Automates à Etats Fini

Automates à états finis;

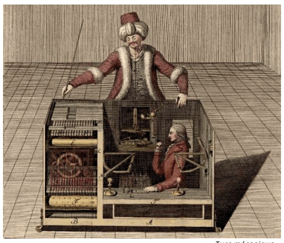
Paradigme déterministe (AFD) et non déterministe (AFN);

Equivalence d'automates;

Algorithme de déterminisation;

Automate AFD canonique

Algorithme de minimisation



Turc mécanique

Auteur : Olivier Raynaud

Version de travail : rentrée 2022

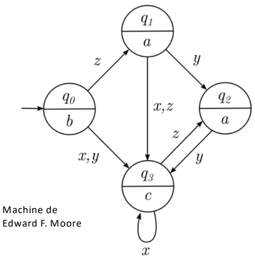
Sensibilité :

Référence :

https://medium.com/idcampus-stories/le-turc-m%C3%A9canique-ou-lart-du-pr%C3%A9totypage-5003f397c3c

1

Automate à états fini



Machine de Edward F. Moore

❑ *Automate*

- Un **automate à états fini** est un modèle mathématique de calcul utilisé pour de nombreuses applications.

Citons la linguistique, la biologie ou l'informatique pour la conception de programmes par exemple.

Définition intuition (automate à états fini déterministe)**:** Un **A.F.D.** est une machine abstraite susceptible d'être dans un nombre fini d'états mais étant à un moment donné dans un seul état appelé état courant. Le passage d'un état à un autre s'appelle une transition.

2

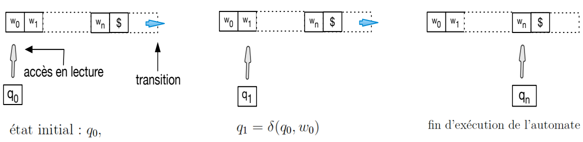
Automate à états fini déterministe - AFD

❑ *L'exécution de l'automate sur un mot d'entrée peut être décrite de la façon suivante :*

- mettre l'automate dans l'état initial q_0, et lire la première lettre w_0 du ruban;
- tant que la dernière lettre w_n de l'entrée n'est pas lue, répéter la transition:

$$q_{i+1} = \delta(q_i, w_i)$$

- si, à la fin de l'exécution, l'état q_n est un état d'accéptation alors le mot w est dit accepté par l'automate.



3

Automate à états fini déterministe – AFD

Définition (Automate à états fini déterministe – AFD) **:**

Formellement, un automate à états fini déterministe est constitué des cinq composants suivants :

(i) Un ensemble fini d'états souvent noté Q ;

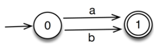
(ii) Un ensemble fini de symboles d'entrée souvent noté Σ ;

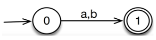
(iii) Une fonction de transition de $Q \times \Sigma \to Q$ souvent noté δ ;

(iv) Un état de départ souvent noté q_0 ;

(v) Un ensemble d'états dit d'acceptation souvent noté F.

Remarque :
Par soucis de simplification, dans un diagramme de transitions, une transition entre deux états peut porter plusieurs symboles.





4

Automate à états fini déterministe - AFD

Diagramme de transitions de A

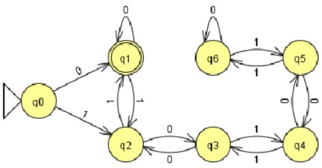


Table de transitions

δ	0	1
q_0	q_1	q_2
q_1	q_1	q_2
q_2	q_3	q_1
q_3	q_2	q_4
q_4	q_5	q_3
q_5	q_4	q_6
q_5	q_6	q_5

Nous notons $A = \{Q, \Sigma, \delta, q_0, F\}$ un A.E.F. où A est le nom de l'automate, Q son ensemble d'états, Σ l'ensemble des symboles d'entrée, q_0 son état initial et F son ensemble d'états d'acceptation.

A DFA (drawn in JFLAP) for the example language

5

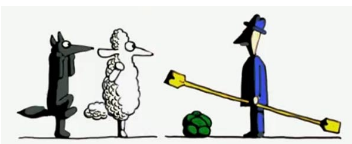
Application

❑ *Problème de passage de rivière*

Problème (le loup, la chèvre et les choux) **:** Un fermier doit passer la rivière dans une barque juste assez grande pour lui et son loup, ou lui et sa chèvre ou encore lui et ses choux.

