Automates, Langages et Logique
4. Théorème de Kleene

L2, Université d'Orléans — S1 2024/2025

Nicolas Ollinger

Rappel des épisodes précédents

La famille des **langages reconnaissables** par **automates finis** sur un alphabet A est notée $\mathrm{Rec}\, A^*$. Elle contient tous les **langages finis**.

Proposition La familles des **langages reconnaissables** est stable par :

(a) union ; $\forall L, L_1, L_2 \in \mathrm{Rec}\, A^*, \quad L_1 + L_2 \in \mathrm{Rec}\, A^*$

(b) concaténation ; $L_1 \cdot L_2 \in \mathrm{Rec}\, A^*$

(c) étoile de Kleene ; $L^* \in \mathrm{Rec}\, A^*$

Rappel des épisodes précédents

La famille des **langages reconnaissables** par **automates finis** sur un alphabet A est notée $\mathrm{Rec}\, A^*$. Elle contient tous les **langages finis**.

Proposition La familles des **langages reconnaissables** est stable par :

(a)	union ;	$\forall L, L_1, L_2 \in \mathrm{Rec}\, A^*, \quad L_1 + L_2 \in \mathrm{Rec}\, A^*$
(b)	concaténation ;	$L_1 \cdot L_2 \in \mathrm{Rec}\, A^*$
(c)	étoile de Kleene ;	$L^* \in \mathrm{Rec}\, A^*$
(d)	intersection ;	$L_1 \cap L_2 \in \mathrm{Rec}\, A^*$
(e)	passage au complémentaire ;	$A^* \setminus L \in \mathrm{Rec}\, A^*$
(f)	image par un morphisme ;	$\varphi(L) \in \mathrm{Rec}\, A^*$
(g)	image inverse.	$\varphi^{-1}(L) \in \mathrm{Rec}\, A^*$

Méthode de travail

📢 Il est **fortement conseillé** de **venir en cours** et de **prendre des notes**.

Ce support est un **résumé** très concis de ce qu'on y raconte ! 🎓

Langages rationnels

Définition La famille $\mathrm{Rat}\, A^*$ des **langages rationnels** sur l'alphabet A est la plus petite famille de langages qui contient tous les **langages finis** sur A et qui est stable par les **opérations rationnelles** :

(i) somme (**union**) ; $\forall L, L_1, L_2 \in \mathrm{Rat}\, A^*, \quad L_1 + L_2 \in \mathrm{Rat}\, A^*$

(ii) produit (**concaténation**) ; $L_1 \cdot L_2 \in \mathrm{Rat}\, A^*$

(iii) itération (**étoile de Kleene**). $L^* \in \mathrm{Rat}\, A^*$

Proposition Les opérations d'**union** et de **concaténation** sont **associatives**. L'opération d'**union** est **commutative**.

$$(A + B) + C = A + (B + C) \qquad (AB)C = A(BC) \qquad A + B = B + A$$

Expressions rationnelles/régulières

Définition Une **expression rationnelle** décrit un **langage rationnel** à partir du langage vide $\emptyset$, des langages à une lettre $\{x\}$, du langage du mot vide $\{\varepsilon\}$ et des opérations de **produit**, **somme** et **étoile**.

La syntaxe est simplifiée pour une écriture plus lisible en retirant les accolades des langages à un mot et en tirant parti de l'associativité :

$$E, F := \emptyset \mid \varepsilon \mid x \mid E + F \mid EF \mid E^* \mid (E)$$

Les priorités sont les priorités usuelles : l'étoile est prioritaire sur le produit qui est prioritaire sur la somme.

$$(a + ab + bba + bb^*a)^*a + \varepsilon$$

Parfois on trouve la notation 0 l'ensemble $\emptyset$ et 1 l'ensemble $\{\varepsilon\}$.

Théorème de Kleene

Théorème de Kleene Un langage est **rationnel** si et seulement si il est **reconnaissable**.

$$\mathrm{Rec}\, A^* = \mathrm{Rat}\, A^*$$

Théorème de Kleene

Théorème de Kleene Un langage est **rationnel** si et seulement si il est **reconnaissable**.

$$\mathrm{Rec}\, A^* = \mathrm{Rat}\, A^*$$

Démontrons-le!

