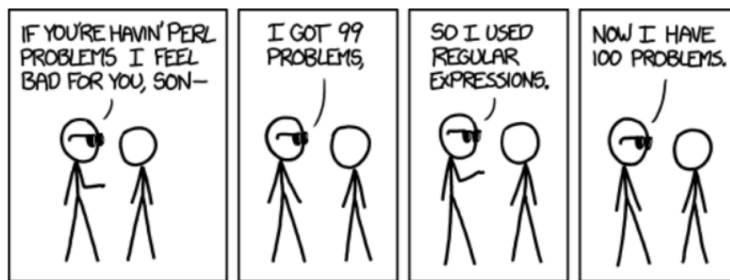


# Résumé du cours de THLR

Amnézic

2023



# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Définitions . . . . .	3
1.2	Opérations sur les mots : . . . . .	3
1.3	Distance d'un mot . . . . .	4
<b>2</b>	<b>Langages décidables</b>	<b>4</b>
2.1	Définition : . . . . .	4
2.2	Préfixes, facteurs et suffixes . . . . .	5
2.3	Étoile de Kleen : . . . . .	5
<b>3</b>	<b>Expressions régulières</b>	<b>6</b>
3.1	Les bases . . . . .	6
3.2	Définition . . . . .	6
<b>4</b>	<b>Automates</b>	<b>8</b>
4.1	Définition . . . . .	8
4.2	Propriétés d'un automate : . . . . .	8
<b>5</b>	<b>Automates et expressions régulières</b>	<b>9</b>
5.1	Complétude : . . . . .	9
5.2	Transitions spontanées : . . . . .	9
5.3	Algorithme de Thompson : . . . . .	9
<b>6</b>	<b>Simplification des <math>\varepsilon</math>-NFA</b>	<b>11</b>
<b>7</b>	<b>Pruning et déterminisme</b>	<b>12</b>
7.1	Pruning : . . . . .	12
7.2	Déterminisme . . . . .	13
<b>8</b>	<b>Lemme de pompage</b>	<b>14</b>
<b>9</b>	<b>Minimisation des automates</b>	<b>14</b>
<b>10</b>	<b>Propriétés des langages rationnels</b>	<b>16</b>
10.1	Algorithme de Brzozowski-McCluskey . . . . .	16
10.2	Stabilité des propriétés des langages rationnels . . . . .	16
<b>11</b>	<b>Précisions à partir des questions de QCM</b>	<b>16</b>

# 1 Introduction

Un langage est un ensemble de séquences d'objets élémentaires auxquels on associe un sens. Chaque langage dépend d'un ensemble limité de symboles élémentaires.

## 1.1 Définitions

### Alphabet:

Un alphabet est un ensemble  $\Sigma$  de symboles. On appelle les éléments de  $\Sigma$  des lettres. Un alphabet ne peut pas contenir plusieurs fois un même élément.

### Mot :

- Un mot  $w$  dans un alphabet  $\Sigma$  est une séquence finie (possiblement vide) de lettres. Le mot vide est noté  $\varepsilon$ .
- L'ensemble  $\Sigma^*$  est l'ensemble des tous les mots sur  $\Sigma$  et l'ensemble  $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$

### Langage :

- Un langage  $L$  sur un alphabet  $\Sigma$  est un ensemble de mots de  $\Sigma^*$
- $L$  peut être fini ou infini
- $L \subseteq \Sigma^* = L$  est un sous-ensemble de l'ensemble de tous les mots

## 1.2 Opérations sur les mots :

Pour ces calculs, il faut obligatoirement donné un alphabet de référence. Les opérations «mathématiques»:

- longueur d'un mot : la longueur d'un mot  $w$  sur un alphabet  $\Sigma$ , notée  $|w|$ , est égale au nombre de lettres ( $|\varepsilon| = 0$ )
- occurrences d'un mot : l'occurrence d'un mot  $w$  sur un alphabet  $\Sigma$  et une lettre  $a \in \Sigma$ ,  $|w|_a$  correspond au nombre d'occurrences de  $a$  dans  $w$
- concaténation : soient deux mots  $w_1 = a_1...a_n$  et  $w_2 = b_1...b_m$ , on définit leur concaténation  $w_1 \bullet w_2 = a_1...a_nb_1...b_m$
- propriétés de la concaténation :

