INF105
Contrôle de connaissances — Corrigé

Théorie des langages

7 février 2017

Consignes.

Les exercices sont totalement indépendants. Ils pourront être traités dans un ordre quelconque, mais on demande de faire apparaître de façon très visible dans les copies où commence chaque exercice.

Le sujet étant long pour le temps imparti, il ne sera pas nécessaire de traiter toutes les questions pour obtenir la totalité des points.

L'usage de tous les documents (notes de cours manuscrites ou imprimées, feuilles d'exercices, livres) est autorisé.

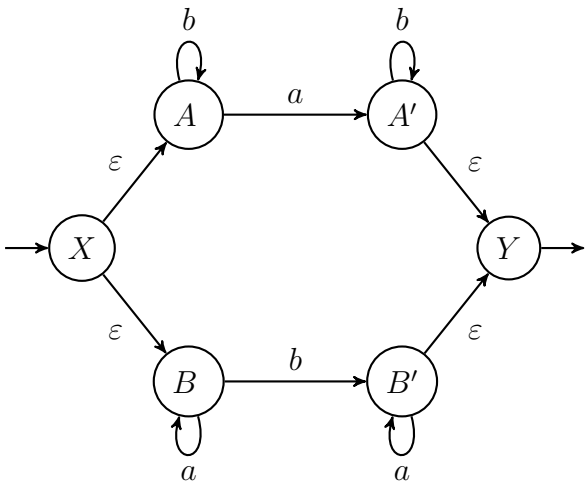
L'usage des appareils électroniques est interdit.

Durée : 1h30

Barème *indicatif* : 8 points par exercices

Exercice 1.

On considère l'automate fini M sur l'alphabet $\Sigma = \{a, b\}$ représenté par la figure suivante :



(0) De quelle sorte d'automate s'agit-il ? (Autrement dit : est-il déterministe ou non ? avec transitions spontanées ou non ?)

(1a) Décrire brièvement, en français, le langage L reconnu (=accepté) par l'automate M, puis donner une expression rationnelle qui le dénote. (On pourra préférer traiter la question (1b) d'abord.)

(1b) Pour chacun des mots suivants, dire s'ils sont dans L ou non : ε, a, b, ab, aa, aab, $aabb$, $abab$, $ababa$. (Note : il est recommandé de réutiliser ces mots pour vérifier rapidement les réponses aux questions suivantes et ainsi détecter d'éventuelles erreurs lors des transformations des automates.)

(2) Éliminer les transitions spontanées de l'automate M. (On supprimera les états devenus inutiles.) On appellera M_2 l'automate obtenu.

(3) Déterminiser l'automate M_2 obtenu en (2), si nécessaire. (On demande un automate déterministe complet.) On appellera M_3 l'automate déterminisé.

Pour simplifier le travail du correcteur, on demande de représenter M_3 de sorte que les transitions étiquetées par a soient, dans la mesure du possible, horizontales de la gauche vers la droite, et celles étiquetées par b, verticales du haut vers le bas.

(4) Minimiser l'automate M_3 obtenu en (3), si nécessaire (justifier).

(5) Donner un automate (de n'importe quelle sorte) qui reconnaît le langage $\overline{L} = \Sigma^* \setminus L$ complémentaire de L.

(6) Décrire brièvement, en français, ce langage complémentaire $\overline{L}$.

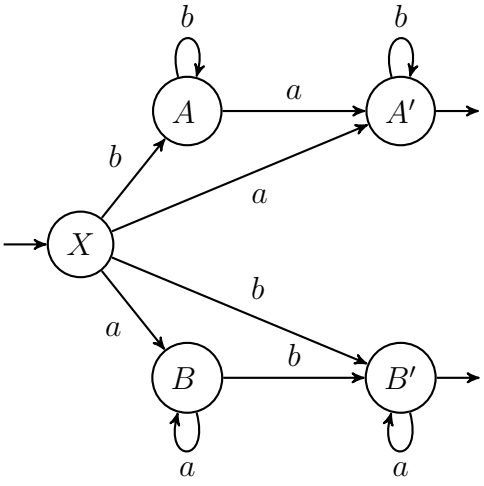
(7) (Question bonus, plus longue, à ne traiter qu'en dernier.) Calculer une expression rationnelle qui dénote ce langage complémentaire $\overline{L}$. (Ne pas hésiter à introduire des notations intermédiaires.)