$\mathrm{Rat}\, A^* \subseteq \mathrm{Rec}\, A^*$

par clôture

$\mathrm{Rec}\, A^*$ contient tous les **langages finis** et est stable par **somme**, **produit** et **itération** donc $\mathrm{Rat}\, A^* \subseteq \mathrm{Rec}\, A^*$.

Étant donnée une **expression rationnelle**, partant des **langages finis**, on construit inductivement un automate fini à partir des **sous-expressions** et des **opérations rationnelles**.

$\mathrm{Rec}\, A^* \subseteq \mathrm{Rat}\, A^*$ par le lemme d'Arden

Soit $L \in \mathrm{Rec}\, A^*$ un langage reconnu par un **automate fini déterministe émondé** (Q, A, δ, q_0, F). Montrons que L est **rationnel** en lui construisant une **expression rationnelle**.

$\mathrm{Rec}\, A^* \subseteq \mathrm{Rat}\, A^*$ par le lemme d'Arden

Soit $L \in \mathrm{Rec}\, A^*$ un langage reconnu par un **automate fini déterministe émondé** (Q, A, δ, q_0, F). Montrons que L est **rationnel** en lui construisant une **expression rationnelle**.

Notons $L_q = \{u \in A^* | \delta^*(q, u) \in F\}$ le **langage reconnu** par un état $q \in Q$ et posons $\nu_q = L_q \cap \{\varepsilon\}$.

$\mathrm{Rec}\, A^* \subseteq \mathrm{Rat}\, A^*$ par le lemme d'Arden

Soit $L \in \mathrm{Rec}\, A^*$ un langage reconnu par un **automate fini déterministe émondé** (Q, A, δ, q_0, F). Montrons que L est **rationnel** en lui construisant une **expression rationnelle**.

Notons $L_q = \{u \in A^* | \delta^*(q, u) \in F\}$ le **langage reconnu** par un état $q \in Q$ et posons $\nu_q = L_q \cap \{\varepsilon\}$. On construit un **système d'équations de langages** :

$$L_q = \nu_q + \sum_{\delta(q,x)=q'} x \cdot L_{q'}$$

$\mathrm{Rec}\, A^* \subseteq \mathrm{Rat}\, A^*$

par le lemme d'Arden

Soit $L \in \mathrm{Rec}\, A^*$ un langage reconnu par un **automate fini déterministe émondé** (Q, A, δ, q_0, F). Montrons que L est **rationnel** en lui construisant une **expression rationnelle**.

Notons $L_q = \{u \in A^* | \delta^*(q, u) \in F\}$ le **langage reconnu** par un état $q \in Q$ et posons $\nu_q = L_q \cap \{\varepsilon\}$. On construit un **système d'équations de langages** :

$$L_q = \nu_q + \sum_{\delta(q,x)=q'} x \cdot L_{q'}$$

On **résout** le système en **éliminant successivement** les états à l'aide du lemme suivant, jusqu'à calculer $L = L_{q_0}$.

Lemme d'Arden Soient $X, Y \subseteq A^*$ deux langages. Toute solution de l'**équation de langages** $L = X \cdot L + Y$ contient le langage X^*Y.
Si X ne contient pas le mot vide alors cette solution est **unique**.

Algorithmes et complexité

Le **théorème de Kleene** est démontré, mais…

Algorithmes et complexité

Le **théorème de Kleene** est démontré, mais…

…en pratique, comment le mettre en œuvre **efficacement** ?

Algorithmes et complexité

Le **théorème de Kleene** est démontré, mais…

…en pratique, comment le mettre en œuvre **efficacement** ?

Approche **constructive** du théorème : des **algorithmes efficaces** pour

- produire des objets de **taille raisonnable** ;
- les produire en un **temps raisonnable**.

Algorithmes et complexité

Le **théorème de Kleene** est démontré, mais…

…en pratique, comment le mettre en œuvre **efficacement** ?

Approche **constructive** du théorème : des **algorithmes efficaces** pour

- produire des objets de **taille raisonnable** ;
- les produire en un **temps raisonnable**.

