Problèmes de non-vacuité d'automates

Correction.

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.

   On produit le graphe dont les sommets sont les états $q \in Q$, et tel qu'il existe un arc de q vers q' si et seulement si $\delta(q, a) = q'$ pour au moins une lettre $a \in \Sigma$. Clairement, il existe un mot tel que $\delta(q, w) = q'$ si et seulement si q' est accessible depuis q. Donc

$L(\mathcal{A})$ est non vide si et seulement si au moins un état de F est accessible depuis q_I. La réduction s'effectue en parcourant la table δ, ce qui est clairement en espace logarithmique.

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

   On sait que l'accessibilité dans les graphes orientés est dans NL, et que NL est stable par réductions en espace logarithmique.

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

   Supposons que le graphe G soit donné par sa matrice d'adjacence, et qu'il a N sommets. On se demande si q' est accessible depuis q. Créons N lettres, et définissons l'automate dont l'ensemble d'états est l'ensemble des sommets de G, dont l'état initial est q, dont l'unique état final est q', et tel que $\delta(i, a) = j$ ($1 \leq i, j \leq N$) si et seulement si l'entrée (i, j) de la matrice d'adjacence vaut 1, et la lettre a est la numéro j. L'automate est clairement déterministe, et la réduction est clairement en espace logarithmique.

   Comme l'accessibilité dans les graphes orientés est NL-complète, et que NON-VACUITÉ est dans NL, il 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?

   C'est le complémentaire de NON-VACUITÉ, et l'on sait par le théorème d'Immermann-Szelépcsenyi, que NL est clos par complémentaire. Donc VACUITÉ est dans NL.

   Maintenant, si L est un langage de NL, son complémentaire aussi. Or son complémentaire se réduit en espace logarithmique à NON-VACUITÉ par la question précédente, qui est le complémentaire de VACUITÉ. Donc L lui-même se réduit en espace logarithmique à VACUITÉ. Donc VACUITÉ est NL-complet.

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.

   Le même codage qu'à la question I.1 montre que le problème est dans NL. De plus, le cas particulier des automates déterministes est déjà NL-complet, par la question I.3, donc ND-NON-VACUITÉ aussi.

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.

   L'algorithme est l'implémentation directe de la sémantique… ou presque. On voudrait écrire l'algorithme qui met le terme en entrée dans une variable x, choisit existentiellement $q \in I$, puis répétitivement: posant $f(t_1, \ldots, t_k)$ le terme dans la variable x, choisit existentiellement $(q, q_1, \ldots, q_k) \in \delta(f)$, choisit universellement i avec $1 \leq i \leq k$, pose $x := t_i$, $q := q_i$, et boucle.

Ceci ne termine pas nécessairement. On peut donc introduire un test de bouclage: si l'on a déjà vu dans le passé du calcul la même paire (x, q), alors rejeter. On a au plus un nombre polynomial de telles paires. On maintient à la place un compteur i, initialisé à cette borne polynomiale, et qu'on décrémente à chaque tour de boucle. Si i passe à 0, on rejette. Noter que la machine accepte sur tout choix universel vide ($k = 0$).

Ce test de bouclage est justifié comme suit. Pour tout $t : q$ dérivable, il en existe une dérivation de taille minimale. Dans une telle dérivation, on ne peut clairement pas trouver de branche sur laquelle on trouve deux fois la même étiquette $t' : q'$. Le test de bouclage ci-dessus revient donc à écarter les dérivations qui ne sont pas minimales dans ce sens.

L'espace utilisé est celui pris par les variables x et i, qui sont des compteurs de valeurs au plus polynomiales, donc de tailles logarithmiques.

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.

Par la question précédente, AA-NON-VACUITÉ est dans ALOGTIME=P. On pouvait de façon équivalente coder directement la non-vacuité en créant une variable propositionnelle A_q pour chaque état q, dénotant le fait qu'il existe un terme reconnu depuis q. Pour chaque transition $(q, q_1, \ldots, q_k) \in \delta(f)$, on crée une clause de Horn $A_q \Leftarrow A_{q_1}, \ldots, A_{q_k}$. Finalement, on crée les clauses $\bot \Leftarrow A_q$ pour chaque $q \in I$.

