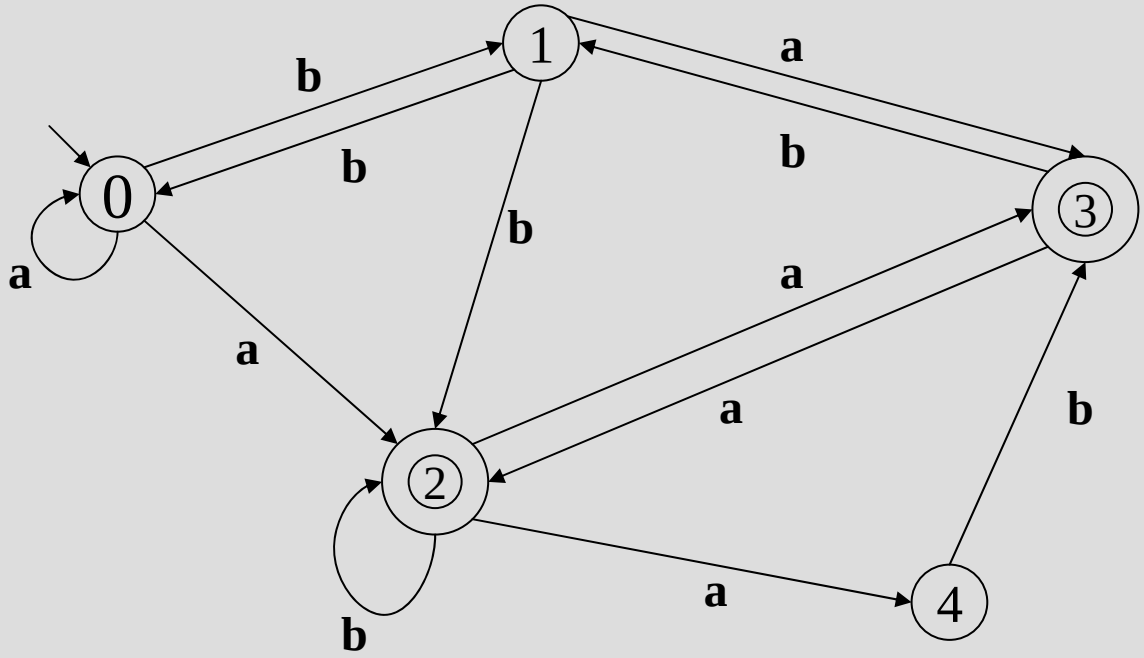
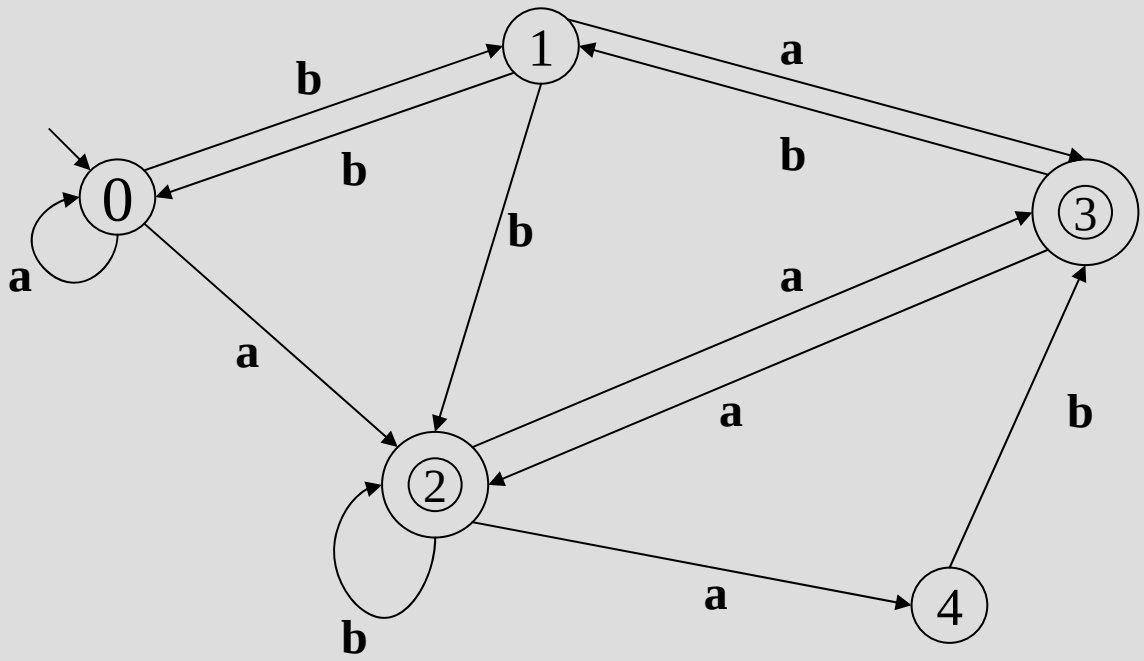
AFN sans ε-transition → AFD

- Application sur un exemple :



AFN sans ε-transition → AFD

- Application sur un exemple :



état	a	b
0	0,2	1
1	3	0,2
2	3,4	2
3	2	1
4	-	3

AFN sans ε-transition → AFD

état	a	b
0	0,2	1
1	3	0,2
2	3,4	2
3	2	1
4	-	3

T={2,3}

Considérer les ensembles d'état comme un nouvel état
Partir de l'état initial et construire les ensembles des atteints par chaque transition

AFN sans ε-transition → AFD

état	a	b
0	0,2	1

état	a	b
0	0,2	1
1	3	0,2
2	3,4	2
3	2	1
4	-	3

Considérer les ensembles d'état comme un nouvel état
Partir de l'état initial et construire les ensembles des atteints par chaque transition

AFN sans ε-transition → AFD

état	a	b
0	0,2	1

état	a	b
0	0,2	1
1	3	0,2
2	3,4	2
3	2	1
4	-	3

Considérer les ensembles d'état comme un nouvel état

Partir de l'état initial et construire les ensembles des atteints par chaque transition

AFN sans ε-transition → AFD

état	a	b
0	0,2	1
0,2	0,2,3,4	1,2
1	3	0,2

état	a	b
0	0,2	1
1	3	0,2
2	3,4	2
3	2	1
4	-	3

Recommencez avec les nouveaux ensembles

AFN sans ε-transition → AFD

état	a	b
0	0,2	1
0,2	0,2,3,4	1,2
1	3	0,2
0,2,3,4		
1,2		
3		

état	a	b
0	0,2	1
1	3	0,2
2	3,4	2
3	2	1
4	-	3

Recommencez avec les nouveaux ensembles

AFN sans ε-transition → AFD

état	a	b
0	0,2	1
0,2	0,2,3,4	1,2
1	3	0,2
0,2,3,4	0,2,3,4	1,2,3
1,2	3,4	0,2
3	2	1

état	a	b
0	0,2	1
1	3	0,2
2	3,4	2
3	2	1
4	-	3

Recommencez avec les nouveaux ensembles

AFN sans ε-transition → AFD

état	a	b
0	0,2	1
0,2	0,2,3,4	1,2
1	3	0,2
0,2,3,4	0,2,3,4	1,2,3
1,2	3,4	0,2
3	2	1
1,2,3	3,4,2	0,2,1
3,4	2	1,3
2	3,4	2

