IUT de Bordeaux
Département Informatique

M2201 – Graphes et Langages

Éléments de Théorie des Langages

- **Déterminisation d'un AF**
- **Minimisation d'un AFD**

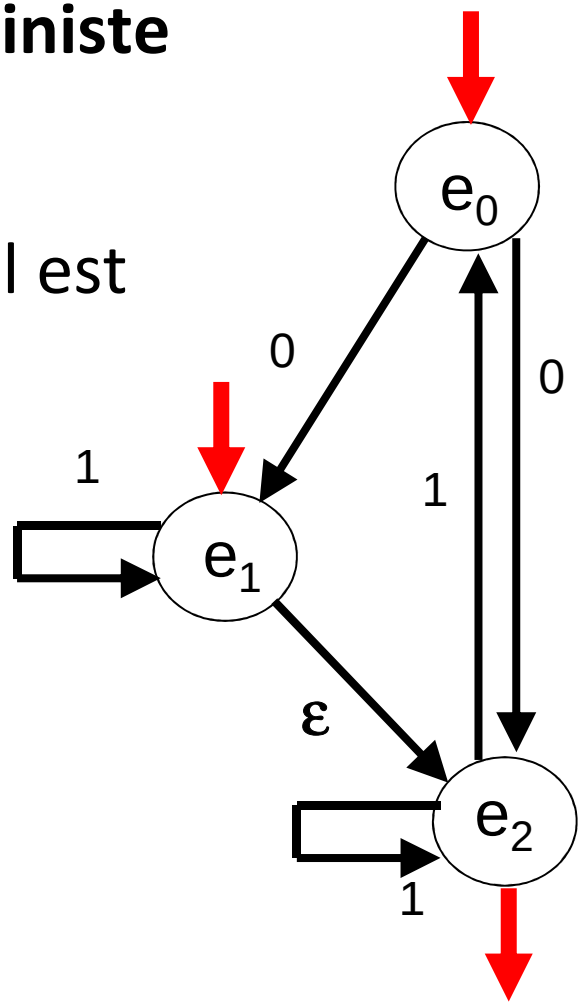
Déterminisation d'un automate fini

Automates non déterministes (rappel)

Nous avons vu qu'il était parfois plus simple de produire un automate fini **non déterministe (AF)**.

Rappelons que pour un tel automate, il est possible d'avoir :

- plusieurs états initiaux,
- plusieurs transitions issus d'un même état et portant la même étiquette,
- des transitions étiquetées par ε (appelées ε-transitions)



Équivalence entre AFs et AFDs

Nous allons démontrer que les AF ne sont pas plus « puissants » que les AFD : les langages reconnus par les AF sont exactement les mêmes que ceux reconnus par les AFD.

Comme tout AFD est un AF, il suffit de montrer la proposition suivante :

Proposition. Pour tout automate fini **M**, il existe un automate fini déterministe **M'** tel que L(**M'**) = L(**M**).

*Pour démontrer ce résultat, nous allons donner un algorithme qui, à partir d'un AF **M**, construit un AFD **M'** équivalent…*

Déterminisation d'un AF : principe

L'idée de base est la suivante :