- $|w_1 + w_2| = |w_1| + |w_2|$
- $\varepsilon \bullet w_1 = w_1 \bullet \varepsilon = w_1$
- $w_1 \bullet (w_2 \bullet w_3) = (w_1 \bullet w_2) \bullet w_3 \Rightarrow$  associative

– pas commutative

- exponentiation d'un mot :  $w^k = w \dots w$  k fois ( $w^0 = \varepsilon$ )

Les mots dérivés : (soient  $w, x, y, z$  d'un alphabet  $\Sigma$ )

- $x$  est un préfixe de  $w$  si  $w = x \bullet y$
- $z$  est un suffixe de  $w$  si  $w = y \bullet z$
- $y$  est un facteur de  $w$  si  $w = x \bullet y \bullet z$
- $\varepsilon$  appartient à  $\text{Pref}(w)$ ,  $\text{Suff}(w)$  et  $\text{Fact}(w)$
- inversion d'un mot : soit  $w = w_1 \dots w_n$  alors son miroir, noté  $w^R$ , est  $w_n \dots w_1$  (si  $w = w_R$ , alors  $w$  est un palindrome)

### 1.3 Distance d'un mot

**Définition :**

La distance d'édition  $d_e(w_1, w_2)$  entre deux mots  $w_1$  et  $w_2 \in \Sigma^*$  est égal au nombre minimal d'insertions et de suppressions lettre par lettre nécessaire pour passer de  $w_1$  à  $w_2$ .

Soit  $E$  un ensemble, une fonction  $d : E^2 \rightarrow R_+$  renvoie la distance si elle vérifie les propriétés suivantes ( $\forall x, y, z \in E$ ):

- séparation :  $d_e(x, y) = 0 \iff x = y$
- symétrie :  $d_e(x, y) = d_e(y, x)$
- inégalité triangulaire :  $d_e(x, y) + d_e(y, z) \geq d_e(x, z)$

## 2 Langages décidables

### 2.1 Définition :

Un langage  $L$  sur un alphabet  $\Sigma$  est dit décidable ou récursif s'il existe un algorithme  $A$  tel que, pour tout  $w \in \Sigma^*$

- if  $w \in L$ , alors  $A(w)$  renvoie true
- sinon,  $A(w)$  renvoie false

Un langage  $L$  sur un alphabet  $\Sigma$  est dit semi-décidable ou récursivement énumérable s'il existe un algorithme  $A$  tel que, pour tout  $w \in \Sigma^*$ .

- si  $w \in L$ , alors  $A(w)$  renvoie true
- si  $w \notin L$ , alors  $A(w)$  renvoie false **ou ne finit jamais**

Note : Si un langage  $L$  est récursif, alors il est récursivement énumérable (l'inverse est faux).

On peut voir les langages comme des ensembles en probabilités, il partage d'ailleurs certaines propriétés:

- $\overline{(\overline{A})} = A$
- $A \cup \overline{A} = \Sigma^*$
- $\overline{(A \cup B)} = \overline{A} \cap \overline{B}$
- $\overline{(A \cap B)} = \overline{A} \cup \overline{B}$
- $A \cap (B \cup C) = (A \cap B) \cap (A \cap C)$
- $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

## 2.2 Préfixes, facteurs et suffixes

Soit  $L$  un langage sur  $\Sigma$ , le préfixe d'un langage est l'ensemble de tous les mots qui sont les préfixes d'au moins un mot de  $L$ . De manière plus formelle on le définit comme ceci :

$$\text{Pref}(L) = \{ w \mid \exists x \in L, w \in \text{Pref}(x) \}$$

Note : cette définition s'applique aussi aux facteurs et aux suffixes

## 2.3 Étoile de Kleen :

Pour tout langage  $L$ , on a les propriétés suivantes :

