



## INF 302 : LANGAGES & AUTOMATES

### Chapitre 6 : Automates à États Finis Déterministes

— Distinguabilité, Équivalence, Minimisation

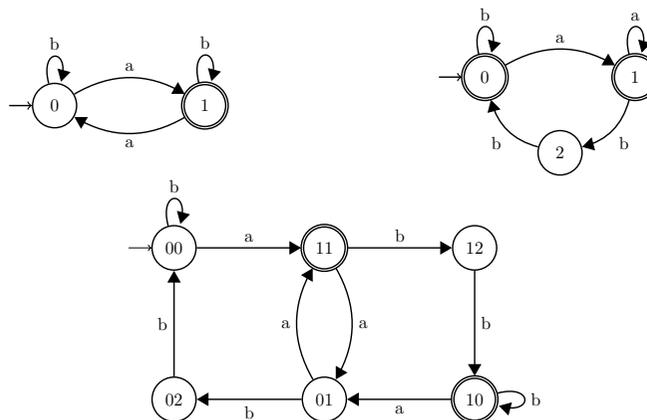
Yliès Falcone

[yliès.falcone@univ-grenoble-alpes.fr](mailto:yliès.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/)

## Intuition et objectifs

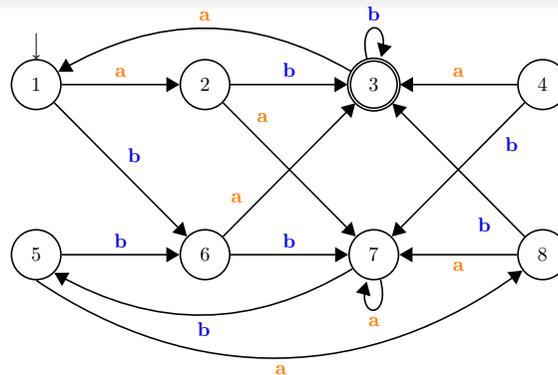


- Ingrédients de base : états (accepteurs), symboles, transitions — *syntaxe*.
- Exécution, mot accepté, langage accepté — *sémantique*.
- Problèmes de décision : langage vide, langage infini.
- Opérations sur automate/opérations sur langage : négation/complémentation, produit/intersection.

## Plan Chap. 6 - AEFD - Distinguabilité, Équivalence, Minimisation

- 1 Tester l'équivalence entre états
- 2 Tester l'équivalence entre automates
- 3 Minimisation d'automates 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 2 automates sont équivalents ?
- Peut-on avoir une représentation « canonique » (on dira minimale) d'un automate ?

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

Dans ce chapitre, nous considérons  $A = (Q, \Sigma, \delta, q_{\text{init}}, F)$  un AEFD **complet** dont tous les états sont accessibles.

## Plan Chap. 6 - AEFD - Distinguabilité, Équivalence, Minimisation

- 1 Tester l'équivalence entre états
- 2 Tester l'équivalence entre automates
- 3 Minimisation d'automates 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)

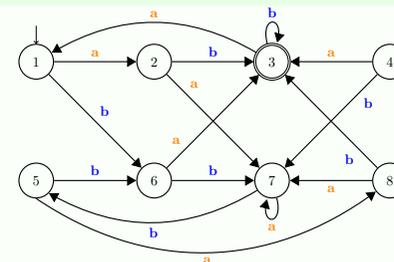
La relation de distinguabilité  $\neq$  entre états sur  $Q$  est définie par :

$$\forall p, q \in Q : \left( p \neq q \text{ ssi } \exists u \in \Sigma^* : (\delta^*(p, u) \in F \Leftrightarrow \delta^*(q, u) \in F) \right)$$

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  $q \equiv q$  pour tout état  $q$ .



