IN 406 – Théorie des Langages
Cours 4 : Expression régulière

Franck Quessette – Franck.Quessette@uvsq.fr

Université de Versailles – Saint-Quentin

V4 2020–2021

Rappel 1

Notions déjà vues :

- lettre, alphabet **fini** ;
- mot, langage **fini** ou **infini** ;
- langage rationnel (opérations ensemblistes) ;
- automate **fini** non-déterminsite (AFN), automate **fini** déterministe ;
- langage reconnaissable par automate **fini**.

Expression régulière

Une $\boxed{\textbf{expression régulière}}$ (e.r. ou regexp) est une chaîne de caractère qui définit un langage. Pour tout alphabet Σ, $\mathcal{L}(\mathbf{e})$ est le langage défini par l'**expression régulière** $\boxed{\mathbf{e}}$:

- $\mathbf{e} = \boxed{\emptyset}$ est une $\boxed{\textbf{e.r.}}$ et $\mathcal{L}(\mathbf{e}) = \boxed{\emptyset}$;
- $\mathbf{e} = \boxed{\varepsilon}$ est une $\boxed{\textbf{e.r.}}$ et $\mathcal{L}(\mathbf{e}) = \boxed{\{\varepsilon\}}$;
- $\mathbf{e} = \boxed{\mathbf{a}}$ pour tout $a \in \Sigma$ est une $\boxed{\textbf{e.r.}}$ et $\mathcal{L}(\mathbf{e}) = \boxed{\{a\}}$;

Soient **x** et **y** deux $\boxed{\textbf{e.r.}}$:

- $\mathbf{e} = \boxed{(\mathbf{x})}$ est une $\boxed{\textbf{e.r.}}$ et $\mathcal{L}(\mathbf{e}) = \mathcal{L}(\mathbf{x})$;
- $\mathbf{e} = \boxed{\mathbf{x*}}$ est une $\boxed{\textbf{e.r.}}$ et $\mathcal{L}(\mathbf{e}) = \mathcal{L}(\mathbf{x})^*$;
- $\mathbf{e} = \boxed{\mathbf{x+y}}$ est une $\boxed{\textbf{e.r.}}$ et $\mathcal{L}(\mathbf{e}) = \mathcal{L}(\mathbf{x}) \cup \mathcal{L}(\mathbf{y})$;
- $\mathbf{e} = \boxed{\mathbf{xy}} = \boxed{\mathbf{x{\cdot}y}}$ est une $\boxed{\textbf{e.r.}}$ et $\mathcal{L}(\mathbf{e}) = \mathcal{L}(\mathbf{x}) \cdot \mathcal{L}(\mathbf{y})$.

Expression régulière

Pour simplifier la notation on utilise la priorité des opérateurs :

$$\boxed{()} > \boxed{*} > \boxed{\cdot} > \boxed{+}$$

Expression régulière

Soit $\Sigma = \{a, b, c\}$ un alphabet et les e.r. suivantes :

- $e_1 = \boxed{\mathbf{(b+ab*aa)*ab*}} = \boxed{\mathbf{((b+((a\cdot(b*))\cdot a\cdot a))*)\cdot a\cdot(b*)}}$;
- $e_2 = \boxed{\mathbf{(a+b+c)*}}$: tous les mots sur Σ, $\mathcal{L}(e_2) = \Sigma^*$;
- $e_3 = \boxed{\mathbf{ab(a+b+c)*}}$: tous les mots qui commencent par ab ;
- $e_4 = \boxed{\mathbf{(a+b+c)*ac}}$: tous les mots qui se terminent par ac ;
- $e_5 = \boxed{\mathbf{((a+b+c)(a+b+c))*}}$: tous les mots avec un nombre pair de lettres ;

Équivalence des représentations

Définition

L est un $\boxed{\textbf{langage régulier}}$ s'il existe une expression régulière **e** telle que $\mathcal{L}(e) = L$.

Définition

Pour tout alphabet Σ, on note :

- $\boxed{\textbf{Rat}(\Sigma^*)}$ l'ensemble des langages rationnels ;
- $\boxed{\textbf{Rec}(\Sigma^*)}$ l'ensemble des langages reconnaissables par automate fini ;
- $\boxed{\textbf{Reg}(\Sigma^*)}$ l'ensemble des langages réguliers.

Théorème de Kleene

$$\mathrm{Rat}(\Sigma^*) = \mathrm{Reg}(\Sigma^*) = \mathrm{Rec}(\Sigma^*)$$

Équivalence des représentations

Preuve de $\text{Rat}(\Sigma^*) = \text{Reg}(\Sigma^*)$

Par définition des expression régulières.

Preuve de $\text{Rat}(\Sigma^*) \subseteq \text{Rec}(\Sigma^*)$

Par construction d'automates, preuve en TD.

Preuve de $\text{Rec}(\Sigma^*) \subseteq \text{Reg}(\Sigma^*)$

À partir d'un automate $\mathcal{A} = (\Sigma, Q, q_0, F, T)$, on va construire une expression régulière $\mathbf{e}$, telle que $L(\mathcal{A}) = \mathcal{L}(\mathbf{e})$.

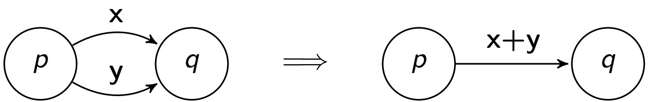
Initialisation

- modifier la syntaxe de l'automate en transformant chaque caractère sur les transitions en e.r. ;
- ajouter un état initial α et une transition $(\alpha, \varepsilon, q_0)$;
- ajouter un état final ω et une transition (q, ε, ω) pour tout état final $q \in F$;
- $F = \{\omega\}$.

