Compilation - TD 1

Analyse lexicale

⋆ *Expressions régulières et automates à la main* ⋆

Question 1.1 – quelques langages

Sur l'alphabet {a,b}, donner un automate fini déterministe reconnaissant les langages suivants :

- tous les mots contenant un nombre pair de "a" et un nombre pair de "b" ;
- tous les mots contenant un nombre pair de "a" ou un nombre pair de "b" ;
- tous les mots qui n'ont pas plus de quatre "a" consécutifs ;
- le langage $L = \{a^n b^p, n = p \mod 3,\ n, p \geq 0\}$.

Question 1.2 – expressions régulières

Donnez si vous pouvez des expressions régulières pour les langages précédents.

Question 1.3 – définitions régulières

On peut rapprocher la présentation des expressions régulières de la définition d'une grammaire en introduisant les *définitions régulières*, de la forme

$$\begin{array}{lcl} d_1 & \rightarrow & e_1 \\ d_2 & \rightarrow & e_2 \\ & \cdots & \\ d_k & \rightarrow & e_j\ , \end{array}$$

où chaque d_i est un nom distinct et chaque e_i est une expression régulière sur les symboles de $\Sigma \cup \{d_1, d_2, \ldots, d_{i-1}\}$ (Σ étant l'alphabet). L'introduction des définitions régulières n'accroît pas le pouvoir expressif.

Donner une définition régulière qui décrit toutes les chaînes composées de chiffres sans chiffre répété. Essayez aussi sans utiliser les définitions régulières, pour rire.

Question 1.4 – sur l'expressivité

Montrer que le langage $\{a^n b^n \mid n \geq 1\}$ ne peut pas être défini par une expression régulière.

⋆ *Analyseurs lexicaux* ⋆

Question 2.1

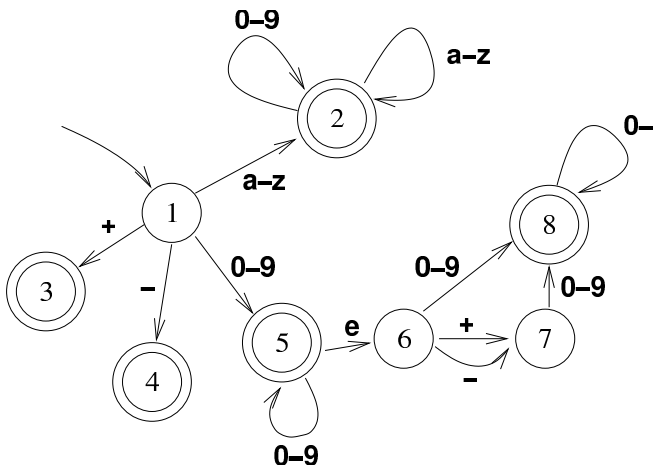

FIG. 1 – Un automate fini déterministe

Si l'on utilise l'automate de la Figure 1 pour trouver des lexèmes dans un fichier d'entrée, combien de caractères doit-on examiner au pire après la fin d'un lexème pour pouvoir l'identifier ?

Question 2.2 – les commentaires

En général un analyseur lexical reconnaît le plus long préfixe qui matche. Ccomment traiter le cas des commentaires, par exemple en Caml, de manière à ne pas oublier du code ?

```
let f x = (* une fonction bidon *)
      if x mod 2 = 0 then x/2 else x+3;;
 (* fin de la fonction bidon *)
```

⋆ *Des expressions régulières aux automates* ⋆

Question 3.1

Soit R une expression régulière sur l'alphabet Γ. Après avoir complété Γ avec un symbole ϵ signifiant "vide", donnez une méthode simple (par induction sur les expressions régulières) de construction d'un automate reconnaissant R.

Bornez le nombre d'états de l'automate construit.

Question 3.2

Cet automate est-il déterministe ou non déterministe ?

Question 3.3

Rappelez le principe de la "déterminisation" d'un automate fini non déterministe. Appliquez-le à un automate reconnaissant une expression régulière. En principe cela doit commencer à ressembler à la méthode vue en cours.