Option Informatique en Spé MP et MP*

Devoir surveillé du mercredi 27 octobre 1999

Résumé

Ce sujet est organisé en deux problèmes, comme les épreuves d'informatique de certains des concours communs.

Le premier problème vous propose de revoir le lemme de l'étoile, puis de découvrir un lemme analogue dû à Guo-Qiang ZHANG et E. Rodney CANFIELD.

Le deuxième problème est orienté vers la programmation ; il définit une structure de données permettant de décrire les parties finies ou cofinies de $\mathbb{N}$, puis demande l'écriture de fonctions réalisant les calculs usuels dans cette algèbre : tests d'appartenance et d'inclusion, complémentation, union, intersection.

Veuillez rédiger chaque problème sur une copie séparée.

Table des matières

1 Lemme de pompage et lemme de non-pompage

▶ On rappelle l'énoncé du lemme de l'étoile :

> Soit L un langage reconnaissable ; il existe un naturel N tel que tout mot u de L, de longueur au moins égale à N, se décompose en $u = xyz$ avec $y \neq \varepsilon$, $|xy| \leqslant N$ et $xy^nz \in L$ pour tout $n \in \mathbb{N}$.

Question 1 • Donnez une démonstration du lemme de l'étoile.

Question 2 • Utilisez le lemme de l'étoile pour montrer que le langage $L_1 = \{a^{n!} \mid n \in \mathbb{N}\}$ n'est pas reconnaissable.

▶ On rappelle que tout réel x possède un et un seul développement décimal propre (c'est-à-dire ne se finissant pas par une infinité de 9) ; et que ce développement est périodique à partir d'un certain rang si et seulement si x est rationnel.

▶ Soit α un réel. Notons $d(\alpha) = d_1d_2d_3\ldots$ le développement décimal propre de $\alpha - \lfloor\alpha\rfloor$ (le zéro et la virgule étant omis), considéré comme un mot infini sur l'alphabet formé des chiffres 0 à 9. Par exemple, $d(1/3) = 333\ldots$ Notons $\mathrm{Lang}(\alpha)$ le langage formé par les préfixes (finis) de ce mot infini ; par exemple, $\mathrm{Lang}(e) = \{\varepsilon, 7, 71, 718, 7182, 71828, \ldots\}$ puisque $e = 2{,}71828\ldots$

Question 3 • Montrez que $\mathrm{Lang}(1/7)$ est reconnaissable ; pour ce faire, vous construirez un automate fini déterministe complet reconnaissant ce langage.

Question 4 • Utilisez le lemme de l'étoile pour montrer que $\mathrm{Lang}(\sqrt{2})$ n'est pas reconnaissable.

▶ Il est clair que, si $\mathrm{Lang}(\alpha)$ contient le mot u, il contient aussi tous les préfixes de u. Un langage qui possède cette propriété est dit *préfixiel*.

Question 5 • Montrez que l'on peut décider (au moyen d'un algorithme que vous décrirez brièvement) si un langage reconnaissable L donné est préfixiel.

▶ Dans la littérature, le lemme de l'étoile est aussi désigné sous le nom de *lemme de pompage* (en anglais : *pumping lemma*) puisque son application permet, une fois que l'on a «amorcé» avec le facteur y, de «pomper» indéfiniment celui-ci en n'obtenant que des mots du langage.

▶ Guo-Qiang ZHANG et E. Rodney CANFIELD ont présenté un *lemme de non-pompage*, dont voici l'énoncé :

> Soit L un langage reconnaissable ; à tout mot y, on peut associer des naturels m et n vérifiant $m > n > 0$ et tels que $y^mz \in L \iff y^nz \in L$ pour tout mot z.

Nous allons commencer par démontrer ce lemme. Notons $\mathcal{A} = (Q, \delta, i, F)$ un automate fini déterministe reconnaissant L. La fonction de transition étendue est notée δ^* ; elle est définie par $\delta^*(q, \varepsilon) = q$ et $\delta^*(q, ux) = \delta\big(\delta^*(q, u), x\big)$ pour $q \in Q$, $x \in A$ et $u \in A^*$.

Question 6 • Expliquez l'affirmation suivante : dans la preuve du lemme de non-pompage, il n'est pas nécessaire de considérer le cas $y = \varepsilon$.

Question 7 • Soit $y \neq \varepsilon$. En observant la suite de terme général $q_k = \delta^*(i, y^k)$, terminez la preuve du lemme de non-pompage.

Question 8 • Utilisez le lemme de non-pompage pour montrer que le langage L_2 formé des mots sur l'alphabet A dont la longueur est un carré parfait, n'est pas reconnaissable.

Question 9 • On note u^R le *miroir* du mot u, défini par les relations $\varepsilon^R = \varepsilon$ et :

$$(u_1u_2\ldots u_n)^R = u_nu_{n-1}\ldots u_2u_1$$

Utilisez le lemme de non-pompage pour montrer que le langage suivant n'est pas reconnaissable :

$$L_3 = \{vv^Rw \mid v, w \in A^+\}$$

Question 10 • Peut-on utiliser le lemme de l'étoile pour montrer que le langage L_3 n'est pas reconnaissable ?

Question 11 ⋆⋆⋆ • Peut-on utiliser le lemme de non-pompage pour montrer que le langage $\mathrm{Lang}(\sqrt{2})$ n'est pas reconnaissable ?

