Centrale Informatique optionnelle MP 2022
Corrigé

Ce corrigé est proposé par Benjamin Monmege (enseignant-chercheur à l'université) ; il a été relu par Jean Starynkévitch (professeur en CPGE) et William Aufort (professeur en CPGE).

Ce sujet est entièrement concentré sur la partie « Motifs, automates et expressions » du programme d'option informatique. Il s'intéresse à trois algorithmes sur les automates finis et les expressions rationnelles.

- Dans la partie I, après quelques questions d'échauffement sur la notion de miroir d'un automate et sur les mots palindromes, l'algorithme de déterminisation d'un automate fini à l'aide de l'automate des parties est implémenté en OCaml. On étudie ensuite l'algorithme de Brzozowski qui permet de minimiser le nombre d'états d'un automate fini déterministe en appliquant deux fois les opérations de miroir et de déterminisation.
- Dans la partie suivante, on étudie les expressions rationnelles et on implémente l'algorithme de Conway permettant de prouver la réciproque du théorème de Kleene, à savoir que tout langage reconnu par un automate fini peut être décrit par une expression rationnelle. Pour cela, on emploie des matrices d'expressions rationnelles sur lesquelles on réalise des opérations de somme, de produit et d'étoile. L'algorithme de Conway utilise le paradigme « diviser pour régner » et sa complexité est analysée très finement.
- La partie III étudie le problème réciproque en proposant la construction d'un automate fini, non déterministe, à partir d'une expression rationnelle. Plutôt que la construction de l'automate de Glushkov au programme, le sujet propose la construction de l'automate d'Antimirov, qui utilise un mécanisme de dérivation formelle des expressions rationnelles. Cette dernière partie ne contient pas de question de programmation.

Ce sujet est très long et nécessite une bonne connaissance des automates et expressions rationnelles. Des questions manipulant des exemples alternent avec des questions de programmation, de complexité algorithmique et des questions plus théoriques, demandant de justifier la correction d'une construction à l'aide de récurrences. Le sujet reste parfaitement adapté au nouveau programme de l'option informatique en MP ou de l'informatique en MPI.

Indications

Partie I

2 Il suffit d'inverser états initiaux et finaux, et d'inverser le sens des flèches des transitions.

3 Après avoir formalisé la construction décrite dans l'indication précédente, on peut la justifier en montrant qu'il existe une bijection entre les chemins de l'automate A et ceux de l'automate $\widetilde{\text{A}}$ dont les étiquettes sont le miroir l'une de l'autre.

4 Commencer par écrire une fonction récursive qui renverse chaque transition dans une liste de transitions.

6 Distinguer les cas d'un mot de longueur paire et de longueur impaire, en essayant de limiter le nombre de tests à effectuer à environ la moitié de la longueur du mot. Utiliser ensuite une boucle `while` ou une fonction récursive.

8 Raisonner par l'absurde en supposant l'existence d'un automate fini déterministe reconnaissant le langage $\text{Pal}(\Sigma) \cap \mathcal{L}(a^*ba^*)$. En notant n son nombre d'états, considérer le mot $w = a^nba^n$ et exhiber une répétition d'états parmi les $n+1$ premiers états du chemin d'étiquette w.

9 Il suffit de modifier les états initiaux et finaux de l'automate A pour montrer que $\text{L}_{q,q'}$ est reconnaissable.

10 Raisonner par double inclusion.

11 Montrer que $\text{D}(\mathcal{L}(a^*b)) = \{a^nbba^n \mid n \in \mathbb{N}\}$ et $\text{R}(\mathcal{L}(a^*ba^*)) = \mathcal{L}(a^*b^*)$.

12 Montrer que D(L) est parfois reconnaissable, et parfois ne l'est pas. Montrer, à l'aide du résultat des questions 3 et 9, que R(L) est reconnaissable.

14 Une coquille s'est glissée dans l'énoncé puisque l'état initial de l'automate des parties est la partie I et non l'ensemble contenant une unique partie {I}.

16 Noter que l'automate des parties reconnaît le même langage que l'automate de départ, alors que l'automate $\widetilde{\text{A}}$ reconnaît le miroir du langage reconnu par A.

17 Commencer par écrire une fonction récursive prenant en arguments une liste et un élément x et qui renvoie la liste de laquelle on a supprimé toutes les occurrences de x.

19 Commencer par utiliser le quotient de la division euclidienne de k par 2^q, puis tester la parité du résultat.

20 Il suffit de parcourir la liste, en utilisant les fonctions `numero` et `est_dans`, ainsi que le tableau `pow` pour chaque élément de la liste.

