Théorie des langages et des automates
&
Compilation

LANGAGES RÉGULIERS & AUTOMATES FINIS

École Nationale d'Ingénieurs de Carthage
2ème année Ingénieur Informatique

E. Menif Abassi

Plan du module

Plan du chapitre

1. Concepts de base
2. Expressions régulières
3. Automates finis
4. Propriétés des langages réguliers

Plan du chapitre

1. Concepts de base
   a. Symboles
   b. Alphabets
   c. Mots
   d. Langages
2. Expressions régulières
3. Automates finis
4. Propriétés des langages réguliers
5. Limites des automates finis

Plan du chapitre

Concepts de base

a. Symboles

- *Éléments indivisibles* (non exprimables en autres symboles) qui servent à construire des mots

- **Les 26 lettres de l'alphabet : a, b, c, d, …**
- **Les chiffres : 0, 1, 2, 3, …**

Plan du chapitre

7

Concepts de base

8

b. Alphabets

- *Ensemble fini*, *non vide*, de symboles
- Par convention, on le désigne par la lettre Σ

- **$\Sigma = \{0,1\} \Rightarrow$ alphabet binaire**
- **$\Sigma = \{a, b, \ldots, z\} \Rightarrow$ alphabet des lettres minuscules**
- **$\Sigma = \{A, C, G, T\} \Rightarrow$ alphabet de l'ADN**

Plan du chapitre

Concepts de base

c. Mots

- Définition
  - Un mot (chaîne) w sur Σ est une *suite finie* de symboles appartenant à Σ
  - **1010 est un mot sur l'alphabet $\Sigma = \{0, 1\}$**
  - **compilation est un mot sur l'alphabet $\Sigma = \{a, b, \ldots, z\}$**
  - **CGTGCCAAAT est un mot sur l'alphabet $\Sigma = \{A, C, G, T\}$**

Concepts de base

c. Mots

- Longueur d'un mot
  - La longueur d'un mot w, notée $|w|$, est le nombre de symboles entrant dans la composition du mot
  - **$|1010| = 4$**
  - **|compilation| = 11**
  - **|CGTGCCAAAT| = 10**

Concepts de base

c. Mots

- Mot vide
  - Le mot vide, noté ε, est un mot ne contenant aucun symbole
  - Peut être construit pour tout alphabet
  - $|\varepsilon| = 0$
- Occurrences d'un symbole
  - La notation $|w|_x$, donne le nombre d'occurrences du symbole $x \in \Sigma$ dans le mot w
  - **$|11001100|_0 = 4$**

Concepts de base

13

c. **Mots**

- Concaténation: soient u et v deux mots non vides définis sur Σ tels que $u = x_1x_2x_3...x_n$ et $v = y_1y_2y_3...y_n$

  Le mot uv est la concaténation de u et de v tel que

  $$uv = x_1x_2x_3...x_ny_1y_2y_3...y_n$$

  - **Soient u = 10 et v= 01 définis sur $\Sigma = \{0, 1\}$ alors uv = 1001**
  - **Soient u = len et v= demain définis sur $\Sigma = \{a, b, c, ...\}$ alors uv = lendemain**

Concepts de base

14

c. **Mots**

- Concaténation
  - On note x^n, $n \in N$ la concaténation de n symboles identiques

    $$x^ix^j = x^{i+j}$$
    $$x^i = x^{i-1}x$$
    $$x^0 = \varepsilon \text{ et } x^1 = x$$

  - Le mot w^n , $n \in N$ la concaténation de n mots identiques

    $$w^iw^j = w^{i+j}$$
    $$w^i = w^{i-1}w$$
    $$w^0 = \varepsilon \text{ et } w^1 = u$$

Concepts de base

15

c. **Mots**

- Concaténation: Propriétés
  - $|uv| = |u| + |v|$
  - $|u^n| = n.|u|$, $n \in N$
  - La concaténation est associative:

    $\forall u, \forall v, \forall w, (uv)w = u(vw)$
  - La concaténation n'est pas commutative:

    Pour tout u, v tel que $u \neq v$ *alors* $uv \neq vu$
  - Élément neutre: $\varepsilon w = w\varepsilon = w$

Concepts de base

16

c. **Mots**

- Miroir
  - Le mot miroir ou transposé d'un mot $u = x_1 \ldots x_n$, où $x_i \in \Sigma$, noté u^R est le mot formé par la suite des symboles composant u mais pris dans l'ordre inverse et défini par : $u^R = x_n \ldots x_1$
  - Un mot qui est égal à son miroir est appelé palindrome
  - **Si u = 0101 alors u^R = 1010**
  - **Si $u = \varepsilon$ alors $u^R = u = \varepsilon$**
  - **Le mot *radar* est un palindrome**

Plan du chapitre

17

1. **Concepts de base**
   a. Symboles
   b. Alphabets
   c. Mots
   d. Langages
2. Expressions régulières
3. Automates finis
4. Propriétés des langages réguliers
5. Limites des automates finis

Concepts de base

18

d. Langages

- **Définition**
  - Ensemble de mots formés à partir d'un alphabet
  - L'ensemble de tous les mots formés à partir de Σ est noté Σ^* ($\varepsilon \in \Sigma^*$)
  - L'ensemble de tous les mots non vides formés à partir de Σ est noté Σ^+ ($\varepsilon \notin \Sigma^+$)
  - L'ensemble des mots de longueur k est noté Σ^k
  - Un langage L défini sur Σ est un sous-ensemble de Σ^* ($L \subseteq \Sigma^*$)

Concepts de base

19

d. Langages

- Exemple: soit l'alphabet $\sum$={0,1}
  - $\sum^0=\{\varepsilon\}$, $\sum^1=\{0,1\}$, $\sum^2=\{00,01,10,11\}$
  - $\sum^*=\{\varepsilon,0,1,00,01,10,11,000,001,010,011,\ldots\}$
  - $\sum^+=\{0,1,00,01,10,11,000,001,010,011,\ldots\}$

Concepts de base

20

d. Langages

- Exemple: soit l'alphabet $\sum$={0,1}
  - $\sum^*$ est un langage quelque soit l'alphabet $\sum$
  - $\varnothing$ est le langage vide quelque soit l'alphabet $\sum$
  - {ε} est le langage formé de la chaîne vide quelque soit l'alphabet $\sum$
  - Langage des mots formés d'une suite de n '0' suivie d'une suite de n '1' avec $n \geq 0$: L = {ε,01,0011,000111, …}
  - Langage des mots formés d'autant de '0' que '1' : L = {ε,01,10, 0011,1010,0101, …}
  - Langage des mots représentant les nombres premiers: L = {10,11,101, 111, …}

Concepts de base

d. Langages

- **Opérations ensembliste**: soient L_1 et L_2 deux langages définis sur Σ
  - Union: $L_1 \cup L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ ou } u \in L_2\}$ est un langage défini sur Σ
  - Intersection: $L_1 \cap L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ et } u \in L_2\}$ est un langage défini sur Σ
  - Complément: $\overline{L} = \{u \in \Sigma^* \mid u \notin L\}$ est un langage défini sur Σ
  - Différence: $L_1 - L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ et } u \notin L_2\}$ est un langage défini sur Σ

Concepts de base

d. Langages

- **Concaténation ou produit de langages**: soient L_1 et L_2 deux langages définis sur Σ:

  $L_1 L_2 = \{xy \in \Sigma^* \mid x \in L_1 \text{ et } y \in L_2\}$ est un langage défini sur Σ

  - $\{\varepsilon\}L = L\{\varepsilon\} = L$
  - $\{0,00\}\{\varepsilon,1,01\} = \{0,01,001,00,0001\}$
  - On note L^n, $n \in N$ la concaténation de n copies du langage L

    $L^i L^j = L^{i+j}$

    $L^0 = \{\varepsilon\}$ et $L^1 = L$

Concepts de base

23

d. Langages

- Fermeture de Kleene ou étoile de Kleene: soit L un langage défini sur Σ. La fermeture de Kleene de L, noté L^*, est l'ensemble de tous les mots qu'il est possible de construire en concaténant un nombre fini (éventuellement réduit à zéro) d'éléments du langage L

$$L^* = \bigcup_{i \geq 0} L^i$$

  - $\varepsilon \in L^*$
  - L^+= ?
  - **L'ensemble des suites de 0 et de 1 contenant la séquence 111 s'écrira {0,1}*{111}{0,1}***

Plan du chapitre

24

Plan du chapitre

25

1. Concepts de base
2. **Expressions régulières**
   a. **Définition**
   b. Construction des expressions régulières
   c. Extension des notations
   d. Lois algébriques sur les expressions régulières
   e. Définition régulière
3. Automates finis
4. Propriétés des langages réguliers
5. Limites des automates finis

Expressions régulières

26

a. Définition

- Une notation algébrique pour spécifier des langages réguliers contenant des motifs ou modèles (*patterns*) <u>récurrents</u>
  - Servent d'entrée aux systèmes qui traitent les chaînes de caractères (ex: commandes de recherche de chaînes)
    - Vérifier si une chaîne correspond à un motif: validation de données
    - Vérifier si une chaîne contient un motif: présence ou absence d'une information
    - Compter ou remplacer des motifs
  - Si E est une expression régulière alors on note $L(E)$ le langage engendré par E

Expressions régulières

27

a. Définition

- Trois opérateurs réguliers:
  - L'opérateur binaire de **somme** des expressions, noté **+** ou **|** , tel que $E \mid F$ ($E + F$) est l'expression régulière décrivant les chaînes générées soit par E, soit par F
  - L'opérateur binaire de **produit** des expressions, noté **.** ou sans notation , tel que EF ($E.F$) est l'expression régulière décrivant les chaînes générées par la concaténation de E et F
  - L'opérateur unaire de l'**itéré** d'une expression, noté *, tel que E^* est l'expression régulière décrivant la chaîne vide ε ou toute chaîne générée par le produit d'un nombre quelconque de E

Plan du chapitre

28

Expressions régulières

29

b. Construction des expressions régulières (définition récursive)

i. ε est une expression régulière et dénote le langage $L(\varepsilon) = \{\varepsilon\}$

ii. $\varnothing$ est une expression régulière et dénote le langage $L(\varnothing) = \varnothing$

iii. $\forall x \in \Sigma$, alors x est une expression régulière et dénote le langage $L(x) = \{x\}$

E. Menif Abassi TLA et compilation ENICAR

Expressions régulières

30

b. Construction des expressions régulières (définition récursive)

iv. Si E et F sont des expressions régulières alors:

