Problèmes de non-vacuité d'automates

Tous documents autorisés. On demande des démonstrations courtes, exhibant avec le minimum de détails les idées essentielles.

Un *automate déterministe* est un quintuplet $\mathcal{A} = (\Sigma, Q, q_I, \delta, F)$, où Σ est un ensemble fini appelé l'*alphabet*, Q est l'ensemble fini dit des *états*, $q_I \in Q$ est l'*état initial*, $F \subseteq Q$ est l'ensemble des *états finaux*, et $\delta : Q \times \Sigma \to Q$ est la *fonction de transition*. On dit que Σ est l'*alphabet de* $\mathcal{A}$.

On étend δ à une fonction de $Q \times \Sigma^* \to Q$ par prolongement homomorphe, c'est-à-dire: $\delta(q, \epsilon) = q$, $\delta(q, aw) = \delta(\delta(q, a), w)$ pour tout $a \in \Sigma$, $w \in \Sigma^*$. (On note ϵ le mot vide.)

Le *langage* $L_q(\mathcal{A})$ de l'état $q \in Q$ de l'automate $\mathcal{A}$ est l'ensemble des $w \in \Sigma^*$ tels que $\delta(q_I, w) = q$. Le *langage* $L(\mathcal{A})$ de l'automate $\mathcal{A}$ est $\bigcup_{q \in F} L_q(\mathcal{A})$.

On va examiner un certain nombre de problèmes sur les automates déterministes. Un automate déterministe sera décrit comme suit sur la bande d'entrée. Les lettres de Σ sont par convention des numéros de 1 à M, et l'on décrit l'entier M en binaire sur la bande. Les états de Q sont numérotés de 1 à N, N étant aussi donné en binaire. L'état initial q_I et les états de F sont données sous forme d'entiers binaires, et δ est décrit sous forme de la liste des $\delta(q, a)$, où $q \in \{1, \ldots, N\}$ et $a \in \{1, \ldots, M\}$ sont listés dans l'ordre lexicographique. On observe en particulier que calculer $\delta(q, a)$, pour n'importe quel $q \in Q$ et $a \in \Sigma$, se fait en temps polynomial et espace logarithmique.

La *taille* $|\mathcal{A}|$ est donc de l'ordre de $\log M + \log N + MN \log N$.

I. Test de vacuité.

On considère le problème de décision suivant NON-VACUITÉ.
ENTRÉE: un automate déterministe $\mathcal{A}$ sur Σ.
QUESTION: $L(\mathcal{A})$ est-il non vide?

1. Montrer que NON-VACUITÉ se réduit en espace logarithmique au problème de l'accessibilité dans les graphes orientés.

2. En déduire de NON-VACUITÉ est dans NL. (Il s'agit de la classe que nous avons appelé NLOGSPACE en cours.)

3. Par une réduction en sens inverse, montrer que NON-VACUITÉ est NL-complet.

4. Montrer que VACUITÉ, c'est-à-dire le problème suivant, est NL-complet aussi.
   ENTRÉE: un automate déterministe $\mathcal{A}$ sur Σ.
   QUESTION: $L(\mathcal{A})$ est-il vide?

II. Automates non déterministes de mots.

On rappelle qu'un automate non déterministe se définit comme un quintuplet $\mathcal{A} = (\Sigma, Q, I, \delta, F)$, où Σ est l'alphabet, Q est l'ensemble des états, $I \subseteq Q$ est l'ensemble des états initiaux, $F \subseteq Q$ est l'ensemble des états finaux, et $\delta : \Sigma \to \mathbb{P}(Q \times Q)$ est la relation de transition (il y a une transition étiquetée a de q vers q' si et seulement si $(q, q') \in \delta(a)$).

Le mot w est *accepté* depuis q par $\mathcal{A}$ si et seulement si, soit $w = \epsilon$ et $q \in F$, soit w est de la forme aw', il existe un état q' tel que $(q, q') \in \delta(a)$ et w' est accepté depuis q'. Le *langage* $L(\mathcal{A})$ est l'ensemble des mots acceptés depuis un état initial.

On considère le problème suivant ND-NON-VACUITÉ.
ENTRÉE: un automate non déterministe $\mathcal{A}$ sur Σ.
QUESTION: $L(\mathcal{A})$ est-il non vide?

1. Montrer que ND-NON-VACUITÉ est NL-complet.

III. Automates non déterministes d'arbres.

Un automate d'*arbres* est un mécanisme permettant de définir des langages non plus de mots mais de termes du premier ordre, sur une signature fixée.

