C39211

Ecole Normale Supérieure de Cachan
61 avenue du président Wilson
94230 CACHAN

_____

Concours d'admission en **3ème année**
Informatique

Session 2009

_____

INFORMATIQUE 1

_____

Durée : 5 **heures**

_____

« Aucun document n'est autorisé »

« Aucun dictionnaire n'est autorisé »

« L'usage de toute calculatrice est interdit »

Informatique I

Ce sujet se propose d'étudier le problème suivant : étant donné un automate déterministe complet, trouver un algorithme qui donne, pour tout mot d'entrée, le calcul qui lui est associé dans l'automate. Afin de résoudre ce problème on représente des mots par des ensembles finis de mots binaires et l'on s'autorise un jeu d'opérations restreint pour travailler sur les mots binaires. On cherche alors à construire un algorithme qui utilise un nombre constant de telles opérations pour résoudre le problème : on parle alors d'algorithme binaire.

La première partie donne les définitions de bases ainsi que quelques exemples. La deuxième partie étudie une sous-classe des automates déterministes complets tandis que la troisième partie montre que les automates de cette classe admettent des algorithmes binaires. Enfin, la dernière partie caractérise précisément les automates qui admettent des algorithmes binaires.

La notation tiendra compte de la rigueur des raisonnements et de la clarté des explications. Chaque question pourra être traitée en admettant les résultats des questions précédentes.

Partie 1. Préliminaires

On commence par quelques définitions et notations qui seront valables pour l'intégralité de ce sujet. On donne ensuite quelques exemples et propriétés de base utiles dans la suite.

Un **alphabet** est un ensemble fini dont les éléments sont appelés **lettres**. Un **mot** de longueur n sur un alphabet A est une concaténation finie $u = a_1 a_2 \cdots a_n$ de n lettres de l'alphabet (c.-à-d. $a_i \in A$ pour tout $i = 1, \ldots, n$). On notera A^* l'ensemble des mots sur l'alphabet A, et le mot vide sera noté ε. Un mot sur l'alphabet $\{0, 1\}$ sera qualifié de **mot binaire**.

Définition. *Un* ***automate déterministe complet*** *est un quintuplet $\mathcal{A} = (Q, A, q_i, F, \delta)$ où Q est un ensemble fini d'états de contrôle, A est un alphabet fini d'entrée, $q_i \in Q$ est l'état initial, $F \subseteq Q$ est l'ensemble des états finaux et $\delta : Q \times A \to Q$ est une fonction (totale) de transition. Dans ce problème, tous les automates considérés seront — sauf mention explicite — déterministes complets, et on se permettra parfois d'omettre ces deux qualificatifs.*

Étant donnés un automate déterministe complet $\mathcal{A} = (Q, A, q_i, F, \delta)$ et un mot $u = a_1 a_2 \cdots a_n$, on définit le ***calcul*** *(unique) de $\mathcal{A}$ sur u comme le mot de longueur $n + 1$, sur l'alphabet Q, $r = q_0 q_1 q_2 \cdots q_n$ tel que $q_0 = q_i$ et $q_j = \delta(q_{j-1}, a_j)$ pour tout $1 \leq j \leq n$.*

Un mot $u \in A^*$ est ***accepté*** *par l'automate $\mathcal{A}$ si et seulement si le calcul de $\mathcal{A}$ sur u se termine par un état final. L'ensemble $L(\mathcal{A})$ des mots acceptés par $\mathcal{A}$ est appelé* ***langage accepté*** *par $\mathcal{A}$. Un* ***langage régulier*** *sur un alphabet A est un ensemble de mots $K \subseteq A^*$ tel qu'il existe un automate déterministe complet $\mathcal{A}$ avec $K = L(\mathcal{A})$.*

On souhaite résoudre le problème suivant. Étant donné un automate déterministe complet $\mathcal{A}$, trouver un algorithme qui, pour tout mot sur l'alphabet d'entrée de l'automate, produit le calcul associé dans l'automate. Notez qu'un tel algorithme ne doit dépendre que de l'automate $\mathcal{A}$.