Corrigé. (0) L'automate M est un automate fini non-déterministe à transitions spontanées, ou ε-NFA (le concept d'« automate déterministe à transitions spontanées » n'aurait tout simplement pas de sens).

(1a) Le chemin par les états X, A, A', Y accepte les mots exactement un a, c'est-à-dire le langage dénoté par $b*ab*$. Le chemin par les états X, B, B', Y accepte les mots comportant exactement un b, c'est-à-dire le langage dénoté par $a*ba*$. L'automate M dans son ensemble accepte les mots comportant exactement un a ou (inclusif) exactement un b (i.e. $L = \{w \in \Sigma^* : |w|_a = 1 \text{ ou } |w|_b = 1\}$ si $|w|_x$ désigne le nombre total d'occurrences de la lettre x dans le mot w). C'est le langage dénoté par l'expression rationnelle $b*ab*|a*ba*$ (nous notons ici et ailleurs | pour la disjonction, qu'on peut aussi noter $+$).

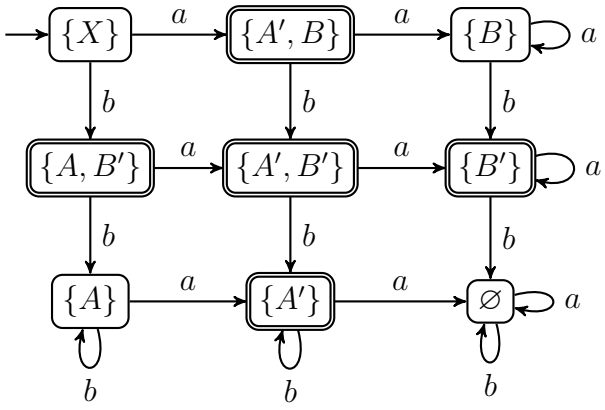
(1b) Parmi les mots proposés, a, b, ab et aab appartiennent à L, tandis que ε, aa, $aabb$, $abab$ et $ababa$ n'y appartiennent pas.

(2) La ε-fermeture (arrière) de l'état X est $\{X, A, B\}$; la ε-fermeture de l'état A' est $\{A', Y\}$ et celle de l'état B' est $\{B', Y\}$; les autres états sont leur propre ε-fermeture (i.e., celle-ci est un singleton). L'élimination des transitions spontanées conduit donc à l'automate M_2 suivant :

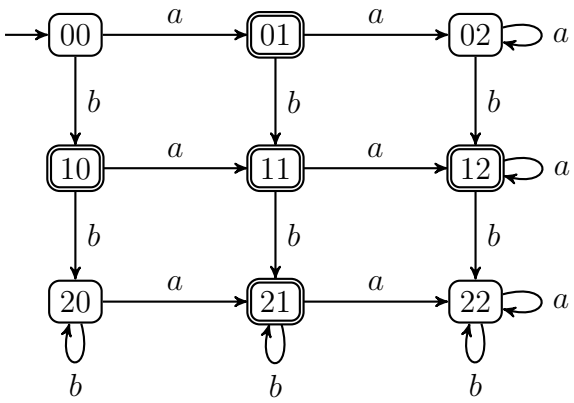


(On a supprimé l'état Y qui est devenu inutile car aucune transition non-spontanée n'y conduit.)

(3) L'algorithme de déterminisation conduit à l'automate M_3 suivant où, pour plus de lisibilité, les états finaux ont été marqués en les entourant deux fois plutôt que par une flèche sortante :



Pour la commodité de la suite de la correction, on renomme les états de cet automate M_3 de la façon suivante :



Ici, l'état $0\bullet$ signifie que l'automate n'a pas rencontré de a, l'état $1\bullet$ qu'il en a rencontré exactement un, et l'état $2\bullet$ qu'il en a rencontré au moins deux ; les états $\bullet 0$, $\bullet 1$ et $\bullet 2$ ont la même signification pour la lettre b.

(4) L'automate M_3 est déjà minimal. En effet, l'algorithme de minimisation commence par séparer les classes $\{01, 10, 11, 12, 21\}$ (états finaux) et $\{00, 02, 20, 22\}$ (états non-finaux) ; ensuite, la transition étiquetée par a sépare la classe $\{01, 10, 11, 12, 21\}$ en $\{01, 21\}$ (qui vont vers un état non-final) et $\{10, 11, 12\}$ (qui vont vers un état final), et la classe $\{00, 02, 20, 22\}$ en $\{00, 20\}$ (qui vont vers un état final) et $\{02, 22\}$ (qui vont vers un non-final). La transition étiquetée par b sépare ensuite en deux chacune des trois classes $\{00, 20\}$, $\{01, 21\}$ et $\{02, 22\}$ (car le premier élément va dans la classe $\{10, 11, 12\}$ tandis que le second reste dans la même classe) et sépare en trois la classe $\{10, 11, 12\}$. On a donc séparé chacun des états.

