

Expressions régulières

OPTION INFORMATIQUE - TP n° 3.8 - Olivier Reynet

À la fin de ce chapitre, je sais :

- ☞ faire le lien entre un ensemble de mots et une expression régulière
- ☞ utiliser la syntaxe des expressions régulières
- ☞ utiliser la sémantique des expressions régulières pour simplifier une expression régulière
- ☞ utiliser le filtrage (pattern matching) sur un type algébrique
- ☞ définir et utiliser un type algébrique

A Exprimer par des mots des expressions régulières

Tenter de décrire en français les langages dénotés par les expressions régulières suivantes :

- A1. $\Sigma\Sigma$
- A2. $(\varepsilon + \Sigma)(\varepsilon + \Sigma)$
- A3. $(\Sigma\Sigma)^*$
- A4. $\Sigma^* a \Sigma^*$
- A5. $\Sigma^* ab \Sigma^*$
- A6. $\Sigma^* a \Sigma^* b \Sigma^*$
- A7. $(ab)^*$

B Des mots aux expressions régulières

Soit l'alphabet $\Sigma = \{a, b\}$. Trouver une expression régulière qui dénote l'ensemble des mots :

- B1. de longueur paire
- B2. de longueur impaire
- B3. de longueur au moins un et au plus trois
- B4. qui possèdent un nombre pair de b
- B5. qui possèdent un nombre impair de a
- B6. qui possèdent un nombre de a multiple de 3

C Combien de mots dans le langage?

Soit l'alphabet $\Sigma = \{a, b\}$. Combien de mots de longueur 100 sont-ils dans $\mathcal{L}_{ER}(e)$?

C1. $e = a(a|b)^*b$

C2. $e = a^*bab^*$

C3. $e = (a|ba)^*$ (On peut utiliser $(u_n)_{n \in \mathbb{N}}$ le nombre de mots de longueur n dans $\mathcal{L}_{ER}(e)$.)

D Simplification d'expressions régulières

Simplifier les expressions régulières suivantes :

D1. $\varepsilon|ab|abab(ab)^*$

D2. $aa(b^*|a)|a(ab^*|aa)$

D3. $a(a|b)^*|aa(ab^*)|aaa(a|b)^*$

E Miroirs et induction

■ **Définition 1 — Mot miroir.** Le mot miroir d'un mot $w = a_1a_2 \dots a_n$ est $w^R = a_n a_{n-1} \dots a_1$.

■ **Définition 2 — Langage miroir.** Soit \mathcal{L} un langage sur Σ . Le langage miroir de \mathcal{L} est :

$$\mathcal{L}^R = \{w^R, w \in \mathcal{L}\} \quad (9)$$

E1. Montrer que pour deux mots v et w d'un langage \mathcal{L} on a $(vw)^R = w^R v^R$.

E2. Montrer que si \mathcal{L}_1 et \mathcal{L}_2 sont deux langages, on a $\mathcal{L}_1^R \cup \mathcal{L}_2^R = (\mathcal{L}_1 \cup \mathcal{L}_2)^R$.

E3. Montrer que si \mathcal{L}_1 et \mathcal{L}_2 sont deux langages, on a $\mathcal{L}_1^R \mathcal{L}_2^R = (\mathcal{L}_2 \mathcal{L}_1)^R$.

E4. Montrer que si \mathcal{L} est un langage, on a $(\mathcal{L}^*)^R = (\mathcal{L}^R)^*$.

E5. Définir de manière inductive une fonction miroir dont le paramètre d'entrée est une expression régulière e et qui renvoie l'expression régulière miroir e^R qui dénote le langage $\mathcal{L}_{ER}^R(e)$.

E6. Démontrer que $\forall e \in ER, \mathcal{L}_{ER}(e^R) = \mathcal{L}_{ER}^R(e)$, c'est-à-dire démontrer que l'algorithme de construction inductive de l'expression régulière miroir est correct.

F Implémentation d'un type expression régulière

■ **Définition 3 — Syntaxe des expressions régulières.** L'ensemble des expressions régulières \mathcal{E}_R sur un alphabet Σ est défini inductivement par :

(Base) $\{\emptyset, \varepsilon, \} \cup \Sigma \in \mathcal{E}_R$,

(Règle de construction (union)) $\forall e_1, e_2 \in \mathcal{E}_R, e_1 | e_2 \in \mathcal{E}_R$

(Règle de construction (concaténation)) $\forall e_1, e_2 \in \mathcal{E}_R, e_1 e_2 \in \mathcal{E}_R$,

(Règle de construction (fermeture de Kleene)) $\forall e \in \mathcal{E}_R, e^* \in \mathcal{E}_R$.

F1. Créer un type algébrique `regexp` OCaml qui représente une expression régulière selon la définition 3.

- F2. Créer en OCaml une variable `e` représentant l'expression régulière $(a^*|b)c$ sur l'alphabet $\Sigma = \{a, b, c\}$.
- F3. Créer une variable `esigma` de type `regexp` dont le langage dénote l'alphabet $\Sigma = \{A, B, C\}$.
- F4. Créer une variable `esigmastar` de type `regexp` dont le langage dénote l'alphabet Σ^* .
- F5. Créer une fonction récursive et utilisant le pattern matching de signature `regexp_to_string : regexp -> string` qui permet d'afficher lisiblement un type `regexp` sur la console. Par exemple, pour l'expression `esigma`, celle-ci renvoie la chaîne de caractère `((A|B)|C)`, pour `e` elle renvoie `((a)*|b)c`. On rappelle que la concaténation de chaîne de caractères se fait via l'opérateur `^` en OCaml.

G Langages vides, réduits au mot vide ou finis

- G1. Créer une fonction de signature `is_empty_language : regexp -> bool` qui teste si une expression régulière dénote le langage vide.
- G2. Créer une fonction de signature `is_reduced_to_epsilon : regexp -> bool` qui teste si une expression régulière dénote le langage réduit au mot vide.
- G3. Créer une fonction de signature `is_finite_language : regexp -> bool` qui teste si une expression régulière dénote un langage fini, c'est-à-dire qui comporte un nombre fini de mots.

H Tester l'appartenance d'un mot à un langage rationnel

- H1. Écrire une fonction de signature `matches_regex : regexp -> string -> bool` qui statue sur le fait qu'un mot appartient à un langage dénoté par une expression rationnelle. On pourra s'appuyer sur les fonctions `String.sub` et `String.length`.
- H2. Quelle est la complexité de cette fonction dans le pire des cas?

I Jouer avec les expressions régulières --> HORS PROGRAMME

Lors d'une campagne de tests, on a collecté l'évolution de la position GPS d'un véhicule. Le fichier contient toutes les positions du test.

- I1. À l'aide d'une ligne de commande et en utilisant `grep`, isoler la latitude et la longitude dans un fichier. Chaque ligne contiendra une information comme suit :
5920.7009,N,01803.2938,E