Comme les automates considérés dans ce problème possèdent un unique état initial, on souhaitera produire le calcul duquel on **tronquera le premier état** (qui est toujours l'état initial). Ainsi, pour un mot sur l'alphabet d'entrée de l'automate, un tel algorithme produira un mot d'états de même longueur que le mot d'entrée.

Considérez par exemple l'automate $\mathcal{A}$ donné dans la Figure 1 (l'état initial est représenté par une flèche entrante, les états finaux par des flèches sortantes, convention qui vaudra pour toutes les figures dans ce texte). Pour un tel automate, on souhaitera donc avoir un algorithme qui sur le mot d'entrée *abbababbabba* produit le mot 123121231231.

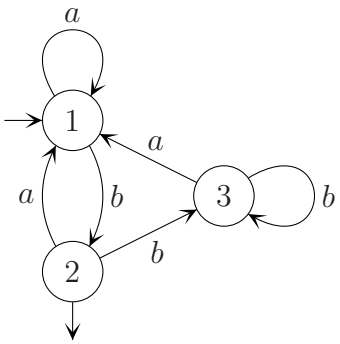

FIGURE 1 – Un exemple d'automate

Question 1. Expliquez brièvement pourquoi il existe, pour tout automate déterministe complet, un algorithme qui produit, pour tout mot en entrée, le calcul associé dans l'automate, et ce en temps linéaire en la taille du mot en entrée.

Considérons un mot $u = a_1a_2 \cdots a_n$ sur un alphabet A. On lui associe $|A|$ mots binaires définis comme suit : pour toute lettre $a \in A$, le mot $p_a(u)$ est défini comme la suite $p_a(u) = b_1b_2 \cdots b_n$ où b_i vaut 1 si a_i vaut a et vaut 0 sinon. Ainsi, si $u = abbababbabba$, on aura $p_a(u) = 100101001001$ et $p_b(u) = 011010110110$. De même, un calcul sera représenté par des mots binaires. Ainsi le calcul r associé au mot u précédent dans l'automate $\mathcal{A}$ de la Figure 1 sera représenté par les mots $p_1(r) = 100101001001$, $p_2(r) = 010010100100$ et $p_3(r) = 001000010010$.

Cette nouvelle représentation, par des mots binaires, des mots d'entrée et des calculs associés permet de travailler sur l'entrée, c'est-à-dire les $(p_a(u))_{a\in A}$, en utilisant des opérations *ad hoc* pour les mots binaires. Dans tout le problème les opérations autorisées sur les mots binaires seront les suivantes :

- Les **opérations logiques lettre à lettre** : $\vee$ (ou logique), $\wedge$ (et logique) et $\neg$ (négation logique). Pour deux mots binaires de même longueur $u = a_1a_2 \cdots a_n$ et $v = b_1b_2 \cdots b_n$, on définit $u \vee v = c_1c_2 \cdots c_n$ où, pour tout $1 \leq i \leq n$, $c_i = 1$ si et seulement si $a_i = 1$ ou $b_i = 1$. De même on définit $u \wedge v = d_1d_2 \cdots d_n$ où, pour tout $1 \leq i \leq n$, $d_i = 1$ si et

seulement si $a_i = 1$ et $b_i = 1$. Enfin, on définit $\neg u = e_1 e_2 \cdots e_n$ où, pour tout $1 \leq i \leq n$, $e_i = 1$ si et seulement si $a_i = 0$.
Ainsi, on aura par exemple $011010 \vee 101011 = 111011$, $011010 \wedge 101011 = 001010$ et $\neg 011010 = 100101$.

- Le **décalage vers la droite** $\uparrow_b u$ avec $b = 0$ ou 1 prend un mot binaire u supprime sa dernière lettre et ajoute la lettre b en tête du mot. Formellement, pour $b = 0$ ou 1, $\uparrow_b e_1 e_2 \cdots e_n = b e_1 e_2 \cdots e_{n-1}$. Notez que les mots u et $\uparrow_b u$ ont même longueur.
  On aura par exemple $\uparrow_0 011010 = 001101$ et $\uparrow_1 011010 = 101101$.
- L'**addition binaire** de mots binaires de même longueur : elle consiste en une addition classique de la gauche vers la droite, sauf que l'on ne garde pas le bit de poids maximal si la longueur du résultat excède celle des mots de départs (ainsi, le résultat n'est pas forcément exact ici). Par exemple $011010 + 101011 = 110100$.

