INF 302 : Langages & Automates

Chapitre 10 : Théorème de Kleene

Yliès Falcone

ylies.falcone@univ-grenoble-alpes.fr — www.ylies.fr

Univ. Grenoble-Alpes, Inria

-

Laboratoire d'Informatique de Grenoble - www.liglab.fr

Équipe de recherche LIG-Inria, Corse - team.inria.fr/corse/

Plan Chap. 8 - Expressions Régulières / Théorème de Kleene

Plan Chap. 8 - Expressions Régulières / Théorème de Kleene

Théorème de Kleene

Théorème de Kleene

Soit Σ un alphabet et $L \subseteq \Sigma^*$.

L est à états finis $\Leftrightarrow$ L est régulier.

De plus :

1. Il existe un algorithme qui transforme un automate fini en une expression régulière équivalente.
2. Inversement, il existe un algorithme qui transforme une expression régulière en un automate fini équivalent.

(Ces algorithmes sont implémentés dans Aude.)

Démonstration : dans les deux prochaines sections

Nous allons :

- montrer ce théorème de manière semi-formelle.
- exhiber une procédure effective de traduction entre ces formalismes.

(L'implication de droite à gauche et le deuxième point du théorème découlent des propriétés de fermeture des automates finis.)

Conséquences du théorème de Kleene

Soient e_1 et e_2 deux expressions régulières.

Décidabilité de l'inclusion des langages dénotés par des expressions régulières

Déterminer si $L(e_1) \subseteq L(e_2)$ est décidable.

Décidabilité de l'équivalence d'expressions régulières

Déterminer si $e_1 \equiv e_2$ est décidable.

Remarque Il n'y a pas d'autre méthode connue pour montrer la décidabilité de ces deux résultats (que d'utiliser le théorème de Kleene et de passer par les automates). □

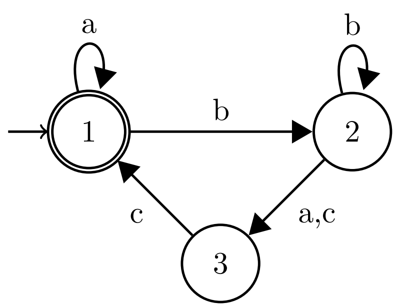
Univ. Grenoble Alpes, Département Licence Sciences et Technologies, Licence deuxième année

Plan Chap. 8 - Expressions Régulières / Théorème de Kleene

Plan Chap. 8 - Expressions Régulières / Théorème de Kleene

L'idée par un exemple : trouver une relation entre les états

Soit $\Sigma = \{a, b, c\}$ et A l'automate suivant :

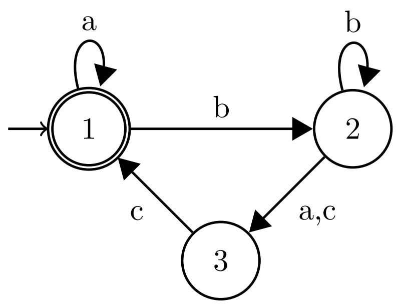


On peut associer le système d'équations suivant à A :

$$\begin{array}{lcl} X_1 & = & \{a\} \cdot X_1 \cup \{b\} \cdot X_2 \cup \{\epsilon\} \\ X_2 & = & \{b\} \cdot X_2 \cup \{a, c\} \cdot X_3 \\ X_3 & = & \{c\} \cdot X_1 \end{array}$$

Intuitivement, X_i décrit les mots acceptés à partir de l'état i.

L'idée par un exemple : écrire le système d'équations



Si on utilise les expressions régulières comme notation, on peut écrire ce système d'équations de la manière suivante :

$$\begin{array}{rcl} X_1 & = & aX_1 + bX_2 + \epsilon \\ X_2 & = & bX_2 + (a+c)X_3 \\ X_3 & = & cX_1 \end{array}$$

Système d'équations associé à un automate

Méthode de Janusz Antoni Brzozowski et Edward Joseph McCluskey, 1964

Soit $A = (Q, \Sigma, q_0, \delta, F)$ un ADEF ou ANDEF.

Soit $SE(A)$ le système d'équations donné par :

$$X_q = \sum_{\delta(q,a)=q'} aX_{q'} + (\text{ si } q \in F \text{ alors } \epsilon \text{ sinon } \emptyset)$$

Question :

Comment résoudre un tel système d'équations ?

Résolution d'équations linéaires (lemme d'Arden)

Lemme

Soient $A, B \subseteq \Sigma^*$ des langages. Considérons l'équation :

$$X = AX + B$$

1. Le langage A^*B est une solution de l'équation
2. (Lemme d'Arden) : Si $\epsilon \notin A$, alors A^*B est la solution unique.

Démonstration (En TD).

1. Vérifier que $A^*B = AA^*B + B = A \cdot A^* \cdot B + \epsilon \cdot B = (AA^* + \epsilon)B$.
2. Preuve par contradiction. $\square$

Attention :

- Pour résoudre un système d'équations correspondant à un automate, on applique le lemme que dans le deuxième cas.
- Remarquons que, comme l'automate d'entrée est soit un ADEF ou un ANDEF (et pas un ϵ-ANDEF), le cas 2 s'applique toujours.

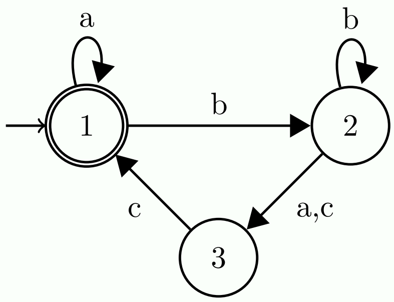
Solution du système d'équations linéaires et langage de l'automate

Théorème

- Soit $A = (Q, \Sigma, q_0, \delta, F)$ un automate fini.
- Soit $(L_q \mid q \in A)$ la plus petite solution de $SE(A)$.
- Alors,

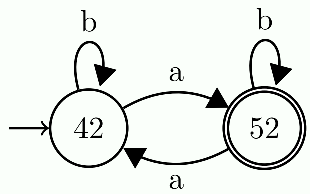
$$L(A) = L_{X_{q_0}}.$$

Le langage accepté par



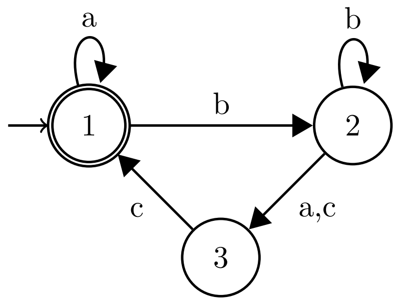
est L_1

Le langage accepté par



est L_{42}

Exemple de résolution de système d'équations

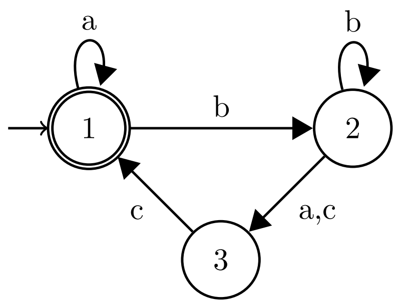


Considérons le système d'équations suivant associé à l'automate ci-dessus :

$$\begin{array}{lcl} X_1 & = & aX_1 + bX_2 + \epsilon \\ X_2 & = & bX_2 + (a+c)X_3 \\ X_3 & = & cX_1 \end{array}$$

Univ. Grenoble Alpes, Département Licence Sciences et Technologies, Licence deuxième année

Exemple de résolution de système d'équations



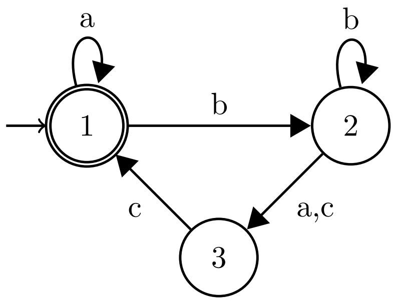
On remplace X_3 par cX_1 dans la deuxième équation :

$$\begin{array}{lcl} X_1 & = & aX_1 + bX_2 + \epsilon \\ X_2 & = & bX_2 + (a+c)cX_1 \\ X_3 & = & cX_1 \end{array}$$

On applique le lemme d'Arden sur la deuxième équation ($\epsilon \notin L(b) = \{b\}$)

$$\begin{array}{lcl} X_1 & = & aX_1 + bX_2 + \epsilon \\ X_2 & = & b^*(a+c)cX_1 \\ X_3 & = & cX_1 \end{array}$$

Exemple de résolution de système d'équations



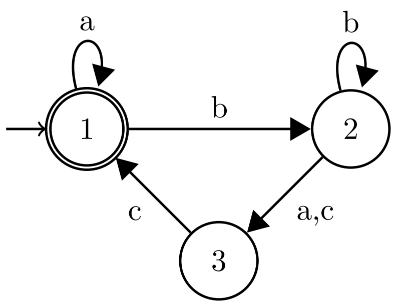
On remplace X_2 par $b^*(a+c)cX_1$ dans la première équation :

$$\begin{array}{rcl} X_1 & = & (a+bb^*(a+c)c)X_1+\epsilon \\ X_2 & = & b^*(a+c)cX_1 \\ X_3 & = & cX_1 \end{array}$$

Et on applique le lemme d'Arden sur la première équation ($\epsilon \notin L(a+bb^*(a+c)c)$) :

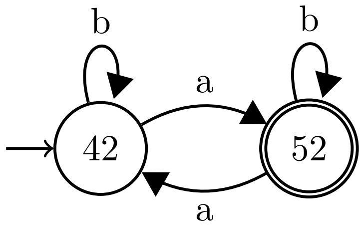
$$\begin{array}{rcl} X_1 & = & (a+bb^*(a+c)c)^* \\ X_2 & = & b^*(a+c)cX_1 \\ X_3 & = & cX_1 \end{array}$$

Exemple de résolution de système d'équations



Le langage accepté est $(a+bb^*(a+c)c)^*$.