Les choux seront mangés s'il les laisse seuls avec la chèvre, et la chèvre sera mangée s'il la laisse seule avec le loup.

Comment faire passer la rivière à tout ce petit monde sans dégât?



6

Résolution de problème

- *Problème de passage de rivière - Interprétation*
  - Un automate permet de caratériser l'ensembles des états du système atteints en fonction des actions entreprises.
  - *Un automate n'est donc pas seulement un outil de résolution d'un problème, mais aussi un outil de description de l'ensemble des solutions acceptables pour le résoudre.*
    Il est en cela du même niveau abstrait qu'un algorithme.

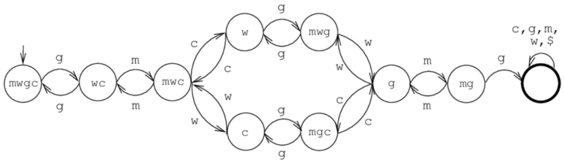


Diagramme de transitions

7

Fonction de transition étendue d'un AFD.

- La fonction de transition étendue est une fonction prenant en entrée un état **q** et une chaîne **w** et retourne un état **p**. **p** est l'état atteint par l'automate en partant de l'état **q** et en lisant la chaîne **w**.

Définition (fonction de transition étendue d'un AFD) :

On définit la fonction $\hat{\delta}$ par induction sur la longueur de la chaîne w de la façon suivante :

- **Base :** $\hat{\delta}(q, \epsilon) = q$;
- **Induction** Supposons que w s'écrive xa avec x une chaîne et a un symbole, nous avons $\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$

Exemple : Soit **w** la chaîne 0010 alors la chaîne **x** vaut 001 et **a** est le symbole 0.

8

Langage des mots contenant un nombre pair de symboles

Soit L le langage $\{w \mid w \textit{ admet un nombre pair de 0 et un nombre pair de 1}\}$

$A = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_0\})$

Table de transition

δ	0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

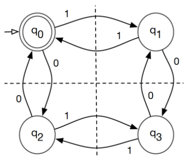


Diagramme de transitions de A

- $\hat{\delta}(q_0, \epsilon) = q_0$
- $\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \epsilon), 1) = \delta(q_0, 1) = q_1$
- $\hat{\delta}(q_0, 11) = \delta(\hat{\delta}(q_0, 1), 1) = \delta(q_1, 1) = q_0$
- $\hat{\delta}(q_0, 110) = \delta(\hat{\delta}(q_0, 11), 0) = \delta(q_0, 0) = q_2$
- $\hat{\delta}(q_0, 1101) = \delta(\hat{\delta}(q_0, 110), 1) = \delta(q_2, 1) = q_3$
- $\hat{\delta}(q_0, 11010) = \delta(\hat{\delta}(q_0, 1101), 0) = \delta(q_3, 0) = q_1$
- $\hat{\delta}(q_0, 110101) = \delta(\hat{\delta}(q_0, 11010), 1) = \delta(q_1, 1) = q_0$

9

Mot accepté par un AFD

- Un mot **w** est accepté si le chemin étiqueté par les symboles de **w** dans l'automate part de l'état initial et termine dans un état final.

Définition (mot accepté par un AFD) **:**
Soit $A = (Q, \Sigma, \delta, q_0, F)$ un automate à états fini déterministe, on dit que A **accepte** un mot w dans Σ^* si $\hat{\delta}(q_0, w) \in F$.

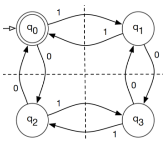


Diagramme de transitions

- $\hat{\delta}(q_0, \epsilon) = q_0$
- $\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \epsilon), 1) = \delta(q_0, 1) = q_1$
- $\hat{\delta}(q_0, 11) = \delta(\hat{\delta}(q_0, 1), 1) = \delta(q_1, 1) = q_0$
- $\hat{\delta}(q_0, 110) = \delta(\hat{\delta}(q_0, 11), 0) = \delta(q_0, 0) = q_2$
- $\hat{\delta}(q_0, 1101) = \delta(\hat{\delta}(q_0, 110), 1) = \delta(q_2, 1) = q_3$
- $\hat{\delta}(q_0, 11010) = \delta(\hat{\delta}(q_0, 1101), 0) = \delta(q_3, 0) = q_1$
- $\hat{\delta}(q_0, 110101) = \delta(\hat{\delta}(q_0, 11010), 1) = \delta(q_1, 1) = q_0$

10

Langage reconnu par un AFD

- Le langage reconnu par un automate est le langage composé de tous les mots qu'il accepte.

