



# INF 302 : LANGAGES & AUTOMATES

## Chapitre 5 : Automates à États Finis Déterministes

### — Distinguabilité, Équivalence, Minimisation

Yliès Falcone

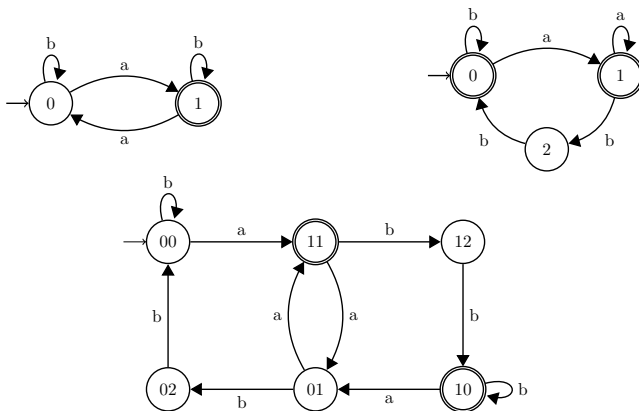
[ylies.falcone@univ-grenoble-alpes.fr](mailto:ylies.falcone@univ-grenoble-alpes.fr) — [www.ylies.fr](http://www.ylies.fr)

Univ. Grenoble-Alpes, Inria

Laboratoire d'Informatique de Grenoble - [www.liglab.fr](http://www.liglab.fr)  
Équipe de recherche LIG-Inria, CORSE - [team.inria.fr/corse/](http://team.inria.fr/corse/)

Année Académique 2018 - 2019

# Intuition et objectifs

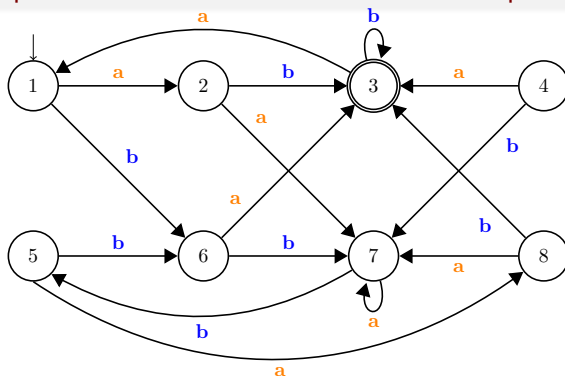


- Ingrédients de base : états (accepteurs), symboles, transitions — *syntaxe*.
- Exécution, mot accepté, langage accepté — *sémantique*.
- Problèmes de décision : langage vide, langage infini.
- Opérations sur automate/opérations sur langage : négation/complémentation, produit/intersection.

## Plan Chap. 5 - AEFD - Distinguabilité, Équivalence, Minimisation

- 1 Tester l'équivalence entre états
- 2 Tester l'équivalence entre automates
- 3 Minimisation d'automates à états finis déterministes
- 4 Résumé

# Équivalence et Minimisation : motivations par un exemple



## Questions

- Quels états peuvent être "distingués" ?
- Quels états sont "équivalents" ?

De manière plus générale :

- Peut-on définir une équivalence entre états ?
- Peut-on dire si des automates sont équivalents ?
- Peut-on avoir une représentation « canonique » (on dira minimale) d'un automate ?

Équivalence/distinguabilité  
sont reliées  
à la notion d'**acceptation**.

Nous considérons  $A = (Q, \Sigma, \delta, q_{\text{init}}, F)$  un AEFD dont tous les états sont accessibles.

## Plan Chap. 5 - AEFD - Distinguabilité, Équivalence, Minimisation

- 1 Tester l'équivalence entre états
- 2 Tester l'équivalence entre automates
- 3 Minimisation d'automates à états finis déterministes
- 4 Résumé

# Distinguabilité entre états

## Définition et exemple

"Deux états sont distinguables s'il existe un mot qui, à partir de l'un des états, mène à un état accepteur, et à partir de l'autre état, mène à un état non accepteur."

### Définition (Relation de distinguabilité entre états)

La relation de distinguabilité  $\neq$  entre états sur  $Q$  est définie par :

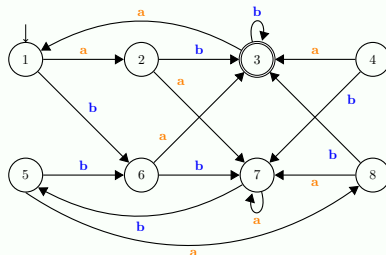
$$\forall p, q \in Q : \left( p \neq q \text{ ssi } \exists u \in \Sigma^* : (\delta^*(p, u) \in F \not\equiv \delta^*(q, u) \in F) \right)$$

*critère unique transition(s) vers états  $\in F$  mais  $\neq$*

Deux états non-distinguables sont dits équivalents (relation  $\equiv$ ).

