

Théorie des langages : THL

CM 2

Uli Fahrenberg

EPITA Rennes

S5 2023

Aperçu

Programme du cours

- ① Langages rationnels
- ② **Automates finis**
- ③ Langages algébriques, grammaires hors-contexte
- ④ Automates à pile
- ⑤ Parsage LL
- ⑥ Parsage LR
- ⑦ flex & bison

Prochainement

Une simple grammaire :

$\text{Var} ::= [\text{a-zA-Z}] [\text{a-zA-Z0-9}_*]$

$\text{Num} ::= -? [1-9] [0-9]^*$

$\text{Aexp} ::= \text{Num} \mid \text{Var} \mid \text{Aexp} + \text{Aexp} \mid \text{Aexp} - \text{Aexp} \mid \text{Aexp} * \text{Aexp}$

$\text{Bexp} ::= \text{True} \mid \text{False} \mid \text{Aexp} == \text{Aexp} \mid \text{Aexp} < \text{Aexp}$

$\mid \neg \text{Bexp} \mid \text{Bexp} \wedge \text{Bexp} \mid \text{Bexp} \vee \text{Bexp}$

$\text{Stmt} ::= \text{Var} = \text{Aexp} \mid \text{Stmt} ; \text{Stmt} \mid \text{while Bexp Stmt}$

$\mid \text{if Bexp then Stmt else Stmt}$

Dernièrement : mots

Soit Σ un ensemble **fini**.

- on appelle les éléments $a, b, \dots \in \Sigma$ des **symboles**

On dénote Σ^* l'ensemble de tous les **suites finies** d'éléments de Σ .

- donc $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots = \bigcup_{n \geq 0} \Sigma^n$
- on appelle les éléments $u, v, w, \dots \in \Sigma^*$ des **mots**

La **concaténation** de deux mots $a_1 \dots a_n$ et $b_1 \dots b_m$ est le mot

$$a_1 \dots a_n b_1 \dots b_m.$$

- ε : le mot vide
- l'opération « . » sur mots est **associative** et a ε comme **élément neutre de deux côtés**

La **longueur** $|u|$ d'un mot $u \in \Sigma^*$: le nombre de symboles de u .

- $|\varepsilon| = 0$ et $|uv| = |u| + |v|$
- u^n : la concaténation de n copies de u
- $|u^n| = n|u|$

Dernièrement : langages

Un **langage** est un sous-ensemble $L \subseteq \Sigma^*$.

- opérations ensemblistes : $L_1 \cup L_2$, $L_1 \cap L_2$, \bar{L}
- concaténation : $L_1 L_2 = \{u_1 u_2 \mid u_1 \in L_1, u_2 \in L_2\}$
- $L^n = L \cdots L$ (n copies de L)
- étoile de Kleene : $L^* = L^0 \cup L_1 \cup L^2 \cup \cdots = \bigcup_{n \geq 0} L^n$

L'opération « . » sur langages est **associative** et a $\{\varepsilon\}$ comme
élément neutre de deux côtés.

- $L.\emptyset = \emptyset.L = \emptyset$
- $\emptyset^* = \{\varepsilon\}^* = \{\varepsilon\}$

Dernièrement : langages rationnels

Les **expressions rationnelles** sur Σ :

- ① \emptyset et ε sont des expressions rationnelles
- ② pour tout $a \in \Sigma$, a est une expression rationnelle
- ③ e_1 et e_2 expressions rationnelles $\Rightarrow e_1 + e_2$, $e_1.e_2$ et e_1^* aussi

Le **langage dénoté** par une expression rationnelle e sur Σ :

- ① $L(\emptyset) = \emptyset$, $L(\varepsilon) = \{\varepsilon\}$
- ② $L(a) = \{a\}$ pour tout $a \in \Sigma$
- ③ $L(e_1 + e_2) = L(e_1) \cup L(e_2)$, $L(e_1.e_2) = L(e_1).L(e_2)$, $L(e^*) = (L(e))^*$

Les **langages rationnels** sur Σ :

- ① \emptyset et $\{\varepsilon\}$ sont des langages rationnels
- ② pour tout $a \in \Sigma$, $\{a\}$ est un langage rationnel
- ③ L_1 et L_2 langages rationnels $\Rightarrow L_1 \cup L_2$, $L_1.L_2$ et L_1^* aussi

Théorème : $L \subseteq \Sigma^*$ est rationnel ssi il existe une expression rationnelle e telle que $L = L(e)$.

Dans le poly

La dernière fois :

- chapitre 2, moins 2.3.2-5 et 2.4.4
- chapitre 3, moins 3.1.3
- plus démonstration que L rationnel \Rightarrow Pref(L) rationnel

Aujourd'hui :

- chapitre 4, moins 4.1.3, 4.2.1, 4.3, 4.4

Correction (partielle) devoir maison

Sur alphabet $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -, ., E\}$, donnez des expressions rationnelles pour les langages suivants :

- ① les entiers positifs en base 10

$$(1+2+3+4+5+6+7+8+9)(0+1+2+3+4+5+6+7+8+9)^* \\ [1-9] [0-9]^*$$

- ② les entiers relatifs en base 10

$$-? [1-9] [0-9]^*$$

- ③ les nombres décimaux positifs en base 10

$$[0-9]^* . [0-9]^*$$

- ④ les nombres décimaux relatifs en base 10

$$-? [0-9]^* . [0-9]^*$$

- ⑤ les nombres décimaux relatifs en base 10 en notation scientifique.

$$-? [0-9]^* . [0-9]^* E [0-9]^*$$

5 minutes de réflexion

Vrai ou faux ?

- ① Si L_1 et L_2 sont rationnels, alors $L_1 \cup L_2$ est rationnel
- ② Si L_1 et L_2 sont rationnels, alors $L_1 \cap L_2$ est rationnel
- ③ Chaque sous-ensemble d'un langage rationnel L est rationnel.

Pour chaque expression rationnelle suivante, trouvez deux mots qui appartiennent de leur langage et deux autres qui ne l'appartiennent pas :

- ④ a^*b^*
- ⑤ $a^* + b^*$
- ⑥ $(aaa)^*$
- ⑦ $(a + b)^*ab(a + b)^*ba(a + b)^*$

5 minutes de réflexion

Vrai ou faux ?

- ① Si L_1 et L_2 sont rationnels, alors $L_1 \cup L_2$ est rationnel ✓
- ② Si L_1 et L_2 sont rationnels, alors $L_1 \cap L_2$ est rationnel ✓
- ③ Chaque sous-ensemble d'un langage rationnel L est rationnel. ✗

Pour chaque expression rationnelle suivante, trouvez deux mots qui appartiennent de leur langage et deux autres qui ne l'appartiennent pas :

- ④ a^*b^*
- ⑤ $a^* + b^*$
- ⑥ $(aaa)^*$
- ⑦ $(a + b)^*ab(a + b)^*ba(a + b)^*$

Automates finis déterministes

Exemple

L'algorithme le plus simple qui décide le langage de tous les mots qui

commencent par *ab* : $L = \{ab, aba, abb, abaa, abab, abba, \dots\}$

(en Python-èskue) :

```
def startsab(stream):
    state = 0
    while x = next(stream):
        if state == 0:
            if x == "a":
                state = 1
            else: return False
        elif state == 1:
            if x == "b":
                state = 2
            else: return False
    if state == 2: return True
    else: return False
```

Exemple

L'algorithme le plus simple qui décide le langage de tous les mots qui

commencent par *ab* : $L = \{ab, aba, abb, abaa, abab, abba, \dots\}$

(en Python-èské) :

