Automates finis [AF3] - Notes de cours 5
D'une expression rationnelle à un automate fini
Automates finis non déterministes avec ϵ-transitions (ou asynchrones)
Algorithme de Thompson

Jean-Marie Rifflet - Novembre 2005

Nous présentons ici la méthode proposée par Ken Thompson pour construire un automate reconnaissant le langage représenté par une expression rationnelle donnée. C'est cette méthode qui est utilisée par la commande `grep` d'Unix.
Elle repose sur l'utilisation d'un type de machines à nombre finis d'états et non déterministes, au sens que nous avons déjà défini, mais qui ont de plus la possibilité de changer d'état sans lecture de lettres : ces autoamtes sont appelés automates finis avec ϵ-transitions (ϵ-AFND) ou automates finis asynchrones.
Après avoir présenté la construction de Thompson pour associer un tel automate à une expression rationnelle, nous verrons comment il est possible de construire un automate déterministe équivalent à un automate de ce type.

1. Les automates finis asynchrones ou ϵ-AFND

1.1. Définition

Un automate fini asynchrone ou ϵ-AFND est un quintuplet $\mathcal{A} =< Q, X, \delta, Q_0, Q' >$ qui ne diffère d'un AFND que dans la définition de sa fonction de transitions :

$$\delta : Q \times (X \cup \{\epsilon\}) \to \mathcal{P}(Q)$$

Les couples (q, ϵ) pour lesquels la fonction δ est définie sont appelées **ϵ-transitions**.

Remarque : il peut s'avérer pratique de distinguer les restrictions de la fonction δ à $Q \times X$ (correspondant à des transitions avec lecture effective de lettre) et à $Q \times \{\epsilon\}$ (correspondant aux ϵ-transitions).
Dans la suite, nous désignerons par δ_X la première et par δ_ϵ la seconde.

Un mot w est reconnu par un ϵ-AFND $\mathcal{A}$ si et seulement si, partant d'un état initial, il est possible d'atteindre l'un des états terminaux après avoir lu toutes les lettres du mot par une suite de transitions autorisées (les lettres du mot étant lues dans l'ordre et des séquences d'ϵ-transitions pouvant entrecouper les transitions avec lecture de lettres du mot).

1.2. Exemple

Considérons l'automate $\mathcal{A}_1$ dont les composants sont les suivants :

- ensemble Q d'états : $\{1, 2, 3, 4, 5, 6, 7\}$
- alphabet X : $\{a, b, c\}$
- ensemble d'états initiaux Q_0 : $\{1\}$
- ensemble d'états terminaux Q' : $\{4, 7\}$
- fonction de transitions δ définie par la table suivante et ses deux restrictions δ_ϵ et δ_X associées :

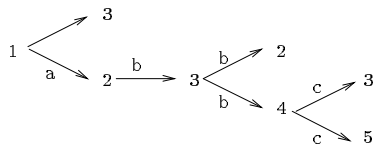
δ	ϵ	a	b	c
1	$\{3\}$	$\{2\}$	$\emptyset$	$\emptyset$
2	$\emptyset$	$\{2\}$	$\{3\}$	$\emptyset$
3	$\emptyset$	$\emptyset$	$\{2,4\}$	$\emptyset$
4	$\emptyset$	$\emptyset$	$\emptyset$	$\{3,5\}$
5	$\emptyset$	$\emptyset$	$\emptyset$	$\{6\}$
6	$\{3\}$	$\{7\}$	$\emptyset$	$\emptyset$
7	$\emptyset$	$\emptyset$	$\{5\}$	$\emptyset$

δ_ϵ	ϵ
1	$\{3\}$
2	$\emptyset$
3	$\emptyset$
4	$\emptyset$
5	$\emptyset$
6	$\{3\}$
7	$\emptyset$

