# COMPOSITION D'INFORMATIQUE n°2

Sujet unique (Durée : 3 heures)

L'utilisation de la calculatrice **n'est pas autorisée** pour cette épreuve.

⋆⋆⋆

L'épreuve est constituée d'un unique problème comportant 36 questions. Après un préliminaire, ce problème est divisé en 5 parties. Pour répondre à une question, un candidat pourra réutiliser le résultat d'une question antérieure, même s'il n'est pas parvenu à démontrer ce résultat.

Le but du problème est d'étudier les relations qui existent entre des automates qui reconnaissent un même langage grâce à la notion de morphismes d'automates.

## Préliminaires

### Concernant la programmation

Les questions de programmation doivent être traitées en langage OCaml uniquement. On autorisera toutes les fonctions des modules `Array` et `List`, ainsi que les fonctions de la bibliothèque standard (celles qui s'écrivent sans nom de module, comme `max`, `incr` ainsi que les opérateurs comme `@`). Sauf précision de l'énoncé, l'utilisation d'autres modules sera interdite.

On identifiera une même grandeur écrite dans deux polices de caractères différentes, en italique du point de vue mathématique (par exemple $n$) et en Computer Modern à chasse fixe du point de vue informatique (par exemple `n`).

### Définition mathématique d'un automate

**Définition :** Dans l'ensemble du sujet, le terme *automate* désigne un automate fini déterministe complet sur l'alphabet $\{a, b\}$, c'est-à-dire un quadruplet $\mathcal{A} = \langle Q, i, \delta, F\rangle$ où $Q$ est l'ensemble des états, $i$ l'état initial $(i \in Q)$, $\delta : Q \times \{a, b\} \to Q$ l'application de transition et $F \subseteq Q$ l'ensemble des états finals.

On note $\varepsilon$ le mot vide. Par extension de $\delta$, on appelle $\delta^*$ l'application $Q \times \{a, b\}^* \to Q$ définie pour tout état $q$ par $\delta(q, \varepsilon) = q$ et, si $\sigma$ est une lettre de $\{a, b\}$ et $w$ un mot de $\{a, b\}^*$, $\delta^*(q, \sigma w) = \delta^*(\delta(q, \sigma), w)$.

Un automate $\langle Q, i, \delta, F\rangle$ est représenté par un graphe orienté. Les sommets de ce graphe sont les éléments de $Q$. Ce graphe admet un arc $(p, q) \in Q \times Q$ étiqueté par la lettre $a$ (resp. $b$) si et seulement si $\delta(p, a) = q$ (resp. $\delta(p, b) = q$). Une flèche sans origine et pointant vers $i$ indique l'état initial. Un état final est représenté par un double cercle.

### Représentation d'automate en OCaml

**Indication OCaml :** Dans toutes les questions demandant d'implémenter une fonction en OCaml, on identifie l'ensemble des états $Q$ d'un automate $\langle Q, i, \delta, F\rangle$ avec l'ensemble des entiers compris entre 0 et $|Q| - 1$. On convient de plus que l'état initial $i$ est toujours identifié à l'entier 0. Les automates seront représentés par des objets du type `automate` suivant :

```
type automate = {
    finals : bool array;
    delta : (int * int) array;
};;
```

où, pour `aut` de type `automate` :

- `aut.finals` est un tableau de longueur $n = |Q|$ qui représente la fonction indicatrice de l'ensemble des états finals ;
- `aut.delta` est un tableau de longueur $n$ qui stocke les couples $(\delta(q,a), \delta(q,b))_{q \in Q}$.

# 1 Premiers exemples

**Question 1** Donner, sans preuve, une description courte en français du langage reconnu par l'automate $\mathcal{A}_1$ de la figure 1.

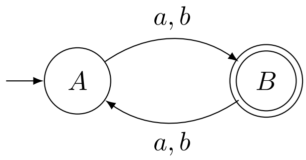


FIGURE 1 – Automate $\mathcal{A}_1$

**Question 2** Donner, sans preuve, une description courte en français du langage reconnu par l'automate $\mathcal{A}_2$ de la figure 2.