21 Utiliser la fonction `List.exists` ou écrire une fonction récursive qui parcourt la liste ℓ et utilise la fonction `est_dans` pour chaque élément de celle-ci.

22 Une fonction récursive fait à nouveau l'affaire, afin de parcourir la liste des transitions et faire appel, pour chaque transition, aux fonctions `etat_suivant` et `est_dans`.

24 S'inspirer d'un parcours en profondeur récursif. Ne pas hésiter à découper le code avec des fonctions auxiliaires, en particulier pour parcourir (et ainsi générer) l'ensemble des états de l'automate des parties (en particulier en utilisant les fonctions `etat_suivant` et `cherche`) tout en maintenant le nombre d'états générés jusque-là et la liste de paires (k, v) décrite dans l'énoncé. Une autre fonction auxiliaire récursive permettra de repérer les états finaux dans la liste de paires générée préalablement, en utilisant la fonction `intersecte`.

26 À nouveau, la même coquille qu'en question 13 perturbe la lecture de l'énoncé : il faut lire $q \in \delta^*(\mathrm{I}, u)$. La notation $\delta^*(\mathrm{I}, u)$ décrit donc l'état X de $\mathrm{A}_{\mathrm{det}}$ qu'on atteint après avoir lu le mot u. Pour répondre à la question, montrer d'abord que pour tout mot $u \in \Sigma^*$ et pour tout état $q \in \delta^*(\mathrm{I}, u)$, il existe un état $q_0 \in \mathrm{I}$ et une suite de transitions dans A de q_0 à q d'étiquette u. Conclure ensuite en utilisant le fait que $\widetilde{\mathrm{A}}$ est accessible.

27 Dans la question 26, on a montré un résultat plus fort : « si $q \in \delta^*(\mathrm{I}, u)$, alors il existe un mot $w \in \Sigma^*$ tel que $uw \in \mathrm{L}$ et $\widetilde{\mathrm{A}}$ possède un chemin de f à q d'étiquette w. » Quant à elle, la propriété $(*)$ nécessite d'être corrigée, toujours à cause de la coquille de la question 13 : elle devient « si l'on prend deux mots u et v dans Σ^* tels que $u^{-1}\mathrm{L} = v^{-1}\mathrm{L}$, alors dans l'automate $\mathrm{A}_{\mathrm{det}}$, $\delta^*(\mathrm{I}, u) = \delta^*(\mathrm{I}, v)$ ». La montrer en raisonnant par double inclusion et en utilisant le résultat élargi de la question 26.

28 Appliquer la question 27 à l'automate $\widetilde{\mathrm{B}}$, après avoir vérifié qu'il satisfait les hypothèses.

Partie II

31 On peut décider récursivement si le langage représenté par une expression rationnelle est vide ou non.

34 Écrire une fonction récursive qui applique la simplification d'abord pour les sous-expressions les plus profondes, puis utilise les fonctions `su`, `sc` ou `se` selon l'opérateur courant.

38 Justifier une inégalité de la forme

$$\forall n \geqslant 2 \quad \mathrm{C}(n) = 2\,\mathrm{C}(n-1) + \mathrm{O}(n^2)$$

Faire apparaître un télescopage en divisant par 2^n puis sommer pour obtenir un majorant de $\mathrm{C}(n)$ en $\mathrm{O}(2^n)$.

39 Adapter la méthode de la question 38 après avoir établi une majoration de la forme

$$\forall n \geqslant 2 \quad \mathrm{C}(n) \leqslant 4\,\mathrm{C}(n/2) + \mathrm{O}(n^3)$$

41 Seul l'algorithme étudié dans les questions 39 et 40 est à implémenter.

42 Utiliser le résultat de la question 9 et faire le lien entre le vecteur X et les états initiaux de A, ainsi qu'entre le vecteur Y et les états finaux.

43 Ne pas hésiter à découper le code en fonction auxiliaire récursive, permettant de construire la matrice de transitions, ainsi que les vecteurs X et Y. Finalement, utiliser les fonctions `produit` et `etoile`.

Partie III

45 Poursuivre le calcul des dérivées initié dans la question 44 pour générer l'ensemble d'états de l'automate d'Antimirov de E. Vous devriez trouver quatre états.

47 Commencer par considérer le cas d'un mot w réduit à une lettre. Utiliser ensuite la propriété admise par l'énoncé pour se ramener au cas où S est un singleton $\{\mathrm{E}\}$. Prouver alors la propriété par récurrence sur la longueur du mot w. Dans la récurrence, on utilisera le résultat de la question 46.