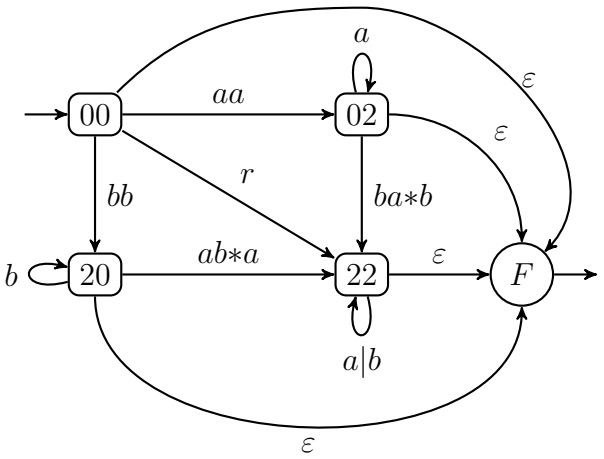
(5) Pour reconnaître le complémentaire du langage reconnu par un automate fini déterministe complet, il suffit d'échanger états finaux et non-finaux : on peut

donc prendre l'automate dessiné en (3) avec, cette fois, la convention que les états simplement entourés sont finaux (et les doublement entourés sont non-finaux). Appelons-le M_5.

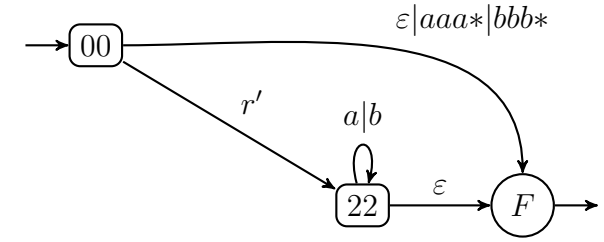
Attention : échanger états finaux et non-finaux ne marche pas pour reconnaître le complémentaire du langage reconnu par un automate non-déterministe ou incomplet (car la négation de « il existe un chemin qui va vers un état final » est « aucun chemin ne va vers un état final » et pas « il existe un chemin qui va vers un état non-final »).

(6) Puisque L est le langage formé des mots comportant exactement un a ou (inclusif) exactement un b, son complémentaire $\overline{L}$ est formé des mots ayant un nombre différent de 1 de a *et* un nombre différent de 1 de b ; si on préfère, il s'agit du langage comportant (0 ou au moins 2 fois la lettre a) *et* (0 ou au moins 2 fois la lettre b).

(7) L'élimination des états n'est pas trop complexe car l'automate M_5 a très peu de boucles. Éliminons simultanément tous les états non-finaux (01, 10, 11, 12 et 21), et profitons-en pour créer un nouvel (et unique) état final F :



où $r := abaa{*}b \,|\, abbb{*}a \,|\, baaa{*}b \,|\, babb{*}a$ (correspondant aux quatre façons de passer de 00 à 22 dans le graphe ci-dessus). Éliminons l'état 02 et l'état 20 :



où $r' := r \mid aaa*ba*b \mid bbb*ab*a = abaa*b \mid abbb*a \mid baaa*b \mid babb*a \mid aaa*ba*b \mid bbb*ab*a$. On obtient finalement l'expression rationnelle suivante pour $\overline{L}$:

$$\varepsilon \mid aaa* \mid bbb* \mid (abaa*b \mid abbb*a \mid baaa*b \mid babb*a \mid aaa*ba*b \mid bbb*ab*a)(a|b)*$$

Pour comprendre cette expression rationnelle, la disjonction de plus haut niveau correspond aux quatre possibilités : (i) 0 fois la lettre a et 0 fois la lettre b, (ii) au moins 2 fois la lettre a et 0 fois la lettre b, (iii) 0 fois la lettre a et au moins 2 fois la lettre b, et (iv) au moins 2 fois la lettre a et au moins 2 fois la lettre b. Pour mieux comprendre l'expression du cas (iv), on peut remarquer que $abaa*b \mid baaa*b \mid aaa*ba*b$ dénote le langage formé des mots comportant au moins deux a et exactement deux b et qui finissent par un b, et symétriquement $abbb*a \mid babb*a \mid bbb*ab*a$ dénote le langage formé des mots comportant au moins deux b et exactement deux a et qui finissent par un a : l'expression du cas (iv) correspond donc à écrire un mot ayant au moins deux a et au moins deux b comme le premier préfixe qui vérifie cette propriété suivi d'un suffixe quelconque. (On pouvait utiliser directement ce raisonnement pour produire l'expression.) ✓

Commentaires. Cet exercice a été noté sur 9 (dans une note finale sur 20).