état	a	b
0	0,2	1
1	3	0,2
2	3,4	2
3	2	1
4	-	3

Recommencez avec les nouveaux ensembles

AFN sans ε-transition → AFD

état	a	b
0	0,2	1
0,2	0,2,3,4	1,2
1	3	0,2
0,2,3,4	0,2,3,4	1,2,3
1,2	3,4	0,2
3	2	1
1,2,3	3,4,2	0,2,1
3,4	2	1,3
2	3,4	2
2,3,4	3,4,2	2,1,3
0,1,2	0,2,3,4	1,0,2
1,3	3,2	0,2,1

état	a	b
0	0,2	1
1	3	0,2
2	3,4	2
3	2	1
4	-	3

Recommencez avec les nouveaux ensembles

AFN sans ε-transition → AFD

état	a	b
0	0,2	1
0,2	0,2,3,4	1,2
1	3	0,2
0,2,3,4	0,2,3,4	1,2,3
1,2	3,4	0,2
3	2	1
1,2,3	3,4,2	0,2,1
3,4	2	1,3
2	3,4	2
2,3,4	3,4,2	2,1,3
0,1,2	0,2,3,4	1,0,2
1,3	3,2	0,2,1
2,3	3,4,2	2,1

état	a	b
0	0,2	1
1	3	0,2
2	3,4	2
3	2	1
4	-	3

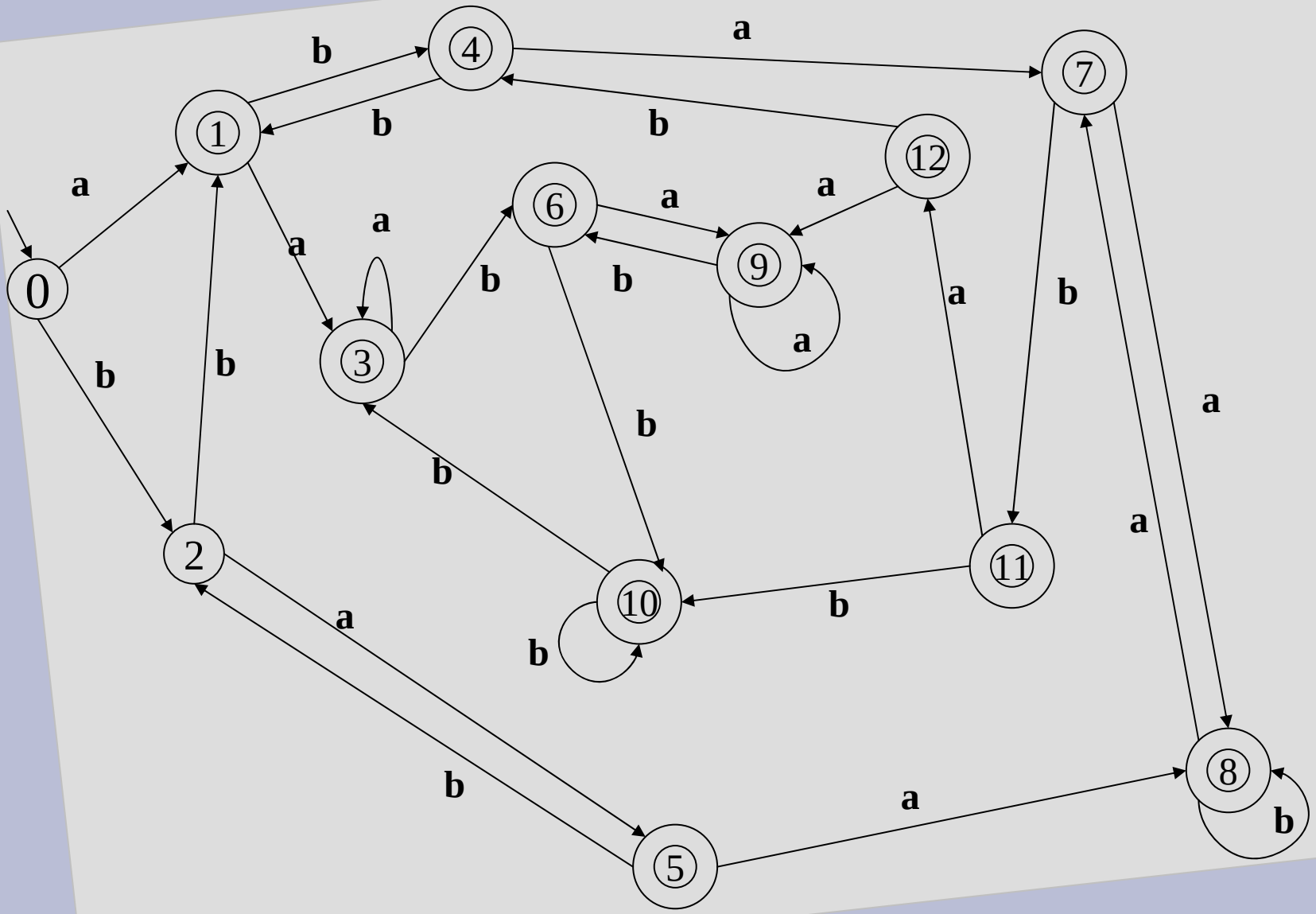
AFN sans ε-transition → AFD

état		a		b	
0	**0**	0,2	**1**	1	**2**
0,2	**1**	0,2,3,4	**3**	1,2	**4**
1	**2**	3	**5**	0,2	**1**
0,2,3,4	**3**	0,2,3,4	**3**	1,2,3	**6**
1,2	**4**	3,4	**7**	0,2	**1**
3	**5**	2	**8**	1	**2**
1,2,3	**6**	3,4,2	**9**	0,2,1	**10**
3,4	**7**	2	**8**	1,3	**11**
2	**8**	3,4	**7**	2	**8**
2,3,4	**9**	3,4,2	**9**	2,1,3	**6**
0,1,2	**10**	0,2,3,4	**3**	1,0,2	**10**
1,3	**11**	3,2	**12**	0,2,1	**10**
2,3	**12**	3,4,2	**9**	2,1	**4**