Nous présentons quelques-uns des **algorithmes classiques**.

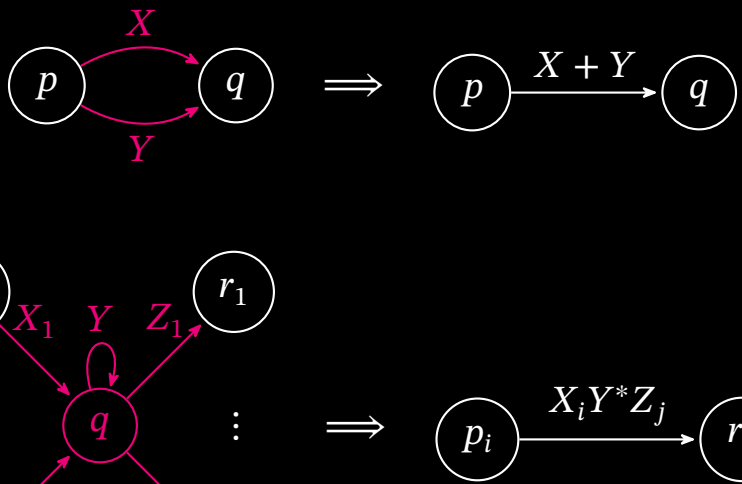
Approche par **élimination des états** : on transforme pas à pas un automate en un expression régulière en le simplifiant.

Définition Un **automate généralisé** est un **AFN** avec un **unique état initial**, un **unique état acceptant** et dont les transitions sont étiquetées par des **expressions rationnelles**. L'état initial n'admet aucune transition entrante. L'état acceptant n'admet aucune transition sortante.

Approche par **élimination des états** : on transforme pas à pas un automate en un expression régulière en le simplifiant.

Définition Un **automate généralisé** est un **AFN** avec un **unique état initial**, un **unique état acceptant** et dont les transitions sont étiquetées par des **expressions rationnelles**. L'état initial n'admet aucune transition entrante. L'état acceptant n'admet aucune transition sortante.

On commence par transformer l'**automate en entrée** en un **automate généralisé** puis on **élimine les états** les uns après les autres en prenant soin de **factoriser les transitions**.



Règles d'élimination

$\mathrm{Rec}\, A^* \subseteq \mathrm{Rat}\, A^*$ algorithme de McNaughton et Yamada

Soit $L \in \mathrm{Rec}\, A^*$ un langage reconnu par un **AFN** (Q, A, T, I, F) où $Q = \{1, \dots, n\}$. L'algorithme calcule récursivement des **expressions rationnelles** pour les langages

$$R_{p,q} = \{u \in A^* \big| q \in \delta^*(p, u)\} \quad \forall p, q \in Q.$$

$\mathrm{Rec}\, A^* \subseteq \mathrm{Rat}\, A^*$ — algorithme de McNaughton et Yamada

Soit $L \in \mathrm{Rec}\, A^*$ un langage reconnu par un **AFN** (Q, A, T, I, F) où $Q = \{1, \ldots, n\}$. L'algorithme calcule récursivement des **expressions rationnelles** pour les langages

$$R_{p,q} = \{u \in A^* \,|\, q \in \delta^*(p, u)\} \quad \forall p, q \in Q.$$

$$R_{p,q}^{(0)} = \sum_{(p,a,q) \in T} a$$

$$R_{p,q}^{(k+1)} = R_{p,q}^{(k)} + R_{p,k+1}^{(k)} \left(R_{k+1,k+1}^{(k)}\right)^* R_{k+1,q}^{(k)} \qquad \forall k < n$$

$\mathrm{Rec}\, A^* \subseteq \mathrm{Rat}\, A^*$ — algorithme de McNaughton et Yamada

Soit $L \in \mathrm{Rec}\, A^*$ un langage reconnu par un **AFN** (Q, A, T, I, F) où $Q = \{1, \ldots, n\}$. L'algorithme calcule récursivement des **expressions rationnelles** pour les langages

$$R_{p,q} = \{u \in A^* \,|\, q \in \delta^*(p,u)\} \quad \forall p, q \in Q.$$

$$R_{p,q}^{(0)} = \sum_{(p,a,q)\in T} a$$

$$R_{p,q}^{(k+1)} = R_{p,q}^{(k)} + R_{p,k+1}^{(k)} \left(R_{k+1,k+1}^{(k)}\right)^* R_{k+1,q}^{(k)} \qquad \forall k < n$$

