Cartouche du document

Année : ING 1 - Matière : Théorie des langages - Activité : Travail dirigé

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

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 —> a où a ∈ T et X ∈ N
  - X —> bY où b ∈ T^* , X ∈ N et Y ∈ N
  - X —> ε si X ne constitue jamais à lui seul la partie droite d'une autre règle

ou

  - X —> a où a ∈ T et X ∈ N
  - X —> Yb où b ∈ T^* , X ∈ N et Y ∈ N
  - X —> ε si X ne constitue jamais à lui seul la partie droite d'une autre règle

Sommaire des exercices

Corps des exercices

1 - Construire des automates

Énoncé :

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)

Énoncé de la question

On considère l'alphabet A = {0,1} et le langage L = { w ∈ A^* / le nombre de 1 dans w est pair }.

Trouver un automate déterministe qui engendre L.

Question 2)

Énoncé de la question

En déduire une grammaire. Quel est son type ?

2 - Construire des automates (2)

Question 1)

Énoncé de la question

Dans notre mini-langage de programmation d'affectation, on s'intéresse au sous-langage des affectations numériques.

Trouver un automate déterministe qui engendre le langage des affectations numériques syntaxiquement correctes.

On rappelle la grammaire associée à ce langage est :

G={

T = {identifiant,nombre,opEgal,operateur}

N = {an,en}

S = an

P = {

an —> identifiant opEgal en

en —> nombre | identifiant | en operateur nombre | en operateur identifiant

}

Trouver un automate déterministe qui engendre L.

Question 2)

Énoncé de la question

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

3 - Construire des automates (3)

Question 1)

Énoncé de la question

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

L = { w $\in$ A^* / w se termine par man }.

Trouver un automate indéterministe qui engendre L.

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 2)

Énoncé de la question

Trouver un automate déterministe directement à partir de ce langage.

Question 3)

Énoncé de la question

Trouver un automate déterministe en utilisant l'algorithme de déterminisation à partir de l'automate indéterministe.

4 - JFLAP

Question 1)

Énoncé de la question

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)

Énoncé de la question

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)

Énoncé de la question

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*.