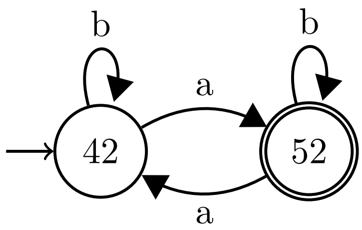
Exemple 2 de résolution de système d'équations



Considérons le système d'équations suivant associé à l'automate ci-dessus :

$$\begin{array}{rcl} X_{42} & = & bX_{42} + aX_{52} \\ X_{52} & = & bX_{52} + aX_{42} + \epsilon \end{array}$$

Exemple 2 de résolution de système d'équations



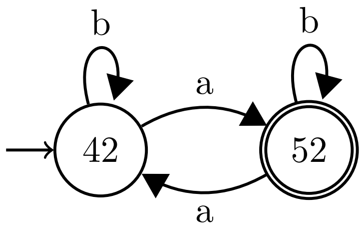
On applique le lemme d'Arden sur la deuxième équation ($\epsilon \notin b$)

$$\begin{array}{rcl} X_{42} & = & bX_{42} + aX_{52} \\ X_{52} & = & b^*(aX_{42} + \epsilon) \end{array}$$

On remplace X_{52} par $b^*(aX_{42} + \epsilon)$ dans la première équation :

$$\begin{array}{rcl} X_{42} & = & bX_{42} + ab^*(aX_{42} + \epsilon) \\ X_{52} & = & b^*(aX_{42} + \epsilon) \end{array}$$

Exemple 2 de résolution de système d'équations



On simplifie et factorise la première équation :

$$\begin{array}{rcl} X_{42} & = & (b + ab^* a)X_{42} + ab^* \\ X_{52} & = & b^*(aX_{42} + \epsilon) \end{array}$$

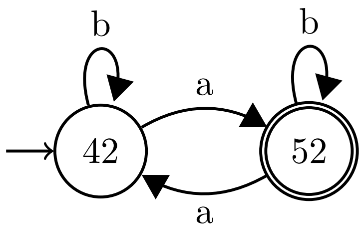
On applique le lemme d'Arden sur la première équation ($\epsilon \notin (b + ab^* a)$) :

$$\begin{array}{rcl} X_{42} & = & (b + ab^* a)^* ab^* \\ X_{52} & = & b^*(aX_{42} + \epsilon) \end{array}$$

Le langage accepté est $(b + ab^* a)^* ab^*$.

Exemple 2 bis de résolution de système d'équations

Remarque Le choix de l'équation sur laquelle on applique le lemme d'Arden influence l'expression régulière résulat.

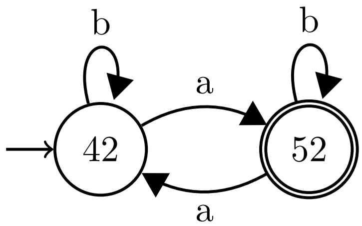


Considérons le système d'équations suivant associé à l'automate ci-dessus :

$$\begin{array}{rcl} X_{42} & = & bX_{42} + aX_{52} \\ X_{52} & = & bX_{52} + aX_{42} + \epsilon \end{array}$$

Exemple 2 bis de résolution de système d'équations

Remarque Le choix de l'équation sur laquelle on applique le lemme d'Arden influence l'expression régulière résulat. □



On applique le lemme d'Arden sur la première équation ($\epsilon \notin b$)

$$\begin{array}{rcl} X_{42} & = & b^* a X_{52} \\ X_{52} & = & bX_{52} + aX_{42} + \epsilon \end{array}$$

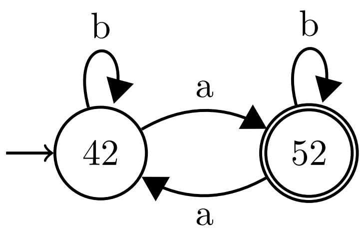
On remplace X_{42} par $b^* a X_{52}$ dans la deuxième équation :

$$\begin{array}{rcl} X_{42} & = & b^* a X_{52} \\ X_{52} & = & bX_{52} + ab^* a X_{52} + \epsilon \end{array}$$

Univ. Grenoble Alpes, Département Licence Sciences et Technologies, Licence deuxième année

Exemple 2 bis de résolution de système d'équations

Remarque Le choix de l'équation sur laquelle on applique le lemme d'Arden influence l'expression régulière résulat. □



On simplifie et factorise la deuxième équation :

$$\begin{array}{rcl} X_{42} & = & b^* a X_{52} \\ X_{52} & = & (b + ab^* a) X_{52} + \epsilon \end{array}$$

On applique le lemme d'Arden sur la deuxième équation ($\epsilon \notin (b + ab^* a)$) :

$$\begin{array}{rcl} X_{42} & = & b^* a X_{52} \\ X_{52} & = & (b + ab^* a)^* + \epsilon = (b + ab^* a)^* \end{array}$$

Le langage accepté est $b^* a \ (b + ab^* a)^*$.

Plan Chap. 8 - Expressions Régulières / Théorème de Kleene

Présentation de la méthode par élimination des états

- Entrée : $A = (Q, \Sigma, q_0, \Delta, F)$, un ϵ-AENFD.
- Sortie : e_A, une expressions régulière telle que $L(e_A) = L(A)$.

Idée :

- Étiqueter les transitions par des expressions régulières.
- Supprimer les états (non initiaux et finaux) en mettant à jour les transitions sans modifier le langage.

La technique d'élimination des états nécessite un automate normalisé :

- état initial sans transition entrante,
- un seul état final sans transition sortante.

Phases de la méthode :

1. Normalisation
2. Élimination des états

Normalisation - définition

Normalisation - pour l'initialisation

S'ils existent $q \in Q$ et $a \in \Sigma$ t.q. $(q, a, q_0) \in \Delta$, alors ajouter un nouvel état i à Q t.q. :

- ajouter (i, ϵ, q_0) à Δ
- i est le nouvel état initial

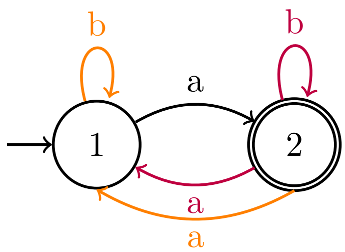
Normalisation - pour la terminaison

Si $|F| > 1$ ou ils existent $q \in F$, $q' \in Q$ et $a \in \Sigma$ tels que $(q, a, q') \in \Delta$, alors ajouter un nouvel état f à Q t.q. :

- ajouter (q, ϵ, f) à Δ, pour tout $q \in F$,
- $\{f\}$ est le nouvel ensemble d'états terminaux/finaux.

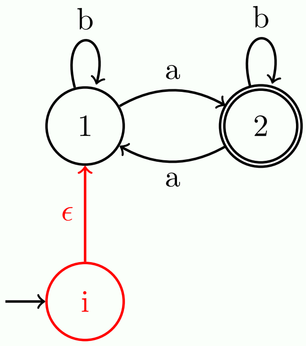
Soit $(Q, \Sigma, i, \Delta, \{f\})$ l'automate résultant de la normalisation de A.

Normalisation - exemple 1

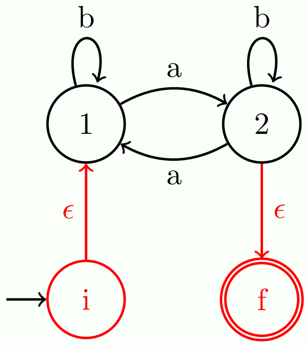


- Considérons l'automate ci-contre.
- Cet automate n'est pas normalisé ni pour l'initialisation ni pour la terminaison.

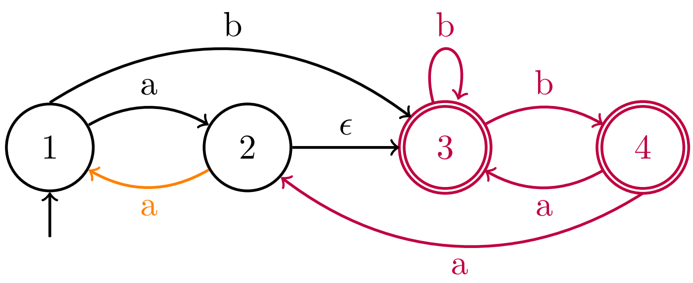
Normalisation - pour l'initialisation



Normalisation - pour la terminaison

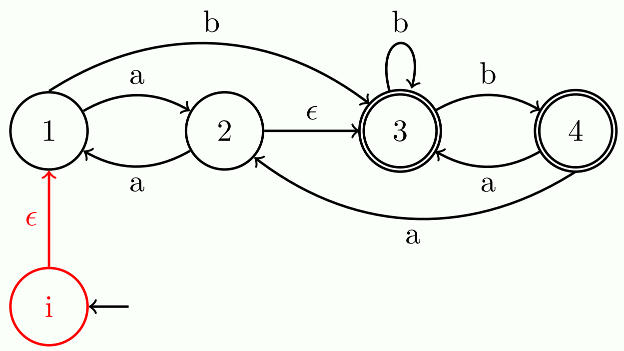


Normalisation - exemple 2

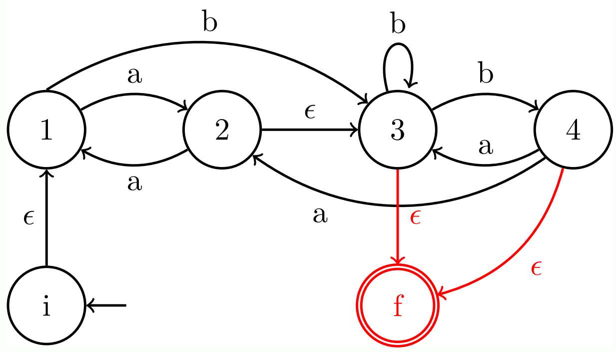


- Considérons l'automate ci-contre.
- Cet automate n'est pas normalisé ni pour l'initialisation ni pour la terminaison.

