6. Automates à états finis (AEF)

Bernard ESPINASSE

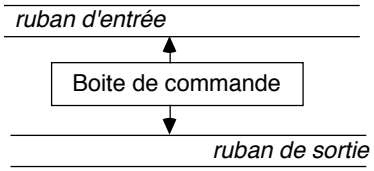
Professeur à l'Université d'Aix-Marseille

1998

- **Introduction : notion d'automate**
- **Présentation générale des automates à états finis (AEF)**
- **Représentations d'un AEF**
- **Langage reconnu par un AEF**
- **AEF déterministes et non déterministes, déterminisation**
- **Expression régulières et théorème de Kleene**
- **Passage d'une grammaire régulière à une expression régulière**
- **Passage d'une grammaire régulière à un AEF**
- **Passage d'une expression régulière à un AEF**
- **Algorithme d'analyse d'un AEF déterministe**

Introduction: notion d'automate

- **"Machine" symbolique** validant l'appartenance d'une chaîne donnée au langage qu'il décrit
- Boite de commande correspondant par une **tête de lecture-écriture** avec une **mémoire infinie** symbolisée par un **ruban** :



- n'est pas caractérisé par sa constitution elle-même mais son **fonctionnement** défini par :
  - des **configurations de l'automate**
  - la ou les **configurations succédant à toute configuration**
  - les configurations **initiales** ou **finales**
- l'automate est **complètement défini** quand on connaît **:**
  - la liste des **états possibles**
  - les **règles de transition d'un état à un autre**

Automates à états finis (AEF)

- associés aux **grammaires régulières**
- reconnaissent une chaîne en passant par des états successifs qui dépendent des symboles lus dans la chaîne
- la chaîne est reconnue par l'automate ssi celui-ci part d'un état initial, lit tous les symboles de la chaîne, puis s'arrête dans un état final
- on se limite aux automates n'ayant qu'un **seul état initial**

Automate à états fini = quintupler A = (V, Q, I, F, μ) avec:

- **V** = **vocabulaire**, un ensemble fini, non vide,
- **Q** = **ensemble d'états**, fini, non vide,
  - Q contient un état particulier **I** : l'**état initial**,
  - Q contient un sous-ensemble d'états particuliers **F**: les **états finals**,
- **μ** = relation de Q x (V ∪ {∧}) vers Q = **fonction de transition de l'automate**

$$\mu(q, a) = q'$$

- si l'automate est dans l'état **q**, si il rencontre le symbole **a** alors il passe dans l'état **q'**
- ∀ q de Q, μ(q, ∧) = q.

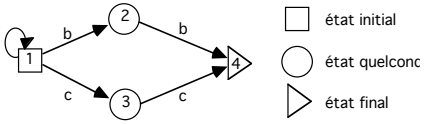
Représentation d'un AEF

- Graphe orienté valué :
  - **arcs** portent des **symboles** du vocabulaire
  - les **noeuds** portent des **états**
- Soit l'automate A = (V, Q, I, F, μ) avec :
  - **V** = (a, b, c)
  - **Q** = {1, 2, 3, 4}
  - **I** = 1
  - **F** = {4}.
  - **μ** = fonction de transition, définie par :

    μ(1, a) = 1 ; μ(1, b) = 2 ; μ(1, c) = 3 ; μ(2, c) = 4 ; μ(3, b) = 4.

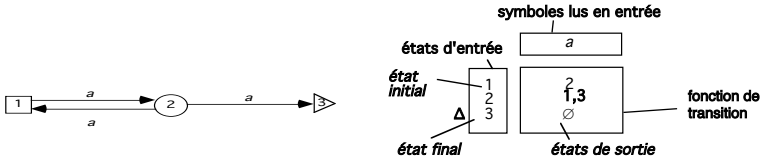
Convention de notation graphique

- les **états terminaux** sont notés par des **triangles**
- l'**état initial** par un **carré**
- un **état quelconque** sera représenté dans un **cercle**
- Si l'état initial est terminal, le carré et le triangle sont superposés.

