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.

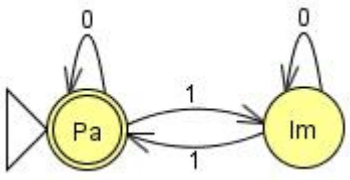
Question 1)

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.

Solution de la question



Question 2)

Énoncé de la question

En déduire une grammaire. Quel est son type ?

Solution de la question

G = {

T = {0,1}

N = (Pa,Im} // Les états de l'automate

S = Pa // L'état initial de l'automate

P = {

Pa —> ε // Car l'état Pa est état final

Im —> 1 Pa | 0 Im

Pa —> 1 Im | 0 Pa

}

}

Les règles de production de cette grammaire sont de la forme :

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

On a donc une grammaire régulière. Ce qui est normal car il y a équivalence entre automate à état fini et grammaire régulière.

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 = {

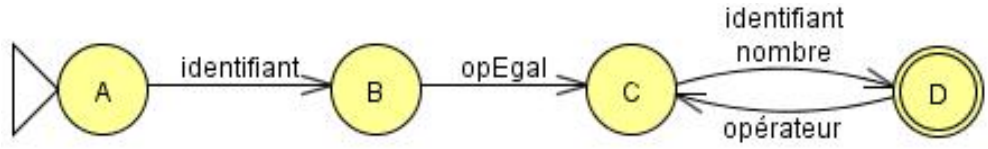
an —> identifiant opEgal en

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

}

Trouver un automate déterministe qui engendre L.

Solution de la question



Question 2)

Énoncé de la question

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

Solution de la question

La grammaire ne suit pas les règles de productions d'une grammaire de type 3 car elle mélange les 2 règles suivante :

$X \longrightarrow Yb$ où $b \in T^*$, $X \in N$ et $Y \in N$

$X \longrightarrow bY$ où $b \in T^*$, $X \in N$ et $Y \in N$

Par contre elle suit les règles de production d'une grammaire de type 2. Or, nous avons trouvé un automate à états finis dans la question précédente, donc il est possible de trouver une grammaire de type 3 pour ce langage.

En se servant de l'automate précédent, on peut en déduire la grammaire suivante :

G = {

T = {identifiant, nombre, opEgal, operateur}

N = {A, B, C, D}

S = A

P = {

A —> identifiant B

B —> opEgal C

C —> identifiant D | nombre D

D —> opérateur C

}

}

Cette grammaire suit les règles suivantes :

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

C'est donc bien une grammaire de type 3.

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 ∈ A^* / w se termine par man }.

Trouver un automate indéterministe qui engendre L.

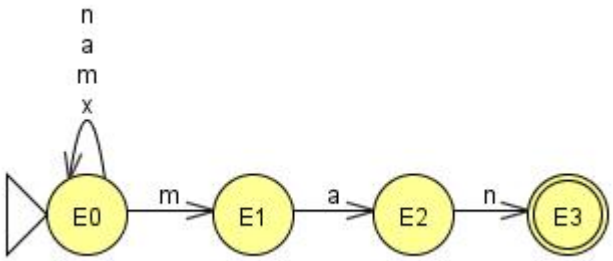
On prend comme convention de ne pas représenter les transitions vers les états poubelles.

Définition :

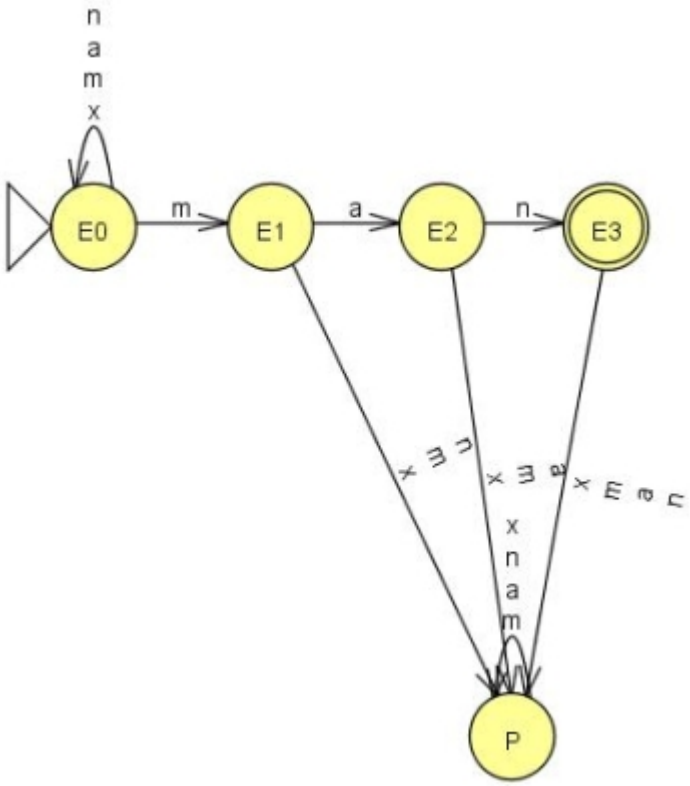
Un état poubelle est un état :

