TD2 - Automates et grammaires de type 3

Objectifs

- Notions de langage rationnel ou régulier ou de type 3
- Notions d'automates à états finis et application aux langages rationnels.
- Mise en oeuvre d'une analyse lexicale/syntaxique d'un langage régulier avec des automates à états finis

Rappels et notations

Définition : Une grammaire régulière (ou rationnelle) est un quadruplet (T, N, S, P) où :

- T : ensemble des éléments terminaux.
- N : ensemble des éléments non terminaux.
- S : élément non terminal initial (axiome)
- P : ensemble de règles de la forme :
  - $X \to a$ où $a \in T$ et $X \in N$
  - $X \to bY$ où $b \in T^*$, $X \in N$ et $Y \in N$
  - $X \to \varepsilon$ si X ne constitue jamais à lui seul la partie droite d'une autre règle

  ou

  - $X \to a$ où $a \in T$ et $X \in N$
  - $X \to Yb$ où $b \in T^*$, $X \in N$ et $Y \in N$
  - $X \to \varepsilon$ si X ne constitue jamais à lui seul la partie droite d'une autre règle.

Exercice 1. Construire des automates

On étudie ici la représentation de certains langages sous forme d'automates.
Nous rappelons qu'il y a équivalence entre automate à état fini et langage régulier.

Question 1. On considère l'alphabet $A = \{0, 1\}$ et le langage $L = \{w \in A^* \mid \text{le nombre de 1 dans } w \text{ est pair}\}$.
Trouver un automate déterministe qui engendre L.

Question 2. En déduire une grammaire. Quel est son type ?

Exercice 2. Construire des automates (2)

Dans notre mini-langage de programmation d'affectation, on s'intéresse au sous-langage des affectations numériques syntaxiquement correctes.
On rappelle la grammaire associée à ce langage est $G =< T, N, S, P >$ avec :

$$T = \{\text{identifiant}, \text{nombre}, \text{opEgal}, \text{operateur}\}$$
$$N = \{\text{an}, \text{en}\}$$
$$S = \text{an}$$
$$P = \left\{ \begin{array}{ll} \text{an} \to & \text{identifiant opEgal en} \\ \text{en} \to & \text{nombre} \mid \text{identifiant} \mid \text{en operateur nombre} \mid \text{en operateur identifiant} \end{array} \right\}$$

Question 1. Trouver un automate déterministe qui engendre ce langage $L_G(S)$.

Question 2. De quel type est la grammaire proposée ? Peut-on trouver une grammaire d'un type plus élevé ?

Exercice 3. Construire des automates (3)

On considère l'alphabet A constitué des lettres de l'alphabet de la langue française et le langage

$$L = \{w \in A^* \mid w \text{ se termine par man}\}$$

On prend comme convention de ne pas représenter les transitions vers les états poubelles.

Définition : Un état poubelle est un état :

- **non final.**
- **sans transition vers un autre état.**

Question 1. Trouver un automate non déterministe qui engendre L.

Question 2. Trouver un automate déterministe directement à partir de ce langage.

Exercice 4. JFLAP

Question 1. Nous pouvons tester la validité d'un automate en JFLAP (http://jflap.org/jflaptmp). Pour cela, on utilise le bouton *Finite Automaton* du menu initial. Ensuite, il suffit de créer l'automate. Celui-ci peut ensuite être testé, en lui soumettant un ensemble de mots dans l'item *Multiple Run* du menu *Input*.

Question 2. Nous pouvons également générer une grammaire régulière (de type 3) à partir de cet automate. Pour cela, nous utilisons l'item *Convert to Grammar* du menu *Convert*.

Question 3. Il est intéressant de noter que JFLAP nous permet de vérifier si un automate possède des états non déterministes. Pour cela, nous utilisons l'item *Highlight Nondeterminism* du menu *Test*.