Analyse lexicale

Cyril Rabat

cyril.rabat@univ-reims.fr

Licence 3 Informatique - Info0602 - Langages et compilation

2019-2020

Cours n°2

Qu'est-ce qu'un analyseur lexical ?

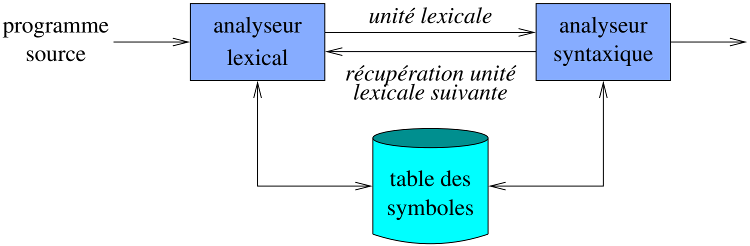
Langages, expressions régulières, automates finis

Version 8 janvier 2020

Table des matières

Un analyseur lexical

- Lecture des caractères en entrée
- Production d'unités lexicales
  ↪ Analysées par l'analyseur syntaxique
- Interaction entre les deux analyseurs



Tâches secondaires

- Élimination...
  - Des commentaires
  - Des caractères "inutiles" (espaces, tabulations, lignes vides...)
- Faciliter la gestion des erreurs :
  - Conservation/calcul du numéro de ligne
  - Associer les messages d'erreur à une ligne
- Principal intérêt de l'analyseur lexical :
  $\hookrightarrow$ Simplification de l'analyseur syntaxique

Unité lexicale, modèle, lexème et attribut

- **Modèle** : règle qui décrit un ensemble de chaînes
- **Unité lexicale** : produite par l'ensemble de chaînes du modèle
  $\hookrightarrow$ Exemple : mots-clés, opérateurs, identificateurs, constantes, chaînes littérales...
- **Lexème** : la suite de caractères du programme source qui correspond au modèle

Exemple

- Unité lexicale : *chiffre*
- Lexèmes : 0, 1, 2
- Modèle : $[0-9]$

- **Attributs** : données liées aux unités lexicales
  $\hookrightarrow$ Exemple : l'entrée dans la table des symboles pour un identificateur

Alphabet et mots

Définition : alphabet

*Un **alphabet** est un ensemble fini de symboles appelés caractères. Il est noté $\mathcal{A}$.*

- Exemples de symboles : lettres et caractères
- Exemples d'alphabets : $\{0, 1\}$ (l'alphabet binaire), l'ASCII

Définition : mot (ou chaîne)

*Un **mot** sur un alphabet est une séquence finie de symboles de cet alphabet. La longueur du mot w (notée $|w|$) est le nombre de symboles dans ce mot. Le mot vide, noté ϵ, est un mot de longueur 0.*

- Exemples de mots sur l'alphabet $\{a, b, c\}$: a, $baba$

Partie de mots

- **Préfixe** de w : mot obtenu en supprimant un nombre quelconque de symboles en fin de w (voire aucun)
- **Suffixe** de w : mot obtenu en supprimant un nombre quelconque de symboles en début de w (voire aucun)
- **Sous-mot** de w : mot obtenu en supprimant un préfixe et un suffixe de w
- **Préfixe propre** de w : tout mot non vide x, préfixe de w tel que $x \neq w$
- Idem pour **suffixe propre** et **sous-chaîne propre** de w
- **Sous-suite** de w : tout mot obtenu en supprimant un nombre quelconque de symboles de w, éventuellement aucun, pas nécessairement consécutifs

Opérations sur les mots

- **Concaténation de mots** : si x et y sont des mots, la concaténation xy est la chaîne formée en joignant x et y
  $\hookrightarrow$ Exemple : pour $\mathcal{A} = \{a, b\}$, si $x = aa$ et y $= bb$, alors $xy = aabb$
- **Exponentiation** : $s^0 = \epsilon$; $s^i = s^{i-1}s$
  $\hookrightarrow$ Exemple : pour $\mathcal{A} = \{a, b\}$, si $x = ba$ alors $x^3 = bababa$

Langage

Définition : langage

Un langage est un ensemble de mots définis sur un même alphabet.