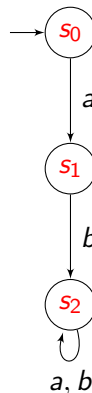
```
def startsab(stream):
    state = 0
    while x = next(stream):
        if state == 0:
            if x == "a":
                state = 1
            else: return False
        elif state == 1:
            if x == "b":
                state = 2
            else: return False
    if state == 2: return True
    else: return False
```



Exemple

L'algorithme le plus simple qui décide le langage de tous les mots qui **commencent par ab** : $L = \{ab, aba, abb, abaa, abab, abba, \dots\}$
 (en Python-èskue) :

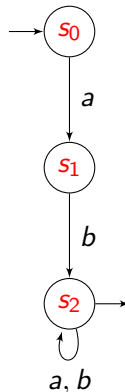
```
def startsab(stream):
    state = 0
    while x = next(stream):
        if state == 0:
            if x == "a":
                state = 1
            else: return False
        elif state == 1:
            if x == "b":
                state = 2
            else: return False
        if state == 2: return True
    else: return False
```



Exemple

L'algorithme le plus simple qui décide le langage de tous les mots qui **commencent par ab** : $L = \{ab, aba, abb, abaa, abab, abba, \dots\}$
(en Python-èské) :

```
def startsab(stream):
    state = 0
    while x = next(stream):
        if state == 0:
            if x == "a":
                state = 1
            else: return False
        elif state == 1:
            if x == "b":
                state = 2
            else: return False
        if state == 2: return True
    else: return False
```



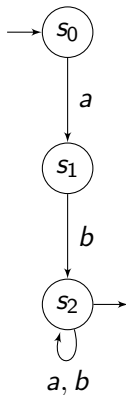
Automates finis déterministes complets

Définition (4.1)

Un **automate fini déterministe complet** est une structure $(\Sigma, Q, q_0, F, \delta)$ où

- Σ est un ensemble fini de **symboles**,
 - Q est un ensemble fini d'**états**,
 - $q_0 \in Q$ est l'**état initial**,
 - $F \subseteq Q$ est l'ensemble des **états finaux**, et
 - $\delta : Q \times \Sigma \rightarrow Q$ est la **fonction de transition**.
- un graphe orienté avec arcs étiquetés dans Σ et certains nœuds distingués comme initial et/ou final

Exemple



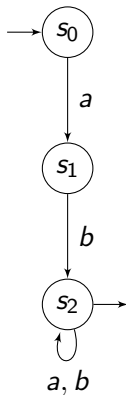
$$\Sigma = \{a, b\}$$

$$Q = \{s_0, s_1, s_2\}$$

$$q_0 = s_0$$

$$F = \{s_2\}$$

Exemple



$$\Sigma = \{a, b\}$$

$$Q = \{s_0, s_1, s_2\}$$

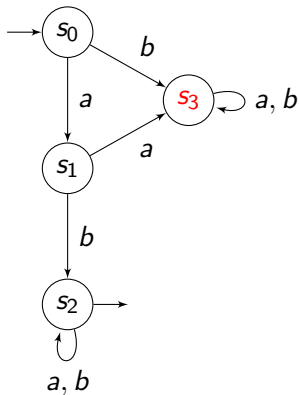
$$q_0 = s_0$$

$$F = \{s_2\}$$

$$\delta :$$

	<i>a</i>	<i>b</i>
<i>s</i> ₀	<i>s</i> ₁	
<i>s</i> ₁		<i>s</i> ₂
<i>s</i> ₂	<i>s</i> ₂	<i>s</i> ₂

Exemple



$$\Sigma = \{a, b\}$$

$$Q = \{s_0, s_1, s_2, s_3\}$$

$$q_0 = s_0$$

$$F = \{s_2\}$$

$\delta :$

	a	b
s_0	s_1	s_3
s_1	s_3	s_2
s_2	s_2	s_2
s_3	s_3	s_3

Comment ça marche

Un automate fini déterministe complet : $A = (\Sigma, Q, q_0, F, \delta)$:

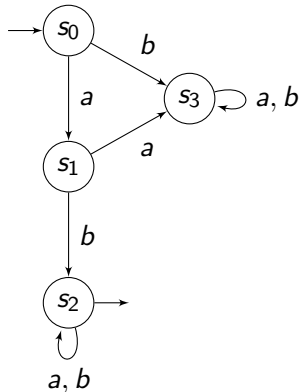
- Σ, Q ensembles finis, $q_0 \in Q, F \subseteq Q$,
- $\delta : Q \times \Sigma \rightarrow Q$: la fonction de transition

On note $q \xrightarrow{a} r$ pour $\delta(q, a) = r$.

Définition

- Un **calcul** dans A est une séquence $\sigma = q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \dots \xrightarrow{a_{n-1}} q_n$.
 - donc $\delta(q_i, a_i) = q_{i+1}$ pour tout $i = 1, \dots, n-1$
- L'**étiquette** d'un calcul comme ci-dessus est
$$\lambda(\sigma) = a_1 a_2 \dots a_{n-1} \in \Sigma^*.$$
- Un calcul comme ci-dessus est **réussi** si $q_1 = q_0$ et $q_n \in F$.
- Le **langage reconnu** par A est
$$L(A) = \{\lambda(\sigma) \mid \sigma \text{ calcul réussi dans } A\}.$$

Exemple



calculs dans A :

- $s_0 \xrightarrow{b} s_3 \xrightarrow{x_1} \dots \xrightarrow{x_n} s_3$
- $s_0 \xrightarrow{a} s_1 \xrightarrow{a} s_3 \xrightarrow{x_1} \dots \xrightarrow{x_n} s_3$
- $s_0 \xrightarrow{a} s_1 \xrightarrow{b} s_2 \xrightarrow{x_1} \dots \xrightarrow{x_n} s_2$

pour tous $x_1, \dots, x_n \in \{a, b\}$

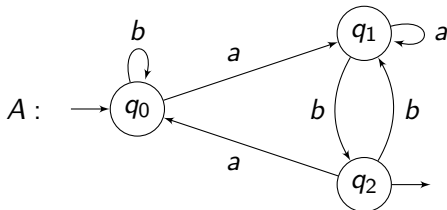
calculs réussis :

- $s_0 \xrightarrow{a} s_1 \xrightarrow{b} s_2 \xrightarrow{x_1} \dots \xrightarrow{x_n} s_2$

langage reconnu par A :

- $L(A) = L(ab(a + b)^*)$

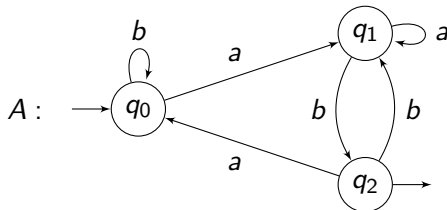
5 minutes de réflexion



Vrai ou faux ?

- ① $baba \in L(A)$
- ② $baab \in L(A)$
- ③ $abab \in L(A)$
- ④ $abaaab \in L(A)$
- ⑤ $\varepsilon \in L(A)$
- ⑥ $L(b^*aa^*b) \subseteq L(A)$

5 minutes de réflexion



Vrai ou faux ?

- ① $baba \in L(A)$
- ② $baab \in L(A)$
- ③ $abab \in L(A)$
- ④ $abaaab \in L(A)$
- ⑤ $\varepsilon \in L(A)$
- ⑥ $L(b^*aa^*b) \subseteq L(A)$

✗

✓

✗

✓

✗

✓

« Déterministe complet » ?