Renommez enfin les états

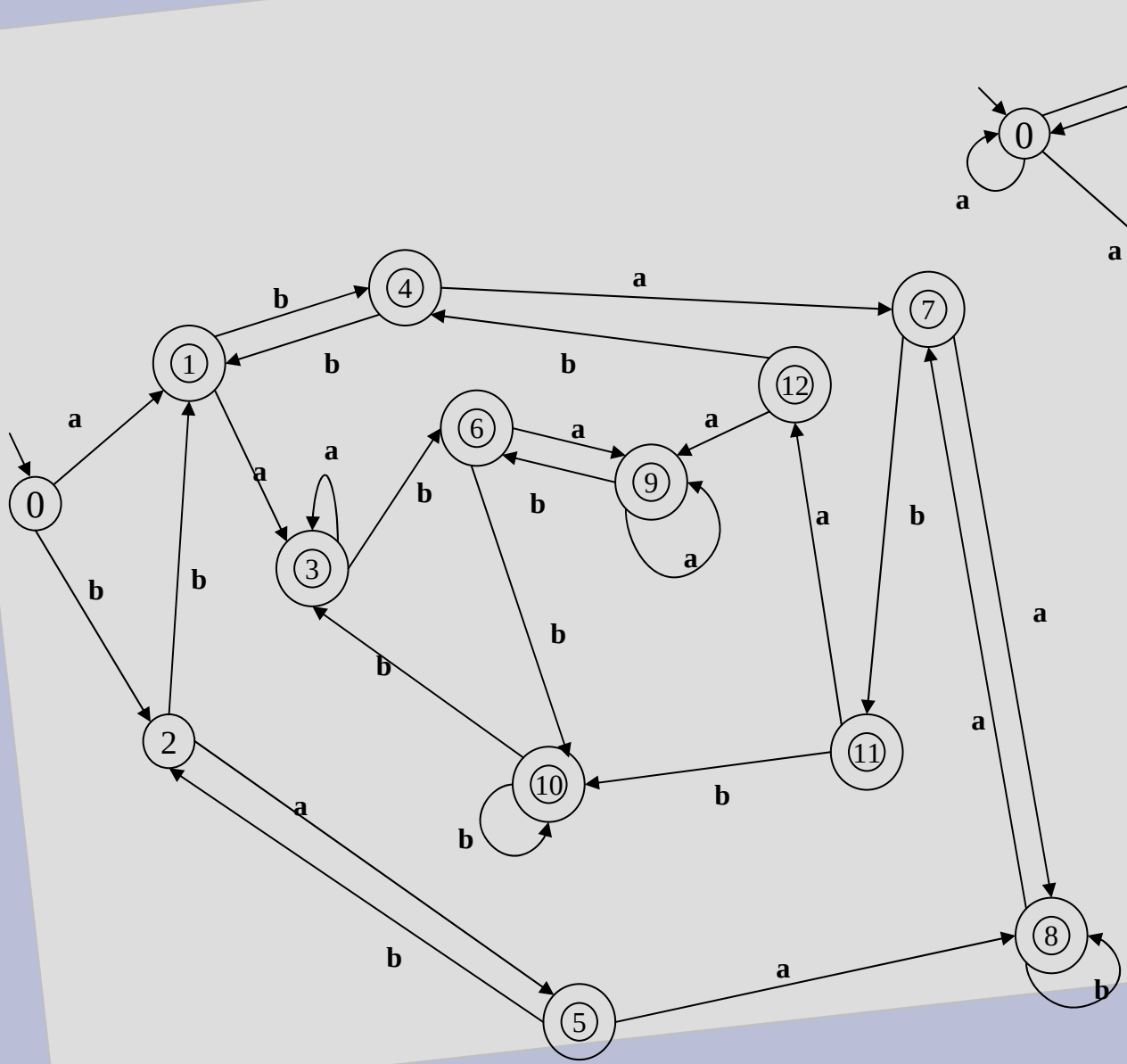
Les états contenant au moins un terminal deviennent terminaux
T={1,3,4,4,5,6, 7,8,9,10,11,12}

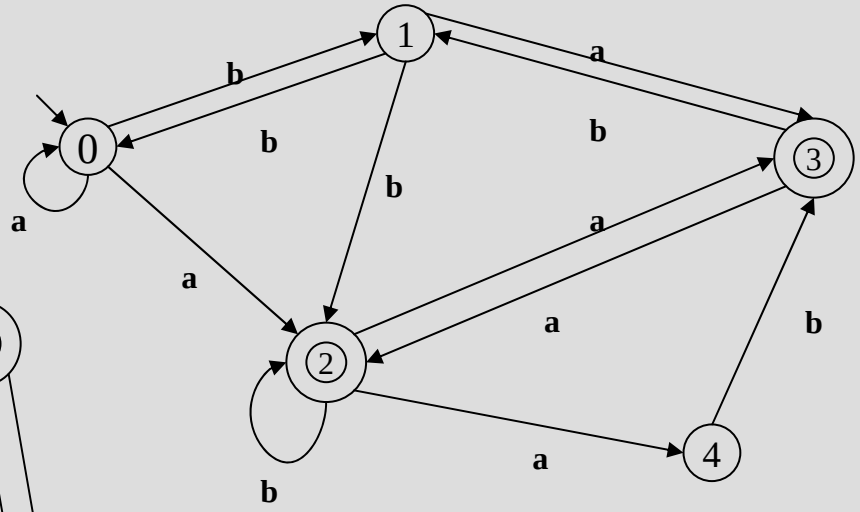
AFN sans ε-transition → AFD



état	a	b
0	1	2
1	3	4
2	5	1
3	3	6
4	7	1
5	8	2
6	9	10
7	8	11
8	7	8
9	9	6
10	3	10
11	12	10
12	9	4

AFN sans ε-transition → AFD





Simplification d'un AFD

- Faire classes d'équivalence : en séparant les états terminaux A et les non terminaux B
- S'il existe un symbole a_k et 2 états e_i et e_j d'une même classe tels que $\Delta(e_i, a_k)$ et $\Delta(e_j, a_k)$ n'appartiennent pas à la même classe, créer alors une nouvelle classe et séparer e_i et e_j . On laisse dans une même classe tous les états qui donnent un état d'arrivée dans la même classe
- Recommencer jusqu'à ce qu'il n'y ait plus rien à séparer
- Chaque classe forme un état du nouvel automate

Simplification d'un AFD

état	a	b
0	**1**	**2**
1	**3**	**4**
2	**5**	**1**
3	**3**	**6**
4	**7**	**1**
5	**8**	**2**
6	**9**	**10**
7	**8**	**11**
8	**7**	**8**
9	**9**	**6**
10	**3**	**10**
11	**12**	**10**
12	**9**	**4**

- A = {0,2} (NT)
- B = {1,3,4,5,6,7,8,9,10,11,12} (T)

Simplification d'un AFD

état	a	b
0	1	2
1	3	4
2	5	1
3	3	6
4	7	1
5	8	2
6	9	10
7	8	11
8	7	8
9	9	6
10	3	10
11	12	10
12	9	4

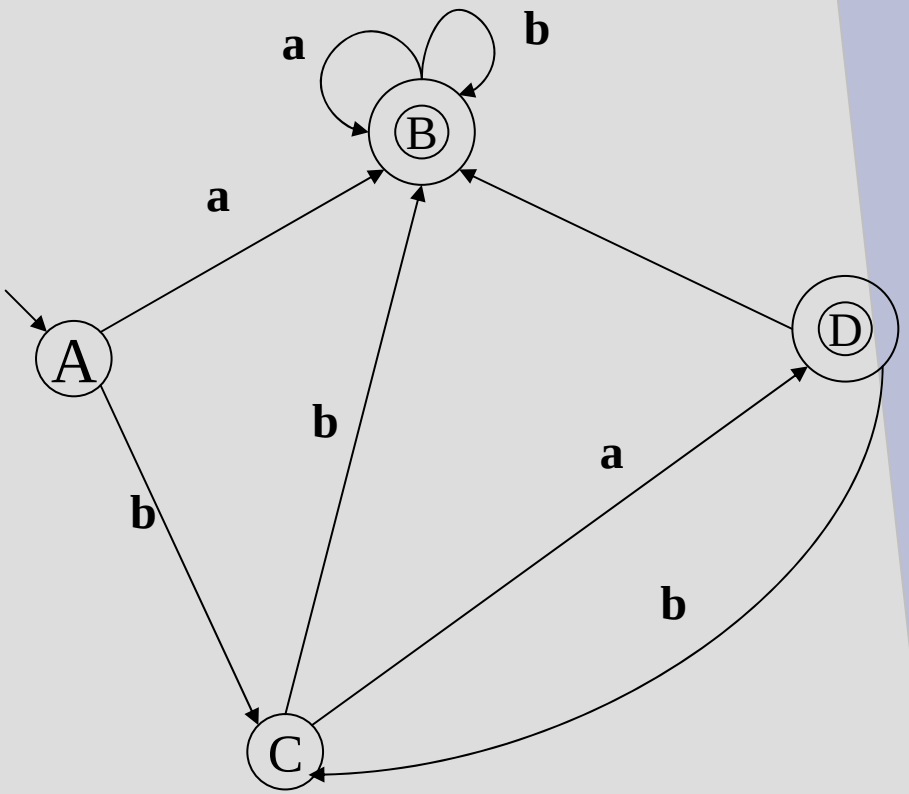
- A = {0,2}
- B = {1,3,4,5,6,7,8,9,10,11,12}
- $\Delta(0,b) = 2 \in A$, $\Delta(2,b) = 1 \in B$
- ➔ A = {0} C = {2}
- $\Delta(5,b) = 2 \in C$, $\Delta(1,b) = 4 \in B$
- ➔ B = {1,3,4,6,7,8,9,10,11,12} D= {5}
- $\forall i \in B \ \Delta(i,a) \in B \quad \Delta(i,b) \in B$
- Terminaux B et D