- Soit $\mathcal{A} = \{1, 2, 3\}$, l'ensemble $\{1, 11, 12, 21\}$ est un langage sur $\mathcal{A}$
- Le langage vide est noté $\emptyset$
- Le langage $\{\epsilon\}$ ne contient que le mot vide

Attention

$$\emptyset \neq \{\epsilon\}$$

Opérations sur les langages (1/2)

Soit deux langages L_1 et L_2 définis respectivement sur les alphabets $\mathcal{A}_1$ et $\mathcal{A}_2$.

Définition : union de deux langages

L'union de L_1 et L_2 définie sur $\mathcal{A}_1 \cup \mathcal{A}_2$ est le langage contenant tous les mots de L_1 et L_2 :

$$L_1 \cup L_2 = \{w \mid w \in L_1 \vee w \in L_2\}$$

Définition : intersection de deux langages

L'intersection de L_1 et L_2 définie sur $\mathcal{A}_1 \cap \mathcal{A}_2$ est le langage contenant tous les mots qui sont à la fois dans L_1 et L_2 :

$$L_1 \cap L_2 = \{w \mid w \in L_1 \wedge w \in L_2\}$$

Opérations sur les langages (2/2)

Définition : complément d'un langage

Le complément de L_1 est le langage défini sur $\mathcal{A}_1$ contenant tous les mots qui ne sont pas dans L_1 :

$$\mathcal{C}(L_1) = \{w \mid w \in \mathcal{A}_1 \wedge w \notin L_1\}$$

Définition : différence de deux langages

La différence de L_1 et L_2 est le langage défini sur $\mathcal{A}_1$ contenant tous les mots de L_1 qui ne sont pas dans L_2 :

$$L_1 - L_2 = \{w \mid w \in L_1 \wedge w \notin L_2\}$$

Produit et puissances

Définition : produit de deux langages

*Le produit ou **concaténation** de L_1 et L_2 est le langage défini sur $\mathcal{A}_1 \cup \mathcal{A}_2$ contenant tous les mots formés d'un mot de L_1 suivi d'un mot de L_2 :*

$$L_1.L_2 = \{w_1 w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$$

Définition : puissances d'un langage

Les puissances successives de L_1 définies sur $\mathcal{A}_1$ sont définies récursivement :

- $L_1^0 = \{\epsilon\}$
- $L_1^n = L_1.L_1^{n-1}$ *pour* $n \geq 1$

Fermeture itérative

Définition : fermeture de Kleene de deux langages

La fermeture de Kleene de L_1, appelée également la ***fermeture itérative****, définie sur $\mathcal{A}_1$, est l'ensemble des mots formés par une concaténation finie des mots de L_1 :*

$$L_1^* = \{w \mid \exists k \geq 0 \wedge w_1 \dots w_k \in L_1 \ \textit{tels que} \ w = w_1 w_2 \dots w_k\}$$

- On définit également L_1^+ :
  $L_1^+ = \{w \mid \exists k > 0 \wedge w_1 \dots w_k \in L \text{ tels que } w = w_1 w_2 \dots w_k\}$

Langage fini et infini

Définition : langage fini

Un langage fini peut être décrit par l'énumération des mots qui le compose. Ce qui n'est pas le cas pour un langage infini.

- Certains langages infinis peuvent être décrits à l'aide d'opérations sur des langages simples
- Certains langages infinis peuvent être décrits à l'aide de règles (grammaires)
- Les langages qui ne peuvent être décrits ni par des opérations, ni par des grammaires sont des langages **indécidables**.

Expressions régulières

Définition : expression régulière

Les expressions régulières pour un alphabet $\mathcal{A}$ sont les expressions formées par les règles suivantes :

- *$\emptyset$, ϵ et les symboles de $\mathcal{A}$ sont des expressions régulières*
- *Si α et β sont des expressions régulières sur $\mathcal{A}$, $(\alpha|\beta)$, $(\alpha.\beta)$ et $(\alpha)^*$ sont des expressions régulières*

- On note indifféremment $\alpha.\beta$ et $\alpha\beta$
- On définit une priorité décroissante sur les opérateurs : $*$, . et $|$

Langage décrit par une expression régulière

Définition : langage décrit par une expression régulière

Le langage $L(E)$ où E est une expression régulière définie sur $\mathcal{A}$, est défini comme suit :