Normalisation - pour l'initialisation



Normalisation - pour la terminaison



Méthode par élimination des états

Élimination des états - algorithme

Soit $(Q, \Sigma, i, \Delta, \{f\})$ l'automate résultant de la normalisation de A.

Soit $R_{q,q'}$ l'expression régulière associée à la transition entre les états q et q'.

Algorithme de suppression des états

EE1 Si $Q = \{i, f\}$, alors l'expression régulière associée à A est $R_{i,f}$ et l'algorithme termine. (Sinon aller à l'étape EE2.)

EE2 **Choisir** $q \in Q \setminus \{i, f\}$. (Aller à l'étape EE3.)

EE3 **Éliminer** q comme suit (EE3a + EE3b), puis aller à l'étape EE1.

EE3a Pour chaque $q_1, q_2 \in Q \setminus \{q\}$, l'expression R_{q_1,q_2} devient

$$R_{q_1,q_2} \quad + \quad R_{q_1,q} \cdot R^*_{q,q} \cdot R_{q,q_2}$$

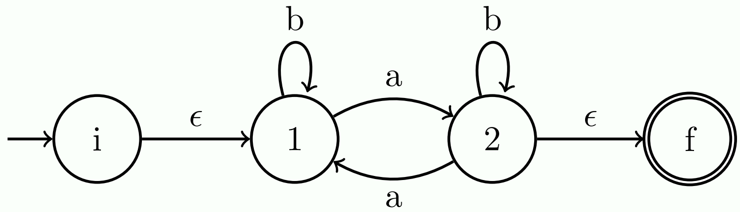
EE3b Considérer $Q \setminus \{q\}$ comme nouvel ensemble d'états.

Remarque L'ordre d'élimination des états influe sur la taille de l'expression finale générée. Des heuristiques existent pour le choix (étape EE2). □

Méthode par élimination des états

Élimination des états - exemple 1

Exemple (Calcul de l'expression régulière associée à un automate par suppression des états)

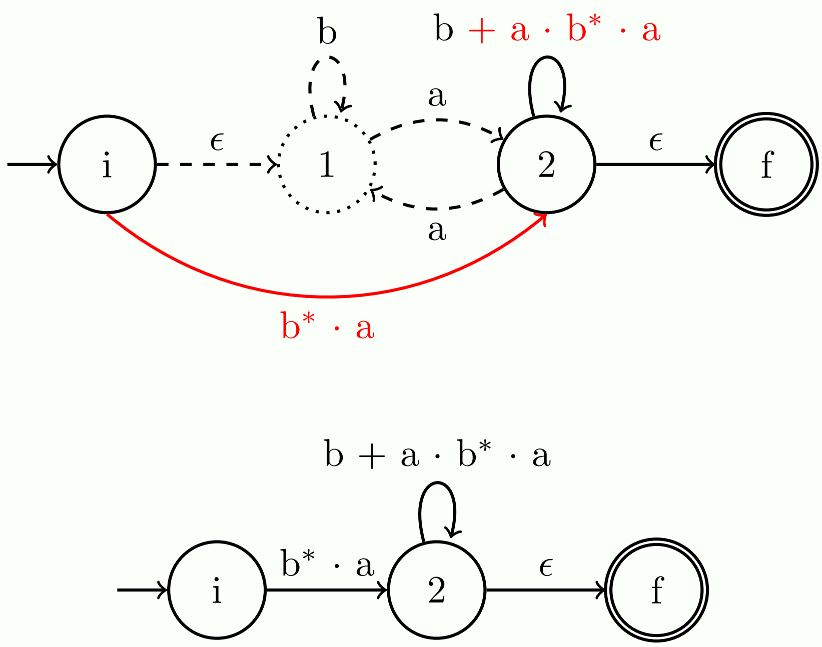


- Considérons l'automate normalisé ci-dessus.
- Nous représentons les expressions régulières $R_{i,j}$ sur les transitions de l'automate.
- Supprimons les états 1 et 2, dans cet ordre.

Méthode par élimination des états

Élimination des états - exemple 1 (suite)

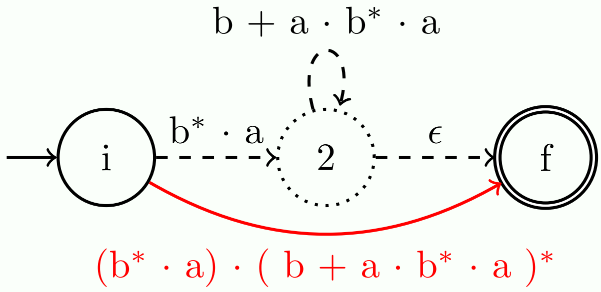
Suppression de l'état 1

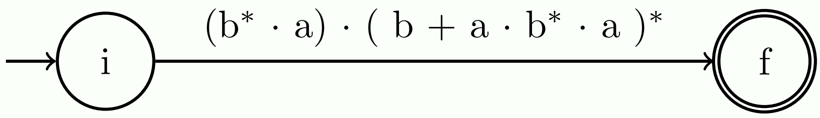


Méthode par élimination des états

Élimination des états - exemple 1 (suite)

Suppression de l'état 2

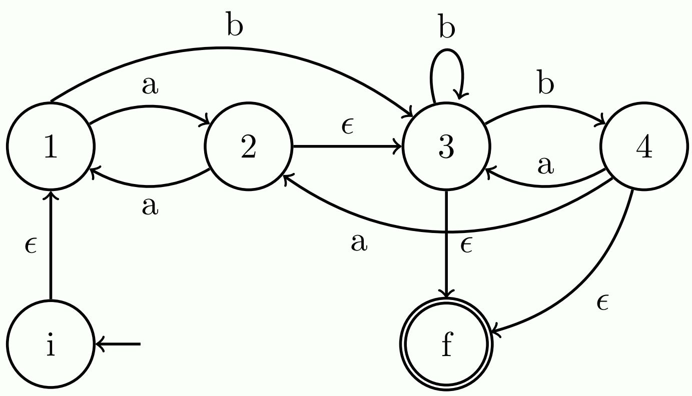




Méthode par élimination des états

Élimination des états - exemple 2

Exemple (Calcul de l'expression régulière associée à un automate par suppression des états)

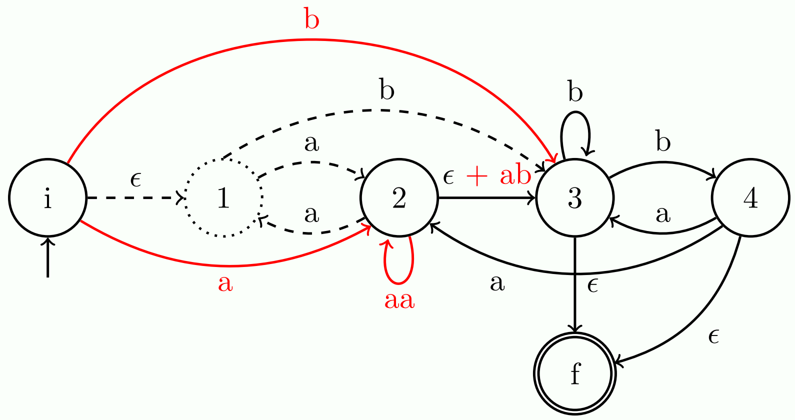


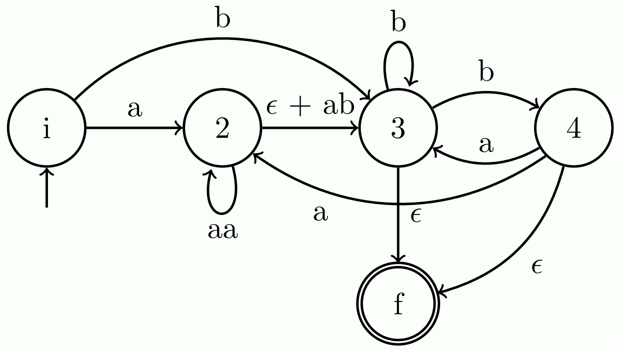
- Considérons l'automate normalisé ci-dessus.
- Nous représentons les expressions régulières $R_{i,j}$ sur les transitions de l'automate.
- Supprimons les états 1, 2, 3 et 4, dans cet ordre.