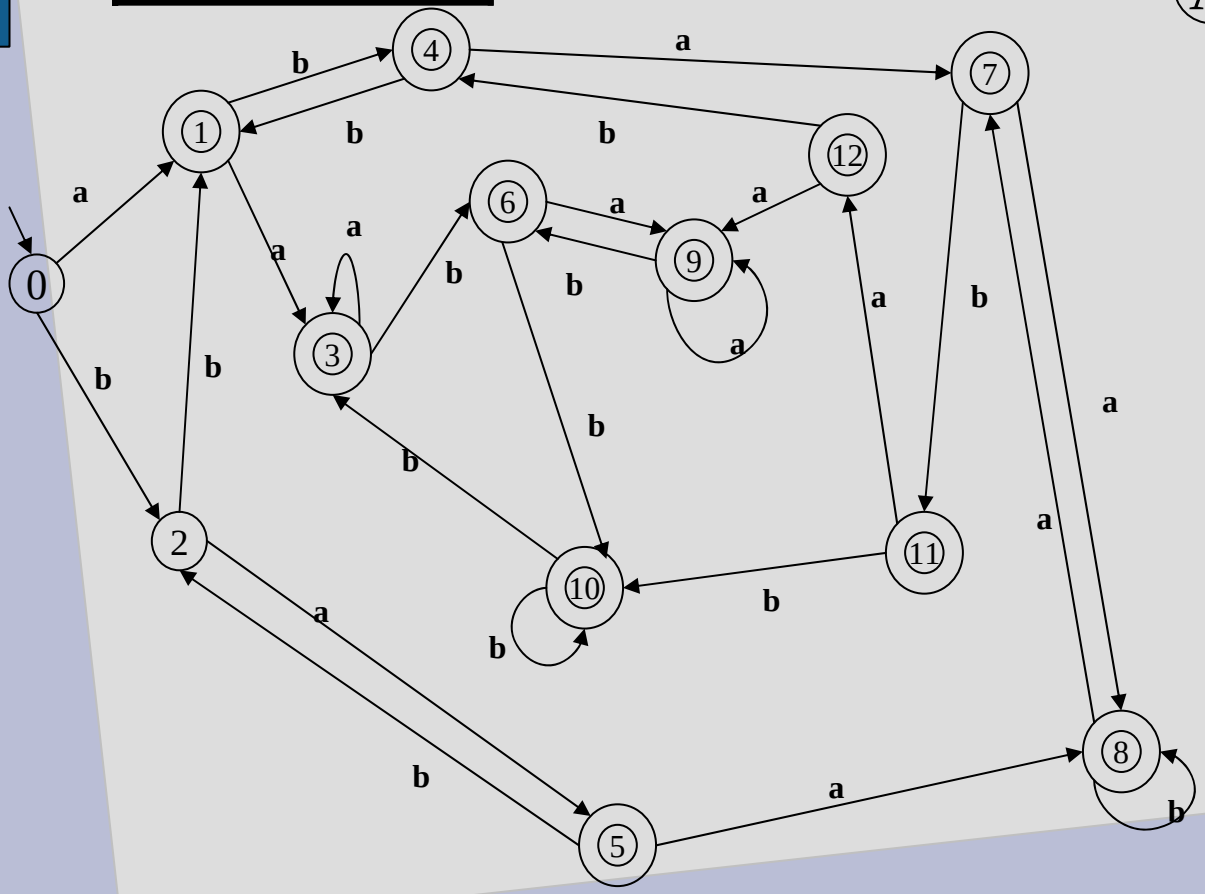
état	a	b
0	1	2
1	3	4
2	5	1
5	8	2

état	a	b
A	B	C
B	B	B
C	D	B
D	B	C

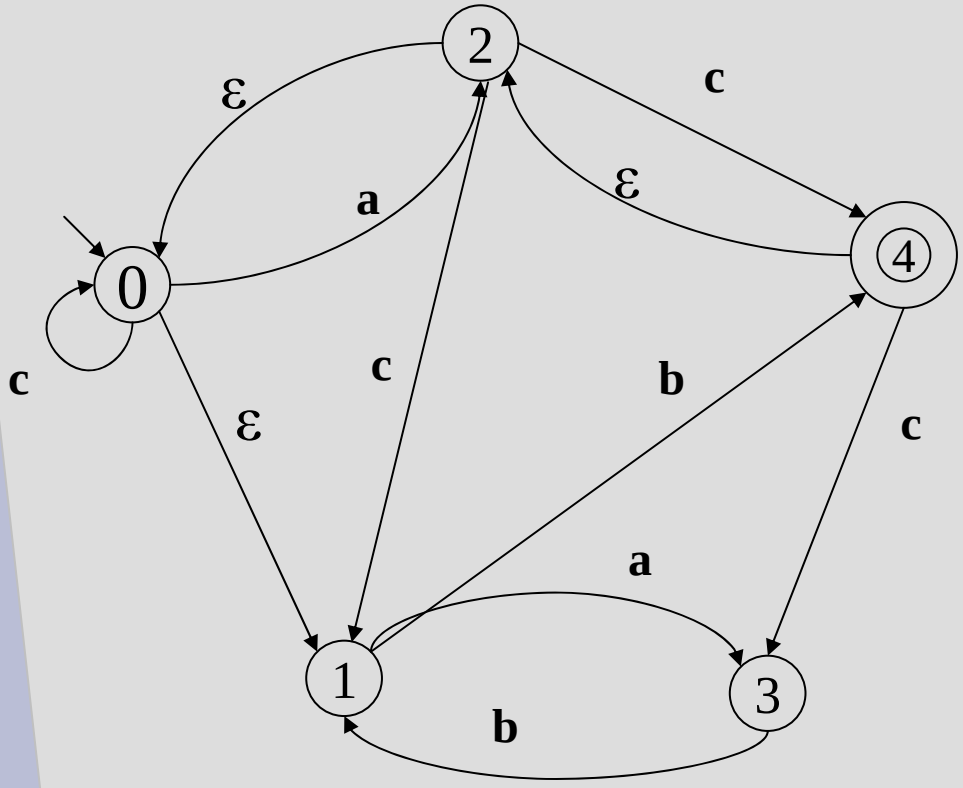
Simplification d'un AFD

état	a	b
A	B	C
B	B	B
C	D	B
D	B	C



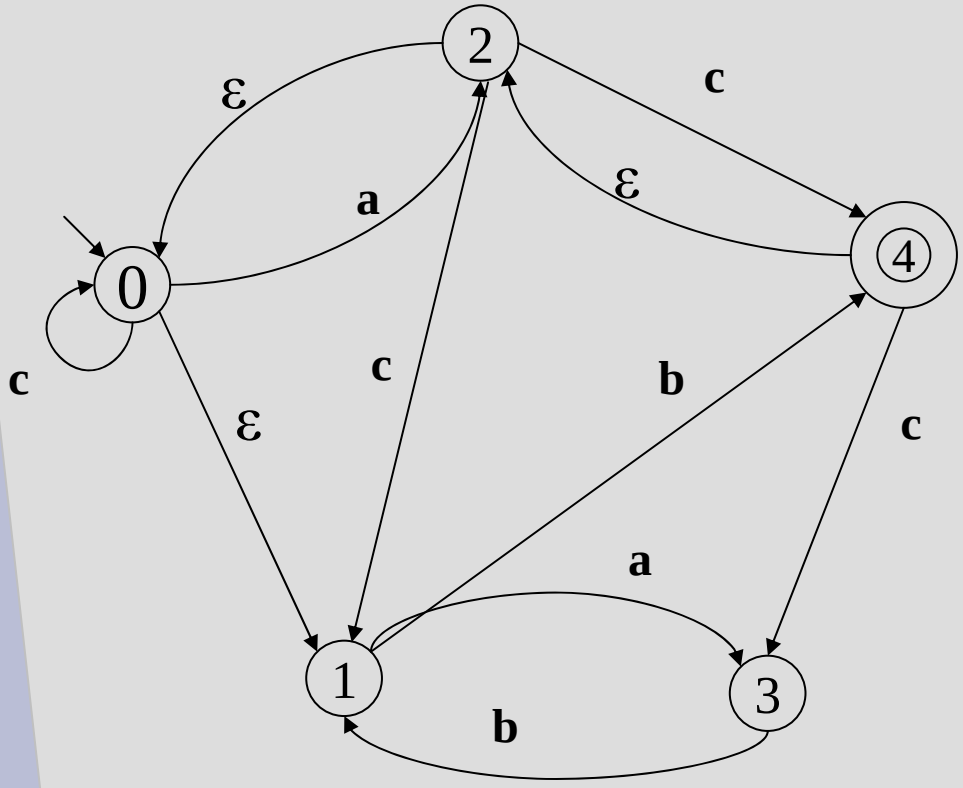


Cas général : AFN → AFD



Cas général : AFN → AFD

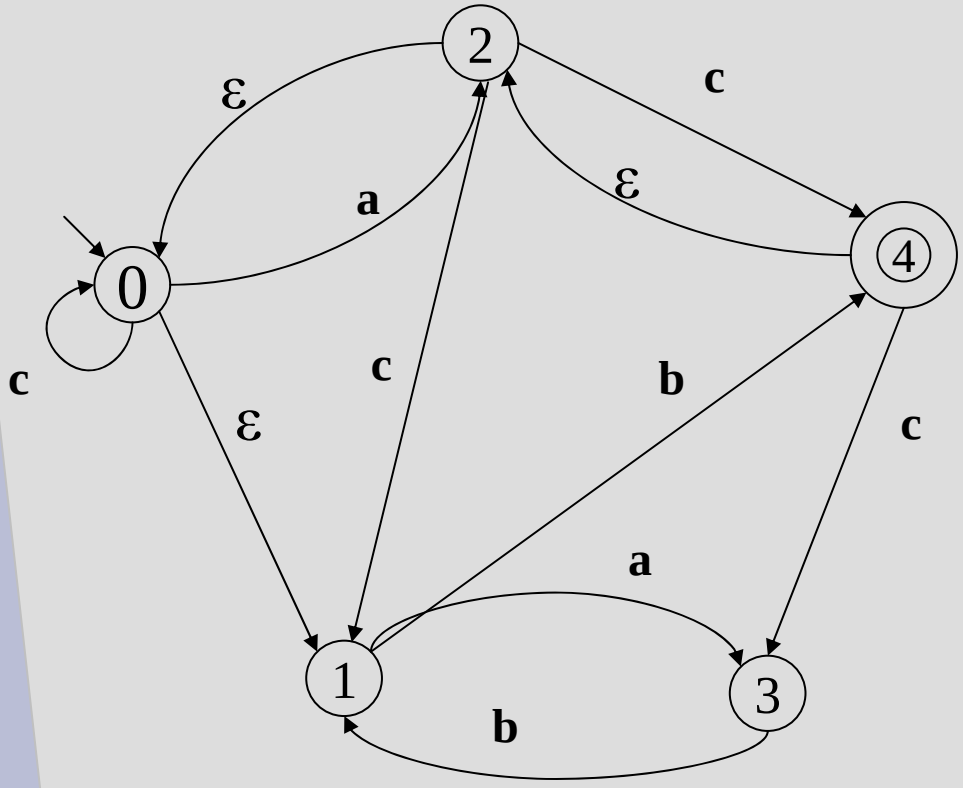
- On appelle ε-fermeture ε_f de l'ensemble d'états E, l'ensemble des états accessible depuis chaque élément de E par des ε-transitions



- $\varepsilon_f(\{e_0\}) = \{e_0, e_1\}$
- $\varepsilon_f(\{e_1\}) = \{e_1\}$
- $\varepsilon_f(\{e_2\}) = \{e_2, e_0, e_1\}$
- $\varepsilon_f(\{e_3\}) = \{e_3\}$
- $\varepsilon_f(\{e_4\}) = \{e_4, e_2, e_0, e_1\}$
- $\varepsilon_f(\{e_3, e_4\}) = \{e_3, e_4, e_2, e_0, e_1\}$
- …

