Mines Informatique MP 2007 — Corrigé

Ce corrigé est proposé par Sattisvar Tandabany (ENS Lyon) ; il a été relu par Samuel Mimram (ENS Lyon) et Vincent Puyhaubert (Professeur en CPGE).

Ce sujet est un long problème qui s'articule en quatre parties dépendant les unes des autres.

- La première partie définit les expressions rationnelles avec une écriture totalement parenthésée. L'implémentation des expressions rationnelles utilisées ici fait intervenir dans un premier temps des tableaux d'entiers, chaque symbole étant associé à un entier. Puis, une structure d'arbre est décrite pour représenter les expressions rationnelles. Cette partie consiste surtout à rédiger des programmes pour manipuler les tableaux codant des expressions rationnelles, et à transformer un tableau en arbre.
- La deuxième partie est consacrée aux langages que décrivent les expressions rationnelles. L'accent est mis sur une classe particulière de langages, dits *locaux*. Cette partie, plus algorithmique, nécessite de manipuler ces langages afin de démontrer des propriétés utiles par la suite.
- La troisième partie revient sur un peu plus de programmation. Une caractérisation des langages locaux est mise en œuvre.
- La quatrième partie traite de l'algorithme de Glushkov, permettant de construire un automate capable de reconnaître le langage décrit par une expression rationnelle. Il est demandé de dessiner quelques automates ; les indications de l'énoncé sont amplement suffisantes pour réussir cette partie.

L'ensemble du problème constitue une bonne révision des langages rationnels mais requiert une dextérité certaine dans la programmation, qui implique de réviser l'utilisation de tableaux et d'arbres.

L'énoncé rappelle que des indications données dans une partie peuvent s'avérer utiles dans les parties suivantes (et même parfois dans celles qui précèdent). C'est pourquoi il est vivement conseillé de lire l'ensemble de l'énoncé dès le début.

Indications

Première partie

3 Écrire la disjonction de cas.

4 Utiliser un compteur qui tient à jour le nombre de parenthèses ouvrantes non encore refermées.

5 Distinguer les cas selon la valeur de la case d'indice `debut` et celle de la case d'indice `fin`.

6 L'algorithme indique comment répondre pour chacune des structures possibles d'une expression rationnelle.

7 Considérer les même cas qu'à la question 5.

8 Suivre l'algorithme décrit à la question 6.

Deuxième partie

10 Étudier l'union de ces deux langages locaux dont on a précisé le quadruplet qui les caractérise :

$$\mathrm{L}_1 \text{ caractérisé par } (\mathrm{I}_1 = \{a\},\ \mathrm{F}_1 = \{b\},\ \mathrm{P}_1 = \{ab\},\ \alpha_1 = false)$$
$$\mathrm{L}_2 \text{ caractérisé par } (\mathrm{I}_2 = \{b\},\ \mathrm{F}_2 = \{c\},\ \mathrm{P}_2 = \{bc\},\ \alpha_2 = false)$$

Montrer que l'intersection de L_1, langage local caractérisé par $(\mathrm{I}_1,\ \mathrm{F}_1,\ \mathrm{P}_1,\ \alpha_1)$, et de L_2, langage local caractérisé par $(\mathrm{I}_2,\ \mathrm{F}_2,\ \mathrm{P}_2,\ \alpha_2)$, est le langage local caractérisé par $(\mathrm{I}_1 \cap \mathrm{I}_2,\ \mathrm{F}_1 \cap \mathrm{F}_2,\ \mathrm{P}_1 \cap \mathrm{P}_2,\ \alpha_1 \wedge \alpha_2)$.

Étudier la concaténation de $\mathrm{L}_1 = \{ab\}$ et de $\mathrm{L}_2 = \{bc\}$.

Utiliser la caractérisation de l'étoile comme solution de l'équation sur les langages $\mathrm{X} \cup \{\varepsilon\} = \mathrm{L.X}$.

11 Le langage $\mathrm{L} - \{\varepsilon\}$ peut être exprimé à l'aide des autres langages dont on demande l'expression dans cette question.

