Langages rationnels et automates finis

Leçon donnée par Baptiste

Le développement présenté est une implémentation en OCaml de l'algorithme de Moore (minimisation d'un automate).

Le candidat a présenté un concept utile pour l'algorithme (la congruence de Nérode), puis a présenté son code et enfin a montré une exécution de son code sur un exemple.

1 Remarques choisies

— Viser l'élégance et la généricité du code : cela évite les questions sur les points de détails...

— Ne pas commencer l'explication du code à la minute 1 ni à la ligne 1. Conseil pour une présentation de code : d'abord la spécification, puis un exemple suffisamment générique, et seulement ensuite le détail du code.

— Ne pas prendre le risque de se lancer dans un développement trop ambitieux (personne n'est sûr d'être à 100 % de ses capacités le jour de l'oral).

2 Questions et remarques du jury

Les réponses du candidat sont indiqués en *italique*.

2.1 Séance de questions

— Peux-tu développer l'algorithme sur l'exemple présenté (montrer étape par étape les congruences calculées) ?

— L'affichage du programme n'est pas très clair. Peux-tu nous le décoder ?

— Quelle est la complexité de l'algorithme ? *La complexité est en $O(|\Sigma|n^2)$, mais mon implémentation est en $O(n^3)$.*

— Quelles structures de données pour les états et pour les lettre ? *Pour les lettre : juste des entiers (pas de structure). Pour les états : chaque état est identifié par un numéro. Cela permet de représenter la relation de transition par une matrice d'entiers, ce qui est efficace d'un point de vue complexité.* On aurait pu essayer d'utiliser des types plus généraux (notamment des états ne sont pas indexés par des entiers).

— Le programme fait 208 lignes (espaces inclus) : c'est trop long !

— Quel est le lien entre le nom des nouveaux états et celui des anciens ? Par quel miracle (sic.) 5 devient 11 ? *Le candidat détaille sa réponse sur l'exemple présenté au tableau. En fait il n'y a aucun lien, les anciens noms sont perdus et les nouveaux noms correspondent à l'ordre dans lequel le programme produit son résultat.*

— Le jury a demandé à quoi cela servait de rajouter un état puis à l'automate, et si cela ne changeait pas la sémantique du programme. Après une longue discussion, la conclusion est qu'il faut éviter les hacks pour éviter les questions sur des points techniques d'optimisation du code. Ces questions ne feront pas augmenter la note. Plutôt que te compléter l'automate (au risque de changer la sémantique), on aurait pu juste ajouter la complétude de l'automate en précondition.

2.2 Remarques

— Tout y était, mais pas dans le bon ordre. Le détail de l'exemple (qui était super !) aurait dû venir avant de montrer la première ligne de code (algorithme non-trivial).

— Annoncer la structure et la taille du code avant de le détailler.

— Penser à typer le code.

— Ne pas hésiter à proposer au jury de soumettre un exemple (après la fin du développement). Cette démarche paraît héroïque mais est en fait peu risquée et peut rapporter des points.

— Le développement est trop complexe (réalisé en dix heures). Pour le simplifier, on aurait pu se concentrer uniquement sur le calcul de la congruence de Nérode.

— Le plan est « vraiment super », mais manque parfois un peu de rigueur (notamment différences entre mot vide, langage vide, et expressions rationnelles ϵ et $\emptyset$).

— La deuxième partie est un peut fourre-tout : les propriétés de clôture sur les automates sont placés à un endroit étrange.

— On aurait pu se contenter d'un « I. Langages » et « II. Automates » et être un peu plus expéditif sur les définitions de base (concaténation, préfixe, suffixe, ...).