Cas général : AFN → AFD

- Voyons sur l'exemple:



état	a	b	c	ε
0	2		0	1
1	3	4		
2			1,4	0
3		1		
4			3	2

Cas général : AFN → AFD

- Partir de ε-fermeture de l'état initial $\varepsilon_f(\{0\})=\{0,1\}$
- Ajouter dans la table de transition toutes les ε-fermeture des nouveaux états produits, ainsi que leur transitions, recommencer jusqu'à ce qu'il n'y ait plus de nouvel état
- Les états contenant au moins un terminal deviennent terminaux
- Renuméroter

Exemple :

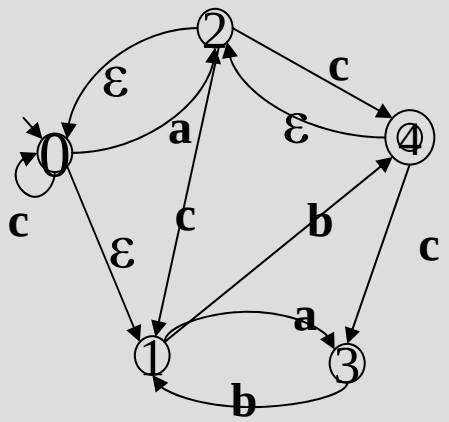
$\varepsilon_f(\Delta(\{1\},b))=\{1,4,2,0\}$

état	a	b	c	ε
0	2		0	1
1	3	4		
2			1,4	0
3		1		
4			3	2

Cas général : AFN → AFD

état	a	b	c
0,1	2,0,1,3	4,2,0,1	0,1

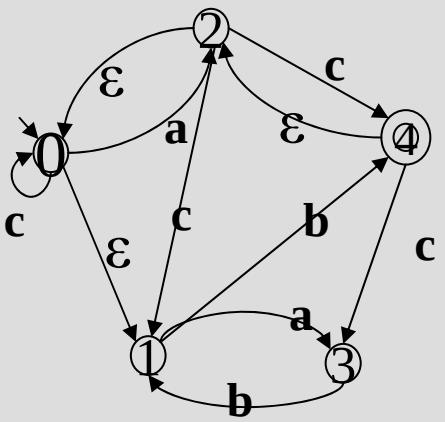
état	a	b	c	ε
0	2		0	1
1	3	4		
2			1,4	0
3		1		
4			3	2