- $L(E) = \emptyset$ *si* $E = \emptyset$
- $L(E) = \{\epsilon\}$ *si* $E = \epsilon$
- $L(E) = \{a\}$ *si* $E = a$ *pour tout* $a \in \mathcal{A}$
- $L(E) = L(E_1) \cup L(E_2)$ *si* $E = E_1|E_2$
- $L(E) = L(E_1).L(E_2)$ *si* $E = E_1.E_2$
- $L(E) = L(E)^*$ *si* $E = E_1^*$

Introduction

- Expression régulière : définie un ensemble de mots
- Nécessité de les compiler pour créer un programme qui la reconnait
- But du programme :
    - Entrée : le mot à reconnaître
    - Sortie : oui ou non suivant si le mot est reconnu par l'expression régulière
- Utilisation d'automates

Présentation des automates

Un automate fini se compose :

- D'un **ruban d'entrée** :
  - Constitué d'un ensemble de cases, chacune contenant un caractère
  - Le mot à traiter est placé dans ces cases
  - Une tête de lecture permet de connaître le caractère suivant
- D'un **ensemble d'états** :
  - L'automate passe d'un état à l'autre en cours d'exécution
  - L'**état initial** : état en début d'exécution
  - Les **états d'acceptation** (ou états finals) : états atteints lorsque le mot est accepté
- D'une **fonction de transition** :
  - Indique pour chaque état et symbole lu, le prochain état

Automates finis déterministes vs non déterministes

- Dans un état donné et pour un symbole donné
  $\hookrightarrow$ La fonction de transition indique le prochain état
- Deux possibilités lorsqu'un symbole est rencontré :
  - Une seule transition possible :
    $\hookrightarrow$ Automate déterministe
  - Plusieurs transitions possibles :
    $\hookrightarrow$ L'automate est non déterministe

Remarque

Des définitions plus formelles sont présentées dans la suite !

Automate fini déterministe

Définition : automate fini déterministe

Un automate fini déterministe (noté AFD) est défini formellement par le quintuplet $M = \{Q, \mathcal{A}, \delta, s, F\}$, où :

- *Q est un ensemble fini d'états*
- *$\mathcal{A}$ est un alphabet*
- *δ est la fonction de transition de $Q \times \mathcal{A}$ dans Q*
- *$s \in Q$ est l'état initial*
- *$F \subseteq Q$ est l'ensemble des états d'acceptation*

Représentation d'un automate fini par un graphe

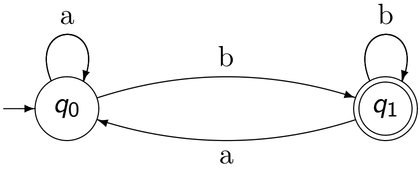
- Chaque état de l'automate est représenté par un nœud du graphe
- Relation de transition représentée par des arcs valués :
  $\hookrightarrow$ Si $\delta(q, a) = q'$, $(q, q') \in Q^2 \wedge a \in \mathcal{A}$ alors il existe un arc a entre les sommets q et q'
- L'état initial est signalé par une flèche
- Les états d'acceptation sont signalés par des doubles cercles

Exemple de représentation

Soit l'automate $M = \{Q, \mathcal{A}, \delta, s, F\}$ suivant :

$\mathcal{A} = \{a, b\}$, $Q = \{q_0, q_1\}$, $s = q_0$, $F = \{q_1\}$ et $\delta =$

q	a	$\delta(q, a)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_0
q_1	b	q_1



Exécution d'un automate

Définition : configuration d'un AFD

Une configuration d'un AFD est une paire composée de l'état de l'automate, ainsi que la partie du mot restant à traiter :

$$(q, w) \in Q \times \mathcal{A}^*$$

- Exécution d'un automate : déterminer les configurations successives de l'automate en fonction du mot d'entrée
- Dérivation : le passage d'une configuration à une autre

Dérivation (1/2)

Définition : dérivation (en une étape)

La configuration (q', w') est dérivable en une étape de (q, w) par M si :

- $w = aw'$ *($a \in \mathcal{A}$, a est le premier caractère de w et w' est égal à w privé de a)*
- $q' = \delta(q, a)$ *(q' est le prochain état déterminé par la fonction de transition δ pour q et a)*

On note $(q, w) \vdash_M (q', w')$

Dérivation (2/2)

Définition : dérivation (en plusieurs étapes)

La configuration (q', w') *est dérivable (en plusieurs étapes) de* (q, w) *s'il existe* $k \geq 0$ *et des configurations* $(q_i, w_i), 0 \leq i \leq k$ *telles que :*

