MITRO210: Automates et données structurées

Feuille d'exercices 1 – Corrigé

Antoine Amarilli

Avertissement : ce corrigé n'a été que peu relu. N'hésitez pas à me contacter pour toute question ou erreur.

1 Automates inambigus

Question 0. On peut simplement énumérer l'ensemble infini de tous les mots, par exemple par ordre shortlex (par longueur puis par ordre lexicographique), et pour chaque mot tester tous ses runs dans l'automate pour voir s'il y en a plusieurs acceptants.

Cet algorithme naïf ne termine que si l'automate est ambigu (et dans ce cas il finit toujours par le signaler), sinon il ne termine pas. Il s'agit donc d'un algorithme de semi-décision, donc le problème de déterminer si un automate est ambigu est semi-décidable, ainsi le problème de déterminer si un automate est inambigu est co-semi-décidable.

On pourrait montrer qu'il y a une borne sur la longueur maximale d'un contre-exemple au fait qu'un automate soit inambigu, ce qui impliquerait la décidabilité. Une telle borne peut se déduire de la suite de l'exercice.

Question 1. On procède par contraposition : si A ne satisfait pas (*), alors il existe un état q et un mot w tel qu'il y a deux runs distincts de A sur w qui finissent par q. Comme A est émondé, l'état q est co-accessible, ainsi il y a un chemin de q à un état final. On peut concaténer chaque run avec ce chemin pour aboutir à deux runs acceptants différents de A sur w. Ainsi. A n'est pas inambigu.

Ce raisonnement utilise l'hypothèse que A est émondé. En effet, si un automate A a un état q et un mot w où il y a deux runs différents de A sur w qui mènent en q, mais que q n'est pas co-accessible, alors cela ne suffit pas à réfuter l'inambiguïté de A.

Question 2. Comme q et q' sont confondables, il existe un mot w tel que A a un run ρ sur w qui mène à q, et A a un run ρ' sur w qui mène à q'. Suivant notre hypothèse, soit a une lettre et q'' un état tel que A a une transition étiquetée a de q à q'' et de q' à q''. On voit alors que les runs $\rho q''$ et $\rho' q''$ sont deux runs de A sur le mot wa qui mènent tous deux en q''. Ainsi, la condition (*) n'est pas vérifiée, comme en témoignent q'' et wa.

Question 3. S'il existe un mot w tel que A a un run sur w qui mène à deux états finaux différents, alors ce sont nécessairement deux runs différents, et ils sont tous deux acceptants. Ainsi A n'est pas inambigu.

Question 4. On propose l'algorithme suivant :

— Cas de base : Initialiser un ensemble S de paires d'états $\{q, q'\}$ pour $q \neq q'$ avec les cas suivants :
  — Si q et q' sont tous deux initiaux, alors on les ajoute tous deux à S.
  — S'il existe un état q'' accessible et une lettre a tel que les transitions (q'', a, q) et (q'', a, q') existent dans δ, alors on ajoute $\{q, q'\}$ à S.
— Induction : pour toute paire $\{q, q'\}$ d'états dans S , pour toute lettre a, pour tous états q_2 et q'_2 tels que les transitions (q, a, q_2) et (q', a, q'_2) existent dans δ, si $q_2 \neq q'_2$, ajouter $\{q_2, q'_2\}$ à S.

On observe d'abord que cet algorithme termine après un nombre polynomial d'étapes : en effet le nombre de paires est quadratique en l'automate. Le temps d'exécution de l'algorithme est donc polynomial, vu que le nombre d'étapes pour chaque paire est polynomial.

On prétend ensuite que les paires de S sont bien des paires d'états distincts et fusionnables. Pour le premier cas de base, c'est clair avec le mot vide. Pour le second cas de base, c'est clair. Pour le cas d'induction, si q et q' sont fusionnables avec un mot w, alors q_2 et q'_2 sont bien différents et le mot wa témoigne du fait que q et q' sont fusionnables.

On prétend enfin que toute paire d'états fusionnables est dans S à la fin de l'exécution. En effet, supposons que q et q' soient fusionnables, soit w un mot qui en atteste, et ρ et ρ' les deux runs correspondants. De deux choses l'une : soit ρ et ρ' diffèrent en chaque position, soit il existe une position i où $\rho_i = \rho'_i$.