- $E \mid F$ est une expression régulière et dénote l'union de $L(E)$ et $L(F)$. Ainsi, $L(E|F)=L(E) \cup L(F)$
- EF est une expression régulière et dénote la concaténation de $L(E)$ et $L(F)$. Ainsi, $L(EF)=L(E)L(F)$
- E^* est une expression régulière et dénote la fermeture de $L(E)$. Ainsi, $L(E^*) = (L(E))^*$
- (E) est une expression régulière et dénote le même langage que E. Ainsi, $L((E)) = L(E)$

E. Menif Abassi TLA et compilation ENICAR

Expressions régulières

31

b. Construction des expressions régulières (définition récursive)

- Priorité des opérateurs:

1. Fermeture de Kleene
2. Concaténation
3. Union

01*|1 équivaut (0(1*))|1

Expressions régulières

32

b. Construction des expressions régulières (définition récursive)

- **Pour un alphabet ∑ = {0,1}, voici une expression régulière qui dénote l'ensemble des mots alternant les 0 et les 1**

(01)*|(10)*|0(10)*|1(01)*

Plan du chapitre

Expressions régulières

c. Extension des notations

- Soit E une expression régulière:
  - L'opérateur unaire $^+$ signifie "au moins une fois" tel que $E^+= EE^*$
  - L'opérateur unaire **?** signifie "0 ou 1 fois" tel que $E?= E|\varepsilon$
  - Classes de caractères:
    - [abc] = a|b|c = a+b+c
    - [a-z] = a|b|c|…|z = a+b+c+…+z
    - [A-Ca-z] = A|B|C|a|b|c|…|z = A+B+C+a+b+c+…+z

Plan du chapitre

Expressions régulières

d. Lois algébriques sur les expressions régulières

- Deux expressions régulières sont égales (équivalentes) si elles engendrent (dénotent) le même langage
  - Commutativité de la réunion: $E \mid F = F \mid E$
  - Associativité de la réunion: $E \mid (F \mid G) = (E \mid F) \mid G$
  - Élément neutre de la réunion: $\varnothing \mid E = E \mid \varnothing = E$
  - Idempotence de la réunion: $E \mid E = E$

Expressions régulières

37

d. Lois algébriques sur les expressions régulières

- Associativité de la concaténation: $E(FG) = (EF)G$
- Élément neutre de la concaténation: $\varepsilon E = E\varepsilon = E$
- Élément absorbant de la concaténation: $\varnothing E = E\varnothing = \varnothing$
- distributivité à gauche de la concaténation par rapport à la réunion: $E(F|G) = (E|F)(E|G)$
- distributivité à droite de la concaténation par rapport à la réunion: $(E|F)G = (E|G)(F|G)$

E. Menif Abassi TLA et compilation ENICAR

Expressions régulières

38

d. Lois algébriques sur les expressions régulières

Quelques équivalences utiles pour la simplification (réduction):

- $(E^*)^* = E^*$
- $\varnothing^* = \varepsilon$
- $\varepsilon^* = \varepsilon$
- $E^+ = EE^* = E^*E$
- $E^+ \mid \varepsilon = E^*$
- $E^{**} = E^*$

- $E^*E^* = E^*$
- $(EF)^*E = E(FE)^*$
- $(E|F)^* = E^*(E|F)^*$
- $(E|F)^* = (E^*F^*)^*$
- $(E|F)^* = (E^*|F)^*$
- $(E|F)^* = (E^*F)^*E^*$

E. Menif Abassi TLA et compilation ENICAR

Plan du chapitre

Expressions régulières

e. Définition régulière

- Pour des raisons de commodité, il est possible de donner des noms explicites à des expressions régulières
- On utilise le symbole "→" pour écrire leur définition:

$$\mathbf{d_1 \rightarrow r_1}$$
$$\mathbf{d_2 \rightarrow r_2}$$
$$\mathbf{\ldots}$$
$$\mathbf{d_n \rightarrow r_n}$$

  - Chaque $\mathbf{d_i}$ est un nouveau symbole avec $\mathbf{d_i \notin \Sigma}$ et pour tout $\mathbf{d_j}$ tel que $\mathbf{i \neq j}$, on a $\mathbf{d_i \neq d_j}$
  - Chaque $\mathbf{r_i}$ est une expression régulière sur $\mathbf{\Sigma \cup \{d_1\ d_2 \ldots d_{i-1}\}}$

Expressions régulières

e. Définition régulière

lettre → [A-Za-z]
chiffre → [0-9]
id → lettre(lettre | chiffre)*
chiffres → chiffre(chiffre)*
fraction → . chiffres | ε
exposant → E(- | + | ε) chiffres | ε
nombre → chiffres fraction exposant

- id reconnaît: a, var0, a0b, begin
- nombre reconnaît : 0, 1.0, 2E4, 1.5E-8,
- nombre ne reconnaît pas : 0., .1, 1E2.0bb

Expressions régulières

- $L = \{w \mid w \in \Sigma *$, w ne termine pas par $aba\}$

  $((a+b)^*(aaa+aab+abb+baa+bab+bba+bbb))+(a+b+\varepsilon)(a+b+\varepsilon)$

- $L = \{w \mid w \in \Sigma *$, le troisième symbole de w est $a\}$.

  $(a+b)(a+b)a(a+b)^*$

- $L = \{w \mid w \in \Sigma *$, w ne contient pas la sous-chaîne $bb\}$.

  $(a+ba)^*(\varepsilon+b)$

Plan du chapitre

Plan du chapitre

1. Concepts de base
2. Expressions régulières
3. Automates finis
   a. Définitions
   b. Représentation graphique
   c. Représentation tabulaire
   d. Reconnaissance
   e. Langage reconnu par un automate fini
   f. Automates finis déterministes
   g. Automates finis non déterministes
   h. Complétion d'un automate
   i. Déterminisation d'un automate
   j. Minimisation d'un automate
4. Propriétés des langages réguliers
5. Limites des automates finis

Automates finis

45

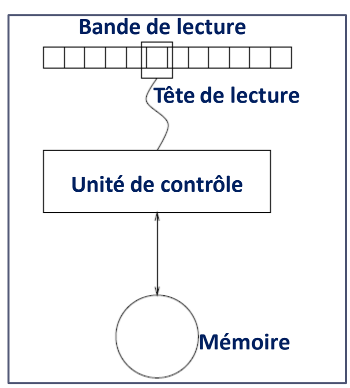
a. Définitions

- Automates: Outils de modélisation qui servent à représenter des dispositifs automatiques = machines abstraites
- Utilisés notamment pour la reconnaissance des langages ⇒ reconnaisseurs

E. Menif Abassi TLA et compilation ENICAR

Automates finis

46



Éléments d'un reconnaisseur

E. Menif Abassi TLA et compilation ENICAR

Automates finis

a. Définitions

- Reconnaisseur: une machine abstraite qui prend en entrée un mot et indique si ce mot appartient ou pas au langage décrit par la machine
  - Bande de lecture:
    - Une succession de cases.
    - Une case par caractère du mot à reconnaître
  - Mémoire:
    - Différentes formes
    - Stocker des éléments de l'alphabet

Automates finis

a. Définitions

- Tête de lecture:
  - Lit une case à la fois.
  - La case sur laquelle se trouve la tête de lecture à un moment donné s'appelle la case courante.
  - La tête peut être déplacée par le reconnaisseur pour se positionner sur la case immédiatement à gauche ou à droite de la case courante.

Automates finis

a. Définitions

- Unité de contrôle: cœur du reconnaisseur
  - Ensemble d'états, parmi lesquels:
    - États initiaux: états dans lesquels doit se trouver le reconnaisseur avant de commencer à reconnaître un mot
    - États d'acceptation: états dans lequel doit se trouver le reconnaisseur après avoir reconnu un mot
  - Fonction de transition: décrit le passage d'un état à un autre en fonction du contenu de la case courante de la bande de lecture et du contenu de la mémoire
  - Décider de la direction dans laquelle déplacer la tête de lecture
  - Choisir quels symboles stocker dans la mémoire

Automates finis

a. Définitions

- État d'un reconnaisseur à un moment donné est décrit par sa configuration:
  - État de l'unité de contrôle
  - Contenu de la bande de lecture et position de la tête de lecture
  - Contenu de la mémoire
- Configuration initiale:
  - Unité de contrôle dans un état initial
  - Tête de lecture sur la case se trouvant la plus à gauche de la bande de lecture
  - Mémoire contient un symbole initial donné

Automates finis

a. Définitions

- Configuration d'acceptation:
  - Unité de contrôle dans un état d'acceptation
  - Tête de lecture sur la case se trouvant la plus à droite de la bande de lecture
  - Mémoire dans un état d'acceptation
- Un reconnaisseur fonctionne en effectuant des mouvements.

Automates finis

a. Définitions

- Mouvement: passage d'une configuration C_i à la configuration C_j : $C_i \vdash C_j$
  - Une séquence de k mouvements est notée $\vdash^k$
  - Si $k \geq 0$, la séquence est notée $\vdash^*$
  - Si $k > 0$, la séquence est noté $\vdash^+$
  - Actions:
    1. Lire le symbole se trouvant dans la case courante
    2. Stocker de l'information dans la mémoire
    3. Changer d'état

Automates finis

a. Définitions

- Un reconnaisseur accepte un mot m:
  - m sur la bande de lecture
  - Il exsite une séquence de mouvements:

$$C_{Initiale} \vdash^{k} C_{Acceptation}$$

- Le langage reconnu par un reconnaisseur est l'ensemble des mots qu'il accepte

Automates finis

a. Définitions

- Automates finis comptent parmi les reconnaisseurs les plus simples avec un nombre fini d'états :
  - Mémoire nulle
  - La tête de lecture ne se déplace que d'une seule case vers la droite à chaque mouvement

Automates finis

55

a. Définitions

- Automate fini (AF) (*Finite Automaton*): est défini par le quintuple $(Q, \Sigma, \delta, q_0, F)$
  - Q est l'ensemble fini d'états
  - Σ est l'alphabet d'entrée
  - $\delta : Q \times \Sigma \rightarrow Q$ est la fonction de transition
  - $q_0 \in Q$ est l'état initial
  - $F \subseteq Q$ est l'ensemble des états d'acceptation

Automates finis

56

a. Définitions

- $\delta(q, \sigma) = p$ notée $q \xrightarrow{\sigma} p$ ou $\delta(q, \sigma, p)$ désigne l'état p auquel doit passer l'automate lorsque q est l'état courant, et σ est le caractère courant
  - = l'automate passe de l'état q à l'état p en lisant le symbole σ
  - = l'automate transite vers p sur σ