Définition (langage reconnu par un AFD) **:**
Soit $A = (Q, \Sigma, \delta, q_0, F)$ un automate à états fini déterministe, on note $L(A)$ le langage **reconnu** par A avec $L(A) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$.

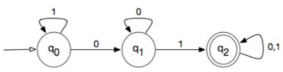


Diagramme de transitions

Table de transition

δ	0	1
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_2	q_2

Cet automate **reconnaît** le langage des mots contenant le facteur 01.

11

Automate à états fini non déterministe – AFN

Définition (automate à états fini non déterministe – AFN) **:**
Formellement, un **automate à états fini non déterministe** est constitué des cinq composants suivants :

(i) Un ensemble fini d'états Q ;
(ii) Un ensemble fini de symboles d'entrée noté Σ ;
(iii) Une fonction de transition δ de $Q \times \Sigma \to 2^Q$;
(iv) Un état de départ q_0 ;
(v) Un ensemble d'états dit d'acceptation F.

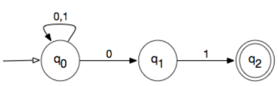


Diagramme de transitions d'un AFN

δ	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\emptyset$	$\emptyset$

12

Fonction de transitions étendue d'un AFN

❑ *Fonction de transition étendue*

➢ La fonction de transition étendue est une fonction prenant en entrée un état **q** et une chaîne **w** et retourne l'ensemble des états que l'on atteint en suivant la chaîne **w**.

Définition (fonction de transition étendue d'un AFN) :

On définit la fonction $\hat{\delta}$ par induction sur la longueur de la chaîne w de la façon suivante :

- **Base :** $\hat{\delta}(q, \epsilon) = \{q\}$
- **Induction** Supposons que w s'écrive xa avec x une chaine et a un symbole, supposons aussi que $\hat{\delta}(q, x) = \{p_1, p_2, ..., p_k\}$.
  Soit $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, ..., r_m\}$, alors nous avons $\hat{\delta}(q, w) = \{r_1, r_2, ..., r_m\}$

13

Exemple de fonction de transitions étendue

Langage des mots terminant par le facteur 01

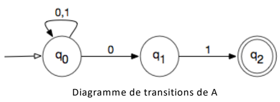


Diagramme de transitions de A

Table de transition

δ	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\emptyset$	$\emptyset$

- $\hat{\delta}(q_0, \epsilon) = \{q_0\}$
- $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$
- $\hat{\delta}(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

14

Langage reconnu par un AFN

➢ Un mot **w** est accepté par un AFN s'il existe une suite de choix permettant de terminer la lecture de **w** sur un état d'acceptation.

Définition (langage reconnu par un AFN) **:**

Soit $A = (Q, \Sigma, \delta, q_0, F)$ un automate à états fini non déterministe, on note $L(A)$ le langage **reconnu** par A avec $L(A) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$.

Table de transitions

δ	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\emptyset$	$\emptyset$

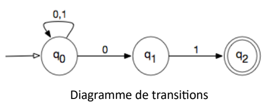


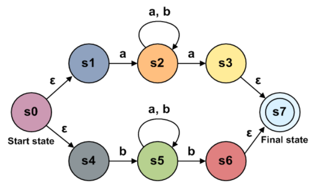
Diagramme de transitions

Cet automate **reconnaît** le langage des mots contenant le facteur 01.

15

Equivalence d'automates

Définition (équivalence) **:** Soient A et A' deux automates à états fini, on dit que A est **équivalent** à A', et on note A ~ A' si L(a) est égal à L(A').



https://www.leniel.net/2009/02/regex-engine-in-csharp-regexparser.html

16

Equivalence de langage

❑ *Exemples*

Diagramme de transitions de A

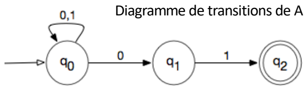


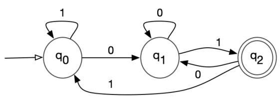
Table de transitions de A

δ	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\emptyset$	$\emptyset$

Table de transitions de A'

δ	0	1
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_0

Diagramme de transitions de A'



Deux automates différents qui reconnaissent le même langage.
Les deux automates A et A' sont **équivalents**.