- $\varepsilon \in L^*$ , même si  $\varepsilon \notin L$
- $\varepsilon \in L^+$  ssi  $\varepsilon \in L$
- $\emptyset^* = \{\varepsilon\}$
- $\Sigma^* = \Sigma^*$

## 3 Expressions régulières

### 3.1 Les bases

Les symboles des motifs de langages :

- $+$  : ou  $\rightarrow a + b = a$  ou  $b$
- "(" et ")" : but uniquement syntaxique
- $(x)^*$  : répétition finie (potentiellement 0 fois) de  $x$
- pour écrire un ensemble assez instinctif, on peut remplacer les ... par -  
 $\rightarrow \{A, \dots, Z\} = [A-Z]$
- $x^? = x + \varepsilon$  : la lettre  $x$  est optionnelle
- $x^+ = xx^*$  :  $x$  est présent au moins une fois

### 3.2 Définition

#### Syntaxe d'expression régulière

L'ensemble  $\text{Reg}_\Sigma$  sur un alphabet finie  $\Sigma$  qui ne contient pas les symboles réservés (ceux utilisés plus haut) est défini inductively par :

- atomes :  $\emptyset, \varepsilon$  et  $a$  ( $\forall a \in \Sigma$ ) sont des expressions régulières
- règles : si  $e_1$  et  $e_2$  sont des expressions régulières, alors  $e_1 + e_2$ ,  $e_1 e_2$  et  $e_1^*$  aussi
- profondeur : (pas compris)

Note: ces règles ne s'appliquent pas pour les intersections et les unions

- If  $e = \emptyset$ ,  $\mathcal{L}(e) = \emptyset$ .
- If  $e = \varepsilon$ ,  $\mathcal{L}(e) = \{\varepsilon\}$ .
- If  $e = a$  for  $a \in \Sigma$ ,  $\mathcal{L}(e) = \{a\}$ .
- If  $e = e_1 e_2$ ,  $\mathcal{L}(e) = \mathcal{L}(e_1) \cdot \mathcal{L}(e_2)$ .
- If  $e = e_1 + e_2$ ,  $\mathcal{L}(e) = \mathcal{L}(e_1) \cup \mathcal{L}(e_2)$ .
- If  $e = e_1^*$ ,  $\mathcal{L}(e) = \mathcal{L}(e_1)^*$ .

Figure 1: Sémantique d'expressions régulières

### Langages rationnels

L'ensemble  $Rat_{\Sigma}$  des langages définis sur un alphabet  $\Sigma$  est défini ainsi :

$$Rat = \{L \subseteq \Sigma^* | \exists e \in Reg_{\Sigma}, L = \mathcal{L}(e)\}$$

- Un langage rationnel est un langage décidable.
- Un même langage peut être décrit par une infinité d'expressions régulières.  
Si  $\mathcal{L}(e_1) = \mathcal{L}(e_2) \iff e_1 \equiv e_2$

$$\begin{aligned} e\emptyset &\equiv \emptyset e \equiv \emptyset \\ e\varepsilon &\equiv \varepsilon e \equiv e \\ e^{**} &\equiv e^* \equiv e^*e^* \\ \emptyset^* &\equiv \varepsilon \\ \varepsilon^* &\equiv \varepsilon \\ e + f &\equiv f + e \\ e + \emptyset &\equiv e \\ e(f + g) &\equiv ef + eg \\ (e + f)g &\equiv eg + fg \end{aligned}$$

Figure 2: Équivalences d'expressions régulières

## 4 Automates

### 4.1 Définition

Un automate fini  $A(Q, \Sigma, \delta, I, F)$  est composé de 5 éléments :

- un ensemble fini  $Q$  d'états : un état est un noeud du graphe
- un alphabet fini  $\Sigma$
- un ensemble de transitions  $\delta \subseteq Q \times \Sigma \times Q$  : une transition est un triplé, et  $\delta$  est un ensemble de triplés
- un ensemble d'états initiaux  $I$
- un ensemble d'états finaux/terminaux/acceptants  $F$

Un automate  $A$  accepte un mot  $w \in \Sigma^*$  s'il existe un chemin depuis un état initial jusqu'à un état final nommé par  $w$ .