Automates finis

57

a. Définitions

- Si δ est totale (pour tout état $q \in Q$ il y a une transition pour tout symbole $\sigma \in \Sigma$), on dit que l'automate est complet

- Un automate fini non complet peut se trouver bloqué
- Un automate fini complet ne sera jamais bloqué

Automates finis

58

a. Définitions

- Une configuration d'un automate fini est un couple $(q,w) \in Q \times \Sigma^*$ avec $q \in Q$ et $w \in \Sigma^*$

- Une configuration initiale est de la forme (q_0,w)
- Une configuration d'acceptation est de la forme (q,ε) avec $q \in F$
- Un mouvement de l'automate fini est représenté comme suit:

$$(q,av) \vdash (q',v) \text{ si } \delta(q, a) = q'$$

Automates finis

a. Définitions

- Un mot w est reconnu par l'automate fini s'ils existent une suite de mouvements menant de la configuration (q_0,w) à (q,ε) avec $q \in F$
- Le langage reconnu par un automate fini A, noté $L(A)$, est l'ensemble des mots reconnus par A :

$$L(A) = \{w \in \Sigma^* \mid (q_0,w) \vdash^* (q, \varepsilon) \text{ avec } q \in F\}$$

Automates finis

- Soit l'automate $A_{2n}=(\{e_1,e_2\},\{0,1\},\delta, e_1,\{e_2\})$ avec:

$\delta(e_1, 1) = e_1$,

$\delta(e_1,0) = e_2$,

$\delta(e_2, 0) = e_2$,

$\delta(e_2, 1) = e_1$.

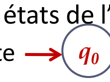
Le langage reconnu par A_{2n} , $L(A_{2n})$, est l'ensemble des nombres pairs en représentation binaire

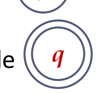
- L'unique séquence de mouvements de A_{2n} pour accepter le mot 0100 est: $(e_1,0100) \vdash (e_2,100) \vdash (e_1,00) \vdash (e_2,0) \vdash (e_2,\varepsilon)$

Plan du chapitre

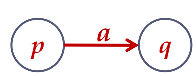
Automates finis

b. Représentation graphique

- Un diagramme de transition:
  - Les sommets, représentés par des cercles, sont les états de l'automate
    - Un état initial est désigné par une flèche entrante 

    - Un état d'acceptation est désigné par un double cercle 

  - Les arcs correspondent à la fonction de transition δ
    - Étiqueté par le symbole en entrée de la fonction
    - Il existe un arc entre les sommets p et q, étiqueté par le symbole a si et seulement si $\delta(p,a) = q$



Automates finis

b. Représentation graphique

- Un mot m est reconnu par un automate s'il existe un chemin dans le diagramme partant de l'état initial et se terminant par un état d'acceptation

$\Rightarrow$ la concaténation des étiquettes des arcs du chemin = w

- Le diagramme de transition pour l'automate A_{2n} =({e_1,e_2},{0,1},δ, e_1,{e_2})

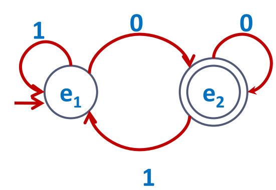
$\delta(e_1, 1) = e_1$,

$\delta(e_1,0) = e_2$,

$\delta(e_2, 0) = e_2$,

$\delta(e_2, 1) = e_1$.

Exécuter la reconnaissance de 0100



Plan du chapitre

Automates finis

65

c. Représentation tabulaire

- Une table de transition T:
  - Une ligne représente un état appartenant à Q
    - Un état initial est précédé par une flèche entrante $\rightarrow$
    - Un état d'acceptation est précédé par une *
  - Une colonne représente un symbole appartenant à Σ
  - Une entrée de la table $T[q,a]$ est l'état résultant de la fonction de transition $\delta(q,a) \Rightarrow T[q,a] = \delta(q,a)$

	σ_1	σ_2	…	σ_n
$\rightarrow q_0$	$\delta(q_0, \sigma_1)$	$\delta(q_0, \sigma_2)$	…	$\delta(q_0, \sigma_n)$
q_1	$\delta(q_1, \sigma_1)$	$\delta(q_1, \sigma_2)$	…	
…	…	…	…	…
$*q_m$ $(\in F)$	$\delta(q_m, \sigma_1)$	$\delta(q_m, \sigma_2)$	…	$\delta(q_m, \sigma_n)$

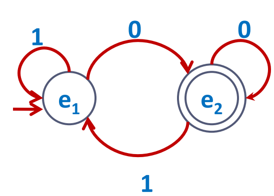
Automates finis

66

c. Représentation tabulaire

- La table de transition pour l'automate A_{2n} = $(\{e_1,e_2\},\{0,1\},\delta,e_1,\{e_2\})$

	0	1
$\rightarrow e_1$	e_2	e_1
$*e_2$	e_2	e_1



Plan du chapitre

67

Automates finis

68

d. Reconnaissance

- La reconnaissance d'un mot w se calcule en exactement $|w|$ étapes
- Chaque étape de calcul correspond à un mouvement de l'automate
  - Lecture d'un symbole du mot
  - Application de la fonction de transition δ
  - Changement d'état

Automates finis

d. Reconnaissance: Définition par induction de la reconnaissance

- L'extension de la fonction de transition δ aux mots est notée δ^* définie comme suit:

$$\delta^*: Q \times \Sigma^* \rightarrow Q$$

tel que $\delta^*(q,w)$ représente l'état où s'arrête la reconnaissance si cette dernière commence à l'état q et reçoit le mot w en entrée

- Pour tout $q \in Q$, $\delta^*(q,\varepsilon) = q$
- Pour tout $q \in Q$, tout $y \in \Sigma^*$ et tout $\sigma \in \Sigma$ on a:

$$\delta^*(q, y\sigma) = \delta(\delta^*(q, y), \sigma)$$

Automates finis

d. Reconnaissance: Algorithme de reconnaissance

// $m = m_1 m_2 \ldots m_n$
$q = q_0$
$i = 1$
Tant que ($i \leq n$) **Faire**
    $q = \delta(q, m_i)$
    $i = i + 1$
Fin Tant que

Si ($q \in F$)
    alors retourner (vrai)
    sinon retourner (faux)
Fin Si

Automates finis

d. Reconnaissance

- Soit le langage L = {w|w **possède un nombre pair de 0 et de 1** }. Le rôle des états de l'automate, qui doit reconnaître ce langage, est de compter le nombre d'occurrences des 0 et des 1 déjà lus modulo 2 ⇒ Chaque état doit représenter si le nombre des 0 et des 1 déjà lus est pair ou impair.
- Quatre états possibles:
  - q_0: un nombre pair de 0 et de 1
  - q_1 : un nombre pair de 0 et un nombre impair de 1
  - q_2 : un nombre impair de 0 et un nombre pair de 1
  - q_3 : un nombre impair de 0 et un nombre impair de 1

Automates finis

d. Reconnaissance

110101

$\delta^*(q_0,110101) = \delta(\delta^*(q_0,11010),1) = \delta(q_1,1) = q_0$

$\delta^*(q_0,11010) = \delta(\delta^*(q_0,1101),0) = \delta(q_3,1) = q_1$

$\delta^*(q_0,1101) = \delta(\delta^*(q_0,110),1) = \delta(q_2,1) = q_3$

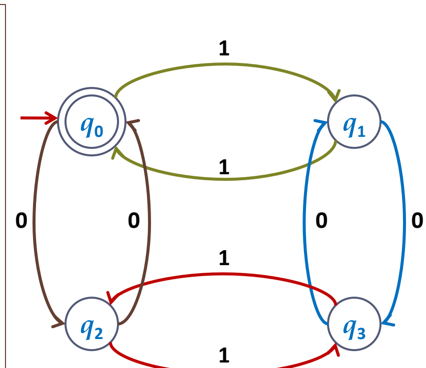
$\delta^*(q_0,110) = \delta(\delta^*(q_0,11),0) = \delta(q_0,1) = q_2$

$\delta^*(q_0,11) = \delta(\delta^*(q_0,1),1) = \delta(q_1,1) = q_0$

$\delta^*(q_0,1) = \delta(\delta^*(q_0,\varepsilon),1) = \delta(q_0,1) = q_1$

$\delta^*(q_0,\varepsilon) = q_0$

A = $(\{q_0, q_1, q_2, q_3\},\{0,1\},\delta, q_0,\{q_0\})$



Plan du chapitre

Automates finis

e. Langage reconnu par un AF

- Soit un automate fini $A = (Q,\Sigma, \delta ,q_0,F)$, un mot w est reconnu par A si $\delta^*(q_0, w) \in F$ sinon il est rejeté par A
- Un langage reconnu par A, noté $L(A)$, est défini comme suit:

$$L(A) =\{w\in\Sigma^* \mid \delta^*(q_0, w) \in F\}$$

- Si un langage L est reconnu par A, alors $L = L(A)$, L est dit un *langage reconnaissable*

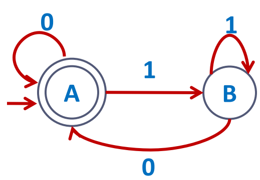
Automates finis

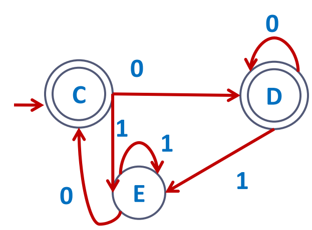
75

e. Langage reconnu par un AF

- Deux automates A_1 et A_2 sont deux automates équivalents ssi ils reconnaissent le même langage

Exemple: **deux automates équivalents**





Plan du chapitre

76

Automates finis

77

f. Automates finis déterministes (AFD)

- Automate fini avec une propriété qui est le déterminisme
- Déterminisme: ne jamais avoir le choix entre plusieurs exécutions
  $\Rightarrow$ pour toute configuration d'un automate fini, il existe au plus un mouvement possible = à partir d'un état courant et pour tout symbole lu, il existe au plus une transition
- δ est une fonction totale: l'automate est complet et non ambigu
- On ne peut pas effectuer de transition sur ε

Plan du chapitre

78

Automates finis

79

g. Automates finis non déterministes (AFND)

- Automate fini qui n'est pas déterministe
- Non déterminisme: pour une configuration d'un automate fini, il peut exister plusieurs mouvements possibles $\Rightarrow$ à partir d'un état courant et un symbole lu, il peut existes plusieurs choix de transitions
- δ est une relation: elle retourne un ensemble d'états
- $\delta : Q \times \Sigma \rightarrow \wp(Q)$

