Étude de la commande de la porte en mode dégradé

Placées en fin de sujet, bien peu de candidats ont abordé les deux dernières questions relatives à la commande séquentielle de la porte en mode dégradé. Les GRAFCET proposés sont le plus souvent entachés d'erreurs rédhibitoires de syntaxe. Le jury a pu cependant corriger certaines propositions tout à fait pertinentes.

Plus généralement, les réponses apportées doivent être concises, exhaustives et précises. Des termes clés doivent être présents dans les réponses. Il ne faut pas produire de longs développements inutiles. Cela participe à une bonne gestion du temps et donc à l'efficacité du candidat.

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

Le sujet de l'option informatique est constitué de deux problèmes totalement indépendants. Le premier traite des mots de Lukasiewicz ; le second implémente un algorithme classique de recherche de motifs. Il balaye une partie raisonnable du programme : l'algorithmique, la complexité, la programmation et les automates finis.

Analyse globale des résultats

Le niveau moyen des candidats nous a semblé en baisse. Sur les questions assez mathématiques, le niveau est à peu près constant, mais sur deux aspects plus informatiques, nous percevons un biais significatif par rapport aux autres années : d'une part la qualité des copies en ce qui concerne la programmation est plutôt en baisse et pas vraiment satisfaisante. D'autre part, les questions relatives aux automates ont mis en évidence une très forte incompréhension de ce modèle de calcul, pour la grande majorité des candidats (disons les trois quarts). Ces deux points seront repris dans l'analyse des deux problèmes.

Commentaires sur les réponses apportées et conseils aux candidats

Mots de Lukasiewicz

Le choix de types de données convenables a posé question lors de la conception du sujet : le choix s'est porté sur les listes en Caml. Elles présentaient bien des avantages, mais étaient plus pénibles quand il s'agissait par exemple de calculer le conjugué. Notons au passage que le retournement de liste (via `rev`, qu'il était inutile de recoder) ou la concaténation étaient tout-à-fait acceptables, lorsqu'ils n'étaient pas appelés n'importe comment (on voyait parfois deux retournements à chaque appel récursif, ou une concaténation de type `l@[x]`, induisant un coût quadratique, et laissant penser que pour l'auteur d'un tel code la tête et la queue de liste jouent des rôles symétriques...). En Pascal, les listes auraient été trop pénibles à manipuler, d'où le choix des chaînes de caractères, ce qui rendait la rédaction de quelques questions plus simples qu'en Caml.

En ce qui concerne la programmation, environ 10 % des candidats arrivent à convaincre le correcteur qu'ils n'ont probablement jamais écrit un programme qui marche vraiment, face à la machine ; soit. Chez les autres, une bonne moitié présentent des lacunes assez sérieuses de divers ordres. Rappelons donc quelques conseils de bon sens, dont les objectifs ne sont pas (uniquement !) de simplifier la tâche des correcteurs : indenter correctement ; ne pas commencer un programme en bas de page ; donner des noms expressifs aux variables et fonctions annexes. Pour le fond : les programmes mélangeant impératif et fonctionnel sont au mieux difficilement lisibles, et en général faux. La reconnaissance de motif en Caml est très mal (et sous-) utilisée, en particulier dans le travail avec les listes. Enfin, la manipulation des booléens semble extrêmement peu naturelle pour beaucoup de candidats. Les « `if <test>=true then true else false` » et toutes les variations font d'abord sourire, mais leur répétition finit par inquiéter.

Pour ce qui est des aspects un peu mathématico-formels, une des tendances lourdes qui se dégage est la difficulté qu'ont de nombreux candidats avec le concept d'unicité. Ne pas réussir à prouver une unicité est une chose que nous comprenons fort bien. Mais environ la moitié des candidats ne comprend manifestement pas la nature même du problème. « Ma construction est unique, donc il y a unicité » est une erreur bien naturelle à la sortie du baccalauréat. Elle est plus fâcheuse après deux années d'enseignement supérieur scientifique.

Pour ce qui est de la complexité, la question 1.A.6. était assez significative. Une bonne majorité des candidats réalise qu'une solution purement récursive induirait « beaucoup de calculs faits plusieurs fois ». Peu de candidats ont signalé que ce nombre d'appels récursifs allait devenir exponentiel (en prenant par exemple en référence le cas classique du calcul de la suite de Fibonacci). La plupart des

candidats pensent que le coût en espace sera meilleur dans le cas où on ne tabule pas. C'est une erreur pardonnable, mais en réalité, les appels récursifs successifs nécessitent de stocker de l'information : le coût en mémoire, même avec un ramasse-miettes efficace sera probablement du même ordre de grandeur que pour la version tabulée.