Méthode par élimination des états

Élimination des états - exemple 2 (suite)

Suppression de l'état 1

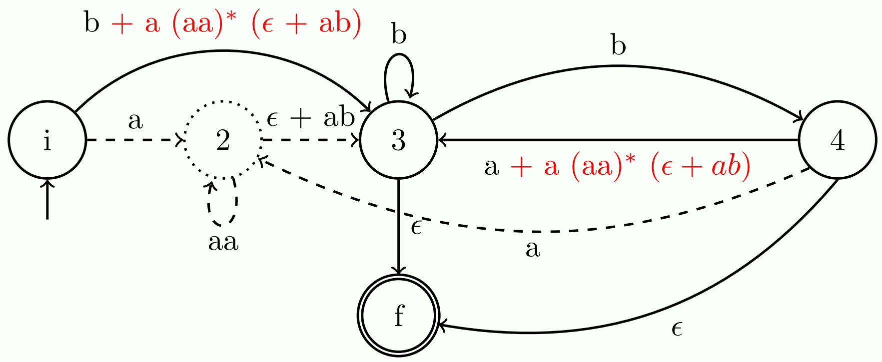


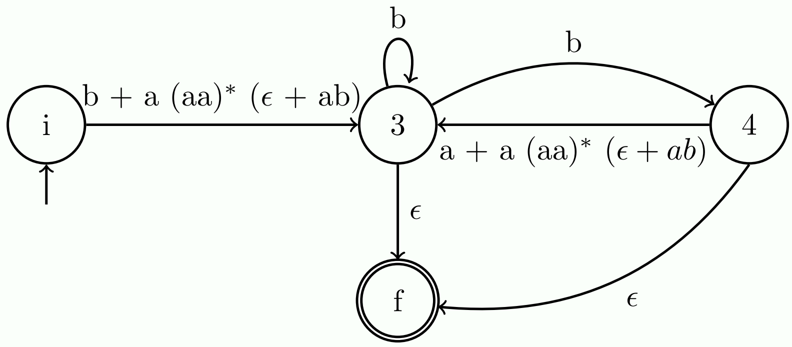


Méthode par élimination des états

Élimination des états - exemple 2 (suite)

Suppression de l'état 2

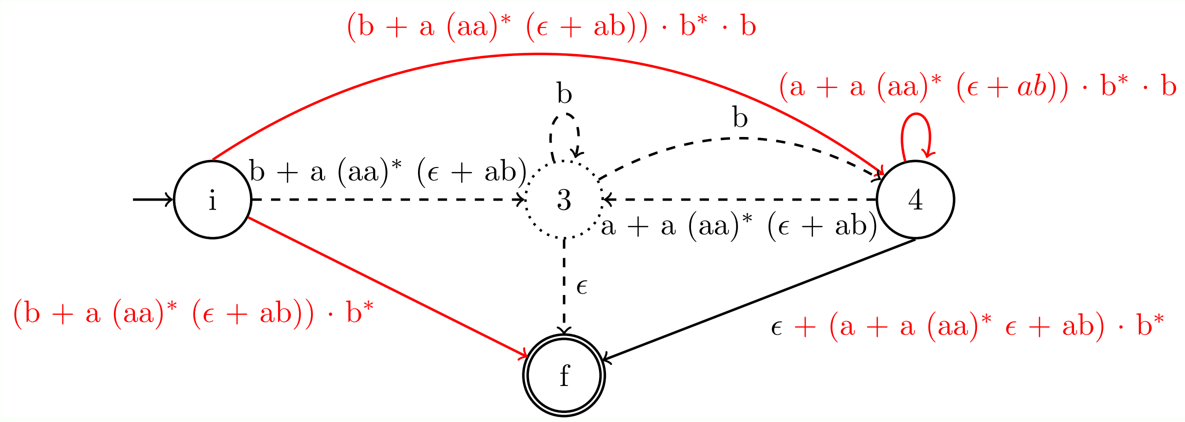


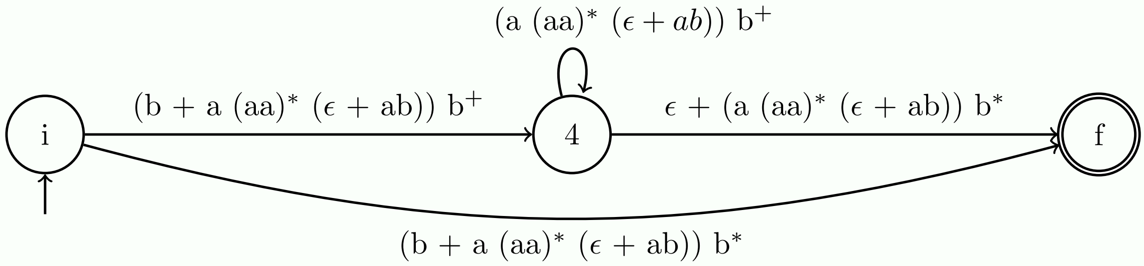


Méthode par élimination des états

Élimination des états - exemple 2 (suite)

Suppression de l'état 3



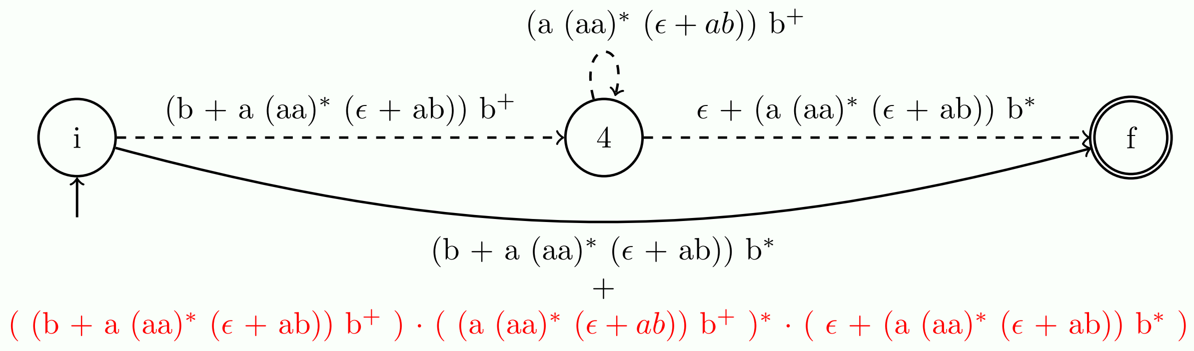


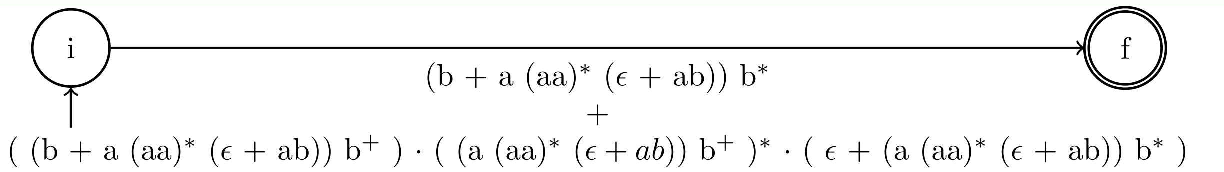
Univ. Grenoble Alpes, Département Licence Sciences et Technologies, Licence deuxième année

Méthode par élimination des états

Élimination des états - exemple 2 (suite)

Suppression de l'état 4





Plan Chap. 8 - Expressions Régulières / Théorème de Kleene

L'idée intuitive

Algorithme de McNaughton et Yamada (1960). [1]

- Associer une expression régulière décrivant le langage entre deux états i et j.
- Induction : on se donne un numéro d'état n maximal pour les états intermédiaires :
  - initialisation : on s'interdit de passer par tous les états,
  - pas d'induction : on autorise un nouvel état intermédiaire dans les chemins et on calcul les chemins où l'état $n+1$ est autorisé en fonction des chemins ou au plus l'état n est autorisé.
- L'expression régulière finale est celle décrivant :
  - l'*union* des chemins depuis l'état initial vers un état accepteur,
  - n'ayant *aucun état interdit*.

Théorème

Soit A un ADEF, alors :

- il existe une expression régulière e telle que $L(e) = L(A)$,
- il existe un algorithme de construction de e.

1. Robert McNaughton et Hisao Yamada, « Regular expressions and state graphs for automata », IRE Trans. Electronic Computers, 1960.

Preuve

Soit $A = (Q, \Sigma, q_{\text{init}}, \delta, F)$ un AEFD.

L'idée

Construire une collection d'expressions régulières qui décrivent progressivement des chemins de moins en moins contraints dans l'automate, par induction.

Début de la démonstration

- Supposons, quitte à utiliser une fonction de renommage, que les états sont numérotés de 1 à n (ce qui est possible car il y a un nombre fini d'états).
- Soit $R_{i,j}^k$ l'expression régulière des mots étiquettes d'un chemin entre l'état i et l'état j qui passe uniquement par des états *intermédiaires* plus petits que k.
  Il n'y a aucune contrainte sur i et j.
- L'expression régulière de l'automate est :

$$\sum_{f \in F} R_{1,f}^n$$

Nous allons calculer les $R_{i,j}^k$ pour $k = 0, \ldots, n$.

Calcul de $R_{i,j}^0$

$R_{i,j}^0$ est l'expression régulière des chemins entre l'état i et l'état j dont tous les états intermédiaires ont un numéro plus petit que 0 ;
$\hookrightarrow$ c'est-à-dire qui n'ont *aucun état intermédiaire*.