(0) De nombreuses copies parlent d'« automate déterministe à transitions spontanées ». Cette notion n'existe tout simplement pas et n'a pas de sens.

(1b) De nombreuses erreurs dans la manipulation des automates, qui auraient pu être attrapées par l'utilisation ce ces mots-tests ne l'ont pas été, et c'est dommage (ces erreurs ont été comptées plus sévèrement).

(3) La consigne sur la présentation de l'automate a été presque universellement ignorée (pourquoi ?). Plus grave, certains ont oublié de fournir un automate déterministe *complet* (pourtant, l'algorithme de déterminisation fournit assez naturellement un automate complet).

(5) De nombreuses copies essaient de construire un automate reconnaissant $\overline{L}$ autrement qu'en échangeant états finaux et non-finaux dans un automate déterministe complet.

(6) Beaucoup croient à tort que la négation de « contenir unique a ou bien contenir un unique b » est « être le mot vide ou bien contenir au moins deux a ou bien au moins deux b » : le mot aab de la question (1b) aurait dû permettre d'éviter cette erreur. À l'inverse, certains croient que cette négation est « être le mot vide ou bien contenir au moins deux a *et* au moins deux b » : le mot aa aurait dû permettre d'éviter cette erreur. Enfin, beaucoup de copies utilisent une description tellement vague qu'il est impossible de savoir ce qu'elle veut dire (par exemple « les mots pour lesquels le nombre d'occurrence des lettres qui s'y trouvent est supérieur ou égal à 2 »). ✠

Exercice 2.

Soit Σ un alphabet (fini, non vide) fixé. Les questions suivantes sont indépendantes (mais on remarquera leur parallélisme). Ne pas hésiter à décrire les algorithmes de façon succincte et informelle.

(1) Expliquer, au moyen des résultats vus en cours, pourquoi il existe un algorithme A_1 qui, étant donnée une expression rationnelle r sur Σ, décide si le langage L_r dénoté par r est différent du langage Σ^* de tous les mots sur Σ. (Autrement dit, l'algorithme A_1 doit prendre en entrée une expression rationnelle r, terminer en temps fini, et répondre « vrai » s'il existe un mot $w \in \Sigma^*$ tel que $w \notin L_r$ et « faux » si $L_r = \Sigma^*$. On ne demande pas que l'algorithme soit efficace.)

(2) Expliquer pourquoi il existe un algorithme A_2 qui, étant donnée une grammaire hors contexte G sur Σ, « semi-décide » si le langage L_G engendré par G est différent du langage Σ^* de tous les mots. (« Semi-décider » signifie que l'algorithme A_2 doit prendre en entrée une grammaire hors contexte G, terminer en temps fini en répondant « vrai » s'il existe un mot $w \in \Sigma^*$ tel que $w \notin L_G$, et ne pas terminer [1] si $L_G = \Sigma^*$.) Indication : on peut tester tous les mots possibles.

(3) Expliquer pourquoi il existe un algorithme A_3 comme suit : on lui fournit en entrée un algorithme T qui décide un langage $L_T \subseteq \Sigma^*$ (c'est-à-dire que T termine toujours en temps fini quand on lui présente un mot sur Σ, et répond « vrai » ou « faux », et L_T est le langage des mots sur lesquels il répond « vrai »), et l'algorithme A_3 doit semi-décider si L_T est différent de Σ^*. (C'est-à-dire que A_3 doit terminer en répondant « vrai » s'il existe un mot $w \in \Sigma^*$ tel que $w \notin L_T$, et ne pas terminer si $L_T = \Sigma^*$.) Indication : la même approche permet de traiter les questions (2) et (3).

