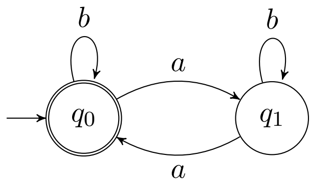
   Poids forts aux poids faibles : à l'inverse, si on a lu une addition qui a besoin d'une retenue, on la promet (en q_1) et on s'attend à ce que la lettre suivante la fournisse. On peut aussi voir ceci comme une inversion du langage précédent.

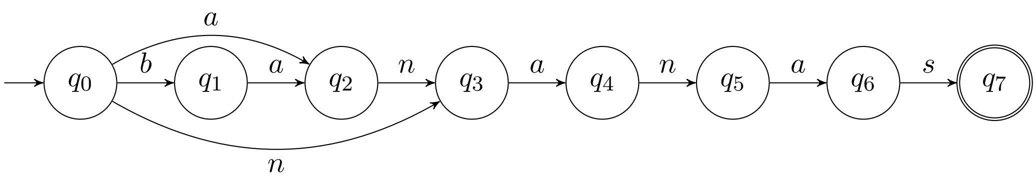
Exercice 2

1. l'automate a reconnaît l'ensemble des mots avec un nombre pair de a. Pour s'en convaincre, on pourrait constater que les deux états non-terminaux et les deux états terminaux reconnaissent deux-à-deux les mêmes mots.

   Alternativement, on peut reconnaître la construction de 1.3 mais avec un cycle de taille 4 plutôt que 2. On peut dire que l'automate accepte les mots avec un nombre de a congru à 0 ou 2 modulo 4, i.e un nombre pair.



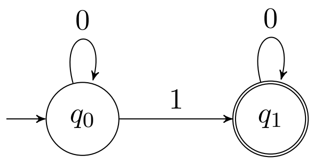
2. l'automate qui reconnaît le langage $L = \{bananas, ananas, nanas\}$ s'obtient en "factorisant" les états qui reconnaissent les mots $s, as, nas, anas, nanas, ananas$ et en envoyant l'état initial sur un de ce états selon la première lettre lue.



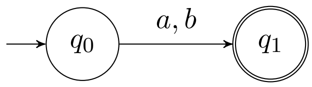
3. l'automate c reconnait 0^*10^*. Pour s'en convaincre : l'état en haut à droite est un état puits, on peut l'ignorer.

   Les trois états terminaux acceptent tous les mots de 0^*, on peut les fusionner.

   Les deux états de gauche ont donc maintenant la même fonction : rester à gauche tant qu'on lit un 0 puis partir dans les états acceptants si on lit un 1. Ils reconnaissent donc tous les deux 0^*10^* et peuvent être fusionnés.



4. l'automate d reconnait le langage $a + b$. L'état de droite est un état puits. Les deux états terminaux ne sont pas des états puits inutiles (ils reconnaissent ε), mais ce sont des culs-de-sac et on peut les fusionner.



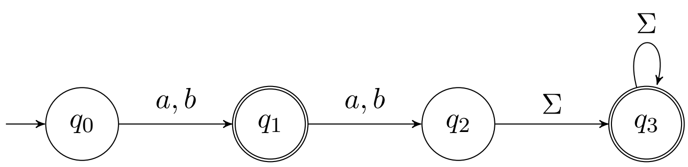
5. l'automate e reconnait le langage $(a + b) + (a + b)^2\Sigma^+$.

   Pour s'en convaincre et minimiser l'automate, on constate que l'état de droite est l'inverse d'un état puits : il est terminal et accepte tous les mots de Σ^*.

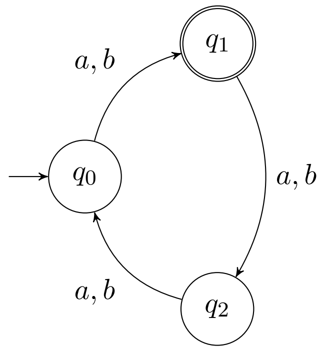
   Les deux états précédents sont non-terminaux et donc n'acceptent pas tous les mots de Σ^*. En revanche, dès qu'on lit une lettre, le mot est accepté : ces états acceptent tous les mots de Σ^+ et peuvent être fusionnés.