Comme signalé plus haut, la partie I.C. a mis à jour de grosses lacunes concernant les automates. Cette partie permettait de voir si les candidats étaient capables de prouver la non-reconnaissabilité d'un langage (dans un cadre très simple et classique) autrement qu'en récitant des hypothèses confuses et plus de la moitié du temps mal quantifiées (lemme de l'étoile). Prouver que l'intersection de deux langages reconnaissables est elle-même reconnaissable est une question très proche du cours : les réactions des candidats en disent souvent long sur leur compréhension du sujet (y compris ceux essayant de réciter approximativement une preuve faite dans un concours « concurrent » la semaine précédent cette épreuve). Il convient réellement de recadrer les connaissances des candidats sur ce sujet.

Le lemme de l'étoile, utilisé par beaucoup question I.C.1., est souvent récité n'importe comment. Seulement une petite minorité des candidats est capable d'utiliser proprement ce résultat, ou bien refait le raisonnement naturel de ce lemme (si on lit assez les lettres, on va visiter deux fois le même état ; on obtient alors un gros langage accepté par l'automate), ou encore prouve le caractère non reconnaissable en exhibant une infinité de résiduels : « en lisant 1^n depuis l'état initial d'un automate déterministe reconnaissant L, on arrive dans un état q_n qui est distinct de tous les autres q_m, puisqu'il existe un mot accepté depuis q_n et pas les autres q_m ». Certains candidats parlent de mots non reconnaissables par automate...

La question I.C.2. permet de « vérifier » que certains candidats confondent l'intersection et la réunion (ou la concaténation) ou bien ne voient dans les automates que des objets formels sans réelle signification concrète. On parle ainsi du « produit des automates » qui reconnaît de façon évidente l'intersection. Malencontreusement, l'ensemble des transitions est parfois défini comme $T_1 \times T_2$, ce qui n'a aucun sens, mais est finalement assez naturel si on doit définir un produit d'automate sans avoir compris ce qu'est un automate.

Signalons que le passage au complémentaire (pour avoir l'intersection via l'union) nécessite tout de même (pour la construction usuelle) de partir d'un automate déterministe. D'une manière générale, il est peu coûteux et souvent payant de partir d'un automate déterministe.

Nous avons enfin bien entendu rencontré régulièrement l'erreur classique : « L_1 est inclus dans L_2 qui n'est pas reconnaissable, donc L_1 n'est pas reconnaissable », ainsi que l'énoncé dual.

Pour la partie I.D., beaucoup de candidats pensent qu'un mot possède un nombre de capsules qui va diminuer strictement à chaque fois qu'on enlève une capsule... Ce n'est malheureusement pas si simple !

L'algorithme de Rabin-Karp

Les candidats rédigeant en Caml ont souvent eu du mal à s'adapter au type de donnée. Certains par mauvaise lecture de l'énoncé, d'autres par méconnaissance totale du type `string`. Il est vrai qu'on l'utilise moins souvent ; le jury a donc pudiquement fermé les yeux sur les confusions entre parenthèses et crochet, ou erreurs dans l'indexation. Par contre, le pattern matching sur les chaînes de caractère est une erreur grossière (ce serait également le cas pour des tableaux...).

La méconnaissance de l'algorithme de Hörner (Question II.B.1.a.) est toujours aussi surprenante, de la part de candidats ayant suivi un cours d'informatique. Pour la question II.B.2.b., l'erreur consistant à ne calculer le reste dans la division euclidienne qu'à la fin de la somme est un peu plus pardonnable ; on peut la mettre souvent au crédit d'un certain « manque de métier ».

Finalement, une proportion plutôt satisfaisante des candidats ayant abordé la fin de l'épreuve a compris le principe général de l'algorithme de Rabin-Karp : d'une part, on ne souhaite lire les caractères du motif qu'une seule fois (sauf lorsqu'on rencontre des (pseudo-) positions) ; d'autre part, travailler modulo q avec q assez grand (dans la mesure de ce qui est autorisé par le type d'entier) diminuera le nombre de fausses positions.

Conclusion

Le ton global de ce rapport est certainement moins optimiste que les années précédentes. Nous invitons les futurs candidats et ceux qui les préparent à se recentrer sur les fondamentaux. Le temps de préparation est restreint, et les étudiants en CPGE ne peuvent en moyenne pas assimiler en 150H de cours/TD/TP des extensions déraisonnables du programme. Nous attendons des futurs candidats qu'ils soient capables de programmer de façon limpide des choses simples (manipulations des structures récursives, par exemple), qu'ils comprennent qu'un automate fini est autre chose qu'un quintuplet, qu'un arbre n'est pas seulement un ensemble muni de quelques gadgets mathématiques ; etc... Connaître correctement le cours (ce qui inclut par exemples les preuves de construction sur les automates) et avoir pratiqué le langage de programmation pendant les heures prévues à cet effet doit être suffisant pour arriver au concours bien préparé.