Toutefois, cette question assez ouverte a donné lieu à d'excellentes prestations.

Conclusion

La préparation de cette épreuve de Sciences Industrielles pour l'Ingénieur ne s'improvise pas. Elle est destinée à valider d'autres compétences que celles évaluées par les autres disciplines en s'appuyant sur des réalisations industrielles qu'il faut appréhender dans leur complexité. Cette préparation doit donc s'articuler autour de l'analyse et de la mise en œuvre de démarches de résolution rigoureuses.

Informatique

Présentation du sujet

Ce sujet traitait des langages polynomiaux, c'est-à-dire des langages L tels qu'il existe un programme prenant en entrée un mot et répondant à la question « Ce mot fait-il partie de L ? » en un temps majoré par un polynôme en la longueur du mot. Le but du sujet était de prouver que si L est polynomial, alors L^* l'est aussi. Ce sujet avait pour objectifs d'évaluer les connaissances des candidats sur les langages reconnaissables, les automates finis, les programmations récursive et dynamique ainsi que la gestion de files dans un algorithme.

Analyse globale des résultats

Les réponses aux questions traitant des automates souffrent toujours de mêmes défauts. Il est aberrant de voir des raisonnements sur les automates sans le moindre dessin. Bien entendu, un dessin ne suffit pas au raisonnement, mais sans ce dernier, les quelques pages d'opérations ensemblistes dans lesquelles se noient certains élèves sont rarement compréhensibles et le plus souvent fausses. Le lemme de l'étoile est toujours aussi mal compris par les candidats qui se perdent dans des pages de quantificateurs. Rappelons que ce lemme n'est autre qu'une application du principe des tiroirs qui dit que si l'on range $n+1$ objets dans n tiroirs, un des tiroirs en contient au moins deux. Certains candidats ouvrent naïvement le tiroir qui les arrange et espèrent trouver plusieurs objets. En ce qui concerne la programmation, les candidats qui utilisent Caml se ruent bien trop souvent sur le style récursif, alors qu'un style impératif est parfois mieux adapté. Les programmes sont souvent mal, voire pas du tout indentés, les noms des variables sont peu explicites et certains candidats semblent avoir oublié que le Caml comme le Pascal possèdent un type booléen, ce qui permet d'éviter de les simuler avec des entiers.

Commentaires sur les réponses apportées, et conseils aux candidats

La première question (I.A.1) du problème a dérouté de nombreux candidats. Le jury attendait essentiellement du candidat un (pseudo) programme qui effectuait une boucle parcourant dans l'ordre les lettres du mot et évaluant si l'état atteint était un état final ou non. De nombreux candidats ne semblent pas savoir ce qu'est un automate déterministe complet : nous avons été surpris de voir de tels automates avec plusieurs états initiaux, des fonctions de transition qui étaient définies partiellement et qui associaient un ensemble d'états à un état et une lettre. La seconde question (I.A.2) était une question de cours, et a été plutôt bien traitée, bien que la plupart des candidats a construit des automates qui reconnaissaient un langage contenant L^*, mais aussi d'autres mots. Il y a en effet deux questions différentes à gérer : ce qui est *a priori* le plus délicat (la simulation des transitions instantanées) est en général bien traité (avec un bémol sur la question du formalisme déterministe/non-déterministe), alors que la question de la reconnaissance du mot vide est souvent traitée avec erreur (l'état initial est déclaré final, ce qui conduit à accepter trop de mots). Moins de dix pour cent des candidat « clonent » l'état initial et ses transitions sortantes pour traiter cette question.

Le lemme de l'étoile, bien que trop souvent noyé dans un amas peu lisible de quantificateurs, a été mieux utilisé que les années précédentes. Le jury tient à rappeler qu'un texte bien construit sera toujours plus convaincant qu'une succession de phrases mathématiques quantifiées qui, pour de nombreuses copies, sont fausses. Nous avons aussi retrouvé bien souvent l'erreur classique qui consiste à dire que si un automate reconnaît un langage L contenant le langage L', alors ce même automate reconnaît L'. La question I.C.2 demandait de montrer que L_1 est polynomial. Il fallait donc prouver qu'on pouvait déterminer en temps polynomial si un entier n est une puissance de 2. De nombreux candidats ont calculé le logarithme en base 2 de n et ont évalué s'il était égal à sa partie entière. Cette approche n'est absolument pas satisfaisante. Une succession de division euclidiennes par 2 (en vérifiant que les restes successifs sont nuls) était bien entendu à privilégier. Pour la question I.D.1, nous n'attendions pas un programme, mais plutôt un argument disant que l'on pouvait vérifier si le mot était de la forme a#b#c# en $O(n)$ opérations puis vérifier si $a \times b = c$ en $O(n^2)$ opérations, ou en $O(1)$ si l'on estimait que le produit pouvait se faire en temps constant. À ce sujet : lors d'une situation non spécifiée comme ici (entiers de longueur majorée ou pas), la règle est que le candidat a toujours raison, pour peu qu'il précise le point de vue qu'il prend.