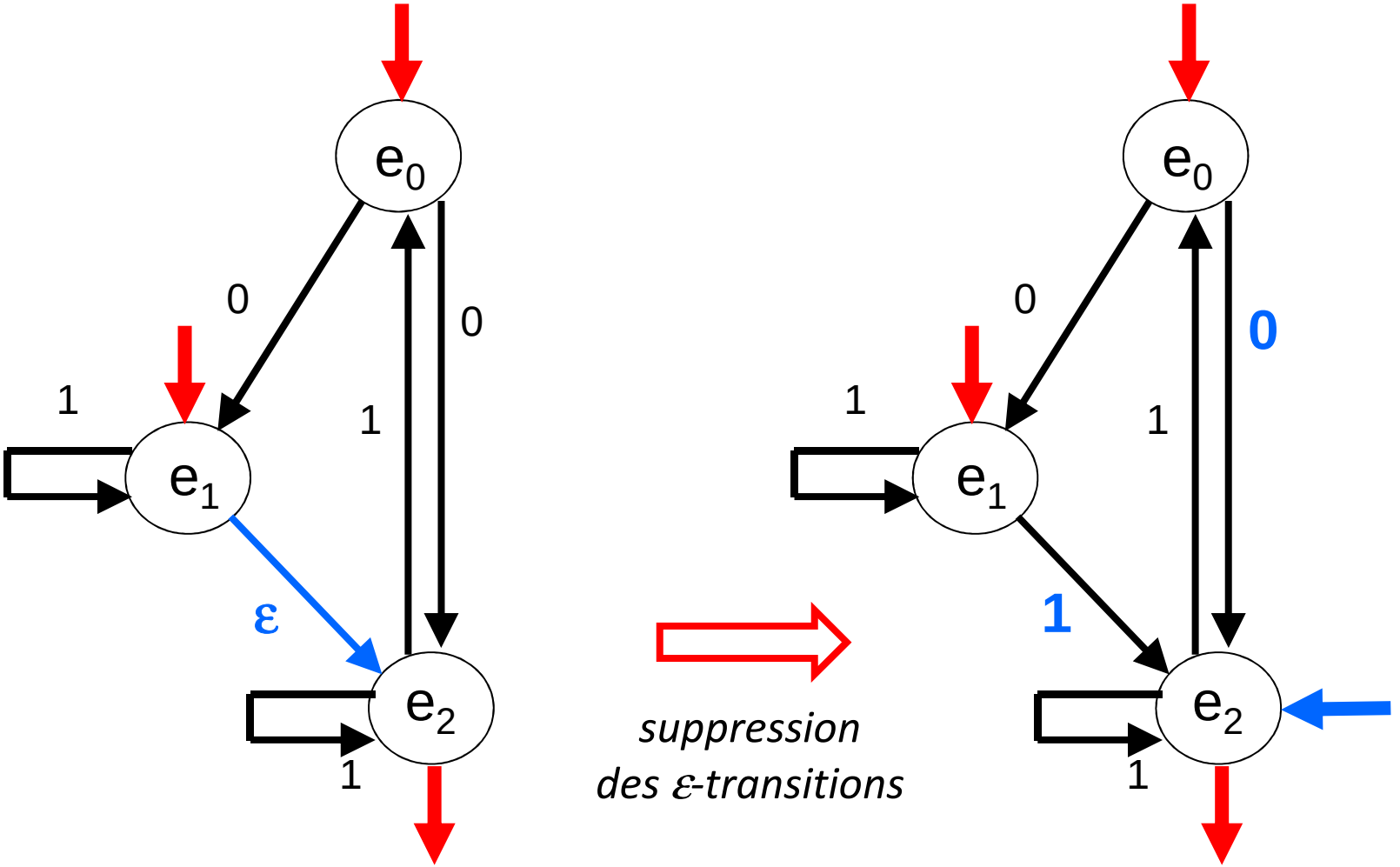
- On supprime les ε-transitions de **M**, une à une, pour obtenir un AF **M*** sans ε-transition :

  si (**e,ε,f**) $\in$ δ et **e** est un état initial alors **f** devient un état initial ;
  si (**e,ε,f**), (**g,a,e**) $\in$ δ alors (**g,a,f**) $\in$ δ^*.

- Les états de **M'** seront des ensembles d'états de **M***.
- L'état initial de **M'** est l'ensemble des états initiaux de **M***.
- Ensuite, pour chaque lettre **a** $\in$ **A** et chaque état **S** de **M***, on pose : $\delta'(S,a) = \{ \delta^*(e_i,a) / e_i \in S \}$.
- Si **S** contient un état terminal de **M***, alors **S** est un état terminal de **M'**.
- Cet algorithme s'arrête car **M*** possède un nombre fini d'états : il n'y a donc qu'un nombre fini d'états possibles pour **M'**…

Déterminisation d'un AF : exemple (1)