Automate fini déterministe complet : $(\Sigma, Q, q_0, F, \delta)$:

- Σ, Q ensembles finis, $q_0 \in Q, F \subseteq Q$,
- $\delta : Q \times \Sigma \rightarrow Q$: la fonction de transition
- très utile dans la théorie

Automate fini déterministe :

- δ fonction **partielle**
- très utile pour l'**implémentation**

Automate fini **non-déterministe** :

- δ **relation**
- très utile dans la théorie

Automate fini non-déterministe **avec transitions spontanées** :

- notion encore plus générale et utile (en théorie)

Automates finis déterministes

Définition (4.4)

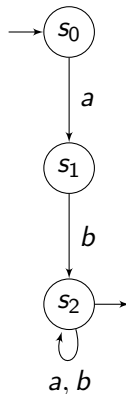
Un **automate fini déterministe** est une structure $(\Sigma, Q, q_0, F, \delta)$ où

- Σ est un ensemble fini de symboles,
 - Q est un ensemble fini d'états,
 - $q_0 \in Q$ est l'état initial,
 - $F \subseteq Q$ est l'ensemble des états finaux, et
 - $\delta : Q \times \Sigma \rightarrow Q$ est la fonction **partielle** de transition.
- tout automate fini déterministe peut être **complété** en ajoutant un **état puits** :

Exemple

Automate fini déterministe et complétion :

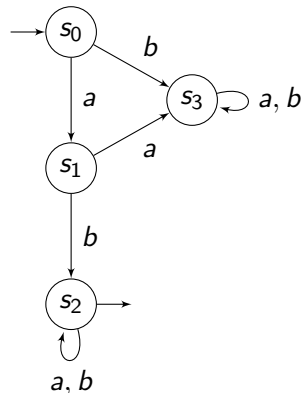
```
def startsab(stream):  
    state = 0  
    while x = next(stream):  
        if state == 0:  
            if x == "a":  
                state = 1  
            else: return False  
        elif state == 1:  
            if x == "b":  
                state = 2  
            else: return False  
    if state == 2: return True  
    else: return False
```



Exemple

Automate fini déterministe et complétion :

```
def startsab(stream):
    state = 0
    while x = next(stream):
        if state == 0:
            if x == "a":
                state = 1
            else: return False
        elif state == 1:
            if x == "b":
                state = 2
            else: return False
        if state == 2: return True
    else: return False
```



Complétion

Lemme

Pour tout automate fini déterministe A il existe un automate fini déterministe **complet** A' tel que $L(A') = L(A)$.

Démonstration.

- ① Soit $A = (\Sigma, Q, q_0, F, \delta)$.
- ② On construit $A' = (\Sigma, Q', q'_0, F', \delta')$ comme suit :
- ③ $Q' = Q \cup \{q_p\}$ où $q_p \notin Q$,
- ④ $q'_0 = q_0$ et $F' = F$.
- ⑤ La fonction $\delta' : Q' \times \Sigma \rightarrow Q'$ est définie par

$$\delta'(q, a) =$$

Complétion

Lemme

Pour tout automate fini déterministe A il existe un automate fini déterministe **complet** A' tel que $L(A') = L(A)$.

Démonstration.

- ① Soit $A = (\Sigma, Q, q_0, F, \delta)$.
- ② On construit $A' = (\Sigma, Q', q'_0, F', \delta')$ comme suit :
- ③ $Q' = Q \cup \{q_p\}$ où $q_p \notin Q$,
- ④ $q'_0 = q_0$ et $F' = F$.
- ⑤ La fonction $\delta' : Q' \times \Sigma \rightarrow Q'$ est définie par

$$\delta'(q, a) = \begin{cases} \delta(q, a) & \text{si } q \in Q \text{ et } \delta(q, a) \text{ est défini,} \\ q_p & \text{sinon,} \end{cases}$$

Complétion

Lemme

Pour tout automate fini déterministe A il existe un automate fini déterministe **complet** A' tel que $L(A') = L(A)$.

Démonstration.

- ① Soit $A = (\Sigma, Q, q_0, F, \delta)$.
- ② On construit $A' = (\Sigma, Q', q'_0, F', \delta')$ comme suit :
- ③ $Q' = Q \cup \{q_p\}$ où $q_p \notin Q$,
- ④ $q'_0 = q_0$ et $F' = F$.
- ⑤ La fonction $\delta' : Q' \times \Sigma \rightarrow Q'$ est définie par

$$\delta'(q, a) = \begin{cases} \delta(q, a) & \text{si } q \in Q \text{ et } \delta(q, a) \text{ est défini,} \\ q_p & \text{sinon.} \end{cases}$$

Complétion

Lemme

Pour tout automate fini déterministe A il existe un automate fini déterministe **complet** A' tel que $L(A') = L(A)$.

Démonstration.

- ① Soit $A = (\Sigma, Q, q_0, F, \delta)$.
- ② On construit $A' = (\Sigma, Q', q'_0, F', \delta')$ comme suit :
- ③ $Q' = Q \cup \{q_p\}$ où $q_p \notin Q$,
- ④ $q'_0 = q_0$ et $F' = F$.
- ⑤ La fonction $\delta' : Q' \times \Sigma \rightarrow Q'$ est définie par

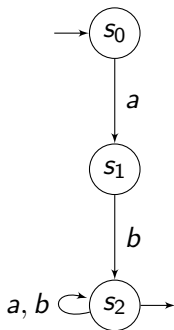
$$\delta'(q, a) = \begin{cases} \delta(q, a) & \text{si } q \in Q \text{ et } \delta(q, a) \text{ est défini,} \\ q_p & \text{sinon.} \end{cases}$$

- ⑥ Maintenant il faut démontrer que, en fait, $L(A') = L(A)$.

Non-déterminisme

Exemple

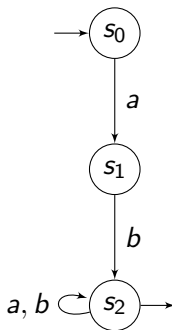
L'algorithme le plus simple qui décide le langage de tous les mots qui **commencent par ab** :



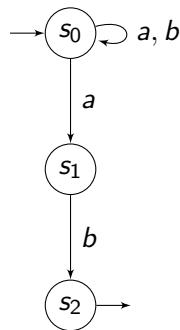
L'algorithme le plus simple qui décide le langage de tous les mots qui **se terminent par ab** :

Exemple

L'algorithme le plus simple qui décide le langage de tous les mots qui **commencent par ab** :

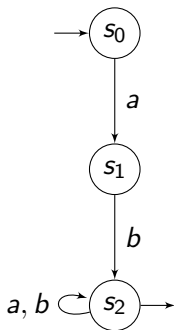


L'algorithme le plus simple qui décide le langage de tous les mots qui **se terminent par ab** :

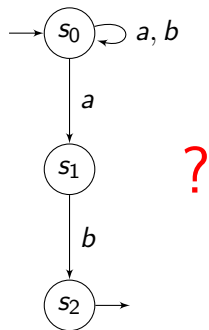


Exemple

L'algorithme le plus simple qui décide le langage de tous les mots qui **commencent par ab** :



L'algorithme le plus simple qui décide le langage de tous les mots qui **se terminent par ab** :



- pas un algorithme !
- $abab$???

Automates finis (non-déterministes)

Définition (4.8)

Un **automate fini** est une structure $(\Sigma, Q, q_0, F, \delta)$ où

- Σ est un ensemble fini de symboles,
- Q est un ensemble fini d'états,
- $q_0 \in Q$ est l'état initial,
- $F \subseteq Q$ est l'ensemble des états finaux, et
- $\delta \subseteq Q \times \Sigma \times Q$ est la **relation** de transition.