Un automate d'arbres (non déterministe) est un quadruplet (Σ, Q, I, δ), où Σ est la *signature*, c'est-à-dire un ensemble fini de symboles de fonction donnés avec leurs arités; Q est un ensemble fini d'*états*, et $I \subseteq Q$ est le sous-ensemble des états initiaux; finalement, δ est une fonction qui à chaque symbole de fonction f d'arité k de Σ associe une relation $(k+1)$-aire $\delta(f) \subseteq Q^{k+1}$. (Intuitivement, il y a une transition de q vers le k-*uplet* $(q_1, \ldots, q_k)$ étiquetée f si et seulement si $(q, q_1, \ldots, q_k) \in \delta(f)$.)

Tout terme clos t sur la signature Σ s'écrit de façon unique $f(t_1, \ldots, t_k)$ pour un certain symbole f d'arité k de Σ, et pour certains termes clos $t_1, \ldots, t_k$ sur la signature Σ.

On dit que le terme clos $t = f(t_1, \ldots, t_k)$ est *accepté* depuis q si et seulement s'il existe un k-uplet d'états $q_1, \ldots, q_k$ tels que $(q, q_1, \ldots, q_k) \in \delta(f)$, et t_1 est accepté depuis q_1, …, t_k est accepté depuis q_k. (Noter qu'on n'a pas besoin d'états finaux: lorsque $k = 0$, $f()$ est accepté si et seulement si $(q) \in \delta(f)$.) Le *langage* $L(\mathcal{A})$ est l'ensemble des termes clos acceptés depuis un état initial.

On peut définit $L(\mathcal{A})$ aussi comme suit. Une *dérivation* de $t : q$, où $t = f(t_1, \ldots, t_k)$ est définie par récurrence structurelle sur t comme étant un arbre fini dont la racine est étiquetée par $t : q$, a k fils qui sont les racines de dérivations de $t_1 : q_1$, …, $t_k : q_k$ respectivement, où $(q, q_1, \ldots, q_k) \in \delta(f)$. $L(\mathcal{A})$ est l'ensemble des termes clos t tels qu'il existe une dérivation de $t : q$ pour au moins un état initial q.

On considère le problème suivant AA-NON-VACUITÉ.
ENTRÉE: un automate d'arbres non déterministe $\mathcal{A}$ sur Σ.
QUESTION: $L(\mathcal{A})$ est-il non vide?

1. Exhiber un algorithme en espace logarithmique alternant décidant AA-NON-VACUITÉ.

2. Montrer que AA-NON-VACUITÉ est P-complet. On se rappellera qu'un problème P-complet est HORNSAT. On se rappellera aussi qu'un ensemble de clauses de Horn S est insatisfiable si et seulement s'il existe une clause non définie $\bot \Leftarrow A_1, \ldots, A_n$ de S, et n *preuves* de A_1, resp. ..., resp. A_n. Une preuve de A est un arbre fini dont la racine est étiquetée par A, telle qu'il existe une clause définie $A \Leftarrow A'_1, \ldots, A'_k$ de S de sorte que la racine ait k fils, qui sont des preuves de A'_1, resp. ..., resp. A'_k.

IV. Intersections d'automates déterministes.

Les automates de cette partie seront de nouveau des automates de mots.

On considère maintenant le problème DFA-INTER-NON-VACUITÉ suivant:
ENTRÉE: une liste finie d'automates déterministes $\mathcal{A}_i = (\Sigma, Q_i, q_{Ii}, \delta_i, F_i)$, $1 \leq i \leq k$, de même alphabet Σ.
QUESTION: $\bigcap_{i=1}^{k} L(\mathcal{A}_i)$ est-il non vide?

La taille n de l'entrée est de l'ordre de $\sum_{i=1}^{k} |\mathcal{A}_i|$. Noter que l'entier k n'est pas borné a priori, et peut atteindre $O(n)$.

On rappelle que l'on peut construire, en temps exponentiel en n, un automate $\mathcal{A}$, de taille exponentielle en n encore, et appelé *automate produit*, tel que $L(\mathcal{A}) = \bigcap_{i=1}^{k} L(\mathcal{A}_i)$.

1. Montrer que DFA-INTER-NON-VACUITÉ est dans PSPACE. On pourra pour cela utiliser la construction d'automate produit et se ramener aux résultats de la partie I, ou bien expliciter directement un algorithme répondant à la question.