### Exemple (États (non) distinguables)

- distinguables :  $1 \neq 2$ ,  $1 \neq 3$ ,  $1 \neq 4$ ,  
 $1 \neq 6$ ,  $2 \neq 3$ , ...
- équivalents :  $4 \equiv 6$ ,  $2 \equiv 8$ , mais aussi  
 $1 \equiv 5$  et  $x \equiv x$  pour tout état  $x$ .



# Distinguabilité entre états

## Propriétés de la relation

$$p \neq q \text{ ssi } \exists u \in \Sigma^* : (\delta^*(p, u) \in F \not\iff \delta^*(q, u) \in F)$$

### Théorème : à propos de la relation de distinguabilité

La relation de distinguabilité  $\neq$  entre états de  $Q$  est :

- anti-réflexive :  $\forall q \in Q : \neg(q \neq q)$ ,
- symétrique :  $\forall p, q \in Q : p \neq q \implies q \neq p$ .

# Distinguabilité entre états à $k$ pas

## Question

Comment calculer  $\neq$  ?

- Limiter la relation de distinguabilité à  $k$  symboles.
- Calculer  $\neq$  de manière itérative.

## Définition (Distinguabilité à $k$ pas)

Pour chaque  $k \in \mathbb{N}$ , on introduit la relation  $\neq_k$  sur  $Q$  :

- 1  $q \neq_0 q'$  ssi  $q \in F \nleftrightarrow q' \in F$ .
- 2 Pour  $k \in \mathbb{N}$ ,  $p \neq_{k+1} q$  ssi

$$(p \neq_k q) \vee (\exists a \in \Sigma : \delta(p, a) \neq_k \delta(q, a)).$$



Construction de  $\neq$  à partir de  $\neq_k$ 

## Lemme

Pour tout  $k \in \mathbb{N}$ ,  $q \neq_k q'$  ssi

$$\exists u \in \Sigma^* : |u| \leq k \wedge (\delta^*(q, u) \in F \not\Rightarrow \delta^*(q', u) \in F).$$

## Corollaire

$$\bigcup_{k \in \mathbb{N}} \neq_k = \neq$$

Nous pouvons en déduire un algorithme de calcul des états distinguables.

**Sortie :**  $D \subseteq Q \times Q$  relation de distinguabilité entre états de  $Q$

1: **ensemble de couples d'états**  $D, D_{pre}$  ; (\*  $D$  contient les couples d'états distinguables \*)  
 (\*  $D_{pre}$  est la valeur de  $D$  à l'itération précédente \*)

2: **ensemble de couples d'états**  $X$  ;

3:  $D := (F \times (Q \setminus F)) \cup (Q \setminus F \times F)$ ; (\*  $D$  initialisé avec états accepteurs/non accepteurs \*)

4:  $D_{pre} := \emptyset$ ; (\* maj de  $D_{pre}$  \*)

5: **tant que**  $D_{pre} \neq D$  **faire**

6:  $D_{pre} := D;$

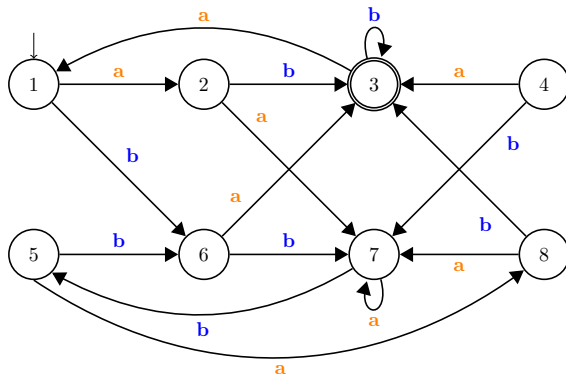
7:  $X := \{(p, q), (q, p) \in Q \times Q \mid \exists a \in \Sigma : (\delta(p, a), \delta(q, a)) \in D\}$ ;  
 (\* calcul des nouveaux états distinguables \*)

8:  $D := D \cup X;$  (\* ajouter les états distinguables à  $D$  \*)

9: fin tant que

```
10: retourner  $D$ ;
```

# Distinguabilité entre états : exemple



2	x						
3	x	x					
4	x	x	x				
5		x	x	x			
6	x	x	x		x		
7	x	x	x	x	x	x	
8	x		x	x	x	x	x
	1	2	3	4	5	6	7

## Distinction entre états : correction de l'algorithme

Soit  $A = (Q, \Sigma, \delta, q_{\text{init}}, F)$  un AEFD dont tous les états sont accessibles.