   Un cran plus à gauche, les deux états terminaux acceptent ε ou envoient, après lecture d'un a ou d'un b, vers les états acceptant Σ^+. Ces états acceptent tous les mots de $\varepsilon + (a + b)\Sigma^+$ et peuvent être fusionnés.

   Enfin l'état initial envoie après lecture d'un a ou d'un b vers les états acceptant $\varepsilon + (a + b)\Sigma^+$. Il accepte donc les mots de $(a + b) + (a + b)^2\Sigma^+$.

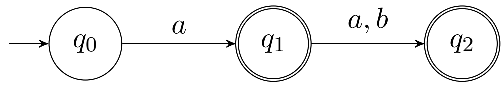


6. l'automate f est un cycle de taille 6 comme on en a construit en 1.2, et l'on peut voir que cet automate reconnait les mots de taille congrue à 1 ou 4 modulo 6. Autrement dit, il reconnait les mots de taille congrue à 1 modulo 3.

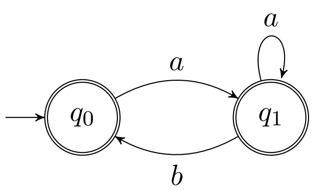


Exercice 3

1. $(a + ab)$

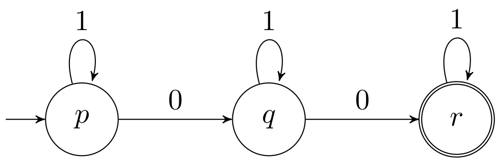


2. $(a + ab)^*$

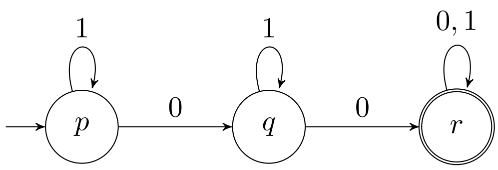


Exercice 4 Donner l'expression régulière et l'automate pour les langages ($\Sigma = \{0, 1\}$) suivants :

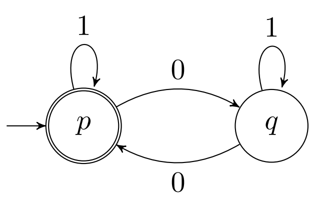
1. $\{w : w$ a exactement deux 0 $\}$: $(1^*01^*)^2$



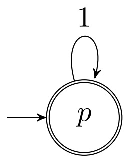
2. $\{w : w$ a au moins deux 0 $\}$: $(1^*01^*)^2\Sigma^*$ ou $\Sigma^*0\Sigma^*0\Sigma^*$



3. $\{w : w$ a un nombre pair de 0 $\}$: $((1^*01^*)^2)^*$



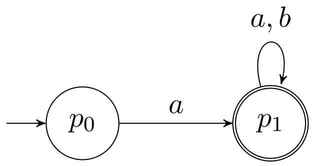
4. $\{w : w$ n'a pas de $0\}$: 1^*



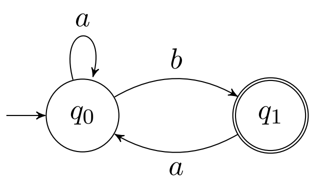
5. $\{w : w$ est un identifiant valide dans le langage de programmation `C` $\}$. Ici Σ contient toutes les lettres et symboles sur votre clavier.

Exercice 5

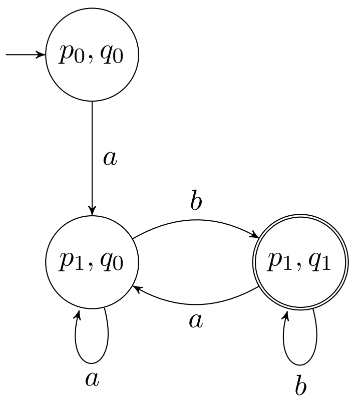
1. $e_1 = a(a+b)^*$, $e_2 = (a+b)^*b$:
   e_1 reconnaît les mots qui commencent par un a.