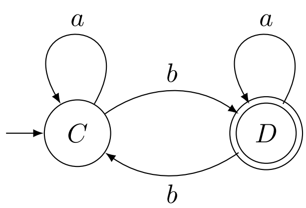


FIGURE 2 – Automate $\mathcal{A}_2$

**Question 3** Donner, sans preuve, une expression rationnelle qui dénote le langage reconnu par l'automate $\mathcal{A}_1$ de la figure 1.

**Question 4** Donner, sans preuve, une expression rationnelle qui dénote le langage reconnu par l'automate $\mathcal{A}_2$ de la figure 2.

**Question 5** Écrire en OCaml la construction d'une instance du type `automate` qui correspond à l'automate $\mathcal{A}_2$ de la figure 2.

# 2 États accessibles d'un automate

**Question 6** Écrire une fonction `numero (n: int) (lst: int list) : int array` telle que si `lst` contient des entiers compris entre 0 et $n-1$, alors `numero n lst` renvoie un tableau `t` de taille $n$ tel que pour $i \in [\![0, n-1]\!]$, `t.(i)` vaut $-1$ si $i$ est absent de `lst`, et `t.(i)` est égal à l'indice d'une des occurrences de $i$ dans `lst` sinon. Par exemple, `numero 5 [3; 2; 0; 3]` peut renvoyer `[|2; -1; 1; 0; -1|]`.

**Définition :** Soit $\mathcal{A} = \langle Q_\mathcal{A}, i_\mathcal{A}, \delta_\mathcal{A}, F_\mathcal{A} \rangle$ un automate et $Q'$ l'ensemble des états accessibles de $\mathcal{A}$. On appelle *partie accessible* de l'automate $\mathcal{A}$ le nouvel automate $\mathcal{A}' = \langle Q', i_\mathcal{A}, \delta', F_\mathcal{A} \cap Q' \rangle$ où $\delta'$ est la restriction de $\delta_a$ au domaine $Q' \times \{a, b\}$.

On dit qu'un automate est *accessible* lorsque tous ses états sont accessibles.

**Question 7** Écrire une fonction `etats_accessibles (aut: automate) : int list` qui renvoie la liste des états accessibles de l'automate donné en argument et que l'on obtient par un parcours de graphe en profondeur depuis l'état initial. La liste renvoyée doit suivre l'ordre dans lequel les états sont rencontrés pour la première fois et ne doit pas contenir de doublons.

Donner la complexité de la fonction écrite.

**Question 8** Écrire une fonction `partie_accessible (aut: automate) : automate` qui construit la partie accessible de l'automate donné en argument.

# 3 Morphisme d'automates

**Définition :** Soient deux automates $\mathcal{A} = \langle Q_\mathcal{A}, i_\mathcal{A}, \delta_\mathcal{A}, F_\mathcal{A} \rangle$ et $\mathcal{B} = \langle Q_\mathcal{B}, i_\mathcal{B}, \delta_\mathcal{B}, F_\mathcal{B} \rangle$. Une application $\varphi : Q_\mathcal{A} \to Q_\mathcal{B}$ est appelée *morphisme d'automates* de l'automate $\mathcal{A}$ vers l'automate $\mathcal{B}$, et est notée $\varphi : \mathcal{A} \to \mathcal{B}$, si elle satisfait les conditions suivantes :

$$\varphi \text{ est surjective,} \tag{1}$$

$$\varphi(i_\mathcal{A}) = i_\mathcal{B} \tag{2}$$

$$\forall q \in Q_\mathcal{A}, \quad \forall \sigma \in \{a, b\}, \quad \varphi(\delta_\mathcal{A}(q, \sigma)) = \delta_\mathcal{B}(\varphi(q), \sigma) \tag{3}$$

$$\forall q \in Q_\mathcal{A}, \quad q \in F_\mathcal{A} \iff \varphi(q) \in F_\mathcal{B} \tag{4}$$