Automates finis (non-déterministes)

Définition (4.8)

Un **automate fini** est une structure $(\Sigma, Q, Q_0, F, \delta)$ où

- Σ est un ensemble fini de symboles,
- Q est un ensemble fini d'états,
- $Q_0 \subseteq Q$ est l'**ensemble des états initiaux**,
- $F \subseteq Q$ est l'ensemble des états finaux, et
- $\delta \subseteq Q \times \Sigma \times Q$ est la **relation** de transition.

- pas trop pratique pour l'implémentation
- mais bien utile en théorie !

Comment ça marche

Un automate fini : $A = (\Sigma, Q, Q_0, F, \delta)$:

- Σ, Q ensembles finis, $Q_0, F \subseteq Q$,
- $\delta \subseteq Q \times \Sigma \times Q$: la relation de transition

On note $q \xrightarrow{a} r$ si $(q, a, r) \in \delta$.

Définition

- Un **calcul** dans A est une séquence $\sigma = q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \dots \xrightarrow{a_{n-1}} q_n$.
- L'**étiquette** d'un calcul comme ci-dessus est
$$\lambda(\sigma) = a_1 a_2 \dots a_{n-1} \in \Sigma^*.$$
- Un calcul comme ci-dessus est **réussi** si $q_1 \in Q_0$ et $q_n \in F$.
- Le **langage reconnu** par A est
$$L(A) = \{\lambda(\sigma) \mid \sigma \text{ calcul réussi dans } A\}.$$

Comment ça marche

Un automate fini : $A = (\Sigma, Q, Q_0, F, \delta)$:

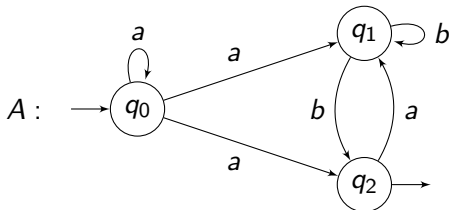
- Σ, Q ensembles finis, $Q_0, F \subseteq Q$,
- $\delta \subseteq Q \times \Sigma \times Q$: la relation de transition

On note $q \xrightarrow{a} r$ si $(q, a, r) \in \delta$. \Leftarrow la seule chose qui a changé !

Définition

- Un calcul dans A est une séquence $\sigma = q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \dots \xrightarrow{a_{n-1}} q_n$.
- L'étiquette d'un calcul comme ci-dessus est $\lambda(\sigma) = a_1 a_2 \dots a_{n-1} \in \Sigma^*$.
- Un calcul comme ci-dessus est réussi si $q_1 \in Q_0$ et $q_n \in F$.
- Le langage reconnu par A est $L(A) = \{\lambda(\sigma) \mid \sigma \text{ calcul réussi dans } A\}$.

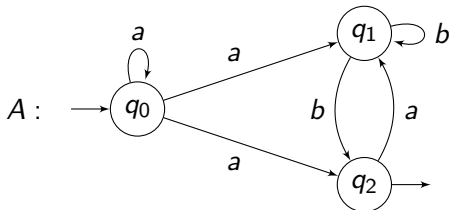
5 minutes de réflexion



Vrai ou faux ?

- ① $baba \in L(A)$
- ② $abab \in L(A)$
- ③ $aaab \in L(A)$
- ④ $aaaa \in L(A)$
- ⑤ $\varepsilon \in L(A)$
- ⑥ $L(a^*ab^*b) \subseteq L(A)$

5 minutes de réflexion



Vrai ou faux ?

① $baba \in L(A)$

✗

② $abab \in L(A)$

✓

③ $aaab \in L(A)$

✓

④ $aaaa \in L(A)$

✓

⑤ $\varepsilon \in L(A)$

✗

⑥ $L(a^*ab^*b) \subseteq L(A)$

✓

Langages reconnaissables

Définition

Un langage $L \subseteq \Sigma^*$ est **reconnaissable** si il existe un automate fini A tel que $L = L(A)$.

Théorème

Un langage $L \subseteq \Sigma^$ est reconnaissable ssi il existe un automate fini*

- *déterministe,*
- *déterministe complet, ou*
- *(non-déterministe) à **transitions spontanées***

A tel que $L = L(A)$.

- donc **sémantiquement** c'est tout là même chose : automates finis non-déterministes, automates finis déterministes, automates finis déterministes complets

Automates finis aux transitions spontanées

Définition (4.11)

Un **automate fini à transitions spontanées** est une structure $(\Sigma, Q, Q_0, F, \delta)$ où

- Σ est un ensemble fini de symboles,
 - Q est un ensemble fini d'états,
 - $Q_0 \subseteq Q$ est l'ensemble des états initiaux,
 - $F \subseteq Q$ est l'ensemble des états finaux, et
 - $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ est la relation de transition.
- peut changer de l'état spontanément sans lire un symbole

Comment ça marche

Un automate fini à transitions spontanées : $A = (\Sigma, Q, Q_0, F, \delta)$:

- Σ, Q ensembles finis, $Q_0, F \subseteq Q$,
- $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$: la relation de transition

On note $q \xrightarrow{a} r$ si $(q, a, r) \in \delta$. \Leftarrow donc a peut être ε

Définition

- Un **calcul** dans A est une séquence $\sigma = q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \dots \xrightarrow{a_{n-1}} q_n$.
- L'**étiquette** d'un calcul comme ci-dessus est $\lambda(\sigma) = a_1 a_2 \dots a_{n-1} \in \Sigma^*$.
- Un calcul comme ci-dessus est **réussi** si $q_1 \in Q_0$ et $q_n \in F$.
- Le **langage reconnu** par A est $L(A) = \{\lambda(\sigma) \mid \sigma \text{ calcul réussi dans } A\}$.

- note $a \varepsilon b \varepsilon a \varepsilon b = abab$, par exemple

Théorème de Kleene

Théorème (Kleene)

Un langage $L \subseteq \Sigma^*$ est *rationnel* ssi il est *reconnaissable*.

syntaxe

aut. finis dét. complets

\cap

aut. finis déterministes

\cap

automates finis

\cap

aut. finis à trans. spontanées

expressions rationnelles

$L(\cdot)$
 \longrightarrow

sémantique

langages reconnaissables

\parallel ✓

langages reconnaissables

\parallel ?

langages reconnaissables

\parallel ?

langages reconnaissables

\parallel ?

langages rationnelles

Fin à la spontanéité

Lemme

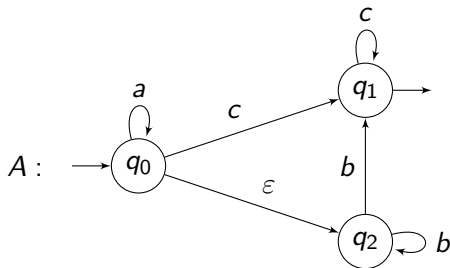
Pour tout automate fini à transitions spontanées A il existe un automate fini A' tel que $L(A') = L(A)$.

- on note $q \xrightarrow{\varepsilon}^* r$ si il existe une suite $q \xrightarrow{\varepsilon} \dots \xrightarrow{\varepsilon} r$ de transitions spontanées

Démonstration.

- 1 Soit $A = (\Sigma, Q, Q_0, F, \delta)$.
- 2 On construit $A' = (\Sigma, Q', Q'_0, F', \delta')$ comme suit :
- 3 $Q' = Q, Q'_0 = Q_0,$
- 4 $F' = \{q \in Q \mid \exists r \in F : q \xrightarrow{\varepsilon}^* r\},$ et
- 5 $\delta' = \{(p, a, r) \mid \exists q \in Q : p \xrightarrow{\varepsilon}^* q \text{ et } (q, a, r) \in \delta\}.$
- 6 Maintenant il faut démontrer que, en fait, $L(A') = L(A)$.

