Théorie des langages rationnels : THLR
CM 3

Uli Fahrenberg

EPITA Rennes

S3 2024

Aperçu

Programme du cours

1. Mots, langages
2. Langages rationnels, expressions rationnelles
3. Automates finis
4. Langages non-rationnels
5. Langages reconnaissables, minimisation

Automates finis déterministes

Exemple

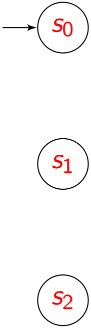
L'algorithme le plus simple qui décide le langage de tous les mots qui commencent par *ab* : $L = \{ab, aba, abb, abaa, abab, abba, \dots\}$
(en Python-èsque) :

```
def startsab(stream):
    state = "attend␣a"
    while x = next(stream):
        if state == "attend␣a":
            if x == "a":
                state = "attend␣b"
            else: return False
        elif state == "attend␣b":
            if x == "b":
                state = "done"
            else: return False
    if state == "done": 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-èsque) :

```
def startsab(stream):
    state = "attend␣a"
    while x = next(stream):
        if state == "attend␣a":
            if x == "a":
                state = "attend␣b"
            else: return False
        elif state == "attend␣b":
            if x == "b":
                state = "done"
            else: return False
    if state == "done": return True
    else: return False
```

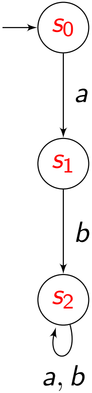


Exemple

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-èsque) :

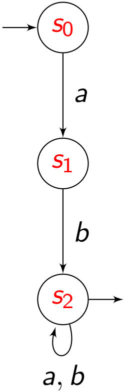
```
def startsab(stream):
    state = "attend a"
    while x = next(stream):
        if state == "attend a":
            if x == "a":
                state = "attend b"
            else: return False
        elif state == "attend b":
            if x == "b":
                state = "done"
            else: return False
    if state == "done": 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-èsque) :

```
def startsab(stream):
    state = "attend␣a"
    while x = next(stream):
        if state == "attend␣a":
            if x == "a":
                state = "attend␣b"
            else: return False
        elif state == "attend␣b":
            if x == "b":
                state = "done"
            else: return False
    if state == "done": return True
    else: return False
```



Automates finis déterministes complets

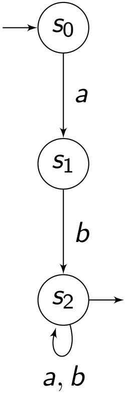
Définition (4.1)

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
- $\delta : Q \times \Sigma \to Q$ est la fonction de transition.

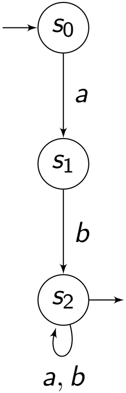
- un graphe orienté avec arcs étiquetés dans Σ et certains nœuds distingués comme initial et/ou final

Exemple



$$\Sigma = \{a, b\}$$
$$Q = \{s_0, s_1, s_2 \quad \}$$
$$q_0 = s_0$$
$$F = \{s_2\}$$

Exemple



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

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

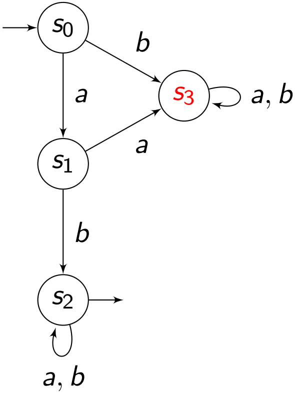
$$q_0 = s_0$$

$$F = \{s_2\}$$

δ :

	a	b
s_0	s_1	
s_1		s_2
s_2	s_2	s_2

Exemple



$$\Sigma = \{a, b\}$$
$$Q = \{s_0, s_1, s_2, s_3\}$$
$$q_0 = s_0$$
$$F = \{s_2\}$$

δ :

	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, $q_0 \in Q$, $F \subseteq Q$,
- $\delta : Q \times \Sigma \to Q$: la fonction de transition

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

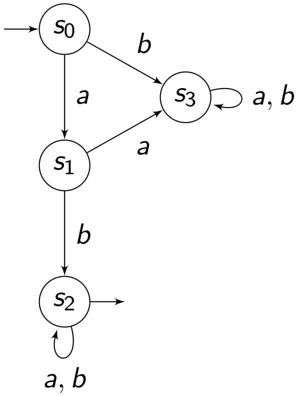
Définition

- Un calcul dans A est une séquence $\sigma = q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} q_n$.
  - 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 :

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

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

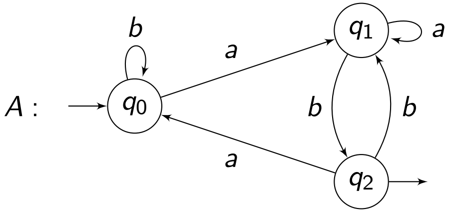
calculs réussis :

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

langage reconnu par A :

