Automates et Langages

Première partie : Automates finis

Nathalie Bertrand
nathalie.bertrand@inria.fr

Prépa agreg 2010/2011

Bibliographie

Aut Jean-Michel Autebert.
Théorie des langages et des automates.
Masson, 1994.

BBC Jean Berstel, Danièle Beauquier et Philippe Chrétienne.
Éléments d'algorithmique.
Masson, 1992.

Car Olivier Carton.
Langages formels, calculabilité et complexité.
Vuibert, 2008.

HU John E. Hopcroft et Jeffrey D. Ullman.
Introduction to automata theory, languages and computation.
Addison-Wesley, 1979.

Saka Jacques Sakarovitch.
Éléments de théorie des automates.
Vuibert informatique, 2003.

Mots et langages

Alphabet : ensemble fini de symboles (lettres).
Mot sur l'alphabet Σ = suite (finie) de lettres : $w = a_1 \cdots a_n$.
On note ε le mot vide.

$w \cdot w'$ est la concaténation de w et w' mots sur Σ.
L'ensemble des mots sur Σ, muni de $\cdot$, est un monoïde, noté Σ^*.

Un langage est une partie de Σ^* : $L \subseteq \Sigma^*$.
Opérations booléennes sur 2^{Σ^*} : union, intersection, complémentaire.
Concaténation : $L \cdot L' = \{w \cdot w' \mid w \in L,\ w' \in L'\}$.
Itération : $L^* = \bigcup_{n \geq 0} L^n$ où $L^0 = \{\varepsilon\}$, et $L^{i+1} = L^i \cdot L$.

Automates finis

Automate fini

$\mathcal{A} = (Q, \delta, I, F)$ où

- Q ensemble fini d'états,
- $I \subseteq Q$ ensemble d'états initiaux,
- $F \subseteq Q$ ensemble d'états finaux,
- $\delta : Q \times \Sigma \to 2^Q$ fonction de transition.

Pour $q' \in \delta(q, a)$, on note $q \xrightarrow{a} q'$.

Exécution de $\mathcal{A}$ sur un mot $w = a_1 \cdots a_n$:

$$q_0 \xrightarrow{a_1} q_1 \cdots q_{n-1} \xrightarrow{a_n} q_n.$$

Extension de δ à $Q \times \Sigma^*$:
$\delta(q, \varepsilon) = q$ and $\delta(q, w \cdot a) = \delta(\delta(q, w), a)$ pour $w \in \Sigma^*$ et $a \in \Sigma$.

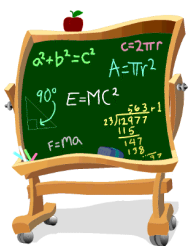
Langages reconnaissables

Langage accepté

$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \exists q_0 \in I,\ \delta(q_0, w) \cap F \neq \emptyset\}$

Langages reconnaissables

Un langage $L \subseteq \Sigma^*$ est reconnaissable s'il existe un automate fini $\mathcal{A}$ tel que $L = \mathcal{L}(\mathcal{A})$.
On note $\mathrm{Rec}(\Sigma^*)$ la famille des langages reconnaissables sur Σ.



Exemples d'automates.

Il existe des langages non reconnaissables (argument de cardinalité).
Quelques exemples :

- $\{a^n b^n \mid n \geq 0\}$
- $\{w \in \Sigma^* \mid |w|_a = |w|_b\}$

Automates déterministes

Automate déterministe

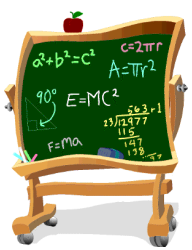
$\mathcal{A} = (Q, \delta, I, F)$ est déterministe si

- $|I| = 1$, et
- $|\delta(q, a)| \leq 1$ pour tout $q \in Q$, $a \in \Sigma$.

À partir d'un état, il y a au plus un calcul sur un mot donné.

Déterminisation

Soit $\mathcal{A}$ un automate fini. On peut construire un automate déterministe $\mathcal{B}$ reconnaissant le même langage que $\mathcal{A}$.



Automate des parties.