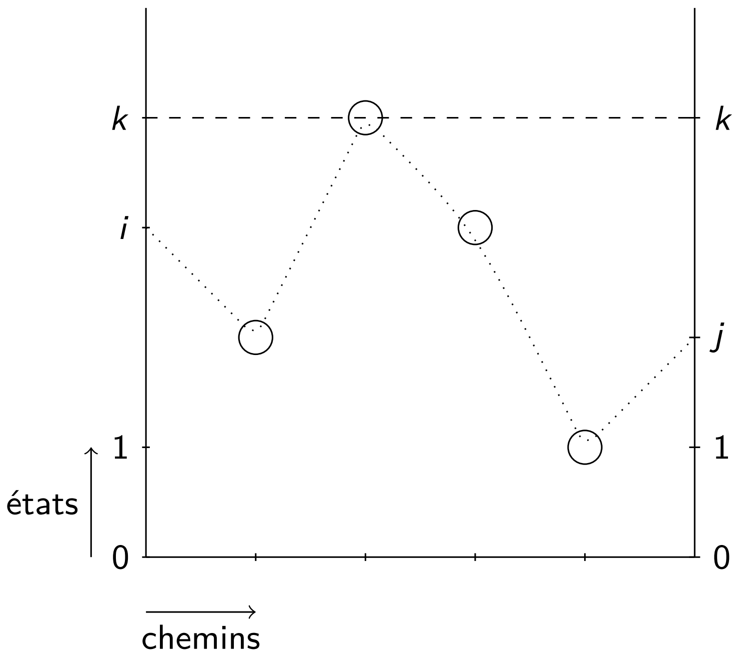
Intéressons nous aux chemins *directs* entre un état i et un état j : il y a deux cas possibles.

- Si $i \neq j$, alors on regarde les transitions *directes* entre i et j :
  - Il n'y a pas de transition :
  $$R_{i,j}^0 = \emptyset.$$
  - Il y a une transition étiquetée par un symbole a :
  $$R_{i,j}^0 = a.$$
  - Il y a plusieurs transitions étiquetées par des symboles $a_1, \ldots, a_n$:
  $$R_{i,j}^0 = a_1 + \ldots + a_n.$$
- Sinon ($i = j$), les cas précédents s'appliquent de la même manière.

  Il faut de plus ajouter ϵ à chaque expression régulière.

Illustration de $R^k_{i,j}$

Première possibilité concernant le positionnement de i et j par rapport à k



Univ. Grenoble Alpes, Département Licence Sciences et Technologies, Licence deuxième année

Illustration de $R^k_{i,j}$

Seconde possibilité concernant le positionnement de i et j par rapport à k

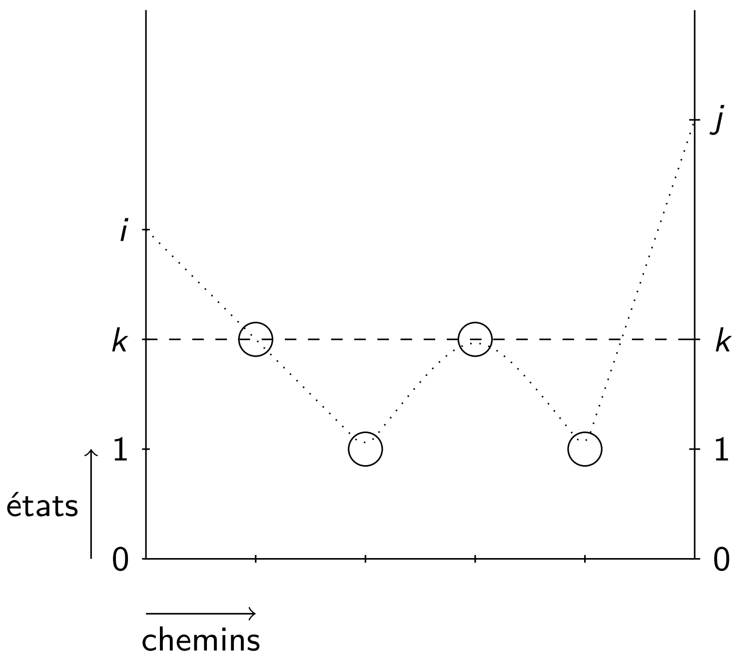
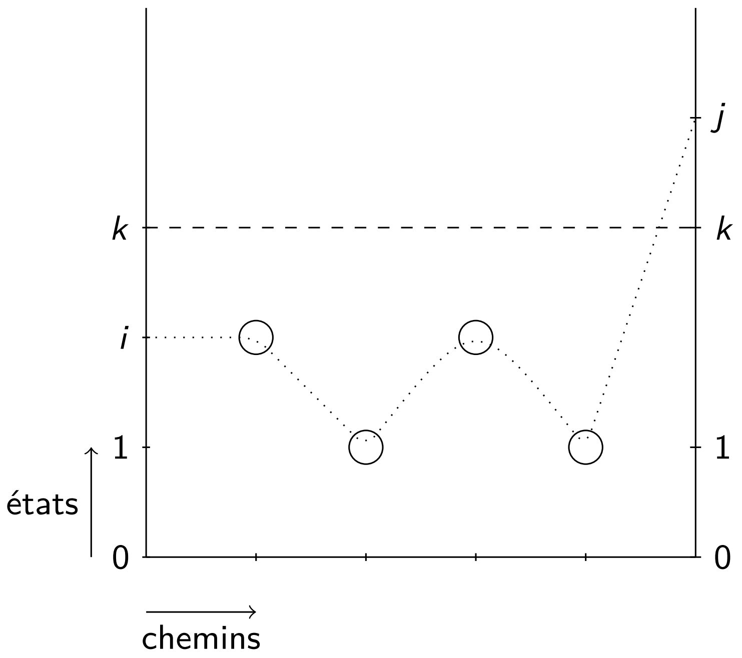


Illustration de $R^k_{i,j}$

Un chemin ne passe pas obligatoirement par l'état k (quelque soit le positionnement de i et j par rapport à k)



Calcul de $R^k_{i,j}$

Résumons :

- Le positionnement de i et j peut être quelconque par rapport à k : la contrainte reliée à $R^k_{i,j}$ ne porte que sur les *états intermédiaires*.
- Un chemin dans $R^k_{i,j}$ peut passer par l'état k, ou non.

Exprimons maintenant $R^k_{i,j}$ en fonction de $R^{k-1}_{i,j}$.

Considérons un chemin de $R^k_{i,j}$:

- Soit il ne passe pas par l'état k, alors c'est un chemin de $R^{k-1}_{i,j}$.
- Soit il passe par l'état k (au moins une fois). On peut décomposer le chemin en chemins qui ne passent pas par un état intermédiaire plus grand que $k-1$.

$$i \xrightarrow{R^{k-1}_{i,k}} k \xrightarrow{{R^{k-1}_{k,k}}^*} k \xrightarrow{R^{k-1}_{k,j}} j$$

(un chemin qui passe 2 fois par k)

$$R^k_{i,j} = R^{k-1}_{i,j} + R^{k-1}_{i,k} \cdot {R^{k-1}_{k,k}}^* \cdot R^{k-1}_{k,j}$$

Application de la méthode

Étapes du calcul de l'expression régulière d'un automate - méthode des chemins

1. Renommer les chemins de l'automate pour qu'ils soient numérotés de 1 à n, où $|Q|$.
2. Calculer $R^0_{i,j}$.
3. Calculer les $R^k_{i,j}$, $k = 1, \ldots, |Q|$ en utilisant l'expression récursive de $R^k_{i,j}$:

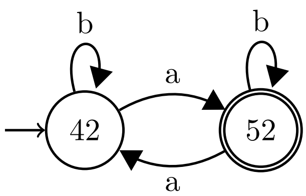
$$R^k_{i,j} = R^{k-1}_{i,j} + R^{k-1}_{i,k} \cdot {R^{k-1}_{k,k}}^* \cdot R^{k-1}_{k,j}$$

4. L'expression régulière de l'automate est $\sum_{f \in F} R^{|Q|}_{1,f}$.

Remarque En pratique, on arrêtera le calcul à $R^{|Q|-1}_{i,j}$ et calculera les $R^{|Q|}_{1,f}$, pour $f \in F$ au besoin. □

Méthode des chemins : exemple 1

Soit $\Sigma = \{a, b\}$ et A l'automate suivant :



Calcul des $R^0_{i,j}$

- $R^0_{1,1} = b + \epsilon$
- $R^0_{1,2} = a$
- $R^0_{2,1} = a$
- $R^0_{2,2} = b + \epsilon$

Calcul des $R^1_{i,j} = R^0_{i,j} + R^0_{i,1} \cdot (R^0_{1,1})^* \cdot R^0_{1,j}$

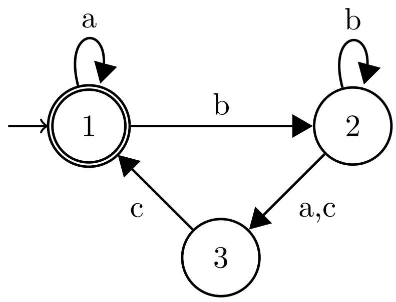
- $R^1_{1,1} = b^*$
- $R^1_{1,2} = b^* \cdot a$
- $R^1_{2,1} = a \cdot b^*$
- $R^1_{2,2} = b + a \cdot b^* \cdot a$

Calcul de $R^2_{1,2} = R^1_{1,2} + R^1_{1,2} \cdot (R^1_{2,2})^* \cdot R^1_{2,2}$

$$\begin{aligned} R^2_{1,2} &= (b^* \cdot a) + (b^* \cdot a) \cdot (b + a \cdot b^* \cdot a)^* \cdot (b + a \cdot b^* \cdot a) \\ &= (b^* \cdot a) + (b^* \cdot a) \cdot (b + a \cdot b^* \cdot a)^+ \\ &= (b^* \cdot a) \cdot (b + a \cdot b^* \cdot a)^* \end{aligned}$$

Méthode des chemins : exemple 2

Soit $\Sigma = \{a, b, c\}$ et A l'automate suivant :