2. Dans les questions qui suivent, on va chercher à coder l'ensemble des traces d'une machine de Turing déterministe en espace polynomial sous forme de l'intersection des langages d'un nombre polynomial de langages d'automates $\mathcal{A}_i$.

   Fixons donc une machine de Turing déterministe $\mathcal{M}$, travaillant en espace $f(n)$. Supposons sans perte de généralité qu'elle n'a qu'une bande, servant à la fois de bande d'entrée, de bande de travail, et de bande de sortie; on demande $f(n) \geq n$ pour ceci. Supposons aussi que tout l'espace de la bande est *préalloué*, c'est-à-dire que la taille de la bande est exactement $f(n)$ à toute étape du calcul. Encore une fois, ceci se fait sans perte de généralité.

   On notera Σ_{tape} l'alphabet de la bande de $\mathcal{M}$, St l'alphabet des états internes de $\mathcal{M}$, $\{\text{oui}, \text{non}\} \subseteq St$ le sous-ensemble des états finaux, et $\delta_{\mathcal{M}} : \Sigma_{tape} \times (St \setminus \{\text{oui}, \text{non}\}) \to \Sigma_{tape} \times St \times \{\leftarrow, =, \rightarrow\}$ la table de transitions de $\mathcal{M}$.

   Lorsque la machine $\mathcal{M}$ est dans la configuration $\triangleright wqw'$, où $\triangleright wqw'$ est un mot de taille $f(n)+2$, $q \in St$ est l'état interne courant, et $|w|$ (la longueur de w) est la position de la tête, on définit la *vue locale* $loc(\triangleright wqw')$ comme étant le triplet $(|w|, a, q)$, où q est la lettre sous la tête, c'est-à-dire par définition la dernière lettre du préfixe $\triangleright w$.

   Une *trace* T de $\mathcal{M}$ est une suite de configurations $\triangleright wqw'$ commençant par une configuration *initiale* $\triangleright q_{init} w_{inp} \sqcup^{f(n)-n}$, où q_{init} est l'état initial de $\mathcal{M}$, w_{inp} sa donnée en entrée

(de taille n), et $\sqcup$ est le caractère blanc, et telle que deux configurations qui se suivent sont reliées par la table de transitions de $\mathcal{M}$ de la manière usuelle. Une trace est *acceptante* si et seulement si l'une de ses configurations est de la forme $\triangleright wqw'$, avec $q =$ oui.

Si T est une trace, on note $Loc(T)$ la suite des vues locales $loc(\triangleright wqw')$ lorsque $\triangleright wqw'$ parcourt T. Toute suite de vues locales ne correspond pas nécessairement à une trace valide de $\mathcal{M}$. On va chercher à caractériser les suites de vues locales de la forme $Loc(T)$.

Supposons que T est la suite des configurations locales $C_0, C_1, \ldots, C_k$, avec $C_0 = \triangleright q_{init} w_{inp} \sqcup^{f(n)-n}$. On admettra que, si l'on pose $(p_i, a_i, q_i) = loc(C_i)$ pour tout i, les conditions suivantes sont vérifiées:

(a) pour tout i, $0 \leq i \leq k$, on a $0 \leq p_i \leq f(n)$, $a_i \in \Sigma_{tape}$, et $q_i \in St$;

(b) La tête avance dans la direction prévue par chaque vue locale: pour tout $i < k$, q_i n'est pas un état final de $\mathcal{M}$, et $\delta_{\mathcal{M}}(a_i, q_i)$ est de la forme (a', q', d), avec: si d vaut $=$, alors $p_{i+1} = p_i$; si d vaut $\leftarrow$, alors $p_i \geq 1$ et $p_{i+1} = p_i - 1$; si d vaut $\rightarrow$, alors $p_i < f(n)$ et $p_{i+1} = p_i + 1$.

(c) L'état interne évolue comme prescrit par la vue locale: pour tout $i < k$, q_i n'est pas un état final de $\mathcal{M}$, et $\delta_{\mathcal{M}}(a_i, q_i)$ est de la forme (a', q', d) avec $q_{i+1} = q'$;

(d) Toute lettre lue sur la bande vaut soit celle qui était en entrée à cette même position, soit la dernière lettre qu'on a écrit à cette position le cas échéant. Autrement dit, pour tout $i \leq k$:

  i. ou bien $p_j \neq p_i$ pour tout $j < i$, et a_i est la lettre en position p_i de la configuration initiale $\triangleright q_{init} w_{inp} \sqcup^{f(n)-n}$;

  ii. ou bien il existe $j < i$ tel que $p_j = p_i$, tel que $p_\ell \neq p_i$ pour tout ℓ avec $j < \ell < i$, et $\delta_{\mathcal{M}}(a_j, q_j)$ est de la forme (a', q', d) avec $a' = a_i$.