Automates finis

80

g. Automates finis non déterministes (AFND)

- $\delta(q, \sigma) = \{p_1, p_2, \ldots, p_n\}$ est l'ensemble des états p_i auxquels peut passer l'automate lorsque q est l'état courant, et σ est le symbole courant

- L'extension de la fonction de transition δ aux mots est notée δ^* définie comme suit: $\delta^*: Q \times \Sigma^* \rightarrow \wp(Q)$
  - Pour tout $q \in Q$, $\delta^*(q,\varepsilon) = \{q\}$
  - Pour tout $q \in Q$, tout $y \in \Sigma^*$ et tout $\sigma \in \Sigma$ on a:

    Si $\delta^*(q, y) = \{p_1, p_2, \ldots, p_k\}$ alors $\delta^*(q, y\sigma) = \bigcup_{i=1}^{k} \delta(p_i, \sigma)$

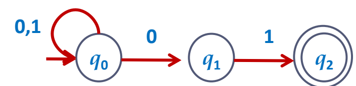
- Un langage reconnu par A est défini comme suit:

$$L(A) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \cap F \neq \varnothing\}$$

Automates finis

81

Soit le langage $L = \{w \mid w \text{ termine par } 01\} \Rightarrow A = (\{q_0, q_1, q_2\},\{0,1\},\delta, q_0,\{q_2\})$



	0	1
$\rightarrow q_0$	$\{q_0,q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_2\}$
$*q_2$	$\varnothing$	$\varnothing$

00101

$\delta^*(q_0,00101) = \delta(q_0,1)\cup\delta(q_1,1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

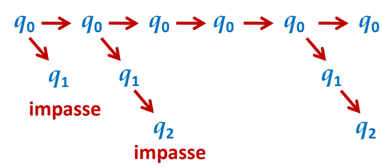
$\delta^*(q_0,0010) = \delta(q_0,0)\cup\delta(q_2,0) = = \{q_0, q_1\} \cup\varnothing = \{q_0, q_1\}$

$\delta^*(q_0,001) = \delta(q_0,1)\cup\delta(q_1,1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

$\delta^*(q_0,00) = \delta(q_0,0)\cup\delta(q_1,0) = \{q_0, q_1\} \cup\varnothing = \{q_0, q_1\}$

$\delta^*(q_0,0) = \delta(\delta^*(q_0,\varepsilon),0) = \delta(q_0,0) = \{q_0, q_1\}$

$\delta^*(q_0,\varepsilon) = \{q_0\}$



Automates finis

82

g. Automates finis non déterministes (AFND)

- Le non-déterminisme ne paye pas: La généralisation du modèle d'automate fini liée à l'introduction de transitions non déterministes est, du point de vue des langages reconnus, sans effet : tout langage reconnu par un automate fini non-déterministe est aussi reconnu par un automate déterministe.
- Déterminisme ou non?:
  - Avantage des automates déterministes : leur efficacité en termes de calcul
  - Avantage des automates non déterministes : Moins de transitions

Automates finis

83

g. Automates finis non déterministes (AFND)

➢ Déterminisme ou non?:

Si un langage est reconnu par un automate fini, alors il est également reconnu par un automate fini déterministe.

- Si l'automate fini du départ est déterministe, c'est évident
- Si l'automate de départ n'est pas déterministe, on se propose de construire un automate fini déterministe qui intègre tous les choix existant dans l'automate de départ par l'algorithme de déterminisation

Plan du chapitre

84

Automates finis

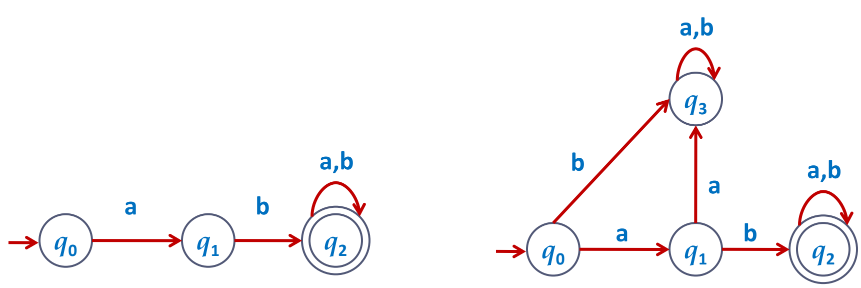
85

h. Complétion d'un automate

- Comment rendre un automate complet?
  - Un automate partiellement spécifié peut être complété par ajout d'un état puits absorbant les transitions absentes de l'automate original, sans pour autant changer le langage reconnu.
  - soit $A = (Q, \Sigma, \delta, q_0, F)$ un automate partiellement spécifié, on définit $A' = (Q', \Sigma, \delta', q_0, F)$ avec :
    - $Q' = Q \cup \{q_p\}$
    - $\forall q \in Q, \sigma \in \Sigma, \delta'(q, \sigma) = \delta(q, \sigma)$ si $\delta(q, \sigma)$ existe, $\delta(q, \sigma) = q_p$ sinon
    - $\forall \sigma \in \Sigma, \delta'(q_p, \sigma) = q_p$

Automates finis

86



A non complet A après complétion

Plan du chapitre

Automates finis

i. Déterminisation d'un automate

- **Objectif**: éliminer l'ambigüité d'un automate. Le résultat sera un automate complet non ambigu

- Soit $N = (Q_N, \Sigma_N, \delta_N, q_N, F_N)$ un automate non déterministe reconnaissant un langage L. On veut construire un automate déterministe $D = (Q_D, \Sigma_D, \delta_D, q_D, F_D)$ qui reconnait le même langage $\Rightarrow L(N) = L(D)$

- L'idée générale de la construction consiste à créer des états qui correspondent à des <u>ensembles d'états</u> de $N \Rightarrow$ lorsque N peut transiter d'un état q vers les états $p_1, \ldots, p_n$ sur un symbole σ, un nouvel état e correspondant à cet ensemble est créé et les différentes transitions sont remplacées par une transition de q vers e sur σ.

Automates finis

89

i. Déterminisation d'un automate

- **Algorithme**:
  1. L'ensemble des états de D est l'ensemble des parties de N: $Q_D = \wp(Q_N)$
  2. L'alphabet de D est le même que pour N: $\Sigma_D = \Sigma_N$
  3. L'état initial de D est l'ensemble constitué de l'état initial de N: $q_D = \{q_N\}$
  4. Pour $S \in Q_D$ et $\sigma \in \Sigma_D$, on calcule $\delta_D(S, \sigma)$ comme suit:

     $$\delta_D(S, \sigma) = \cup\, \delta_N(S, \sigma)$$

  5. Les états d'acceptation de D sont les ensembles qui contiennent au moins un état d'acceptation de N :

     $$F_D = \{S \in Q_D \mid S \cap F_N \neq \varnothing\}$$

Automates finis

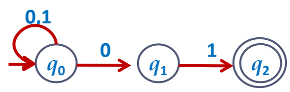
90

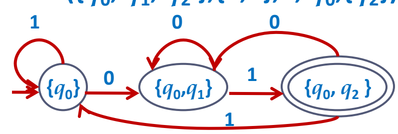
i. Déterminisation d'un automate

- **Algorithme**:
  - Dans la pratique, pour construire un automate déterministe D à partir d'un automate non déterministe N, on ne commence pas par créer tous les états de D , car ils peuvent être nombreux ($|\wp(Q)| = 2^{|Q|}$).
  - ⇒ La construction des états de D se fait au fur et à mesure de leur création en partant de l'état initial.

Automates finis

Soit le langage L = {w|w termine par 01 } $\Rightarrow$ N = ({q_0, q_1, q_2 },{0,1},δ, q_0,{q_2})





	0	1
$\varnothing$	$\varnothing$	$\varnothing$
$\rightarrow\{q_0\}$	$\{q_0,q_1\}$	$\{q_0\}$
$\{q_1\}$	$\varnothing$	$\{q_2\}$
$*\{q_2\}$	$\varnothing$	$\varnothing$
$\{q_0, q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$
$*\{q_0, q_2\}$	$\{q_0,q_1\}$	$\{q_0\}$
$*\{q_1, q_2\}$	$\varnothing$	$\{q_2\}$
$*\{q_0, q_1, q_2\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$

	0	1
$\cancel{\varnothing}$	$\cancel{\varnothing}$	$\cancel{\varnothing}$
$\rightarrow\{q_0\}$	$\{q_0,q_1\}$	$\{q_0\}$
$\cancel{\{q_1\}}$	$\cancel{\varnothing}$	$\cancel{\{q_2\}}$
$\cancel{*\{q_2\}}$	$\cancel{\varnothing}$	$\cancel{\varnothing}$
$\{q_0, q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$
$*\{q_0, q_2\}$	$\{q_0,q_1\}$	$\{q_0\}$
$\cancel{*\{q_1, q_2\}}$	$\cancel{\varnothing}$	$\cancel{\{q_2\}}$
$\cancel{*\{q_0, q_1, q_2\}}$	$\cancel{\{q_0,q_1\}}$	$\cancel{\{q_0,q_2\}}$

Plan du chapitre

1. Concepts de base
2. Expressions régulières
3. **Automates finis**
    a. Définitions
    b. Représentation graphique
    c. Représentation tabulaire
    d. Reconnaissance
    e. Langage reconnu par un automate fini
    f. Automates finis déterministes
    g. Automates finis non déterministes
    h. Complétion d'un automate
    i. Déterminisation d'un automate
    j. **Minimisation d'un automate**
4. Propriétés des langages réguliers
5. Limites des automates finis

Automates finis

93

j. Minimisation d'un automate

- Plusieurs AFD peuvent accepter le même langage ⇒ Il paraît naturel de vouloir minimiser le nombre d'états d'un AFD acceptant un langage donné

- L'automate minimal qui reconnaît un langage donné est unique. L'unicité de l'automate minimal permet de vérifier que deux automates sont équivalents (reconnaissent le même langage)

Automates finis

94

j. Minimisation d'un automate

- Le principe est de partitionner les états en blocs ou classes tels que chaque bloc ou classe contient des états équivalents

- Deux états p et q sont équivalents si pour tout mot w , si $\delta^* (p , w) \in F$ alors $\delta^* (q , w) \in F$ (mènent à des états d'acceptation qui ne sont pas forcément les mêmes). Dans le cas contraire, p et q sont distinguables

