MITRO210 : Automates et données structurées

Feuille d'exercices 2

Antoine Amarilli

1 Automates alternants

Dans cet exercice, on cherche à calculer, étant donné un automate alternant, un automate nondéterministe qui reconnaît le même langage. On suit la construction expliquée dans [2].

On rappelle qu'un automate alternant est défini comme $A = (Q_\exists \sqcup Q_\forall, \Sigma, I, F, \delta)$ où l'ensemble d'états Q est scindé entre les états de $Q_\exists$ (dits *existentiels*), et les états de $Q_\forall$ (dits *universels*). Le *langage reconnu par A* est le langage des mots acceptés par A à partir d'un état q_0 de I, où le *langage de A accepté à partir d'un état q* est défini inductivement :

- Le mot vide ϵ est accepté à partir de q si q est final ;
- Pour un mot non-vide de la forme av avec $a \in \Sigma$ et $v \in \Sigma^*$, pour un état q, le mot av est accepté à partir de q dans les cas suivants :
  - Si $q \in Q_\exists$, alors av est accepté à partir de q s'il existe un état q' avec $(q, a, q') \in \delta$ et tel que v soit accepté à partir de q'.
  - Si $q \in Q_\forall$, alors av est accepté à partir de q si, pour tout état q' avec $(q, a, q') \in \delta$, on a que v est accepté à partir de q'.

Question 0. À quoi correspond le cas où tous les états de A sont existentiels ?

Question 1. Supposons que A a un unique état initial q_0 qui est universel, qu'il n'y a pas de transition qui pointe vers q_0, et que tous les autres états sont existentiels. Proposer un algorithme permettant de déterminer, étant donné un mot w, si w est reconnu par A.

Question 2. Sous la même hypothèse qu'en question 1, proposer un algorithme pour calculer un automate unidirectionnel qui reconnaisse le même langage que A. Quel est, asymptotiquement, le nombre d'états de l'automate obtenu ? Quel genre d'automate obtient-on ?

Question 3. Généraliser cette construction à un automate alternant A arbitraire. Quelle est la complexité de cet algorithme ? Quel genre d'automate obtient-on ?

2 Recherche de sous-mots non-contigus

Dans cet exercice, on cherche à trouver un mot u comme sous-mot (non nécessairement contigu) d'un mot v.

Question 0. Proposer un algorithme naïf pour déterminer, étant donné u et v, si u est un sous-mot de v. Quelle est sa complexité ?

Question 1. Proposer un algorithme plus efficace pour cette tâche et déterminer sa complexité.

Question 2. On souhaite indexer un mot v pour pouvoir répondre rapidement à la question suivante : étant donné n'importe quel mot u, déterminer si u est un sous-mot de v. Proposer une structure de données pour ce faire. Décrire la complexité du précalcul en fonction de v et la complexité de la recherche en fonction de u.

On souhaite à présent répondre au problème du *sous-mot commun le plus court* (en anglais *longest common subsequence*) : étant donné deux mots u et v, quelle est la plus grande longueur d'un mot w qui soit un sous-mot de u et de v simultanément.

Question 3. Expliquer comment la question précédente nous permet d'obtenir, à partir d'un mot v, un automate qui accepte précisément les mots qui sont des sous-mots de v. Quelle est la complexité de cette construction ?

Question 4. En déduire la construction d'un automate qui, étant donné deux mots u et v, accepte précisément les mots qui sont des sous-mots de u et de v. Quelle est la complexité de cette construction ?

Question 5. En déduire un algorithme pour le problème du sous-mot commun le plus court. Quelle est sa complexité ? Peut-on interpréter différemment cet algorithme ?

On considère maintenant le problème du sous-mot commun le plus court sur un ensemble de chaînes : étant donné des mots $U = \{u_1, \ldots, u_n\}$, quelle est la plus grande longueur d'un mot w qui soit un sous-mot de tous les mots de U ?

Question 6. Si la valeur de n est une constante fixée au départ (par exemple, $n = 42$), quelle complexité peut-on atteindre ?

Question 7. On ne suppose plus que n est fixé au départ. Si la longueur maximale ℓ des mots de U est bornée par une constante fixée (par exemple longueur d'au plus 42), quelle complexité peut-on atteindre ?

Remarque : on peut montrer que, si n et ℓ ne sont pas fixés, le problème est NP-difficile : cf [1], Proposition 1.

Références

[1] G. Blin, L. Bulteau, M. Jiang, P. J. Tejada, and S. Vialette. Hardness of longest common subsequence for sequences with bounded run-lengths. In *Combinatorial Pattern Matching : 23rd Annual Symposium, CPM 2012, Helsinki, Finland, July 3-5, 2012. Proceedings 23*, pages 138–148. Springer, 2012.

[2] H. J. (`https://cs.stackexchange.com/users/4287/hendrik-jan`). How to convert AFA to ϵ-NFA / NFA / DFA ? Computer Science Stack Exchange. URL : `https://cs.stackexchange.com/q/151660` (version : 2022-05-17).