5 minutes de réflexion

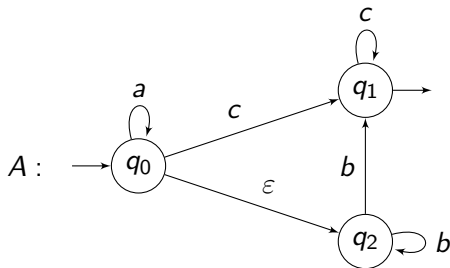


Vrai ou faux ?

- ❶ $acc \in L(A)$
- ❷ $acb \in L(A)$
- ❸ $abc \in L(A)$
- ❹ $abb \in L(A)$

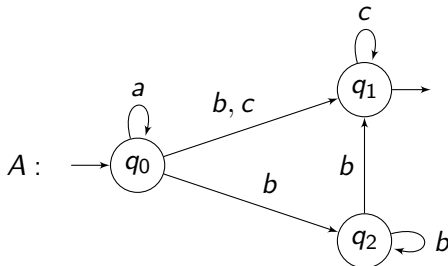
Construire l' ε -fermeture arrière de A .

5 minutes de réflexion



Vrai ou faux ?

- ① $acc \in L(A)$ ✓
- ② $acb \in L(A)$ ✗
- ③ $abc \in L(A)$ ✓
- ④ $abb \in L(A)$ ✓



Construire l' ε -fermeture arrière de A.

Déterminisation

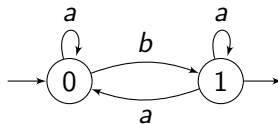
Automate des parties

Définition

Soit $A = (\Sigma, Q, Q_0, F, \delta)$ un automate fini. L'**automate des parties** de A est l'automate fini déterministe complet $A' = (\Sigma, Q', q'_0, F', \delta')$ définit comme suite :

- $Q' = \mathcal{P}(Q)$, l'ensemble des parties de Q ,
- $q'_0 = Q_0$,
- $F' = \{P \subseteq Q \mid P \cap F \neq \emptyset\}$, et
- $\delta'(P, a) = \{q \in Q \mid \exists p \in P : (p, a, q) \in \delta\}$.

Exemple (sur tableau)



Le non-déterminisme ne paye pas

Théorème

Pour tout automate fini A il existe un automate fini **déterministe complet** A' tel que $L(A') = L(A)$.

Démonstration.

- ① Notons $A = (\Sigma, Q, Q_0, F, \delta)$.
- ② Soit A' l'automate des parties de A , on montre que $L(A') = L(A)$.

Le non-déterminisme ne paye pas

Théorème

Pour tout automate fini A il existe un automate fini **déterministe complet** A' tel que $L(A') = L(A)$.

Démonstration.

- ① Notons $A = (\Sigma, Q, Q_0, F, \delta)$.
- ② Soit A' l'automate des parties de A , on montre que $L(A') = L(A)$.
- ③ Soit $w \in L(A)$, alors il existe un calcul réussi $\sigma = q_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} q_n$ dans A t.q. $\lambda(\sigma) = w$.

Le non-déterminisme ne paye pas

Théorème

Pour tout automate fini A il existe un automate fini **déterministe complet** A' tel que $L(A') = L(A)$.

Démonstration.

- ① Notons $A = (\Sigma, Q, Q_0, F, \delta)$.
- ② Soit A' l'automate des parties de A , on montre que $L(A') = L(A)$.
- ③ Soit $w \in L(A)$, alors il existe un calcul réussi
 $\sigma = q_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} q_n$ dans A t.q. $\lambda(\sigma) = w$.
- ④ Soit $Q_1 = \delta'(Q_0, a_1)$, $Q_2 = \delta'(Q_1, a_2)$ etc., alors
 $\sigma' = Q_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} Q_n$ est un calcul dans A' t.q. $\lambda(\sigma') = w$.

Le non-déterminisme ne paye pas

Théorème

Pour tout automate fini A il existe un automate fini **déterministe complet** A' tel que $L(A') = L(A)$.

Démonstration.

- ① Notons $A = (\Sigma, Q, Q_0, F, \delta)$.
- ② Soit A' l'automate des parties de A , on montre que $L(A') = L(A)$.
- ③ Soit $w \in L(A)$, alors il existe un calcul réussi
 $\sigma = q_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} q_n$ dans A t.q. $\lambda(\sigma) = w$.
- ④ Soit $Q_1 = \delta'(Q_0, a_1)$, $Q_2 = \delta'(Q_1, a_2)$ etc., alors
 $\sigma' = Q_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} Q_n$ est un calcul dans A' t.q. $\lambda(\sigma') = w$.
- ⑤ On a $q_i \in Q_i$ pour tout i , donc $q_n \in Q_n \cap F$, c.à.d. $Q_n \in F'$, alors σ' est un calcul réussi, donc $w \in L(A')$.

Le non-déterminisme ne paye pas

Théorème

Pour tout automate fini A il existe un automate fini **déterministe complet** A' tel que $L(A') = L(A)$.

Démonstration.

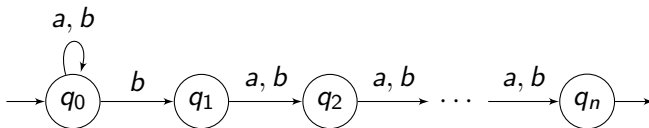
- ① Notons $A = (\Sigma, Q, Q_0, F, \delta)$.
- ② Soit A' l'automate des parties de A , on montre que $L(A') = L(A)$.
- ③ Soit $w \in L(A)$, alors il existe un calcul réussi
 $\sigma = q_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} q_n$ dans A t.q. $\lambda(\sigma) = w$.
- ④ Soit $Q_1 = \delta'(Q_0, a_1)$, $Q_2 = \delta'(Q_1, a_2)$ etc., alors
 $\sigma' = Q_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} Q_n$ est un calcul dans A' t.q. $\lambda(\sigma') = w$.
- ⑤ On a $q_i \in Q_i$ pour tout i , donc $q_n \in Q_n \cap F$, c.à.d. $Q_n \in F'$, alors
 σ' est un calcul réussi, donc $w \in L(A')$.

Et l'autre direction ?

Le non-déterminisme paye

- le non-déterminisme est utile pour des **spécifications partielles**
- des automates finis non-déterministes peuvent être **exponentiellement plus distinctes** que des automates finis déterministes :

Exercice : Pour $n \geq 2$ soit A_n l'automate fini comme suit :

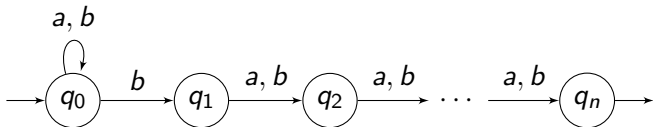


- 1 Trouver une expression rationnelle e_n telle que $L(e_n) = L(A_n)$.
- 2 Quelle est le nombre d'états le plus petit d'un automate fini **déterministe** A'_n tel que $L(A'_n) = L(A_n)$?

Le non-déterminisme paye

- le non-déterminisme est utile pour des **spécifications partielles**
- des automates finis non-déterministes peuvent être **exponentiellement plus distinctes** que des automates finis déterministes :

Exercice : Pour $n \geq 2$ soit A_n l'automate fini comme suit :



- 1 Trouver une expression rationnelle e_n telle que $L(e_n) = L(A_n)$.