Dans le premier cas, on sait par application du premier cas de base que la paire $\{\rho_1, \rho'_1\}$ est dans S, puisque ce sont deux états initiaux distincts. On constate par une induction immédiate que $\{\rho_i, \rho'_i\}$ est dans S pour chaque i, puisque ce sont deux états distincts par hypothèse. Ainsi, pour $i = |\rho|$, on voit que $\{q, q'\}$ est dans S.

Le second cas est analogue mais on considère la dernière position i où ρ et ρ' sont égaux (on a $i < |\rho|$ nécessairement car leurs derniers états sont différents), on applique le second cas de base pour constater que $\{\rho_{i+1}, \rho'_{i+1}\}$ est dans S, et on conclut comme dans le premier cas.

Question 5. On commence par émonder l'automate, ce qui ne change pas la réponse (les états non-accessibles et non-coaccessibles n'interviennent pas dans la définition des runs acceptants donc ne changent pas le caractère ambigu ou non de l'automate).

On applique l'algorithme de la question précédente pour matérialiser l'ensemble S des paires d'états confondables.

On sait que, si deux états confondables sont fusionnables, alors la condition (*) n'est pas vérifiée et ainsi l'automate n'est pas inambigu (car il est émondé). On peut tester en temps polynomial, pour chaque paire d'états de S, si les états sont fusionnables ou non. On sait aussi que si deux états finaux sont confondables alors l'automate n'est pas inambigu.

Il reste à démontrer que la direction inverse de la caractérisation proposée est vraie, c'est-à-dire que si un automate n'est pas inambigu alors on le détecte correctement. Supposons que A n'est pas inambigu, et soit A un mot sur lequel A a deux runs acceptants ρ et ρ'. On veut montrer que l'une des conditions des questions 3 et 4 sont vérifiées. On sait grâce à ρ et ρ' que, pour chaque $1 \leq i \leq |\rho|$, les états ρ_i et ρ'_i sont soit égaux soit confondables. Or comme ρ et ρ' sont différents il y a forcément un i où ρ_i et ρ'_i sont différents. Prenons le dernier tel i. Il y a deux cas : soit $i < |\rho|$, soit $i = |\rho|$. Dans le premier cas, on voit que ρ_i et ρ'_i sont deux états confondables et ρ_{i+1} témoigne du fait qu'ils sont fusionnables. Dans le second cas, on voit que ρ_i et ρ'_i sont deux états confondables finaux. Ainsi, dans les deux cas, l'algorithme a détecté que A n'était pas ambigu.

Question 6. On peut généraliser la construction de cet exercice à la détection de la k-inambiguïté, en considérant des $k+1$-uplets d'états qui peuvent être atteints simultanément. En revanche une subtilité est qu'il faut considérer des états non nécessairement distincts, et se souvenir également de la relation d'équivalence sur les positions du tuple correspondant à des états égaux mais qui ont été précédemment distingués dans le run.

Plus précisément, l'algorithme est le suivant :

— Initialisation : tous les $k+1$-uplets d'états initiaux (non nécessairement distincts), avec la relation d'équivalence correspondant à l'égalité entre états. (Noter que dans le cas $k = 1$, on ne fait ici que le premier cas de base de l'algorithme précédent : mais comme on s'autorise à ce que les deux états soient égaux, le second cas de base sera capturé par la récurrence qui suit.)
— Récurrence : pour tout $k+1$-uplet et relation d'équivalence sur les états, pour toute lettre, pour tout choix de transitions pour chaque état, on parvient au $k+1$-uplet des nouveaux états. La relation d'équivalence est raffinée de la façon suivante : toutes les positions distinguées restent distinguées, et on distingue à présent les positions précédemment non distinguées qui portent des états différents.
— Terminaison : si on atteint un tuple où tous les états sont finaux et toutes les positions sont distinguées, l'automate est ambigu ; sinon il est inambigu.

L'invariant est que, pour tout tuple d'états atteint avec une relation sur les positions du tuple, il existe un mot dont la lecture par l'automate peut conduire à chacun des états du tuple, et les runs entre deux positions distinguées doivent être différents. Ainsi on dit que l'automate est inambigu s'il y a $k+1$ runs différents sur le même mot qui mènent à des états acceptants.