- **Objectif**: Soit $A = (Q, \Sigma, \delta, q_0, F)$ un automate déterministe (complet, non ambigu) reconnaissant un langage L. On veut construire un automate déterministe minimal $A' = (Q', \Sigma, \delta', q_0', F')$ qui reconnait le même langage ⇒ $L(A) = L(A')$

Automates finis

95

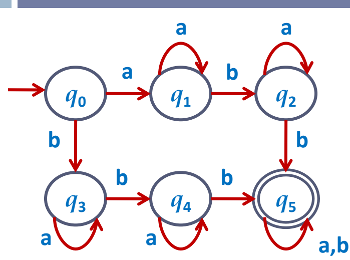
j. Minimisation d'un automate

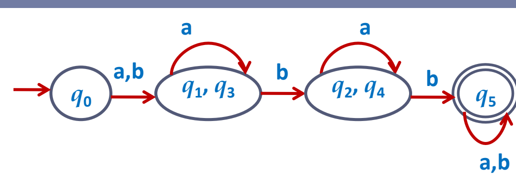
- **Méthode1: Algorithme de Moore**:

1. Construire une partition Π_0 initiale de Q composée de deux groupes: les états d'acceptation F et les états de non acceptation $Q \setminus F$
2. Itérer jusqu'à stabilisation:
   - Pour toute paire d'états p et q dans la même classe d'une partition Π_k s'il existe $\sigma \in \Sigma$ tel que $\delta(p, \sigma)$ et $\delta(q, \sigma)$ ne sont pas dans la même classe pour Π_k, alors ils sont dans deux classes différentes de Π_{k+1}. On dit que σ est un symbole séparant
3. Choisir un état dans chaque classe de la partition comme représentant de ce groupe.
   - Ces états constituent Q', les états de l'automate minimal
   - La fonction de transition δ' est construite comme suit: Pour chaque état q de Q', si $\delta(q, \sigma) = p$ alors $\delta'(q, \sigma) = p'$ avec p' est le représentant de la classe de p.
   - q_0'est le représentant de la classe de q_0
   - F'est l'ensemble des représentants des classes des éléments de F
4. Supprimer les états puits et les états non accessibles

Automates finis

96





	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_1	q_2
q_2	q_2	q_5
q_3	q_3	q_4
q_4	q_4	q_5
$*q_5$	q_5	q_5

$\Pi_0 = \{\{q_0; q_1; q_2; q_3; q_4\}; \{q_5\}\}$

$\Pi_1 = \{\{q_0; q_1; q_3\};\{q_2; q_4\};\{q_5\}\}$

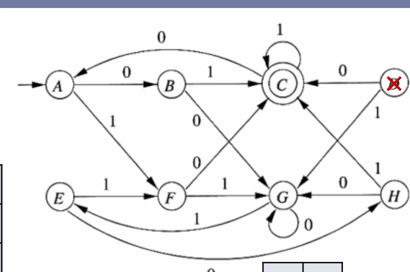
$\Pi_2 = \{\{q_0\}; \{q_1; q_3\};\{q_2; q_4\};\{q_5\}\}$

Automates finis

j. Minimisation d'un automate

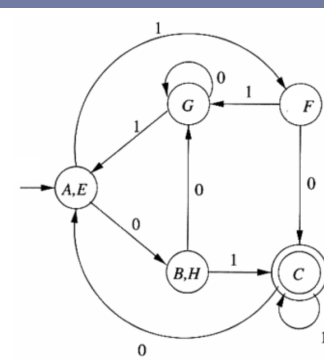
- **Méthode2: Table-filling algorithm**:
  1. Éliminer les états non accessibles
  2. Si p est un état d'acceptation et q est un état de non acceptation alors (p, q) est une paire d'états distinguables
  3. Soient p et q deux états tels que pour un symbole σ nous avons $\delta(p, \sigma) = r$ et $\delta(q, \sigma) = s$ avec (r,s) une paire d'états distinguables alors (p,q) est aussi une paire d'états distinguables
  4. Les états de l'automate minimal est l'ensemble des états mutuellement équivalents

Automates finis

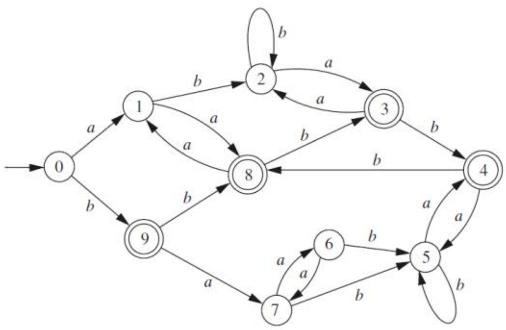


	0	1
A	B	F
B	G	C
C	A	C
D	C	G
E	H	F
F	C	G
G	G	E
H	G	C

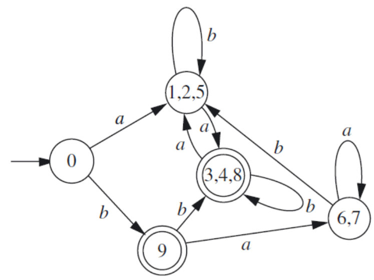
B	2					
C	1	1				
E		2	1			
F	2	2	1	2		
G	3	2	1	3	2	
H	2		1	2	2	2
	A	B	C	E	F	G



Automates finis



Automate non minimal



Automate minimal

Plan du chapitre

1. Concepts de base
2. Expressions régulières
3. Automates finis
4. **Propriétés des langages réguliers**
   a. Théorème de Kleene
   b. Propriétés de fermeture
   c. Expressions régulières ⇔ Automates finis
5. Limites des automates finis

Plan du chapitre

Propriétés des langages réguliers

a. Théorème de Kleene

- Théorème: Un langage est reconnaissable (reconnu par un automate fini) si et seulement si il est régulier (dénoté par une expression rationnelle)

- On s'intéresse à un type de propriétés permettant de démontrer si un langage est régulier ou non:
  - Propriétés de fermetures

Plan du chapitre

103

Propriétés des langages réguliers

104

b. Propriétés de fermeture

- Les langages réguliers sont fermés par les opérations régulières

- La classe des langages réguliers est fermée pour une opération donnée si pour deux automates A_1 et A_2 on peut construire l'automate A qui reconnaît le langage qui résulte de l'application de cette opération aux deux langages $L(A_1)$ et $L(A_2)$.

- Théorème: Soient L_1 et L_2 deux langages réguliers. Les langages $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 L_2$, $\overline{L}$, L^R et L^* sont des langages réguliers

Plan du chapitre

1. Concepts de base
2. Expressions régulières
3. Automates finis
4. **Propriétés des langages réguliers**
   a. Théorème de Kleene
   b. **Propriétés de fermeture**
      i. **Union**
      ii. **Intersection**
      iii. **Concaténation**
      iv. **Complémentation**
      v. **Fermeture de Kleene**
      vi. **Image miroir**
   c. Expressions régulières ⇔ Automates finis
5. Limites des automates finis

Plan du chapitre

1. Concepts de base
2. Expressions régulières
3. Automates finis
4. **Propriétés des langages réguliers**
   a. Théorème de Kleene
   b. **Propriétés de fermeture**
      i. **Union**
      ii. Intersection
      iii. Concaténation
      iv. Complémentation
      v. Fermeture de Kleene
      vi. Image miroir
   c. Expressions régulières ⇔ Automates finis
5. Limites des automates finis

Propriétés des langages réguliers

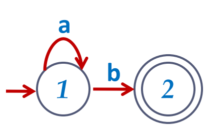
107

i. Union (Méthode 1)

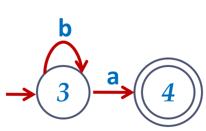
- Soient L_1 et L_2 deux langages réguliers reconnus respectivement par $A_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ et $A_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ deux AF.
On construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$ qui reconnait le langage $L_1 \cup L_2$ comme suit:
  - $Q = Q_1 \cup Q_2 \cup \{q_0\}$
  - $\Sigma = \Sigma_1 \cup \Sigma_2$
  - δ :
    - $\delta(p,\sigma) = \delta_1(p, \sigma)$ pour tout $p \in Q_1$ et pour tout $\sigma \in \Sigma$ (les transitions du premier automate sont gardées)
    - $\delta(p,\sigma) = \delta_2(p, \sigma)$ pour tout $p \in Q_2$, pour tout $\sigma \in \Sigma$ (les transitions du deuxième automate sont gardées)
    - $\delta(q_0,\sigma) = \delta_1(q_1, \sigma) \cup \delta_2(q_2, \sigma)$ pour tout $\sigma \in \Sigma$ (le nouvel état initial q_0 doit simuler le comportement des états initiaux q_1 et q_2)
  - F :
    - $F_1 \cup F_2 \cup \{q_0\}$ si $q_1 \in F_1$ ou $q_2 \in F_2$
    - $F_1 \cup F_2$ sinon

Propriétés des langages réguliers

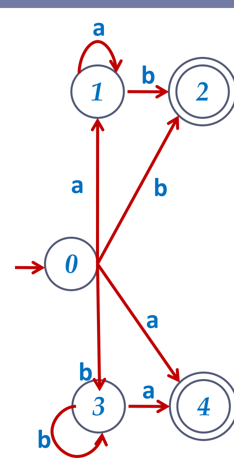
108



Automate A_1



Automate A_2



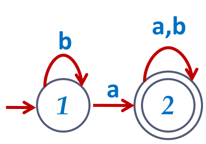
Automate A qui reconnait $L(A_1) \cup L(A_2)$

Propriétés des langages réguliers

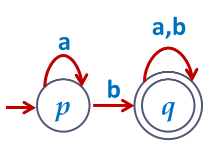
i. Union (Méthode 2)

- Soient L_1 et L_2 deux langages réguliers reconnus respectivement par $A_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ et $A_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ deux AF. On construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$ qui reconnait le langage $L_1 \cup L_2$ comme suit:
  - $Q = Q_1 \times Q_2$
  - $\Sigma = \Sigma_1 \cup \Sigma_2$
  - $\delta((p,q),\sigma) = (\delta_1((p,\sigma), \delta_2((q,\sigma))$ pour tout $p \in Q_1$, pour tout $q \in Q_2$ et pour tout $\sigma \in \Sigma$
  - $q_0 = (q_1, q_2)$
  - $F = \{(p,q) \mid p \in F_1$ <u>ou</u> $q \in F_2\} = (F_1 \times Q_2) \cup (Q_1 \times F_2)$