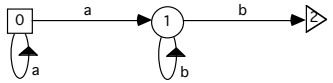


Représentation tabulaire d'un AEF

Exemple 1 :



Exemple 2 :

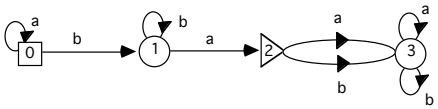


Δ	a	b
0	**0, 1**	
1		**1, 2**
2		

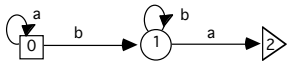
Exemple d'AEF

Soit l'automate A = (V, Q, I, F, μ) avec :

- **V** = (a, b)
- **Q** = {0, 1, 2, 3}
- **I** = 0
- **F** = {2}.
- **μ** = fonction de transition, définie par :
  μ(0, a) = 0 ; μ(0, b) = 1 ; μ(1, a) = 2 ; μ(1, b) = 1 ;
  μ(2, a) = 3 ; μ(2, b) = 3 ; μ(3, a) = 3 ; μ(3, b) = 3 ;



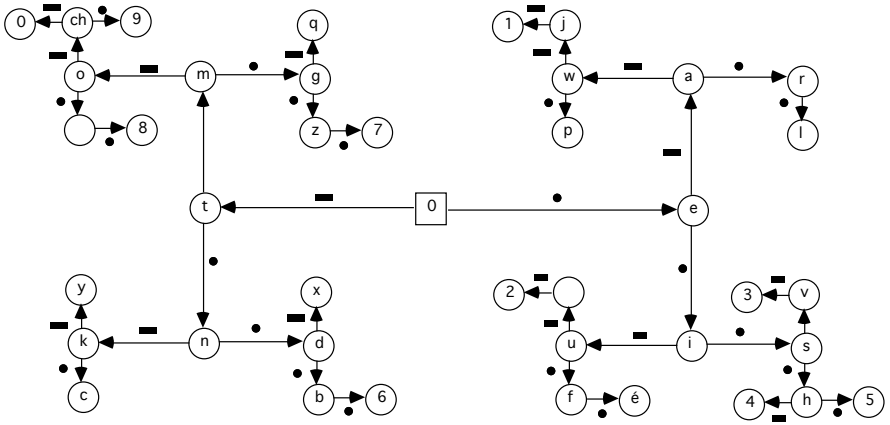
- langage reconnu par cet AEF = {mots} constitués d'une suite de **a** (évent.vide) suivie d'une suite de **b** (contenant au moins un b) et se terminant par un **a**
- **état 3 =** état **"puits"** ou **"inutile"**: tous les mots amenant au cours de leur reconnaissance à 3 ne sont pas reconnus (impossible d'en sortir) : **peut être supprimé => nouvel automate A':**



A et A' reconnaissent le même langage

Exemple d'AEF

- **l'automate reconnaissant si une suite de • et de – représente un caractère du langage Morse :**



- tous les états sont finals, sauf 0 et les états blancs

Langage reconnu par un AEF

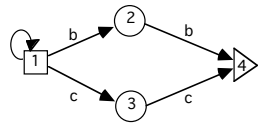
- **langage reconnu par un AEF =** {chaînes} acceptées à l'aide d'une dérivation I- définie à partir de **μ** .
- chaque dérivation consiste à effectuer une transition correcte :

$$\forall\, q \in Q, \forall a \in V, \forall x \in V^*, (q, ax)\ \text{I-}\ (q', x) \quad \Leftrightarrow \quad q' \in \mu(q, a)$$

- **le langage accepté par un AEF** = {chaînes} dont les symboles font passer de l'état initial de l'automate jusqu'à un de ses états finals par une succession de transition utilisant tous ces symboles dans l'ordre :

$$L(A) = \{x \in V^* \text{ tels que } (q_i, x)\ \text{I-}^*\ (q, \wedge), q \in F\}$$