2 Automates bidirectionnels

Question 0. On définit simplement le 2NFA A' à partir du NFA A avec le même ensemble d'états, avec le même ensemble d'état initiaux, le même ensemble d'états finaux, et la relation de transition $\{(q, a, 1, q') \mid (q, a, q') \in \delta\}$, où δ est la relation de transition de A.

On voit alors que tout run acceptant $q_0, \ldots, q_n$ de A sur un mot w donne lieu à un run acceptant $(q_0, 1), \ldots, (q_n, n+1)$ sur w, et ainsi w est également accepté par A'.

À l'inverse, pour un run acceptant de A' sur un mot w de longueur n, par construction la seconde composante doit être incrémentée à chaque transition donc le run est de la forme $(q_0, 1), \ldots, (q_n, n+1)$ où q_n est final ; le run $q_0, \ldots, q_n$ témoigne alors du fait que A accepte également w.

Question 1. La séquence $S_1, \ldots, S_n$ satisfait les conditions suivantes, où on note $w = a_1 \cdots a_n$:

— Pour tout état initial q, on a $q \in S_1$;
— Pour chaque $1 \leq i \leq n$, pour chaque état $q \in S_i$, pour chaque transition de la forme $(q, a_i, 0, q')$, on a $q' \in S_i$;
— Pour chaque $1 \leq i \leq n$, pour chaque état $q \in S_i$, pour chaque transition de la forme $(q, a_i, 1, q')$, on a $q' \in S_{i+1}$;
— Pour chaque $1 < i \leq n$, pour chaque état $q \in S_i$, pour chaque transition de la forme $(q, a_i, -1, q')$, on a $q' \in S_{i-1}$;

Question 2. Si le mot n'est pas accepté, on a par ailleurs que S_{n+1} ne contient aucun état final.

Question 3. Supposons qu'il existe un tel étiquetage $S_1, \ldots, S_{n+1}$. Considérons un run (q_0, i_0), $\ldots, (q_m, i_m)$ de A sur le mot $w = a_1 \cdots a_n$. On va démontrer que, pour chaque $0 \leq j \leq m$, on a $q_j \in S_{i_j}$. Ainsi, on en déduira notamment que $q_m \in S_{i_m}$. Comme S_{n+1} ne contient aucun état final (hypothèse de la question 2), on saura alors que, si $i_m = n+1$, alors q_m n'est pas final, ce qui garantira que le run n'est pas acceptant.

Le cas de base est que q_0 est forcément un état final et $i_0 = 1$, or on a bien que S_1 contient tous les états finaux.

Les cas d'induction sont immédiats, en appliquant un des trois derniers cas de la question 1 en fonction de la direction de la transition. Plus précisément, pour $0 \leq j < m$, on a par hypothèse d'induction que $q_j \in S_{i_j}$. On pose $d_j = i_{j+1} - i_j$. On sait que $(q_j, a_{i_j}, d_j, q_{j+1}) \in \delta$ par définition d'un run. On distingue selon la valeur de d_j. Si $d_j = 0$, alors $i_{j+1} = i_j$, et le deuxième point de la question 1 nous donne que $q_{j+1} \in S_{i_{j+1}}$. Les deux autres cas sont analogues.

La conclusion de l'induction ainsi démontrée est que $q_m \in S_{i_m}$ et ainsi le mot w n'est pas accepté par A.

Question 4. On considère le 2NFA A et son ensemble d'états Q, et on cherche à définir un NFA A' qui reconnaisse le complémentaire du langage de A. L'ensemble d'états de A' sera $Q' := 2^Q \sqcup 2^Q \times 2^Q$, c'est-à-dire les sous-ensembles d'états de Q et les couples de sous-ensembles d'états de Q. Appelons *condition 0* le 2e point de la définition de la question 1. Les états initiaux de A' seront les sous-ensembles dans 2^Q qui contiennent tous les états initiaux de A. Les états finaux de Q' seront de deux types :

— Les sous-ensembles de 2^Q qui ne contiennent pas d'état final ;
— Les couples $(X, Y) \in 2^Q \times 2^Q$ de sous-ensembles de Q où Y ne contient pas d'état final.