17

Construction par sous-ensembles

➢ Pour tout A.F.N. nous pouvons appliquer une méthode appelée *construction par sous-ensembles* permettant de construire un A.F.D. reconnaissant le même langage.

Définition (construction par sous-ensembles) **:**

Soit $A = \{Q_N, \Sigma, \delta_N, q_0, F_N\}$ un A.F.N., la construction par sous-ensembles est un processus de calcul d'un A.F.D. $A' = \{Q_D, \Sigma, \delta_D, \{q_0\}, F_D\}$ tel que $L(A) = L(A')$.

En particulier nous avons :

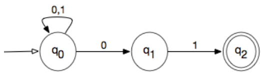
- Q_D est l'ensemble des sous ensembles de Q_N
- F_D est l'ensemble des sous-ensembles S de Q_N tel que $S \cap F_N \neq \emptyset$
- Pour chaque ensemble $S \subseteq Q_N$ et pour chaque symbol a de Σ nous avons :

$$\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$$

18

Exemple de construction

Cet automate **reconnaît** le langage des mots terminant le facteur 01.

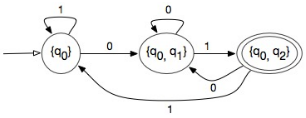


Automate AFN A

Table de transitions

δ_N	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\emptyset$

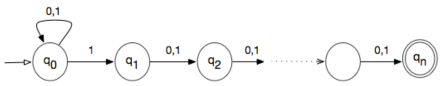
δ_D	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$*\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$*\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



Automate AFD A'
tel que **L(A) = L(A')**

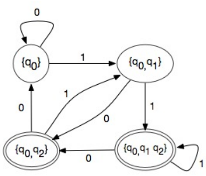
19

Cas difficile

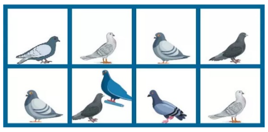


$$s_1 = a_1a_2...a_n \rightarrow \mathbf{1}a_2...a_n$$
$$s_2 = b_1b_2....b_n \rightarrow \mathbf{0}b_2....b_n$$

$$s_1 = a_1a_2...a_n \rightarrow a_1a_2...\mathbf{a_i}....a_n.0...0$$
$$s_2 = b_1b_2....b_n \rightarrow b_1b_2...\mathbf{b_i}.....b_n.0...0$$



AFD pour n=2



https://math24.net/pigeonhole-principle.html

20

Algorithme de déterminisation

Algorithme 1: *determinisation*()

Données : $A_N = \{Q_N, \Sigma, \delta_N, q_0, F_N\}$ un automate non déterministe
Résultat : $A_D = \{Q_D, \Sigma, \delta_D, q_0, F_D\}$ un automate déterministe
début
- $Q_0 \leftarrow \{q_0\}$, $aTraiter \leftarrow Q_0$, $Q_D \leftarrow \{Q_0\}$;
- **tant que** *($aTraiter \neq \emptyset$)* **faire**
  - Choisir $Q \in aTraiter$; $aTraiter \leftarrow aTraiter - Q$;
  - **pour** *(chaque caractère $c \in \Sigma$)* **faire**
    - **pour** *(chaque état $q \in Q$)* **faire**
      - $\delta_D(Q, c) \leftarrow \delta_D(Q, c) \cup \delta_N(q, c)$;
      - **si** $(\delta_D(Q, c) \notin Q_D)$ **alors**
        - $Q_D \leftarrow Q_D \cup \{\delta_D(Q, c)\}$; $aTraiter \leftarrow aTraiter \cup \{\delta_D(Q, c)\}$
- **pour** *(chaque état $Q \in Q_D$)* **faire**
  - **si** $(Q \cap F_N)$ **alors**
    - $F_D \leftarrow F_D \cup Q$;
- **retourner** A_D ;

fin

21

Théorème d'équivalence

Théorème (construction par sous-ensembles) **:**
Soit $N = \{Q_N, \Sigma, \delta_N, q_0, F_N\}$ un AFN, et $D = \{Q_D, \Sigma, \delta_D, \{q_0\}, F_D\}$ un AFD construit par la méthode dite des sous-ensembles alors on a $L(D) = L(N)$.