2 Calculs dans l'algèbre des parties finies ou cofinies de $\mathbb{N}$

▶ Les premières questions de ce problème vous demandent de rédiger quelques fonctions simples, dont certaines font d'ailleurs partie de la bibliothèque Caml.

▶ Pour les évaluations de coûts, l'unité est le *Cons*, c'est-à-dire l'opérateur Caml `::` (qui permet aussi bien d'isoler, dans une liste, la tête et la queue, que de construire une nouvelle liste à partir d'une tête et d'une queue). La longueur d'une liste ℓ est notée $|\ell|$.

▶ Un *prédicat* est une fonction à valeurs booléennes. On dit que le prédicat p est *satisfait* par l'objet x lorsque $p(x) =$ `true`

Question 1 • Rédigez une fonction :

```
mem : 'a -> 'a list -> bool
```

spécifiée comme suit : `mex x l` indique si x est présent dans la liste ℓ ; le coût en sera un $\mathcal{O}\big(|\ell|\big)$.

Question 2 • Rédigez une fonction :

```
filtre : ('a -> bool) -> 'a list -> 'a list
```

spécifiée comme suit : `filtre p l` construit la liste des éléments de ℓ qui satisfont le prédicat p ; le coût en sera un $\mathcal{O}\big(|\ell|\big)$.

Question 3 • Rédigez une fonction :

```
intersection_simple : 'a list -> 'a list -> 'a list
```

spécifiée comme suit : `intersection_simple l1 l2` construit la liste des éléments communs aux deux listes ℓ_1 et ℓ_2 ; le coût en sera un $\mathcal{O}\big(|\ell_1| \times |\ell_2|\big)$. Vous ferez en sorte que, si aucune de ces listes ne contient de doublon, le résultat n'en contienne pas non plus.

Question 4 • Rédigez une fonction :

```
union_simple : 'a list -> 'a list -> 'a list
```

spécifiée comme suit : `union_simple l1 l2` construit la liste des éléments qui sont présents dans l'une au moins des deux listes ℓ_1 et ℓ_2 ; le coût en sera un $\mathcal{O}\big(|\ell_1| \times |\ell_2|\big)$. Vous ferez en sorte que, si aucune de ces listes ne contient de doublon, le résultat n'en contienne pas non plus.

Question 5 • Rédigez une fonction :

```
forall : ('a -> bool) 'a list -> bool
```

spécifiée comme suit : `forall p l` rend la valeur `true` si et seulement si tous les éléments de la liste ℓ satisfont le prédicat p ; le coût en sera un $\mathcal{O}\big(|\ell|\big)$.

▶ Une partie A de $\mathbb{N}$ est *cofinie* si son complémentaire est une partie finie de $\mathbb{N}$. On note $\mathcal{W}$ l'ensemble des parties finies ou cofinies de $\mathbb{N}$.

Question 6 • Montrez que $\mathcal{W}$ est la plus petite famille de parties de $\mathbb{N}$ qui contienne les parties finies et qui soit stable pour les opérations booléennes : union, intersection, complémentation. Bien entendu, ≪petite≫ est à comprendre au sens de l'inclusion.

▶ Pour décrire les éléments de $\mathcal{W}$, on définit le type Caml suivant :

```
type ficof = Finie of int list | Cofinie of int list;;
```

L'interprétation de ce type est évidente : `Finie [2;8;1]` désigne la partie $\{1, 2, 8\}$ de $\mathbb{N}$ tandis que `Cofinie [8;1;2]` désigne le complémentaire de cette partie. Comme on le constate sur ces exemples, l'ordre d'énumération dans les listes est indifférent. En revanche, il ne doit pas y avoir de doublon : par exemple, la partie $\{3, 6, 8\}$ ne devra pas être représentée par `Finie [6;3;6;8]`.

Question 7 • Rédigez une fonction :

```
appartient : int -> ficof -> bool
```

spécifiée comme suit : `appartient n p` indique si le naturel n appartient à la partie finie ou cofinie p de $\mathbb{N}$.

Question 8 • Rédigez une fonction :

```
contient : ficof -> ficof -> bool
```

spécifiée comme suit : `contient p q` indique si la partie finie ou cofinie p de $\mathbb{N}$ contient la partie finie ou cofinie q de $\mathbb{N}$.

Question 9 • Rédigez une fonction :

```
complement : ficof -> ficof
```

spécifiée comme suit : `complement p` calcule le complémentaire de la partie finie ou cofinie p de $\mathbb{N}$.

Question 10 • Rédigez une fonction :

```
union : ficof -> ficof -> ficof
```

spécifiée comme suit : `union p q` calcule la réunion des parties finies ou cofinies p et q de $\mathbb{N}$.

Question 11 • Rédigez une fonction :

```
intersection : ficof -> ficof -> ficof
```

spécifiée comme suit : `intersection p q` calcule l'intersection des parties finies ou cofinies p et q de $\mathbb{N}$.

Question 12 • Donnez une estimation du coût de vos fonctions, en nombre de *Cons*, et en fonction des longueurs des listes représentant les parties finies ou cofinies manipulées.

Question 13 • Quelles propositions faites-vous pour abaisser ces coûts ?

Question 14 • Les fonctions que vous avez rédigées ne peuvent permettre que les calculs dans l'algèbre des parties finies ou cofinies de $\mathbb{N}$. Quelle propriété du langage Caml permet à peu de frais de travailler dans l'algèbre des parties finies ou cofinies d'un autre ensemble ? Qu'en est-il alors des possibilités de diminution des coûts évoquées à la question précédente ?

FIN