Les automates à états finis

DR Abdelmoghit Souissi

Les automates déterministes (DFA) et non déterministes (NFA)

1- Les automates à états finis sont des reconnaisseurs; ils disent simplement "oui" ou "non" à propos de chaque chaîne à analyser.

2- Il y'a 2 types d'automates à états finis:

- Les automates à états finis non déterministes (AFN) n'ont aucune restriction sur les étiquettes de leurs arcs. Un caractère peut étiqueter plusieurs arcs partant d'un même état, et la chaîne vide ε est une étiquette possible.
- Les automates à états finis déterministes (AFD), pour lesquels, ne peuvent pas partir plusieurs transitions du même état avec le même caractère et n'acceptent pas d'ε-transition

Les automates déterministes (DFA)

sont deux types d'automates finis utilisés pour reconnaître des langages réguliers dans la théorie des langages et des automates.

1. Un DFA est un automate fini dans lequel, pour chaque état et chaque symbole de l'alphabet, il existe exactement une transition vers un autre état ou vers le même état.
2. Cela signifie qu'à chaque étape de traitement d'une chaîne, il n'y a qu'une seule option à suivre, ce qui rend l'automate déterministe.
3. Structure: Un DFA est défini comme un quintuplet (Q,Σ,δ,q_0,F)où :Q : un ensemble fini d'états. Σ : un alphabet fini de symboles. δ : une fonction de transition $\delta:Q\times\Sigma\rightarrow Q$, qui donne un état pour chaque couple (état, symbole).
4. q_0 : l'état initial, $q_0\in Q$
5. F : un ensemble d'états finaux (ou acceptants), $F\subseteq Q$

Les automates non-déterministes finies (NFA)

- Un NFA est un automate fini dans lequel, pour un état donné et un symbole donné, il peut y avoir plusieurs transitions possibles ou même aucune.

- Peut avoir des transitions "epsilon" (ϵ), c'est-à-dire des transitions qui se produisent sans lire de symbole. Contrairement au DFA, où il n'y a qu'une seule option à suivre à chaque étape, un NFA peut suivre plusieurs chemins possibles simultanément.

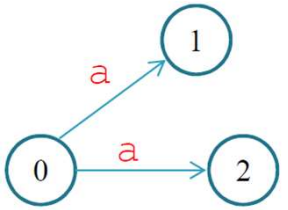
- Structure d'un NFA :Un NFA est défini de manière similaire à un DFA, mais avec une différence importante dans la fonction de transition :δ : une fonction de transition $\delta: Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$, qui donne un ensemble d'états pour chaque couple (état, symbole).

- Un NFA peut aller vers plusieurs états simultanément, ou rester dans le même état avec une transition ϵ.

Les automates non-déterministes finies (NFA)

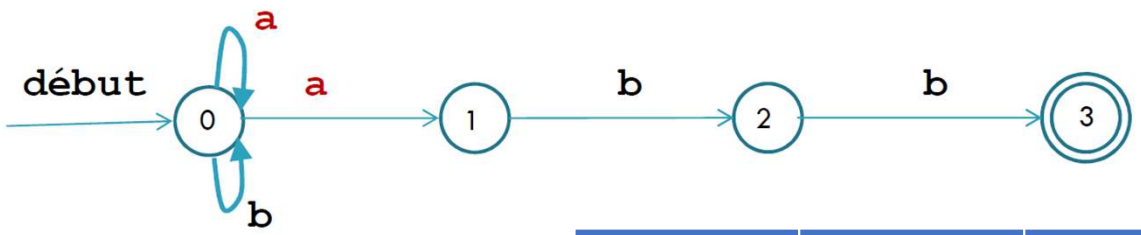
Un AFN se compose de:

- ✓ Un seul état de début.
- ✓ Un ou plusieurs états de transitions
- ✓ Un ou plusieurs états d'acceptation a
- ✓ Un caractère peut étiqueter 2 transitions partant du même état.