48 Montrer la propriété par récurrence sur la longueur du mot w.

49 Conclure à l'aide des questions 47 et 48.

50 Procéder par induction sur la structure de l'expression rationnelle E (ou par récurrence sur le nombre d'opérations utilisées dans E). En déduire un majorant du nombre d'états de l'automate d'Antimirov.

I. Mots et automates

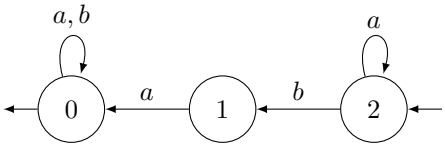
1 L'automate $\mathcal{A}_1$ est un automate non déterministe qui reconnaît tous les mots sur l'alphabet $\{a, b\}$ qui commencent par un nombre quelconque de lettres a et b, suivies du mot ab, puis d'éventuelles lettres a.

> On peut décrire ce langage par l'expression rationnelle $(a+b)^*\, a\, b\, a^*$.

Son langage miroir consiste donc en l'ensemble des mots qui commencent par un nombre arbitraire de a, suivi du mot ba, puis d'un mot quelconque composé des lettres a et b.

$$\mathrm{L}_1 = \{waba^n \mid w \in \{a,b\}^*, n \in \mathbb{N}\} \quad \text{et} \quad \widetilde{\mathrm{L}}_1 = \{a^n baw \mid w \in \{a,b\}^*, n \in \mathbb{N}\}$$

2 On obtient l'automate $\widetilde{\mathcal{A}}_1$ ci-dessous qui reconnaît le langage miroir $\widetilde{\mathrm{L}}_1$ en inversant états initiaux et finaux et en inversant le sens des transitions :



> Notons que cet automate est déterministe, alors que l'automate $\mathcal{A}_1$ est non déterministe.

3 Généralisons l'approche appliquée dans la question 2, en construisant l'automate miroir $\widetilde{\mathrm{A}} = (\mathrm{Q}, \mathrm{I}', \mathrm{F}', \mathrm{T}')$ de l'automate non déterministe $\mathrm{A} = (\mathrm{Q}, \mathrm{I}, \mathrm{F}, \mathrm{T})$ en définissant :

- l'ensemble $\mathrm{I}' = \mathrm{F}$ des états initiaux de $\widetilde{\mathrm{A}}$ comme les états finaux de A ;
- l'ensemble $\mathrm{F}' = \mathrm{I}$ des états finaux de $\widetilde{\mathrm{A}}$ comme les états initiaux de A ;
- l'ensemble $\mathrm{T}' = \{(q', a, q) \mid (q, a, q') \in \mathrm{T}\}$ de transitions de $\widetilde{\mathrm{A}}$ obtenu en inversant le sens des transitions de A.

Montrons que l'automate miroir $\widetilde{\mathrm{A}}$ reconnaît le langage miroir $\widetilde{\mathrm{L}}$. Pour cela, exhibons une bijection f entre les chemins c dans l'automate A d'un état de I à un état de F et les chemins c' dans l'automate $\widetilde{\mathrm{A}}$ d'un état de I′ à un état de F′ tel que si le chemin c est étiqueté par un mot w alors le chemin c' est étiqueté par le mot $\widetilde{w}$. Pour tout chemin c dans A décrit par la suite de transitions

$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \cdots q_{n-1} \xrightarrow{a_{n-1}} q_n$$

définissons la suite $f(c)$ par

$$q_n \xrightarrow{a_{n-1}} q_{n-1} \xrightarrow{a_{n-2}} \cdots q_1 \xrightarrow{a_0} q_0$$

Tout d'abord notons que si $w = a_0 a_1 \cdots a_{n-1}$ est l'étiquette du chemin c, alors le mot $a_{n-1}a_{n-2}\cdots a_0 = \widetilde{w}$ est bien l'étiquette de la suite $f(c)$. Montrons ensuite que $f(c)$ est un chemin dans $\widetilde{\mathrm{A}}$:

- il commence dans l'état q_n qui est final dans A et donc initial dans $\widetilde{\mathrm{A}}$;
- il termine dans l'état q_0 qui est initial dans A donc final dans $\widetilde{\mathrm{A}}$;
- pour tout $i \in [\![\, 0\,;n-1\,]\!]$, (q_i, a_i, q_{i+1}) est une transition de A, donc (q_{i+1}, a_i, q_i) est une transition de $\widetilde{\mathrm{A}}$.

On a donc bien toutes les propriétés assurant que $f(c)$ est un chemin de $\widetilde{\mathrm{A}}$ d'étiquette $\widetilde{w}$.