Centrale Informatique MP 2010 — Corrigé

Ce corrigé est proposé par Olivier Levillain (École Polytechnique) ; il a été relu par Benjamin Monmege (ENS Cachan) et Guillaume Batog (ENS Cachan).

Ce sujet étudie les langages qui peuvent être reconnus par un algorithme dont le temps d'exécution est majoré par un polynôme en la longueur du mot testé. Ces langages sont appelés polynomiaux. En particulier, on cherche à montrer que si un langage L est polynomial, il en est de même du langage $L^\star$.

- La première partie propose des exemples mettant en œuvre des langages reconnus par un automate fini : les langages *reconnaissables*. Cette partie utilise la théorie des automates : on y montre que la propriété d'être reconnaissable est préservée en passant à l'étoile, et on emploie le lemme de l'étoile. Elle fait aussi intervenir de la programmation et des calculs de complexité.
- La deuxième partie propose trois algorithmes différents pour construire un programme reconnaissant $L^\star$ à partir d'un programme reconnaissant L. Les questions consistent essentiellement à écrire des fonctions et à en évaluer la complexité. Les connaissances mises en jeu sont notamment la récursivité et la programmation dynamique.
- Enfin, la troisième partie expose un quatrième algorithme utilisant les graphes. Cette partie, plus courte que les deux autres, se termine par une comparaison des algorithmes étudiés.

Cette épreuve aborde de nombreux domaines : langages, automates, graphes, programmation dynamique, calculs de complexité. Il s'agit donc d'un sujet très complet, bien adapté aux révisions de fin d'année.

Indications

Partie I

I.A.1 Pour manipuler un automate fini $\langle \mathrm{A}, \mathrm{Q}, \delta, q_i, \mathrm{F} \rangle$ en Caml, on commence par utiliser le type `char` pour représenter A. On considère ensuite Q comme un sous-ensemble $[\![1 ; n]\!]$ des entiers `int` ; l'état initial q_i est alors un entier, et F une liste d'entiers. En Caml, on peut supposer que l'on dispose d'une fonction `delta` comme indiquée dans l'énoncé, ainsi que d'une fonction `appartient_a` pour décider de l'appartenance d'un élément à une liste.

I.A.2 Pour construire un automate reconnaissant $\mathrm{L}^\star$ à partir d'un automate $\mathcal{A}$ reconnaissant L, ajouter des transitions issues d'un état final de A et d'extrémité un successeur d'un état initial de A.

I.B.1 Afin de démontrer ce résultat, on peut utiliser le lemme de l'étoile. Comme celui-ci admet plusieurs variantes, on choisira celle qui permet d'avoir $|xy| < p$ dans la décomposition d'un mot assez long en xyz.

I.C.1 Utiliser le lemme de l'étoile pour montrer que L_1 n'est pas reconnaissable.

I.C.3 Il faut se souvenir que l'alphabet considéré se réduit à la lettre a.

I.D.2 Tout comme pour la question I.C.1, on utilisera le lemme de l'étoile pour prouver que le langage n'est pas reconnaissable.

I.E.2 Utiliser le lemme de l'étoile pour montrer que la propriété est fausse.

Partie II

II.A.3 On considérera différemment d'une part les indices des lettres des sous-mots pairs et d'autre part les indices des lettres des sous-mots impairs.

II.B.3 Établir une relation de récurrence sur C_n. Puis poser $\mathrm{D}_n = \mathrm{C}_n + 1$.

II.C.2 On pourra utiliser le résultat de la question II.B.1.

II.C.3 La programmation dynamique repose sur l'idée de considérer les sous-mots des plus courts aux plus longs. On se sert en effet des résultats calculés pour les mots les plus petits afin de construire les valeurs pour les mots plus longs.

Partie III

III.A.2 Cette implémentation des files FIFO se nomme « tampon circulaire »...

III.B.3 Attention à bien ajuster la taille des structures de données utilisées.

Les conseils du jury

Le jury attend des candidats des « démonstrations claires et concises » au lieu de « pages » de quantificateurs ou d'opérations ensemblistes. En outre, il juge « aberrant de voir des raisonnements sur les automates sans le moindre dessin », rappelant qu'un dessin n'est qu'un support pour faciliter la compréhension d'un raisonnement. Concernant la programmation, il s'avère que « le style impératif est parfois mieux adapté » que le style récursif en Caml. Enfin, le jury rappelle quelques rudiments sur l'écriture des programmes : une bonne indentation, des noms de variable explicites et l'emploi du type `bool` pour simuler les booléens.