Automates complets/émondés

Automate complet. Automate émondé

- $\mathcal{A} = (Q, \delta, I, F)$ est complet si $\forall q \in Q,\ \forall a \in \Sigma,\ \delta(q, a) \neq \emptyset$.
- $\mathcal{A} = (Q, \delta, I, F)$ est émondé si $\forall q \in Q$
  - $\exists i \in I,\ \exists w \in \Sigma^*$ avec $q \in \delta(i, w)$ (q est accessible depuis i)
  - $\exists f \in F,\ \exists w \in \Sigma^*$ avec $f \in \delta(q, w)$ (q est co-accessible de f).

Calcul itératif des (co-)accessibles.

Pour tout automate fini, on peut construire un automate complet (resp. émondé) qui reconnaît le même langage.

Problème du vide

Soit $\mathcal{A}$ un automate fini. On peut décider si $\mathcal{L}(\mathcal{A}) = \emptyset$.

Automates avec ε-transitions

Automate avec ε-transitions

$\mathcal{A} = (Q, \delta, I, F)$ où

- Q ensemble fini d'états, $I \subseteq Q$ états initiaux, $F \subseteq Q$ états finaux,
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \longrightarrow 2^Q$ fonction de transition.

$\pi_\Sigma : (\Sigma \cup \{\varepsilon\})^* \longrightarrow \Sigma^*$ projection sur l'alphabet Σ.

Une exécution $q_0 \xrightarrow{a_1} q_1 \cdots q_{n-1} \xrightarrow{a_n} q_n$ lit $\pi_\Sigma(a_1 \cdots a_n)$.

Élimination des ε-transitions

Pour tout automate avec ε-transitions, on peut construire un automate sans ε-transitions reconnaissant le même langage.

Problèmes du vide et du mot

Problèmes du vide et du mot

- Problème du vide : étant donné $\mathcal{A}$, décider si $\mathcal{L}(\mathcal{A}) = \emptyset$.
- Problème du mot : étant donnés $\mathcal{A}$ et $w \in \Sigma^*$, décider si $w \in \mathcal{L}(\mathcal{A})$.

Théorème

Les problèmes du vide et du mot sont décidables en NLOGSPACE.

Rq : peu importe que $\mathcal{A}$ soit déterministe ou non, avec ou sans ε-transitions.

Propriétés de clôture

Opérations ensemblistes [BBC,p.301]

La famille Rec(Σ^*) des langages reconnaissables sur Σ est close par les opérations ensemblistes (union, intersection, complément).

Corollaires

- L'égalité et l'inclusion de langages reconnaissables sont décidables.
- L est reconnaissable si et seulement si $L \setminus \{\varepsilon\}$ est reconnaissable.

Concaténation et itération [BBC,p.303]

La famille Rec(Σ^*) est close par concaténation et itération.

Propriétés de clôture (2)

Pour $L \subseteq \Sigma^*$, on définit :

- $\text{Pref}(L) = \{w \in \Sigma^* \mid \exists w' \in \Sigma^*,\ w \cdot w' \in L\}$,
- $\text{Suff}(L) = \{w \in \Sigma^* \mid \exists w' \in \Sigma^*,\ w' \cdot w \in L\}$,
- $\text{Fact}(L) = \{w \in \Sigma^* \mid \exists w_i, w_f \in \Sigma^*,\ w_i \cdot w \cdot w_f \in L\}$.

Préfixe, suffixe, facteur

La famille $\text{Rec}(\Sigma^*)$ est close par préfixe, suffixe, facteur.

Pour $L, K \subseteq \Sigma^*$, on définit :

- $K^{-1} \cdot L = \{w \in \Sigma^* \mid \exists k \in K,\ k \cdot w \in L\}$,
- $L \cdot K^{-1} = \{w \in \Sigma^* \mid \exists k \in K,\ w \cdot k \in L\}$.

Quotient

La famille $\text{Rec}(\Sigma^*)$ est close par quotients :

$$L \in \text{Rec}(\Sigma^*) \text{ et } K \subseteq \Sigma^* \Rightarrow (K^{-1} \cdot L) \in \text{Rec}(\Sigma^*) \text{ et } (L \cdot K^{-1}) \in \text{Rec}(\Sigma^*).$$

Propriétés de clôture (3)

Morphisme (de monoïdes)

$\varphi : \Sigma_1^* \to \Sigma_2^*$ est un morphisme si $\forall w_1, w_1' \in \Sigma_1^*$, $\varphi(w_1 \cdot w_1') = \varphi(w_1) \cdot \varphi(w_1')$.