## Distinguabilité entre états

### Propriétés de la relation

#### Rappel de la définition de la relation de distinguabilité

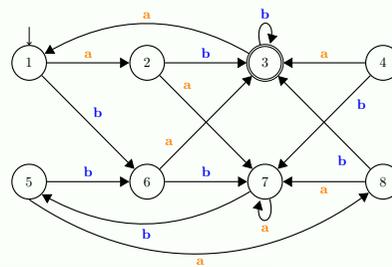
$$p \neq q \text{ ssi } \exists u \in \Sigma^* : (\delta^*(p, u) \in F \Leftrightarrow \delta^*(q, u) \in F)$$

#### 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 :  $\forall p, q \in Q : p \neq q \implies q \neq p$ .

#### Illustration du théorème



- anti-réflexivité :  $q \neq q$ , pour tout état  $q$ ,
- symétrie :  $6 \neq 1$  et  $1 \neq 6$ .

## Comment calculer la relation de distinguabilité ?

#### Comment calculer $\neq$ ?

La définition de la relation de distinguabilité n'est pas directement utilisable pour la calculer.

- Elle requière de lire des mots arbitrairement longs.
- Recherche de mots dans un ensemble infini ( $\Sigma^*$ ).

#### Technique pour calculer $\neq$

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

## Distinguabilité entre états à $k$ pas

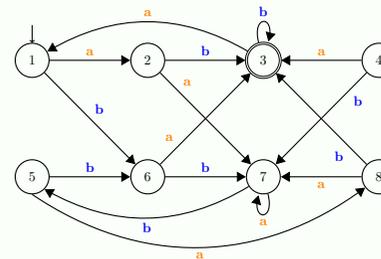
### Définition (Distinguabilité à $k$ pas)

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

### Exemple (États distinguables à $k$ pas)

- $k = 0$  :  $x \neq_0 3$  et  $3 \neq_0 x$ , avec  $x \in \{1, 2, 4, \dots, 8\}$  ;
- $k = 1$  :  $1 \neq_1 \{2, 6, 8\}$ ,  $2 \neq_1 \{1, 4, 5, 7\}$ ,  
... , et  $x \neq_1 y$  pour  $x \neq_0 y$  ;
- $k = 2$  :  $x \neq_2 y$  pour  $x \neq_1 y$ .



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

### Lemme

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

$$\exists u \in \Sigma^* : |u| \leq k \wedge (\delta^*(q, u) \in F \nleftrightarrow \delta^*(q', u) \in F).$$

### Corollaire

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

En utilisant la définition de la définition de distinguabilité à  $k$  pas, nous déduisons le lemme suivant.

### Lemme

Pour tout  $k \in \mathbb{N}$ ,

$$\neq_{k+1} = \neq_k \cup \bigcup_{a \in \Sigma} \{(q, q') \mid (\delta(q, a), \delta(q', a)) \in \neq_k\}$$

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

## Distinguabilité entre états

### Algorithme 1 Calcul des états distinguables

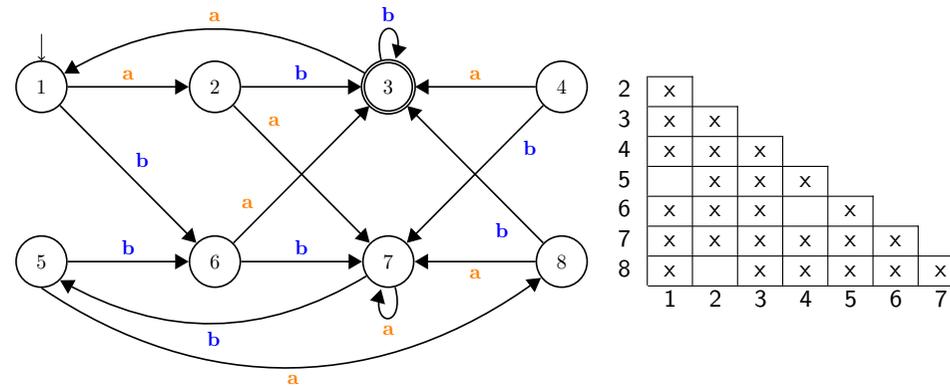
**Entrée :**  $A = (Q, \Sigma, \delta, q_{init}, F)$  (\* un AEFD complet dont tous les états sont accessibles \*)