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

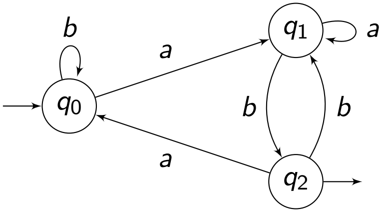
5 minutes de réflexion



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

A : 

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, $q_0 \in Q$, $F \subseteq Q$,
- $\delta : Q \times \Sigma \to Q$: 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

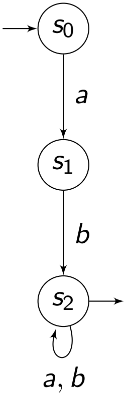
- Σ est un ensemble fini de symboles,
- Q est un ensemble fini d'états,
- $q_0 \in Q$ est l'état initial,
- $F \subseteq Q$ est l'ensemble des états finaux, et
- $\delta : Q \times \Sigma \rightharpoonup Q$ est la fonction partielle de transition.

- tout automate fini déterministe peut être complété en ajoutant un état puits (voir p. 30)

Exemple

Automate fini déterministe et complétion :

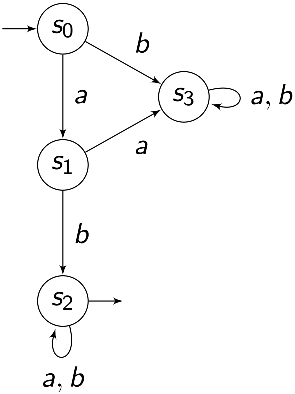
```
def startsab(stream):
    state = 0
    while x = next(stream):
        if state == 0:
            if x == "a":
                state = 1
            else: return False
        elif state == 1:
            if x == "b":
                state = 2
            else: return False
    if state == 2: return 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 $L(A') = L(A)$.

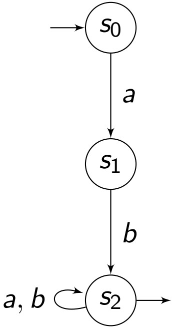
Démonstration.

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. $q_o' = q_0$ et $F' = F$.
5. La fonction $\delta : Q' \times \Sigma \to Q'$ 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

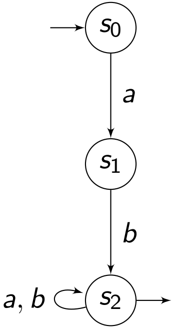
L'algorithme le plus simple qui décide le langage de tous les mots qui commencent par *ab* :



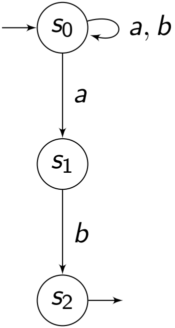
L'algorithme le plus simple qui décide le langage de tous les mots qui se terminent par *ab* :

Exemple

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

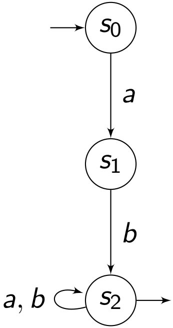


L'algorithme le plus simple qui décide le langage de tous les mots qui se terminent par *ab* :

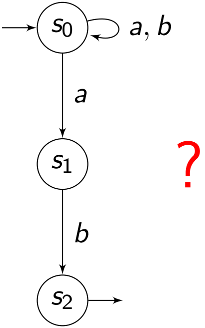


Exemple

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 $\sigma = q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} q_n$.
- 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 $(q, a, r) \in \delta$. $\Longleftarrow$ la seule chose qui a changé !

Définition

- Un calcul dans A est une séquence $\sigma = q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} q_n$.
- L'étiquette d'un calcul comme ci-dessus est

  $\lambda(\sigma) = a_1 a_2 \ldots a_{n-1} \in \Sigma^*$.
- Un calcul comme ci-dessus est réussi si $q_1 \in Q_0$ et $q_n \in F$.
- Le langage reconnu par A est

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

Langages reconnaissables

Théorème

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

- pour la démonstration faut attendre plus tard
- en fait, tout les automates qu'on a vu sont équivalent :

Langages reconnaissables

Définition

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

Théorème

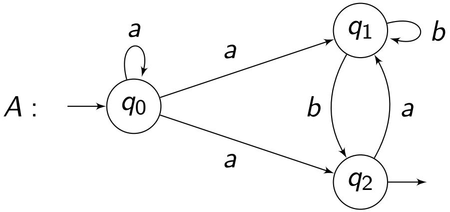
Un langage $L \subseteq \Sigma^*$ est reconnaissable ssi il existe un automate fini

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

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

- démonstration plus tard

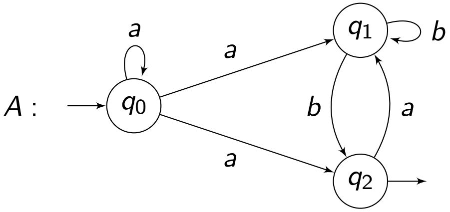
Exercise



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. $L(a^* ab^* b) \subseteq L(A)$

Exercise



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. $L(a^* ab^* b) \subseteq L(A)$ ✓

Des expressions rationnelles aux automates

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 $(q, a, r) \in \delta$. $\Longleftarrow$ donc a peut être ε

Définition

- Un calcul dans A est une séquence $\sigma = q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} q_n$.
- 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