δ_X	a	b	c
1	$\{2\}$	$\emptyset$	$\emptyset$
2	$\{2\}$	$\{3\}$	$\emptyset$
3	$\emptyset$	$\{2,4\}$	$\emptyset$
4	$\emptyset$	$\emptyset$	$\{3,5\}$
5	$\emptyset$	$\emptyset$	$\{6\}$
6	$\{7\}$	$\emptyset$	$\emptyset$
7	$\emptyset$	$\{5\}$	$\emptyset$

Comme c'est le cas des AFD et des AFND, il est commode de représenter graphiquement les ϵ-AFND. Ainsi l'automate $\mathcal{A}_1$ précédent peut être représenté par le graphe :

- une dernière situation se présente avec le mot *abbc* pour lequel on a les calculs possibles suivants :



L'automate peut lire le mot dans sa totalité de différentes manières, c'est-à-dire par des suites de transitions différentes, mais aucune d'elles ne le conduit à un état terminal : les deux suites de transitions lisant tout le mot conduisent l'une à 3 et l'autre à 5 qui ne sont ni l'un, ni l'autre terminal. Le mot n'est donc pas reconnu.

1.3.2 Configuration d'un ϵ-AFND

Nous donnons une définition du calcul d'un tel automate en terme de suites de configurations.

Une **configuration** d'un ϵ-AFND est un couple $(Q, m) \in \mathcal{P}(Q) \times X^*$. Une telle configuration exprime le fait qu'on souhaite réaliser un calcul à partir d'un ensemble d'états Q en lisant le mot m.

On dit que la configuration $C_2 = (Q_2, m_2)$ **se déduit directement** de $C_1 = (Q_1, m_1)$ dans l'automate $\mathcal{A}$, (ce qu'on note $C_1 \vdash_{\mathcal{A}} C_2$ ou $C_1 \rightarrow_{\mathcal{A}} C_2$) si et seulement si :

- ou bien $m_1 = m_2$ et $\exists q_1 \in Q_1$ tel que $Q_2 = Q_1 \cup \delta(q_1, \epsilon)$
- ou bien $\exists x \in X$ telle que $m_1 = x.m_2$ et $Q_2 = \bigcup_{q \in Q_1}^{x \in X} \delta(q, x)$

Une configuration C **se déduit** d'une configuration C' ($C \overset{*}{\vdash}_{\mathcal{A}} C'$ ou $C \overset{*}{\rightarrow}_{\mathcal{A}} C'$) si et seulement si $\exists\ C_1, C_2, \ldots, C_{n-1}$, configurations telles que

- $C = C_1$,
- $C' = C_n$
- $\forall i = 1, \ldots, n-1$, $C_i \vdash_{\mathcal{A}} C_{i+1}$.

La relation $\overset{*}{\vdash}_{\mathcal{A}}$ est donc la fermeture réflexive et transitive de la relation $\vdash_{\mathcal{A}}$.

Un mot m est alors **reconnu par l'ϵ-AFND** $\mathcal{A}$ si et seulement si il existe une configuration (Q, ϵ) telle que $Q \cap Q' \neq \emptyset$ se déduisant de la configuration initiale (Q_0, m) (c'est-à-dire telle que $(Q_0, m) \overset{*}{\vdash}_{\mathcal{A}} (Q, \epsilon)$).

1.4. ϵ-AFND normalisés

1.4.1. Définition

Un ϵ-AFND est dit normalisé s'il possède les propriétés suivantes :

- il possède un seul état initial : $Q_0 = \{q_0\}$;
- il possède un seul état final : $Q' = \{q_f\}$;
- sa fonction de transitions δ satisfait les deux propriétés suivantes : de sa fonction de transitions δ ont les propriétés suivantes :
  - $\forall q \in Q, \forall a \in X \cup \{\epsilon\}$, on a : $q_0 \notin \delta(q, a)$.
    En d'autres termes, dans le graphe correspondant à l'automate, aucun arc n'a q_0 comme extrémité terminale ;
  - $\forall a \in X \cup \{\epsilon\}$, $\delta(q_f, a) = \emptyset$.
    En d'autres termes, dans le graphe correspondant à l'automate, aucun arc n'a q_f comme origine (extrémité initiale).