Le langage  $L(A) = \{w \in \Sigma^* | A \text{ accepte } w\}$  d'un automate  $A$  est l'ensemble des tous les mots dans  $\Sigma^*$  accepté par  $A$ . On dit que  $\mathcal{L}(A)$  est reconnu par  $A$ .

### 4.2 Propriétés d'un automate :

- Deux automates  $A_1$  et  $A_2$  sont dit équivalents si  $\mathcal{L}(A_1) = \mathcal{L}(A_2)$ .
- Un automate  $A$  est dit déterministe (et appelé DFA, NFA si non déterministe) si
  - il a un unique état initial
  - pour chaque lettre  $a$  dans  $\Sigma$  et chaque état dans  $Q$  de  $A$ , il y a **au plus** une transition sortante nommée  $a$



## 5 Automates et expressions régulières

### 5.1 Complétude :

Définitions :

- complétude : un automate  $A$  est dit complet si  $\forall a \in \Sigma$  et  $\forall$  état  $q$  de  $A$ , il existe depuis  $q$  au moins une transition sortant nommé  $a$
- lemme : si un automate  $A$  est complet, alors pour tous les mots  $w \in \Sigma^*$  il y a **au moins** une transition nommé  $w$

### 5.2 Transitions spontanées :

Pour tout automate fini, on utilise une transition par lettre, mais on peut utiliser des  $\varepsilon$  (mot vide) transitions (aka transitions spontanées), ce qui permet de prendre des transitions sans lire de lettre.

Contrairement aux automates, les algorithmes qui utilisent des  $\varepsilon$ -transitions multiples à partir d'un même état, donc par définition non-déterministe. Ces automates sont appelés *automate fini non déterministe avec  $\varepsilon$ -transitions* ou en plus court  $\varepsilon$ -NFA.

### 5.3 Algorithme de Thompson :

On peut généraliser ces automates aux Regex :

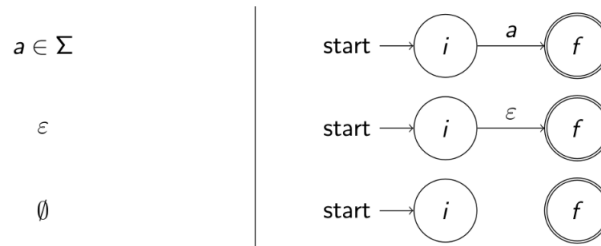
**Théorème de Kleene :**

Soit une expression régulière  $e \in \text{Reg}_\Sigma$ , il existe un  $\varepsilon$ -NFA  $A$  sur l'alphabet  $\Sigma$  tel que  $L(e) = L(A)$ .

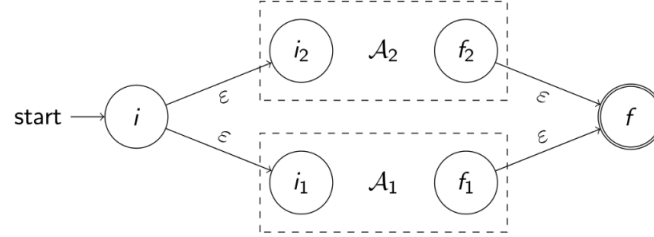
Les Regex sont définies "inductively".

A la manière d'un automate, on commence par créer un état initial  $i$  distinct d'un état final  $f$ . Voici les différents motifs pour la création d'un  $\varepsilon$ -NFA : (avec  $(a,b) \in \Sigma^2$  et  $(u_1, u_2) \in (\Sigma^*)^2$ )

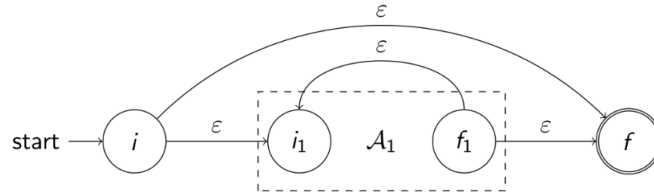
- début et fin de l'automate :