Ainsi, toutes les opérations précédentes prennent en entrée un ou deux mots binaires (de même longueur) et retourne un mot binaire de même longueur que l'entrée.

Définition. *Étant donné un automate déterministe complet (d'alphabet d'entrée A, d'états Q), un* ***algorithme binaire*** *est un algorithme qui, pour tout mot $u \in A^*$, calcule les mots binaires $(p_q(r))_{q \in Q}$ à partir des mots binaires $(p_a(u))_{a \in A}$ en appliquant un nombre fini d'opérations* ***indépendant*** *du mot u, où $r \in Q^*$ est le calcul associé à u dans l'automate, calcul duquel on a tronqué le premier état. Lorsqu'un tel algorithme existe on dira que l'automate* ***admet un algorithme binaire*** *et on parlera de* ***l'algorithme binaire associé à l'automate****.*

Supposons que l'on possède un modèle de calcul permettant de réaliser les opérations sur les mots binaires en temps constant : un algorithme binaire peut alors être vu comme un algorithme fonctionnant en temps constant.

Question 2. Expliquez pourquoi l'algorithme binaire suivant est associé à l'automate de la Figure 1.

$$\begin{cases} p_1(r) = p_a(u) \\ p_2(r) = p_b(u) \wedge \uparrow_1 p_a(u) \\ p_3(r) = p_b(u) \wedge \uparrow_0 p_b(u) \end{cases}$$

Question 3. Soit un alphabet A et soit un mot $u \in A^*$ de longueur n. Expliquez comment définir les constantes (de longueur n) $\mathbb{0} = \underbrace{00\cdots0}_{n}$ et $\mathbb{1} = \underbrace{11\cdots1}_{n}$ à l'aide des opérations précédentes appliquées aux mots binaires $\{p_a(u) \mid a \in A\}$. En déduire que tout automate déterministe complet à un seul état admet un algorithme binaire.

Considérez l'automate $\mathcal{B}$ donné dans la la Figure 2. Le calcul associé à un mot u est uniquement déterminé par la première lettre de u. Il est clair que l'on ne va pas pouvoir associer à un tel automate un algorithme binaire qui n'utilise que les opérations logiques lettre à lettre et le décalage vers la droite. C'est pour de tels automates que l'addition va se révéler précieuse.

Question 4. Expliquez pourquoi l'algorithme binaire suivant est associé à l'automate $\mathcal{B}$ de la Figure 2 (les mots $\mathbb{0}$ et $\mathbb{1}$ sont ceux introduits dans la Question 3).

$$\begin{cases} p_0(r) = \mathbb{0} \\ p_1(r) = \neg[\mathbb{1} + (\uparrow_1 \mathbb{0} \wedge p_a(u))] \\ p_2(r) = [\mathbb{1} + (\uparrow_1 \mathbb{0} \wedge p_a(u))] \end{cases}$$

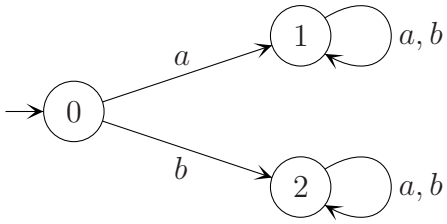

FIGURE 2 – L'automate $\mathcal{B}$

PARTIE 2. AUTOMATES RESET-DÉCOMPOSABLES

On définit une sous-famille d'automates déterministes complets — les automates reset-décomposables — et on prouve plusieurs propriétés de ces automates.