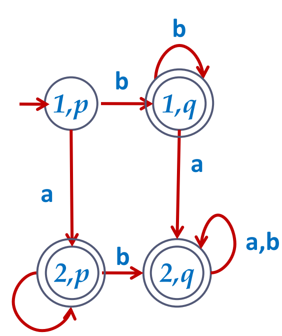
Propriétés des langages réguliers



Automate A_1



Automate A_2



Automate A qui reconnait L(A_1)∪L(A_2)

Plan du chapitre

Propriétés des langages réguliers

ii. Intersection (Méthode 1)

- Soient L_1 et L_2 deux langages réguliers reconnus respectivement par $A_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ et $A_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ deux AF. On construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$ qui reconnait le langage $L_1 \cap L_2$ en construisant l'automate qui reconnait le langage $\overline{\overline{L_1} \cup \overline{L_2}}$

Propriétés des langages réguliers

113

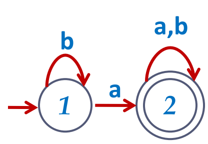
ii. Intersection (Méthode 2)

➢ Soient L_1 et L_2 deux langages réguliers reconnus respectivement par $A_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ et $A_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ deux AF. On construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$ qui reconnait le langage $L_1 \cap L_2$ comme suit:

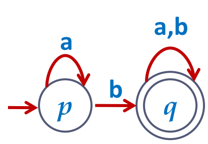
- $Q = Q_1 \times Q_2$
- $\Sigma = \Sigma_1 \cup \Sigma_2$
- $\delta((p,q),\sigma) = (\delta_1((p, \sigma), \delta_2((q, \sigma))$ pour tout $p \in Q_1$, pour tout $q \in Q_2$ et pour tout $\sigma \in \Sigma$
- $q_0 = (q_1, q_2)$
- $F = \{(p,q) \mid p \in F_1$ <u>et</u> $q \in F_2\} = F_1 \times F_2$

Propriétés des langages réguliers

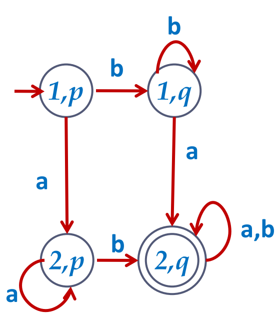
114



Automate A_1



Automate A_2



Automate A qui reconnait $L(A_1) \cap L(A_2)$

Plan du chapitre

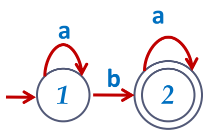
Propriétés des langages réguliers

iii. Concaténation

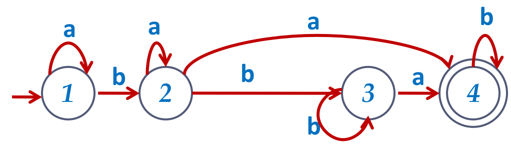
- Soient L_1 et L_2 deux langages réguliers reconnus respectivement par $A_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ et $A_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ deux AF. On construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$ qui reconnait le langage $L_1 L_2$ comme suit:
  - $Q = Q_1 \cup Q_2$
  - $\Sigma = \Sigma_1 \cup \Sigma_2$
  - δ :
    - $\delta(p,\sigma) = \delta_1(p, \sigma)$ pour tout $p \in Q_1$ et pour tout $\sigma \in \Sigma$ (les transitions du premier automate sont gardées)
    - $\delta(p,\sigma) = \delta_2(p, \sigma)$ pour tout $p \in Q_2$, pour tout $\sigma \in \Sigma$ (les transitions du deuxième automate sont gardées)
    - $\delta(p,\sigma) = q$ pour tout $p \in F_1$, tout $\sigma \in \Sigma$ et tout $q \in Q_2$ avec $\delta_2(q_2, \sigma)= q$ (tout état final du premier automate doit simuler le comportement de l'état initial du deuxième automate)
  - $q_0 = q_1$
  - F:
    - $F_1 \cup F_2$ si $q_2 \in F_2$
    - F_2 sinon

Propriétés des langages réguliers

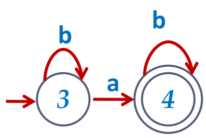
117



Automate A_1



Automate A qui reconnait L(A_1)L(A_2)



Automate A_2

Plan du chapitre

118

1. Concepts de base
2. Expressions régulières
3. Automates finis
4. **Propriétés des langages réguliers**
   a. Théorème de Kleene
   b. **Propriétés de fermeture**
      i. Union
      ii. Intersection
      iii. Concaténation
      iv. **Complémentation**
      v. Fermeture de Kleene
      vi. Image miroir
   c. Expressions régulières ⇔ Automates finis
5. Limites des automates finis

Propriétés des langages réguliers

iv. Complémentation

- Soit L un langage régulier reconnu par $A = (Q, \Sigma, \delta, q_0, F)$. On construit l'automate $A' = (Q', \Sigma', \delta', q_0', F')$ qui reconnait le langage $\overline{L}$ comme suit:
  - *Déterminiser l'automate*
  - $Q' = Q$
  - $\Sigma' = \Sigma$
  - $\delta' = \delta$
  - $q_0' = q_0$
  - $F' = Q \setminus F$ (les états finaux deviennent non finaux et inversement)

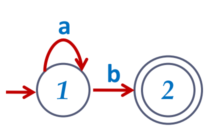
Plan du chapitre

Propriétés des langages réguliers

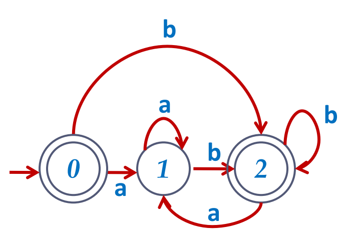
V. Fermeture de Kleene

- Soit L un langage régulier reconnu par $A = (Q, \Sigma, \delta, q_0, F)$. On construit l'automate $A' = (Q', \Sigma', \delta', q_0', F')$ qui reconnait le langage L^* comme suit:
  - $Q' = Q \cup \{q_0'\}$ (ajouter un nouvel état initial qui est aussi final)
  - $\Sigma' = \Sigma$
  - δ' :
    - $\delta'(q,\sigma) = \delta(q,\sigma), \forall q \in Q$ et $\forall \sigma \in \Sigma$ (les transitions de l'automate sont gardées)
    - $\delta'(q,\sigma) = p, \forall q \in F'$, $\delta(q_0,\sigma) = p$ et $\forall \sigma \in \Sigma$ (tout état final doit simuler le comportement de l'ancien état initial)
  - $F' = F \cup \{q_0'\}$

Propriétés des langages réguliers



Automate A



Automate A qui reconnait L(A)*

Plan du chapitre

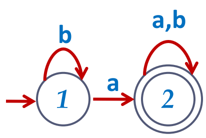
Propriétés des langages réguliers

vi. Image miroir

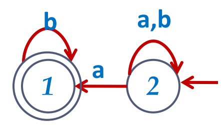
- Soit L un langage régulier reconnu par $A = (Q, \Sigma, \delta, q_0, F)$. On construit l'automate $A^R = (Q^R, \Sigma^R, \delta^R, q_0^R, F^R)$ qui reconnait le langage L^R comme suit:
  - $Q^R = Q \cup \{q_0^R\}$ ajouter un nouvel état initial s'il y a plus d'un état final sinon $Q^R = Q$ et q_0^R est l'état final
  - $\Sigma' = \Sigma$
  - δ^R:
    - $\delta^R(q,\sigma) = \{p \in Q \mid q \in \delta(p,\sigma)\}, \forall q \in Q, \sigma \in \Sigma$ (les transitions de l'automate sont inversées)
    - $\delta^R(q_0^R,\sigma) = \{p \in Q \mid p \in \delta^R(q,\sigma) \text{ et } q \in F\}$ (le nouvel état initial simule le comportement des anciens états finaux)
  - $F^R = \{q_0\}$

Propriétés des langages réguliers

125



Automate A



Automate A^R

Plan du chapitre

126

1. Concepts de base
2. Expressions régulières
3. Automates finis
4. Propriétés des langages réguliers
   a. Théorème de Kleene
   b. Propriétés de fermeture
   c. Expressions régulières ⇔ Automates finis
      i. Expressions régulières ⇒ Automates finis
      ii. Automates finis ⇒ Expressions régulières
5. Limites des automates finis

Plan du chapitre

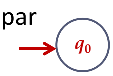
127

Propriétés des langages réguliers

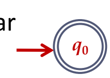
128

i. Expressions régulières $\Rightarrow$ Automates finis

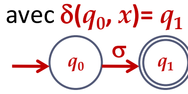
- Soit E une expression régulière qui dénote un langage L. Il existe un automate A qui reconnait $L \Rightarrow L(E) = L(A) = L$.
  1. Si $E = \varnothing$. E dénote le langage $\varnothing$ qui est aussi le langage reconnu par l'automate suivant : $A = (\{q_0\},\Sigma,\delta, q_0,\varnothing)$ avec $\delta(p,\sigma)=\varnothing$, $\forall p,\forall \sigma$



  2. Si $E = \varepsilon$. E dénote le langage $\{\varepsilon\}$ qui est aussi le langage reconnu par l'automate suivant : $A = (\{q_0\},\Sigma,\delta, q_0,\{q_0\})$ avec $\delta(p, \sigma)=\varnothing$, $\forall p,\forall \sigma$



  3. Si $E = x$ ($x\in\Sigma$). E dénote le langage $\{x\}$ qui est aussi le langage reconnu par l'automate suivant : $A = (\{q_0,q_1\},\Sigma,\delta, q_0,\{q_1\})$ avec $\delta(q_0, x)= q_1$ et $\delta(p, \sigma)=\varnothing$, $\forall p\neq q_0,\forall \sigma \neq x$



Propriétés des langages réguliers

i. Expressions régulières ⇒ Automates finis

- Soit E une expression régulière qui dénote un langage L. Il existe un automate A qui reconnait $L \Rightarrow L(E) = L(A) = L$.
  4. Si $E = E_1 \mid E_2$. E dénote le langage $L(E_1) \cup L(E_2)$ qui est aussi le langage reconnu par l'automate construit à partir de la propriété de fermeture pour l'union des langages $L(E_1)$ et $L(E_2)$
  5. Si $E = E_1\, E_2$. E dénote le langage $L(E_1)L(E_2)$ qui est aussi le langage reconnu par l'automate construit à partir de la propriété de fermeture pour la concaténation des langages $L(E_1)$ et $L(E_2)$
  6. Si $E = E_1{}^*$. E dénote le langage $L(E_1)^*$ qui est aussi le langage reconnu par l'automate construit à partir de la propriété de fermeture pour la fermeture de Kleene du langage $L(E_1)$

Propriétés des langages réguliers

Construire un automate pour l'expression régulière suivante:

$$(a+b)^*aba$$

Plan du chapitre

131

1. Concepts de base
2. Expressions régulières
3. Automates finis
4. Propriétés des langages réguliers
   a. Théorème de Kleene
   b. Propriétés de fermeture
   c. Expressions régulières ⇔ Automates finis
      i. Expressions régulières ⇒ Automates finis
      ii. Automates finis ⇒ Expressions régulières
         - Par élimination des états
         - Par résolution d'équations
5. Limites des automates finis

Plan du chapitre

132

1. Concepts de base
2. Expressions régulières
3. Automates finis
4. Propriétés des langages réguliers
   a. Théorème de Kleene
   b. Propriétés de fermeture
   c. Expressions régulières ⇔ Automates finis
      i. Expressions régulières ⇒ Automates finis
      ii. Automates finis ⇒ Expressions régulières
         - Par élimination des états
         - Par résolution d'équations
5. Limites des automates finis

Propriétés des langages réguliers

133

ii. Automates finis ⇒ Expressions régulières: Par élimination des états

- Soit A un AFD qui reconnait un langage L. Il existe une expression régulière E qui dénote $L \Rightarrow L(A) = L(E) = L$.