Exemple (NFA)

On cherche à représenter L'AFN qui reconnait le langage défini par l'expression régulière :
(a|b)* abb



Nous pouvons représenter un AFN par une table de transition, dont les lignes correspondent aux états et les colonnes aux symboles d'entrée et à ε.

Symbole/Etat	a	b	ε
0	{0,1}	{0}	-
1	-	{2}	-
2	-	{3}	-
3	-	-	-

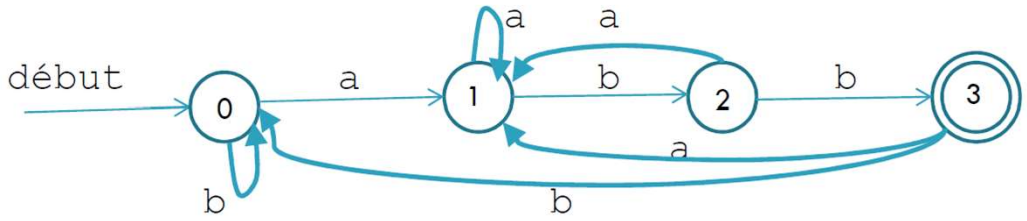
Automates à états finis déterministes (AFD):

Un AFD est un cas particulier d'un AFN où:

- Pas plus d'un arc avec le même symbole sortant du même état.
- Un AFN est une représentation abstraite d'un algorithme de reconnaissance des chaînes d'un langage.
- Un AFD est un algorithme concret de reconnaissance de chaînes.
- Remarque: Toute expression régulière et tout AFN peuvent être convertis en un AFD.

Exemple (AFD):

L'AFD qui reconnait le langage défini par l'expression régulière : (a|b)* abb



- Exemples de chaines valides: **"abb", "aaabb", "bbababb", "abbaabb", "abbbbabb"**
- Exemples de chaines non valides: "ab", "abab", "abba"

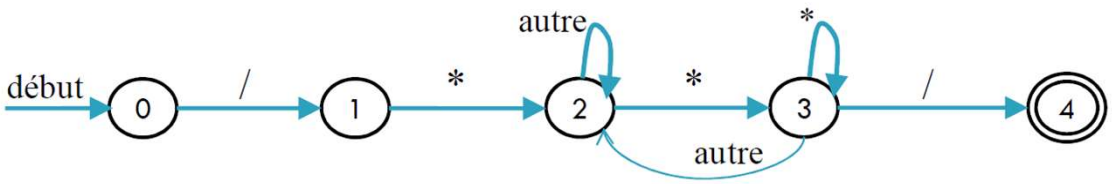
Exercice (AFD):

Exemple 2: L'automate à états finis déterministe d'un commentaire avec le langage C
exemple: /* comment */

Exemple 3: L'automate à états finis déterministe d'un commentaire avec le langage C++
exemple: // comment

Solution exercice (AFD):

Exemple 2: L'automate à états finis déterministe d'un commentaire avec le langage C
exemple: /* comment */ **^/*[\s\S]*?*/$**

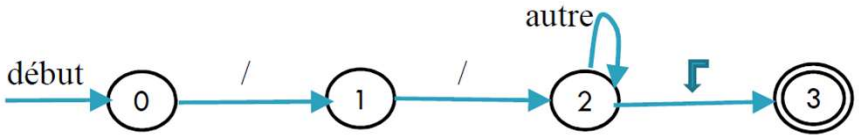


exemple:

/* comment */

Exemple 3: L'automate à états finis déterministe d'un commentaire avec le langage C++
exemple: // comment