La question II.A.1 a été très mal traitée. Il est anormal qu'après deux ans d'option informatique, des copies, par ailleurs plutôt bonnes, présentent des algorithmes faux ou très mauvais pour calculer la décomposition d'un nombre en base 2. Le plus simple est bien entendu de faire des divisions euclidiennes successives par 2. On commence ainsi par obtenir le bit de poids faible pour finir par l'obtention du bit de poids fort. De trop nombreuses copies cherchent à obtenir la décomposition en base 2 dans l'autre ordre

et oublient qu'en Caml, la fonction puissance n'existe pas. Nous ne pouvions mettre tous les points à ce type de réponse sur un algorithme aussi classique. De nombreux utilisateurs de Caml ont oublié de donner le type de la fonction demandée dans la question II.A.4. Nous rappelons que, même si ce n'est pas explicitement demandé, il est toujours agréable pour le correcteur de voir le type des fonctions Caml, et que s'imposer cette tâche est un moyen pour le candidat de vérifier que son code est compatible avec le type des variables.

La question II.B.2 a souvent donné lieu à l'erreur suivante : lorsqu'un préfixe strict est détecté comme appartenant à L, la fonction retourne `est_dans_L_etoile s`, avec s le suffixe correspondant. Ceci n'est pas correct car il est possible qu'un mot m de L^* s'écrive comme le produit de deux mots $m_1 \cdot m_2$ où m_1 est dans L et m_2 n'est pas dans L^*. La complexité évaluée à la question suivante a souvent été (très!) sous-évaluée (quadratique, voire linéaire) et parfois sur-évaluée (factorielle). Aucun candidat n'a réussi à évaluer précisément le coût dans un cas « le plus défavorable » où on cherche par exemple à évaluer l'appartenance de $a^{n-1}b$.

La partie sur la programmation dynamique n'a pas toujours été comprise. Il y avait essentiellement deux idées à mettre en avant : d'une part on tabule les $T_{i,j}$ pour ne pas avoir à les refaire en permanence (et obtenir une complexité apocalyptique), et d'autre part on les calcule à $j-i$j croissant. Les codes proposées avec des boucles en i puis j ne fonctionnaient pas (ils faisaient appels à des valeurs non encore calculées).

La dernière partie commençait par la mise en place d'une structure de file. De nombreux candidats utilisant Caml ont écrit au programme faux pour la fonction `get`, souvent sur le modèle :

```
let get f=
  begin
    f.contenu.(f.debut);
    f.debut<-f.debut+1
  end;
```

Il convient de rappeler aux candidats qu'en Caml, si des calculs séparés par des « ; » seul le dernier peut être autre chose qu'un effet de bord. Enfin, dans la question III.A.2 nous avons mieux récompensé les personnes utilisant une structure de donnée circulaire que les candidats décalant l'ensemble des éléments chaque fois qu'un élément sortait de la liste. Ceux décalant l'ensemble à chaque ajout d'élément n'étaient que très peu récompensés.

Pour terminer cette analyse du sujet, signalons une troisième façon de prouver le résultat abordé dans ce problème (qui était prouvé par les deux dernières approches, comme certains candidats lucides ont eu la bonne idée de le signaler au moment de faire le bilan comparé des quatre attaques du problème) : calculer de proche en proche les ensembles $E_k \subset \{0,\ldots,k\}$ constitués des indices f tels que $w_0 \ldots w_f \in L^*$, avec $w_{f+1} \ldots w_k$ ne contenant aucun préfixe non-vide appartenant à L. Les détails manquants constituent un bon exercice pour les futurs candidats!

Conclusion

En conclusion, nous attirons l'attention des candidats sur le fait que nous attendons d'eux des démonstrations claires et concises qui sont rarement compatibles avec un emploi abusif de quantificateurs (conduisant le plus souvent à des raisonnements difficiles à lire et en général faux). Nous aimerions aussi que les algorithmes de base soient mieux maîtrisés.

Tout cela ne doit pas faire oublier que nous avons aussi vu d'excellentes copies qui montrent que de nombreux candidats ont acquis, au terme de ces deux années d'option informatique, des connaissances solides pour la suite de leurs études scientifiques.

Langues

Allemand

Présentation du sujet

Version

Ce texte journalistique sur le harcèlement au travail ne constituait pas une surprise pour les candidats et permettait à ceux d'entre eux ayant approfondi le champ lexical du monde du travail de valoriser leurs connaissances avec des termes comme *Arbeitsplatz – Mitarbeiter-Mobing- eine Arbeit erledigen – das Unternehmen – Personalführungskräfte – Seminar- Kündigung …*

Il permettait également de valoriser les connaissances quant au lexique lié au monde de la justice avec des termes comme *ungerechtfertigt – gesetzlich geschützt-nicht nachweisbar - die Opfer-sich vor Gericht wehren-Vorfälle protokollieren…*