Calcul des $R^0_{i,j}$

- $R^0_{1,1} = a + \epsilon$
- $R^0_{1,2} = b$
- $R^0_{1,3} = \emptyset$
- $R^0_{2,1} = \emptyset$
- $R^0_{2,2} = b + \epsilon$
- $R^0_{2,3} = a + c$
- $R^0_{3,1} = c$
- $R^0_{3,2} = \emptyset$
- $R^0_{3,3} = \epsilon$

Méthode des chemins : exemple 2 (suite)

$R^0_{i,j}$

- $R^0_{1,1} = a + \epsilon$
- $R^0_{1,2} = b$
- $R^0_{1,3} = \emptyset$
- $R^0_{2,1} = \emptyset$
- $R^0_{2,2} = b + \epsilon$
- $R^0_{2,3} = a + c$
- $R^0_{3,1} = c$
- $R^0_{3,2} = \emptyset$
- $R^0_{3,3} = \epsilon$

Calcul des $R^1_{i,j} = R^0_{i,j} + R^0_{i,1} \cdot {R^0_{1,1}}^* \cdot R^0_{1,j}$

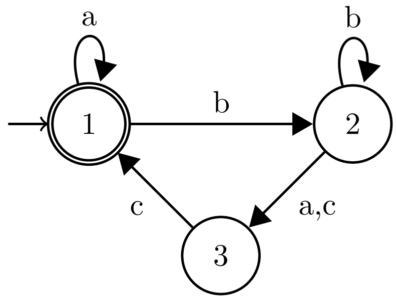
- $R^1_{1,1} = a^*$
- $R^1_{1,2} = a^* b$
- $R^1_{1,3} = \emptyset$
- $R^1_{2,1} = \emptyset$
- $R^1_{2,2} = b + \epsilon$
- $R^1_{2,3} = a + c$
- $R^1_{3,1} = c \cdot a^*$
- $R^1_{3,2} = c \cdot a^* \cdot b$
- $R^1_{3,3} = \epsilon$

Calcul des $R^2_{i,j} = R^1_{i,j} + R^1_{i,2} \cdot {R^1_{2,2}}^* \cdot R^1_{2,j}$

- $R^2_{1,1} = a^*$
- $R^2_{1,2} = a^* \cdot b^+$
- $R^2_{1,3} = a^* \cdot b^+ \cdot (a + c)$
- $R^2_{2,1} = \emptyset$
- $R^2_{2,2} = b^*$
- $R^2_{2,3} = b^* \cdot (a + c)$
- $R^2_{3,1} = c \cdot a^*$
- $R^2_{3,2} = c \cdot a^* \cdot b^+$
- $R^2_{3,3} = \epsilon + c \cdot a^* \cdot b^+ \cdot (a + c)$

Méthode des chemins : exemple 2 (fin)

Soit $\Sigma = \{a, b, c\}$ et A l'automate suivant :



Expression régulière de A

$$R^3_{1,1} = \quad a^* + (a^* \cdot b^+ \cdot (a+c)) \cdot (\epsilon + c \cdot a^* \cdot b^+ \cdot (a+c))^* \cdot c \cdot a^*$$
$$R^3_{1,1} = \quad a^* + (a^* \cdot b^+ \cdot (a+c)) \cdot (c \cdot a^* \cdot b^+ \cdot (a+c))^* \cdot c \cdot a^*$$

Cette expression régulière est équivalente à :

$$a^* \cdot (b^+ \cdot (a+c) \cdot c \cdot a^*)^*$$

Plan Chap. 8 - Expressions Régulières / Théorème de Kleene

Méthode par élimination des états

Élimination des états - exemple

Méthode associant des équations aux états

+ élégance
+ génère des expressions régulières raisonnablement compacte
- pas aussi simple à implémenter que les autres méthodes

Méthode par élimination des états

+ intuitive
+ pratique pour la vérification manuelle

Méthode associant des équations aux chemins

+ implémentation claire et simple
- fastidieux manuellement
- tendance à créer des expressions régulières très longues

Plan Chap. 8 - Expressions Régulières / Théorème de Kleene

Plan Chap. 8 - Expressions Régulières / Théorème de Kleene

L'idée

Objectif :

Montrer que tout langage décrit par une expression régulière peut être reconnu par un automate
$\hookrightarrow$ pour chaque expression régulière e, on peut trouver un automate A tel que

$$L(e) = L(A)$$

$\hookrightarrow$ les langages réguliers sont des langages à états

Automates construits

Les automates que nous allons construire sont des ϵ-ANDEF tels que :

- un seul état accepteur,
- pas de transition depuis l'état acepteur,
- pas de transition vers l'état initial.

Les langages réguliers sont des langages à états

Théorème : Les langages réguliers sont des langages à états

Tout langage reconnu par une expression régulière peut être défini/reconnu par un automate à état fini.

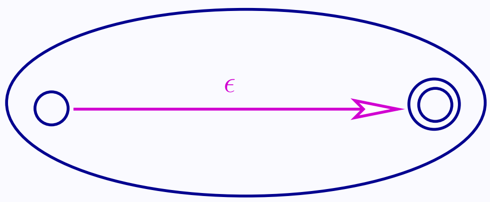
Induction sur les expressions régulières

- éléments de bases : ϵ, symboles seuls, $\emptyset$
- expressions composées : union, concaténation, fermeture (en fonction d'automates définis pour les opérandes)

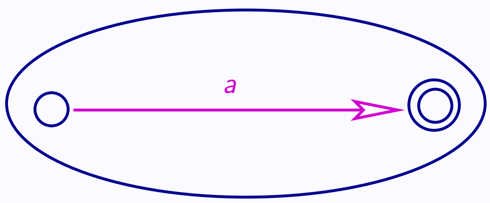
Les langages réguliers sont des langages à états

Élements de base

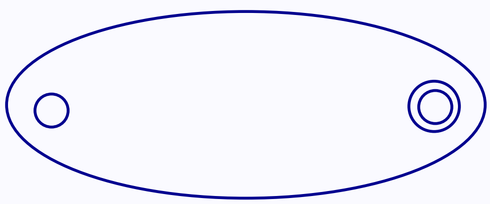
Mot vide (ϵ)



Symboles seuls ($a \in \Sigma$)



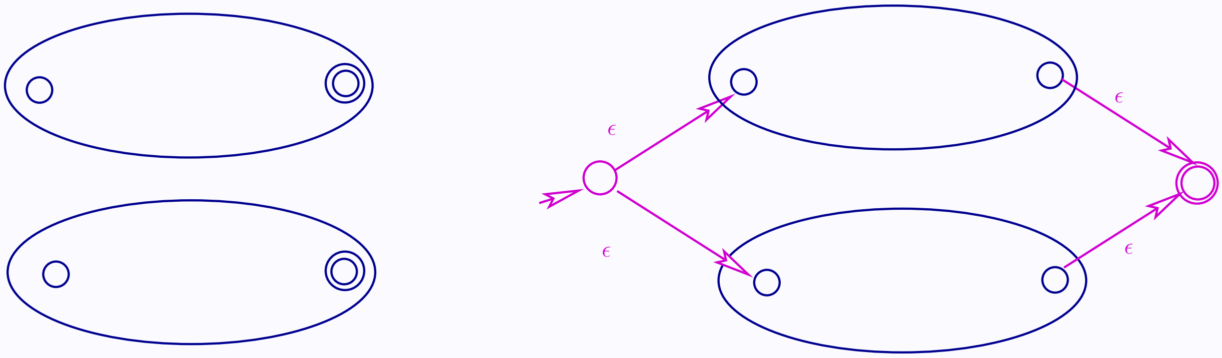
Ensemble vide ($\emptyset$)



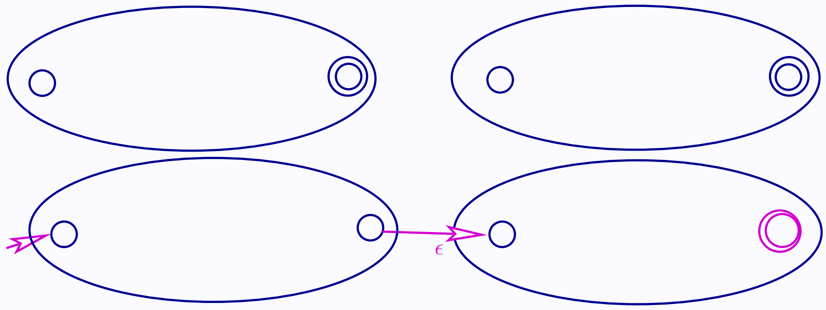
Les langages réguliers sont des langages à états

Élements composés

$e_1 + e_2$



$e_1 \cdot e_2$

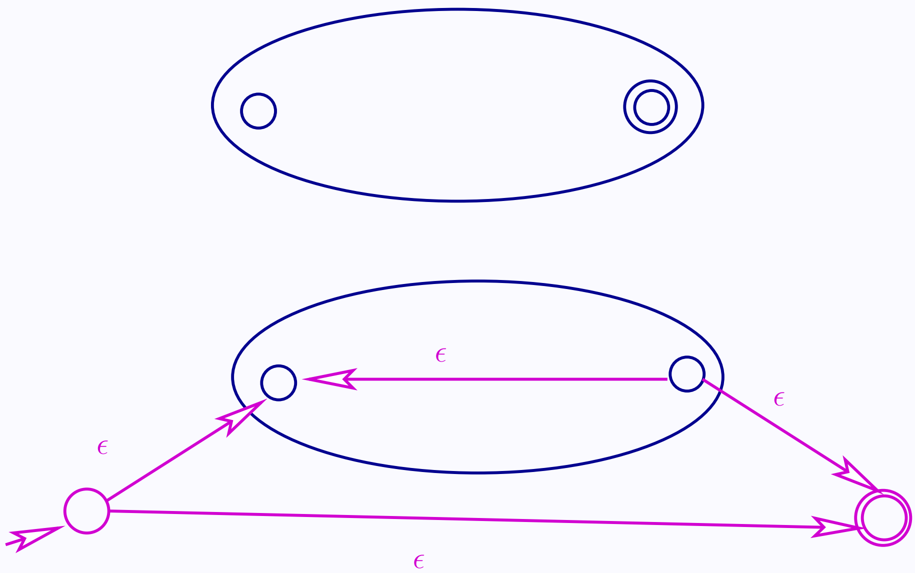


Univ. Grenoble Alpes, Département Licence Sciences et Technologies, Licence deuxième année