- $(q_0, w_0) = (q, w)$, $(q_k, w_k) = (q', w')$
- $\forall i \in [0, k], (q_i, w_i) \vdash_M (q_{i+1}, w_{i+1})$

On note $(q, w) \vdash_M^* (q', w')$

- Exécution d'un automate (si le mot est accepté) :
$$(s, w) \vdash (q_1, w_1) \vdash \ldots \vdash (q_n, \epsilon)$$
- s est l'état initial, ϵ le mot vide et $n = |w|$

Une seule exécution possible pour chaque mot

Acceptation

Définition : acceptation

Un mot est accepté par un automate M si le dernier état est un état d'acceptation :

$$w \text{ est accepté par } M \text{ si } (s, w) \vdash_M^* (q, \epsilon) \wedge q \in F$$

Le langage accepté par M (noté $L(M)$) est défini par l'ensemble des mots acceptés par M :

$$L(M) = \{w \in \mathcal{A}^* \mid (s, w) \vdash_M^* (q, \epsilon) \wedge q \in F\}$$

Simulation du comportement d'un AFD

Fonction **simulationAFD(**Q**,** $\mathcal{A}$**,** δ**,** s**,** F**,** w**)** : booléen

etat $\leftarrow$ s
position $\leftarrow$ 0
erreur $\leftarrow$ *faux*
Tant que !erreur $\wedge$ position $<$ $|w|$ **Faire**
  Si w[position] $\notin \mathcal{A}$ **Alors**
    erreur $\leftarrow$ *vrai*
  Sinon
    Si $\exists\ \delta(etat, w[position])$ **Alors**
      etat $\leftarrow \delta(etat, w[position])$
      position $\leftarrow$ position $+$ 1
    Sinon
      erreur $\leftarrow$ *vrai*
retourner $\leftarrow$!erreur $\wedge$ etat $\in$ F

Les automates finis non déterministes

Automates où les constructions suivantes sont autorisées :

- Plusieurs transitions sur le même symbole partant d'un même état
- Transitions sur le mot vide acceptées
- Transitions sur des mots de longueur supérieure à 1 sont possibles
  $\hookrightarrow$ Regroupement de transitions

Remarque

Les automates finis non déterministes sont généralement plus faciles à écrire que les AFD mais moins rapides à simuler.

Automate fini non déterministe

Définition : automate fini non déterministe

Un automate fini non déterministe (noté AFN) est défini formellement par le quintuplet $M = \{Q, \mathcal{A}, \Delta, s, F\}$, où :

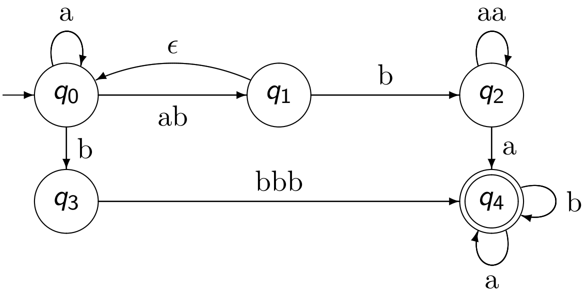
- *Q est un ensemble fini d'états*
- *$\mathcal{A}$ est un alphabet*
- *Δ est la fonction de transition de $Q \times \mathcal{A}^*$ dans Q*
- *$s \in Q$ est l'état initial*
- *$F \subseteq Q$ est l'ensemble des états d'acceptation*

Exemple d'AFN

Soit l'automate $M = \{Q, \mathcal{A}, \delta, s, F\}$ suivant :
$\mathcal{A} = \{a, b\}$, $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $s = q_0$, $F = \{q_4\}$

$\Delta =$

q	u	$\Delta(q,u)$	q	u	$\Delta(q,u)$	q	u	$\Delta(q,u)$
q_0	a	q_0	q_0	ab	q_1	q_0	b	q_3
q_1	ϵ	q_0	q_1	b	q_2	q_2	aa	q_2
q_2	a	q_4	q_3	bbb	q_4	q_4	a	q_4
q_4	b	q_4						



Dérivation d'un AFN

Définition : dérivation (en une étape)

La configuration (q', w') est dérivable en une étape de (q, w) par M si :