**Indication OCaml :** En OCaml, on représente un morphisme $\varphi : \mathcal{A} \to \mathcal{B}$ par un tableau de type `int array`, de longueur $|Q_\mathcal{A}|$, contenant les valeurs $(\varphi(q))_{q \in Q_\mathcal{A}}$, comprises entre 0 et $|Q_B| - 1$. On utilisera le type défini par l'alias :

```
type morphisme = int array
```

## 3.1 Exemple de morphismes d'automates

**Question 9** À partir des figures 2 et 3 représentant les automates $\mathcal{A}_1$ et $\mathcal{A}_3$, recopier le tableau suivant et le compléter, sans justification, par des états de $\mathcal{A}_2$ de sorte que ce tableau représente un morphisme d'automate $\varphi$ de l'automate $\mathcal{A}_3$ vers l'automate $A_2$.

| $q$ | $\varphi(q)$ |
|---|---|
| $E$ | |
| $F$ | |
| $G$ | |

**Question 10** À partir des figures 2 et 4 représentant les automates $\mathcal{A}_2$ et $\mathcal{A}_4$, donner, sans justification, un morphisme d'automates de l'automate $\mathcal{A}_4$ vers l'automate $\mathcal{A}_2$.

**Question 11** À partir des figures 1 et 2, montrer qu'il n'existe pas de morphisme d'automate de l'automate $\mathcal{A}_1$ vers l'automate $\mathcal{A}_2$.

**Question 12** À partir des figures 2 et 5, montrer qu'il n'existe pas de morphisme d'automate de l'automate $\mathcal{A}_5$ vers l'automate $\mathcal{A}_2$.

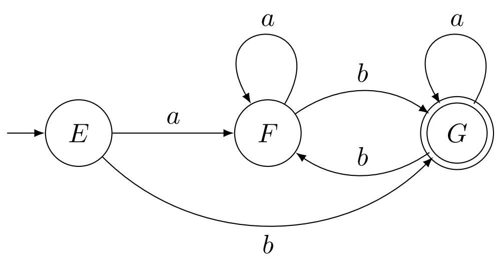


Figure 3 – Automate $\mathcal{A}_3$

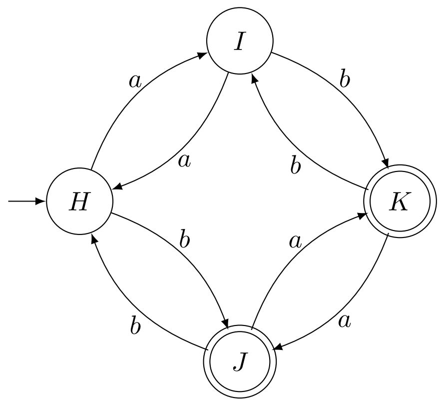


Figure 4 – Automate $\mathcal{A}_4$

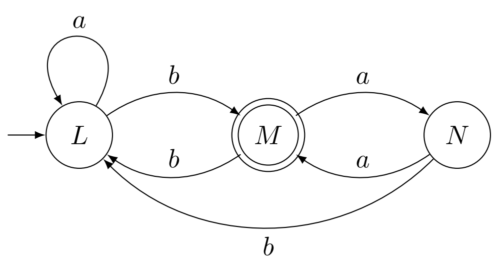


Figure 5 – Automate $\mathcal{A}_5$

## 3.2 Propriétés des morphismes d'automates

**Question 13** Montrer que s'il existe un morphisme d'automates entre deux automates, alors les deux automates acceptent le même langage.

**Question 14** Montrer qu'un morphisme $\varphi$ entre deux automates ayant le même nombre d'états est nécessairement une application bijective, et que l'application $\varphi^{-1}$ est encore un morphisme d'automates.

On dit dans ce cas que $\varphi$ est un *isomorphisme d'automates.*

**Question 15** Montrer que la composition de deux morphismes d'automates est encore un morphisme d'automates.

## 3.3 Existence de morphismes d'automates entre automates accessibles

**Question 16** Montrer que le point (1) de la définition des morphismes d'automates découle des points (2), (3) et (4) quand les deux automates considérés sont accessibles.