Exemple



- **langage reconnu** : **bc, cb, abc, acb, aabc, aacb, aaabc, aaacb, ...**
  = le produit du langage {a} (le monoïde des chaînes finies formées de symboles a) avec le langage {bc, cb}*
- on le note par l'**expression régulière** : **a*bc + a*cb** ou **a*(bc + cb)**

AEF déterministes

= automate tel que sa fonction de transition soit une fonction, c'est-à-dire tel que:

- **pour tout état et tout symbole,**
- **il existe au plus un état dans lequel l'automate peut passer**

AEF non déterministe

= automate tel qu'**il existe au moins un couple**, formé d'un état et d'un symbole, qui **admette plus d'une image par la fonction de transition** :

alors :

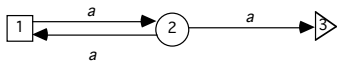
- l'automate doit faire des choix pour progresser
- il reconnaît une phrase si parmi tous les choix possibles pour l'analyser, il en existe au moins un qui le laisse dans un état final

Inconvénient :

- **analyse les chaînes plus lentement que les automates déterministes** : si un mauvais choix est pris dans une transition, **il faut revenir en arrière** jusqu'à ce choix pour parvenir à la reconnaissance de la chaîne.
- **complexité de la reconnaissance d'une chaîne est dite exponentielle** : à chaque état et pour chaque symbole, existent **n transitions possibles**, la reconnaissance d'une chaîne de longueur **p** demandera jusqu'à **n^p** transitions.

Exemple d'AEF non déterministe

Soit l'automate suivant :



- **non déterministe**, car : **μ(2, a) = {1, 3}**
- langage reconnu par cet automate est : **(aa)* aa**

Par exemple :

reconnaissance de la chaîne **aaaa** :

- Au premier passage à l'état 2, deux possibilités : aller en 3 ou en 1.
- si on passe en 3: on est dans un état final d'où l'on ne peut plus effectuer de transition. Or la chaîne n'est pas encore entièrement lue, dans ce cas, puisque seuls les 2 premiers caractères ont été traités ⇒ retour à l'état 2 et choisir de passer à l'état 1
- lors du passage suivant à l'état 2, il faut également effectuer le choix correct de passer à l'état 3

Passage d'un AEF non déterministe à un AEF déterministe

La **déterminisation** d'un AEF ou **passage d'un AEF non déterministe à un AEF déterministe** reconnaissant le **même langage** se fait par l'algorithme :

- **étape 1** : suppression des **cycles vides**,
- **étape 2** : suppression des autres **transitions vides**,
- **étape 3** : **créations de nouveaux états** pour éliminer des choix multiples,
- **étape 4** : **réduction** de ce dernier automate pour obtenir l'automate minimal équivalent.

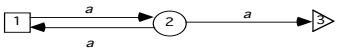
Étapes 1 et 2 : remplacer toute transition vide par une transition demandant un symbole en modifiant les transitions arrivant sur l'état d'où part la transition vide

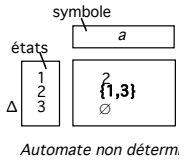
L'étape 3 : créer pour le nouvel automate autant d'états que d'ensembles d'états pour l'automate précédent

Étape 4 : il existe des algorithmes pour la réaliser :

- s'assurer que tous les états qu'il comporte sont bien atteints à partir de l'état initial
- diminuer le nombre de ses états en supprimant les chemins distincts réalisant la même reconnaissance.