**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**  $X$ ; (\* nouveaux distinguables pour chaque itération \*)
- 3:  $D := (F \times (Q \setminus F)) \cup (Q \setminus F) \times F$ ; (\*  $D$  initialisé avec états accepteurs/non-accepteurs,  $D \neq \emptyset$  \*)
- 4:  $D_{pre} := \emptyset$ ; (\* maj de  $D_{pre}$  (sauvegarde de  $D$ ) \*)
- 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**  $D$ ; (\*  $D \neq \emptyset$  \*)

**Remarque** Dans une implémentation de cet algorithme et la représentation de son exécution, il est souhaitable d'utiliser l'anti-réflexivité et la symétrie de  $D, D_{pre}$  (et  $X$ ). □

## Distinguabilité entre états : exemple



## Distinction entre états : correction de l'algorithme

Rappelons que  $A = (Q, \Sigma, q_{\text{init}}, \delta, F)$  est un AEFD complet 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 existe 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  $\{p, q\}$

- $w \notin \epsilon$  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. 6 - AEFD - Distinguabilité, Équivalence, Minimisation

- 1 Tester l'équivalence entre états
- 2 Tester l'équivalence entre automates
- 3 Minimisation d'automates 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 ?

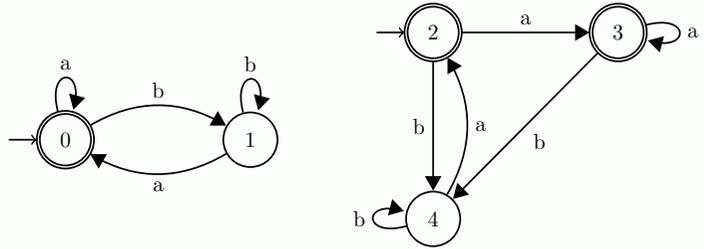
## Procédure pour tester l'équivalence entre deux automates

- 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. 6 - AEFD - Distinguabilité, Équivalence, Minimisation

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

## Équivalence entre états

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

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

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

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

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

- réflexive,
- symétrique,
- transitive.

Notations :

- $[q]$  : la classe d'équivalence de l'état  $q$
- $Q_{\equiv}$  : 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.

## La relation d'équivalence $\equiv_k$

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

### Question

Comment calculer  $\equiv$  ?

### Définition (Équivalence à $k$ pas)

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

- ①  $q \equiv_0 q'$  ssi  $q \in F \iff q' \in F$ .
- ② Pour  $k \in \mathbb{N}$ ,  $q \equiv_{k+1} q'$  ssi

$$q \equiv_k q' \wedge \forall a \in \Sigma : \delta(q, a) \equiv_k \delta(q', a).$$

Construction de  $\equiv$  à partir de  $\equiv_k$ 

## Lemme

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

$$\forall u \in \Sigma^* : |u| \leq k \implies (\delta^*(q, u) \in F \iff \delta^*(q', u) \in F).$$

## Corollaire

$$\bigcap_{k \in \mathbb{N}} \equiv_k = \equiv$$

En utilisant la définition de la définition d'équivalence à  $k$  pas, nous déduisons le lemme suivant.

## Lemme

Pour tout  $k \in \mathbb{N}$ ,

$$\equiv_{k+1} = \equiv_k \cap \bigcap_{a \in \Sigma} \{(q, q') \mid (\delta(q, a), \delta(q', a)) \in \equiv_k\}$$

Nous pouvons en déduire un algorithme de calcul des classes d'équivalence.