(4) Expliquer pourquoi il *n'existe pas* d'algorithme A_4 qui, dans les mêmes conditions que A_3, décide (au lieu de seulement semi-décider) si L_T est différent de Σ^*. (C'est-à-dire que A_4 est censé terminer toujours, et répondre « vrai » s'il existe un mot $w \in \Sigma^*$ tel que $w \notin L_T$, et « faux » si $L_T = \Sigma^*$.) Indication : expliquer comment on pourrait utiliser un tel A_4 pour résoudre le problème de l'arrêt, en cherchant à fabriquer un T qui rejette un mot précisément si un programme donné s'arrête.

Corrigé. (1) Donnée une expression rationnelle r, on sait qu'on peut algorithmiquement fabriquer un automate fini non-déterministe à transitions spontanées qui reconnaît exactement le langage L_r dénoté par r, et ensuite éliminer les transitions spontanées et déterminiser l'automate pour obtenir un automate fini déterministe complet reconnaissant L_r. Sur un tel automate, savoir si $L_r \neq \Sigma^*$ est trivial : dès lors qu'il existe un état q non-final accessible, il existe un mot rejeté par l'automate (i.e., n'appartenant pas à L_r), à savoir le mot lu en suivant les étiquettes d'un chemin quelconque de l'état initial q_0 jusqu'à q, et inversement, si tous les états sont finaux, il est trivial que l'automate accepte

1. On peut admettre qu'il termine parfois en répondant « faux », mais ce ne sera pas utile.

tous les mots. (On pouvait aussi minimiser l'automate et le comparer à l'automate minimal trivial qui reconnaît le langage Σ^*.)

(2) On sait qu'il existe un algorithme qui, donnée une grammaire hors-contexte G et un mot w, décide si $w \in L_G$. Pour semi-décider s'il existe un w tel que $w \notin L_G$, il suffit de tester tous les mots possibles : plus exactement, on construit un algorithme A_2 qui effectue une boucle infinie sur tous les $w \in \Sigma^*$ (il est évidemment algorithmiquement faisable d'énumérer tous les mots sur Σ) et, pour chacun, teste si $w \in L_G$, et si ce n'est pas le cas, termine immédiatement en répondant « vrai » (on a trouvé un w n'appartenant pas à L_G), tandis que si c'est le cas, l'algorithme A_2 ne terminera jamais.

(3) On procède exactement comme en (2) : par hypothèse on dispose d'un algorithme T qui, donné un mot w, décide si $w \in L_T$. Pour semi-décider s'il existe un w tel que $w \notin L_T$, il suffit de tester tous les mots possibles : plus exactement, on construit un algorithme A_3 qui effectue une boucle infinie sur tous les $w \in \Sigma^*$ et, pour chacun, teste si $w \in L_T$ (en lançant l'algorithme T qui, par hypothèse, termine toujours), et si ce n'est pas le cas, termine immédiatement en répondant « vrai » (on a trouvé un w n'appartenant pas à L_T), tandis que si c'est le cas, l'algorithme A_3 ne terminera jamais.

(4) Supposons par l'absurde qu'on dispose d'un algorithme A_4 comme on vient de dire, et montrons pour arriver à une contradiction qu'on peut s'en servir pour résoudre le problème de l'arrêt. On se donne donc un algorithme S et une entrée x de S et on cherche à savoir (en utilisant A_4) si S termine sur l'entrée x. Pour cela, on va construire un T auquel appliquer A_4.

Voici une solution possible : donné un mot $w \in \Sigma^*$, le programme T ne considère que la longueur $|w|$ de w, et lance (=simule) l'exécution de S sur l'entrée x pour au plus $|w|$ étapes : si l'exécution termine dans le temps imparti, alors T rejette le mot w, sinon, il l'accepte (dans tous les cas, T termine et répond « vrai » ou « faux », donc il est une entrée légitime à A_4). Cette construction fait que L_T rejette au moins un mot précisément lorsque S termine sur x : si au contraire S ne termine pas sur x, alors $L_T = \Sigma^*$. L'utilisation de A_4 sur T permet donc de savoir algorithmiquement si S termine sur x, ce qui contredit l'indécidabilité du problème de l'arrêt.