Preuve de $\mathrm{Rec}(\Sigma^*) \subseteq \mathrm{Reg}(\Sigma^*)$

$\boxed{\textbf{Fusion}}$: Algo de suppression d'une transition

Soient **x** et **y** des e.r.,



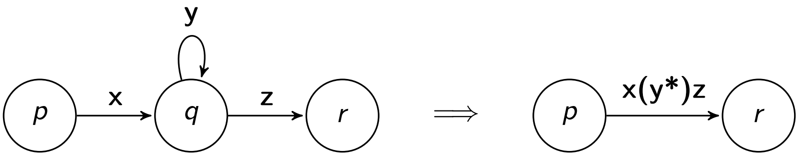
Cet algo supprime une transition de l'automate.

S'il y a plusieurs transitions entre les états p et q, on peut, bien sûr, les fusionner en une seule étape.

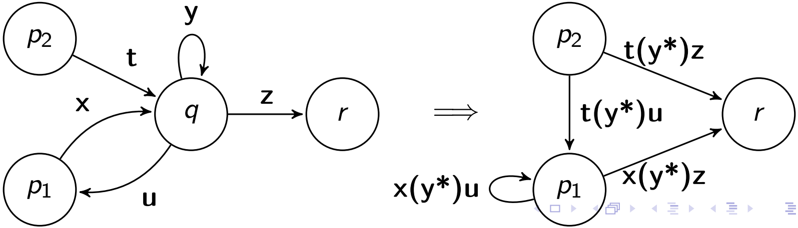
Preuve de $\mathrm{Rec}(\Sigma^*) \subseteq \mathrm{Reg}(\Sigma^*)$

Contraction : Algo de suppression d'état

Soient **x**, **y** et **z** des e.r.,



Cet algo supprime un état mais peut rajouter des transitions. Si q n'a pas de boucle et a n prédécesseurs et m successeurs, l'algo supprime $n + m$ transitions et en ajoute $n \times m$.



Preuve de $\mathrm{Rec}(\Sigma^*) \subseteq \mathrm{Reg}(\Sigma^*)$

Répéter les algos **Fusion** et **Contraction** tant qu'un des deux est applicable.

Comme **Contraction** supprime des états (en ajoutant éventuellement des transitions) et que **Fusion** supprime des transitions, à la fin il ne reste que deux états et une transition.

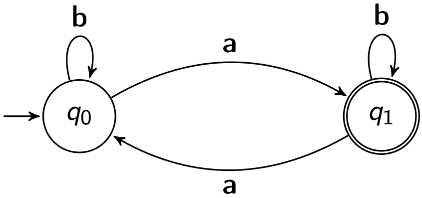
Fin Il ne reste plus que les états α et ω avec l'e.r **e** sur la transition entre α et ω : $(\alpha, \mathbf{e}, \omega)$.

Alors $\boxed{\mathcal{L}(\mathbf{e}) = L(\mathcal{A})}$.

Exemple de AFD $\rightarrow$ e.r.

$\boxed{\textbf{Initialisation}}$: AFD complet avec **a** et **b** qui sont des e.r.

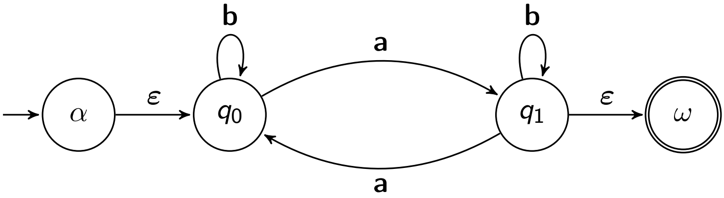
$\mathcal{A}$:



Exemple de AFD $\rightarrow$ e.r.

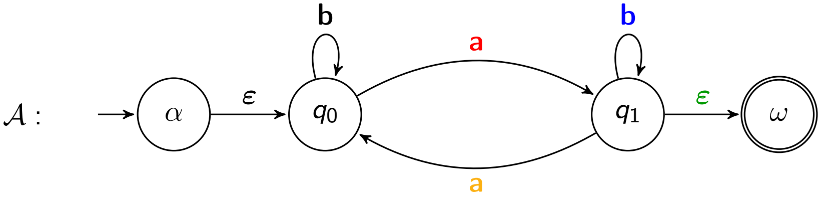
Initialisation : ajout de α et ω

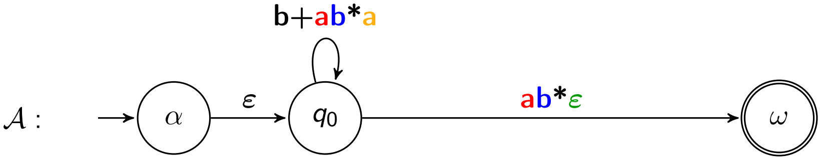
$\mathcal{A}$:



Exemple de AFD $\rightarrow$ e.r.

Contraction : Suppression de l'état q_1



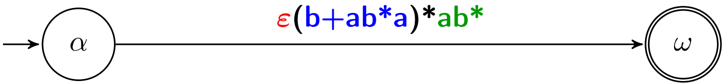


Exemple de AFD $\rightarrow$ e.r.

Contraction : Suppression de l'état q_0

$\mathcal{A}$:

$\mathcal{A}$:



$$\mathbf{e} = \mathbf{(b{+}ab^*a)^*ab^*} \quad \text{et} \quad L(\mathcal{A}) = \mathcal{L}(\mathbf{e}).$$