Définition. *Soit un automate déterministe complet $\mathcal{A} = (Q, A, q_i, F, \delta)$. Un état $s \in Q$ est* ***reset-décomposable*** *si et seulement si :*

$$\forall a \in A,\ [\exists p \neq s \text{ tel que } \delta(p, a) = s] \Rightarrow [\forall q \in Q,\ \delta(q, a) = s]$$

Dans l'automate de la figure 3, l'état 1 est reset-décomposable.

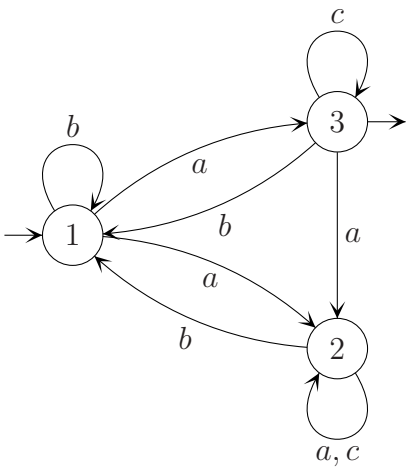

FIGURE 3 – Un exemple d'automate reset-décomposable : $\mathcal{A}_r$.

Question 5. Tout automate déterministe complet à au moins deux états possède-t-il un état reset-décomposable ?

Soit un automate $\mathcal{A} = (Q, A, q_i, F, \delta)$ et soit un état $s \in Q$ reset-decomposable. On définit, informellement, l'automate $\mathcal{A} \setminus \{s\}$ à partir de $\mathcal{A}$ comme suit :

- on supprime l'état s ;
- on supprime les transitions qui partent de s et celles qui y arrivent ;
- on supprime de l'alphabet d'entrée les lettres qui ne sont plus utilisées.

Notez que l'automate $\mathcal{A} \setminus \{s\}$ peut ne pas avoir d'état initial. Par exemple si l'on appelle $\mathcal{A}_r$ l'automate de la figure 3, l'automate $\mathcal{A}_r \setminus \{1\}$ est celui donné dans la figure 4 : outre l'absence de l'état 1 vous remarquerez que la lettre b ne fait pas partie de l'alphabet d'entrée de l'automate $\mathcal{A}_r \setminus \{1\}$.

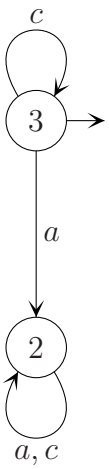

FIGURE 4 – L'automate $\mathcal{A}_r \setminus \{1\}$

Question 6. Montrez que si $\mathcal{A}$ est un automate déterministe complet dans lequel s est un état reset-décomposable, alors $\mathcal{A} \setminus \{s\}$ est également déterministe et complet (on ne demande par contre pas ici que $\mathcal{A} \setminus \{s\}$ possède un état initial).

La définition qui suit ne requiert pas que les automates possèdent un état initial : c'est plus une propriété du graphe étiqueté sous-jacent à l'automate qu'une propriété de l'automate en tant que machine lisant des mots depuis un état initial.

Définition. *Un automate $\mathcal{A} = (Q, A, q_i, F, \delta)$ déterministe complet est* ***reset-décomposable*** *s'il vérifie l'une des deux conditions suivantes :*

- *Q est un singleton.*
- *$\mathcal{A}$ possède un état reset-décomposable $s \in Q$ et l'automate $\mathcal{A}\setminus\{s\}$ est reset-décomposable.*

Question 7. L'automate $\mathcal{A}_r$ de la figure 3 est-il reset-décomposable ?

Question 8. Proposez un algorithme qui prend en entrée un automate déterministe complet et décide, en **temps polynomial** en la taille de l'automate, si celui-ci est ou non reset-décomposable. Expliquez comment vous représentez votre automate et donnez précisément la complexité de votre algorithme.

On rappelle que l'on peut associer à tout automate déterministe complet $\mathcal{A}$ un automate déterministe complet (unique à un renommage près des états) $\mathcal{A}_{min}$ qui reconnaît le même langage que $\mathcal{A}$ — c'est-à-dire que $L(\mathcal{A}) = L(\mathcal{A}_{min})$ — et qui possède un nombre minimal d'états — c'est-à-dire qu'il n'existe pas d'automate possèdant strictement moins d'états que $\mathcal{A}_{min}$ et reconnaissant $L(\mathcal{A})$.