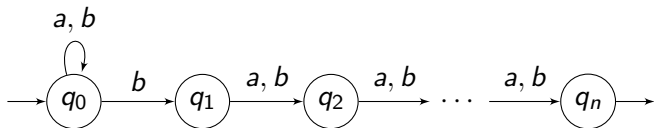
$$(a + b)^* b (a + b)^{n-1}$$

- 2 Quelle est le nombre d'états le plus petit d'un automate fini **déterministe** A'_n tel que $L(A'_n) = L(A_n)$?

Le non-déterminisme paye

- le non-déterminisme est utile pour des **spécifications partielles**
- des automates finis non-déterministes peuvent être **exponentiellement plus distinctes** que des automates finis déterministes :

Exercice : Pour $n \geq 2$ soit A_n l'automate fini comme suit :



- ① Trouver une expression rationnelle e_n telle que $L(e_n) = L(A_n)$.

$$(a + b)^* b (a + b)^{n-1}$$

- ② Quelle est le nombre d'états le plus petit d'un automate fini **déterministe** A'_n tel que $L(A'_n) = L(A_n)$?

$$2^n$$

Théorème de Kleene

Théorème de Kleene

Théorème (Kleene)

Un langage $L \subseteq \Sigma^*$ est *rationnel* ssi il est *reconnaissable*.

syntaxe

aut. finis dét. complets

\cap

aut. finis déterministes

\cap

automates finis

\cap

aut. finis à trans. spontanées

expressions rationnelles

$L(\cdot)$
 \longrightarrow

sémantique

langages reconnaissables

\parallel ✓

langages reconnaissables

\parallel ✓

langages reconnaissables

\parallel ✓

langages reconnaissables

\parallel ?

langages rationnelles

Algorithme de Thompson

Lemme (Thompson)

Pour toute expression rationnelle e il existe un automate fini à transitions spontanées A tel que $L(e) = L(A)$.

Démonstration.

- 1 Soit e une expression rationnelle.
- 2 On construit, par induction structurelle, un automate fini $A(e)$ à transitions spontanées tel que $L(A(e)) = L(e)$.
- 3 Nos automates vont être **pures**, avec un unique état initial sans transitions entrantes et symétriquement pour l'état final.
- 4 Si $e = \emptyset$, alors soit $A(e) = \longrightarrow \bigcirc \quad \bigcirc \longrightarrow$ (sans transitions).

Algorithme de Thompson

Lemme (Thompson)

Pour toute expression rationnelle e il existe un automate fini à transitions spontanées A tel que $L(e) = L(A)$.

Démonstration.

- 1 Soit e une expression rationnelle.
- 2 On construit, par induction structurelle, un automate fini $A(e)$ à transitions spontanées tel que $L(A(e)) = L(e)$.
- 3 Nos automates vont être **pures**, avec un unique état initial sans transitions entrantes et symétriquement pour l'état final.
- 4 Si $e = \emptyset$, alors soit $A(e) = \longrightarrow \bigcirc \quad \bigcirc \longrightarrow$ (sans transitions).
- 5 Si $e = \varepsilon$, alors soit $A(e) =$

Algorithme de Thompson

Lemme (Thompson)

Pour toute expression rationnelle e il existe un automate fini à transitions spontanées A tel que $L(e) = L(A)$.

Démonstration.

- ① Soit e une expression rationnelle.
- ② On construit, par induction structurelle, un automate fini $A(e)$ à transitions spontanées tel que $L(A(e)) = L(e)$.
- ③ Nos automates vont être **pures**, avec un unique état initial sans transitions entrantes et symétriquement pour l'état final.
- ④ Si $e = \emptyset$, alors soit $A(e) = \rightarrow \bigcirc \quad \bigcirc \rightarrow$ (sans transitions).
- ⑤ Si $e = \varepsilon$, alors soit $A(e) = \rightarrow \bigcirc \xrightarrow{\varepsilon} \bigcirc \rightarrow$.

Algorithme de Thompson

Lemme (Thompson)

Pour toute expression rationnelle e il existe un automate fini à transitions spontanées A tel que $L(e) = L(A)$.

Démonstration.

- ① Soit e une expression rationnelle.
- ② On construit, par induction structurelle, un automate fini $A(e)$ à transitions spontanées tel que $L(A(e)) = L(e)$.
- ③ Nos automates vont être **pures**, avec un unique état initial sans transitions entrantes et symétriquement pour l'état final.
- ④ Si $e = \emptyset$, alors soit $A(e) = \rightarrow \bigcirc \quad \bigcirc \rightarrow$ (sans transitions).
- ⑤ Si $e = \varepsilon$, alors soit $A(e) = \rightarrow \bigcirc \xrightarrow{\varepsilon} \bigcirc \rightarrow$.
- ⑥ Si $e = a \in \Sigma$, alors soit $A(e) =$

Algorithme de Thompson

Lemme (Thompson)

Pour toute expression rationnelle e il existe un automate fini à transitions spontanées A tel que $L(e) = L(A)$.

Démonstration.

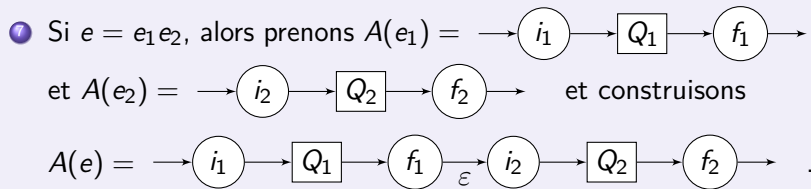
- ① Soit e une expression rationnelle.
- ② On construit, par induction structurelle, un automate fini $A(e)$ à transitions spontanées tel que $L(A(e)) = L(e)$.
- ③ Nos automates vont être **pures**, avec un unique état initial sans transitions entrantes et symétriquement pour l'état final.
- ④ Si $e = \emptyset$, alors soit $A(e) = \rightarrow \bigcirc \quad \bigcirc \rightarrow$ (sans transitions).
- ⑤ Si $e = \varepsilon$, alors soit $A(e) = \rightarrow \bigcirc \xrightarrow{\varepsilon} \bigcirc \rightarrow$.
- ⑥ Si $e = a \in \Sigma$, alors soit $A(e) = \rightarrow \bigcirc \xrightarrow{a} \bigcirc \rightarrow$.

Algorithme de Thompson

Lemme (Thompson)

Pour toute expression rationnelle e il existe un automate fini à transitions spontanées A tel que $L(e) = L(A)$.

Démonstration (suite).



Algorithme de Thompson

Lemme (Thompson)

Pour toute expression rationnelle e il existe un automate fini à transitions spontanées A tel que $L(e) = L(A)$.

Démonstration (suite).

8 Si $e = e_1 + e_2$, alors prenons $A(e_1) = \rightarrow i_1 \rightarrow Q_1 \rightarrow f_1 \rightarrow$
 et $A(e_2) = \rightarrow i_2 \rightarrow Q_2 \rightarrow f_2 \rightarrow$ et construisons

$A(e) =$

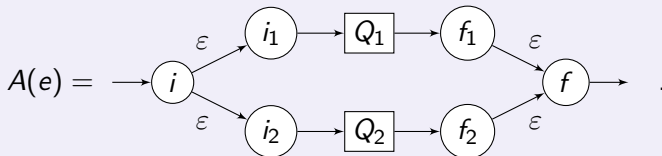
Algorithme de Thompson

Lemme (Thompson)

Pour toute expression rationnelle e il existe un automate fini à transitions spontanées A tel que $L(e) = L(A)$.

Démonstration (suite).