Cas général : AFN → AFD

état	a	b	c
0,1	0,1,2,3	0,1,4,2	0,1
0,1,2,3	2, 0, 1, 3	4, 2, 0, 1	0,1,4, 2
0,1,2,4	2, 0, 1, 3	4, 2, 0, 1	0, 1, 4, 2, 3

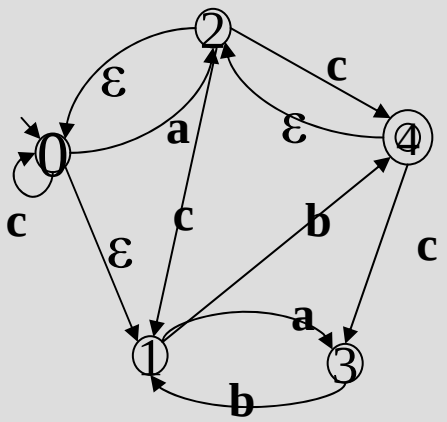
état	a	b	c	ε
0	2		0	1
1	3	4		
2			1,4	0
3		1		
4			3	2



Cas général : AFN → AFD

état	a	b	c
0,1	0,1,2,3	0,1,2,4	0,1
0,1,2,3	0,1,2,3	0,1,2,4	0,1,2,4
0,1,2,4	0,1,2, 3	0,1,2,4	0,1,2,3,4
0,1,2,3,4	2, 0, 1, 3	4, 2, 0, 1	0, 1, 4, 2, 3

état	a	b	c	ε
0	2		0	1
1	3	4		
2			1,4	0
3		1		
4			3	2



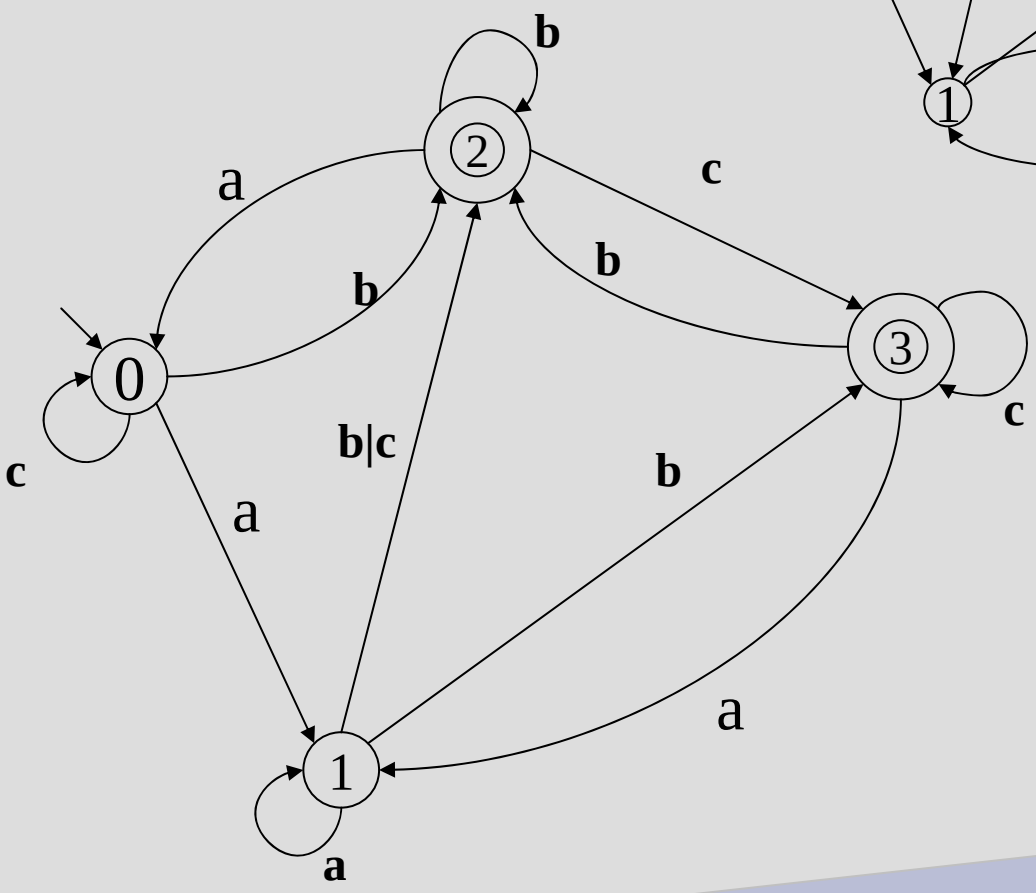
Cas général : AFN → AFD

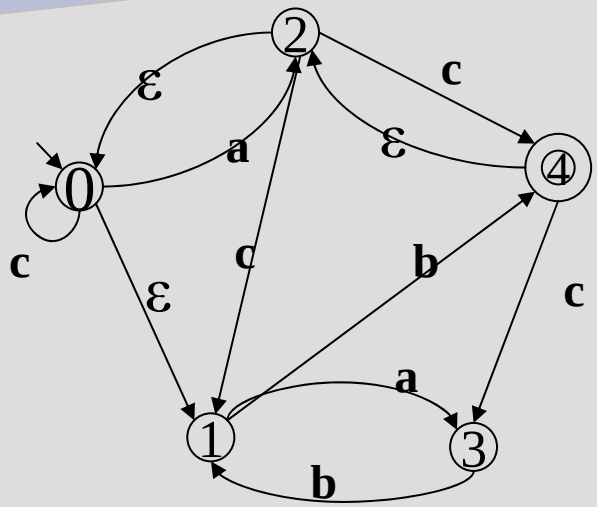
état	a	b	c
0,1 0	0,1,2,3 1	0,1,2,4 2	0,1 0
0,1,2,3 1	0,1,2,3 1	0,1,2,4 2	0,1,2,4 2
0,1,2,4 2	0,1,2, 3 1	0,1,2,4 2	0,1,2,3,4 3
0,1,2,3,4 3	0, 1, 2, 3 1	0, 1, 2, 4 2	0, 1 , 2, 3, 4 3