### Théorème : Correction de l'algorithme de distinction entre états

- L'algorithme distingue *uniquement* des états distinguables.
  - L'algorithme distingue *tous* les états distinguables.
- 
- Pour le premier point, il suffit de montrer que l'algorithme calcule la limite de la suite  $(D_i)_{i \in \mathbb{N}}$  définie comme suit :
    - $D_0 = F \times (Q \setminus F) \cup (Q \setminus F) \times F$  ;
    - $X_{n+1} = \{(p, q), (q, p) \mid \exists a \in \Sigma : (\delta(p, a), \delta(q, a)) \in D_n\}$  ;
    - $D_{n+1} = D_n \cup X_{n+1}$ .
  - Pour le second point, nous faisons une preuve par l'absurde. La démonstration utilise  $\delta^*$ , la fonction de transition  $\delta$  étendue aux mots.

# Distinction entre états : correction de l'algorithme – preuve

## Démonstration.

Supposons que le théorème soit faux (cad, il y a un automate contre-exemple). Alors, il existe au moins une "mauvaise paire" d'états  $\{p, q\}$  t.q. :

- $p$  et  $q$  sont distinguables : il existe  $w \in \Sigma^*$  tel que soit  $\delta^*(p, w) \in F$  soit  $\delta^*(q, w) \in F$  (ou exclusif),
- l'algorithme ne distingue pas ces états.

Soit  $w = a_1 \cdot a_2 \cdots a_n$  le plus court mot distinguant une mauvaise paire  $\{p, q\}$

- $w \neq \epsilon$  d'après l'initialisation de l'algorithme (ligne 5)
- soient  $p' = \delta(p, a_1)$  et  $q' = \delta(q, a_1)$ 
  - $p'$  et  $q'$  sont distingués par  $a_2 \cdots a_n$  car  $\delta^*(p', a_2 \cdots a_n) = \delta^*(p, w)$  et  $\delta^*(q', a_2 \cdots a_n) = \delta^*(q, w)$
  - $a_2 \cdots a_n$  est plus court que n'importe quel mot distinguant une mauvaise paire
  - $\{p', q'\}$  ne peut pas être une mauvaise paire
- l'algorithme déclarera donc  $\{p', q'\}$  comme distinguables
- d'après le corps de la boucle de l'algorithme, dans le pire des cas, à l'itération suivante,  $\{p, q\}$  sera marquée.



# Plan Chap. 5 - AEFD - Distinguabilité, Équivalence, Minimisation

- 1 Tester l'équivalence entre états
- 2 Tester l'équivalence entre automates
- 3 Minimisation d'automates à états finis déterministes
- 4 Résumé

# Tester l'équivalence entre deux automates

Considérons deux AEFDs *complets* :

- $A = (Q^A, \Sigma, q_{\text{init}}^A, \delta^A, F^A)$ ,
- $B = (Q^B, \Sigma, q_{\text{init}}^B, \delta^B, F^B)$ .

## Question

Comment savoir si  $A$  et  $B$  acceptent le même langage ?

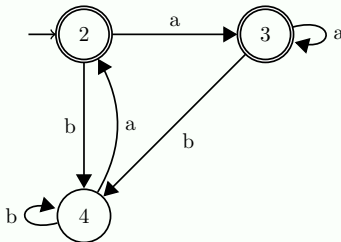
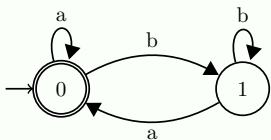
## Procédure pour tester l'équivalence entre deux automates

- 1 Construire l'automate  $E = (Q^A \cup Q^B, \Sigma, q_{\text{init}}^A, \delta^A \cup \delta^B, F^A \cup F^B)$ .
- 2 Tester si  $q_{\text{init}}^A$  et  $q_{\text{init}}^B$  sont distinguables dans  $E$ .

# Tester l'équivalence entre deux automates

## Exemple

### Exemple (Deux automates équivalents)



1	x			
2		x		
3		x		
4	x		x	x
	0	1	2	3



## Plan Chap. 5 - AEFD - Distinguabilité, Équivalence, Minimisation

- 1 Tester l'équivalence entre états
- 2 Tester l'équivalence entre automates
- 3 Minimisation d'automates à états finis déterministes
- 4 Résumé

## Équivalence entre états

Soit  $A = (Q, \Sigma, \delta, q_{\text{init}}, F)$  un AEFD dont tous les états sont accessibles

### Définition (Relation d'équivalence entre états)

La relation d'équivalence  $\approx$  entre états sur  $Q$  est définie par :

$$\forall p, q \in Q : \quad p \approx q \quad \text{ssi} \quad \forall u \in \Sigma^* : (\delta^*(p, u) \in F \iff \delta^*(q, u) \in F)$$

$\approx$  est effectivement une relation d'équivalence. C'est une relation :

- réflexive,
- symétrique,
- transitive.

