Introduction aux langages

OPTION INFORMATIQUE - TP n° 3.7 - Olivier Reynet

À la fin de ce chapitre, je sais :

- ☞ utiliser le lemme de Levi
- ☞ manipuler un alphabet, un langage et ses puissances,
- ☞ programmer en OCaml des outils pour manipuler les langages.

A Mots, alphabets et lemme de Levi

On considère une alphabet Σ contenant au moins deux éléments.

A1. Soit Σ un alphabet. Soient a et b deux **lettres** de Σ. Montrer que $\forall u \in \Sigma^*, ua = bu \implies a = b$ et $u \in \{a\}^*$. On utilisera la définition inductive des mots.

A2. Soient r,s,u,v et w quatre mots de Σ^* tels que $w = ur$ et $w = vs$. Montrer que u est un préfixe de v ou que v est un préfixe de u.

A3. Soient u et v deux mots de Σ^* qui vérifient $uv = vu$. Montrer que :

$$\exists w \in \Sigma^*, \exists n, m \in \mathbb{N}, u = w^n \text{ et } v = w^m$$

A4. Soient deux mots u et v de Σ^*. Montrer que :

$$\exists p, q \in \mathbb{N}, u^p = v^q \iff \exists w \in \Sigma^*, \exists n, m \in \mathbb{N}, u = w^n \text{ et } v = w^m$$

A5. On définit les mots de Fibonacci sur l'alphabet $\Sigma = \{a, b\}$ par :

$$w_0 = \epsilon \tag{1}$$
$$w_1 = a \tag{2}$$
$$w_2 = b \tag{3}$$
$$w_n = w_{n-1} w_{n-2}, \forall n > 2 \tag{4}$$

(a) On suppose $n > 2$. Montrer que le suffixe de longueur deux de w_n est ba si n est impair et ab sinon.

(b) On suppose $n > 3$ et on définit le mot v_n comme le préfixe de w_n obtenu en supprimant les deux dernières lettres. Montrer que v_n est un palindrome.

(c) Écrire une fonction OCaml de signature `is_palindrome : string -> bool` qui permet de tester si une chaîne de caractères est un palindrome. Le pattern matching sur le type `string` n'est pas possible en OCaml car, contrairement aux listes, ce n'est pas type défini inductivement.

(d) Écrire quatre version de la fonction signature `fib_word : int -> string` qui renvoie le nième mot de Fibonnacci. Ces quatre versions correspondent à :

1. une version à récursivité multiple,
2. une version à récursivité terminale,
3. une version itérative (programmation dynamique par le bas)
4. une version avec mémoïsation (programmation dynamique récursive).

Vérifier le résultat de la question b.

B Langage et concaténation

B1. Soit $\mathcal{L}$ un langage sur Σ. Démontrer que $\mathcal{L}.\emptyset = \emptyset.\mathcal{L} = \emptyset$.

B2. Soit Σ un alphabet. Que vaut le cardinal de Σ^n en fonction du cardinal de Σ? (S'appuyer sur la définition inductive de la puissance d'un langage)

B3. On se donne l'alphabet `let sigma = ["a"; "b"; "c"]`, c'est à dire qu'on l'implémente par une liste. Écrire une fonction OCaml de signature `sigma_k : 'a list -> int -> 'a list list` qui génère le langage Σ^k sous la forme d'un liste de liste. Les éléments de cette liste seront les mots. Par exemple, `sigma_k sigma 2` renvoie :

```
[["a"; "a"]; ["a"; "b"]; ["a"; "c"]; ["b"; "a"]; ["b"; "b"]; ["b"; "c"]; ["c";
    "a"]; ["c"; "b"]; ["c"; "c"]]
```

B4. Peut-on représenter Σ^* avec cette implémentation?