Variante de la même idée : on appelle « trace d'exécution » de S sur x un mot w qui code le calcul complet de l'exécution de S sur x (par exemple, si on voit S comme une machine de Turing, l'état courant et le contenu du ruban à chaque étape), du début à l'arrêt. Une telle trace d'exécution existe donc précisément si S termine sur x. Or il est visiblement décidable de savoir si un mot w donné est une trace d'exécution (il suffit de vérifier qu'à chaque étape la machine a bien fait ce qu'elle devait faire). On peut donc écrire un algorithme T qui termine toujours et accepte précisément les mots qui *ne sont pas* une trace d'exécution de

S sur x. Le fait que L_T soit différent de Σ^* signifie alors exactement qu'une trace d'exécution existe, donc que S termine sur x. Ainsi l'utilisation de A_4 permet de savoir algorithmiquement si S termine sur x, ce qui contredit l'indécidabilité du problème de l'arrêt. ✓

Commentaires. Cet exercice a été noté sur 7 (dans une note finale sur 20).

Il a été très peu traité, ce qui est dommage car la question (1) consistait essentiellement à dire « oui : on a vu dans le cours tous les algorithmes nécessaires » et rapportait 2 points à elle seule. ✠

Exercice 3.

On considère la grammaire hors-contexte G d'axiome S et de nonterminaux $N = \{S, T\}$ sur l'alphabet $\Sigma = \{a, b, c\}$ donnée par

$$S \to TS \mid \varepsilon$$
$$T \to aSbSc$$

On notera $L(S) = L_G$ le langage qu'elle engendre, et $L(T)$ le langage des mots qui dérivent de T (c'est-à-dire, si on préfère, le langage engendré par la grammaire identique à G mais ayant T pour axiome).

(0) Donner quelques exemples de mots de $L(S)$ et de $L(T)$ (au moins deux de chaque).

(1) Expliquer brièvement pourquoi $L(S)$ est l'ensemble des mots de la forme $u_1 \cdots u_k$ avec $u_i \in L(T)$, et pourquoi $L(T)$ est l'ensemble des mots de la forme $awbw'c$ avec $w, w' \in L(S)$. (Par conséquent, $L(T)$ est l'ensemble des mots de la forme $au_1 \cdots u_k b u'_1 \cdots u'_\ell c$ avec $u_1, \ldots, u_k, u'_1, \ldots, u'_\ell \in L(T)$.)

(2) Comment exprimer $L(S)$ à partir de $L(T)$ au moyen d'une ou plusieurs opérations rationnelles [2] ? Y a-t-il inclusion de l'un dans l'autre ?

(3) Montrer que tout mot w appartenant à $L(S)$ ou à $L(T)$ a le même nombre de a, de b et de c, c'est-à-dire $|w|_a = |w|_b = |w|_c$ où $|w|_x$ désigne le nombre total d'occurrences de la lettre x dans le mot w.

(4) Montrer par récurrence sur la longueur $|u|$ d'un mot $u \in L(T)$ que si v est un préfixe de u de longueur $0 < |v| < |u|$, alors on a $|v|_c < |v|_a$. Pour cela, on pourra écrire u sous la forme $au_1 \cdots u_k b u'_1 \cdots u'_\ell c$ obtenue en (1), considérer un préfixe [3] d'une telle expression, et appliquer la question (3) et l'hypothèse de récurrence.

(5) Déduire des questions (3) et (4) que si $u \in L(T)$ et si v est un préfixe de u autre que u lui-même, alors [4] $v \notin L(T)$. En déduire que $u \in L(T)$ et $z \in \Sigma^*$,

2. C'est-à-dire : union, concaténation, étoile de Kleene.

3. On signale à toutes fins utiles le fait évident suivant : un préfixe non vide de $t_1 \cdots t_n$ où $t_1, \ldots, t_n \in \Sigma^*$ s'écrit sous la forme $t_1 \cdots t_{i-1} y$ où $y \neq \varepsilon$ est un préfixe de t_i.

4. L'énoncé d'origine comportait par erreur la question $u \notin L(T)$ ici.

alors u est l'*unique* préfixe du mot $w := uz$ qui appartienne à $L(T)$ (autrement dit, aucun préfixe de w de longueur $<|u|$ ni $>|u|$ n'appartient à $L(T)$).

(6) En déduire que si un mot w s'écrit $w = u_1 \cdots u_k$ avec $u_1, \ldots, u_k \in L(T)$ alors cette factorisation est unique. (Comment peut-on caractériser u_1 comme préfixe de w ?) Montrer de même la conclusion analogue pour $u_1 \cdots u_k b u'_1 \cdots u'_\ell$ (on pourra noter que $bz \notin L(T)$ quel que soit $z \in \Sigma^*$).

(7) En déduire que la grammaire G est inambiguë.