Théorème (équivalence AFD et AFN) **:**
Un langage L est accepté par un AFN **si et seulement si** le langage L est accepté par un AFD.

22

AFN avec ε-transition

- Nous pouvons rajouter au modèle des AFN la possiblité de transitions portant le symbole ε. Dans un AFN-ε, il est autorisé de suivre spontannément une transition portant le symbol ε sans recevoir de symbol d'entrée.

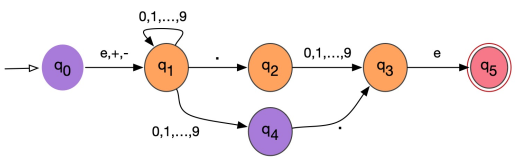
Définition (AFN-ε) **:**
Formellement, un **automate à états fini non déterministe** avec $\epsilon-transition$ est composé des cinq composants suivants :

(i) Un ensemble fini d'états Q ;
(ii) Un ensemble fini de symboles d'entrée noté Σ ;
(iii) Une fonction de transition δ de $Q \times (\Sigma \cup \epsilon) \to 2^Q$;
(iv) Un état de départ q_0 ;
(v) Un ensemble d'états dit d'acceptation F.

23

AFN avec ε-transition

AFN-ε reconnaissant les nombres décimaux



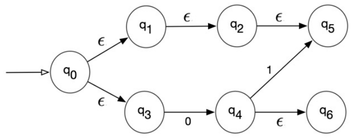
$$E = (\{q_0, q_1, ..., q_5\}, \{., +, -, 1, 2, , ..., 9\}, \delta, q_0, \{q_5\})$$

δ	ϵ	$+, -$	$.$	$0, 1, ..., 9$
q_0	$\{q_1\}$	$\{q_1\}$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	$\{q_2\}$	$\{q_1, q_4\}$
q_2	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_3	$\{q_5\}$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_4	$\emptyset$	$\emptyset$	$\{q_3\}$	$\emptyset$
q_5	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

24

ϵ-fermeture

Définition (ε-fermeture d'un état)**:**
Soit $A = \{Q_N, \Sigma, \delta_N, q_0, F_N\}$ un AFN-ϵ, on appelle ϵ-fermeture d'un état $q \in Q_N$, l'ensemble des états $q_i \in Q_N$ atteignables à partir de q par un chemin composé uniquement de transitions étiquetées par ϵ.

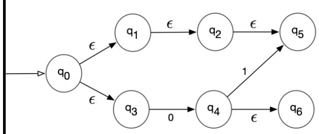


$$\epsilon - fermeture(q_0) = \{q_0, q_1, q_2, q_3, q_5\}$$
$$\epsilon - fermeture(q_4) = \{q_4, q_6\}$$

25

ε-fermeture

Définition (ε-fermeture d'un ensemble d'états)**:**
Soit $A = \{Q_N, \Sigma, \delta_N, q_0, F_N\}$ un AFN-ϵ, on appelle ϵ-fermeture d'un ensemble d'états $Q \subseteq Q_N$, l'union des ϵ-fermetures des états de Q.



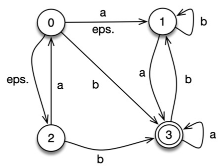
$$\epsilon - fermeture(q_0) = \{q_0, q_1, q_2, q_3, q_5\}$$
$$\epsilon - fermeture(q_4) = \{q_4, q_6\}$$

$$\epsilon - fermeture(\{q_0, q_4\}) = \epsilon - fermeture(q_0) \ \cup \ \epsilon - fermeture(q_4)$$

$$\epsilon - fermeture(\{q_0, q_4\}) = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$$

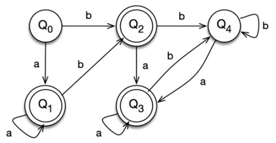
26

Déterminisation pour AFN-ε



Liste des états	ϵ-fermeture
ϵ-fermeture(q_0)	$\{q_0, q_1, q_2\}$
ϵ-fermeture(q_1)	$\{q_1\}$
ϵ-fermeture(q_2)	$\{q_2\}$
ϵ-fermeture(q_3)	$\{q_3\}$