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

Solution de la question



L'automate indéterministe sans l'état poubelle



L'automate indéterministe avec l'état poubelle

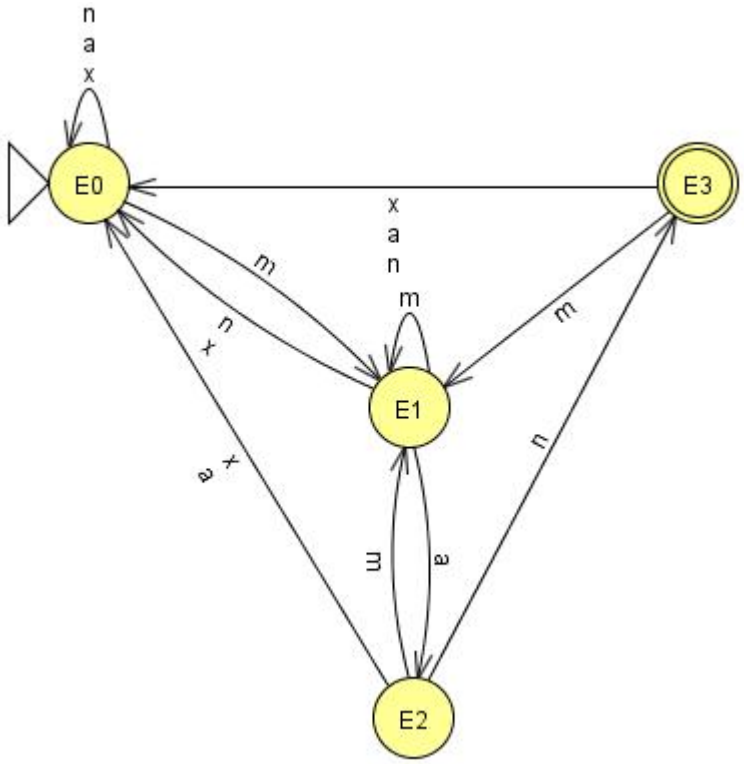
Le symbole x représente toute lettre autre que m, a ou n.

Question 2)

Énoncé de la question

Trouver un automate déterministe directement à partir de ce langage.

Solution de la question



L'automate déterministe sans l'état poubelle

Le symbole x représente toute lettre autre que m, a ou n.

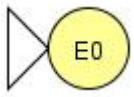
Question 3)

Énoncé de la question

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

Solution de la question

Pour déterminiser l'automate indéterministe, nous commençons par récupérer l'état initial de l'automate indéterministe :



Puis nous prenons en considération toutes les transitions à partir de cet état :

E0 —>m—> {E0, E1} // Car dans l'automate indéterministe, à partir de l'état E0, la transition m peut nous amener dans les états E0 ou E1

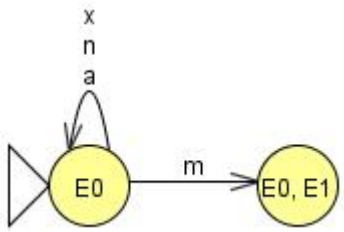
E0 —>a—> {E0}

E0 —>n—> {E0}

E0 —>x—> {E0}

Un nouvel état doit être créé : {E0, E1} l'état regroupant les états E0 et E1.