Enfin, on conclut :

$$R_{p,q} = \begin{cases} R_{p,q}^{(n)} & \text{si } p \neq q \\ \varepsilon + R_{p,q}^{(n)} & \text{si } p = q \end{cases}$$

$$L = \sum_{p\in I} \sum_{q\in F} R_{p,q}$$

```
1: function McNaughton-Yamada(Q, A, T, I, F)
2:     n ← |Q|
3:     R ← CreateMatrix(n, n, ∅)
4:     R′ ← CreateMatrix(n, n, ∅)
5:     # Initialisation
6:     for p = 1 ... n do
7:         for q = 1 ... n do
8:             R_{p,q} ← ∑_{(p,a,q)∈T} a
```

```
9:  # Calcul itératif
10: for k = 1 ... n do
11:     for p = 1 ... n do
12:         for q = 1 ... n do
13:             R'_{p,q} ← R_{p,q} + R_{p,k} (R_{k,k})^* R_{k,q}
14:     R ← R'
```

```
15:     # Expression finale
16:     E ← ∅
17:     ν ← ∅
18:     for all p ∈ I do
19:         for all q ∈ F do
20:             E ← E + R_{p,q}
21:             if p = q then
22:                 ν ← ε
23:     return ν + E
```

L'**automate de Glushkov** d'une **expression rationnelle** e correspondant à un langage $L \subseteq A^*$ est un automate non déterministe de taille similaire.

Les **états** Q de l'automate sont les **positions de l'expression** qui portent des **lettres**. On construit inductivement à partir de e : les **positions initiales** $P \subseteq Q$, les **positions terminales** $D \subseteq Q$ et les **transitions** $T \subseteq Q \times Q$.

Pour traiter le mot vide on pose $\Lambda = L \cap \{\varepsilon\}$.

$$L = \pi\left((PQ^* \cap Q^*D) \setminus Q^*\left(Q^2 \setminus T\right)Q^*\right) \cup \Lambda$$

L'automate obtenu est $(Q \cup \{i\}, A, \mathcal{T}, \{i\}, \mathcal{F})$ où

$$\begin{aligned} \mathcal{F} &= \begin{cases} D \cup \{i\} & \text{si } \varepsilon \in L \\ D & \text{sinon} \end{cases} \\ \mathcal{T} &= \{(p, \pi(q), q) | (p,q) \in T\} \cup \{(i, \pi(q), q) | q \in P\} \end{aligned}$$

Tester inductivement si $\varepsilon \in L$

$$
\begin{aligned}
\Lambda(\emptyset) &= \emptyset \\
\Lambda(\varepsilon) &= \{\varepsilon\} \\
\Lambda(x) &= \emptyset && \forall x \in A \\
\Lambda(E + F) &= \Lambda(E) \cup \Lambda(F) && \forall E, F \in \operatorname{Rat} A^* \\
\Lambda(E \cdot F) &= \Lambda(E) \cdot \Lambda(F) \\
\Lambda(E^*) &= \{\varepsilon\}
\end{aligned}
$$

Lettres en tête d'une expression régulière

$$
\begin{aligned}
P(\emptyset) &= \emptyset \\
P(\varepsilon) &= \emptyset \\
P(x_k) &= \{k\} && \forall x \in A \\
P(E + F) &= P(E) \cup P(F) && \forall E, F \in \operatorname{Rat} A^* \\
P(E \cdot F) &= P(E) \cup \Lambda(E) P(F) \\
P(E^*) &= P(E)
\end{aligned}
$$

x_k : la lettre $x \in A$ apparaissant en position k

Lettres en queue d'une expression régulière

$$
\begin{aligned}
D(\emptyset) &= \emptyset \\
D(\varepsilon) &= \emptyset \\
D(x_k) &= \{k\} && \forall x \in A \\
D(E + F) &= D(E) \cup D(F) && \forall E, F \in \operatorname{Rat} A^* \\
D(E \cdot F) &= D(F) \cup D(E)\Lambda(F) \\
D(E^*) &= D(E)
\end{aligned}
$$