**Question 17** Écrire une fonction de signature `trouver_morphisme (aut1: automate) (aut2: automate) : morphisme option` qui renvoie une option de morphisme qui vaut `None` s'il n'existe pas de morphisme du premier automate vers le deuxième, et vaut `Some phi` avec `phi` un morphisme du premier automate vers le deuxième sinon.

# 4 Constructions de morphismes d'automates

## 4.1 Automate produit

**Définition :** Soient deux automates $\mathcal{A} = \langle Q_\mathcal{A}, i_\mathcal{A}, \delta_\mathcal{A}, F_\mathcal{A} \rangle$ et $\mathcal{B} = \langle Q_\mathcal{B}, i_\mathcal{B}, \delta_\mathcal{B}, F_\mathcal{B} \rangle$. On appelle *automate produit*, et on note $\mathcal{A} \times \mathcal{B}$, l'automate :

$$\mathcal{A} \times \mathcal{B} = \langle Q_\mathcal{A} \times Q_\mathcal{B}, (i_\mathcal{A}, i_\mathcal{B}), \delta_{\mathcal{A}\times\mathcal{B}}, F_\mathcal{A} \times F_\mathcal{B} \rangle$$

où l'application $\delta_{\mathcal{A}\times\mathcal{B}}$ est définie, pour tout couple d'états $(p,q) \in Q_\mathcal{A} \times Q_\mathcal{B}$ et pour toute lettre $\sigma \in \{a,b\}$, par $\delta_{\mathcal{A}\times\mathcal{B}}((p,q),\sigma) = (\delta_\mathcal{A}(p,\sigma), \delta_\mathcal{B}(q,\sigma))$.

**Question 18** On considère les automates $\mathcal{A}_3$ et $\mathcal{A}_4$ qui sont représentés par les figures 3 et 4. Dessiner, sans justification, la partie accessible du produit d'automates $\mathcal{A}_3 \times \mathcal{A}_4$.

**Question 19** Écrire une fonction `produit (aut1: automate) (aut2: automate) : automate` qui renvoie le produit des deux automates donnés en argument.

**Question 20** Soient $(p,q)$ un état accessible du produit de deux automates qui acceptent le même langage. Montrer que $p$ est un état final du premier automate si et seulement si $q$ est un état final du second automate.

**Question 21** Montrer qu'il existe toujours un morphisme d'automates de la partie accessible du produit de deux automates accessibles qui acceptent le même langage vers chacun de ces deux automates.

## 4.2 Diagramme d'automates

Dans toute cette sous-section, on considère qu'il existe trois automates accessibles $\mathcal{A}$, $\mathcal{B}$ et $\mathcal{B}'$ et deux morphismes d'automates $\varphi : \mathcal{A} \to \mathcal{B}$ et $\varphi' : \mathcal{A} \to \mathcal{B}'$. Le but de cette sous-section est de construire un nouvel automate accessible $\mathcal{C}$ et trois morphismes $\psi$, $\psi'$ et $\eta$ dont la situation est résumée dans le diagramme suivant.

$$\begin{tikzcd}
& \mathcal{A} \arrow[dl, "\varphi"'] \arrow[dr, "\varphi'"] \arrow[dd, "\eta"'] & \\
\mathcal{B} \arrow[dr, "\psi"'] & & \mathcal{B}' \arrow[dl, "\psi'"] \\
& \mathcal{C} &
\end{tikzcd}$$

**Définition :** On définit une relation sur $Q_\mathcal{A}$, notée $\equiv$. Pour tout couple d'états $(p,q) \in Q_\mathcal{A}^2$, $p \equiv q$ s'il existe une suite finie de longueur $k+1$, avec $k \in \mathbb{N}$, de termes $p = q_0, q_1, q_2, \ldots, q_k = q$ d'états de $Q_\mathcal{A}$ telle que :

$$\forall 0 \leqslant j < k, \quad \varphi(q_j) = \varphi(q_{j+1})) \text{ ou } \varphi'(q_j) = \varphi'(q_{j+1})$$

**Question 22** Montrer que la relation $\equiv$ est une relation d'équivalence.

**Question 23** Montrer que pour $(p,q) \in Q_{\mathcal{A}}^2$, si $p \equiv q$, alors pour tout lettre $\sigma \in \{a,b\}$, on a $\delta_{\mathcal{A}}(p,\sigma) \equiv \delta_{\mathcal{A}}(q,\sigma)$.