δ'	a	b
$Q_0 = \{q_0, q_1, q_2\}$	Q_1	Q_2
$Q_1 = \{q_0, q_1, q_2, q_3\}$	Q_1	Q_2
$Q_2 = \{q_1, q_3\}$	Q_3	Q_4
$Q_3 = \{q_3\}$	Q_3	Q_4
$Q_4 = \{q_1\}$	Q_3	Q_4

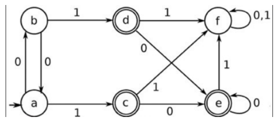


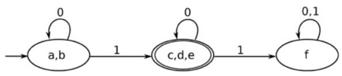
27

AFD minimum

Théorème (AFD canonique) **:**
Soit *L* un langage rationnel, il existe un automate à états fini déterministe minimum qui reconnait L. Cet automate dit **canonique** admet un nombre minimum d'états et est unique.

[MN 74, HMU 1979]



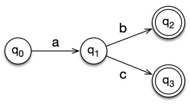


28

Minimisation d'un AFD

➢ Minimisation par le principe algorithmique d'éclatement de partition

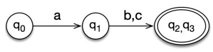
Principe algorithmique d'éclatement (ou affinage) **de partition:**
- Le processus est initialisé à partir d'une ou plusieurs classes
- On applique un critère qui permet de partitionner une classe en plusieurs classes plus restreintes.
- Le processus se termine lorsque aucune des classes de la partition ne peux plus être partitionner.



$$\delta(q_0, b) = q_p \in C_1$$
$$\delta(q_1, b) = q_2 \in C_2$$
$$\delta(q_p, b) = q_p \in C_1$$

$$\mathcal{P} = \{\{q_0, q_p\}, \{q_1\}, \{q_2, q_3\}\}$$

$$\mathcal{P} = \{\{q_0, q_1, q_p\}, \{q_2, q_3\}\}$$



$C_1 = \{q_0, q_1, q_p\}$ $\quad C_2 = \{q_2, q_3\}$

29

Algorithme de minimisation

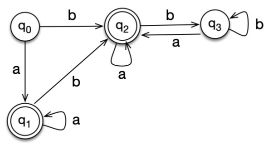
Algorithme 2: *minimisation*()

Données : $A_D = \{Q_D, \Sigma, \delta_D, q_0, F_D\}$ un automate déterministe
Résultat : $A_m = \{Q_m, \Sigma, \delta_m, Q_0, F_m\}$ un automate déterministe équivalent minimum
début
- $\mathcal{P} \leftarrow \{Q_D \setminus F_D, F_D\}$, $b \leftarrow 1$;
- Supprimer de A tous les états non atteignables ;
- **tant que** *($b = 1$)* **faire**
  - $b \leftarrow 0$;
  - **pour** *(chaque classe C de $\mathcal{P}$)* **faire**
    - **pour** *(chaque caractère x de Σ)* **faire**
      - Identifier la classe d'appartenance de $\delta_D(x)$;
    - **si** *(pour différents x il existe plusieurs classes d'appartenance)* **alors**
      - Partitionner C dans $\mathcal{C}$ par les classes obtenues ;
      - $b \leftarrow 1$;
- $\delta_m \leftarrow$ fonction de passage d'une classe à une autre ;
- $F_m \leftarrow$ ensemble des classes de $\mathcal{P}$ contenant au moins un état de F_D ;
- **retourner** A_m ;

fin

30

Mise en application



$$\mathcal{P} = \{\{q_0, q_3, q_p\}, \{q_1, q_2\}\}$$

31

Pour résumer

Quelques points essentiels de l'exposé

- **Définitions :** Automate à états fini (AFD), fonction de transition, mot et langage accepté par un AFD.
- **Définition :** Automate à états fini non déterministes (AFN), fonctions de transition, mot et langage reconnus par un AFN;
- **Définition :** Equivalence d'automate;
- **Algorithme** : Déterminisation d'un AFN par la méthode des sous-ensembles.
- **Définition :** AFN avec epsilon-transition
- **Théorème** : Existence d'un automate (AFD) canonique
- **Algorithme :** Algorithme de minimisation par affinage de partition

32

Bibliographie

- [HMU] **Introduction to Automata Theory, Langage and Computation**
  J.E. Hopcroft, R. Motwani, J.D. Ullman Edition Adison Wesley 2001;
- [Berry 00] **Support de cours de Théorie des Langages**
  A. Berry – Université Clermont-Auvergne 2000 – 2016
- [Wikipedia] **Multiples références dans le texte.**

33