(e) $p_0 = 0$;

(f) $q_0 = q_{init}$.

(C'est plus ou moins évident; dans le cas (d).ii., l'indice j est le $j < i$ maximal tel que $p_j = p_i$, c'est-à-dire la dernière fois où l'on a écrit un caractère à la position p_i.)

Réciproquement, soit $(p_0, a_0, q_0), (p_1, a_1, q_1), \ldots, (p_k, a_k, q_k)$ une suite de vues locales vérifiant les conditions (a)–(f). Montrer comment reconstruire une trace valide $T = C_0, C_1, \ldots, C_k$ telle que $C_0 = \triangleright q_{init} w_{inp} \sqcup^{f(n)-n}$ et $loc(C_i) = (p_i, a_i, q_i)$ pour tout i, $0 \leq i \leq k$. On supposera sans perte de généralité que $\delta_{\mathcal{M}}(\triangleright, q)$ est de la forme $(\triangleright, q', \rightarrow)$; autrement dit que la machine, quand elle lit le caractère délimiteur gauche, ne peut pas le remplacer par une autre lettre, et doit se déplacer à droite.

3. On suppose que $f(n)$ est borné par un polynôme en n. Sans perte de généralité, nous supposerons que $f(n)$ est aussi une puissance de deux (ceci demande éventuellement à rajouter des blancs inutiles en fin de bande): $f(n) = 2^{g(n)}$, où $g(n) = O(\log n)$.

   Soit Σ_{bool} l'alphabet $\{0, 1\}$, Σ l'alphabet union disjointe de Σ_{bool}, Σ_{tape}, St, et du symbole spécial $\sharp$, supposé distinct de tous les autres.

On code les suites de vues locales $(p_0, a_0, q_0), (p_1, a_1, q_1), \ldots, (p_k, a_k, q_k)$ sous forme de mots $p_0 a_0 q_0 \sharp p_1 a_1 q_1 \sharp \ldots \sharp p_k a_k q_k$ de Σ^*, où les p_i sont codés en binaire sous forme de mots de $\Sigma_{bool}^{g(n)}$, $a_i \in \Sigma_{tape}$, $q_i \in St$. En traduisant les conditions (a) à (f) des questions précédentes en conditions d'appartenance à des langages réguliers, montrer que l'on peut construire, connaissant w_{inp}, un nombre polynomial en n d'automates non déterministes (ou d'expressions régulières, selon votre préférence) $\mathcal{A}_i$ tels que les mots $m \in \Sigma^*$ (s'ils existent) reconnus par tous les $\mathcal{A}_i$ en même temps soient les codages des suites de vues locales vérifiant (a)–(f). (Indication: on pourra se demander quels sont les motifs de sous-mots qui ne doivent *pas* apparaître dans m pour que les conditions (a)–(f) soient vérifiées.)

4. Soit $\mathcal{A}$ un automate non déterministe. On rappelle que l'on peut construire, en temps $O(2^{|\mathcal{A}|})$, un automate déterministe reconnaissant le même language $L(\mathcal{A})$, et un automate déterministe reconnaissant le complémentaire $\overline{L(\mathcal{A})}$ de $L(\mathcal{A})$. On peut faire de même en partant d'une expression régulière plutôt que d'un automate non déterministe, avec les mêmes complexités.

   Montrer que l'on peut effectuer la construction de la question précédente en remplaçant les automates non déterministes $\mathcal{A}_i$ par des automates déterministes reconnaissant les mêmes langages, mais de tailles *polynomiales* en n, et fabricables en temps polynomial en n. (Indication: on vérifiera par exemple que l'on peut s'arranger pour que les automates à déterminiser ou à complémenter soient de tailles logarithmiques.)

5. En déduire que l'on peut construire, connaissant w_{inp}, et *en temps polynomial*, un nombre polynomial en n d'automates déterministes dont l'intersection des langages est non vide si et seulement si $\mathcal{M}$ accepte w_{inp}.

6. En déduire que DFA-INTER-NON-VACUITÉ est PSPACE-complet pour les réductions en temps polynomial.

7. Donner une idée de démonstration du fait que le problème analogue sur les automates d'arbres est DEXPTIME-difficile. (Le problème est DEXPTIME-complet.) Quelle sorte de machine de Turing recoderiez-vous?