1.4.2. Normalisation d'un ϵ-AFND

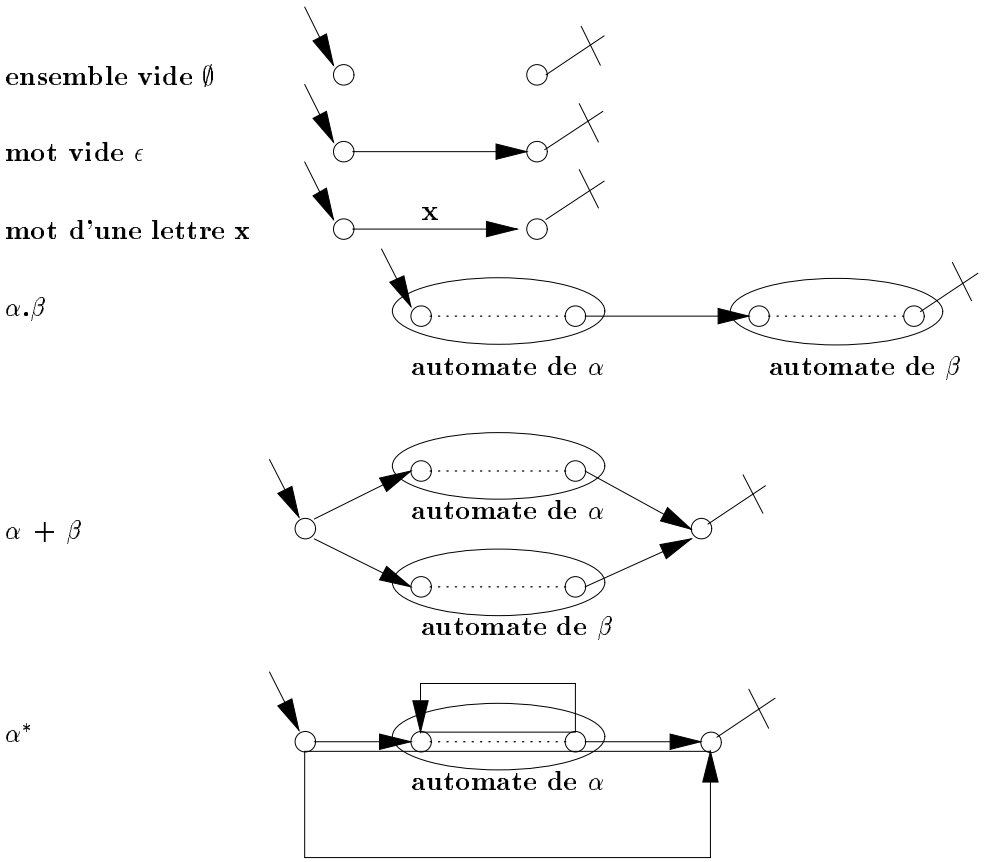
Pour tout ϵ-AFND $\mathcal{A}$, il est facile de construire un ϵ-AFND normalisé $\mathcal{B}$ qui lui est équivalent. La structure de $\mathcal{B}$ étant par ailleurs identique à celle de $\mathcal{A}$), il suffit de :

1. ajouter deux nouveaux états q_- et q_+ qui seront respectivement le seul état initial et le seul état final de $\mathcal{B}$;
2. rajouter, pour chaque état initial q de l'automate $\mathcal{A}$, une ϵ-transition de q_- vers q ;
3. rajouter, pour chaque état terminal q de $\mathcal{A}$, une ϵ-transition de q vers q_+.