état	a	b	c	ε
0	2		0	1
1	3	4		
2			1,4	0
3		1		
4			3	2

- T={2,3}

Cas général : AFN → AFD



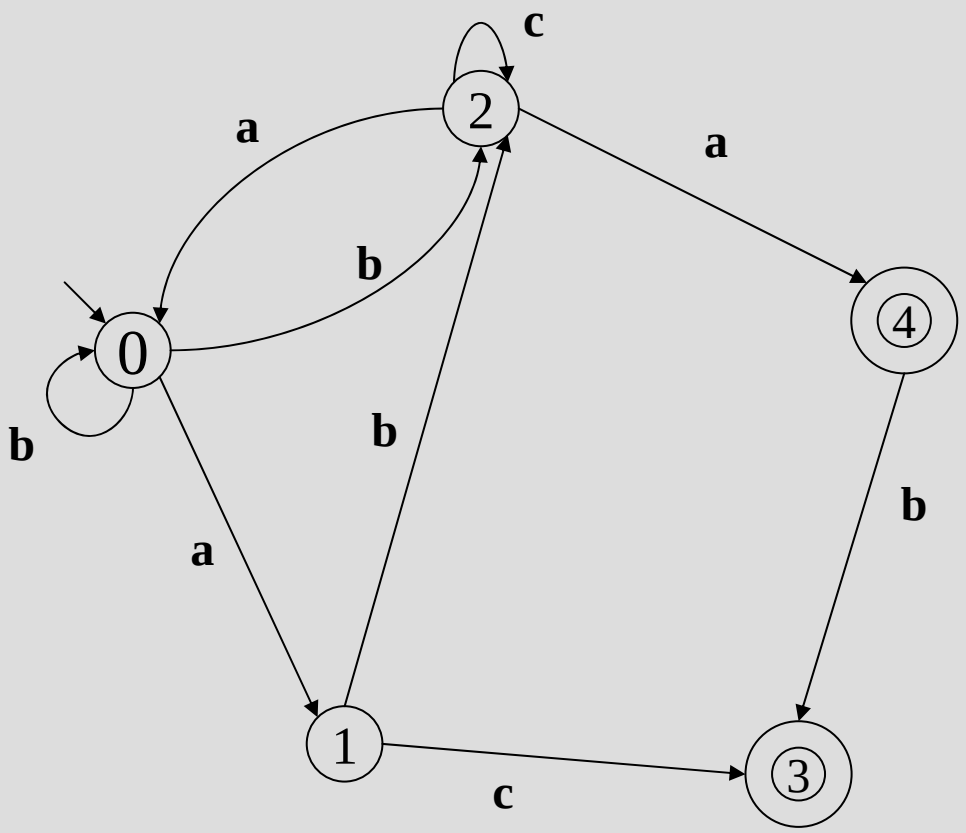


état	a	b	c
0	1	2	0
1	1	2	2
2	1	2	3
3	1	2	3

AEF → ER

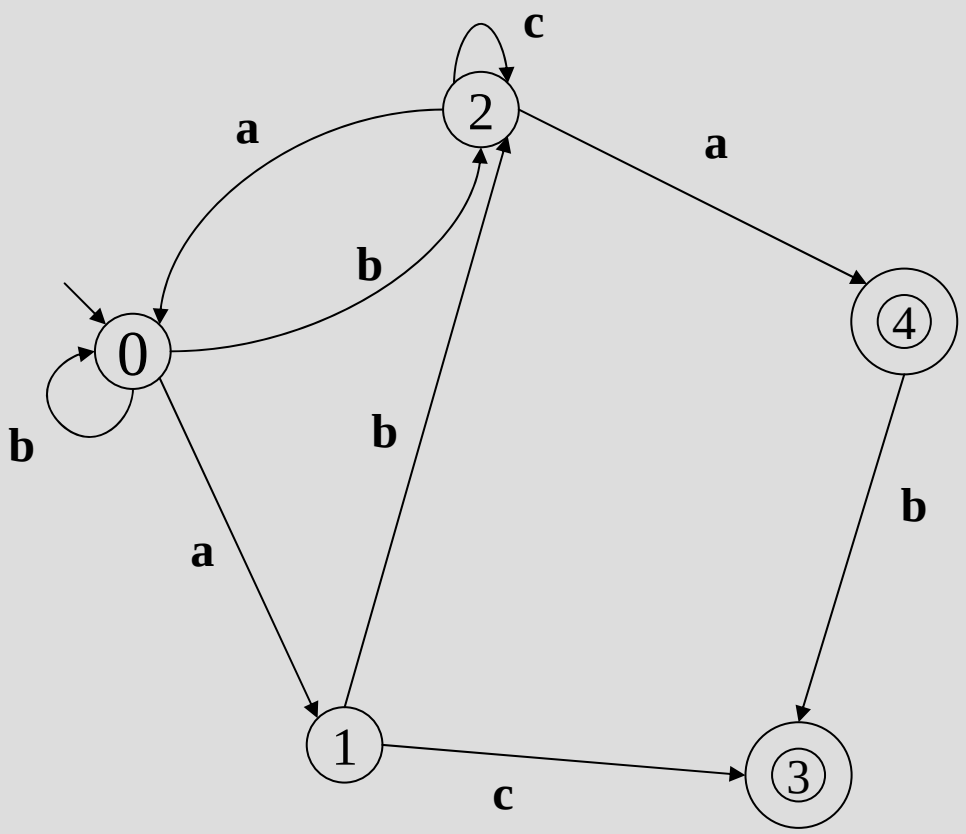
- L'automate d'état initial e_i reconnaît le langage L_i
- Système d'équation liant les L_i :
- $\forall\ \Delta(e_i,a) = e_j \Rightarrow L_i = a\ L_j$
- $\forall\ e_i \in T \Rightarrow L_i = \varepsilon$
- $L_i = \alpha$ et $L_i = \beta \Rightarrow L_i = \alpha|\beta$
- $L = \alpha\ L \mid \beta \Rightarrow L = \alpha^*\beta$ ($L = \alpha\ L \not\Rightarrow L = \alpha^*$: bouclage! $\Rightarrow$ erreur)

AEF → ER : Exemple



AEF → ER : Exemple

- $L_0 = b\ L_0 \mid a\ L_1 \mid b\ L_2$
- $L_1 = b\ L_2 \mid c\ L_3$
- $L_2 = a\ L_0 \mid c\ L_2 \mid a\ L_4$
- $L_3 = \varepsilon$
- $L_4 = \varepsilon \mid b\ L_3$



AEF → ER : Exemple

- $L_0 = b\ L_0 \mid a\ L_1 \mid b\ L_2$
- $L_1 = b\ L_2 \mid c\ L_3$
- $L_2 = a\ L_0 \mid c\ L_2 \mid a\ L_4$
- $L_3 = \varepsilon$
- $L_4 = \varepsilon \mid b\ L_3$

- $L_3 = \varepsilon$
- $L_4 = \varepsilon \mid b$
- $L_2 = c\ L_2 \mid a\ L_0 \mid a\ (\varepsilon \mid b)$
  $= c^*\ (a \mid ab \mid a\ L_0)$
- $L_1 = b\ L_2 \mid c$
  $= b\ c^*\ (a \mid ab \mid a\ L_0) \mid c$
- $L_0 = b\ L_0 \mid a\ L_1 \mid b\ L_2$
  $= b\ L_0 \mid abc^*(a|ab|aL_0)|ac$
  $|bc^*\ (a \mid ab \mid a\ L_0)$
  $= (b|aabc^*|abc^*)\ L_0 \mid$
  $abc^*(a|ab|ac|abc)$

$(b|aabc^*|abc^*)^* \mid bc^*(a|ab|ac|abc)$

Analyse lexicale

But : Reconnaître des classes de symboles :

- Un entier, un réel, un identificateur (java) , le mot-clé if
- Il est possible aussi, de préciser, par exemple, que l'on veut les constantes réelles ont un exposant optionnel à 2 chiffres

Analyse lexicale

C’est la première étape d’un compilateur.