Rappelons également comment construire l'automate $\mathcal{A}_{min}$ associé à un automate déterministe complet $\mathcal{A} = (Q, A, q_i, F, \delta)$ (il s'agit d'un résultat classique que vous pouvez admettre en tant que tel si vous ne le connaissez pas). Pour cela, on définit dans un premier temps, pour tout état $q \in Q$ et tout mot $u \in A^*$, $\delta^*(q, u) = q$ si u est le mot vide et $\delta^*(q, u) = \delta^*(\delta(q, a), v)$ si $u = a \cdot v$ où $a \in A$ est une lettre et $v \in A^*$. On définit ensuite une relation d'équivalence $\sim$ sur les états de Q en posant : $q \sim q'$ si et seulement si l'on a $\delta^*(q, u) \in F \Leftrightarrow \delta^*(q', u) \in F$ pour tout $u \in A^*$. Pour tout $q \in Q$, on note $[q]_\sim = \{q' \mid q \sim q'\}$. On prouve facilement que $q \sim q'$ si et seulement si l'on a $\delta^*(q, u) \in F \Leftrightarrow \delta^*(q', u) \in F$ pour tout mot $u \in A^*$ de longueur inférieure ou égale à $|Q|$.

On a alors $\mathcal{A}_{min} = (Q_{/\sim}, A, [q_i]_\sim, F_{/\sim}, \delta_\sim)$ où :

- Les états $Q_{/\sim} = \{[q]_\sim \mid q \in Q\}$ de $\mathcal{A}_{min}$ sont les classes d'équivalence de Q pour la relation $\sim$.
- L'état initial $[q_i]_\sim$ de $\mathcal{A}_{min}$ est la classe d'équivalence de q_i pour la relation $\sim$.
- Les états finaux $F_{/\sim} = \{[q_f]_\sim \mid q_f \in F\}$ de $\mathcal{A}_{min}$ sont les classes d'équivalence des états finaux F pour la relation $\sim$.
- La relation de transition $\delta_\sim$ de $\mathcal{A}_{min}$ est donnée par $\delta_\sim([q]_\sim, a) = [\delta(q, a)]_\sim$ pour tout $q \in Q$ et tout $a \in A$. Notez que $\delta_\sim$ est bien définie.

Question 9. Donnez l'automate minimal associé à l'automate déterministe complet de la Figure 5.

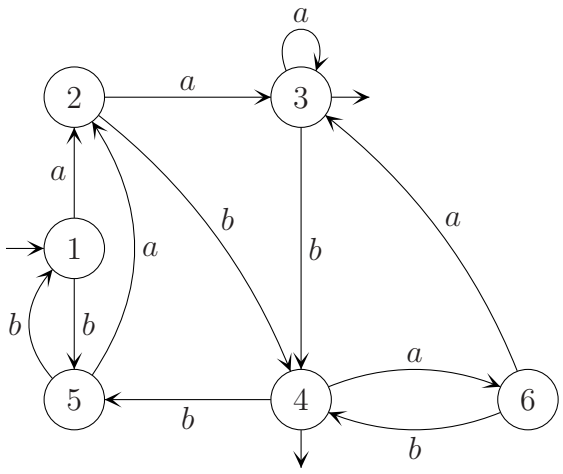

FIGURE 5 – Automate à minimiser

Question 10. Montrez que si un automate est reset-décomposable il en est de même pour son automate minimal. On pourra considérer la définition précédente de l'automate minimal.

Un langage régulier est **reset-decomposable** si et seulement s'il existe un automate qui le reconnaît et qui est reset-décomposable.

Question 11. Peut-on décider si un langage est reset-décomposable ?

Question 12. La classe des langages reset-décomposables est-elle fermée par complémentaire ?

Question 13. En considérant les langages $L_1 = A^*aa$ et $L_2 = A^*ab$ sur l'alphabet $A = \{a, b\}$, montrez que la classe des langages reset-décomposables n'est pas fermée par union.

