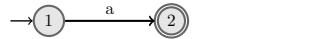


Rappel à propos des consignes et quelques conseils et remarques

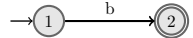
- Durée : 2 heures.
- Aucune sortie avant 30 minutes. Aucune entrée après 30 minutes.
- 1 feuille A4 R/V autorisée.
- Tout dispositif électronique est interdit (calculatrice, téléphone, tablette, montre connectée, etc.).
- Le soin de la copie sera pris en compte.
- Les exercices sont indépendants et en ordre croissant de difficulté.
- Le barème est donné à titre indicatif.
- L'examen est sur 22 points, vous devez obtenir 20 points pour obtenir la note maximale.

Solution de l'exercice 1

1. Faux. Il faudrait également que l'automate soit complet. Sinon, l'algorithme donné est incorrect.
2. Faux. Prenons Σ tel que $\{a\} \subseteq \Sigma$ et $L = \Sigma^* \cdot \{a\}$, nous avons $L \neq \Sigma^*$ et $\text{Pref}(L) = \Sigma^*$.
3. Faux. Prenons les automates suivants :



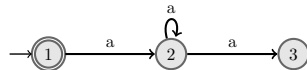
- Automate déterministe reconnaissant L_2 :



- Automate produit des deux automates précédents :



4. Vrai. (Argument simplifié). Tout autre automate déterministe de la même taille mais différent aurait soit un état marqué comme final ou non-final différemment de l'automate minimal ou alors aurait une transition en plus ou en moins entre les états (qui sont tous accessibles et co-accessibles) et ajouterait ou enlèverait un chemin vers un état accepteur.
5. Vrai. Si deux états p et q sont équivalents, alors $\forall w \in \Sigma^* : \delta^*(p, w) \in F \Leftrightarrow \delta^*(q, w) \in F$. Nous en déduisons $\forall w \in \{s\} \cdot \Sigma^* : \delta^*(p, w) \in F \Leftrightarrow \delta^*(q, w) \in F$ puis $\forall w \in \Sigma^* : \delta^*(\delta(p, s), w) \in F \Leftrightarrow \delta^*(\delta(q, s), w) \in F$. Ceci signifie que $\delta(p, s)$ et $\delta(q, s)$ sont équivalents.
6. Faux. Il faut également que ce cycle soit co-accessible. Voici deux contre-exemples :



7. Faux. Voir par exemple l'exemple utilisé dans le cours sur la détermination.
8. Vrai. L'automate minimal est unique à un renommage des états prêt.

Solution de l'exercice 2

1. Nous commençons par compléter l'automate.

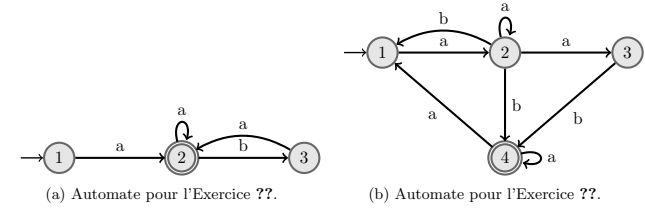
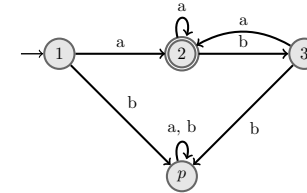
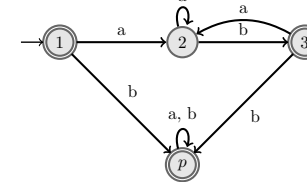


FIGURE 1 – Automates pour les exercices de complémentation et de détermination.

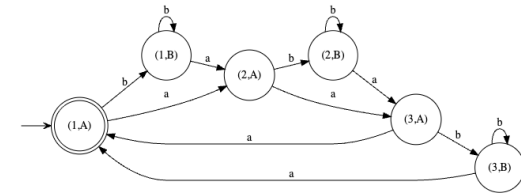


Dans cet automate complet, nous inversons états accepteurs et non-accepteurs. L'automate complémentaire est représenté ci-dessous :



Solution de l'exercice 3

1. L'automate produit est représenté ci-dessous :



2. Le premier automate reconnaît l'ensemble des mots avec un nombre d'occurrences du symbole 'a' multiple de trois. Deuxième auto, l'automate reconnaît l'ensemble des mots qui terminent par le symbole 'a'. L'automate produit reconnaissant l'intersection des langages, il reconnaît donc l'ensemble des mots avec un nombre d'occurrences du symbole 'a' qui est multiple de trois et qui terminent par le symbole 'a' s'il y a au moins une occurrence de 'a' ; c'est-à-dire que le langage contient également ϵ . Alternativement, le langage reconnu est l'ensemble des mots avec un nombre d'occurrences de 'a' multiple de trois.

