Théorie des langages rationnels : THLR

CM 10

Uli Fahrenberg

EPITA Rennes

S3 2022

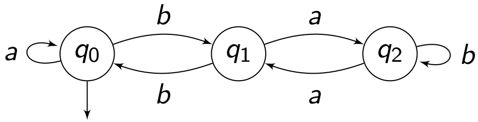
Aperçu

Programme du cours

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

Pour aller plus loin

A :



- soit L_i le langage reconnu par A avec état initial q_i
- alors
$$\begin{aligned} L_0 &= \{a\}L_0 \cup \{b\}L_1 \cup \{\varepsilon\} \\ L_1 &= \{a\}L_2 \cup \{b\}L_0 \\ L_2 &= \{a\}L_1 \cup \{b\}L_2 \end{aligned}$$
- comme équation matrice-vecteur : $\begin{bmatrix} L_0 \\ L_1 \\ L_2 \end{bmatrix} = \begin{bmatrix} a & b & \emptyset \\ b & \emptyset & a \\ \emptyset & a & b \end{bmatrix} \begin{bmatrix} L_0 \\ L_1 \\ L_2 \end{bmatrix} \cup \begin{bmatrix} \varepsilon \\ \emptyset \\ \emptyset \end{bmatrix}$

Slogan

Un automate fini sur Σ est une transformation affine dans un espace vectoriel sur le demi-anneau $\mathcal{P}(\Sigma^*)$.

Pour aller plus loin

Un demi-anneau est une structure algébrique $(S, \oplus, \otimes, \mathbb{0}, \mathbb{1})$ telle que

- $(S, \oplus, \mathbb{0})$ forme un monoïde commutatif,
- $(S, \otimes, \mathbb{1})$ forme un monoïde,
- $x(y \oplus z) = xy \oplus xz$, $(x \oplus y)z = xz \oplus yz$ et $x\mathbb{0} = \mathbb{0}x = \mathbb{0}$

S est idempotent si $x \oplus x = x$.

Théorème

L'ensemble de langages finis forme le demi-anneau idempotent libre.

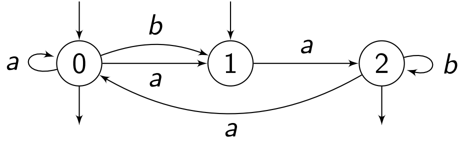
Une algèbre de Kleene est un demi-anneau idempotent S équipé avec toutes les sommes géométriques $\bigoplus_{n\geq 0} x^n$, pour tout $x \in S$, et telle que $x\left(\bigoplus_{n\geq 0} y^n\right)z = \bigoplus_{n\geq 0}(xy^n z)$ pour tout $x, y, z \in S$.

Théorème

L'ensemble de langages rationnels forme l'algèbre de Kleene libre.

Pour aller plus loin

A :



Définition (variante)

Un automate fini (pondéré) avec n états sur un demi-anneau S est composé d'une matrice $\Delta \in S^{n\times n}$ et deux vecteurs $i, f \in S^n$.

- ici : $S = \mathcal{P}(\Sigma^*)$, $\Delta = \begin{bmatrix} \{a\} & \{a,b\} & \emptyset \\ \emptyset & \emptyset & \{a\} \\ \{a\} & \emptyset & \{b\} \end{bmatrix}$, $i = \begin{bmatrix} \{\varepsilon\} \\ \{\varepsilon\} \\ \emptyset \end{bmatrix}$, $f = \begin{bmatrix} \{\varepsilon\} \\ \emptyset \\ \{\varepsilon\} \end{bmatrix}$

Théoréme / définition

Si S est une algèbre de Kleene, alors $S^{n\times n}$ l'est aussi, et le langage reconnu par A est $L(A) = i\Delta^* f$.

Langages reconnaissables

Théorème de Kleene

Théorème (Kleene)

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

Démonstration.

$\Longrightarrow$ algorithme de Thompson : convertir une expression rationelle dans un automate fini à transitions spontanées

$\Longleftarrow$ algorithme de Brzozowski & McCluskey : convertir un automate fini dans une expression rationelle $\longleftarrow$ 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.