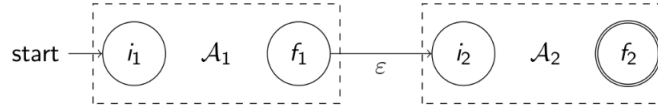
- $e_1 + e_2$  :



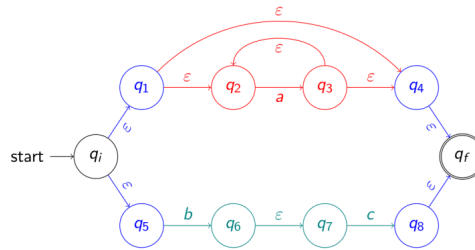
- $e_1^*$  :



- $e_1 \bullet e_2$  :



- par exemple, à partir de la Regex  $a^* + bc$  sur l'alphabet  $\Sigma = \{a,b,c\}$  on obtient l'algorithme de Thompson suivant :



- Soit  $e \in \text{Reg}_\Sigma$  une Regex et  $n$  le nombre d'occurrences de  $\{\varepsilon, \emptyset, +, *, \text{le nombre de lettres dans } \Sigma\}$  (on exclue les parenthèses). Le nombre d'états est alors égale à  $2n$ .

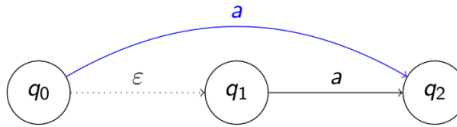
À partir de l'automate obtenu par l'algorithme de Thompson, on réduit à 0 les  $\varepsilon$ -transitions, en reliant les différents états nécessaires à l'état initial. Puis on modifie les derniers états en les modifiant par des états finaux.

## 6 Simplification des $\varepsilon$ -NFA

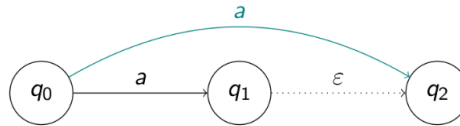
C'est la partie du cours la plus compliquée, il y aura un certain nombre de termes techniques. Une bonne connaissance du cours vu jusqu'ici est absolument nécessaire à la compréhension de cette partie.

En reprenant le dernier automate obtenu (le dernier automate de la page précédente), on remarque qu'il y a beaucoup d'états qu'un automate déterministe aurait pu éviter. On va alors procéder à des *éliminations arrière* des  $\varepsilon$ -transitions et des *éliminations avant* des  $\varepsilon$ -transitions

- élimination arrière des transitions spontanées :



- élimination avant des transitions spontanées :



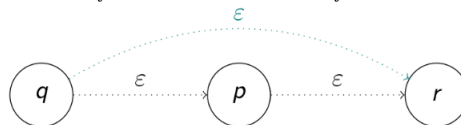
On définit les  $\varepsilon$ -fermeture de la manière suivante :

**$\varepsilon$ -fermeture d'un état :**

La  $\varepsilon$ -fermeture  $\varepsilon_{forward}^A(q)$  d'un état  $q$  d'un automate  $A$  est l'ensemble  $\{p \in Q \mid q \xrightarrow{\text{epsilon}}^*_A p\}$   
 Note :  $q \in \varepsilon_{forward}^A(q)$  car on peut atteindre  $q$  à partir de  $q$  en ne lisant rien du tout.

Théorème :

Si  $p \in \varepsilon_{forward}^A(q)$  and  $r \in \varepsilon_{forward}^A(p)$  alors  $r \in \varepsilon_{forward}^A(q)$ .



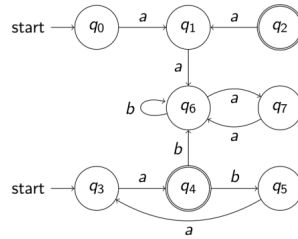


Figure 3: automate  $A$  sur l'alphabet  $\Sigma = \{a,b\}$

## 7 Pruning et déterminisme

### 7.1 Pruning :

Le NFA généré par l'algorithme de Thompson est compliqué, même avec les suppressions arrières des  $\varepsilon$ -transitions. On va donc supprimer les états et transitions qui ne servent à rien.