Les langages réguliers sont des langages à états

Élements composés (suite)

e^*



Traduction des expressions régulières vers automates à états finis

Exemple (Traduction des expressions régulières vers ϵ-ANDEF)

- $0 + 1$
- $(0 + 1)^*$
- $(0 + 1)^* \cdot 1$
- $(0 + 1)^* \cdot 1 \cdot (0 + 1)$

Plan Chap. 8 - Expressions Régulières / Théorème de Kleene

Introduction de la méthode

Introduite par J. Brzozowski (notion de dérivée) (1964) et Antimirov (notions de dérivées partielles) (1996).

Méthode purement algébrique ne nécessitant pas de construction explicite de l'automate.

Méthode fonctionne par calcul de la dérivée d'une expression régulière pour laquelle on veut construire un automate.

Intuitivement, la dérivée d'une expression régulière e sur un symbole a est une expression régulière décrivant ce qu'il manque après avoir lu a pour former un mot de e.

Avantage de la méthode :

- travaille uniquement sur la *syntaxe* des expressions régulières
- peut se faire *"à la volée"*

Soit Σ un alphabet (utilisé dans la suite pour construire des expressions régulières).

Préliminaire : terme constant d'une expression régulière

Opérateur qui indique si ϵ appartient au langage dénoté par une expression régulière.

Définition (Terme constant d'une expression régulière)

Le terme constant d'une expression régulière est une expression régulière donnée par la fonction c : $ER \to \{\epsilon, \emptyset\}$ définie par :

$$\mathrm{c}(e) = \begin{cases} \epsilon & \text{si } \epsilon \in L(e) \\ \emptyset & \text{sinon} \end{cases}$$

Afin de pouvoir être calculer directement à partir de l'expression régulière, le terme constant peut être également défini inductivement par :

- $\mathrm{c}(\emptyset) = \emptyset$
- $\mathrm{c}(\epsilon) = \epsilon$
- $\mathrm{c}(a) = \emptyset$, pour $a \in \Sigma$
- $\mathrm{c}(e + e') = \mathrm{c}(e) + \mathrm{c}(e')$
- $\mathrm{c}(e \cdot e') = \mathrm{c}(e) \cdot \mathrm{c}(e')$
- $\mathrm{c}(e^*) = \epsilon$

Exemple (Terme constant d'expressions régulières sur $\Sigma = \{a, b\}$)

- $\mathrm{c}(a) = \emptyset$.
- $\mathrm{c}(a^*) = \epsilon$.
- $\mathrm{c}(a \cdot b) = \emptyset$.
- $\mathrm{c}(a \cdot b)^* = \epsilon$.

Remarque Calculer le terme constant suppose de simplifier l'expression régulière résultat (dans le cas où il est calculé pour des expressions régulières composées). □

Dérivée d'une expression régulière par rapport à un symbole

Rappel : intuitivement, la dérivée d'une expression régulière e sur un symbole a est une expression régulière décrivant ce qu'il manque après avoir lu a pour former un mot de e.

Définition (Dérivée d'une expression régulière : une première définition)

La dérivée de $e \in ER$ sur $a \in \Sigma$ est l'expression régulière notée $\frac{\partial}{\partial a} e$ et dénotant le langage

$$\{w \in \Sigma^* \mid a \cdot w \in L(e)\} .$$

Nous donnons la précédence à l'opérateur de dérivation par rapport aux autres opérateurs.

Exemple (Dérivée d'une expression régulière)

- $\frac{\partial}{\partial a} a = \epsilon$
- $\frac{\partial}{\partial a} \epsilon = \emptyset$
- $\frac{\partial}{\partial a} (a + b) = \epsilon$
- $\frac{\partial}{\partial a} (a \cdot b) = b$
- $\frac{\partial}{\partial a} (a \cdot b)^* = b \cdot (a \cdot b)^*$

Comment calculer la dérivée d'une expression régulière quelconque ?

Dérivée d'une expression régulière par rapport à un symbole

Définition (Dérivée d'une expression régulière : définition inductive (pour le calcul))

La dérivée par rapport à un symbole $a \in \Sigma$ est définie inductivement sur la syntaxe des expressions régulières par :

- $\frac{\partial}{\partial a} \emptyset = \emptyset$
- $\frac{\partial}{\partial a} \epsilon = \emptyset$
- $\frac{\partial}{\partial a} a = \epsilon$
- $\frac{\partial}{\partial a} b = \emptyset$, lorsque $b \neq a$
- $\frac{\partial}{\partial a} (e + e') = \frac{\partial}{\partial a} e + \frac{\partial}{\partial a} e'$
- $\frac{\partial}{\partial a} (e \cdot e') = \frac{\partial}{\partial a} e \cdot e' + \mathrm{c}(e) \ \cdot \frac{\partial}{\partial a} e'$
- $\frac{\partial}{\partial a} e^* = \frac{\partial}{\partial a} e \cdot e^*$

Exemple (Dérivée d'une expression régulière par rapport un symbole)

- $\frac{\partial}{\partial a} a = \epsilon$
- $\frac{\partial}{\partial a} \epsilon = \emptyset$
- $\frac{\partial}{\partial a} (a + b) = \frac{\partial}{\partial a} a + \frac{\partial}{\partial a} b = \epsilon + \emptyset = \epsilon$
- $\frac{\partial}{\partial a} (a \cdot b) = \frac{\partial}{\partial a} a \cdot b + c(a) \cdot \frac{\partial}{\partial a} b = \epsilon \cdot b + \emptyset \cdot \emptyset = b$
- $\frac{\partial}{\partial a} (ab)^* = \frac{\partial}{\partial a} (ab) \cdot (a \cdot b)^* = b \cdot (a \cdot b)^*$

Dérivée d'une expression régulière par rapport à un mot

Intuitivement, dériver par rapport à un mot consiste à dériver *récursivement* et par rapport à chaque lettre du mot dans l'ordre de lecture de ce mot.

Définition (Dérivée d'une expression régulière par rapport à un mot)

La dérivée par rapport à un mot dans Σ^* est défini inductivement sur les mots par

- $\frac{\partial}{\partial \epsilon} e = e$
- $\frac{\partial}{\partial a \cdot u} e = \frac{\partial}{\partial u} dea$, avec $dea = \frac{\partial}{\partial a} e$

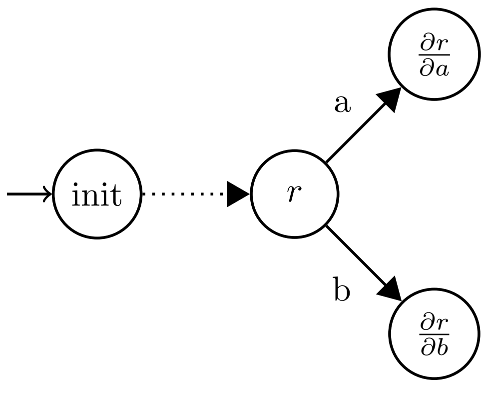
Exemple (Dérivée d'une expression régulière par rapport à un mot)

- $\frac{\partial}{\partial ab} ab = \epsilon$
- $\frac{\partial}{\partial a \cdot b}(a \cdot b)^* = \frac{\partial}{\partial b}\frac{\partial}{\partial a}(a \cdot b)^*$
  On a vu que $\frac{\partial}{\partial a}(a \cdot b)^* = b \cdot (a \cdot b)^*$.
  Donc $\frac{\partial}{\partial a \cdot b}(a \cdot b)^* = \frac{\partial}{\partial b} b \cdot (a \cdot b)^* = \underbrace{\frac{\partial}{\partial b} b}_{\epsilon} \cdot (a \cdot b)^* + \underbrace{\mathrm{c}(b)}_{\emptyset} \cdot \frac{\partial}{\partial b}(a \cdot b)^* = (a \cdot b)^*$

Construction d'un automate par la dérivée d'une expression régulière

Intuition

- Utiliser des expressions régulières pour les états de l'automate.
- On passe d'un état à l'autre sur un symbol en calculant sa dérivée sur ce symbole.
- La dérivée d'une expression régulière représente "ce qu'il reste à lire" après avoir lu un symbole pour reconnaître l'expression qu'on a dérivée.
- L'état initial contient l'expression régulière pour laquelle on construit l'automate.
- Un état q est terminal s'il ne reste plus qu'à lire ϵ, cad si $\mathrm{c}(q) = \epsilon$.



Construction d'un automate par la dérivée d'une expression régulière

Définition de l'automate

Finitude de l'ensemble des dérivées

L'ensemble des dérivées d'une expression régulière est fini modulo associativité, commutativité et idempotence des opérateurs.

Démonstration.