AEF non déterministe

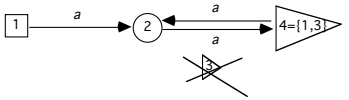


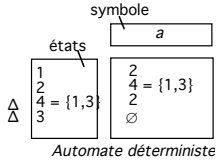


Automate non déterm

AEF déterministe minimal équivalent

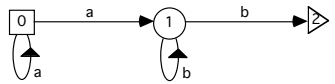
- les transitions s'obtiennent à partir des transitions de l'automate d'origine :
  - **création de nouveaux états** pour éliminer les choix multiples : 4 = {1, 3}
  - **définition des nouvelles transitions associées** : ici la transition par le symbole ***a*** à partir de l'état 4 = {1, 3} = cumul des états atteints par la transition à partir de 1 ou 3 et acceptant ***a***, soit l'état 2 (à partir de 1) et une transition vide (à partir de 3) soit ici l'état 2
  - **définition des états finals du nouvel automate** : tous les états qui contiennent au moins un état final de l'automate initial





Automate déterministe

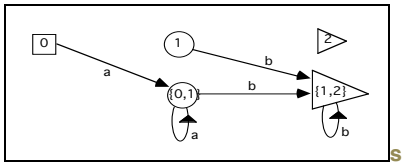
AEF non déterministe



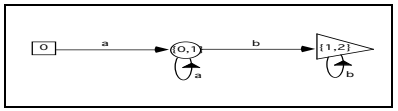
Δ		a	b
	0	**0, 1**	
	1		**1, 2**
	2		

AEF déterministe minimal équivalent

Δ		a	b
	0	{0, 1}	
	1		{1, 2}
	2		
Δ	{0, 1}	{0, 1}	{1, 2}
Δ	{1, 2}		{1, 2}



soit:



Expressions régulières (rappels)

- Langages réguliers définis à partir des **grammaires régulières**
- Les expressions régulières fournissent une **autre méthode de définition des langages réguliers**
- Chaque expression régulière décrit un **ensemble de chaînes**
- Le formalisme utilisé **n'est pas celui des règles de dérivation**, mais un ensemble de trois opérations de priorités décroissantes :
  - la **concaténation**

    est la mise bout à bout des chaînes de deux expressions
  - la **fermeture** notée ***

    est l'ensemble des chaînes obtenues en concaténant n fois

    les chaînes d'une expression (n ∈ N)
  - l'**alternative** notée **+**

    est le choix entre les chaînes de 2 expressions.

Les langages décrits par ces expressions sont les mêmes que ceux décrits par les grammaires régulières

Définition d'une expression régulière (rappels)

Les **expressions régulières sur un vocabulaire X*** sont définies par :

- **∧ : le mot vide** et **∅ l'ensemble vide** (∧ et ∅ sont des expressions régulières)
- **∀ a ∈ X*, a** est une expression régulière,
- **si e et e'** sont des expressions régulières, alors :

e.e', e* , (e+e') = expressions régulières

- on appelle R(X*) l'ensemble des expressions régulières définies sur un vocabulaire X*.
- l'ensemble des chaînes associé à une expression régulière e se note E(e)

On a:

- E(∧) = {∧ } ; E(∅) = ∅
- ∀ a ∈ X*, E(a) = {a}
- ∀ e, e' ∈ R(X*), E(e+e') = E(e) ∪ E(e')
- ∀ e, e' ∈ R(X*), E(ee') = {c'' = cc', c ∈ E(e)c' ∈ E(e')}
- ∀ e ∈ R(X*), E(e*) = {c^n , n ∈ N, c ∈ E(e)}

Exemple

a*b + b*a = {b, ab, aab, aaab, ... a, ba, bba, bbba...}

(a+ b)* = (^, a, b, aa, ab, ba, bb ... }

ab* (c+a) = {ac, aa, abc, aba, abbc, abba...}

Passage d'une Expression Régulière à une Grammaire Régulière (ER ->GR)

- en suivant pas à pas la construction par **concaténation (.)** des expressions régulières et
- en créant un appel récursif pour chaque **fermeture (*)**
- en créant autant de règles qu'il y a d'alternatives dans les cas de **disjonction (+)**

Exemple 1

ER = 0 (0+1)*0

GR associée :

S → 0A
A → 0A
A → 1 A
A → 0

Exemple 2

ER = (0+1)* 0 (0+1) (0+1)