^//.*$

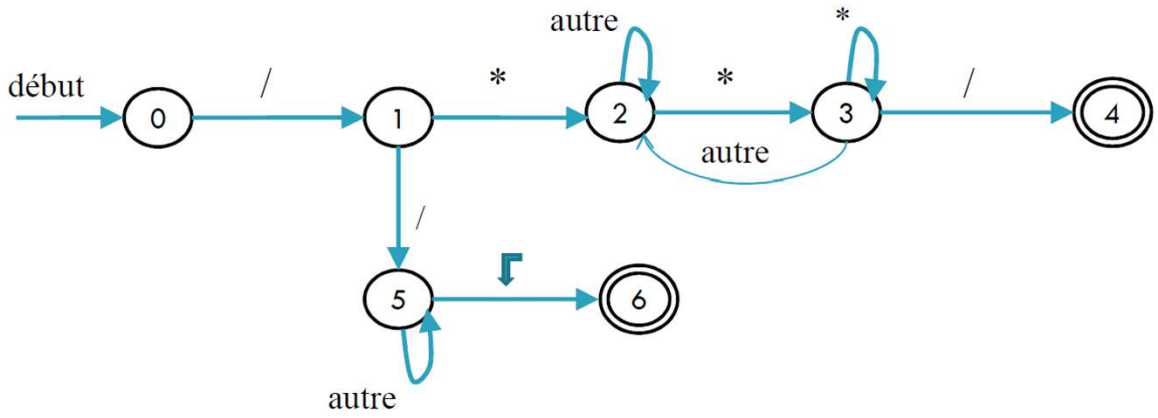


exemple:

// comment '\n'

Solution exercice (AFD):

Exemple 4: L'automate à états finis déterministe des **2 commentaires groupés**.



Définition d’un grammaire régulier

Une grammaire est régulière si toutes ses productions vérifient une des 2 formes:

$A \longrightarrow a\ B$
ou
$A \longrightarrow a$

avec:

- A et B des non-terminaux
- a un terminal ou une chaîne vide ε
- Ces grammaires régulières sont appelées des grammaires linéaires droites.

Définition d'un grammaire régulier

Par analogie, il est possible de définir des grammaires linéaires gauches:

A $\longrightarrow$ B a
ou
A $\longrightarrow$ a

Remarque:
Les grammaires régulières sont une sous-classe des grammaires hors contextes.
Elles permettent de décrire les langages réguliers.

Définition d’un grammaire régulier

Correspondance entre une grammaire régulière et un automate:

Nous pouvons faire la correspondance entre un automate et une grammaire régulière de la manière suivante:

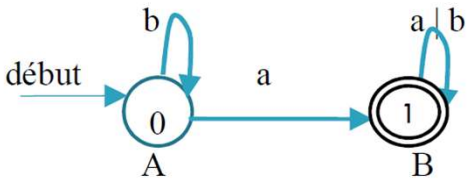
- Chaque état de l'automate correspond à un non terminal de la grammaire.
- Chaque transition correspond à une production de la grammaire.
- L'état initial de l'automate correspond à l'axiome de la grammaire.
- Un état d'acceptation final correspond à la production de la chaine vide ε.

Exemple1 d’un grammaire régulier

Exemple 1: Soit l'expression régulière : (a|b)*a(a|b)*

- Donner l'automate à états finis déterministe qui accepte les mots de cette expression régulière.
- Donner une grammaire régulière équivalente.

AFD



La grammaire régulière:

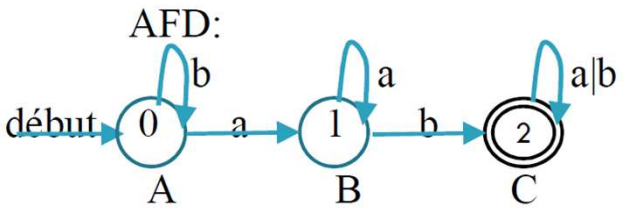
A → b A | a B

B → a B | b B | ε

Exemple2 d’un grammaire régulier

Exemple 2: Soit l'expression régulière : (a|b)*ab(a|b)*

- Donner l'automate à états finis déterministe qui accepte les mots de cette expression régulière.
- Donner une grammaire régulière équivalent.



Grammaire régulière:

A → b A | a B

B → a B | b C

C → a C | b C | ε