**Utilité des états :** (soit  $q$  un état de l'automate  $A$ )

- $q$  est dit *accessible* s'il peut être atteint à partir d'un état initial
- $q$  est dit *co-accessible* si à partir de  $q$ , on peut atteindre un état acceptant
- $q$  est dit *utile* s'il est accessible **et** co-accessible.

Si un état n'est ni accessible, ni co-accessible (donc par définition, pas utile), on dit qu'il est *inutile*.

Ces états permettent d'introduire la notion de *pruning* d'un automate.

Soient  $A$  un automate et  $A'$  l'automate obtenu à partir de la suppression des états inutiles de  $A$ . On dit que  $A$  est équivalent à  $A'$ .

Ce théorème permet de donner l'algorithme de pruning:

1. trouver les états accessibles (par un parcours profondeur ou largeur) depuis les états initiaux
2. trouver les états co-accessibles (on "remonte" les flèches à partir des états acceptants)
3. ne garder que les états accessibles et co-accessibles

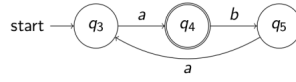


Figure 4: automate  $A'$  :  $A$  et  $A'$  sont équivalents

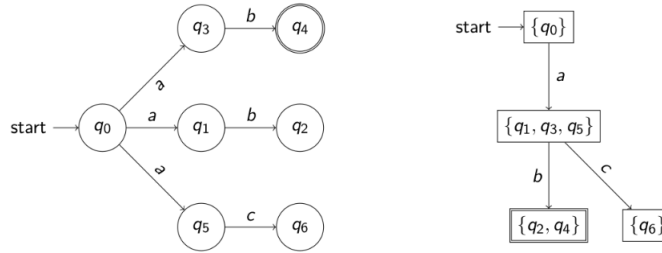


Figure 5: Un NFA et son équivalent DFA

Figure 5: Un NFA et son équivalent DFA

## 7.2 Déterminisme

Petit rappel :

Un NFA est dit déterministe si pour toute lettre  $a$  et état  $q$ , il existe au plus une transition sortant de  $q$  étiqueté par  $a$ .

Un NFA est dit complet si pour toute lettre  $a$  et état  $q$ , il existe au moins une transition sortant de  $q$  étiqueté par  $a$ .

Théorème

$\forall A$  NFA sur un alphabet  $\Sigma$ ,  $\exists$  un équivalent DFA  $A'$  sur  $\Sigma$

## 8 Lemme de pompage

Théorème :

Tous les langages rationnels sont décidables. La contraposée n'est pas toujours vraie.

Lemme de pompage :

Soit un langage rationnel  $L \in \text{Rat}_\Sigma$ , il existe un niveau de pompage  $n_0$  tel que pour tout mot  $w \in L$  de longueur  $n \geq n_0$ , il existe 3 mots  $x, y$  et  $z \in \Sigma^*$  tel que  $w = xyz$ ,  $y \neq \varepsilon$ , and  $L(xy^*z) \subseteq L$

Théorème :

Le langage  $L = \{a^n b^n \mid n \in \mathbf{N}\}$ , sur l'alphabet  $\Sigma = \{a, b\}$ , est décidable mais non rationnel.

Un langage rationnel est reconnaissable par un algorithme qui utilise une quantité constante de données qui ne dépend pas de la taille de l'input.

## 9 Minimisation des automates

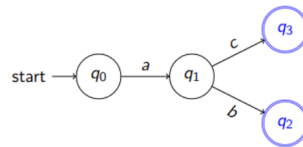
Le déterminisme d'un automate d'une regex crée un nombre très élevé d'états. Dans l'ordre on a :

1. expression régulière :  $n$  symboles
2.  $\varepsilon$ -NFA :  $2n$  états
3. NFA :  $2n$  états
4. pruned NFA :  $\leq 2n$  états
5. DFA :  $\leq 2^{2n}$

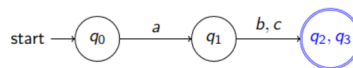
États indistinguishables:

Deux états  $q_1$  et  $q_2$  d'un DFA  $A$  sont dits **indistinguishables** si pour tout mot  $w \in \Sigma^*$ , de  $q_1$ ,  $A$  accepte  $w$  ssi, depuis  $q_2$ ,  $A$  accepte  $w$ . On dit que  $L_{q_1}(A) = L_{q_2}(A)$ .