L’analyseur lexical va fournir des symboles à l’analyseur syntaxique

- Soit sous forme de liste
- Soit par appels successifs next_token()

Analyse lexicale : difficultés

Soit l'entrée "2.3E5xy" , (exposant optionnel à 2 chiffres) et c le caractère courant :

- c=getChar() : '2' ⇒ a reconnu un ENTIER
- c=getChar() : '.' ⇒ tente de reconnaître un REEL
- c=getChar() : '3' ⇒ a reconnu un REEL
- c=getChar() : 'E' ⇒ tente de reconnaître un REEL
- c=getChar() : '5' ⇒ tente de reconnaître un REEL
- c=getChar() : 'x' ⇒ "2.3E5x" pas un REEL !
- return Token.REEL("2.3") : dernier symbole reconnu

mémorisation du dernier symbole reconnu

Analyse lexicale : difficultés

Soit l'entrée "if19"

- Mot-Cle-IF et Entier(19)

Ou

- Ident("if19")

Solutions :

- Reconnaissance du plus long fragment (préfixe)
- Séparateurs obligatoires
- Priorités (mots-clé prioritaires)
- …

Analyse lexicale :**Fonctionnement**

Un analyseur lexical ne fonctionne pas exactement comme un automate classique.

- L'automate :
- reconnaît un langage
- accepte ou rejette un mot

L'analyseur :

- découpe un mot en sous-mots (priorités, + long prefixe …)
- en associant un symbole a chaque sous-mot
- accepte ou rejette le mot

On parlera dans ce cours d'automate fonctionnant comme un analyseur lexical.

Analyse lexicale :**Fonctionnement**

État de reconnaissance d'un symbole = état final.

Tant qu'on peut transiter sur un caractère, on le fait.

Mémorisation du dernier état terminal traverse + sous-mot associe.

Quand on ne peut plus transiter :

- si état terminal :
  - émission du symbole associé ;
  - retour dans l'état initial.
- si état non terminal :
  - si état mémorisé : émission du symbole associé,
  - repositionnement éventuel de la tête de lecture
- sinon erreur.

Analyse lexicale : Définitions

- Une unité lexicale est une suite de caractères qui a une signification collective
- Un modèle est une règle associée à une unité lexicale qui décrit l'ensemble des chaînes d'une source qui peuvent correspondre à cette unité lexicale
- On appelle lexème toute suite de caractères d'une source qui concorde avec le modèle d'une unité lexicale

Analyse lexicale : Exemples

L'unité lexicale :

- opR (nom arbitraire) peut représenter les opérateurs relationnels >= <= < > …
- ID : toto, a, tab, … seraient des identificateurs (variables, types, fonction, procédures …)
- MtC : if, else, while … seraient les mots-clé

Le modèle :

- D'un identificateur pourrait être une lettre ou _ suivi de lettres ou _ ou chiffres
- D'un entier : toute suite non vide de chiffres précédés éventuellement de + ou –

>= <= < > … toto, a, tab, … if, else, while … sont des lexèmes

Analyse lexicale : Mise en Œuvre

- Le mécanisme de base pour décrire les unités lexicale est l'expression régulière.
- On a vu qu'elles étaient facilement transformable en AFN puis AFD puis AFD simplifiés : les transformations faites à la main dans la partie précédente peuvent être facilement programmés.
- Des analyseurs lexicaux tels Lex et FLex en langage C et JLex et JFLex pour java réalisent ce travail.

Analyse lexicale : Expression régulière (exemples)

Le chiffre 1 : 1

- Un chiffre quelconque : 0|1|2|3|4|5|6|7|8|9
- Un nombre : 0|(1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)*
- Éventuellement un point suivi d'un nombre : ε|.(0|1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)*
- Un exposant éventuel : ε|E(0|1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)
- Les constantes entières et réelles : (0|1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)*(|.(0|1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)*(|E(0|1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)))

Analyse lexicale : Expression régulière (lourdeur)

besoin d'augmenter le confort de spécification ;

- lisibilité ;
- concision.

sans toucher à l'expressivité (garder un langage régulier).

- ➔ classes de caractères et descriptions régulières.

Pour rassembler et nommer des ensembles de caractères. Ex :

- ch = [0-9]
- ca = [a-zA-Z]
- Ce qui donne pour les nombres exprimés précédemment :
- ch ch* (ε |.ch ch* (ε | E ch ch))

Analyse lexicale : Description régulière

Descriptions régulières ou définitions régulières

- Pour nommer des expressions régulières et s'en resservir :
- intconst = ch ch*
- realconst = intconst.intconst(E ch ch|)

Formellement : suite de définitions

- N1 = E1
- . . .
- $N_n = E_n$
- ou :
- les N_i sont des noms distincts 2 à 2 ;
- les E_i sont des expressions régulières sur $\Sigma \cup \{N_1,\ldots,N_{i-1}\}$.
- Pas de descriptions récursives.

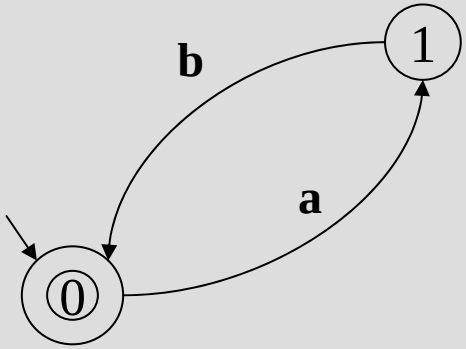
Analyse lexicale : Détection d'erreurs

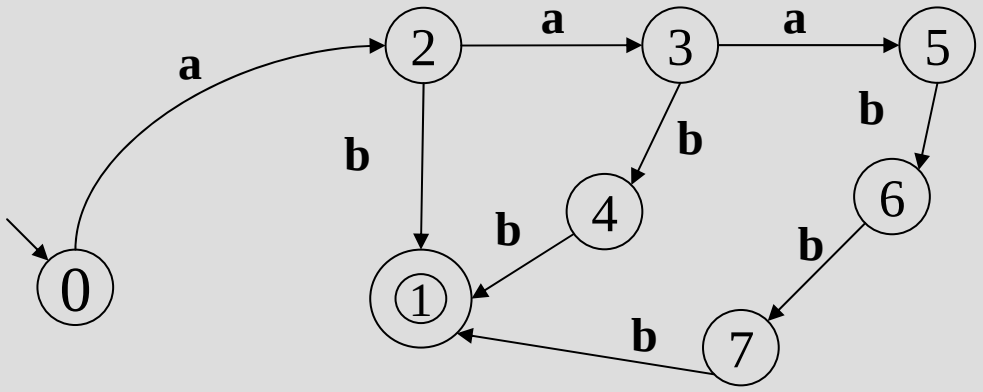
- Peu d'erreurs sont détectables à ce niveau : les seules détectables sont les suites de caractères ne correspondant à aucun modèle d'unité lexicale.
- Stratégie en cas de détection :
  - Mode panique (avertir et ignorer les caractères qui posent problème)
  - Correction d'erreur
  - Arrêt immédiat
- Ex : esle faute de frappe pour else ou identificateur

1a identificateur a1 ou a ou constante 1 …

Description régulière : Limitation

- Le théorème de Kleene affirme qu'un langage est rationnel si et seulement s'il est reconnu par un automate fini
- Malheureusement, tous les langages ne sont pas rationnels par exemple
  « autant de a que de b » ne sont pas tous rationnels si $(ab)^n$ l'est a^nb^n ne l'est pas





a^nb^n avec n<4 l'est, mais pas dans le cas général

Description régulière : Limitation

- Sur l'exemple précédent (que l'on peut rapprocher du cas concret autant de parenthèses ouvrantes que fermantes) on sent bien intuitivement qu'il faudrait pour $a^n b^n$ ($\forall n$) un nombre indéterminé d'états (de « mémoires »)
- Pour une démonstration plus formelle, on utilisera le Lemme de l'étoile encore appelé Lemme de pompage (hors cours)