TD/TP : Expressions régulières, langages locaux, automate de Glushkov

Durant ce TP, on manipulera en OCaml des expressions régulières et on construira l'automate de Glushkov associé à une expression régulière linéaire (pas de répétitions de lettres)

On pourra réutiliser les définitions du TP sur les automates finis :

```
type etat = int;;

type alphabet = char list;;

type afd = {
    taille: int;
    init: etat;
    final: etat list;
    trans: (char * etat) list array};;
```

1 Expressions régulières

On se propose de représenter les expressions régulières ainsi :

```
type regexp =
    | Mot of string
    | Concat of regexp * regexp
    | Union of regexp * regexp
    | Etoile of regexp
;;
```

1. Construire les expressions régulières dénotant les langages suivants :
   (a) Mots sur $\{a, b\}$ finissant par $ababa$
   (b) Mots sur $\{a, b, c\}$ commençant par a et finissant par c.
   (c) $e_3 = (a|b|c) \star d(e|f)$
2. Écrire une fonction `lettres : regexp -> alphabet` renvoyant la liste des lettres utilisées dans une expression régulière et `nb_lettres : regexp -> int` déterminant le nombre de lettres utilisées. On pourra compter les répétitions éventuelles.
3. Écrire une fonction `denote_epsilon : regexp -> bool` déterminant si une expression régulière dénote un langage contenant le mot vide.

2 Langage local, automate de Glushkov

On rappelle qu'un langage est local lorsqu'il est entièrement défini par un ensemble de premières lettres, un ensemble de dernières lettres et un ensemble de facteurs de 2 lettres autorisés.

1. On considère que e est une expression régulière **linéaire**.
   (a) Écrire les fonctions `first : regexp -> alphabet`, `last : regexp -> alphabet` permettant de calculer les premières et dernières lettres possibles pour le langage dénoté par e.
   (b) Écrire une fonction `produit_cartesien : 'a list -> 'b list -> ('a * 'b) list`
   (c) En déduire la fonction `facteurs : regexp -> (char * char) list` retournant la liste des paires de lettres consécutives autorisées.
2. Construire l'automate de Glushkov associé à un langage local, on rappelle que cet automate possède $|\Sigma| + 1$ états et qu'il est déterministe.
   (a) Écrire `numerotation : alphabet -> (int -> char) * (char -> int)` permettant d'obtenir une numérotation des lettres d'un alphabet.
   (b) Écrire `auto_local : alphabet -> alphabet -> alphabet -> (char * char) list -> afd` qui s'utilisera ainsi : `auto_local sigma debut fin trans` renvoit l'afd local correspondant au langage local fourni en entrée.
   (c) En déduire une fonction `gushkov : regexp -> afd` retournant un automate reconnaissant le même langage qu'une expression régulière linéaire donnée. On prendra garde à noter l'état initial comme final si nécessaire.
   (d) Vérifiez la bonne construction de votre automate sur l'expression e_3.