On obtient l'automate déterministe partiel suivant :



Un nouvel état a été créé ({E0, E1}), nous recommençons l'étape précédente et prenons en considération toutes les transitions à partir de cet état :

{E0, E1} —>m—> {E0, E1} // Car dans l'automate indéterministe, à partir de l'état E0, la transition m peut nous amener dans les états E0 ou E1 et à partir de l'état E1, la transition m nous amène dans l'état poubelle

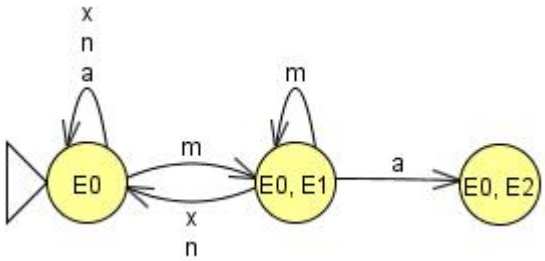
{E0, E1} —>a—> {E0, E2} // Car dans l'automate indéterministe, à partir de l'état E0, la transition a nous amène dans l'état E0 et à partir de l'état E1, la transition a nous amène dans l'état E2

{E0, E1} —>n—> {E0}

{E0, E1} —>x—> {E0}

Un nouvel état doit être créé : {E0, E2} l'état regroupant les états E0 et E2.

On obtient l'automate déterministe partiel suivant :



Un nouvel état a été créé ({E0, E2}), nous recommençons l'étape précédente et prenons en considération toutes les transitions à partir de cet état :

{E0, E2} —>m—> {E0, E1} // Car dans l'automate indéterministe, à partir de l'état E0, la transition m peut nous amener dans les états E0 ou E1 et à partir de l'état E2, la transition m nous amène dans l'état poubelle

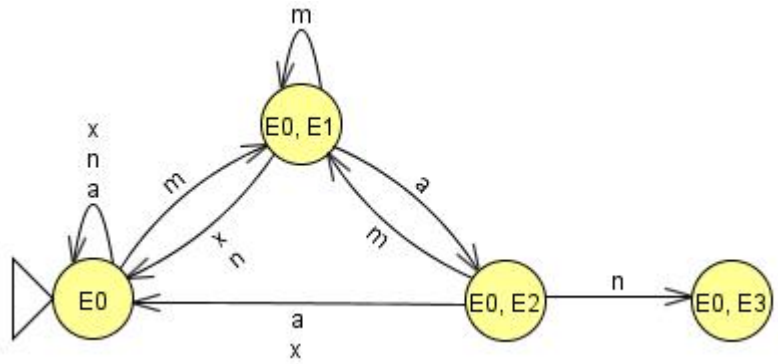
{E0, E2} —>a—> {E0} // Car dans l'automate indéterministe, à partir de l'état E0, la transition a nous amène dans l'état E0 et à partir de l'état E2, la transition a nous amène dans l'état poubelle

{E0, E2} —>n—> {E0, E3}

{E0, E2} —>x—> {E0}

Un nouvel état doit être créé : {E0, E3} l'état regroupant les états E0 et E3.

On obtient l'automate déterministe partiel suivant :



Un nouvel état a été créé ({E0, E3}), nous recommençons l'étape précédente et prenons en considération toutes les transitions à partir de cet état :

{E0, E3} —>m—> {E0, E1}

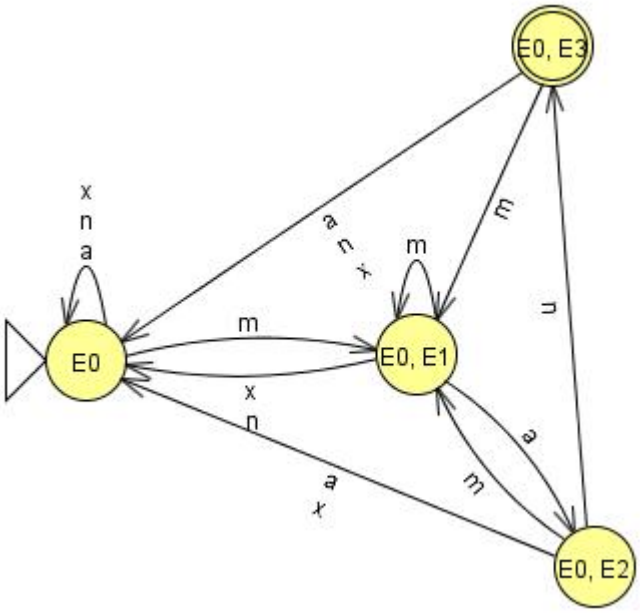
{E0, E3} —>a—> {E0}

{E0, E3} —>n—> {E0}

{E0, E3} —>x—> {E0}

Aucun nouvel état n'a été créé, l'algorithme est terminé.

On obtient l'automate déterministe final suivant :



A noter que les états finaux de notre automate déterministe sont les états contenant au moins un état final 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*.

Solution de la question

Se conférer à la réponse globale de l'exercice

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*.

Solution de la question

Se conférer à la réponse globale de l'exercice

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*.

Solution de la question

Se conférer à la réponse globale de l'exercice