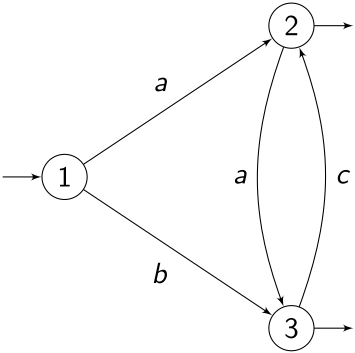
- un calcul dans A : $\sigma = q_1 \xrightarrow{e_1} q_2 \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} q_n$
- l'étiquette d'un calcul : $\lambda(\sigma) = e_1 e_2 \dots e_{n-1} \in RE(\Sigma)$
- un calcul réussi : $q_1 \in Q_0$ et $q_n \in F$
- Le langage reconnu par A :

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

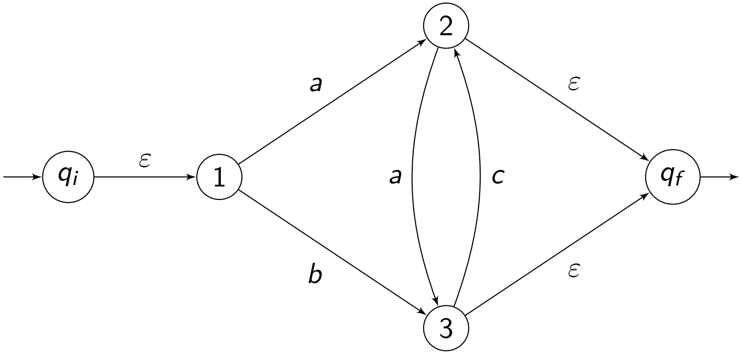
Algorithme de Brzozowski & McCluskey

1. Soit A un automate fini
2. « Convertir » A en automate fini généralisé
3. 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 transition sortante
4. `while` $Q \neq \{q_0, q_f\}$:
   - supprimer un état $q \notin \{q_0, q_f\}$
   - corriger étiquettes
5. `return` l'étiquette de la transition unique

