TP 11 : Mots et Langages

I. Manipulation de chaînes de caractères en OCaml

Dans ce TP, on va manipuler des mots sous la forme de chaînes de caractères, représentés par le type `string`. Pour rappel :

- Un caractère simple, de type `char` se construit avec des apostrophes simples : `let c = 'a'`.
- Une chaîne de caractère se construit à l'aide de doubles guillements : `let s = "toto"`.
- On peut calculer la longueur d'une chaîne avec la fonction `String.length`.
- On peut accéder au `i`-ème caractère d'une chaîne avec la notation `s.[i]`.
- On peut concaténer deux chaînes avec l'opérateur `^`.
- D'autres fonctions sont disponibles dans le module `String` : https://v2.ocaml.org/api/String.html. En particulier, on y trouve deux fonctions `make` et `init`, dont le fonctionnement est proche de leurs homonymes du module `List`.
- Enfin, on rappelle que les chaînes sont immuables en OCaml. Cela signifie qu'il est impossible de modifier un caractère d'une chaîne. (Écrire `s.[1] <- 'a'` est invalide)

I. A. Mots et opérations basiques

> **Question 1**. Écrire une fonction `repeter : string → int → string` telle que `repeter s n` doit renvoyer une nouvelle chaîne contenant la chaîne `s` repétée n fois.

> **Question 2**. Écrire une fonction `miroir : string → string` qui, pour une chaîne donnée, renvoie la même chaîne lue dans l'autre sens. Par exemple, l'appel `miroir "abbab"` doit renvoyer la chaîne `"baaba"`. Quelle est la complexité de cette opération ?

> **Question 3**. Écrire une fonction `concatener : string list → string` qui prend en paramètre une liste de `string` et renvoie la concaténation de toutes les chaînes de ceette liste.

I. B. Facteurs, sous-mots

> **Question 4**. Écrire la fonction `est_prefixe : string → string → bool`, telle que l'appel `est_prefixe u v` renvoie `true` si `u` est un préfixe de `v`, et `false` sinon.

> **Question 5**. Écrire la fonction `est_suffixe : string → string → bool`, telle que l'appel `est_suffixe u v` renvoie `true` si `u` est un suffixe de `v`, et `false` sinon.

> **Question 6**. Écrire la fonction `est_facteur : string → string → bool`, telle que l'appel est `est_facteur u v` renvoie `true` si `u` est un facteur de `v`, et `false` sinon.

> **Question 7**. Écrire la fonction `est_sous_mot : string → string → bool`, telle que l'appel est `est_sous_mot u v` renvoie `true` si `u` est un sous-mot de `v`, et `false` sinon.

> **Question 8**. Écrire la fonction `coupe : string → int → (string * string)`, telle que l'appel `coupe u i` renvoie un couple `(v,w)` où `v` est le préfixe de longueur `i` de `u`, et où `u` est la concaténation de `v` et `w`. Quelle est sa complexité ?

> **Question 9**. À partir de la fonction précédente, écrire deux fonctions `prefixe : string → int → string` et `suffixe : string → int → string` telles que `prefixe u i` (resp. `suffixe u i`) renvoie le préfixe (resp. suffixe) de `u` de longueur `i`.

II. TD

II. A. Exercices sur les mots

Exercice 10. Sur l'alphabet $\Sigma = \{a, b\}$, donner la liste des facteurs du mot $abba$. Préciser lesquels sont des préfixes ou des suffixes.

Exercice 11. Soient Σ un alphabet, $a, b \in \Sigma$ et $u, v, x, y \in \Sigma^*$. Montrer que :
1. Si $au = bv$, alors $a = b$ et $u = v$.
2. Si $xu = yv$ et $|x| = |y|$, alors $u = v$.
3. Si $xu = xv$, alors $u = v$.
4. Si $xu = yv$ et $|x| \leq |y|$, alors x est préfixe de y et v est suffixe de u.

Exercice 12 : *Lemme de Levy*. Soient Σ un alphabet, et $u, v, w, z \in \Sigma^*$ des mots tels que $uv = wz$. Montrer qu'il existe un unique mot $m \in \Sigma^*$ tel que l'une des deux propositions suivantes est vraie :
1. $w = um$ et $v = mz$
2. $u = wm$ et $z = mv$

Exercice 13. Soient u et v deux mots sur Σ. Montrer que $uv = vu$ si et seulement si il existe $\gamma \in \Sigma^*$ et $p, q \in \mathbb{N}$ tels que $u = \gamma^p$ et $v = \gamma^q$.

II. B. Exercices sur les langages

Exercice 14. Soit Σ un alphabet quelconque. Considérons les langages $L_p = \{u \in \Sigma^* \mid |u| \text{ est pair}\}$ et $L_i = \{u \in \Sigma^* \mid |u| \text{ est impair}\}$. Exprimez les langages suivants : $L_i \mid L_p \quad ; \quad L_pL_p \quad ; \quad L_iL_i \quad ; \quad L_pL_i$.

Exercice 15. Soit le langage $\Sigma = \{a, b\}$, et les langages $L_1 = \{a, ba\}$, $L_2 = \{a^n b \mid n \in \mathbb{N}\}$, $L_3 = \varnothing$. Exprimer (formellement ou par une phrase) les langages suivants : $L_1^* \quad ; \quad L_2^* \quad ; \quad L_3^*$.

Exercice 16. Calculer LM pour les langages suivants :
1. $L = \{a, ab, bb\}$ et $M = \{\varepsilon, b, aa\}$
2. $L = \varnothing$ et $M = \{a, ba, bb\}$
3. $L = \{\varepsilon\}$ et $M = \{a, ba, bb\}$
4. $L = \{aa, ab, ba\}$ et $M = \{a, b\}^*$

Exercice 17. Soient L_1 et L_2 deux langages sur un même alphabet.
1. Si $L_1 \subset L_2$, justifier que $L_1^* \subset L_2^*$.
2. Comparer (au sens de l'inclusion) dans le cas général $(L_1 \mid L_2)^*$ et $L_1^* \mid L_2^*$.
3. Comparer (au sens de l'inclusion) dans le cas général $(L_1 \cap L_2)^*$ et $L_1^* \cap L_2^*$.

Exercice 18. Soient L, L_1, L_2 et L_3 des langages sur un alphabet Σ. Pour chaque une des affirmations suivantes, montrer si elle est vraie ou fausse.
1. $(L_1 \mid L_2)L_3 = (L_1L_3) \mid (L_2L_3)$
2. $\bigcup_{i>0} L^i = LL^i$
3. $LL^* = L^* \setminus \{\varepsilon\}$
4. $L^* = \{\varepsilon\} \mid LL^*$
5. $(L_2L_1)^*L_2 = L_2(L_1L_2)^*$