  1. Transformation de l'automate A en un *automate généralisé* AG
  2. Transformation de AG en une expression régulière E

Propriétés des langages réguliers

134

ii. Automates finis ⇒ Expressions régulières : Par élimination des états

- Automate généralisé
  - Un automate fini dont les transitions sont étiquetées par des expressions régulières et non pas simplement des symboles ou le mot vide.
  - Il lit le mot à reconnaître par blocs de symboles.
  - Trois contraintes:
    - **i. L'état initial possède des transitions vers les autres états, mais aucun état n'a de transition vers l'état initial**
    - **ii. Il n'y a qu'un état d'acceptation qui ne possède aucune transition vers d'autres états, mais tous les autres états possèdent une transition vers l'état d'acceptation. De plus, l'état d'acceptation est distinct de l'état initial**
    - **iii. À l'exception de l'état d'acceptation et de l'état initial, tous les états possèdent une transition et une seule vers tous les autres états et vers l'état lui même**

Propriétés des langages réguliers

ii. Automates finis ⇒ Expressions régulières: Par élimination des états

- Transformation d'un AFD en AG
  1. Ajouter un nouvel état initial avec une transition ε vers l'ancien état initial
  2. Ajouter un nouvel état d'acceptation vers lequel il existe une transition ε partant des anciens états d'acceptation
  3. S'il existe plusieurs transitions entre deux états, elles sont remplacées par une transition unique étiquetée par l'union des étiquettes des différentes transitions
  4. Finalement, des transitions étiquetées $\varnothing$ sont ajoutées entre les états qui ne sont reliés par aucune transition

Propriétés des langages réguliers

ii. Automates finis ⇒ Expressions régulières: Par élimination des états

- Transformation d'un AG en expression régulière
  - Transformation itérative
  - Initialement le nombre d'états $k \geq 2$
  - À chaque itération le nombre d'états est réduit de 1 ($k = k$-1) ⇒ réduction d'états
  - Condition d'arrêt: $k = 2$ ⇒ Il ne reste que l'état initial et l'état d'acceptation ⇒ l'étiquette de la transition entre ces deux états constitue l'expression régulière équivalente à l'automate de départ

Propriétés des langages réguliers

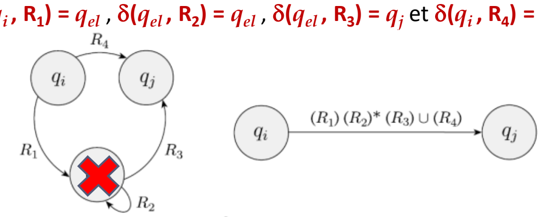
137

ii. Automates finis ⇒ Expressions régulières: Par élimination des états

- Étape de réduction d'états: AG (k états)→ AG'(k-1 états)
  1. Choisir un état à éliminer $q_{el} \in Q$ avec $q_{el} \neq q_0$ et $q_{el} \notin F$
  2. $AG'(Q', \Sigma, \delta', q_0, q_F)$ est définit comme suit:
     - $Q' = Q - \{q_{el}\}$
     - Pour tout $q_i \in Q' - \{q_0\}$ et tout $q_j \in Q' - \{q_F\}$ alors:

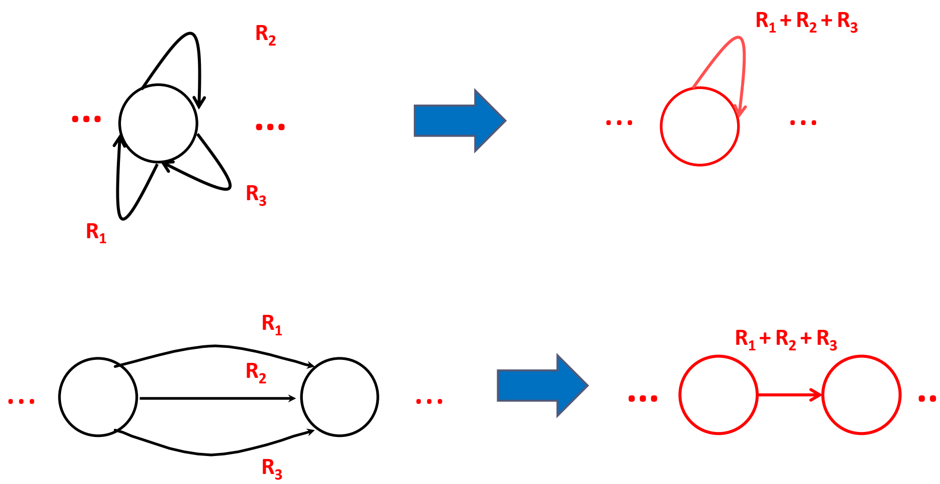
$\delta'(q_i, r) = q_j$ avec r = $(R_1)(R_2)^*(R_3) \mid (R_4)$ tel que

$\delta(q_i, R_1) = q_{el}$, $\delta(q_{el}, R_2) = q_{el}$, $\delta(q_{el}, R_3) = q_j$ et $\delta(q_i, R_4) = q_j$