Troisième partie

14 Procéder par récurrence sur la longueur de n. Dans la récurrence, pour le cas de l'étoile, procéder encore par une récurrence sur la taille des mots du langage.

18 L'énoncé permet d'utiliser la fonction `calcul_F`. Calculer $\mathrm{P}(e)$ par induction sur la structure d'arbre de l'expression e.

19 Utiliser la question 13 ainsi que les questions 16 et 18.

20 Utiliser un tableau temporaire pour tenir à jour les lettres rencontrées.

Quatrième partie

22 Lire l'indication donnée par l'énoncé pour la question 23.

24 S'inspirer de l'indication de la question 23 pour décrire l'automate général.

I. Expressions rationnelles

1 Il convient de distinguer 8 cas. Ci-dessous, nous présentons pour chaque expression demandée, l'expression rationnelle équivalente qui justifie que l'expression rationnelle étudiée vérifie la propriété (P) :

1. $(\varepsilon + \varepsilon)$ est équivalente à ε ;
2. $(e1 + \varnothing)$ est équivalente à $e1$, c'est-à-dire à une expression rationnelle ne contenant pas les symboles $\varnothing$ et ε par hypothèse ;
3. $(\varepsilon.\varnothing)$ est équivalente à $\varnothing$;
4. $(e1.\varnothing)$ est équivalente à $\varnothing$;
5. $(e1.\varepsilon)$ est équivalente à $e1$;
6. $((e1 + \varepsilon) + (e2 + \varepsilon))$ est équivalente à $(e' + \varepsilon)$, où e' est l'expression rationnelle $(e1{+}e2)$ où ne figurent pas les symboles $\varnothing$ et ε, car ni $e1$ ni $e2$ ne les contiennent ;
7. $(e1.(e2 + \varepsilon))$ est équivalente à $((e1.e2) + e1)$ où ne figurent pas les symboles $\varnothing$ et ε, car ni $e1$ ni $e2$ ne les contiennent ;
8. $((e1 + \varepsilon).(e2 + \varepsilon))$ est équivalente à $(e' + \varepsilon)$, où e' est l'expression rationnelle $(((e1.e2) + e1) + e2)$ où ne figurent pas les symboles $\varnothing$ et ε, car ni $e1$ ni $e2$ ne les contiennent.

2 Montrons que la propriété $\mathscr{P}(n)$, qui stipule que « si e est une expression rationnelle de longueur n, alors e vérifie (P) », est vraie pour tout $n \geqslant 1$.

- $\mathscr{P}(1)$ est vraie, car une expression rationnelle de longueur 1 est soit $\varnothing$, soit ε, soit une lettre de l'alphabet Σ, et donc ne contient pas $\varnothing$ et ε.
- $(\forall\ 1 \leqslant i \leqslant n, \quad \mathscr{P}(i)) \Longrightarrow \mathscr{P}(n+1)$: soit e une expression rationnelle de longueur $n+1$. Comme $n+1 > 1$, e n'est ni $\varnothing$ ni ε. Si e ne contient ni $\varnothing$ ni ε, alors $\mathscr{P}(n+1)$ est vraie. Sinon, trois cas sont possibles :
  - *1er cas* : $e = (e')^*$. D'après l'hypothèse de récurrence, e' étant une expression rationnelle de longueur inférieure à n, e' peut être équivalente à l'une des quatre expressions listées dans la première colonne du tableau ci-dessous. Pour chaque cas, figure dans la deuxième colonne une écriture équivalente de e montrant qu'elle vérifie la propriété (P).

e'	$e = (e')^*$
$\varnothing$	$\varnothing$
ε	ε
e' sans $\varnothing, \varepsilon$	$(e')^*$
$(e'' + \varepsilon)$	$(e'')^*$

  - *2e cas* : $e = (e_1 + e_2)$. D'après l'hypothèse de récurrence, e_1 et e_2 étant des expressions rationnelles de longueurs inférieures à n, elles peuvent être équivalentes à l'une des quatre expressions listées dans le tableau ci-dessous. Pour chaque cas, figure dans le tableau une écriture équivalente de e montrant qu'elle vérifie la propriété (P).