Admis. □

Soit $e \in ER$ une expression régulière définie sur un alphabet Σ et $\partial e \in \mathcal{P}(ER)$ l'ensemble des (expressions régulières) dérivées.

Définition (Automate des dérivées)

L'automate dérivée de e est l'AEFD $(\partial e, \Sigma, e, \delta_e, F_e)$ avec :

- $\delta_e(d, a) = \frac{\partial}{\partial a} d$, pour $d \in \partial e$ et $a \in \Sigma$,
- $F_e = \{d \in \partial e \mid \mathrm{c}(d) = \epsilon\}$.

Construction d'un automate par la dérivée d'une expression régulière

Exemple de construction de l'automate

Exemple (Construction de l'automate pour l'expression régulière $(a \cdot b)^*$)

On a vu que :

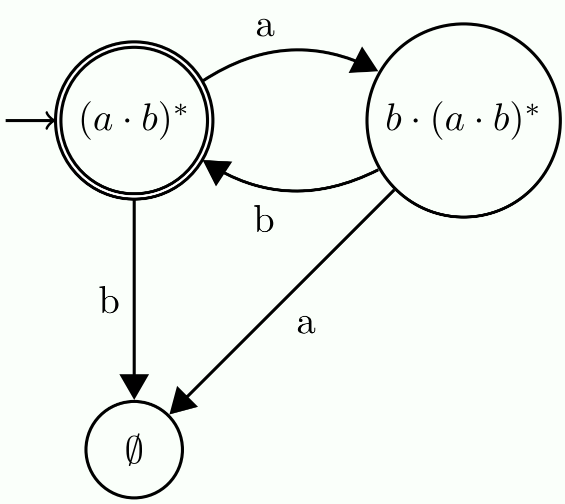
- $\frac{\partial}{\partial a}(a \cdot b)^* = b \cdot (a \cdot b)^*$
- $\frac{\partial}{\partial a \cdot b}(a \cdot b)^* = (a \cdot b)^*$

De plus, nous avons :

- $\frac{\partial}{\partial b}(a \cdot b)^* = \emptyset$
- $\frac{\partial}{\partial b}(b \cdot (a \cdot b)^* = (a \cdot b)^*$

Par ailleurs :

- $\mathrm{c}((a \cdot b)^*) = \epsilon$
- $\mathrm{c}(b \cdot (a \cdot b)^*) = \emptyset$
- $\mathrm{c}(\emptyset) = \emptyset$



Plan Chap. 8 - Expressions Régulières / Théorème de Kleene

Analyse lexicale

- Distinction entre la syntaxe (les phrases) et le lexique (les mots)
- Spécification des mots du langage du langage de programmation

Exemple (Lexique d'un langage de programmation)

- identificateurs
- constantes entières
- l'opérateur "+"
- mots clés du langage

Analyse lexicale

Entrée : sequence de caractères

Sortie : sequence de classes d'unités lexicales ($\sim$ séquence de mots)

1. Calcul de la plus grand séquence parmi un ensemble de classes lexicales
   $\hookrightarrow$ *lexemes* du programme
2. Insertion d' une référence dans la *table des symboles* pour les identificateurs.
3. Retour à l'analyseur syntaxique :
   - la classe lexicale (*token*) : constantes, identificateurs, mots clés, opérateurs, separateurs, . . .
   - l'élement associé à cette classe : le *lexeme*
4. Suppression des éléments hors du langage (espaces, commentaires, retours à la ligne, tabulations)
5. Token special : error – lorsque les règles du lexique ne sont pas respectées.

Basé sur des outils formels : les langages réguliers

- décrits par des expressions régulières
- reconnus par des automates à états finis (déterministes)

Exemple d'analyseur lexicale : LeX (générateur de code implémentant des automates à partir d'expressions régulières)

Univ. Grenoble Alpes, Département Licence Sciences et Technologies, Licence deuxième année

Une calculette avec deux opérations

Spécification du lexique

Syntaxe (grammaire hors-contexte) :

```
E : E + T | T
T : T * F | F
F : ID | NUM |(E)
```

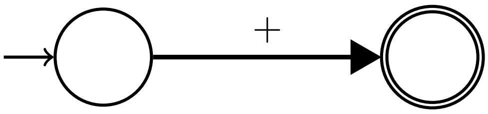
Lexique

- opérateurs : $\{+, *\}$
- séparateurs : ( )
- une constante entière NUM
- un identificateur ID

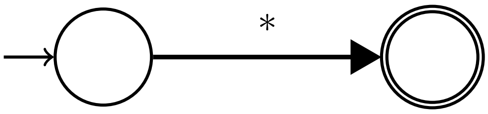
Une calculette avec deux opérations

Expressions régulières et automates reconnaisseurs pour le lexique

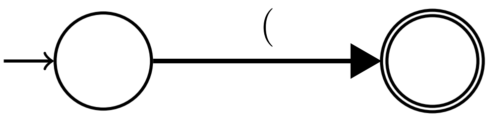
- +



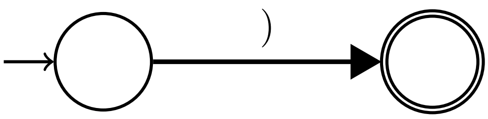
- *



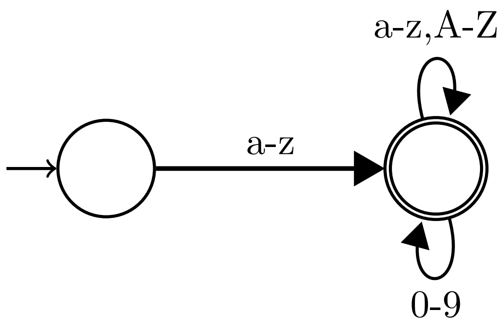
- (



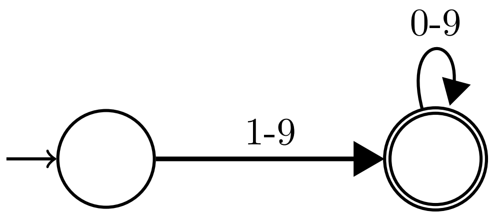
-)



- `[a-z][a-zA-Z0-9]*`



- `[1-9][0-9]*`



LeX : un générateur d'analyseurs lexicaux

Outil de génération d'analyseurs lexicaux écrits en langage C

Spécifications en LeX

```
déclarations
%%
règles
%%
procédures
```

Règles

```
modele1 {action1}
   ...
modele2 {action2}
```

LeX : un générateur d'analyseurs lexicaux

Exemple

Exemple (Règles)

1. Suppression des espaces redondants

```
%%
[ \t]+ printf(" ");
%%
yywrap(){return(1);}
```

2. Reconnaissance d'un entier

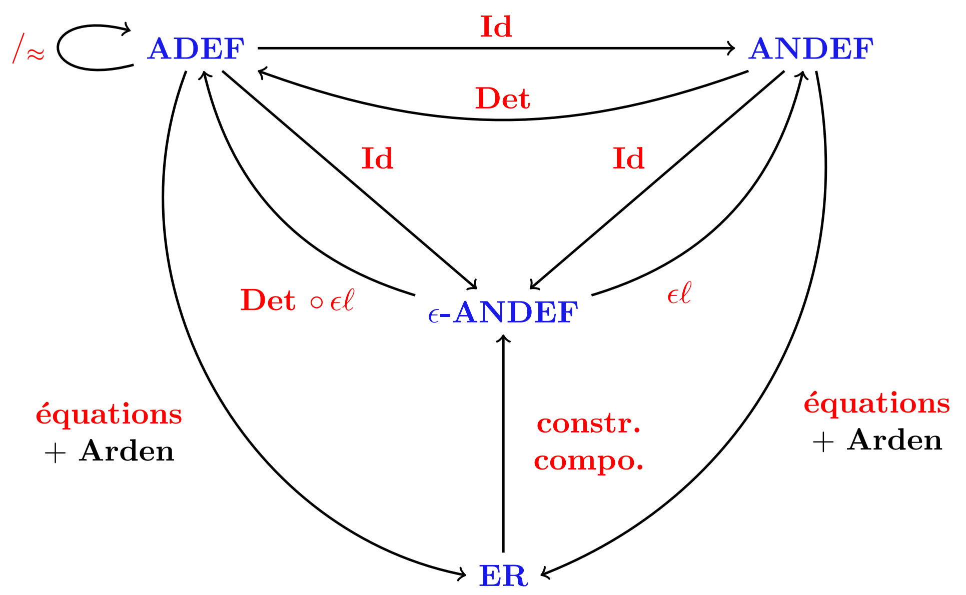
```
integer  {digit}+
%%
{integer} {attribut=atoi(yytext);return(Integer);}
```


Plan Chap. 8 - Expressions Régulières / Théorème de Kleene

Résumé I

- Théorème de Kleene et ses conséquences.
- Traduction des automates vers les expressions régulières :
  - Méthode associant des équations aux états (langages associés aux états) :
    - Équations associées aux états d'un automate
    - Lemme d'Arden
  - ~~Méthode par suppression des états~~
  - ~~Méthode associant des équations aux (langages associés aux chemins)~~
- Traduction des expressions régulières vers les automates
  - Méthode compositionnelle
  - ~~Méthode par calcul des dérivées~~
- Application en informatique : analyse lexicale en compilation

Résumé II



ADEF : automate déterministe
ANDEF : automate non-déterministe
ϵ-ANDEF : automate non-déterministe avec ϵ-transitions
ER : expressions régulières

$/_{\approx}$: minimisation
Id : identité
Det : déterminisation
$\epsilon\ell$: élimination des ϵ-transitions
constr. compo : construction compositionnelle