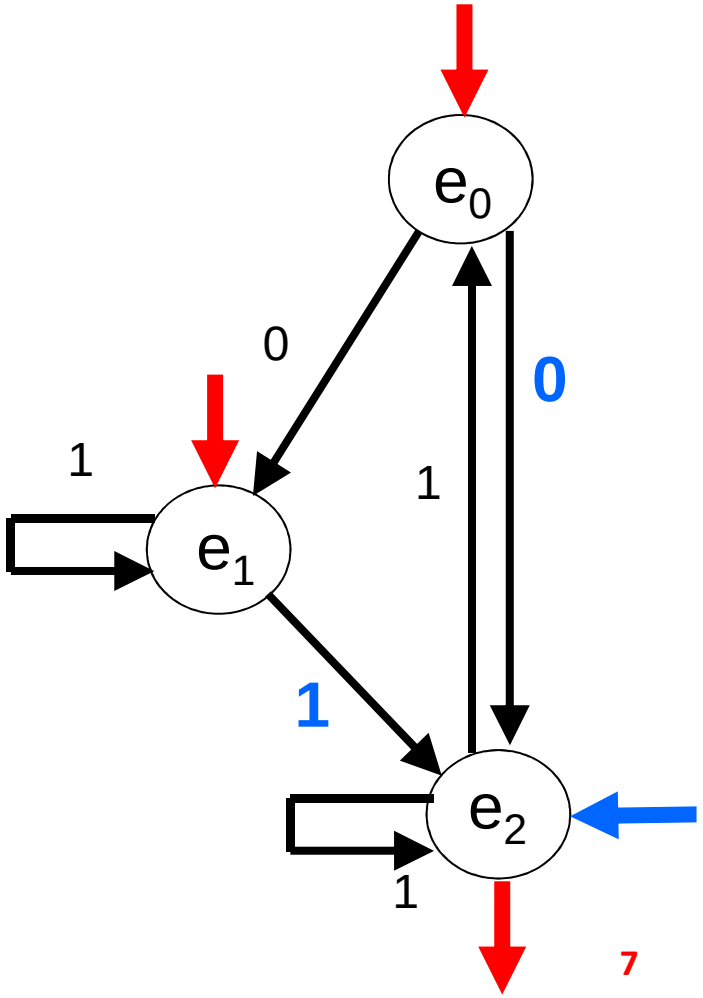
On notera e_3 l'état-puits de notre AF.



Déterminisation d'un AF : exemple (2)

La fonction de transition de l'AFD s'obtient ainsi :

état		0	1	∈ T'
S_0 initial	$e_0e_1e_2$	$e_1e_2e_3$	$e_0e_1e_2e_3$	O
S_1	$e_1e_2e_3$	e_3	$e_0e_1e_2e_3$	O
S_2	$e_0e_1e_2e_3$	$e_1e_2e_3$	$e_0e_1e_2e_3$	
S_3 puits	e_3	e_3	e_3	O

