Composition d'Informatique (4 heures), Filière MP
(XULC)

Rapport de MM. Jean-Christophe FILLIÂTRE, Paulin de NAUROIS, Steve OUDOT et Mme Stéphanie DELAUNE, correcteurs.

1. L'épreuve

Il s'agissait dans ce problème de construire un algorithme simplement exponentiel pour vérifier la satisfiabilité d'une formule en logique temporelle. La construction de cet algorithme faisait appel à certaines notions de théorie des automates. L'épreuve alternait des questions relatives à la compréhension de la logique temporelle, des questions d'algorithmique, des questions de programmation et des questions de théorie des automates.

La partie 1 introduisait la logique temporelle et donnait quelques exemples de formules en logique temporelle. Les candidats devaient répondre à des questions simples de véracité d'une formule donnée sur un mot donné, de construction d'une formule exprimant des propriétés simples des mots la satisfiant, de construction d'un automate fini acceptant les mots satisfiant une formule donnée dans l'énoncé, et d'équivalence entre deux formules données.

La partie 2 avait pour objectif de construire un algorithme récursif exponentiel permettant de vérifier la véracité d'une formule donnée sur un mot donné. Cette partie contenait principalement des questions de programmation.

La partie 3 avait pour objectif de prouver la rationalité des langages décrits par des formules de la logique temporelle. Cette partie contenait des questions de programmation et des questions de théorie des automates. Les candidats devaient construire un automate fini (non-déterministe) reconnaissant le langage décrit par une formule.

La partie 4 avait pour objectif d'utiliser l'automate de la partie 3 pour écrire un algorithme simplement exponentiel vérifiant la satisfiabilité d'une formule de la logique temporelle. La dernière question invitait les candidats à démontrer l'existence de langages rationnels non décrits par une formule de la logique temporelle.

2. Remarques générales

Il s'agissait de la troisième édition d'une épreuve écrite commune pour les concours d'admission de l'École Polytechnique et des Écoles Normales Supérieures.

Les notes se répartissent selon le tableau suivant, avec une moyenne de 11,04 et un écart-type de 3,50. Le nombre de notes éliminatoires, c'est-à-dire inférieures à 2, est de 5.

$0 \leqslant \mathrm{N} < 4$	16	1,90 %
$4 \leqslant \mathrm{N} < 8$	150	17,84 %
$8 \leqslant \mathrm{N} < 12$	355	42,21 %
$12 \leqslant \mathrm{N} < 16$	249	29,61 %
$16 \leqslant \mathrm{N} \leqslant 20$	71	8,44 %
Total	841	100 %
Nombre de copies : 841		
Note moyenne : 11,04		
Écart-type : 3,50		

Concernant les questions de programmation, presque toutes les fonctions demandées pouvaient être écrites en moins de 10 lignes, indépendamment du langage de programmation choisi. Les candidats doivent être conscients du fait qu'une réponse longue doit être expliquée en détail et que très souvent un programme très long contient un grand nombre d'erreurs.

3. Commentaire détaillé

Pour chaque question, le pourcentage de candidats ayant traité la question et le pourcentage de candidats ayant obtenu la totalité des points sont indiqués entre crochets.

Partie I

Question 1 [100% - 79%]. Cette question élémentaire avait pour but d'aider les candidats à comprendre les formules écrites en logique temporelle. L'erreur la plus typique concerne la 4e formule et correspond à une mauvaise interprétation de la définition du connecteur **U**.

Question 2 [100% - 67%]. Là encore, il s'agissait d'une question élémentaire visant à aider les candidats dans leur compréhension des formules de logique temporelle. Beaucoup de tentatives ratées de combiner les connecteurs **F** et **U**, du type $(\mathbf{F}p_a)\mathbf{U}p_b$; ou alors oubli d'un connecteur **F**, e.g. $p_a \wedge (\mathbf{F}p_b)$.

Question 3 [98% - 61%]. Il s'agissait encore d'une question élémentaire mais elle semble avoir posé plus de difficultés. En particulier, beaucoup de réponses fausses du type **X** ($\neg$ **VRAI**).

Question 4 [99% - 84%]. Cette question a été bien traitée dans l'ensemble.

Question 5 [94% - 30%]. Il s'agissait d'écrire une formule de logique temporelle pour représenter un certain langage. Dans l'ensemble, l'alternance de a et de b est souvent bien exprimée, mais les conditions de bord (un a au début, un b à la fin) sont parfois oubliées ou incorrectement traduites.

Question 6 [96% - 52%]. Certains candidats ne traduisent pas la séquentialité entre l'occurrence de a et celle de bc (i.e. ne traduisent que la conjonction des deux). Par ailleurs, plusieurs tentatives (souvent ratées) de déterminiser l'automate, ce qui n'était pas demandé dans la question.

Note : L'énoncé de cette question contenait une typo ($\phi \models u$ au lieu de $u \models \phi$), heureusement sans conséquence.

Question 7 [95% - 39%]. La preuve demandée dans cette question était relativement facile puisqu'il s'agissait de dérouler les définitions des différents connecteurs. Malheureusement, beaucoup d'approximations dans les raisonnements proposés.

Partie II

Question 8 [99% - 88%]. Il s'agissait d'une question extrêmement simple, prétexte à écrire une première fonction récursive sur les formules. Quelques erreurs sur les cas de base (0 au lieu de 1), ou encore utilisation de l'opérateur max au lieu de +.

Question 9 [98% - 58%]. Cette question, un peu plus difficile, a été globalement bien traitée. Diverses solutions ont été proposées pour la formule équivalente à $\mathbf{F}_\varphi$ (parfois inutilement compliquées). Le code proposé était également souvent juste. L'erreur la plus fréquente est l'oubli de l'appel récursif.