I. Quelques exemples

I.A.1 Soit L un langage reconnaissable. Par définition, il existe un automate acceptant L. Supposons, sans perte de généralité, que cet automate est déterministe complet et notons-le comme un quintuplet $\langle \mathrm{A}, \mathrm{Q}, \delta, q_i, \mathrm{F} \rangle$, où A est l'alphabet, Q l'ensemble d'états, δ la fonction de transition, q_i l'état initial et F la liste des états finaux. Supposons de plus que δ, la fonction de transition décrite dans l'énoncé, se calcule en temps O(1).

> Il est important de se rappeler que tout automate fini non déterministe (et possédant éventuellement des ε-transitions) peut être transformé en un automate fini déterministe et complet équivalent. Cette transformation peut mener à une explosion exponentielle du nombre d'états.
>
> Ainsi, la supposition faite au début de la question n'est pas restrictive, mais rend l'implémentation plus simple car on ne conserve qu'un état en mémoire, pas un ensemble d'états ; de plus, le caractère complet permet d'avoir une fonction de transition qui soit une *application*.

Montrons qu'il existe un algorithme vérifiant l'appartenance d'un mot m à L en un temps linéaire en sa longueur $|m|$. Pour cela, appelons `delta` la fonction Caml correspondant à la fonction de transition δ. Supposons de plus que F est codé comme une liste `F` d'états, et que l'on dispose de l'état initial `q_i` et d'une fonction `appartient_a` telle que `appartient_a E x` renvoie un booléen indiquant si `x` appartient à la liste `E`.

```
let reconnait_L m =
  let n = string_length m in
  let q = ref q_i in
  for i = 0 to n - 1 do
    q := delta !q m.[i]
  done;
  appartient_a F !q;;
```

Le parcours d'un mot m consiste en $|m|$ appels à la fonction de transition `delta`, suivis d'un appel à `appartient_a F x` (qui consiste au pire en un parcours de la liste F, donc est en O(1)). Ainsi, la complexité de `reconnait_L` est linéaire en $|m|$.

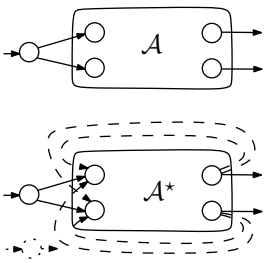
Tout langage reconnaissable est polynomial.

> La fonction `appartient_a` peut s'écrire simplement ainsi :
>
> ```
> let rec appartient_a e x =
>   match e with
>     | [] -> false
>     | y::r -> y=x or (appartient_a r x);;
> ```
>
> On aurait également pu représenter F par un tableau de |Q| booléens indiquant si un état donné est final ou non. Cette représentation, potentiellement plus gourmande en espace, permet de vérifier qu'un état est final en une opération élémentaire.
>
> 
>
> Comme l'indique l'énoncé, le rapport du jury rappelle qu'un « pseudo programme » pouvait suffire pour répondre à la question.

I.A.2 Soit L un langage reconnaissable. Il existe un automate fini déterministe $\mathcal{A}$ acceptant L. Supposons que l'automate soit de la forme $\langle \mathrm{A}, \mathrm{Q}, \Delta, q_i, \mathrm{F}\rangle$, avec Δ un sous-ensemble de $\mathrm{Q}\times\mathrm{A}\times\mathrm{Q}$ tel que, si (q_1, a, q_2) appartient à Δ, alors il existe une transition étiquetée par a entre q_1 et q_2.

> Cette représentation ensembliste des transitions permet d'ajouter simplement des transitions pouvant rendre l'automate non déterministe, ce qui n'est pas interdit par l'énoncé. On notera $\longrightarrow^\star$ la succession d'un nombre quelconque (potentiellement nul) de transitions.