GR associée :

S → 0S
S → 1S
S → 0A
A → 0B
A → 1B
B → 0
B → 1

Expressions Régulières et théorème de Kleene

- **Théorème de KLEENE**
  - **Pour tout AEF, il ∃ une Expression Régulière qui représente le langage reconnu.**

  et
  - **Pour toute Expression Régulière, il ∃ un AEF qui reconnaît le langage représenté**

  soit : AEF ⇔ ER

- **Corollaire**
  - l'ensemble des langages réguliers (ou des expressions régulières) est l'ensemble des langages reconnus par les AEF
  - on suppose ici, que les langages réguliers peuvent reconnaître le mot vide en ajoutant éventuellement un nouvel axiome S' et 2 nouvelles règles :

S' → S et **S' → ∧**

Passage d'une Grammaire Régulière à un Automate à États Fini (GR -> AEF)

- Se fait directement **en associant une transition à chaque règle de la grammaire**.
- L'automate à états fini ainsi défini n'est pas nécessairement déterministe ou minimal.

⇒ un travail doit être effectué sur l'automate pour le rendre optimal

- Soit une Grammaire Régulière **G**, l'Automate à États Fini **(V, Q, I, F, μ)** associé est défini par :
  - **V** (vocabulaire) = l'ensemble des **terminaux** de la grammaire
  - **Q** (ensemble des états) = ensemble des non terminaux de la grammaire et d'un état final supplémentaire **f**
  - **I** (l'état **initial)** = l'état qui représente **l'axiome**
  - **F** (l'état **final)** = **f**
  - **μ** (fonction de transition) est définie par :
    - **B ∈ μ(A, a) pour toute règle A → aB**
    - **f ∈ μ(A, b) pour toute règle A → b**

Exemple 1 de passage GR -> AEF

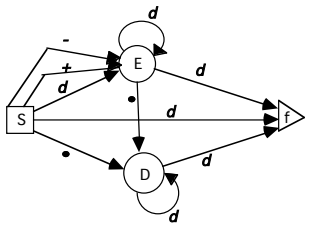
Soit la grammaire régulière dont les règles sont :

S → + E I - E I d I dE I • D avec X_T = {+, -, •, d} et X_N = {S, E, D}

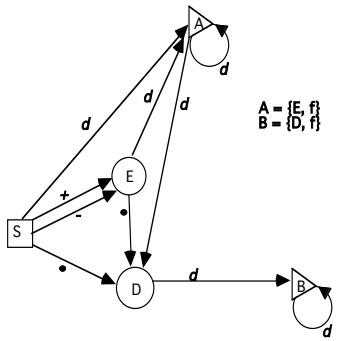
E → E I d I • D

D → dD I d

- automate **non déterministe** reconnaissant le même langage que cette grammaire:



- automate **déterministe** correspondant:



Exemple 2 de passage GR -> AEF

Soit la grammaire régulière dont les règles sont :

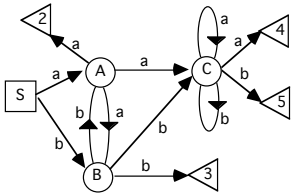
S → aA I bB avec X_T = {a, b} et X_N = {S, A, B, C}

A → bB I aC I a

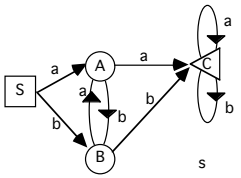
B → aA I bC I b

C → aC I a I b I bC

Automate 1:
par transposition pas à pas la grammaire en un automate:



Automate 1 simplifié :
en transformant l'état C en un état terminal:



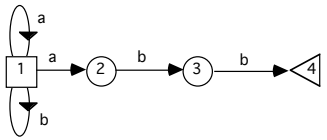
Passage d'une Expression Régulière à un Automate à États Fini (ER -> AEF)

La construction de l'automate se fait :

- par enchaînement des transitions pour la **concaténation (.)**,
- par transitions multiples à partir d'un état donné pour la **disjonction (+)**
- et par "bouclage" sur un état antérieur pour la **fermeture (*)**.