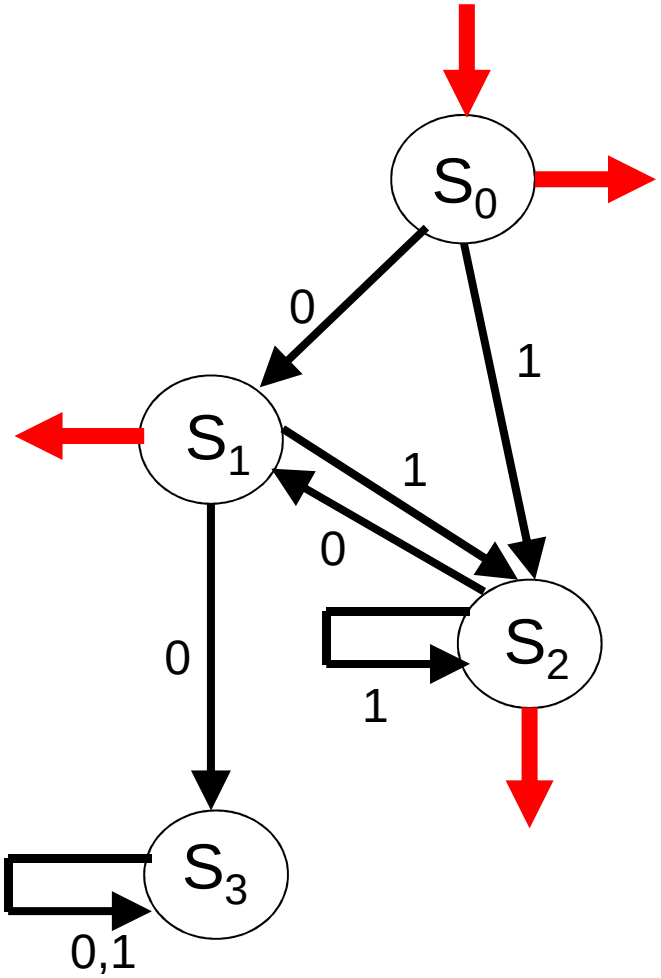


On démarre avec l'état initial $S_0 = \{e_0, e_1, e_2\}$.
Quand un nouvel état « apparaît », on crée une nouvelle ligne…

Déterminisation d'un AF : exemple (3)

On obtient l'AFD suivant :

état		0	1	∈ T'
S_0 initial	$e_0e_1e_2$	$e_1e_2e_3$	$e_0e_1e_2e_3$	O
S_1	$e_1e_2e_3$	e_3	$e_0e_1e_2e_3$	O
S_2	$e_0e_1e_2e_3$	$e_1e_2e_3$	$e_0e_1e_2e_3$	O
S_3 puits	e_3	e_3	e_3	

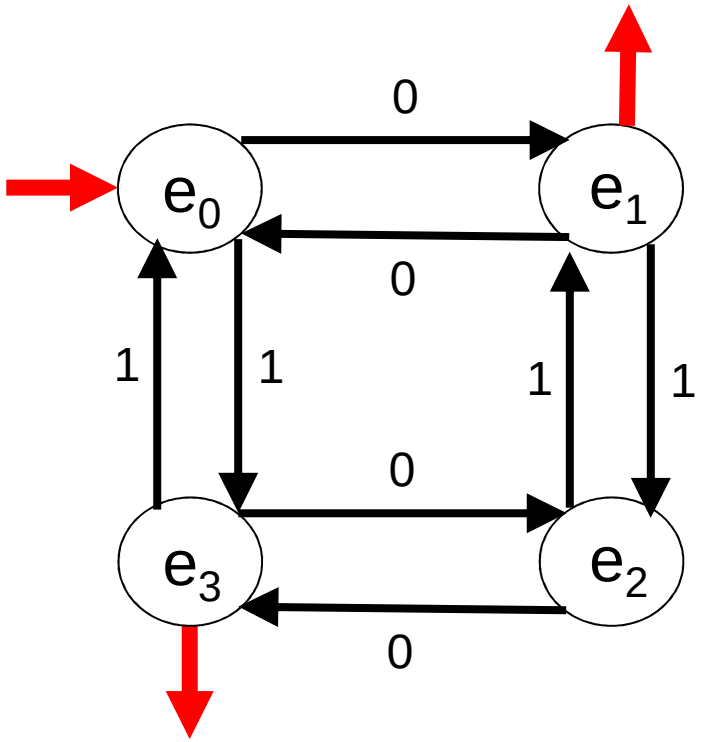


Remarque. *L'automate ainsi construit n'est pas nécessairement minimal…*

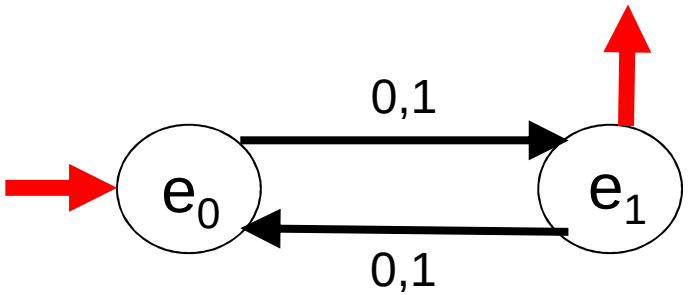
Minimisation d'un automate fini déterministe

AFD minimal

Nous avons vu que plusieurs AFD distincts pouvaient reconnaître le même langage. Un tel AFD est dit **minimal** s'il contient un nombre minimum d'états.



non minimal…



minimal !

Minimisation d'un AFD : principe (1)

L'idée de base est la suivante : on va raisonner sur des **classes d'équivalence** d'états.