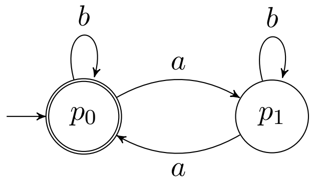
   e_2 reconnaît les mots qui terminent par un b.



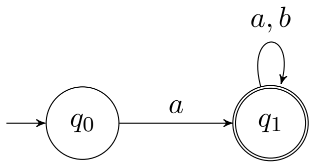
   On fait le produit de ces automates



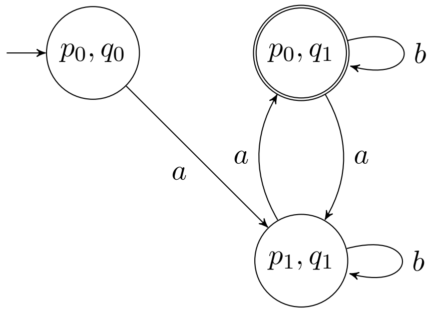
2. $e_1 = (b^*ab^*ab^*)^*$, $e_2 = a(a+b)^*$;
   e_1 reconnaît les mots qui ont un nombre pair de a.



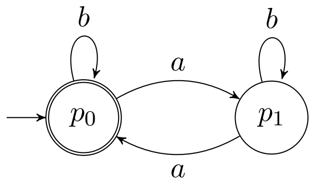
   e_2 reconnaît les mots qui commencent par un a.



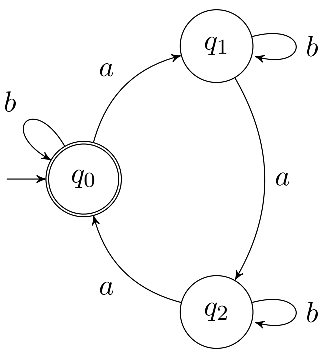
On fait le produit de ces automates



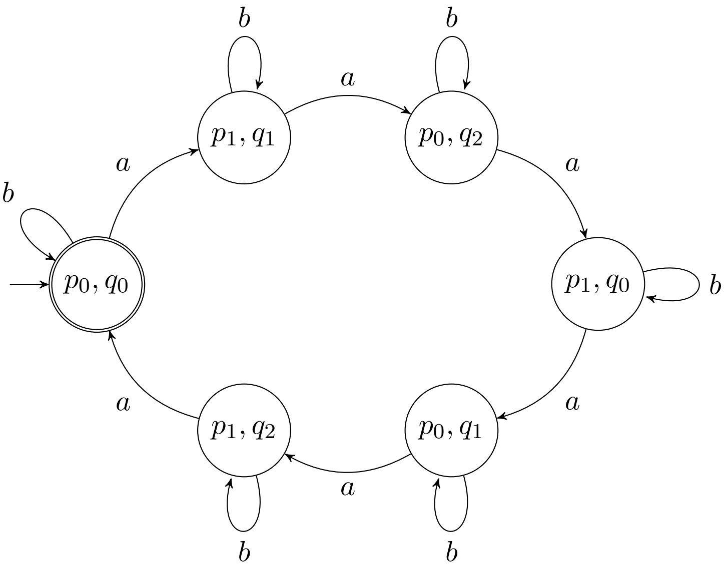
3. $e_1 = (b^*ab^*ab^*)^*$, $e_2 = (b^*ab^*ab^*ab^*)^*$;
   e_1 reconnaît les mots qui ont un nombre pair de a.



   e_2 reconnaît les mots qui ont un nombre de a multiple de 3.



   On fait le produit de ces automates


b
b
a
p_1, q_1
p_0, q_2
a
a
b
p_0, q_0
p_1, q_0
b
a
a
p_1, q_2
p_0, q_1
a
b
b