Réciproquement, soit S un ensemble de clauses de Horn. On peut supposer sans perte de généralité que les seules clauses non définies (de tête $\bot$) sont de la forme $\bot \Leftarrow A$, avec exactement un atome A dans le corps. En effet, s'il existe une clause non définie avec aucun atome dans le corps, S est trivialement insatisfiable. S'il existe une clause $\bot \Leftarrow A_1, \ldots, A_n$ avec $n \geq 2$, on peut la remplacer par $\bot \Leftarrow A$ et $A \Leftarrow A_1, \ldots, A_n$, où A est une nouvelle variable propositionnelle.

Pour chaque variable propositionnelle A dans S, créer un nouvel état q_A. Pour chaque clause définie $A \Leftarrow A_1, \ldots, A_n$, créer un symbole n-aire f, et créer une transition $(q_A, q_{A_1}, \ldots, q_{A_n})$. Finalement, les états initiaux sont les q_A, lorsque $\bot \Leftarrow A$ parcourt l'ensemble des clauses non définies de S. Clairement, si $t : q_A$ a une dérivation, alors il existe une preuve de A, obtenue en effaçant les termes à gauche des deux points, et à utiliser à chaque nœud la clause indiquée par le symbole de fonction en tête de t. Réciproquement, si A a une preuve de S, alors l'arbre des clauses utilisées forme un terme clos t tel que $t : q_A$ a une dérivation. Donc S est insatisfiable si et seulement si le langage de l'automate d'arbres fabriqué est non vide. La réduction étant clairement en espace logarithmique, AA-NON-VACUITÉ est P-complet.

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.

   Première solution: on utilise la procédure NL de décision de non vacuité de l'automate produit, mais comme d'habitude on ne construit pas ce dernier. La non-vacuité se réduisant à l'accessibilité, on a juste besoin d'interroger la fonction de transition de l'automate produit, ce que l'on peut faire en simulant la machine construisant l'automate produit. Pour ceci, on a besoin juste d'espace logarithmique en la taille (exponentielle) des objets, ce qui est borné par un polynôme.

   Une solution plus directe est de remarquer que si $\bigcap_{i=1}^{k} L(\mathcal{A}_i)$ est non vide, alors il existe un mot de longueur au plus le nombre d'états de $\mathcal{A}$, donc au plus exponentielle, dans $\bigcap_{i=1}^{k} L(\mathcal{A}_i)$. L'idée de l'algorithme est de parcourir la construction produit sans la construire, en devinant le mot lettre à lettre, et pour chaque lettre. En clair, initialiser une variable x au k-uplet $(q_{I1}, \ldots, q_{Ik})$ des états initiaux de chaque $\mathcal{A}_i$, initialiser le compteur i à $\prod_{i=1}^{k} |Q_i|$, le produit des nombres d'états de chacun des automates. Tant que $i \neq 0$, si $x = (q_1, \ldots, q_k)$ est un k-uplet d'états finaux, accepter (on a trouvé un mot appartenant à l'intersection); sinon deviner une lettre a, poser $x := (\delta_1(q_1, a), \ldots, \delta_n(q_n, a))$ et décrémenter i. Lorsqu'on sort de la boucle (sur $i = 0$), rejeter. Ceci fournit une machine non-déterministe en espace polynomial, puisque x prends un espace polynomial, ainsi que le compteur i.

   On déduit de l'un ou l'autre argument que le problème est dans NPSPACE. Or NPSPACE=PSPACE par le théorème de Savitch.