Corrigé. (0) Les mots abc et $aabcbc$ et $ababcc$ appartiennent à $L(T)$. Les mots ε et abc et $abcabc$ appartiennent à $L(S)$.

(1) En bref : il s'agit simplement d'une reformulation des règles de la grammaire. En plus détaillé : si on considère un arbre d'analyse d'un mot w de $L(S)$, soit il est la dérivation triviale du mot vide, soit sa racine a deux fils étiquetés T et S, l'un dont descend un arbre d'analyse d'un mot u_1 de $L(T)$ et l'autre dont descend un arbre d'analyse d'un autre mot w' de $L(S)$, avec $w = u_1 w'$: en répétant cette remarque jusqu'à tomber sur le mot vide, on voit que $w = u_1 u_2 \cdots u_k$ pour des mots $u_i \in L(T)$; et réciproquement, tout mot de cette forme a un arbre d'analyse dans G obtenu en mettant ensemble les arbres d'analyse des u_i (en les associant deux par deux par la droite). Le cas d'un mot u de $L(T)$ est plus simple : son arbre d'analyse donne directement une écriture sous la forme $awbw'c$ avec w, w' les mots analysés par les arbres qui descendent des deux fils étiquetés S de la racine (étiquetée T).

(2) La description faite en (1) signifie notamment que $L(S) = L(T)^*$ où « $*$ » est l'étoile de Kleene. (Si on veut, la règle $S \to TS \mid \varepsilon$ peut s'interpréter comme $S \to T*$.) En particulier, on a $L(T) \subseteq L(S)$.

(3) La propriété $|\gamma|_a = |\gamma|_b = |\gamma|_c$ (ici $\gamma \in (\Sigma \cup N)^*$) est vérifiée pour $\gamma = S$ et elle est préservée par toute dérivation immédiate pour G puisqu'elle est préservée en remplaçant S par TS ou par le mot vide et en remplaçant T par $aSbSc$. Elle est donc vérifiée pour tout mot de $L(S)$ (et en particulier, pour tout mot de $L(T)$), ce qu'il fallait démontrer.

Autre possibilité : il suffit de montrer $|w|_a = |w|_b = |w|_c$ pour tout mot $w \in L(S)$ (puisque $L(T) \subseteq L(S)$) : on va procéder par récurrence sur la longueur $|w|$. D'après (1), on peut écrire $w = u_1 \cdots u_k$ avec $u_i \in L(T)$: si $k > 1$, alors chaque u_i est de longueur strictement plus petite, donc l'hypothèse de récurrence assure que la propriété est vraie pour eux, donc elle l'est pour w car $|w|_x = \sum_{i=1}^k |u_i|_x$ pour chaque $x \in \{a, b, c\}$. Reste le cas $k = 1$, c'est-à-dire, $w \in L(T)$: mais on a alors (toujours d'après (1)) $w = avbv'c$ avec $v, v' \in L(S)$ qui sont de longueur strictement plus petite que w, donc l'hypothèse de récurrence assure que la propriétée st vraie pour v et v', et comme $|w|_x = 1 + |v|_x + |v'|_x$ pour chaque $x \in \{a, b, c\}$, on a la propriété voulue pour w.

(4) On procède par récurrence sur $|u|$, ce qui permet de supposer la propriété

déjà connue pour tout préfixe v d'un mot de longueur strictement plus courte que u. Si v est un préfixe d'un mot $u \in L(T)$, qu'on peut écrire sous la forme $u = au_1 \cdots u_k b u'_1 \cdots u'_\ell c$, et si $0 < |v| < |u|$, on a soit $v = au_1 \cdots u_{i-1} y$ où $y \neq \varepsilon$ est un préfixe de u_i, soit $v = au_1 \cdots u_k b u'_1 \cdots u'_{i-1} y$ où $y \neq \varepsilon$ est un préfixe de u'_i. Dans tous les cas, tous les facteurs t qui interviennent dans cette écriture vérifient $|t|_c \leq |t|_a$: c'est trivial pour a et b, c'est le cas pour chaque u_j par la question (3), et pour y cela découle soit de l'hypothèse de récurrence (lorsque $|y| < |u_i|$ resp. $|y| < |u'_i|$) soit par la question (3) (lorsque y est en fait égal à u_i resp. u'_i). Comme par ailleurs $|a|_c = 0 < 1 = |a|_a$, une égalité est stricte et on en déduit $|v|_c < |v|_a$, ce qui conclut la récurrence.