Question 10 [95% - 19%]. Cette question, pourtant similaire à la précédente, a été beaucoup moins bien traitée. Beaucoup d'erreurs dans la formule permettant d'exprimer le connecteur **G**. Par exemple, la formule φ**U**Fin ne convient pas puisqu'elle ne permet pas de vérifier φ au dernier indice. Beaucoup de codes corrects mais ne respectant pas la complexité demandée.

Question 11 [93% - 4%]. Les candidats avaient ici le choix entre écrire une fonction qui suppose que i est un indice valide dans le mot, et l'assurer dans les appels récursifs, ou écrire au contraire une fonction qui ne suppose rien sur i mais renvoie faux lorsque i n'est pas un indice valide. Beaucoup de solutions se sont avérées incorrectes par manque d'un tel choix, i.e. le candidat ne suppose rien sur i mais oublie de vérifier s'il s'agit d'un indice valide.

De nombreux candidats ont également traité le cas du construction **U** à l'aide d'une ou deux boucles `while`, ce qui est inutilement compliqué (et très propice aux erreurs). Une écriture récursive utilisant le résultats de la question 7 était beaucoup plus appropriée.

Pour justifier la terminaison de la fonction `veriteN`, beaucoup de candidats écrivent « soit la taille de la formule diminue, soit l'indice i augmente ». Ce n'est pas suffisamment rigoureux. Est-ce la somme de deux quantités qui diminue, ou la paire pour un ordre lexicographique ? D'autre part, il faut justifier que l'indice i est borné supérieurement.

En ce qui concerne la complexité, peu d'analyses correctes même si beaucoup de candidats ont compris intuitivement que le code pouvait être exponentiel dans le cas le pire.

Partie III

Question 12 [92% - 72%]. Il s'agissait d'une question simple, prétexte à écrire une nouvelle fonction récursive. Cette question a été plutôt bien traitée. Quelques erreurs de tapage en Caml qui traduisaient une mauvaise compréhension du sujet, ou parfois même l'oubli de l'appel récursif.

Question 13–14 [74% - 18%]. (Les questions 13 et 14 ont été corrigées comme une seule question). Beaucoup d'approximations dans les réponses, même si la plupart des candidats ont compris qu'il fallait raisonner par induction. Le code fourni à la question 14 aidait souvent à comprendre les raisonnements proposés par les candidats à la question 13. Pour traiter le cas du connecteur **U**, il fallait se souvenir de la question 7.

Au niveau du code, beaucoup de petites erreurs traduisant peut-être une certaine fébrilité des candidats à ce niveau de l'épreuve. Enfin, beaucoup de candidats ayant bien répondu à la question 13, et ayant donné un code correct à la question 14, ont oublié de justifier la terminaison et la complexité de leur code.

Question 15 [59% - 25%]. Beaucoup de candidats ont tenté une approche par induction, oubliant tout le travail fait aux questions 13 et 14. Dans la quasi-totalité des cas, cette approche menait à une impasse. Un certain nombre de copies ont adopté la bonne approche mais itéré par ordre croissant sur les indices dans le mot au lieu d'itérer par ordre décroissant. L'énoncé demandait également de justifier la terminaison et de préciser la complexité, ce que certains candidats ont oublié de faire.

Question 16 [58% - 50%]. Peu d'erreurs à cette question facile.

Question 17 [21% - 4%]. Bien que l'automate découle naturellement des questions précédentes, relativement peu de candidats ont adopté la bonne approche. Beaucoup de tentatives de construction de l'automate par induction avec des produits en cascade menant systématiquement à des impasses.

Question 18 [34% - 15%]. Question de cours généralement bien traitée. Beaucoup de candidats ont cependant répondu à la moitié de la question seulement, en oubliant de fournir un majorant.

Partie IV

Question 19 [70% - 13%]. Plusieurs approches possibles à cette question, dont un parcours de graphe (en largeur ou en profondeur) ou une méthode de point fixe. Toutes ont été proposées dans les copies avec divers degrés de maîtrise des algorithmes. L'approche par point fixe, au programme, était plus facile à expliquer informellement et était de fait généralement mieux traitée. Décrire un algorithme informellement ne dispense pas d'expliquer pourquoi il termine.

Question 20 [46% - 7%]. Très peu de réponses correctes, la plupart des candidats ayant oublié les questions 17 et 18, et pensant que $|u_{\min}| \leq \texttt{taille}(\varphi)$. Les tentatives de preuve de cette inégalité par induction menaient évidemment toutes à l'échec.

Question 21 [21% - 1%]. Question un peu redondante avec la question 19, mais étrangement peu de candidats ont pensé à réutiliser l'approche décrite à la question 19. Au contraire, du fait de leur erreur à la question 20, beaucoup de candidats ont opté pour une itération naïve sur l'ensemble des mots de longueur inférieure à $\texttt{taille}(\varphi)$, ce qui permettait d'atteindre la borne de complexité demandée mais était incorrect puisqu'en réalité il aurait fallu tester tous les mots de taille inférieure à $2^{\texttt{taille}(\varphi)}$, ce qui pour le coup donne une complexité doublement exponentielle en $\texttt{taille}(\varphi)$ (contre simplement exponentielle comme demandé dans l'énoncé).

Question 22 [21% - 1%]. Là encore, très peu de réponses à cette question. La plupart des réponses optaient pour une induction structurelle sur la formule. Malheureusement, les détails laissaient souvent à désirer, en particulier au niveau du connecteur **U** (le plus difficile à traiter).