   On pouvait aussi donner directement une machine déterministe en espace polynomial qui résout le problème, en refaisant la construction de Savitch, c'est-à-dire en cherchant un mot w de longueur au plus 2^j tel que $\delta_1(q_{I1}, w)$, ..., $\delta_k(q_{Ik}, w)$ soient tous finaux récursivement sur j, en énumérant tous les k-uplets d'états $(q_1, \ldots, q_k)$ qui sont atteints par un préfixe de w de longueur au plus 2^{j-1}, et depuis lesquels on peut atteindre un k-uplet d'états finaux via un suffixe de w de longueur au plus 2^{j-1}.

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 $q = \text{oui}$.

   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.

> *Pour tout i, $0 \leq i \leq k$, soit $\triangleright w_i$ le mot de longueur $f(n) + 1$ dont la lettre a_{ip} en position p, pour tout p, $0 \leq p \leq f(n)$, est intuitivement la dernière lettre écrite par la machine en position p. Autrement dit, si $p_j \neq p$ pour tout $j < i$, alors a_{ip} vaut la lettre en position p de $\triangleright q_{init} w_{inp} \sqcup^{f(n)-n}$, et sinon, en posant j le plus grand entier strictement plus petit que i tel que $p_j = p$, alors $a_{ip} = a'$, où $\delta_{\mathcal{M}}(a_j, q_j) = (a', q', d)$. Le mot obtenu commence bien par $\triangleright$, par l'hypothèse faite dans l'énoncé.*
>
> *On pose maintenant $C_i = \triangleright w_i' q_i w_i''$, où w_i' est le préfixe de longueur p_i de w_i, et w_i'' le reste des caractères de w_i.*
>
> *Par construction, C_0 est la configuration initiale $\triangleright q_{init} w_{inp} \sqcup^{f(n)-n}$. On montre ensuite que $T = C_0, C_1, \ldots, C_k$ est une trace valide de la machine de Turing $\mathcal{M}$. Ceci est légèrement fastidieux, et procède par construction, en observant notamment qu'au plus une lettre est modifiée entre le mot w_i et le mot w_{i+1}, que c'est celle qui est sous la tête.*

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.)

La condition (a) revient à demander que m *soit dans le language* $St\Sigma_{tape}\Sigma_{bool}^{g(n)}(\sharp St\Sigma_{tape}\Sigma_{bool}^{g(n)})^*$*, qui est bien régulier.*

Pour chaque position p*, posons* P_p *le langage de toutes les positions différentes de* p*. Il s'agit bien sûr d'un langage régulier. On peut même remarquer qu'il se décrit par un automate déterministe de taille* $O(g(n))$*: il a une suite d'états* $q_0, q_1, \ldots, q_{g(n)}$ *avec le* i*ième bit* b_i *de* p *qui étiquette la transition de* q_{i-1} *vers* q_i*, autrement dit* $\delta(q_{i-1}, b_i) = q_i$*;* $\delta(q_{g(n)}, b) = q_{g(n)}$ *pour tout booléen* b*; il y a d'autre part* $g(n)$ *états* q'_1*, ...,* $q'_{g(n)}$ *reconnaissant respectivement les mots quelconques de longueurs* $g(n) - 1, \ldots, 0$*: on a* $\delta(q_{i-1}, \neg b_i) = q'_i$*,* $\delta(q'_{i-1}, b) = q'_i$ *pour tout* b*. Finalement, l'unique état final est* $q'_{g(n)}$*. Ceci sera important dans la question suivante.*

On peut exprimer la condition (b) en énumérant tous les couples de vues locales (q, a, p)*. Pour chacune, posons* $\delta_{\mathcal{M}}(a, q) = (a', q', d)$*, et définissons* p' *par: si* $d = \leftarrow$ *alors* $p' = p - 1$*, si* d *vaut* $=$*, alors* $p' = p$*, et si* $d = \rightarrow$ *alors* $p' = p + 1$*. La condition (b) est violée si et seulement si on trouve dans* m *un sous-mot* $(q, a, p)\sharp(q'', a'', p'')$ *avec* $p'' \neq p'$*. Pour chaque* $(q, a, p) \in \Sigma_1$*, calculons* p'*, et fabriquons l'automate déterministe reconnaissant* $\overline{\Sigma^* qap\sharp St\Sigma_{tape}P_{p'}\Sigma^*}$*, Notons que, puisque* $P_{p'}$ *peut se décrire par un automate (déterministe ou non))de taille* $O(g(n))$*, il en est de même pour* $\Sigma^* qap\sharp St\Sigma_{tape}P_{p'}\Sigma^*$*.*