2. AFND$_\epsilon$ normalisé associé à une expression rationnelle

2.1 La construction

Étant donnée une expression rationnelle E, un AFND$_\epsilon$ normalisé $\mathcal{A}$ reconnaissant le langage $\mu(E)$ (c'est-à-dire tel que le langage reconnu par l'AFND$_\epsilon$ $\mathcal{A}$ soit exactement le langage $\mu(E)$ représenté par l'expression rationnelle E) peut être construit par le procédé standard suivant, en s'appuyant sur la structure de l'expression E :



2.2. Propriétés de l'AFND$_\epsilon$ obtenu

L'AFND$_\epsilon$ obtenu par la construction précédente possède, outre le fait qu'il soit normalisé, quelques propriétés supplémentaires qui peuvent s'avérer intéressantes dans la perspective de sa représentation en machine :

1. il possède exactement $2n$ états où n est le nombre d'occurrences dans l'expression E de symboles appartenant $X \cup \{+ *\}$;

2. pour un état donné q différent de l'état terminal q_f, il y a deux possibilités :

   (a) **ou bien** $\delta_\epsilon(q) = \emptyset$ et dans ce cas il existe une et une seule lettre x telle que $\delta_X(q, x)$ soit définie et $\mathrm{Card}(\delta_X(q, x)) = 1$.
   En d'autres termes, pour un tel état, il n'y a pas d'ϵ-transition possible et il n'existe qu'une seule transition possible avec lecture effective de lettre.

   Il est intéressant d'observer que le nombre d'états ayant cette propriété est exactement égal au nombre d'occurrences de lettres de X dans l'expression E ;

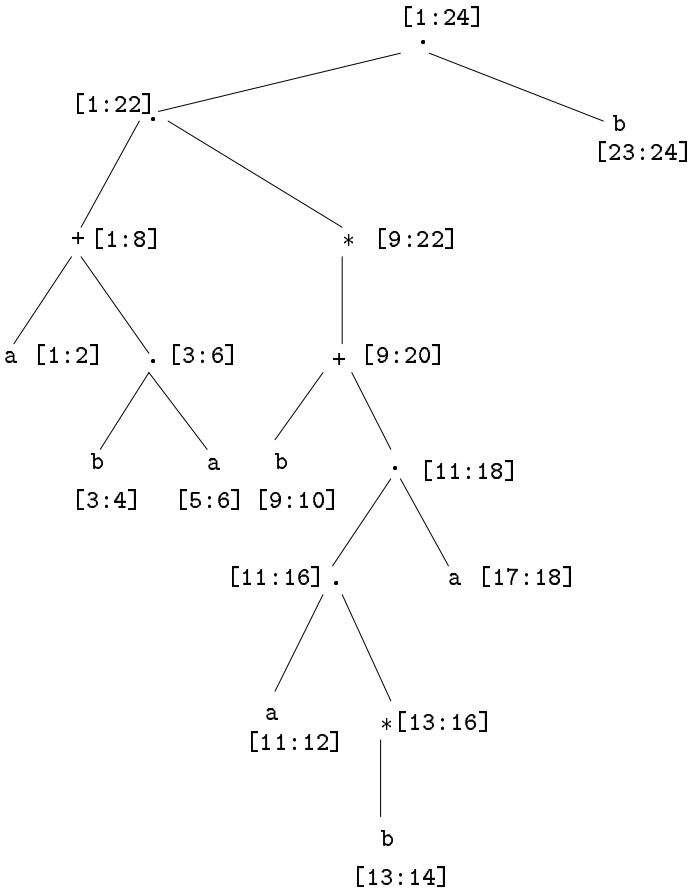
   (b) **ou bien** $\delta_\epsilon(q) \neq \emptyset$ et dans ce cas, $\forall x \in X$, $\delta_X(q, x) = \emptyset$.
   De plus on a alors $\mathrm{Card}(\delta_\epsilon(q)) \leq 2$.
   En d'autres termes, pour un tel état, il y a au plus deux ϵ-transitions possibles et il n'y aucune transition avec lecture effective de lettre possible.