Deux états **e** et **f** seront considérés comme **équivalents** si, pour toute lettre **a** $\in$ **A**, δ**(e,a)** et δ**(f,a)** sont deux états équivalents… (Intuition : ils ont « même futur ».)

Pour déterminer ces classes d'équivalence, on va procéder par raffinements successifs en « divisant » les ensembles contenant des états non équivalents…

Remarque. *Un pré-traitement de l'AFD permet d'éliminer les états non accessibles à partir de l'état initial…*

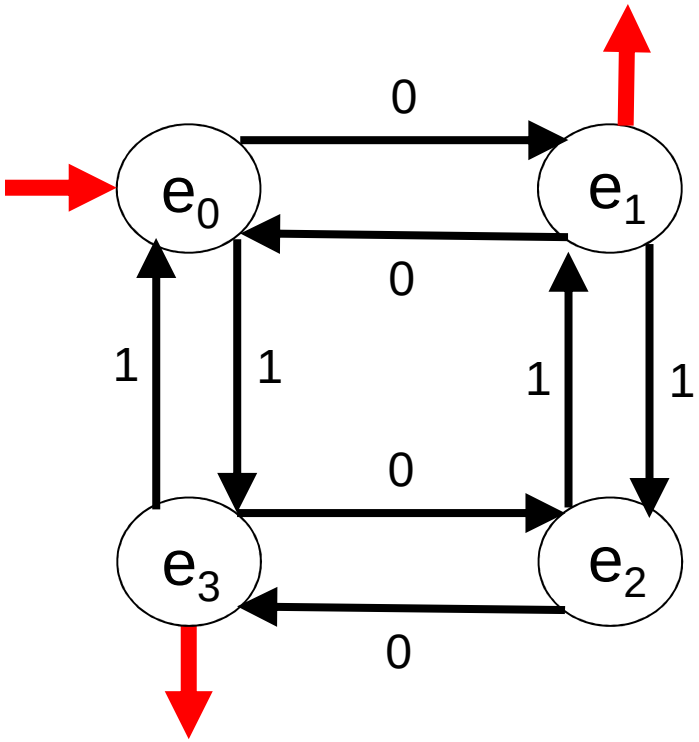
Minimisation d'un AFD : principe (2)

Principe de minimisation : **Algorithme de Moore.**

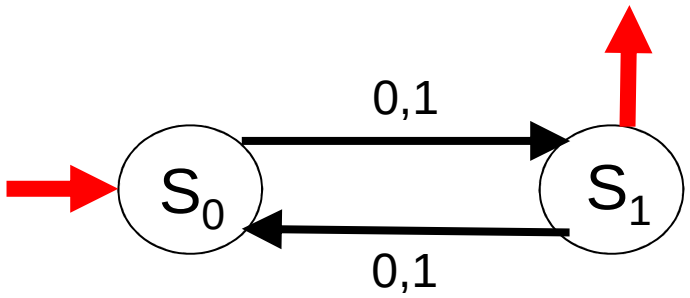
On suppose que tous les états sont accessibles à partir de l'état initial.

- On démarre avec deux ensembles, **T** et **E \ T.**
- Si, dans un ensemble **S**, il existe deux états **e** et **f** et une lettre a tels que **δ(e,a)** et **δ(f,a)** n'appartiennent pas au même ensemble, on divise **S** de façon à mettre ensemble tous les états qui ont même « ensemble successeur » par la lettre **a**.
- L'algorithme se termine lorsque plus aucun ensemble ne doit être divisé : chaque ensemble est alors un état de l'automate minimal. Les sous-ensembles de T sont des états terminaux, et l'ensemble contenant l'état initial d'origine est l'état initial.