De même pour la condition (c), soit q' *comme ci-dessus,* $Q_{q'}$ *le langage fini donc régulier des* q'' *avec* $q'' \neq q'$*. Noter que* $Q_{q'}$ *est de taille constante (indépendante de* n*). On fabrique l'automate déterministe reconnaissant* $\overline{\Sigma^* qap\sharp Q_{q'}\Sigma_{tape}\Sigma_{bool}^{g(n)}\Sigma^*}$*. Noter qu'encore une fois, l'expression régulière sous la barre se décrit par un automate de taille* $O(g(n))$*.*

Pour la condition (d), les configurations interdites sont celles où:

- *cas i.:* m *commence par le préfixe* $(q_0, a_0, p_0)\sharp(q_1, a_1, p_1)\sharp \ldots \sharp(q_i, a_i, p_i)$ *avec* $p_j \neq p_i$ *pour tout* $j < i$*, mais* a_i *est différente de la lettre* a' *en position* p_i *de la configuration initiale* $\triangleright q_{init} w_{inp} \sqcup^{f(n)-n}$*. Pour chaque* $(q, a, p) \in \Sigma_1$ *tel que* a *est différente de la lettre en position* p *de la configuration initiale (qu'on énumère, et tient le rôle de* (q_i, a_i, p_i)*), soit* P_p *le langage des* (q'', a'', p'') *avec* $p'' \neq p$*, comme ci-dessus, on veut exprimer que* m *ne commence pas par un préfixe dans* $\overline{(St\Sigma_{tape}P_p\sharp)^*(q, a, p)}$*. On fabrique donc l'automate déterministe reconnaissant* $\overline{(St\Sigma_{tape}P_p\sharp)^*(q, a, p)\Sigma^*}$ *pour chacun de ces* (q, a, p)*. Encore une fois, c'est le complémentaire d'un automate non déterministe de taille* $O(g(n))$*.*
- *cas ii.:* m *contient un sous-mot* $(q_j, a_j, p_j)\sharp(q_{j+1}, a_{j+1}, p_{j+1}\sharp \ldots \sharp(p_i, a_i, p_i)$*, où* $p_j = p_i$*,* $p_\ell \neq p_i$ *pour tout* ℓ*,* $j < \ell < i$*, mais si l'on pose* $(a', q', d) = \delta_{\mathcal{M}}(a_j, q_j)$*, on a* $a' \neq a_i$*. On énumère tous les* $(q, a, p) \in \Sigma_1$*; pour chacun, on calcule* a' *comme la première composante de* $\delta_{\mathcal{M}}(a, q)$*, les configurations interdites sont alors celles dans le langage régulier* $qap(\sharp St\Sigma_{tape}P_p)^*\sharp\Sigma_{tape}L_{a'}p$*, où* $L_{a'}$ *est le langage* $\Sigma_{tape} \setminus \{a'\}$*.* $L_{a'}$ *est un langage fini, donc régulier, décrit par un automate de taille constante en* n*. Encore une fois,* $qap(\sharp St\Sigma_{tape}P_p)^*\sharp\Sigma_{tape}L_{a'}p$ *est de taille* $O(g(n))$*. On fabrique l'automate déterministe reconnaissant*

$\overline{qap(\sharp St\Sigma_{tape}P_p)^*\sharp\Sigma_{tape}L_{a'}p}$ *pour chaque* (q,a,p)*, avec* a' *décrit comme ci-dessus.*

Pour (e) et (f) (en remarquant de plus que $a_0 = \triangleright$ *par (d).i.), on fabrique l'automate déterministe reconnaissant* $q_{init} \triangleright 0\sharp\Sigma^*$*, qui est clairement de taille* $O(g(n))$*.*

Le nombre d'automates à fabriquer est borné par une constante multipliée par le nombre de configurations (q,a,p) *possibles, qui est en* $O(f(n))$*. Comme* $f(n)$ *est polynomial en* n*, on fabrique un nombre polynomial d'automates.*

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.)