Pour construire un automate $\mathcal{A}^\star$ reconnaissant $\mathrm{L}^\star$ sans ajouter de transition vide, commençons par ajouter un deuxième état initial $q^\star$, qui sera également final, de telle sorte que le mot vide soit accepté. Ensuite, afin de gérer l'enchaînement de plusieurs mots de L dans le nouvel automate, on ajoute les transitions suivantes, entre tous les états finaux et tous les successeurs de l'état initial original q_i :



$$\Delta^\star = \{(q_f, a, q) \mid q_f \in \mathrm{F} \ \wedge\ (q_i, a, q) \in \Delta\}$$

On obtient l'automate $\mathcal{A}^\star = \langle \mathrm{A}, \mathrm{Q}\cup\{q^\star\}, \Delta\cup\Delta^\star, \{q_i, q^\star\}, \mathrm{F}\cup\{q^\star\}\rangle$ non déterministe, mais ne contenant pas de transition vide.

> Le rapport du jury signale que « moins de dix pour cent des candidats *clonent* l'état initial et ses transitions sortantes. » Il ne suffit pas de déclarer final l'état initial au risque d'accepter des mots n'appartenant pas à $\mathrm{L}^\star$.

Soit m un mot de $\mathrm{L}^\star$, montrons qu'il est reconnu par l'automate $\mathcal{A}^\star$. Si m est vide, il est reconnu par l'automate puisque $q^\star$ est à la fois initial et final. Sinon, il est composé de k mots non vides w_1 à w_k appartenant à L. En notant a_j la première lettre du mot $w_j = a_j x_j$, il existe k chemins reconnaissant chacun des w_j dans $\mathcal{A}$ de la forme $q_i \xrightarrow{a_j} q_1^{(j)} \xrightarrow{x_j}{}^\star\, q_f^{(j)}$. Pour les mots w_2 à w_k, on remplace la première transition par une transition $q_f^{(j-1)} \xrightarrow{a_j} q_1^{(j)}$ présente dans $\Delta^\star$, pour obtenir le chemin suivant, acceptant m dans $\mathcal{A}^\star$:

$$q_i \xrightarrow{a_1} q_1^{(1)} \xrightarrow{x_1}{}^\star\, q_f^{(1)} \xrightarrow{a_2} q_1^{(2)} \xrightarrow{x_2}{}^\star\, q_f^{(2)} \qquad \cdots \qquad \xrightarrow{a_k} q_1^{(k)} \xrightarrow{x_k}{}^\star\, q_f^{(k)}$$

Inversement, soit m un mot reconnu par $\mathcal{A}^\star$, montrons qu'il appartient à $\mathrm{L}^\star$. Si m est vide, il appartient trivialement à $\mathrm{L}^\star$. Dans le cas contraire, il est accepté par un chemin non vide dans $\mathcal{A}^\star$ qu'on peut décomposer de la façon suivante :

$$q_i \xrightarrow{a_1} q_1^{(1)} \xrightarrow{x_1}{}^\star\, q_f^{(1)} \xrightarrow{a_2} q_1^{(2)} \xrightarrow{x_2}{}^\star\, q_f^{(2)} \qquad \cdots \qquad \xrightarrow{a_k} q_1^{(k)} \xrightarrow{x_k}{}^\star\, q_f^{(k)}$$

où les transitions de la forme $q_f^{(j-1)} \xrightarrow{a_j} q_1^{(j)}$ sont exactement celles appartenant à $\Delta^\star$. Nécessairement, les états $q_f^{(j)}$ appartiennent à F, soit en tant qu'origine d'une transition de $\Delta^\star$, soit en tant qu'état acceptant le mot.

Considérons séparément les chemins étiquetés par les mots $a_j x_j$. Par construction, le premier chemin $q_i \xrightarrow{a_1} q_1^{(1)} \xrightarrow{x_1}{}^\star\, q_f^{(1)}$ ne contient que des transitions qui appartiennent à Δ, part de l'état initial q_i et arrive à un état final $q_f^{(1)}$, donc $w_1 = a_1 x_1$ appartient à L. De manière similaire, les chemins $q_f^{(j-1)} \xrightarrow{a_j} q_1^{(j)} \xrightarrow{x_j}{}^\star\, q_f^{(j)}$ ne contiennent qu'une transition appartenant à $\Delta^\star$ de la forme $q_f^{(j-1)} \xrightarrow{a_j} q_1^{(j)}$. Or, il existe par construction une transition équivalente dans Δ, de la forme $q_i \xrightarrow{a_j} q_1^{(j)}$. Le mot