À partir de l'automate suivant : on peut donner l'automate



équivalent suivant :



Théorème de Myhill-Nerode :

Soit un langage rationnel  $L$ , il existe un unique DFA  $A$  avec un nombre minimal d'états (= au nombre d'états indistinguables de  $L$ ) tel que  $L(A) = L$ . On dit que cet automate est l'automate canonique de  $L$ .

Ainsi, si deux états  $p_1$  et  $p_2$  d'un DFA  $A$  sont indistinguables et  $p_1 \rightarrow (a) q_1$  et  $p_2 \rightarrow (a) q_2$ , alors  $q_1$  et  $q_2$  sont eux aussi indistinguables.  
(section à compléter, pas compris la fin du ppt)

## 10 Propriétés des langages rationnels

### 10.1 Algorithme de Brzozowski-McCluskey

L'algorithme de Brzozowski-McCluskey consiste à transformer un automate avec plusieurs transitions à un automate avec un seul état initial, un seul état final et une seule transition entre les deux états. Cette transition doit contenir l'expression régulière qui doit être égale au langage initial. L'état initial ne doit pas avoir de transitions entrantes et l'état final ne doit pas avoir de transitions sortantes. Au final, l'automate doit avoir exactement cette structure :

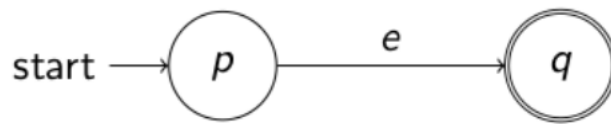


Figure 6:  $e$  doit être équivalent au langage initial

Cet algorithme a pour conséquence le théorème suivant :

Soit un automate  $A$  sur un alphabet  $\Sigma$ , il existe une expression régulière  $e \in \text{Reg}_\Sigma$  tel que  $L(A) = L(e)$ , c'est-à-dire que  $e$  est **équivalent** à  $A$ .

### 10.2 Stabilité des propriétés des langages rationnels

On considère un langage  $L$  rationnel ( $L \in \text{Rat}_\Sigma$ ):

- $\bar{L} \in \text{Rat}_\Sigma$ .
- $\text{Pref}(L)$ ,  $\text{Fact}(L)$  et  $\text{Suff}(L) \in \text{Rat}_\Sigma$
- $L^R$  (langage miroir)  $\in \text{Rat}_\Sigma$
- Si  $L_1, L_2 \in \text{Rat}_\Sigma$ , alors  $L_1 \cap L_2 \in \text{Rat}_\Sigma$
- on peut savoir si  $L = \emptyset$  ou non (si on peut accéder à un état acceptant depuis l'état initial)
- soit  $L' \in \text{Rat}_\Sigma$ , on peut savoir si  $L = L'$  ou non

## 11 Précisions à partir des questions de QCM

- Un langage peut être infini mais un alphabet est nécessairement fini.
- Un langage récursif n'est pas défini par récurrence.



- Le facteur d'un langage peut être le préfixe du suffixe et le suffixe du préfixe d'un langage.
- L'étoile de Kleene d'un langage fini peut être infini.
- Les langages rationnels sont stables par étoile de Kleene, concaténation et union.
- Tout langage rationnel vérifie le lemme de pompage mais tout langage qui vérifie le lemme de pompage n'est pas rationnel.
- L'inclusion ne préserve pas la rationalité.
- Il n'existe pas de mot à longueur infini.
- Soit  $L$  un langage ne contenant pas  $\varepsilon$ .  $\varepsilon$  appartient à  $L^*$  mais pas à  $L^+$ .
- Tout langage rationnel est décidable mais tout langage décidable n'est pas rationnel.
- $\varepsilon$  n'appartient à aucun alphabet, puisqu'il représente l'absence de lettres.
- L'union de 2 langages correspond au "+" dans l'arithmétique des regex.