Exemple de passage ER -> AEF

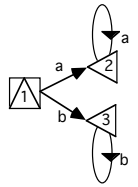
- Soit l'expression régulière **E = (a + b)* abb**
- E décrit les chaînes finies formées de **a** et de **b** se terminant par **abb**.
- Automate associé :



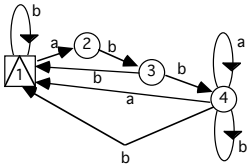
Exemples de passage ER -> AEF

1) E = (a+ b)*

2) E = a* + b*



3) E = ((ʌ + b)* abb(a + b)*)*

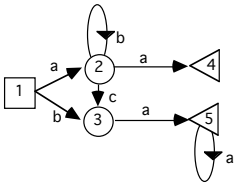


Passage d'un Automate à États Fini à une Expression Régulière (AEF -> ER)

- aux **boucles** sont associées des **fermetures (*)**,
- aux **successions de transition** des **concaténations (.)**
- et aux **transitions multiples issues d'un même état** sont associées des **disjonctions (+)**

Exemple de passage AEF -> ER

- Soit l'automate à états fini suivant :



- Une expression régulière décrivant le langage reconnu par l'automate précédent est donnée par : **a b* a + a b* c a a* + b a a***

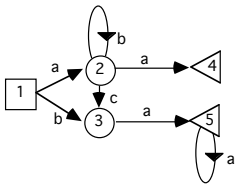
Passage d'un Automate à États Fini à une Grammaire Régulière (AEF -> GR)

encore plus immédiat :

- à **chaque transition** est associée **une règle de grammaire**
- les **états terminaux** nécessitent éventuellement un **doublement des règles** pour rendre compte de l'arrêt possible en ce point.

Exemple de passage AEF -> GR

- Soit l'Automate à États Fini suivant :

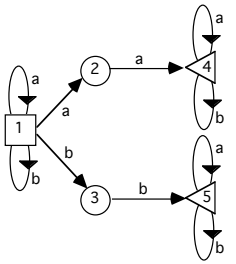


- Grammaire Régulière associée :

S → aA	A → bA	A → a	
S → bB	A → cB	B → aB	B → a

Exemple de passage AEF -> GR

• Soit l'Automate à États Fini suivant :



• Grammaire Régulière associée :

S → aS	S → bS	S → aA	S → bB
A → aC	A → a	C → aC	C → bC
C → a	C → b	B → b	B → bC

Représentation matricielle d'un AEF déterministe

• les transitions d'un automate à états fini déterministe peuvent être représentées par :

  • un tableau à double entrée :

    • en **ligne : les états de l'automate**

    • en **colonne : les symboles du vocabulaire**

• l'automate déterministe précédent se représente par la matrice :

Symboles :	+	-	•	*d*
États : S	E	E	D	A
E	∅	∅	D	A
D	∅	∅	∅	B
A	∅	∅	D	A
B	∅	∅	∅	B

où ∅ représente les transitions impossibles

• matrice généralement très **creuse**

• pour en diminuer la taille, on représente les transitions par un **tableau à 3 colonnes** :
état initial, symbole lu et état final.

Algorithme d'analyse d'un AEF déterministe

Cette représentation permet de donner l'algorithme général de l'analyseur correspondant à un automate à états fini déterministe :

```
Lire (Chaîne)
État <- S;
Tant que Chaîne ≠ ^ et État ≠ ∅
faire
   Symbole <- symbole suivant de Chaîne;
   État <- Matrice[État, Symbole]
fin Tant Que;
si État = ∅
Alors
   Chaîne n'appartient pas au langage;
Sinon
   Si Final(État)
   Alors
      Chaine appartient au langage;
   Sinon
      Chaine n'appartient pas au langage;
   fin Si;
fin Si.
```