Morphisme et morphisme inverse

La famille $\mathrm{Rec}(\Sigma^*)$ est close par morphisme et morphisme inverse.

Propriétés de clôture (4)

Substitution

Une substitution est une fonction $\psi : \Sigma_1 \to \mathcal{P}(\Sigma_2^*)$. ψ s'étend en un morphisme $\psi : \Sigma_1^* \to \mathcal{P}(\Sigma_2^*)$ défini par $\psi(\varepsilon) = \{\varepsilon\}$ et $\psi(wa) = \psi(w)\psi(a)$.
Une substitution est rationnelle si elle est définie par $\psi : \Sigma_1 \to \mathrm{Rec}(\Sigma_2^*)$.

Pour $L \subseteq \Sigma_1^*$, $\psi(L) = \cup_{w \in L}\ \psi(w)$.
Pour $L \subseteq \Sigma_2^*$, $\psi^{-1}(L) = \{w \in \Sigma_1^* \mid \psi(w) \cap L \neq \emptyset\}$.

Substitution rationnelle et substitution rationnelle inverse

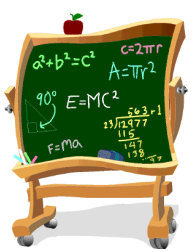
La famille $\mathrm{Rec}(\Sigma^*)$ est close par substitution rationnelle et substitution rationnelle inverse.

Lemmes d'itération

Lemmes de l'étoile [Car,p.53][Saka,p.78]

Soit $L \in \mathrm{Rec}(\Sigma^*)$. Alors il existe $N \in \mathbb{N}$ tel que pour tout $w \in L$

1. si $|w| \geq N$ alors il existe une factorisation $w = w_1 w_2 w_3$ avec $w_2 \neq \varepsilon$ telle que $w_1 w_2^* w_3 \subseteq L$.
2. pour toute factorisation $w = u_1 u_2 u_3$ avec $|u_2| \geq N$, il existe une factorisation $u_2 = w_1 w_2 w_3$ avec $w_2 \neq \varepsilon$ telle que $u_1 w_1 w_2^* w_3 u_3 \subseteq L$.
3. pour toute factorisation $w = u w_1 w_2 \cdots w_N v$ où $w_i \neq \varepsilon$, il existe $0 \leq j < k \leq N$ tel que $u w_1 \cdots w_j (w_{j+1} \cdots w_k)^* w_{k+1} \cdots w_N v \subseteq L$.



Applications des lemmes de l'étoile, exemples de non-reconnaissables.

Caractérisation

Théorème (Ehrenfeucht, Parikh, Rozenberg) [Saka,p.128][Car,p.54]

Soit $L \subseteq \Sigma^*$. Les propriétés suivantes sont équivalentes :

(i) L est reconnaissable ;

(ii) il existe $N \in \mathbb{N}$ tel que pour tout mot $w \in \Sigma^*$ et toute factorisation $w = uw_1 \cdots w_N v$ avec $w_i \neq \varepsilon$, il existe $0 \leq j < k \leq N$ tels que :

$$\forall n \in \mathbb{N} \quad w \in L \iff uw_1 \cdots w_j (w_{j+1} \cdots w_k)^n w_{k+1} \cdots w_N \in L$$

(iii) il existe $N \in \mathbb{N}$ tel que pour tout mot $w \in \Sigma^*$ et toute factorisation $w = uw_1 \cdots w_N v$ avec $w_i \neq \varepsilon$, il existe $0 \leq j < k \leq N$ tels que :

$$w \in L \iff uw_1 \cdots w_j w_{k+1} \cdots w_N \in L$$

Corollaire

L est reconnaissable ssi L et $\bar{L}$ satisfont le lemme de l'étoile par bloc (3.).

Expressions rationnelles

Expressions rationnelles

L'ensemble des expressions rationnelles est défini par :

- $\emptyset$ et a, pour $a \in \Sigma$, sont des expressions rationnelles ;
- si E et F sont des expressions rationnelles, alors $(E + F)$, $(E \cdot F)$ et (E^*) aussi.

On note $\mathcal{E}_\Sigma$ l'ensemble des expressions rationnelles.

Sémantique

Le langage dénoté par une expression rationnelle est défini inductivement :

- $[\![\emptyset]\!] = \emptyset$, et $[\![a]\!] = \{a\}$, pour tout $a \in \Sigma$;
- $[\![(E + F)]\!] = [\![E]\!] \cup [\![F]\!]$, $[\![(E \cdot F)]\!] = [\![E]\!] \cdot [\![F]\!]$ et $[\![(E^*)]\!] = [\![E]\!]^*$.