Algorithme de calcul de  $Q_{/\equiv}$ **Algorithme 2** Calcul des classes d'équivalence

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

**Sortie** :  $R = Q_{/\equiv}$

- 1: **ensemble de couples d'états**  $R$ ; (\*  $R$  est la partition finale "la plus fine" recherchée \*)
- 2: **ensemble de couples d'états**  $R_{\text{pre}}$  (\*  $R_{\text{pre}}$  est la valeur de  $R$  à l'itération précédente \*)
- 3: **ensemble de couples d'états**  $X$ ; (\* temporaire pour chaque itération, états distinguables trouvés \*)
- 4:  $R := (F \times F) \cup ((Q \setminus F) \times (Q \setminus F))$ ; (\*  $R$  initialisée à la partition entre états accepteurs/non accepteurs,  $R = \equiv_0$  \*)
- 5:  $R_{\text{pre}} := \emptyset$ ;
- 6: **tant que**  $R_{\text{pre}} \neq R$  **faire**
- 7:      $R_{\text{pre}} := R$ ; (\* maj de  $R_{\text{pre}}$  \*)
- 8:      $X := \{(p, q) \in R \mid \exists a \in \Sigma : ((\delta(p, a), \delta(q, a)) \notin R)\}$ ; (\* calcul des états distinguables \*)
- 9:      $R := R \setminus X$ ; (\* enlever les états distinguables découverts à cette itération de  $R$  \*)
- 10: **fin tant que**
- 11: **retourner**  $R$ ;

**Remarque** Dans une implémentation de cet algorithme et la représentation de son exécution, il est souhaitable d'utiliser la réflexivité et la symétrie de  $R$ ,  $R_{\text{pre}}$  (et  $X$ ).  $\square$

## Minimisation : automate minimisé et équivalence

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

**Définition (Minimisé d'un automate – aussi appelé automate quotient)**

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

- $\delta_{/\equiv}$  est la fonction de transition t. q. :

$$\delta_{/\equiv} : Q_{/\equiv} \times \Sigma \rightarrow Q_{/\equiv}$$

$$\delta_{/\equiv}([q], a) \stackrel{\text{def}}{=} [\delta(q, a)]$$

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

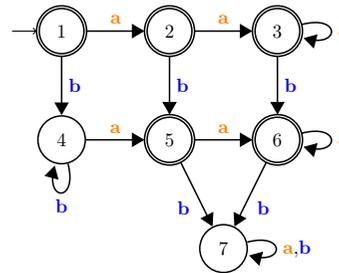
**Remarque** Comme  $\delta$  est une application,  $\delta_{/\equiv}$  l'est également. □

### Théorème

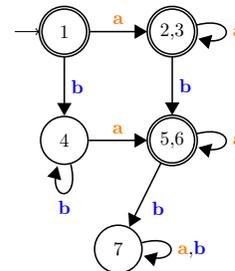
Étant donné  $A$  et son minimisé  $A_{/\equiv}$  :

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

## 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 obtenu par l'algorithme de minimisation.

Supposons qu'il existe un automate  $N$  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 indistinguable car  $L(M) = L(N)$ .
- Remarquer que si  $p$  et  $q$  sont indistinguable alors tous les successeurs sur n'importe quel symbole sont indistinguable (sinon  $p$  et  $q$  seraient distinguable).
- Tous les états de  $M$  sont indistinguable 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 indistinguable
- Comme  $N$  a moins d'états que  $M$ , il y a deux états de  $M$  qui sont indistinguable du même état dans  $N$ .
- Par transitivité de la relation d'indistinguable, ces deux états sont indistinguable l'un de l'autre.
- Contradiction : d'après la correction de l'algorithme de minimisation,  $M$  est tel que tous ses états sont distinguable.

## Plan Chap. 6 - AEFD - Distinguabilité, Équivalence, Minimisation

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

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

### *Équivalence et Minimisation d'Automate Déterministes*

- Distinguabilité ( $\neq$ ) et équivalence ( $\equiv$ ) entre états,
- Distinguabilité et équivalence entre automates,
- Minimisation d'automate ( $A_{\equiv}$ ).

### Bonus

- Déterminer pourquoi les algos de calculs des relations d'états distinguables et équivalents terminent.