(5) Si $u \in L(T)$ et si v est un préfixe de u autre que u lui-même, alors soit $v = \varepsilon$, qui n'appartient pas à $L(T)$ (par exemple par la question (1)), soit $0 < |v| < |u|$, auquel cas la question (4) donne $|v|_c < |v|_a$, ce qui interdit $v \in L(T)$ d'après (3). Dans tous les cas, $v \notin L(T)$.

Si maintenant $u \in L(T)$ et $z \in \Sigma^*$, un préfixe de $w := uz$ est soit un préfixe de u soit de la forme uz' avec z' un préfixe non vide de z (cf. la note 3). Un préfixe de w de longueur $<|u|$ n'appartient pas à $L(T)$ d'après le paragraphe précédent, et un mot de la forme uz' ne peut pas y appartenir non plus car c'est alors u lui-même qui serait un préfixe strict de $uz' \in L(T)$ appartenant à $L(T)$, ce qui contredit de nouveau le paragraphe précédent.

(6) D'après la question précédente, on peut définir u_1 comme l'unique préfixe de w qui appartient à $L(T)$ (tout préfixe strictement plus court ou strictement plus long n'appartient pas à $L(T)$). Une fois que u_1 est défini, en appliquant le même raisonnement au suffixe correspondant $u_2 \cdots u_k$, on voit que u_2 est défini de façon unique. En procédant ainsi (par récurrence sur k si on veut), les u_i sont définis de façon unique.

Le même raisonnement vaut pour $u_1 \cdots u_k b u'_1 \cdots u'_\ell$: on a une factorisation en mots de $L(T) \cup \{b\}$ et à chaque fois le premier facteur est défini comme le seul préfixe possible appartenant à $L(T) \cup \{b\}$. (On utilise le fait que $bz \notin L(T)$, qui découle du (1), pour voir que $bu'_1 \cdots u'_\ell$ n'a pas de préfixe dans $L(T)$.)

(7) En reprenant l'analyse du (1), il s'agit de montrer que la factorisation $u_1 \cdots u_k$ d'un mot de $L(S)$ est unique et que la factorisation $au_1 \cdots u_k b u'_1 \cdots u'_\ell c$ d'un mot de $L(T)$ l'est aussi. La première partie est exactement une conclusion du (6), et la seconde l'est aussi dès lors qu'on retire le a initial et le c final. ✓

Commentaires. Cet exercice a été noté sur 9 (dans une note finale sur 20).

(3) Les deux démonstrations proposées dans le corrigé ci-dessus ont effectivement été trouvées (rédigées de façon généralement incorrecte, mais on n'a pas sanctionné trop sévèrement les imprécisions).

Un nombre étonnant de copies commettent l'erreur de croire que pour montrer quelque chose pour tout w de $L(S)$ ou bien $L(T)$, lorsqu'on a $L(T) \subseteq L(S)$, il

suffit de le montrer pour $w \in L(T)$: c'est au contraire $w \in L(S)$ qu'on peut supposer. (Il est vrai que comme $L(S) = L(T)^*$, il n'est pas non plus difficile de passer de $L(T)$ à $L(S)$, mais il faut au moins dire un mot.)

(4) Malgré la note 3 en bas de page, la plupart des copies pensent qu'un préfixe de $au_1 \cdots u_k b u'_1 \cdots u' \ell c$ est de la forme $au_1 \cdots u_i$ ou bien $au_1 \cdots u_k b u'_1 \cdots u'_i$, et oublient que le dernier facteur peut lui-même être coupé (c'était bien ça le sens de la note). Cette erreur n'a pas été sanctionnée trop sévèrement.

(5) Cette question comportait deux sous-questions. La première (si $u \in L(T)$ et v préfixe de u différent de u alors $v \notin L(T)$) a été correctement traitée à ceci près que beaucoup oublient le cas $v = \varepsilon$ qui n'était pas couvert par la question (4) ; la seconde (concernant les préfixes de uz) a été souvent oubliée, et certains oublient de considérer le cas d'un préfixe de longueur $> |u|$.

En outre, pour l'application à la question (6), certains parlent de « unique préfixe de w » pour « unique préfixe de w qui appartienne à $L(T)$ ». Il n'est pas clair s'il s'agit d'une confusion ou d'un oubli.

(6) La seconde sous-question (celle portant sur $u_1 \cdots u_k b u'_1 \cdots u' \ell$) n'a quasiment pas été traitée. ✠