   On note que tous les automates produits sont soit directement de taille $O(g(n))$*, ce qui est logarithmique, donc polynomial en* n*, soit des complémentaires d'automates de taille* $O(g(n))$*. La déterminisation et la complémentation prenant un temps exponentiel, et fabriquant des automates de tailles exponentielles, la déterminisation ou la complémentation de tels automates s'effectue en temps* $O(2^{g(n)}) = O(f(n))$*, ce qui est polynomial en* n*.*

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}.

   Construisons en temps polynomial (question précédente) les automates $\mathcal{A}_i$*. Ajoutons-y l'automate déterministe reconnaissant le langage* $(St\Sigma_{tape}\Sigma_{bool}^{g(n)}\sharp)^*$*oui* $\Sigma_{tape}\Sigma_{bool}^{g(n)}$*, qui est de taille* $O(f(n))$ *par le même argument qu'à la question précédente.*

   Si $\mathcal{M}$ *accepte* w_{inp}*, alors il existe une trace* $T = C_0, C_1, \ldots, C_k$ *de* $\mathcal{M}$*, avec* $C_0 = \triangleright q_{init} w_{inp} \sqcup^{f(n)-n}$*, et telle que l'état interne en configuration* C_k *soit oui. Par l'énoncé de la question IV.2,* $Loc(T)$ *est une suite de vues locales vérifiant (a)–(f) et telle que* $q_k =$ *oui. Par le codage des questions précédentes, ceci se traduit en un mot reconnu par tous les automates mentionnés ci-dessus.*

   Réciproquement, si m *est un mot reconnu par tous les automates ci-dessus, il est le codage d'une suite de vues locales* (p_i, a_i, q_i)*,* $0 \leq i \leq k$*, vérifiant (a)–(f) et telles que* $q_k =$ *oui. Par la question IV.2, il existe donc une trace valide* $T = C_0, C_1, \ldots, C_k$ *de* $\mathcal{M}$ *telle que* $C_0 = \triangleright q_{init} w_{inp} \sqcup^{f(n)-n}$*. De plus, comme* $q_k =$ *oui,* T *est acceptante. Donc* $\mathcal{M}$ *accepte* w_{inp}*.*

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

   Soit L un langage quelconque de PSPACE, et $\mathcal{M}$ une machine de Turing déterministe reconnaissant L en temps $f(n)$, où $f(n) = 2^{g(n)}$ est borné par un polynôme en n. Pour chaque w_{inp} de taille n, on produit les automates $\mathcal{A}_i$ de la question précédente. Leur intersection est non vide si et seulement si $w_{inp} \in L$ par construction. On a donc bien défini ainsi une réduction en temps polynomial de M vers DFA-INTER-NON-VACUITÉ, ce qui montre que ce problème est PSPACE-difficile pour les réductions en temps polynomial. La PSPACE-complétude s'obtient en utilisant la question IV.1.

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?

   La partie III devrait donner quelques idées. On peut coder les traces d'une machine de Turing alternante en espace polynomial comme des termes du premier ordre, les symboles k-aires, $k \geq 2$, codant les transitions universelles. Les transitions existentielles sont codées comme avant comme une quantification existentielle sur les traces. (Une adaptation simple de la construction des questions précédentes montre comment recoder les machines en espace polynomial non *déterministes.) Les détails sont laissés au lecteur... Pour finir, on utilise le fait que APSPACE=DEXPTIME.*