- $w = uw'$ *(*$u \in \mathcal{A}^*$*)*
- $(q, u, q') \in \Delta$ *(le triplet* (q, u, q') *est un élément de la relation de transition)*

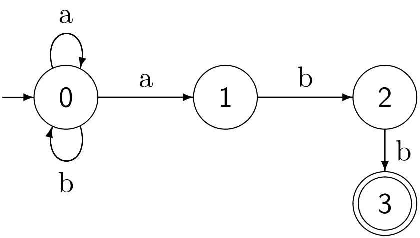
- Acceptation d'un mot par un AFN : il existe une dérivation qui accepte ce mot
  $\hookrightarrow$ Des dérivations possibles peuvent le refuser !
- Construction plus simple que les AFD mais sans forcément reconnaître plus de langages

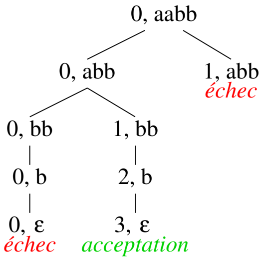
Simulation d'un AFN (1/2)

Problème : plusieurs dérivations possibles pour un même mot !

- Vérifier si l'une d'elles permet d'accepter le mot
- Être en mesure de revenir en arrière (*backtrack*)

$\hookrightarrow$ Utilisation de la récursivité





Simulation d'un AFN (2/2)

Fonction **simulationAFN(**M, $etat$, w, $position$**)** : booléen

- **Si** $position \geq |w|$ **Alors**
  - **retourner** $etat \in M(F)$
- **Sinon**
  - accepte $\leftarrow$ *faux*
  - couples $\leftarrow M(\Delta)(etat, w)$
  - **Tant que** couples $\neq \emptyset \wedge$!accepte **Faire**
    - choisir $(e', l) \in$ couples
    - couples $\leftarrow$ couples $\setminus$ (e', l)
    - accepte $\leftarrow$ simulationAFN(M, e', w, $p + l$)
  - **retourner** accepte

Remarques

M est l'automate et w le mot à reconnaître

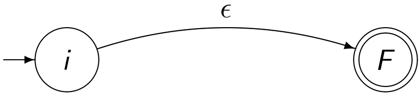
Fonction exécutée avec $etat = s$ et $position = 0$

Construction de Thomson

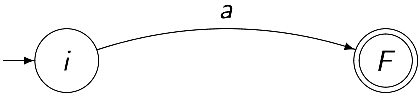
- Objectif : construire de manière automatique un AFN à partir de toute expression régulière
- Idée :
  - Décomposition de l'expression régulière en sous-expressions
  - Construction d'un AFN pour chaque sous-expression
  - Combinaison des AFN

Règles de base

- **Règle n°1** : pour l'expression ϵ, nous construisons l'AFN suivant :



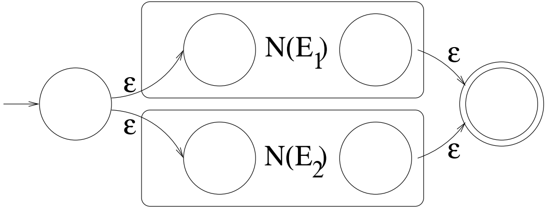
- **Règle n°2** : pour l'expression $a \in \mathcal{A}$, nous construisons l'AFN suivant :



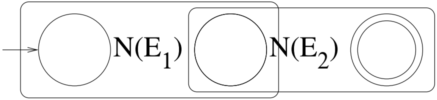
Combinaisons (1/2)

Supposons que $N(E_1)$ et $N(E_2)$ sont des AFN pour les expressions régulières E_1 et E_2.

- **Combinaison n°1** : pour l'expression $E_1|E_2$

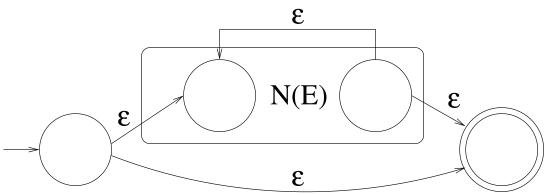


- **Combinaison n°2** : pour l'expression $E_1.E_2$



Combinaisons (2/2)

