

# ACTIVITÉ N° 1 : LECTURE D'UN MOT PAR UN AUTOMATE FINI DÉTERMINISTE

**Piste Bleue**

## 1. Rappel : Le type afd

**Le type etat** Pour définir les automates finis déterministes, nous introduisons tout d'abord le type produit suivant :

```
type etat = {accept : bool ; t : char -> int} ; ;
```

Ce type permet de décrire les états de l'automate. Le booléen `accept` est égal à `true` si l'état est acceptant, `false` sinon. La fonction `t` décrit les transitions à partir de cet état.

**Le type afd** Nous représenterons les automates finis déterministes par le type suivant :

```
type afd = {sigma : char list ; nQ : int ; init : int ; e : int -> etat} ; ;
```

Ici, `sigma` désigne l'alphabet, `nQ` est le nombre d'états numérotés de 1 à  $nQ$ , `init` est le numéro de l'état initial et enfin, `e` est une fonction qui à chaque entier désignant un état associe sa description de type `etat`.

## 2. Lecture d'un mot par un automate

**Partie 1 : Lecture d'un mot.** Écrire une fonction `lire : afd -> int -> string -> int` qui effectue la lecture d'un mot  $w$  (de type `string`), à partir d'un état  $i$  dans un AFD  $a$ . La fonction renvoie l'état dans lequel on arrive à la fin de la lecture du mot. Vous utiliserez les fonctions sur les chaînes de caractères données dans le sujet de TP. Dans un premier temps, vous pourrez écrire votre fonction en supposant que l'automate est *complet*, puis la réécrire dans le cas général.

**Partie 2 : Mot reconnu par un automate.** Écrire une fonction `accepte : afd -> string -> bool` qui teste si un mot est reconnu ou non par un automate.

```
let ac1 = accepte a1 ; ;
val ac1 : string -> bool = <fun>
List.map ac1 ["abba" ; "bbaaa" ; "bbaaba" ; "ba" ; "ab" ; ""] ; ;
- : bool list = [true ; false ; true ; false ; true ; false]
accepte a2 "babb" ; ;
- : bool = false
```

**Partie 3 : Tests** Définir un automate qui reconnaît les mots de la forme  $a^n b$ . Tester vos fonctions.