Option Informatique en Spé MP et MP*

Devoir surveillé du mardi 5 décembre 2000

Extractions conservant la rationalité

Résumé

On s'intéresse dans ce texte à l'effet sur les langages rationnels de l'opération d'*extraction* : dans cette opération, on fixe une partie infinie S de $\mathbb{N}^*$ et, d'un mot donné, on ne garde que les lettres dont les indices appartiennent à S.

Il est possible de traiter la partie 4 indépendamment des autres ; il suffit de lire la définition d'une partie ultimement périodique (page 2) et d'admettre le résultat de la question 7.

Veuillez rédiger chaque partie sur une copie séparée.

Table des matières

Une triple étoile ⋆⋆⋆ signale une question plus difficile.

1 L'effaceur d'indices pairs

▶ Notons Σ l'alphabet $\{a,b\}$ et T le langage formé des mots sur cet alphabet, dans lesquels le facteur bbb n'apparaît pas. Ainsi, $\Sigma^* \setminus T$ est décrit par l'expression rationnelle $(a+b)^*bbb(a+b)^*$

Question 1 Donnez une expression rationnelle décrivant T.

Question 2 Décrivez un automate fini déterministe complet reconnaissant T.

$\boxed{\varphi}$ ▶ Notons φ l'application qui, au mot $v = v_1v_2v_3\ldots$, associe le mot $\varphi(v) = v_1v_3\ldots$ déduit de v par effacement des lettres d'indice pair ; par exemple $\varphi(\underline{a}\,a\,\underline{b}\,a\,\underline{b}\,b\,\underline{a}\,b) = abba$. Nous avons souligné les lettres qui ne sont pas effacées. Vous noterez que $\varphi(\varepsilon) = \varepsilon$, et que $\varphi(x) = x$ pour toute lettre x.

Question 3 Montrez que $\varphi(T)$ est rationnel.

▶ Nous allons généraliser le résultat de la question précédente, en prouvant que, si L est un langage rationnel, alors $\varphi(L)$ est lui aussi un langage rationnel.

Question 4 ⋆⋆⋆ Soit $\mathcal{A} = (Q, \delta, i, F)$ un automate fini reconnaissant L. Construisez un automate fini $\mathcal{A}'$ reconnaissant $\varphi(L)$. Vous donnerez des preuves complètes des deux affirmations suivantes :

1. si u est reconnu par $\mathcal{A}'$, alors u appartient à $\varphi(L)$;
2. si u appartient à $\varphi(L)$, alors u est reconnu par $\mathcal{A}'$.

▶ Nous allons montrer que $\varphi(L)$ peut être rationnel sans que L ne le soit. Commençons par définir une application x de $\mathbb{N}^*$ vers Σ par :

- $x(n) = a$ si n est un carré parfait pair ;
- $x(n) = b$ sinon.

On définit ensuite le mot $v_n = x(1)x(2)\ldots x(n)$, et enfin le langage $L = \{v_n \mid n \in \mathbb{N}^*\}$.

Question 5 Montrez que L n'est pas rationnel.

Question 6 Montrez que $\varphi(L)$ est rationnel ; vous donnerez une caractérisation très simple des mots de $\varphi(L)$.

2 Parties ultimement périodiques de $\mathbb{N}^*$

▶ Une partie S de $\mathbb{N}^*$ est *ultimement périodique* s'il existe $n_0 \geqslant 1$ et $p \geqslant 1$ tels que, pour $n \geqslant n_0$: $n \in S \iff n+p \in S$. Vous pourrez abréger *partie ultimement périodique* en p.u.p. sans encourir les foudres du correcteur.

▶ Une p.u.p. G de $\mathbb{N}^*$ est décrite par la donnée de n_0, p, de la *prépériode* $\mathcal{P} = G \cap [\![1, n_0-1]\!]$ et du *motif* répété $\mathcal{M} = G \cap [\![n_0, n_0+p-1]\!]$. Notons que la prépériode $\mathcal{P}$ est vide si $n_0 = 1$, auquel cas G est périodique ; la réciproque est fausse : on peut par exemple décrire l'ensemble des naturels impairs par $n_0 = 4$, $p = 2$, $\mathcal{P} = \{1,3\}$ et $\mathcal{M} = \{5\}$. Le motif $\mathcal{M}$ est vide ssi G est finie.

Question 7 Montrez que l'ensemble des parties ultimement périodiques de $\mathbb{N}^*$ est stable par union finie, intersection finie et complémentation.

▶ Soit L un langage quelconque. Notons $\|L\|$ l'ensemble des longueurs des mots de L : $n \in \|L\|$ si et seulement si L contient au moins un mot de longueur n. Nous nous proposons de montrer que, si L est rationnel, alors $\|L\|$ est ultimement périodique.

Question 8 Soit L un langage rationnel sur un alphabet à une seule lettre. Montrez que $\|L\|$ est une p.u.p. de $\mathbb{N}^*$.