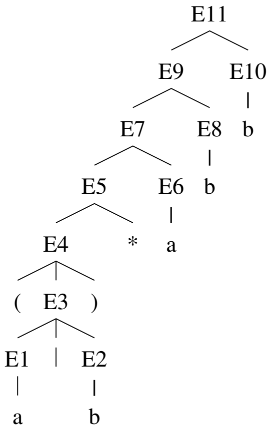
- **Combinaison n°3** : pour l'expression E^*



- **Combinaison n°4** : pour l'expression (E), c'est N(E) lui-même

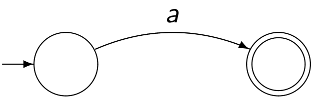
Exemple (1/4)

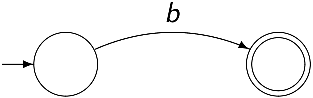
- Supposons l'expression régulière $(a|b)^*abb$
- Première étape : construire l'arbre syntaxique
  $\hookrightarrow$ Construction des AFN pour chaque expression de l'arbre



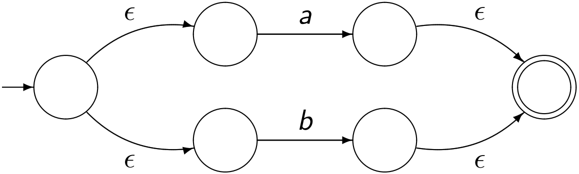
Exemple (2/4)

- Pour $E_1 = E_6 = a$ et $E_2 = E_8 = E_{10} = b$:



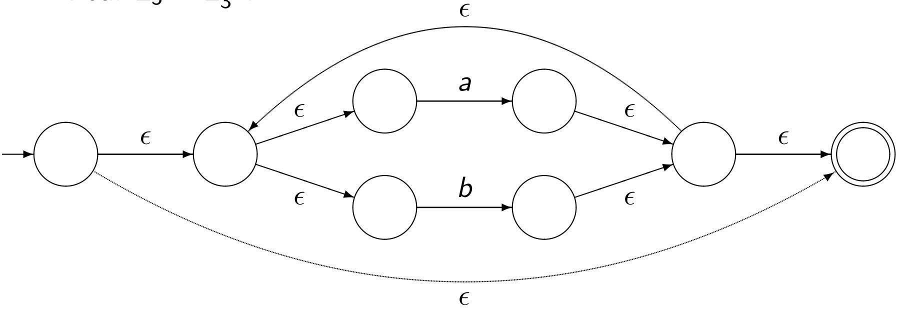


- Pour $E_3 = E_1 | E_2$:



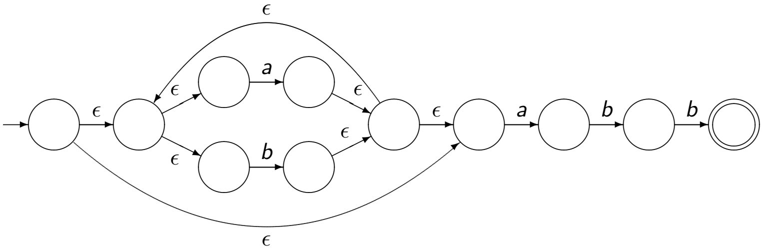
Exemple (3/4)

- Pour $E_4 = (E_3)$, c'est E_3 lui-même
- Pour $E_5 = E_3^*$:



Exemple (4/4)

- Pour finir en combinant avec E_7, E_9 et E_{11}

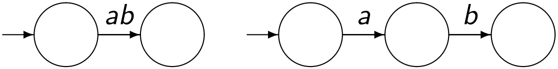


Équivalence entre un AFN et un AFD

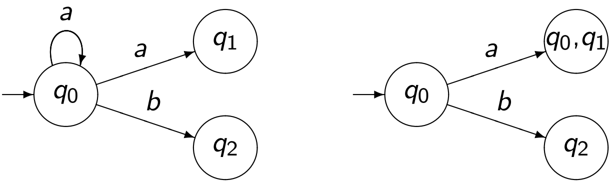
Définition : équivalence entre un AFN et un AFD

Un AFD est équivalent à un AFN s'il accepte le même langage.

- Exemple 1 :



- Exemple 2 :



Définition d'opérations sur un AFN

Définition : ϵ-fermeture d'un état

Soit l'AFN $M = \{Q, \mathcal{A}, \Delta, s, F\}$ et $e \subset Q$, ϵ-fermeture(e) est l'ensemble des états de M accessibles depuis e par des ϵ-transitions uniquement. $e \in \epsilon$-fermeture(e).