Langages rationnels

Langage rationnel

$L \subseteq \Sigma^*$ est rationnel s'il existe $E \in \mathcal{E}_\Sigma$ tel que $L = [\![E]\!]$.
On note $\mathrm{Rat}(\Sigma^*)$ la famille des langages rationnels sur Σ.

$\mathrm{Rat}(\Sigma^*)$ est la plus petite partie de 2^{Σ^*} contenant $\emptyset$ et $\{a\}$, pour $a \in \Sigma$, et fermée par union, concaténation et itération.

Équivalence d'expressions rationnelles

$E, F \in \mathcal{E}_\Sigma$ sont équivalentes, noté $E \equiv F$, si $[\![E]\!] = [\![F]\!]$.

Théorème de Kleene

Théorème (Kleene)

$$\mathrm{Rec}(\Sigma^*) = \mathrm{Rat}(\Sigma^*)$$

Preuve

($\supseteq$) $\mathrm{Rec}(\Sigma^*)$ contient $\emptyset$, $\{a\}$ (pour $a \in \Sigma$) et est close par union, concaténation et étoile.
Construction : algo de Thompson, Glushkov, et Antimirov.

($\subseteq$) Construction : algo de McNaughton-Yamada, Brzozowski-McCluskey, et par résolution d'équations.

Corollaire

L'équivalence des expressions rationnelles est décidable.

Des automates aux expressions : McNaughton-Yamada [HU, p.33] [Car, p.38]

Soit $\mathcal{A} = (Q, \delta, I, F)$ un automate fini, avec $Q = \{1, \cdots, n\}$.
On définit, pour $k \in \{0, \cdots, n\}$ et $p, q \in Q$

$$L_{p,q}^{(k)} = \{a_1 \cdots a_n \mid p \xrightarrow{a_1} p_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q \text{ avec } p_1, \cdots, p_{n-1} \in \{1, \cdots, k\}\}.$$

$$\mathcal{L}(\mathcal{A}) = \bigcup_{i \in I, f \in F} L_{i,f}^{(n)}.$$

Principe : calcul inductif d'une expression rationnelle pour les $L_{p,q}^{(k)}$.

Initialement,

$$L_{p,q}^{(0)} = \begin{cases} \sum_{(p,a,q) \in \delta} a + \varepsilon & \text{si } p = q, \\ \sum_{(p,a,q) \in \delta} a & \text{sinon.} \end{cases}$$

Induction :

$$L_{p,q}^{(k+1)} = L_{p,q}^{(k)} + L_{p,k+1}^{(k)} \cdot (L_{k+1,k+1}^{(k)})^* \cdot L_{k+1,q}^{(k)}.$$

Des automates aux expressions : Brzozowski-McCluskey [Saka, p.105] [Car, p.39]

Principe : élimination d'états en utilisant des automates généralisés.

Automate généralisé

Un automate généralisé sur Σ est un automate fini sur $\mathrm{Rat}(\Sigma^*)$.

Les étiquettes des transitions sont des expressions rationnelles.

Proposition

Si $\mathcal{A}$ est un automate généralisé sur Σ, alors $\mathcal{L}(\mathcal{A}) \in \mathrm{Rat}(\Sigma^*)$.

Preuve : construction d'un automate généralisé équivalent
$\mathcal{B} = (\{q_0, q_f\}, \delta_{\mathcal{B}}, \{q_0\}, \{q_f\})$ où $\delta_{\mathcal{B}} = \{(q_0, \mathcal{L}(\mathcal{A}), q_f)\}$.

Des automates aux expressions : résolution d'équations

Soit $\mathcal{A} = (Q, \delta, I, F)$ un automate fini.

Pour $p \in Q$, X_p est le langage des mots acceptés avec p comme état initial.

$$X_p = \begin{cases} \sum_{(p,a,q)\in\delta} aX_q + \varepsilon & \text{si } p \in F \\ \sum_{(p,a,q)\in\delta} aX_q & \text{sinon.} \end{cases}$$

$\mathcal{L}(\mathcal{A}) = \sum_{i\in I} X_i$

Principe : Résolution du système d'équations linéaires par élimination gaussienne.