▶ Soient X et Y deux alphabets. Un *morphisme* de X^* sur Y^* est une application $\psi : X^* \mapsto Y^*$ vérifiant $\psi(uv) = \psi(u)\psi(v)$ quels que soient les mots u et v pris dans X^*. Il est clair qu'un tel morphisme est parfaitement défini si l'on connaît les images des lettres de l'alphabet X.

Question 9 Montrez que, si $L \subset X^*$ est rationnel, alors $\psi(L)$ est lui aussi rationnel.

Question 10 Utilisez les résultats des questions 8 et 9 pour montrer que, si L est rationnel, alors $\|L\|$ est une p.u.p. de $\mathbb{N}^*$.

Question 11 Exhibez un langage L non rationnel, tel que $\|L\|$ soit une p.u.p. de $\mathbb{N}^*$.

Question 12 Montrez que, si $M \subset Y^*$ est rationnel, alors $\psi^{-1}(M)$ est lui aussi rationnel.

3 Extractions

▶ Une *extraction* est une application strictement croissante de $\mathbb{N}^*$ dans lui-même. Il est clair que la composée de deux extractions est encore une extraction ; et que, si s est une extraction, alors $s(n) \geqslant n$ pour tout $n \in \mathbb{N}^*$.

▶ À toute extraction s, on associe son image $S = s(\mathbb{N}^*)$. Inversement, à toute partie infinie S de $\mathbb{N}^*$, on associe une extraction s définie comme suit : $s(1) = \min S$, et $s(n+1)$ est le plus petit élément de $S \setminus \{s(k) \mid 1 \leqslant k \leqslant n\}$.

▶ Fixons une extraction s, et notons S son image. Nous noterons φ_S ou φ_s l'application de Σ^* dans lui-même, qui, au mot $v = v_1 v_2 \dots v_n$, associe le mot $v_{s(1)} v_{s(2)} \dots$ obtenu en effaçant les lettres dont l'indice n'appartient pas à S. Par exemple, la fonction φ définie page 2 (à l'endroit marqué d'un $\boxed{\varphi}$ dans la marge gauche) correspond au cas où S est l'ensemble des naturels impairs.

▶ Nous dirons d'une extraction s qu'elle *conserve la rationalité* si $\varphi_s(L)$ est rationnel dès que L est rationnel. Il est clair que l'ensemble des extractions qui conservent la rationalité est stable par composition.

▶ Une extraction s est ultimement périodique lorsque l'ensemble S associé l'est. Ainsi, l'exemple de la première partie correspond à une extraction ultimement périodique. Nous nous proposons de montrer que toute extraction ultimement périodique conserve la rationalité.

Question 13 Soient $n_0 \geqslant 1$ et $S = \mathbb{N}^* \setminus \{n_0\}$. Montrez que φ_S conserve la rationalité.

Question 14 Soient F une partie finie de $\mathbb{N}^*$ et $S = \mathbb{N}^* \setminus F$. Montrez que φ_S conserve la rationalité.

Question 15 $\star\star\star$ Soient $n \geqslant 1$, $p \geqslant 1$, $T = \{n + kp \mid k \in \mathbb{N}\}$ et $S = \mathbb{N}^* \setminus T$. Montrez que φ_S conserve la rationalité.

Question 16 Soient s une extraction ultimement périodique et L un langage rationnel. Montrez que $\varphi_s(L)$ est lui aussi un langage rationnel.

▶ Notons P l'ensemble des nombres premiers.

Question 17 P est-elle une partie ultimement périodique de $\mathbb{N}^*$?

Question 18 Notons L le langage décrit par l'expression rationnelle $(bba)^*$. Montrez que $\varphi_P(L)$ est rationnel.

Question 19 Comment s'énoncerait une éventuelle réciproque du résultat de la question 16 ?

4 Programmation en Caml

▶ Vous noterez que la représentation d'une p.u.p. n'est pas unique : par exemple, on peut remplacer n_0 par $n_0 + 1$ (quitte à incorporer n_0 dans la prépériode) ; on peut aussi remplacer p par $2p$ (quitte à «doubler» le motif). Nous dirons qu'une représentation est *réduite* si n_0 est minimal ; nous dirons qu'une représentation est *canonique* si elle est réduite, et si, de plus, p est minimale.

Question 20 Montrez qu'une représentation est réduite ssi l'une des deux conditions suivantes est satisfaite :

- $n_0 = 1$;
- $n_0 > 1$, et un et un seul des deux naturels $n_0 - 1$ et $n_0 + p - 1$ appartient à G.

Question 21 Donnez la représentation canonique de la p.u.p. G de $\mathbb{N}^*$ définie par :

$$n \in G \iff \big(n < 50 \wedge n \equiv 3\ [7]\big) \vee \big(n > 25 \wedge n \equiv 5\ [8]\big)$$

Rappel pour les étourdi(e)s : $\vee$ est le connecteur «ou», et $\wedge$ est le connecteur «et».