Notations :

- $[q]$  : la classe d'équivalence de l'état  $q$
- $Q_{/\approx}$  : l'ensemble des classes d'équivalence (dans un automate avec ensemble d'états  $Q$ ).

### Équivalence et distinguabilité sont duales

Deux états sont équivalents si et seulement s'ils ne sont pas distinguables.

## Minimisation : automate minimisé et équivalence

Soit  $A = (Q, \Sigma, \delta, q_{\text{init}}, F)$  un AEFD complet dont tous les états sont accessibles.

### Définition (Minimisé d'un automate)

Le minimisé de  $A$  est l'automate  $A_{/\approx} = (Q_{/\approx}, \Sigma, [q_{\text{init}}], \delta_{/\approx}, F_{/\approx})$  où :

- $\delta_{/\approx}$  est l'application de transition t.q. :

$$\begin{array}{ccc} \delta_{/\approx} & : & Q_{/\approx} \times \Sigma \rightarrow Q_{/\approx} \\ \delta_{/\approx}([q], a) & \stackrel{\text{def}}{=} & [\delta(q, a)] \end{array}$$

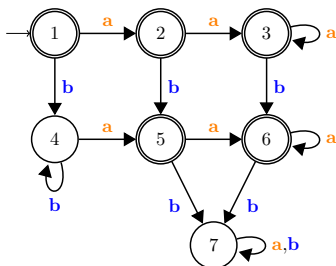
- $F_{/\approx} = \{[q] \mid q \in F\}$ .

### Théorème

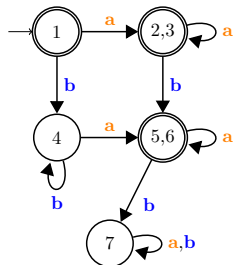
Étant donnés  $A$  et son minimisé  $A_{/\approx}$  :

- 1  $L(A_{/\approx}) = L(A)$  ;
- 2  $A_{/\approx}$  est minimal pour  $L(A)$  : il n'existe pas d'AEFD complet qui reconnait  $L(A)$  et contient moins d'états que  $A_{/\approx}$

# Minimisation d'automate : exemple



$\equiv_0$	$\equiv_1$	$\equiv_2$	$\equiv_3$
1	2	2	2
2	3	3	3
3	1	1	1
5	5	5	5
6	6	6	6
4	4	4	4
7	7	7	7



## Pourquoi l'algorithme de minimisation est optimal

Soit  $A = (Q, \Sigma, \delta, q_{\text{init}}, F)$  un AEFD complet dont tous les états sont accessibles.

Soit  $M$  l'automate minimisé par l'algorithme de minimisation.

Supposons qu'il existe un automate minimisé  $N$  qui accepte le même langage que  $A$  mais avec moins d'états que  $M$ .

- Appliquer la procédure pour tester l'équivalence entre automates sur  $M$  et  $N$ .
- Les états initiaux de  $M$  et  $N$  sont indistinguables car  $L(M) = L(N)$ .
- Remarquer que si  $p$  et  $q$  sont indistinguables alors tous les successeurs sur n'importe quel symbole sont indistinguables (sinon  $p$  et  $q$  seraient distinguables).
- Tous les états de  $M$  sont indistinguables d'au moins un état de  $N$ .
  - prenons  $p$  de  $M$ , il existe  $w \in \Sigma^*$  depuis l'état initial de  $M$  vers  $p$
  - par  $w$  nous pouvons atteindre un état de  $N$  depuis son état initial
  - par induction,  $p$  et l'état atteint dans  $N$  par  $w$  sont indistinguables
- Comme  $N$  a moins d'états que  $M$ , il y a deux états de  $M$  qui sont indistinguables du même état dans  $N$ .
- Par transitivité de la relation d'indistinguabilité, ces deux états sont indistinguables l'un de l'autre.
- Contradiction :  $M$  a été conçu tel que tous ses états sont distinguables.

## Plan Chap. 5 - AEFD - Distinguabilité, Équivalence, Minimisation

- 1 Tester l'équivalence entre états
- 2 Tester l'équivalence entre automates
- 3 Minimisation d'automates à états finis déterministes
- 4 **Résumé**

## Résumé du Chapitre 5 : *Équivalence et Minimisation* d'Automate à États Fini Déterministes

### *Équivalence et Minimisation* d'Automate à États Fini Déterministes

- équivalence entre états,
- équivalence entre automates,
- minimisation d'automate.

### Pour le prochain cours

- Déterminer pourquoi les algos de calculs des relations d'états distinguables et équivalents terminent.
- (Chapitre précédent) Définir des procédures permettant de calculer des automate reconnaissant l'union et le xor des langages d'automates passés en paramètres.