2.3. Exemple

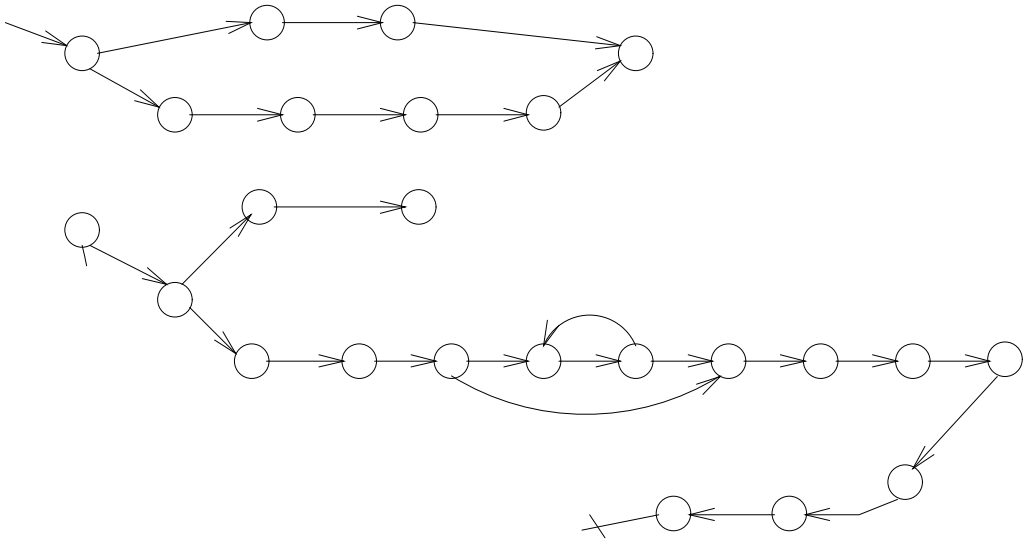
Considérons l'expression rationnelle

```
(a+ba)(b+ab*a)*b
```

dont la structure correspond à l'arbre suivant dans lequel on a ajouté pour chaque noeud les numéros associés aux états de l'AFND$_\epsilon$ associé à l'expression (la notation `[`i`:`j`]` désigne l'intervalle fermé des entiers compris entre i et j) :



La construction standard conduit à l'AFND$_\epsilon$ suivant dans lequel l'état initial est 7 et l'état final est 24 :



L'algorithme suivant, décrit dans un pseudo-langage, calcule la clôture d'un ensemble d'états d'un AFND$_\epsilon$.

Rappelons que pour les automates qui sont obtenus à partir d'une expression rationnelle, pour un état donné il existe au plus deux ϵ-transitions (pour le graphe des ϵ-transitions correspondant, cela signifie qu'un sommet a au plus deux successeurs).

```
fonction clôture(E : ensembleEtats) : ensembleEtats
{ renvoie l'union des clôtures des éléments de E }
    variable D : ensembleEtats; q : état
    début
        D ← ∅
        ∀q ∈ E faire
            ajouterCloture(q, D)
        finFaire
        retourner D
    fin
```

```
procédure ajouterCloture(q état, référence D : ensembleEtats)
{ Cette procédure réalise l'opération D ← D ∪ δ̂ε(s) }
    variable s : état
    début
        si q ∉ D alors
            D ← D ∪ {q}
            ∀s ∈ δε(q) faire
                ajouterCloture(s, D)
            finFaire
        finSi
    fin
```

Le fonctionnement d'un AFND$_\epsilon$ pour un mot w est alors simulé par la fonction suivante :

```
fonction reconnaît(A : automate, w : mot) : booléen
{ w = x1x2...xn, q0 état initial et qf état final }
    variables C : ensembleEtats; x : lettre ∈ X;
              i : entier
    début
        C ← clôture({s0})
        pour i ← 1 à n faire
            C ← clôture(δ(C, x))
        finFaire
        si qf ∈ C alors
            retourner vrai
        sinon
            retourner faux
        finSi
```

Appliquons la procédure précédente à l'automate obtenu pour l'expression rationnelle de notre exemple et le mot `baa`.