x_k : la lettre $x \in A$ apparaissant en position k

Paires de lettres d'une expression régulière

$$
\begin{aligned}
T(\emptyset) &= \emptyset \\
T(\varepsilon) &= \emptyset \\
T(x_k) &= \emptyset && \forall x \in A \\
T(E + F) &= T(E) \cup T(F) && \forall E, F \in \operatorname{Rat} A^* \\
T(E \cdot F) &= T(E) \cup T(F) \cup D(E) \times P(F) \\
T(E^*) &= T(E) \cup D(E) \times P(E)
\end{aligned}
$$

x_k : la lettre $x \in A$ apparaissant en position k

La **méthode de Brzozowski** construit l'**automate des quotients** d'un **langage rationnel** L à partir de la **dérivation** des **expressions rationnelles**.

La **dérivée** $\frac{\partial}{\partial x}E$, par rapport à une lettre $x \in A$, d'une expression E, calculant un langage $L \in \mathrm{Rat}\, A^*$, est définie inductivement. Elle calcule le langage quotient $x^{-1}L$.

La notion de **dérivée** est étendue aux mots $u \in A^*$ inductivement, pour toute expression E et lettre $x \in A$, par :

$$\frac{\partial}{\partial \varepsilon}E = E \qquad \frac{\partial}{\partial u \cdot x}E = \frac{\partial}{\partial x}\left(\frac{\partial}{\partial u}E\right)$$

Notons $c(E) = \varepsilon$ si $\varepsilon \in L_E$ et $c(E) = \emptyset$ sinon.

Dérivée de Brzozowski

$$\frac{\partial}{\partial x}\emptyset = \frac{\partial}{\partial x}\varepsilon = \frac{\partial}{\partial x}y = \emptyset \qquad \forall x, y \in A, x \neq y$$

$$\frac{\partial}{\partial x}x = \varepsilon$$

$$\frac{\partial}{\partial x}(E + F) = \frac{\partial}{\partial x}E + \frac{\partial}{\partial x}F \qquad \forall E, F \in \operatorname{Rat} A^*$$

$$\frac{\partial}{\partial x}(E \cdot F) = \left(\frac{\partial}{\partial x}E\right) \cdot F + c(E) \cdot \frac{\partial}{\partial x}F$$

$$\frac{\partial}{\partial x}E^* = \left(\frac{\partial}{\partial x}E\right) \cdot E^*$$

Identitées rationnelles

Quelques identitées bien utiles. Pour toutes expressions rationnelles E, F, G, on vérifie :

$$E \cdot \emptyset \equiv \emptyset \equiv \emptyset \cdot E \equiv \emptyset \quad E \cdot \varepsilon \equiv \varepsilon \cdot E \equiv E \qquad (\mathbf{T}_p)$$

$$E + \emptyset \equiv \emptyset + E \equiv E \qquad (\mathbf{T}_s)$$

$$(E + F) + G \equiv E + (F + G) \qquad (\mathbf{A}_s)$$

$$E + F \equiv F + E \qquad (\mathbf{C})$$

$$E + E \equiv E \qquad (\mathbf{I})$$

où $E \equiv F$ si les langages associés à E et F sont égaux.

Théorème L'ensemble des **dérivées** d'une expression rationnelle est **fini** modulo les identitées $\mathbf{T}_p$, $\mathbf{T}_s$, $\mathbf{A}_s$, **C** et **I**.

Les identitées sont faciles à maintenir dans une structure de données qui représente une expression rationnelles : il suffit de représenter les sommes comme des ensembles et de veiller à simplifier les expressions avec $\emptyset$ et ε.

On obtient un **AFD** pour une expression en construisant l'automate dont les états sont les dérivées et les transitions sont du type :

$$E \xrightarrow{x} \frac{\partial}{\partial x} E \qquad \forall x \in A$$

Les états acceptants sont les expressions E pour lesquelles $c(E) \neq \emptyset$.