Question 14. La classe des langages reset-décomposable est-elle fermée par intersection ?

PARTIE 3. LES AUTOMATES RESET-DÉCOMPOSABLES ADMETTENT DES ALGORITHMES BINAIRES

Dans cette partie on montre que tout automate reset-décomposable admet un algorithme binaire et qu'un tel algorithme peut être construit de façon simple à partir de l'automate.

Question 15. Prouvez le lemme suivant.

Lemme d'addition. *L'automate $\mathcal{C}$ donné dans la Figure 6 admet l'algorithme binaire suivant :*

$$\begin{cases} p_0(r) = p_b(u) \vee \{p_c(u) \wedge [\neg p_b(u) + \neg(p_b(u) \vee p_c(u))]\} \\ p_1(r) = p_a(u) \vee \{p_c(u) \wedge \neg[p_a(u) + (p_a(u) \vee p_c(u))]\} \end{cases}$$

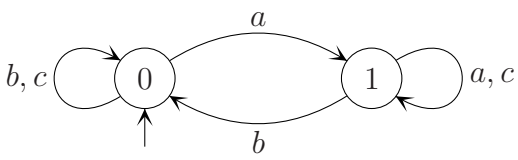

FIGURE 6 – L'automate $\mathcal{C}$

On définit la fonction suivante, où s est un état d'un automate, x et y sont des mots binaires :

$$Ind(s, x, y) = \begin{cases} x \vee [y \wedge (\neg x + \neg(x \vee y))] & \text{si } s \text{ est initial} \\ x \vee [y \wedge \neg(x + (x \vee y))] & \text{sinon} \end{cases}$$

Question 16. Soient $\mathcal{A} = (Q, A, q_i, F, \delta)$ un automate déterministe complet et $s \in Q$ un état reset-décomposable. On note E_s l'ensemble des lettres qui envoient tous les états sur s : $a \in E_s$ ssi $\forall q \in Q$, $\delta(q, a) = s$. On note B_s l'ensemble des lettres qui bouclent sur s et qui ne sont pas dans E_s : $a \in B_s$ ssi $a \notin E_s$ et $\delta(s, a) = s$. Montrez qu'on a alors :

$$p_s(r) = Ind(s, p_{E_s}(u), p_{B_s}(u))$$

où l'on pose, pour tout $B \subseteq A$, $p_B(u) = \bigvee_{a \in B} p_a(u)$.

Question 17. Prouvez le résultat suivant.

Théorème. *Soit $\mathcal{A}$ un automate reset-décomposable. L'automate $\mathcal{A}$ admet un algorithme binaire. De plus la taille de l'algorithme binaire (c.-à-d. son nombre d'opérations) peut être choisie linéaire en la taille de $\mathcal{A}$.*

Partie 4. Une caractérisation des automates admettant un algorithme binaire

Dans cette dernière partie, on donne une caractérisation structurelle des automates admettant des algorithmes binaires.

Définition. *Soit $\mathcal{A} = (Q, A, q_i, F, \delta)$ un automate déterministe complet. Pour un mot $u = a_1 \cdots a_n \in A^*$ et deux états $p, q \in Q$, on note $p \xrightarrow{u} q$ le fait que la lecture du mot u depuis l'état p termine en q, c'est à dire que $q = \delta(\ldots\delta(\delta(p, a_0), a_1), \ldots, a_n)$. Un mot $u \in A^*$ non vide est qualifié de* ***compteur pour l'automate*** *$\mathcal{A}$ s'il existe un entier $k \geq 2$ et k états deux à deux distincts $q_1, \cdots, q_k$ tels que $q_1 \xrightarrow{u} q_2 \xrightarrow{u} \ldots \xrightarrow{u} q_k \xrightarrow{u} q_1$. Un automate est* ***sans compteur*** *s'il n'existe pas de compteur pour cet automate.*

Définition. *Un mot $u = a_0 a_1 a_2 \cdots a_n$ est* ***périodique de période ℓ à partir du rang*** *r si pour tout $i, j \geq 0$ tels que $r + (i+1)\ell + j \leq n$, $a_{r+i\ell+j} = a_{r+(i+1)\ell+j}$.*

Question 18. Montrez que les opérations sur les mots binaires ($\vee$, $\wedge$, $\neg$, $\uparrow_0$, $\uparrow_1$, $+$) appliquées à des mots binaires périodiques à partir d'un rang r et de même période ℓ donnent des mots binaires périodiques de période ℓ à partir du rang r', où r' ne dépend que de r et de l'opération appliquée.