⑧ Si $e = e_1 + e_2$, alors prenons $A(e_1) = \rightarrow i_1 \rightarrow Q_1 \rightarrow f_1 \rightarrow$
 et $A(e_2) = \rightarrow i_2 \rightarrow Q_2 \rightarrow f_2 \rightarrow$ et construisons



Algorithme de Thompson

Lemme (Thompson)

Pour toute expression rationnelle e il existe un automate fini à transitions spontanées A tel que $L(e) = L(A)$.

Démonstration (suite).

9 Si $e = e_1^*$, alors prenons $A(e_1) = \rightarrow (i_1) \rightarrow [Q_1] \rightarrow (f_1) \rightarrow$
et construisons

$A(e) =$

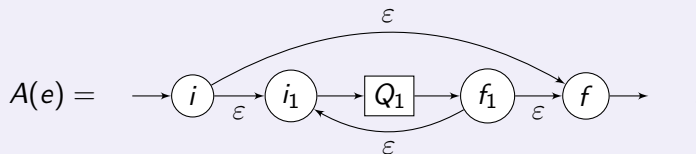
Algorithme de Thompson

Lemme (Thompson)

Pour toute expression rationnelle e il existe un automate fini à transitions spontanées A tel que $L(e) = L(A)$.

Démonstration (suite).

9 Si $e = e_1^*$, alors prenons $A(e_1) = \rightarrow (i_1) \rightarrow [Q_1] \rightarrow (f_1) \rightarrow$ et construisons



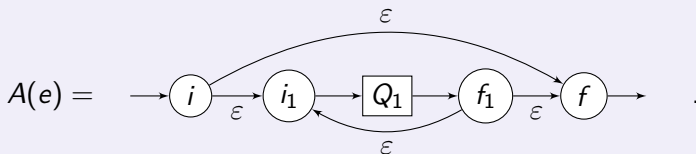
Algorithme de Thompson

Lemme (Thompson)

Pour toute expression rationnelle e il existe un automate fini à transitions spontanées A tel que $L(e) = L(A)$.

Démonstration (suite).

- 9 Si $e = e_1^*$, alors prenons $A(e_1) = \rightarrow (i_1) \rightarrow [Q_1] \rightarrow (f_1) \rightarrow$ et construisons



- 10 Maintenant il faut démontrer que $L(A(e)) = L(e)$ en chaque cas.

Exercice

Utiliser l'algorithme de Thompson pour convertir l'expression rationnelle $a(b^*a + b)$ en automate fini à transitions spontanées.

Théorème de Kleene

Théorème (Kleene)

Un langage $L \subseteq \Sigma^*$ est *rationnel* ssi il est *reconnaisable*.

Démonstration.

⇒ algorithme de Thompson : convertir une expression rationnelle dans un automate fini à transitions spontanées ✓

← algorithme de Brzozowski & McCluskey : convertir un automate fini dans une expression rationnelle ← maintenant

- outil : automates finis *généralisés*, avec transitions étiquetées en expressions rationnelles

Automates finis généralisés

Définition

Un **automate fini généralisé** est une structure $(\Sigma, Q, Q_0, F, \delta)$ où

- Σ est un ensemble fini de symboles,
- Q est un ensemble fini d'états,
- $Q_0 \subseteq Q$ est l'ensemble des états initiaux,
- $F \subseteq Q$ est l'ensemble des états finaux, et
- $\delta \subseteq Q \times RE(\Sigma) \times Q$ est la relation de transition.

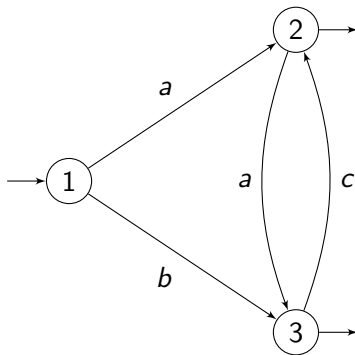
- un **calcul** dans A : $\sigma = q_1 \xrightarrow{e_1} q_2 \xrightarrow{e_2} \dots \xrightarrow{e_{n-1}} q_n$
- l'**étiquette** d'un calcul : $\lambda(\sigma) = e_1 e_2 \dots e_{n-1} \in RE(\Sigma)$
- un calcul **réussi** : $q_1 \in Q_0$ et $q_n \in F$
- Le **langage reconnu** par A :

$$L(A) = \bigcup \{L(\lambda(\sigma)) \mid \sigma \text{ calcul réussi dans } A\}$$

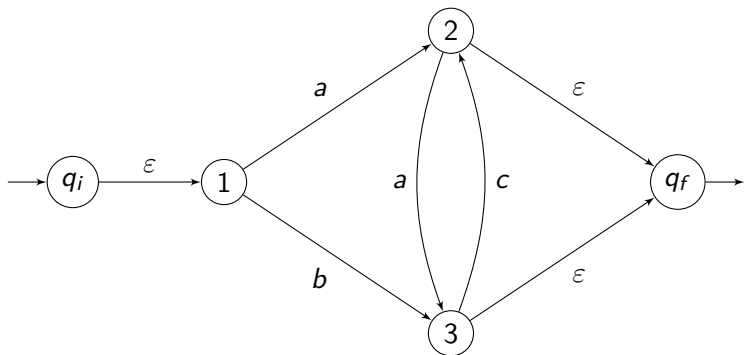
Algorithme de Brzowski & McCluskey

- ❶ Soit A un automate fini
- ❷ « Convertir » A en automate fini généralisé
- ❸ Convertir A en automate fini généralisé **pure** :
 - une unique transition entre chaque pair d'états
 - un état initial unique q_0 sans transitions entrantes
 - un état final unique q_f sans transitions sortantes
- ❹ while $Q \neq \{q_0, q_f\}$:
 - supprimer un état $q \notin \{q_0, q_f\}$
 - corriger étiquettes
- ❺ return l'étiquette de la transition unique

