MITRO210: Automates et données structurées

Feuille d'exercices 1

Antoine Amarilli

1 Automates inambigus

Dans cet exercice, on cherche à définir un algorithme permettant de déterminer, étant donné en entrée un automate nondéterministe (NFA), si cet automate est inambigu. L'idée de cet algorithme, et de sa généralisation au cas k-inambigu étudié en dernière question, est dans [2], Observation 5.1.

On rappelle qu'étant donné un NFA $A = (Q, \Sigma, I, F, \delta)$ où $\delta \subseteq Q \times \Sigma \times Q$, un *run* de A sur un mot $w = a_1 \cdots a_n$ de Σ^* est une séquence d'états $q_0, \ldots, q_n$ avec $q_0 \in I$ et $(q_i, a_{i+1}, q_{i+1}) \in \delta$ pour chaque $0 \leq i < n$. Le run *finit* en l'état q_n. Un run est *acceptant* si l'état par lequel il finit est un état final, i.e., si $q_n \in F$. On rappelle que le mot w est *accepté* par A s'il existe un run acceptant de A sur w.

On dit que A est *inambigu* si, pour chaque mot w, le NFA A a au plus un run acceptant sur w.

Question 0. Proposer un algorithme naïf permettant de vérifier si un automate est inambigu ou non. En termes de décidabilité, que nous apprend cet algorithme sur le problème de décision consistant à déterminer si un automate est inambigu ?

On rappelle qu'un état q de A est *accessible* s'il existe un chemin dans A depuis un état initial vers q ; et que q est *co-accessible* s'il existe un chemin dans A depuis q vers un état final. L'automate A est dit *émondé* si chaque état est à la fois accessible et co-accessible.

Question 1. Montrer que, si A est émondé et inambigu, alors il satisfait la condition (*) : pour tout $q \in Q$, pour chaque mot $w = a_1 \cdots a_n$, il y a au plus un unique run de A sur w qui finisse par q. Ceci est-il vrai sans supposer que A est émondé ?

On dit que deux états $q \neq q'$ sont *confondables* s'il existe un mot w tel que A a un run sur w qui finisse en q et A a un run sur w qui finisse en q'.

Question 2. On dit que deux états q et q' sont *fusionnables* s'ils ont tous deux une transition étiquetée par une certaine lettre a vers un même état q''. Montrer que, si deux états confondables de A sont fusionnables, alors la condition (*) n'est pas vérifiée.

Question 3. Montrer que, si deux états finaux sont confondables, alors A n'est pas inambigu.

Question 4. Proposer un algorithme pour calculer les paires d'états confondables.

Question 5. En conclure comment tester en temps polynomial, étant donné un NFA, s'il est inambigu ou non.

Question 6 (bonus). Pour une constante $k \in \mathbb{N}$ fixée, un NFA A est *k-inambigu* si, pour chaque mot $w = a_1 \cdots a_n$, il existe au plus k runs acceptants de A sur w. Peut-on tester en temps polynomial, étant donné un NFA, s'il est k-inambigu ou non ?

2 Automates bidirectionnels

Dans cet exercice, on cherche à calculer, étant donné un automate bidirectionnel, un automate unidirectionnel qui reconnaît le même langage. Cet exercice suit [3, 1].

On rappelle qu'un NFA bidirectionnel (2NFA) est défini comme un quintuplet $A = (Q, \Sigma, I, F, \delta)$ avec $\delta \subseteq Q \times \Sigma \times \{-1, 0, 1\} \times Q$. Une *configuration* de A est une paire (q, i) avec $i \in \mathbb{N}$. Un *run* de A sur un mot $w = a_1 \cdots a_n$ de Σ^* est une séquence de configurations $(q_0, i_0), \ldots, (q_m, i_m)$ telle qu'on ait :

— $q_0 \in I$ et $i_0 = 1$,
— $1 \leq i_j \leq n$ pour chaque $0 \leq j < m$
— pour chaque $0 \leq j < m$, en écrivant $d_j = i_{j+1} - i_j$, on a $(q_j, a_{i_j}, d_j, q_{j+1}) \in \delta$.

Le run est *acceptant* si $q_m \in F$ et $i_m = n + 1$.

Question 0. Rappeler pourquoi, étant donné un NFA, on peut calculer un 2NFA reconnaissant le même langage.

Étant donné le 2NFA A et un mot w de longueur n, pour chaque $1 \leq i \leq n + 1$, on définit l'ensemble S_i comme l'ensemble des états q de A tel qu'il existe un run de A sur w se finissant sur la paire (q, i).

Question 1. Quelles conditions sont satisfaites par la séquence $S_1, \ldots, S_n, S_{n+1}$? Exprimer d'abord une condition satisfaite par S_1, puis des conditions satisfaites par chaque S_i, et enfin des conditions satisfaites par les paires (S_i, S_{i+1}) d'ensembles correspondant à deux positions contiguës.

Question 2. Si le mot w n'est pas accepté par A, quelle autre condition est satisfaite par $S_1, \ldots, S_{n+1}$?

Un *étiquetage* de w pour A est une séquence de sous-ensembles d'états $S_1, \ldots, S_{n+1}$.

Question 3. Montrer que, s'il existe un étiquetage de w pour A qui satisfait les conditions des questions 1 et 2, alors w n'est pas accepté par A.

Question 4. En déduire un algorithme pour construire, à partir de A, un automate unidirectionnel qui reconnaisse le complémentaire du langage de A. Quel type d'automate obtient-on ? quelle est la complexité de l'algorithme ?

Question 5. Si on souhaite obtenir un automate unidirectionnel qui reconnaisse le langage de A et non son complémentaire, comment peut-on procéder ? quelle est la complexité ?

Question 6 (bonus). Peut-on obtenir un automate unidirectionnel déterministe qui reconnaisse le langage de A de manière plus efficace qu'à la question précédente ?

Références

[1] J. C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3(2), 1959. https://ieeexplore.ieee.org/abstract/document/5392614.

[2] R. E. Stearns and H. B. Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM Journal on Computing*, 14(3), 1985. https://epubs.siam.org/doi/abs/10.1137/0214044.

[3] M. Y. Vardi. A note on the reduction of two-way automata to one-way automata. *Information Processing Letters*, 30(5), 1989. https://www.cs.rice.edu/~vardi/papers/ipl88rj.pdf.