- ↪ on calcule tout d'abord la clôture de l'état initial 7. On obtient $\{7, 1, 3\}$ (l'ordre reflète celui dans lequel les sommets ont été insérés);
- ↪ on lit ensuite la première lettre du mot, c'est-à-dire `b`. On a $\delta(3, b) = 4$ et $\delta(7, b) = \delta(1, b) = \emptyset$. Après lecture de `b`, on a donc $C = \{4\}$;
- ↪ on calcule la clôture de 4 et on obtient $\{4, 5\}$;
- ↪ on lit alors la seconde lettre du mot, c'est-à-dire `a`. On a $\delta(5, a) = 6$ et $\delta(4, a) = \emptyset$. D'où on déduit la nouvelle valeur de C : $\{6\}$;
- ↪ la clôture de $\{6\}$ est $\{6, 8, 21, 19, 9, 11, 22, 23\}$;
- ↪ on lit la troisième lettre du mot, à savoir `a`.
  On a $\delta(11, a) = 12$ et $\delta(6, a) = \delta(8, a) = \delta(21, a) = \delta(19, a) = \delta(9, a) = \delta(22, a) = \delta(23, a) = \emptyset$, d'où on déduit $C = \{12\}$;
- ↪ on calcule finalement la clôture de 12, ce qui donne la valeur finale $C = \{12, 15, 13, 16, 17\}$.

Le mot `baa` n'est donc pas reconnu puisque $24 \notin C$

Par contre, si on applique l'automate au mot `bab`,

- ↪ après avoir lu les deux premières lettres, on se retrouve dans la même situation que précédemment après avoir lu deux lettres de `baa`, et donc $C = \{6, 8, 21, 19, 9, 11, 22, 23\}$. ;
- ↪ quand on lit la troisième et dernière lettre du mot, c'est-à-dire `b`, on obtient C ent $\rightarrow$

Cette construction revient à regrouper dans une même transition une suite d'ϵ-transitions et une transition avec lecture de lettre.

L'automate ainsi construit n'est pas nécessairement déterministe.

Les états terminaux de cet automate $\mathcal{B}$ sont ceux dont la clôture contient l'état final q_f de l'AFND$_\epsilon$ de départ $\mathcal{A}$.

La construction de l'automate $\mathcal{B}$ suppose le calcul de la clôture de tous les états de l'AFND$_\epsilon$ $\mathcal{A}$, ce qui revient à calculer la fermeture transitive du graphe associé à l'automate en ne conservant que les ϵ-transitions. Il existe différents algorithmes pour effectuer ce calcul. Une solution, qui n'est pas la plus efficace car certains calculs seront réalisés plusieurs fois, consiste à appliquer à chaque état de $\mathcal{A}$ la fonction `clôture` décrite précédemment.

Pour notre exemple, les clôtures des différents états sont :

clôture$(1) = \{1\}$	clôture$(2) = \{2, 8, 21, 19, 9, 11, 22, 23\}$
clôture$(3) = \{3\}$	clôture$(4) = \{4, 5\}$
clôture$(5) = \{5\}$	clôture$(6) = \{6, 8, 21, 19, 9, 11, 22, 23\}$
clôture$(7) = \{7, 1, 3\}$	clôture$(8) = \{8, 21, 19, 9, 11, 22, 23\}$
clôture$(9) = \{9\}$	clôture$(10) = \{10, 20, 19, 9, 11, 22, 23\}$
clôture$(11) = \{11\}$	clôture$(12) = \{12, 15, 13, 16, 17\}$
clôture$(13) = \{13\}$	clôture$(14) = \{14, 13, 16, 17\}$
clôture$(15) = \{15, 13, 16, 17\}$	clôture$(16) = \{16, 17\}$
clôture$(17) = \{17\}$	clôture$(18) = \{18, 20, 19, 9, 11, 22, 23\}$
clôture$(19) = \{19, 9, 11\}$	clôture$(20) = \{20, 19, 9, 11, 22, 23\}$
clôture$(21) = \{21, 19, 9, 11, 22, 23\}$	clôture$(22) = \{22, 23\}$
clôture$(23) = \{23\}$	clôture$(24) = \{24\}$