  $L(A) = \{\lambda(\sigma) \mid \sigma$ 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		sémantique
aut. finis dét. complets		langages reconnaissables
$\rotatebox{90}{$\subsetneq$}$		$\parallel$ ✓
aut. finis déterministes		langages reconnaissables
$\rotatebox{90}{$\subsetneq$}$		$\parallel$?
automates finis	$\xrightarrow{L(\cdot)}$	langages reconnaissables
$\rotatebox{90}{$\subsetneq$}$		$\parallel$!
aut. finis à trans. spontanées		langages reconnaissables
		$\parallel$ $\Uparrow$
expressions rationnelles		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 $q \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} r$ 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, $L(A') = L(A)$.

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 $e = \varnothing$, alors soit $A(e) =$  (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.

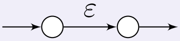
1. Soit e une expression rationnelle.
2. On construit, par induction structurelle, un automate fini $A(e)$ à transitions spontanées tel que $L(A(e)) = L(e)$.
3. Nos automates vont être pures, avec un unique état initial sans transitions entrantes et symétriquement pour l'état final.
4. Si $e = \varnothing$,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

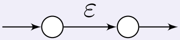
1. Soit e une expression rationnelle.
2. On construit, par induction structurelle, un automate fini $A(e)$ à transitions spontanées tel que $L(A(e)) = L(e)$.
3. Nos automates vont être pures, avec un unique état initial sans transitions entrantes et symétriquement pour l'état final.
4. Si $e = \emptyset$,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

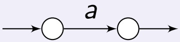
Algorithme de Thompson

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 $A(e) =$  (sans transitions).
5. Si $e = \varepsilon$, alors soit $A(e) =$ 

.
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 (suite).

7. Si $e = e_1 e_2$, alors prenons $A(e_1) = \longrightarrow (i_1) \longrightarrow \boxed{Q_1} \longrightarrow (f_1) \longrightarrow$ et $A(e_2) = \longrightarrow (i_2) \longrightarrow \boxed{Q_2} \longrightarrow (f_2) \longrightarrow$ et construisons

$$A(e) = \longrightarrow (i_1) \longrightarrow \boxed{Q_1} \longrightarrow (f_1) \xrightarrow{\varepsilon} (i_2) \longrightarrow \boxed{Q_2} \longrightarrow (f_2) \longrightarrow \quad .$$

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 $A(e_1) = \longrightarrow (i_1) \longrightarrow \boxed{Q_1} \longrightarrow (f_1) \longrightarrow$

   et $A(e_2) = \longrightarrow (i_2) \longrightarrow \boxed{Q_2} \longrightarrow (f_2) \longrightarrow$ et construisons

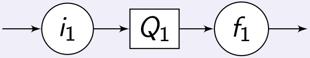
   $A(e) =$

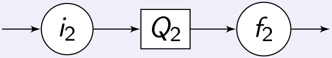
Algorithme de Thompson

Lemme (Thompson)

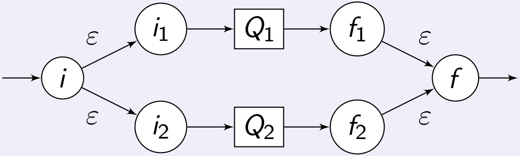
Pour toute expression rationnelle e il existe un automate fini à transitions spontanées A tel que $L(e) = L(A)$.

Démonstration (suite).

8. Si $e = e_1 + e_2$, alors prenons $A(e_1) =$ 

et $A(e_2) =$ 

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 $A(e_1) = \longrightarrow (i_1) \longrightarrow \boxed{Q_1} \longrightarrow (f_1) \longrightarrow$

   et construisons

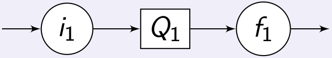
   $A(e) =$

Algorithme de Thompson

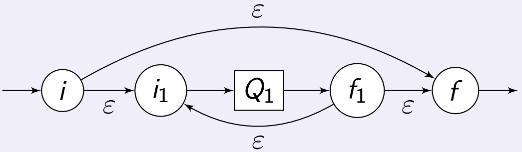
Lemme (Thompson)

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 $A(e_1) =$ 

   et construisons

$A(e) =$ 

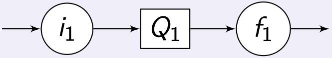
.

Algorithme de Thompson

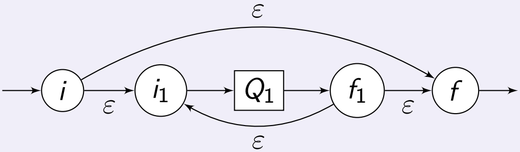
Lemme (Thompson)

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 $A(e_1) =$ 

   et construisons

$A(e) =$ 

.

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.


That's all Folks!