Lemme (Arden) [Car,p.40][Saka, p.108]

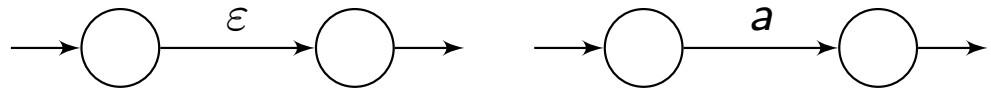
Soient $K, L \subseteq \Sigma^*$, avec $\varepsilon \notin K$. Alors K^*L est l'unique solution de l'équation $X = KX + L$.

Des expressions aux automates : Thompson

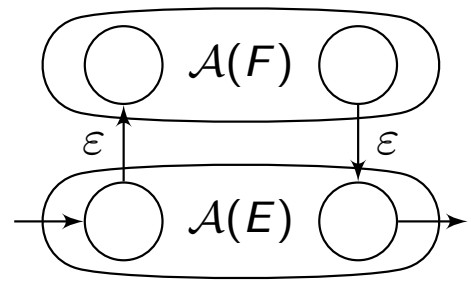
[HU, p.30],[Saka, p.157]

Principe : Construction d'un automate $\mathcal{A}(E)$ par induction structurelle sur E.
Propriétés des automates : un unique état initial sans transition entrante, et un unique état final sans transition sortante.

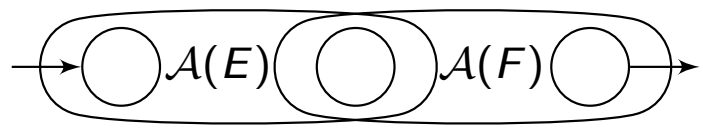
- $\mathcal{A}(\{\varepsilon\})$ et $\mathcal{A}(\{a\})$



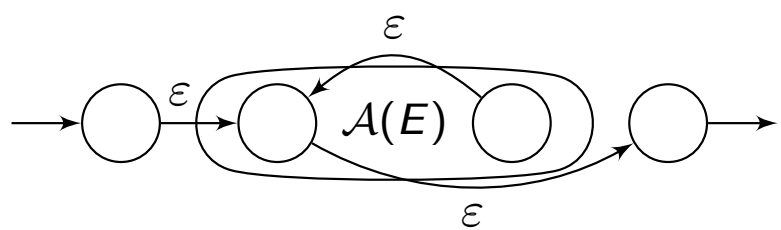
- Somme : $\mathcal{A}(E + F)$



- Concaténation : $\mathcal{A}(E \cdot F)$



- Étoile : $\mathcal{A}(E^*)$



Des expressions aux automates : Glushkov

Expression linéaire

$E \in \mathrm{Rat}(\Sigma^*)$ est linéaire si chaque symbole de Σ apparaît au plus une fois.

Automate local

$\mathcal{A} = (Q, \delta, \{i\}, F)$ déterministe est local si pour tout $a \in \Sigma$, $|\{q \in Q \mid \exists p \in Q,\ (p, a, q) \in \delta\}| \leq 1$.

Proposition

Pour toute expression linéaire E sur Σ, on peut construire un automate local $\mathcal{A}$ tel que $\mathcal{L}(\mathcal{A}) = [\![E]\!]$.

Des expressions aux automates : Antimirov

[Saka, p.159]

Dérivée partielle

Soient $E \in \mathrm{Rat}(\Sigma)$ et $a \in \Sigma$. La dérivée partielle $\partial_a(E)$ de E par rapport à a est l'ensemble des expressions rationnelles défini inductivement par :

- $\partial_a(\emptyset) = \emptyset$
- $\partial_a(a) = \{\varepsilon\}$ et $\partial_a(b) = \emptyset$ pour $b \neq a$
- $\partial_a(E + F) = \partial_a(E) \cup \partial_a(F)$
- $\partial_a(E^*) = \partial_a(E) \cdot E^*$
- $\partial_a(E \cdot F) = \begin{cases} \partial_a(E) \cdot F & \text{si } \varepsilon \notin [\![E]\!] \\ \partial_a(E) \cdot F \cup \partial_a(F) & \text{sinon.} \end{cases}$

Extension à des mots de Σ^* : $\partial_\varepsilon(E) = \{E\} \quad \partial_{wa}(E) = \partial_a(\partial_w(E))$.