Les transitions de A' seront de deux types, où on appelle *condition 1* et *condition -1* les deux derniers points de la définition de la question 1 :

— Une transition étiquetée a de $X \in 2^Q$ à $(X, Y) \in 2^Q \times 2^Q$ si X satisfait la condition 0 et X et Y satisfont la condition 1 avec la lettre a
— Une transition étiquetée a de $(X, Y) \in 2^Q \times 2^Q$ à $(Y, Z) \in 2^Q \times 2^Q$ si Y satisfait la condition 0 et Y et X satisfont la condition -1 avec a et Y et Z satisfont la condition 1 avec a.

Attention, ces définitions sont relativement subtiles : il faut prendre garde au fait que la dernière transition du run nous mènera à un état dont on ne pourra intuitivement pas revenir (vu que la 2e composante du dernier tuple de la configuration vaudra $n+1$), ainsi par exemple il ne faut pas imposer la condition 0 au dernier tuple atteint.

On prétend que l'automate A' accepte si et seulement si l'automate A n'accepte pas. Supposons d'abord que l'automate A' accepte. Si le run contient un seul état, alors c'est un run sur le mot vide, c'est un état initial et final, et ainsi il y a un sous-ensemble de Q qui contient tous les états initiaux et ne contient pas d'état final, c'est-à-dire qu'aucun état initial de Q n'est final, et ainsi A n'accepte pas le mot vide (noter qu'un run acceptant de A sur le mot vide contient nécessairement une unique configuration).

Si le run contient plusieurs états, il est nécessairement de la forme $X_1, (X_1, X_2), \ldots, (X_n, X_{n+1})$. On prétend alors que $X_1, \ldots, X_{n+1}$ est un étiquetage satisfaisant les conditions des questions 1 et 2. De fait, X_1 contient tous les états initiaux de A par définition de A', chaque X_i pour $1 \leq i \leq n$ satisfait la condition 0 pour la bonne lettre (c'est imposé par chaque transition sur

le premier membre du couple), et de même chaque paire (X_i, X_{i+1}) avec $1 \leq i \leq n$ satisfait la condition 1 et chaque paire (X_i, X_{i+1}) avec $1 \leq i < n$ satisfait la condition -1. Enfin, X_{n+} ne contient aucun état final par définition des états finaux de A' du 2e type. Donc dans tous les cas on obtient un étiquetage satisfaisant les conditions de la question 1 et 2, donc on sait que A n'accepte pas le mot.

Réciproquement, supposons que A n'accepte pas le mot. Les questions 1 et 2 nous montrent alors qu'il existe un étiquetage $S_1, \ldots, S_{n+1}$ défini comme avant la question 1. Si le mot est vide alors on sait que A a un état à la fois initial et final, et A' accepte aussi. Sinon, l'étiquetage $S_1, \ldots, S_{n+1}$ nous donne un run $S_1, (S_1, S_2), \ldots, (S_n, S_{n+1})$ de l'automate A, et l'automate A accepte.

L'automate A obtenu est un NFA : en effet les transitions suivies ne sont pas déterministes, puisque les états atteints ne sont pas forcément les successeurs des états précédents mais peuvent être obtenus indirectement. L'algorithme a une complexité exponentielle en l'automate de départ.

Question 5. On ne peut pas facilement modifier la construction précédente pour obtenir un automate qui reconnaisse le langage de A et non son complémentaire. En effet, un étiquetage justifiant du fait qu'un mot est accepté devrait non seulement satisfaire les propriétés des questions 1 et 2, mais être minimal, c'est-à-dire que chaque état atteint doit pouvoir effectivement être atteint (il ne faut pas qu'il y ait de "justification circulaire" entre états).

On peut en revanche complémenter l'automate de la question 4 ; pour ce faire, il faut d'abord le déterminiser, puis le compléter, et enfin le complémenter. La déterminisation implique une nouvelle exponentielle, on obtient donc une complexité doublement exponentielle en l'automate de départ.

Question 6. La construction de Shepherdson, décrite dans l'article de Vardi, permet d'obtenir un automate déterministe qui reconnaisse le même langage qu'un 2NFA de manière plus efficace qu'à la question précédente, mais elle est un peu plus difficile à décrire. Elle permet d'obtenir un automate déterministe pour le langage, mais au prix d'une complexité exponentielle en $|A|^2$, alors que la construction précédente (qui donne un automate nondéterministe pour le complémentaire) est exponentielle en $|A|$.