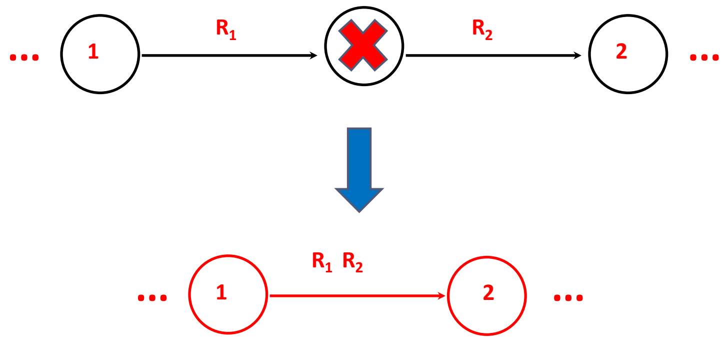
Propriétés des langages réguliers

138



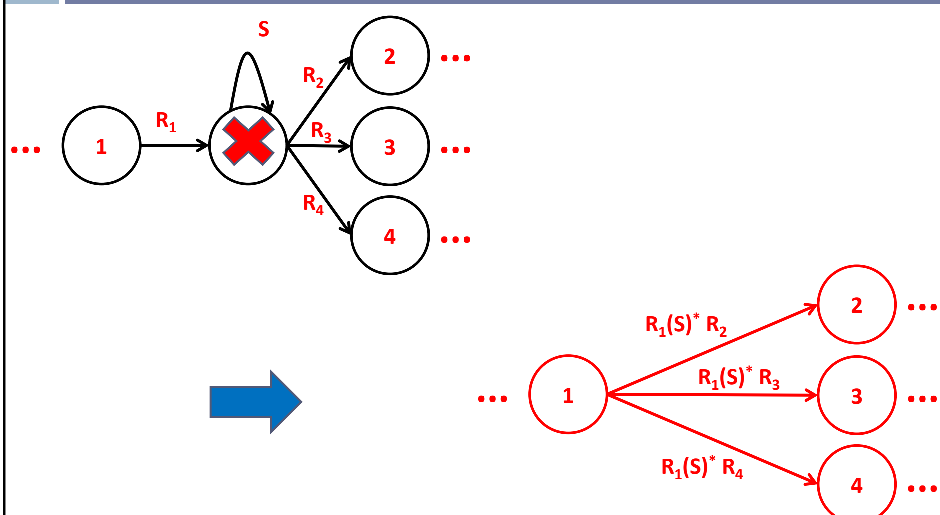
Propriétés des langages réguliers

139



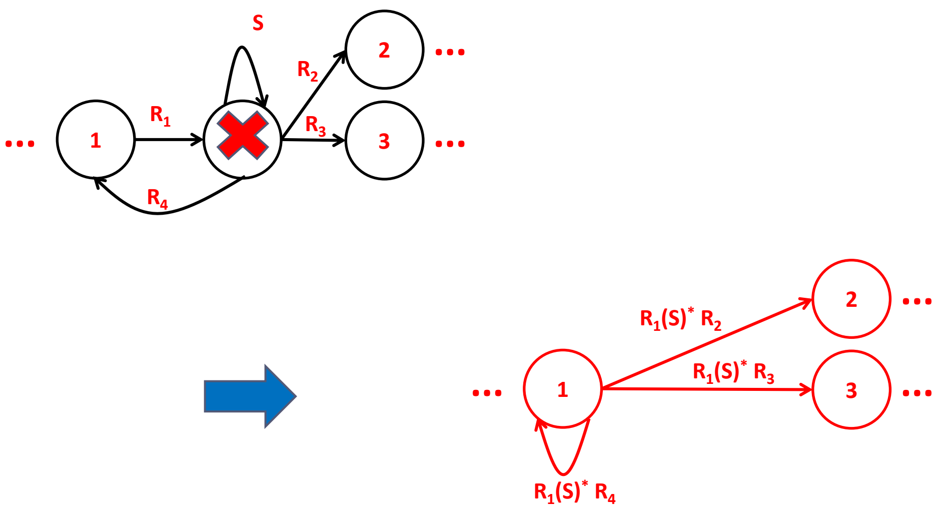
Propriétés des langages réguliers

140



Propriétés des langages réguliers

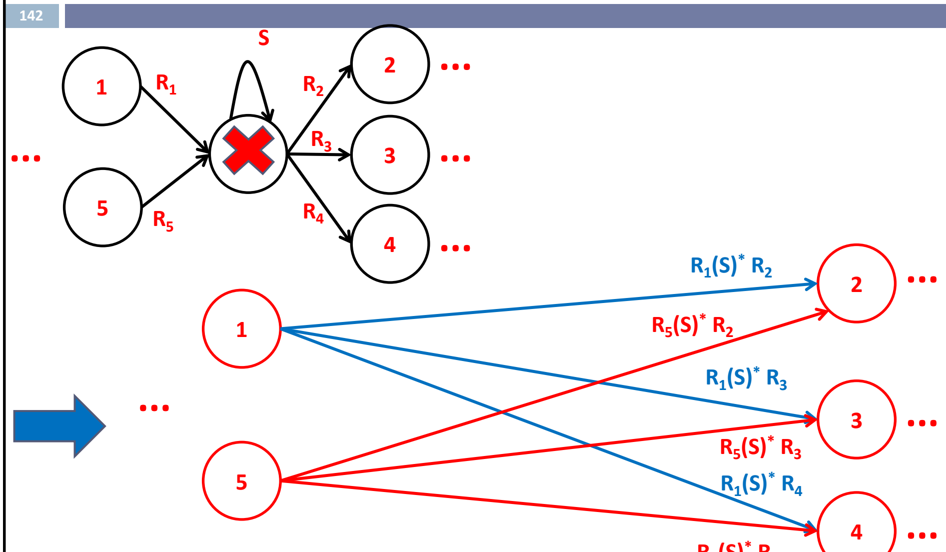
141



E. Menif Abassi TLA et compilation ENICAR

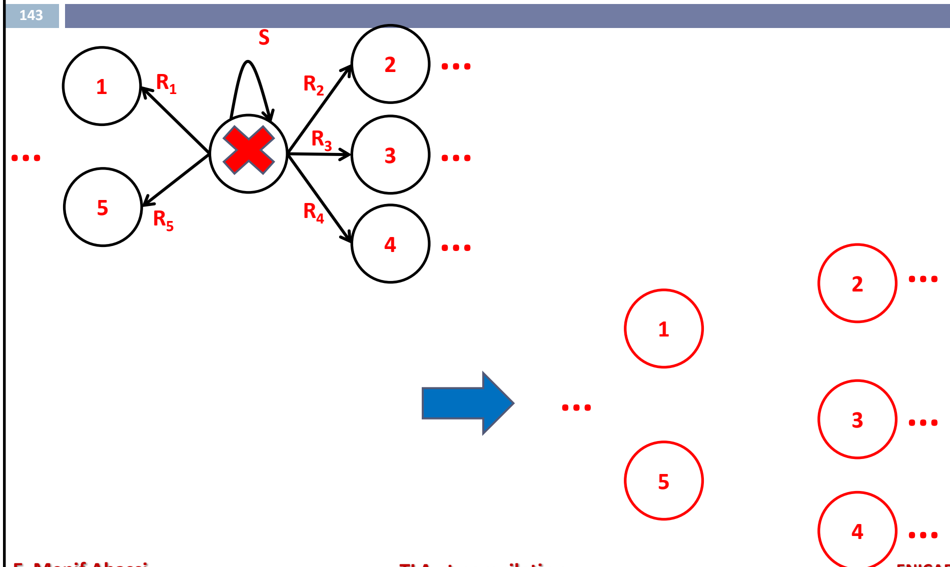
Propriétés des langages réguliers

142

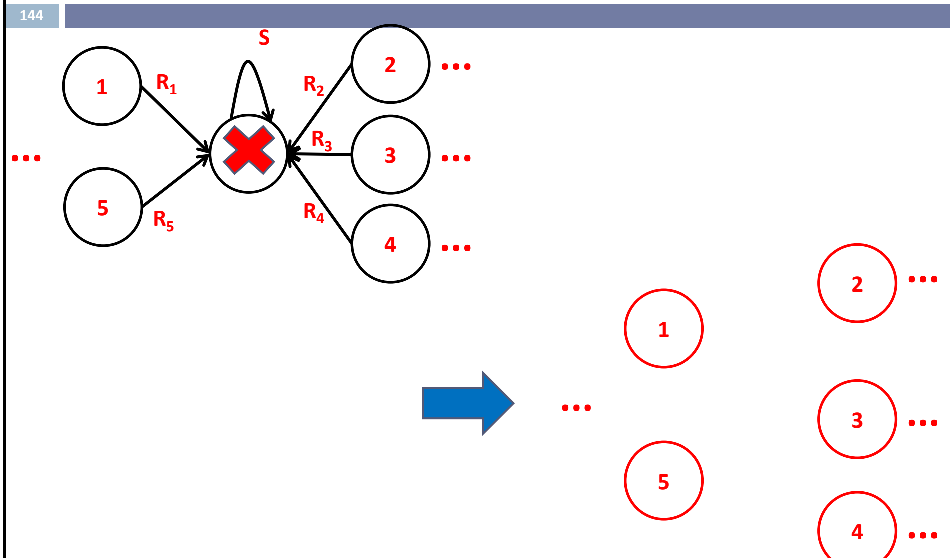


E. Menif Abassi TLA et compilation ENICAR

Propriétés des langages réguliers

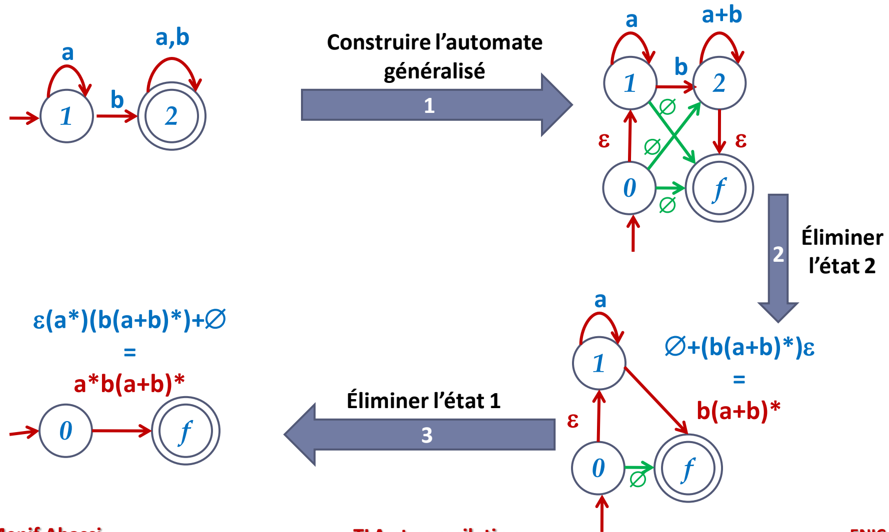


Propriétés des langages réguliers



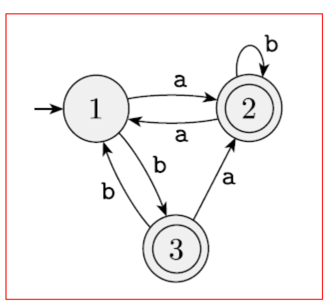
Propriétés des langages réguliers

145



Propriétés des langages réguliers

146



Quelle est l'expression régulière qui dénote
le langage reconnu par cet automate?

Plan du chapitre

147

1. Concepts de base
2. Expressions régulières
3. Automates finis
4. **Propriétés des langages réguliers**
   a. Théorème de Kleene
   b. Propriétés de fermeture
   c. **Expressions régulières $\Leftrightarrow$ Automates finis**
      i. Expressions régulières $\Rightarrow$ Automates finis
      ii. **Automates finis $\Rightarrow$ Expressions régulières**
         - Par élimination des états
         - **Par résolution d'équations**
5. Limites des automates finis

Propriétés des langages réguliers

148

ii. Automates finis $\Rightarrow$ Expressions régulières: Par résolution d'équations

- Pour tout état p de l'automate A, notons L_p l'ensemble des mots qui font passer A de l'état p à un état d'acceptation
- $L(A) = L_{q_0}$
- Le système $L_p = \begin{cases} \sum_{q \in Q} \alpha_{p,q}.L_q \text{ si } p \text{ n'est pas final} \\ \varepsilon + \sum_{q \in Q} \alpha_{p,q}.L_q \text{ si } p \text{ est final} \end{cases}$ définit le langage L_p
- Avec $\alpha_{p,q} = \{\sigma \in \Sigma, \delta(p,\sigma) = q\}$

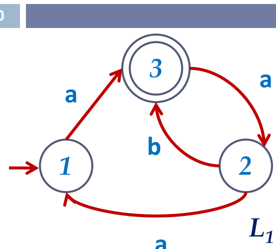
Propriétés des langages réguliers

ii. Automates finis ⇒ Expressions régulières: Par résolution d'équations

- On résout un tel système grâce au *Lemme d'Arden*

- *Lemme d'Arden*: Soient X , L_1 et L_2 trois langages définis sur Σ.
  On considère l'équation $X = L_1 X + L_2$ alors $L_1{}^*L_2$ est la solution unique à cette équation si $\varepsilon \notin L_1$
  On considère l'équation $X = XL_1 + L_2$ alors $L_2L_1{}^*$ est la solution unique à cette équation si $\varepsilon \notin L_1$

Propriétés des langages réguliers



$X = L_1 X + L_2 \Rightarrow X = L_1{}^*L_2$

$L_1 = aL_3 = aaL_2 + a$
$L_3 = aL_2 + \varepsilon$
$L_2 = aL_1 + bL_3 = aL_1 + baL_2 + b = baL_2 + aL_1 + b$

La solution pour L_2 *est* $(ba)^*(aL_1+b)$ *d'où :*

$L_1 = aa\,(ba)^*(aL_1+b) + a$
$\quad = aa\,(ba)^*aL_1 + aa\,(ba)^*b + a$

La solution pour L_1 *est* $(aa\,(ba)^*a)^*(aa\,(ba)^*b + a)$

Plan du chapitre

151

1. Concepts de base
2. Expressions régulières
3. Automates finis
4. Propriétés des langages réguliers
5. **Limites des automates finis**

Limites des automates finis

152

- Impossible de « compter » dans le cas général: capacité de calcul limitée:
  - Pas d'automate qui décrive un mot ayant autant de a que de b
  - Pas d'automate pour reconnaître une expression arithmétique bien formée