Minimisation d'un AFD : exemple (1)

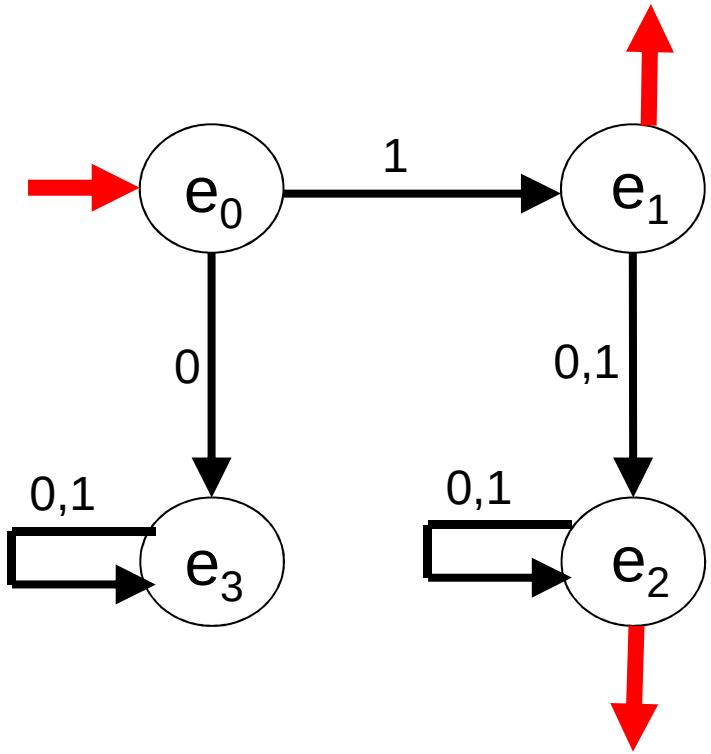


- étape 1 : { e_0, e_2 }, { e_1, e_3 }.
- étape 2 : rien à faire, aucune partie ne satisfait la condition de séparation !
- $S_0 \leftarrow$ { e_0, e_2 }, initial
  $S_1 \leftarrow$ { e_1, e_3 }, terminal.



Trop facile !... ;-)

Minimisation d'un AFD : exemple (2)

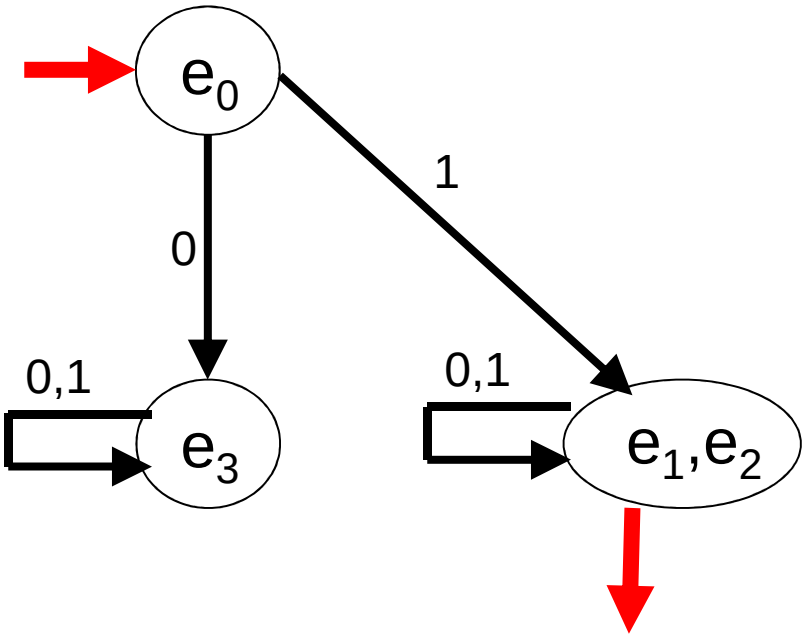


- étape 1 : { e_0, e_3 }, { e_1, e_2 }.
- étape 2 :
  { e_0, e_3 } → séparés par 1
  { e_0 }, { e_3 }, { e_1, e_2 }
- étape 3 :
  { e_1, e_2 } → OK
  { e_0 }, { e_3 }, { e_1, e_2 }

	e_0	**e_1**	**e_2**	**e_3**
0	e_3	e_2	e_2	e_3
1	e_1	e_2	e_2	e_3

- résultat final, 3 états :
  { e_0 }, { e_3 }, { e_1, e_2 }

Minimisation d'un AFD : exemple (3)



	e_0	e_1	e_2	e_3
0	e_3	e_2	e_2	e_3
1	e_1	e_2	e_2	e_3

- résultat final, 3 états :
  $\{ e_0 \}, \{ e_3 \}, \{ e_1, e_2 \}$