Définition : ϵ-fermeture d'un ensemble d'états

Soit l'AFN $M = \{Q, \mathcal{A}, \Delta, s, F\}$ et $T \subset Q$, ϵ-fermeture(T) est l'ensemble des états de M accessibles depuis tout $e \in T$ par des ϵ-transitions uniquement. $T \subseteq \epsilon$-fermeture(T).

Définition : transiter

Soit l'AFN $M = \{Q, \mathcal{A}, \Delta, s, F\}$, $T \subset Q$ et $a \in \mathcal{A}$, transiter(T,a) est l'ensemble des états de M tels qu'il existe une transition sur a à partir d'un $e \in T$. Si transiter(T,a)=T' alors on note $T \xrightarrow{a} T'$

Principes de la transformation

- Soit l'AFN $M = \{Q, \mathcal{A}, \Delta, s, F\}$
- Construction de l'AFD $M' = \{Q', \mathcal{A}, \delta, s', F'\}$
- Chaque état de l'AFD = ensemble d'états de l'AFN
- Construction d'un ensemble V (états à visiter) :
  $\hookrightarrow$ Au départ, $V = \{$ ϵ-fermeture(s) $\}$
- Pour chaque état q de V :
  $\hookrightarrow$ Ajout de q à Q' et retrait de q de V
  $\hookrightarrow$ Pour chaque $a \in \mathcal{A}$, calcul de $q' = \epsilon$-fermeture(transiter(q,a))
  $\hookrightarrow$ Ajout de q' dans V si $q' \notin Q'$
- F est formé par tous les états de M' contenant un $q \in F$

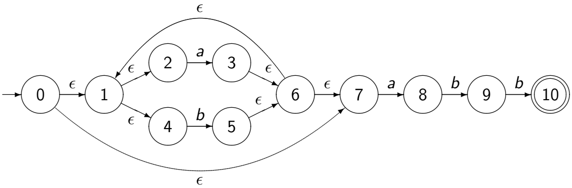
Algorithme de transformation AFN en AFD

Fonction **AFNversAFD(M)** : M'

$s' \leftarrow \epsilon$-fermeture(s)
$Q' \leftarrow \{s'\}; \delta = \emptyset; V \leftarrow \{s'\}$
Tant que $\exists x \in V$ **Faire**
$V \leftarrow V \backslash x$
Pour tout $a \in \mathcal{A}$ **Faire**
$y \leftarrow \epsilon$-fermeture(transiter(x, a)
Si $y \notin Q'$ **Alors**
$Q' \leftarrow Q' \cup y$
$V \leftarrow V \cup y$
$\delta \leftarrow \delta \cup \{(x, y, a)\}$
Fin Pour
$F' \leftarrow \{x \in Q' \mid \exists f \in x \wedge f \in F\}$

Exemple (1/2)

$M((a|b)^*abb)$:



x	transiter	y	δ
s'=A={0,1,2,4,7}	$A \xrightarrow{a} \{3,8\}$	{3,8,6,1,7,2,4} = B	$A \xrightarrow{a} B$
	$A \xrightarrow{b} \{5\}$	{5,6,7,1,2,4} = C	$A \xrightarrow{b} C$
B={1,2,3,4,6,7,8}	$B \xrightarrow{a} \{3,8\}$		$B \xrightarrow{a} B$
	$B \xrightarrow{b} \{5,9\}$	{5,9,6,7,1,2,4}=D	$B \xrightarrow{b} D$
C={1,2,4,5,6,7}	$C \xrightarrow{a} \{3,8\}$		$C \xrightarrow{a} B$
	$C \xrightarrow{b} \{5\}$		$C \xrightarrow{b} C$
D={1,2,4,5,6,7,9}	$D \xrightarrow{a} \{3,8\}$		$D \xrightarrow{a} B$
	$D \xrightarrow{b} \{5,10\}$	{5,6,1,2,4,7,10}=E	$D \xrightarrow{b} E$
E={1,2,4,5,6,7,10}	$E \xrightarrow{a} \{3,8\}$		$E \xrightarrow{a} B$
	$E \xrightarrow{b} \{5\}$		$E \xrightarrow{b} C$

Exemple (2/2)

Sachant que seul E contient un état d'acceptation, $E \in Q'$. On obtient :