**Question 24** Montrer que pour $(p,q) \in Q_{\mathcal{A}}^2$, si $p \equiv q$, alors $p$ est un état final de $\mathcal{A}$ si et seulement si $q$ est un état final de $\mathcal{A}$.

**Définition :** La classe d'équivalence d'un état $q \in Q_{\mathcal{A}}$, notée $[q]$, est l'ensemble :

$$[q] = \{p \in Q_{\mathcal{A}}, p \equiv q\}$$

Dans ce qui suit, on appelle $\ell$ le nombre de classes d'équivalence de la relation $\equiv$ et on note $S_0$, $S_1$, ..., $S_{\ell-1}$ ces classes. On choisira $S_0$ de sorte que $i_{\mathcal{A}} \in S_0$. On note $\eta$ l'application $Q_{\mathcal{A}} \to \{S_0, S_1, \ldots, S_{\ell-1}\}$ qui, à chaque état $q \in Q_{\mathcal{A}}$, associe la classe d'équivalence $[q]$.

**Question 25** Construire un automate accessible $\mathcal{C}$ dont l'ensemble d'états est $\{S_0, \ldots, S_{\ell-1}\}$ et tel que $\eta$ est un morphisme de l'automate $\mathcal{A}$ vers l'automate $\mathcal{C}$. Justifier.

**Question 26** Construire deux morphismes d'automates $\psi : \mathcal{B} \to \mathcal{C}$ et $\psi' : \mathcal{B}' \to \mathcal{C}$, où $\mathcal{C}$ est l'automate construit à la question précédente.

**Indication OCaml :** En OCaml, on représente la classe d'équivalence $S_j$ par l'indice $j$.

**Question 27** Écrire une fonction `renomme (t: int array) : int array` qui renomme le contenu d'un tableau contenant des entiers positifs prenant $\ell$ valeurs distinctes en utilisant les entiers entre 0 et $\ell - 1$. Le premier élément du résultat devra être égal à 0.

Par exemple, `renomme [|4; 4; 5; 0; 4; 5|]` peut renvoyer `[|0; 0; 1; 2; 0; 1|]`.

Déterminer la complexité de la fonction `renomme`.

**Question 28** Écrire une fonction `relation (phi1: morphisme) (phi2: morphisme) : morphisme` qui, à partir des tableaux $[\varphi(q)]_{q \in Q_{\mathcal{A}}}$ et $[\varphi'(q)]_{q \in Q_{\mathcal{A}}}$, renvoie le tableau $[\eta(q)]_{q \in Q_{\mathcal{A}}}$.

# 5 Réduction d'automates

## 5.1 Existence et unicité

**Question 29** Montrer que si deux automates accessibles $\mathcal{A}$ et $\mathcal{B}$ acceptent le même langage, alors on peut construire un automate $\mathcal{C}$ et deux morphismes $\varphi : \mathcal{A} \to \mathcal{C}$ et $\psi : \mathcal{B} \to \mathcal{C}$.

**Question 30** Déterminer l'automate $\mathcal{C}$ définir à la question précédente pour les automates $\mathcal{A}_3$ et $\mathcal{A}_4$ (figures 3 et 4) et préciser les applications $\varphi$ et $\psi$.

Soit $L$ un langage rationnel et $\mathfrak{A}_L$ l'ensemble des automates (complets déterministes) accessibles qui acceptent le langage $L$. On note $m_L$ le plus petit nombre d'états d'un automate de $\mathfrak{A}_L$.