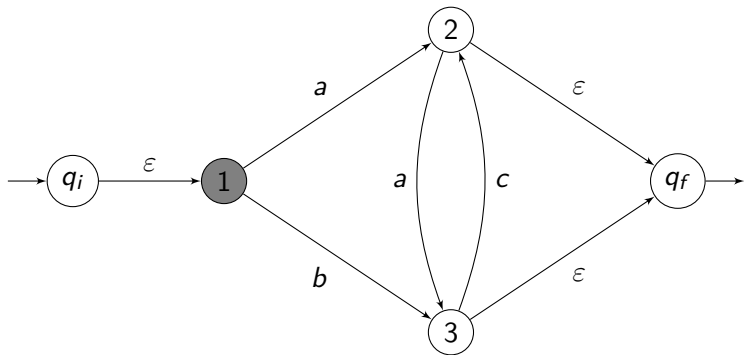
Exemple



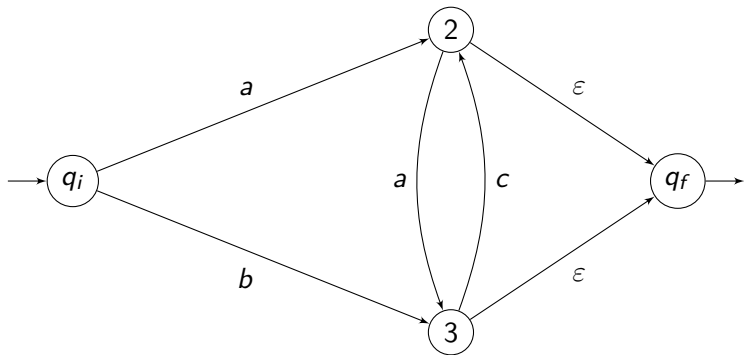
Exemple



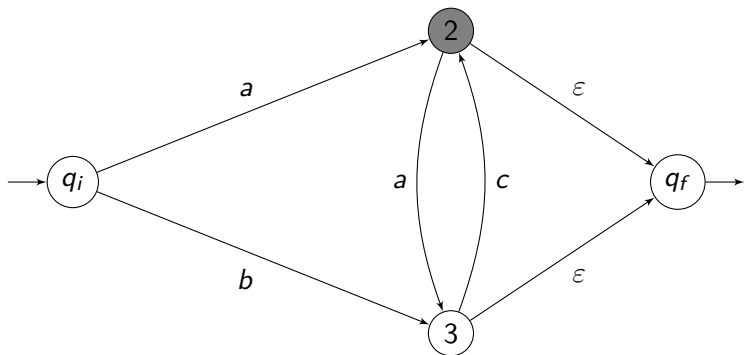
Exemple



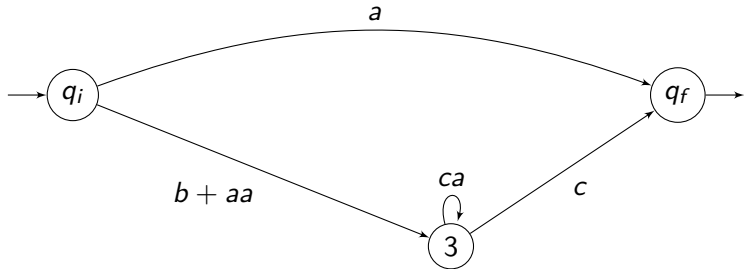
Exemple



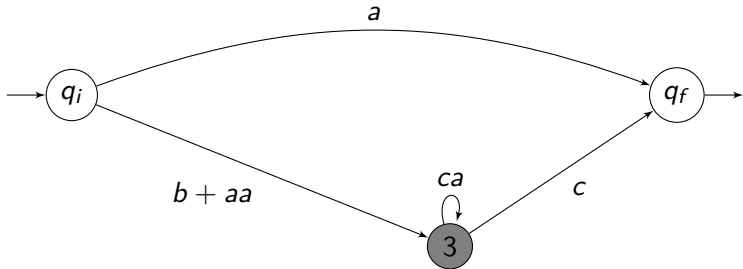
Exemple



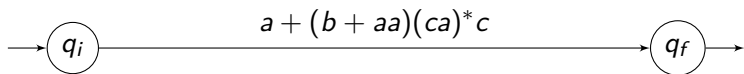
Exemple



Exemple



Exemple



Algorithme de Brzozowski & McCluskey, détail

- ① Soit $(\Sigma, Q, Q_0, F, \delta)$ un automate fini
- ② Convertir A en automate fini généralisé **pure** $(\Sigma, Q', q_0, f, \Delta)$:
 - une unique transition entre chaque pair d'états
 - un état initial unique q_0 sans transitions entrantes
 - un état final unique q_f sans transition sortante

Algorithme de Brzozowski & McCluskey, détail

- ① Soit $(\Sigma, Q, Q_0, F, \delta)$ un automate fini
- ② Convertir A en automate fini généralisé **pure** $(\Sigma, Q', q_0, f, \Delta)$:
 - une unique transition entre chaque pair d'états
 - un état initial unique q_0 sans transitions entrantes
 - un état final unique q_f sans transition sortante
- $Q' = Q \cup \{q_0, q_f\}$ pour $q_0, q_f \notin Q$
- $\Delta : Q' \times Q' \rightarrow RE(\Sigma)$
- $\Delta(q_1, q_2) = \sum \{a \mid (q_1, a, q_2) \in \delta\}$ pour $q_1, q_2 \in Q$
 - c.à.d. $\Delta(q_1, q_2) = \emptyset$ si $\{a \mid (q_1, a, q_2) \in \delta\} = \emptyset$
- $\Delta(q_i, q_2) = \begin{cases} \varepsilon & \text{si } q_2 \in Q_0 \\ \emptyset & \text{sinon} \end{cases} \quad \Delta(q_1, q_f) = \begin{cases} \varepsilon & \text{si } q_1 \in F \\ \emptyset & \text{sinon} \end{cases}$

Algorithme de Brzozowski & McCluskey, détail

- ① Soit $(\Sigma, Q, Q_0, F, \delta)$ un automate fini
- ② Convertir A en automate fini généralisé **pure** $(\Sigma, Q', q_0, f, \Delta)$:
- ③ while $Q \neq \{q_0, q_f\}$:
 - supprimer un état $q \notin \{q_0, q_f\}$
 - corriger étiquettes

Algorithme de Brzozowski & McCluskey, détail

- ① Soit $(\Sigma, Q, Q_0, F, \delta)$ un automate fini
- ② Convertir A en automate fini généralisé **pure** $(\Sigma, Q', q_0, f, \Delta)$:
- ③ while $Q \neq \{q_0, q_f\}$:
 - supprimer un état $q \notin \{q_0, q_f\}$
 - corriger étiquettes
 - $Q' \leftarrow Q' \setminus \{q\}$
 - pour tout $p, r \in Q'$ (donc aussi pour $p = r$!) :
 - $\Delta(p, r) \leftarrow \Delta(p, r) + \Delta(p, q)\Delta(q, q)^*\Delta(q, r)$

Algorithme de Brzozowski & McCluskey, détail

- ❶ Soit $(\Sigma, Q, Q_0, F, \delta)$ un automate fini
- ❷ Convertir A en automate fini généralisé **pure** $(\Sigma, Q', q_0, f, \Delta)$:
- ❸ while $Q \neq \{q_0, q_f\}$:
 - supprimer un état $q \notin \{q_0, q_f\}$
 - corriger étiquettes
 - $Q' \leftarrow Q' \setminus \{q\}$
 - pour tout $p, r \in Q'$ (donc aussi pour $p = r$!) :
 - $\Delta(p, r) \leftarrow \Delta(p, r) + \Delta(p, q)\Delta(q, q)^*\Delta(q, r)$
- ❹ return l'étiquette de la transition unique
 - donc $\Delta(q_i, q_f)$

Exercice

Utiliser

- 1 l'algorithme de Thompson pour convertir l'expression rationnelle $a(b^*a + b)$ en automate fini à transitions spontanées A ;
- 2 l'algorithme de Brzozowski et McCluskey pour reconverter A en expression rationnelle.

Conclusion

Récapitulatif

- ① Mots, langages
- ② Langages rationnels
- ③ Expressions rationnelles
- ④ Automates finis
- ⑤ Langages reconnaissables
 - poly chapitres 1-4
 - moins 2.3.2-2.3.5, 2.4.4, 3.1.3, 4.1.3, 4.2.1, 4.3, 4.4

Applications

- automate fini $\hat{=}$ algorithme en **mémoire constante**
- lien vers les algorithmes online / streaming
- passage, analyse lexicale, grep etc. : expression rationnelle \rightsquigarrow automate fini déterministe / non-déterministe (!)
- apprentissage par automates : automate fini $\hat{=}$ représentation compacte – algorithme de **Angluin**
- traduction automatique : automates **probabilistes**
- vérification : modélisation par automates probabilistes / pondérés / temporisés / hybrides / etc.
- **et après ?** langages algébriques, automates à pile, analyse syntaxique, compilation \rightsquigarrow **la prochaine fois**

The image features a classic target graphic with concentric circles. The outer rings are a deep red, while the inner rings transition to a lighter, more vibrant red. At the very center is a solid dark blue circle. Overlaid on this target is the text "That's all Folks!" in a white, elegant cursive script. The text is positioned diagonally, starting from the lower left and ending towards the upper right, with the central blue bullseye acting as a focal point for the phrase.

That's all Folks!