▶ Pour décrire une p.u.p. de $\mathbb{N}^*$, nous définissons le type Caml suivant :

```
type pup = {n_0 : int; p : int; pre : int list; motif : int list};;
```

Les champs `pre` et `motif` donnent respectivement la prépériode et le motif. Nous dirons qu'une telle description est *valide* lorsqu'elle vérifie les conditions suivantes :

- $n_0 \geqslant 1$ et $p \geqslant 1$;
- dans chacune des listes `pre` et `motif`, les éléments apparaissent une seule fois (il n'y a pas de doublons) ;
- tous les éléments de `pre` sont dans l'intervalle discret $[\![1, n_0 - 1]\!]$;
- tous les éléments de `motif` sont dans l'intervalle discret $[\![n_0, n_0 + p - 1]\!]$.

▶ L'emploi de références et de structures de données mutables est totalement interdit. Vous pouvez faire appel aux fonctions décrites dans l'annexe, page 5 ; pour chacune de ces fonctions, on donne la signature et la spécification. Vous n'avez pas besoin de rédiger effectivement ces fonctions.

Question 22 Rédigez en Caml une fonction de signature

```
valide : pup -> bool
```

spécifiée comme suit : `valide g` indique si `g` est la description valide d'une p.u.p.

Question 23 Rédigez en Caml une fonction de signature

```
est_vide : pup -> bool
```

spécifiée comme suit : `est_vide g` indique si la p.u.p. décrite par `g` est vide.

Question 24 Rédigez en Caml une fonction de signature

```
est_finie : pup -> bool
```

spécifiée comme suit : `est_finie g` indique si la p.u.p. décrite par `g` est finie.

Question 25 Rédigez en Caml une fonction de signature

```
appartient : int -> pup -> bool
```

spécifiée comme suit : `appartient x g` indique si x appartient à la p.u.p. décrite par `g`.

Question 26 Rédigez en Caml une fonction de signature

```
complementaire : pup -> pup
```

spécifiée comme suit : `complementaire g` construit une représentation de la p.u.p. complémentaire de la p.u.p. décrite par `g`.

Question 27 ⋆⋆⋆ Rédigez en Caml une fonction de signature

```
union : pup -> pup -> pup
```

spécifiée comme suit : `union g h` construit une représentation de l'union des p.u.p. décrites respectivement par `g` et `h`.

Question 28 Rédigez en Caml une fonction de signature

```
intersection : pup -> pup -> pup
```

spécifiée comme suit : `intersection g h` construit une représentation de l'intersection des p.u.p. décrites respectivement par `g` et `h`.

Question 29 ⋆⋆⋆ Rédigez en Caml une fonction de signature

```
contient : pup -> pup -> bool
```

spécifiée comme suit : `contient g h` indique si la p.u.p. décrite par `g` contient la p.u.p. décrite par `h`.

Question 30 ⋆⋆⋆ Rédigez en Caml une fonction de signature

```
reduction : pup -> pup
```

spécifiée comme suit : `reduction g` construit une représentation réduite de la p.u.p. décrite par `g`.

Question 31 ⋆⋆⋆ Rédigez en Caml une fonction de signature

```
canonisation : pup -> pup
```

spécifiée comme suit : `canonisation g` construit la représentation canonique de la p.u.p. décrite par `g`.

Annexe : boîte à outils Caml

▶ Voici la description de la boîte à outils dont vous pouvez disposer. Tout d'abord, les signatures des différentes fonctions :

```
intervalle : int -> int -> int list
mem : 'a -> 'a list -> bool
filtre : ('a -> bool) -> 'a list -> 'a list
forall :  ('a -> bool) -> 'a list -> bool
uniq : 'a list -> 'a list
pgcd : int -> int -> int
```

Voici maintenant la spécification de chacune de ces fonctions :

- `intervalle i j` construit l'intervalle discret $[\![i, j]\!]$, réduit à la liste vide si $i > j$; par exemple, `intervalle 2 5` rendra la liste `[2;3;4;5]` (à l'ordre près).
- `mem x l` indique si x apparaît dans la liste ℓ ; par exemple, `mem 3 [7;2;3;0]` rendra la valeur `true`.
- `filtre p l` construit, à partir de la liste ℓ, une nouvelle liste ne contenant que les éléments qui satisfont le prédicat p ; par exemple, `filtre (fun x -> x>0) [2;0;3;-3]` rendra la liste `[2;3]` (à l'ordre près).
- `forall p l` indique si tous les éléments de la liste ℓ satisfont le prédicat p ; par exemple, `forall (fun x -> x>0) [6;0;8]` rendra la valeur `false`.
- `uniq l` construit, à partir de la liste ℓ, une nouvelle liste sans «doublons» ; par exemple, `uniq [2;8;3;2;7;3]` rendra la liste `[2;8;3;7]` (à l'ordre près).
- `pgcd a b` calcule le PGCD de a et b ; par exemple, `pgcd 84 330` rendra la valeur 6.

One does not learn computing by using a hand calculator,
but one can forget arithmetic.
Alan J. Perlis, Epigrams on Programming

FIN