**Question 31** Montrer que deux automates de $\mathfrak{A}_L$ ayant $m_L$ états sont nécessairement isomorphes.

**Question 32** Montrer que pour tout automate $\mathcal{A} \in \mathfrak{A}_L$, il existe un morphisme $\varphi : \mathcal{A} \to \mathcal{M}_L$, où $\mathcal{M}_L$ est un automate de $\mathfrak{A}_L$ à $m_L$ états.

## 5.2 Construction d'un automate réduit par fusion d'états

**Définition :** Soient deux automates $\mathcal{A} = \langle Q_{\mathcal{A}}, i_{\mathcal{A}}, \delta_{\mathcal{A}}, F_{\mathcal{A}} \rangle$ et $\mathcal{B} = \langle Q_{\mathcal{B}}, i_{\mathcal{B}}, \delta_{\mathcal{B}}, F_{\mathcal{B}} \rangle$. On dit que deux états $p$ et $q$ de l'automate $\mathcal{A}$ ont été fusionnés dans l'automate $\mathcal{B}$ s'il existe un morphisme d'automates $\varphi$ de $\mathcal{A}$ vers $\mathcal{B}$ tel que $\varphi(p) = \varphi(q)$ et si le nombre d'états satisfait $|Q_{\mathcal{B}}| < |Q_{\mathcal{A}}|$.

**Question 33** On considère l'automate $\mathcal{A}_6$ de la figure 6. Dessiner un automate $\mathcal{A}_6^{O,P}$ dans lequel les états $O$ et $P$ ont été fusionnés. On donnera un morphisme d'automates $\mathcal{A}_6 \to \mathcal{A}_6^{O,P}$.

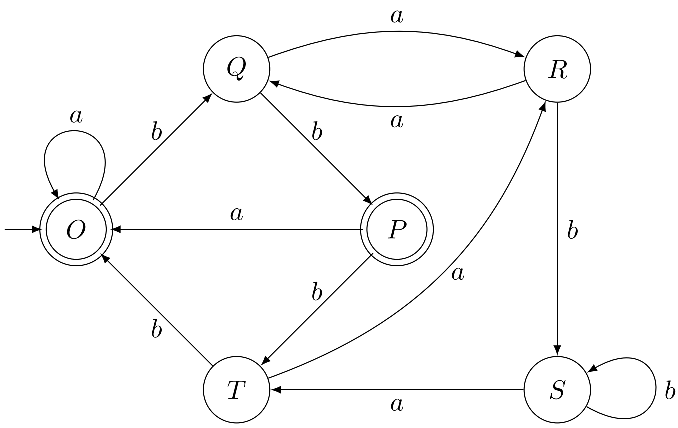


Figure 6 – Automate $\mathcal{A}_6$

**Question 34** Expliquer brièvement pourquoi il n'est pas possible de construire un automate $\mathcal{A}_6^{Q,R}$ muni d'un morphisme d'automates $\psi : \mathcal{A}_6 \to \mathcal{A}_6^{Q,R}$ tel que $\psi(Q) = \psi(R)$.

**Question 35** Quels états faut-il encore fusionner dans $\mathcal{A}_6^{O,P}$ pour obtenir un automate à trois états $\mathcal{M}_{L_6}$ qui reconnait le même langage que $\mathcal{A}_6$ ?

Soit $\mathcal{A} = \langle Q_{\mathcal{A}}, i_{\mathcal{A}}, \delta_{\mathcal{A}}, F_{\mathcal{A}} \rangle$ un automate accessible. On note $G_{\mathcal{A}} = (S, A)$ le graphe orienté tel que :

- $S = Q \times Q$ ;
- pour $\sigma \in \{a, b\}$ et $(p, q) \in S$, $((p, q), (\delta(p, \sigma), \delta(q, \sigma))) \in A$.

**Question 36** En considérant des chemins depuis $F \times \overline{F} \cup \overline{F} \times F$ dans le graphe transposé de $G_{\mathcal{A}}$, décrire en français un algorithme permettant de calculer l'automate $\mathcal{M}_L$ associé au langage $L$ reconnu par $\mathcal{A}$.

Déterminer la complexité temporelle de cet algorithme.

⋆⋆⋆