Question 19. En déduire qu'étant donné un algorithme binaire, si ce dernier est appliqué à des mots binaires périodiques à partir d'un rang r, de même période ℓ, le résultat est périodique de période ℓ à partir d'un rang r' qui ne dépend que de r et de l'algorithme.

Question 20. Prouvez que tout automate qui admet un algorithme binaire est sans compteur.

Définition. *Soient $\mathcal{A} = (Q, A, q_i, F, \delta)$ un automate déterministe complet et $a \in A$ une lettre. On dira que a est un* ***reset*** *si la fonction $x \mapsto \delta(x, a)$ est constante ou égale à l'identité. L'automate $\mathcal{A}$ est un* ***automate reset****, si pour tout $a \in A$, a est un reset.*

Question 21. Montrez que tout automate reset admet un algorithme binaire.

Définition. *Soient $\mathcal{B}_1 = (Q_1, A, q_{i,1}, \delta_1)$ et $\mathcal{B}_2 = (Q_2, Q_1 \times A, q_{i,2}, \delta_2)$ deux automates déterministes complets dont on ignore ici les états finaux. Leur* ***produit en cascade*** *$\mathcal{B}_1 \circ \mathcal{B}_2 = (Q, A, q_i, \delta)$ est l'automate déterministe complet défini de la façon suivante :*

- $Q = Q_1 \times Q_2$

– $q_i = (q_{i,1}, q_{i,2})$
– $\overline{\delta}((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, (q_1, a)))$

Le produit en cascade de k automates se définit récursivement par : $\mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_k = (\ldots((\mathcal{B}_1 \circ \mathcal{B}_2) \circ \mathcal{B}_3) \circ \ldots) \circ \mathcal{B}_k$.

Question 22. Soient $\mathcal{B}_1$ et $\mathcal{B}_2$ deux automates admettant chacun un algorithme binaire. Montrez qu'il en est de même pour leur produit en cascade $\mathcal{B}_1 \circ \mathcal{B}_2$.

Un **homomorphisme surjectif** d'un automate déterministe complet (dont on ignore les états finaux) $\mathcal{A}_1 = (Q_1, A, q_{i,1}, \delta_1)$ vers un automate déterministe complet (dont on ignore les états finaux) $\mathcal{A}_2 = (Q_2, A, q_{i,2}, \delta_2)$ est une fonction totale surjective $\varphi : Q_1 \to Q_2$ telle que $\varphi(q_{i,1}) = q_{i,2}$ et $\delta_1(q, a) = q' \Rightarrow \delta_2(\varphi(q), a) = \varphi(q')$.

Question 23. Soient $\mathcal{A}_1$ et $\mathcal{A}_2$ deux automates déterministes complets et soit φ un homomorphisme surjectif de $\mathcal{A}_1$ vers $\mathcal{A}_2$. Montrez que si $\mathcal{A}_1$ admet un algorithme binaire, il en est de même pour $\mathcal{A}_2$.

Définition. *Soit $\mathcal{A}$ un automate déterministe complet. Une décomposition en cascade de $\mathcal{A}$ est la donnée de $k \geq 1$ automates reset $\mathcal{B}_1, \cdots, \mathcal{B}_k$ et d'un homomorphisme surjectif partiel φ de $\mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_k$ vers $\mathcal{A}$.*

On admettra le théorème suivant :

Théorème (Krohn-Rhodes) Tout automate sans compteur admet une décomposition en cascade.

Question 24. Conclure des questions précédentes qu'un automate admet un algorithme binaire si et seulement s'il est sans compteur.