

Langages réguliers - Implémentation

Florian Bourse

La définition en OCaml des langages réguliers découle directement de leur définition par induction :



```
type 'a reglang =  
  | Empty  
  | Eps  
  | Sing of 'a  
  | Union of 'a reglang * 'a reglang  
  | Cat of 'a reglang * 'a reglang  
  | Star of 'a reglang
```

- 0 Écrire une fonction qui permet de tester si un langage régulier est vide :
`is_empty (l : 'a reglang) : bool`
- 1 Écrire une fonction qui permet de tester si un langage régulier contient le mot vide :
`has_eps (l : 'a reglang) : bool`
- 2 Écrire une fonction qui permet de tester si un langage régulier est égal au singleton $\{\varepsilon\}$:
`is_eps (l : 'a reglang) : bool`
- 3 Écrire une fonction qui permet de tester si un langage régulier est infini :
`is_infinite (l : 'a reglang) : bool`
- 4 Écrire une fonction qui permet d'appliquer l'opérateur miroir à un langage régulier, c'est-à-dire de calculer le langage obtenu en inversant l'ordre des lettres de tous les mots du langage :
`mirror (l : 'a reglang) : 'a reglang`
- 5 Écrire une fonction qui permet de tester l'appartenance d'un mot à un langage régulier :
`mem (u : 'a list) (l : 'a reglang) : bool`
- 6 Écrire une fonction qui permet d'énumérer tous les mots d'un langage régulier :
`to_list (l : 'a reglang) : 'a list`
`print_reglang (l : 'a reglang) : unit`
- 7 Écrire une fonction qui permet de calculer une expression régulière désignant un langage régulier :
`regex (l : 'a reglang) : string`

- 8 Écrire une fonction qui permet de calculer le cardinal d'un langage régulier :
- ```
length (l : char reglang) : string
```

On utilisera la définition suivante :



```
type cardinal =
 | Finite of int
 | Infinite
```

- 9 Écrire deux fonctions qui permettent de calculer l'ensemble des mots de longueur pair et de longueur impair d'un langage régulier :

```
odds (l : 'a reglang) : 'a reglang
evens (l : 'a reglang) : 'a reglang
```