e_1/e_2	$\varnothing$	ε	e_2	$(e_2' + \varepsilon)$
$\varnothing$	$\varnothing$	ε	e_2	$(e_2' + \varepsilon)$
ε	ε	ε	$(e_2 + \varepsilon)$	$(e_2' + \varepsilon)$
e_1	e_1	$(e_1 + \varepsilon)$	$(e_1 + e_2)$	$((e_1 + e_2') + \varepsilon)$
$(e_1' + \varepsilon)$	$(e_1' + \varepsilon)$	$(e_1' + \varepsilon)$	$((e_1' + e_2) + \varepsilon)$	$((e_1' + e_2') + \varepsilon)$

– *3e cas* : $e = (e_1.e_2)$. D'après l'hypothèse de récurrence, e_1 et e_2 étant des expressions rationnelles de longueurs inférieures à n, elles peuvent être équivalentes à l'une des quatre expressions listées dans un tableau similaire à celui du cas précédent. On obtient cette fois le tableau suivant :

e_1/e_2	$\varnothing$	ε	e_2	$(e_2' + \varepsilon)$
$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
ε	$\varnothing$	ε	e_2	$(e_2' + \varepsilon)$
e_1	$\varnothing$	e_1	$(e_1.e_2)$	$((e_1.e_2') + e_1)$
$(e_1' + \varepsilon)$	$\varnothing$	$(e_1' + \varepsilon)$	$((e_1'.e_2) + e_2)$	$((((e_1'.e_2') + e_1') + e_2') + \varepsilon)$

- Conclusion : Pour tout n, $\mathscr{P}(n)$ est vraie.

> Dans la récurrence précédente, il existe des entiers n pour lesquels on ne peut pas trouver d'expression rationnelle de longueur $n+1$. Par exemple pour $n = 1$, il n'existe pas d'expression rationnelle de longueur 2, car toute expression rationnelle est, ou bien un symbole unique, ou bien une expression faisant intervenir deux parenthèses et un troisième symbole. Dans ce cas, $\mathscr{P}(n+1)$ est automatiquement vraie, car fondée sur un quantificateur universel dans un ensemble vide.

3 La fonction demandée procède par une disjonction des cas.

```
let est_symbole n =
  n = ETOILE || n = PLUS || n = POINT || n = P_O || n = P_F;;
```

> Les cinq constantes étant choisies à valeur négative, une manière simple de déterminer si le paramètre est un symbole pourrait être de vérifier que l'entier est négatif. Il est cependant préférable de faire intervenir le nom de chacune des constantes pour que le programme soit facilement modifiable, au cas où les valeurs des constantes changeraient.

4 Afin que l'indice `i` vérifie simultanément les trois conditions, parcourons le tableau à partir de l'indice `debut+1`, et maintenons à jour un compteur qui donne le nombre de parenthèses ouvrantes rencontrées et qui n'ont pas encore été refermées.

Il s'agit simplement d'incrémenter ce compteur lorsqu'on rencontre une parenthèse ouvrante, et de le décrémenter dans le cas d'une parenthèse fermante. Ainsi, lors du parcours du tableau, le plus petit `i` vérifiant les trois conditions sera l'indice de la première case rencontrée qui contient la valeur `PLUS` ou la valeur `POINT` avec le compteur de parenthèses égal à zéro. Si un tel indice n'est pas rencontré, on renvoie -1.

```
let cesure expr debut fin =
  let cmpt_par = ref 0 in
  let indice = ref (debut+1) in
    while !indice < fin
      && ((expr.(!indice) <> PLUS && expr.(!indice) <> POINT)
           || !cmpt_par != 0) do
      if expr.(!indice) = P_O then cmpt_par := !cmpt_par + 1;
      if expr.(!indice) = P_F then cmpt_par := !cmpt_par - 1;
      indice := !indice + 1;
    done;
    if !indice = fin then -1 else !indice;;
```