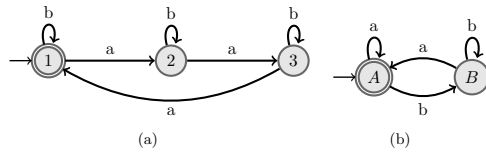


FIGURE 2 – Automates dont il faut calculer le produit.

multiple de 3 et qui ne terminent pas par 'b'.

3. Le mot *abaab* n'est pas accepté par l'automate. L'exécution de ce mot est donnée ci-dessous :

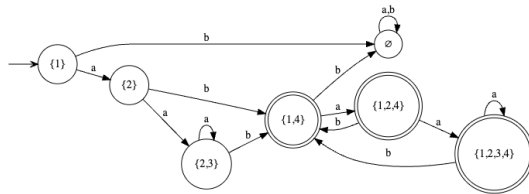
$$(1A, abaab) \cdot (2A, baab) \cdot (2B, aab) \cdot (3B, ab) \cdot (1A, b) \cdot (1B, \epsilon)$$

(Nous dénotons l'état (X, Y) par XY .)

L'état $1B$ n'étant pas accepteur, le mot n'est pas accepté.

Solution de l'exercice 4

1. Le déterminisé est représenté ci-après.



Solution de l'exercice 5

1. Un automate est sans impasse si et seulement si tous ses états accessibles sont aussi co-accessibles.
2. L'algorithme est donné ci-après.

Algorithme 1 : Vérifier l'absence d'impasse dans un automate

Entrée : Un automate $A = (Q, \Sigma, q_{\text{init}}, \delta, F)$

Sortie : Vrai si et seulement si A est sans impasse

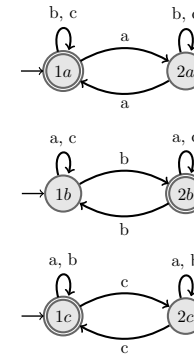
Accessibles $\leftarrow$ accessibles(A)

co_Accessibles $\leftarrow$ co_accessibles(A)

retourner Accessibles $\subseteq$ co_Accessibles

Solution de l'exercice 6

1. Un automate qui reconnaît l'ensemble des mots avec un nombre pair de a ou un nombre impair de b ou un nombre pair de c est représenté ci-après :



2. Un automate avec plusieurs états initiaux est donné par un 5-tuple $(Q, \Sigma, Q_{\text{init}}, \delta, F)$ tel que $Q_{\text{init}} \subseteq Q$ et les autres éléments sont définis comme pour les automates déterministes.
3. Une configuration de l'automate est un couple formé par un état et un mot, comme dans le cas des automates déterministes, non-déterministes et non-déterministes avec ϵ -transitions.
4. La relation de dérivation est définie comme dans le cours.

Note : Pour les deux dernières questions, on acceptera les réponses différentes mais cohérentes ; par exemple les configurations où il y a un ensemble d'états et une relation de dérivation qui calcule l'ensemble des successeurs.

Solution de l'exercice 7

1. — Exemple de langage : $\{aaa, bbb, aab\}$.
— Contre-exemple de langage : $\{a, bb, ccc\}$.
2. L'algorithme est donné ci-après.

Algorithme 2 : Vérifier si un automate déterministe est à longueur fixe

Entrée : Un automate déterministe $A = (Q, \Sigma, q_{\text{init}}, \delta, F)$

Sortie : Vrai si et seulement si A est à longueur fixe

// Construction de l'automate émondé

$E \leftarrow \text{emonde}(A)$

// Si l'automate possède un cycle accessible et co-accessible, alors le langage est infini

si $\text{detection_cycle}(E)$ **alors**

└ **retourner faux**

// Parcours en largeur dans l'automate sans cycle, sans mémorisation des états déjà visités

$A.\text{visiter} : \text{file d'états} \leftarrow \text{file vide}$

$A.\text{visiter}.\text{ajouter}(q_{\text{init}})$

// Fonction Distances associant à chaque état l'ensemble de ses distances à l'état initial

// Chaque ensemble est fini car l'automate est acyclique

$\text{Distances} : Q \rightarrow 2^{\mathbb{N}}$

$\text{Distances}(q_{\text{init}}) \leftarrow \{0\}$

tant que $A.\text{visiter} \neq \emptyset$ **faire**

└ $q \leftarrow A.\text{visiter}.\text{defiler}()$

└ **pour** $q' \in \text{Succ}(q)$ **faire**

└└ $\text{Anciennes_distances} \leftarrow \text{Distances}(q')$

└└ $\text{Distances}(q') \leftarrow \text{Distances}(q') \cup \{d + 1 \mid d \in \text{Distances}(q)\}$

└└ **si** $\text{Distances}(q') \neq \text{Anciennes_distances}$ **alors**

└└└ $A.\text{visiter}.\text{enfiler}(q')$

$\text{longueurs_acceptees} \leftarrow \bigcup_{q \in F} \text{Distances}(q)$

retourner $|\text{longueurs_acceptees}| = 1$ et $\text{longueurs_acceptees} \neq \{0\}$