L'automate $\mathcal{B}$ a comme ensemble d'états $\{2, 4, 6, 10, 12, 14, 18, 24, 7\}$ et a comme fonction de transitions :

$\delta_\mathcal{B}$	a	b	*initial/final*
7	$\{2\}$	$\{4\}$	**état initial**
2	$\{12\}$	$\{10, 24\}$	
4	$\{6\}$	$\emptyset$	
6	$\{12\}$	$\{10, 24\}$	
10	$\{12\}$	$\{10, 24\}$	
12	$\{18\}$	$\{14\}$	
14	$\{18\}$	$\{14\}$	
18	$\{12\}$	$\{10, 24\}$	
24	$\emptyset$	$\emptyset$	**état terminal**

3.2.2. Déterminisation de l'automate

Un automate déterministe $\mathcal{C}$, équivalent à l'automate précédent est construit par le procédé standard étudié précédemment.
Cet automate a comme ensemble d'états le sous-ensemble utile de l'ensemble des parties de $\{2, 4, 6, 10, 12, 14, 18, 24, 7\}$ et sa fonction de transitions est :

$\delta_{\mathcal{C}}$	a	b	*initial/final*
$\{7\}$	$\{2\}$	$\{4\}$	**état initial**
$\{2\}$	$\{12\}$	$\{10,24\}$	
$\{4\}$	$\{6\}$	$\emptyset$	
$\{12\}$	$\{18\}$	$\{14\}$	
$\{10,24\}$	$\{12\}$	$\{10,24\}$	**état final**
$\{6\}$	$\{12\}$	$\{10,24\}$	
$\{18\}$	$\{12\}$	$\{10,24\}$	
$\{14\}$	$\{18\}$	$\{14\}$	
$\emptyset$	$\emptyset$	$\emptyset$	

3.2.3. Vérification

Nous vérifions maintenant que l'automate précédent reconnaît effectivement le langage correspondant à l'expression initiale.

Nous commençons par renommer de 1 à 8 (l'état $\emptyset$ qui est inutile étant omis) les états de l'automate $\mathcal{C}$ précédent et associons à l'automate le système d'équations suivant :

- $L_1 = aL_2 + bL_3$
- $L_2 = aL_4 + bL_5$
- $L_3 = aL_6$
- $L_4 = aL_7 + bL_8$
- $L_5 = aL_4 + bL_5 + \epsilon$
- $L_6 = aL_4 + bL_5$
- $L_7 = aL_4 + bL_5$
- $L_8 = aL_7 + bL_8$

On constate tout d'abord que $L_4 = L_8$, $L_2 = L_6 = L_7$, ce qui permet d'éliminer facilement L_8, L_6 et L_7.
Par ailleurs, $L_5 = L_2 + \epsilon$, ce qui permet d'éliminer L_5.
De même on peut éliminer L_3 car $L_3 = aL_6 = aL_2$.

On obtient ainsi le système de trois équations suivant :

- $L_1 = aL_2 + baL_2 = (a + ba)L_2$
- $L_2 = aL_4 + b(L_2 + \epsilon) = aL_4 + bL_2 + b$
- $L_4 = aL_2 + bL_4$

En utilisant le lemme d'Arden, on obtient $L_4 = b^*aL_2$

On élimine L_4 en reportant cette valeur dans l'équation définissant L_2 :

- $L_2 = ab^*aL_2 + bL_2 + b = (ab^*a + b)L_2 + b$

Par application du lemme d'Arden, on obtient alors :

- $L_2 = (ab^*a + b)^*b$

D'où finalement

- $L_1 = (a + ba)(ab^*a + b)^*b$

Ici, nous retombons exactement sur l'expression dont nous étions partis (on