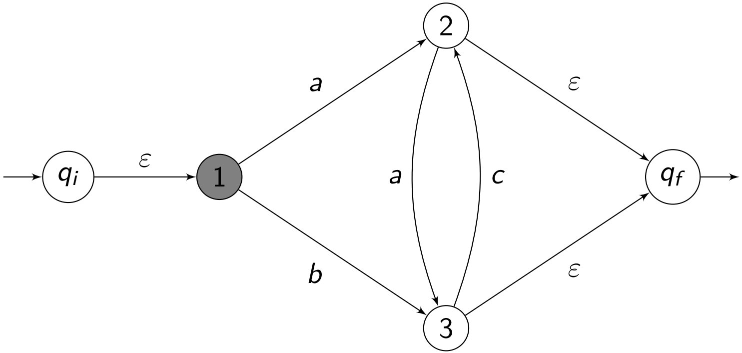
Exemple



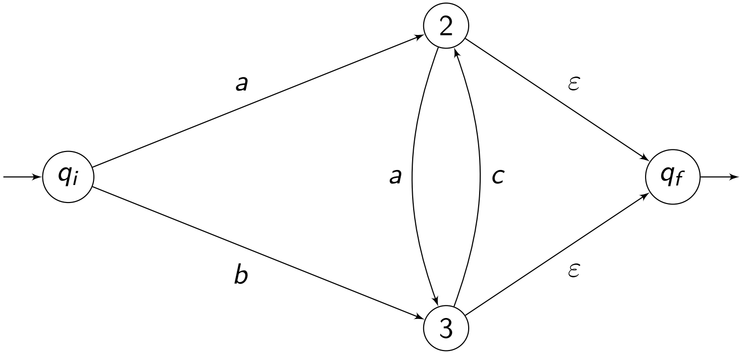
Exemple



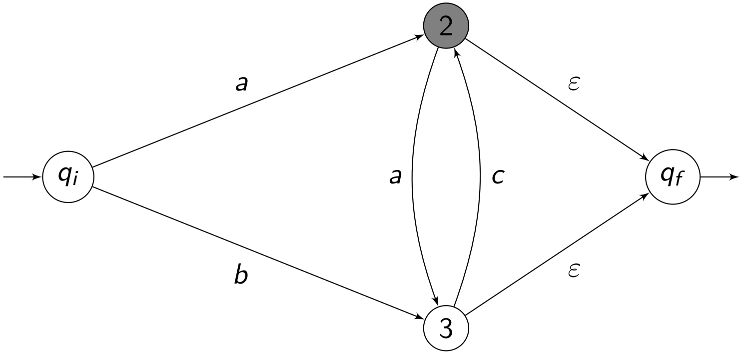
Exemple



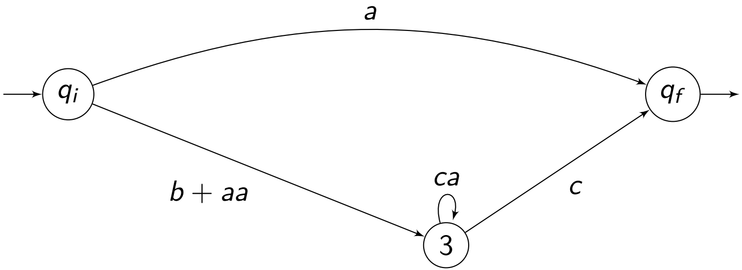
Exemple



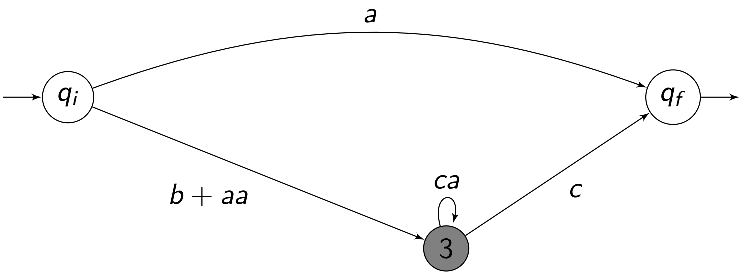
Exemple



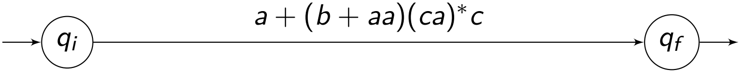
Exemple



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 $q_0, q_f \notin Q$
- $\Delta : Q' \times Q' \to RE(\Sigma)$
- $\Delta(q_1, q_2) = \sum \{a \mid (q_1, a, q_2) \in \delta\}$ pour $q_1, q_2 \in Q$
  - c.à.d. $\Delta(q_1, q_2) = \varnothing$ si $\{a \mid (q_1, a, q_2) \in \delta\} = \emptyset$
- $\Delta(q_i, q_2) = \begin{cases} \varepsilon & \text{si } q_2 \in Q_0 \\ \varnothing & \text{sinon} \end{cases}$ $\qquad \Delta(q_1, q_f) = \begin{cases} \varepsilon & \text{si } q_1 \in F \\ \varnothing & \text{sinon} \end{cases}$

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 $p = q$!) :
- $\Delta(p, r) \leftarrow \Delta(p, r) + \Delta(p, q)\Delta(q, q)^*\Delta(q, r)$

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 $p = q$!) :
- $\Delta(p, r) \leftarrow \Delta(p, r) + \Delta(p, q)\Delta(q, q)^*\Delta(q, r)$

4. `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 reconvertir A en expression rationnelle.

Conclusion

Récapitulatif

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

- poly chapitres 1-4
- moins 2.3.2, 2.3.5, 2.4.4 et 4.1.3

Applications

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

Pour aller plus loin

- (le poly !)
- O. Carton, *Langages formels*. Vuibert 2014
- J. Sakarovitch, *Eléments de théorie des automates*. Vuibert 2003
- T.A. Sudkamp, *Languages and Machines*. Addison-Wesley 2005
- D.C. Kozen, *Automata and Computability*. Springer 2012

- algèbre de Kleene pour la vérification des programmes
- Tony Hoare et.al : Concurrent Kleene algebra pour la vérification des systèmes distribués

Relational and Algebraic Methods in Computer Science (RAMICS 2021)

19th International Conference on Relational and Algebraic Methods in Computer Science RAMICS 2021

RAMICS 2021 will take place at CIRM close to Marseille from **2 to 5 November** 2021.

Since 1994, the RAMiCS conference series has been the main venue for research on relation algebras, Kleene similar algebraic formalisms, and their applications as conceptual and methodological tools in computer scienc

Theoretical aspects include semigroups, residuated lattices, semirings, Kleene algebras, relation algebras, other algebras; their connections with program logics and other logics; their use in the theories of automata formal languages, games, networks and programming languages; the development of algebraic, algorithr theoretic, coalgebraic and proof-theoretic methods for these theories; their formalisation with theorem provers

Applications include tools and techniques for program correctness, specification and verification; quantitative models and semantics of computing systems and processes; algorithm design, automated reasoning, net analysis, social choice, optimisation and control.

Invited Speakers

- Marcelo Frias, Argentina
- Barbara König, Germany
- Dmitriy Zhuk, Russia

Accepted papers

List of accepted papers.