Automate des dérivées : $\mathcal{A} = (Q, \delta, I, F)$ où
$Q = \{E_1 \mid \exists w \in \Sigma^*,\ E_1 \in \partial_w(E)\}$
$I = \{E\} \quad F = \{E_1 \mid \varepsilon \in [\![E_1]\!]\} \quad \delta = \{(E_1, a, E_2) \mid E_2 \in \partial_a(E_1)\}$

Propriétés

L'automate des dérivées est un automate fini et satisfait : $\mathcal{L}(\mathcal{A}) = [\![E]\!]$.

Automate des résiduels

Résiduel

Le résiduel de $L \subseteq \Sigma^*$ par $u \in \Sigma^*$ est le quotient $u^{-1}L = \{v \in \Sigma^* \mid uv \in L\}$.

Automate des résiduels

Soit $L \subseteq \Sigma^*$. L'automate des résiduels de L est $R(L) = (Q_L, \delta_L, \{i_L\}, F_L)$ avec :

- $Q_L = \{u^{-1}L \mid u \in \Sigma^*\}$,
- $\delta_L(u^{-1}L, a) = a^{-1}(u^{-1}L) = (ua)^{-1}L$,
- $i_L = L = \varepsilon^{-1}L$,
- $F_L = \{u^{-1}L \mid \varepsilon \in u^{-1}L\} = \{u^{-1}L \mid u \in L\}$.

Proposition [BBC, p.312]

L est reconnaissable ssi L a un nombre fini de résiduels.

Morphismes d'automates

Morphisme d'automates

Soient $\mathcal{A}_1 = (Q_1, \delta_1, I_1, F_1)$ et $\mathcal{A}_2 = (Q_2, \delta_2, I_2, F_2)$ deux automates sur Σ. Un morphisme $\varphi : \mathcal{A}_1 \to \mathcal{A}_2$ est une application de Q_1 dans Q_2 telle que :

- $\varphi(I_1) \subseteq I_2$,
- $\varphi(F_1) \subseteq F_2$,
- $\forall p, q \in Q_1 \quad q \in \delta_1(p, a) \Rightarrow \varphi(q) \in \delta_2(\varphi(p), a)$.

Proposition

Soient $\mathcal{A}_1 = (Q_1, \delta_1, \{i_1\}, F_1)$ et $\mathcal{A}_2 = (Q_2, \delta_2, \{i_2\}, F_2)$ deux automates déterministes complets. Si $\varphi : \mathcal{A}_1 \to \mathcal{A}_2$ est un morphisme d'automates surjectif (*i.e.* $\varphi^{-1}(F_2) = F_1$ et $\varphi(Q_1) = Q_2$), alors $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$.

Proposition [Saka, p.122]

Soient $\mathcal{A} = (Q, \delta, \{i\}, F)$ un automate déterministe complet pour L, et $R(L) = (Q_L, \delta_L, i_L, F_L)$ l'automate des résiduels associé à L. Alors $\varphi : Q \to Q_L$ défini par $\varphi(q) = L_q = \{w \in \Sigma^* \mid \delta(q, w) \in F\}$ est un morphisme surjectif.

Automate minimal

Quotient d'automates

Soient $\mathcal{A}_1$ et $\mathcal{A}_2$ deux automates déterministes complets. On dit que $\mathcal{A}_2$ est un quotient de $\mathcal{A}_1$, noté $\mathcal{A}_2 \preceq \mathcal{A}_1$, s'il existe un morphisme surjectif $\varphi : \mathcal{A}_1 \to \mathcal{A}_2$.

$\preceq$ est un ordre partiel sur les automates déterministes complets.

Minimalité de l'automate des résiduels

Soit $L \in \mathrm{Rec}(\Sigma^*)$

- $R(L)$ est minimal pour l'ordre quotient $\preceq$ parmi les automates déterministes complets reconnaissant L
- Tout automate déterministe complet reconnaissant L avec un nombre minimal d'états est isomorphe à $R(L)$.

Algorithme par renversement

[Saka, p.125]

Proposition

Soit $L \in \mathrm{Rec}(\Sigma^*)$. Le déterminisé d'un automate co-déterministe co-accessible qui reconnaît L est minimal.

Soit $\mathcal{A} = (Q, \delta, I, F)$. Le transposé de $\mathcal{A}$ est $\mathrm{tr}(\mathcal{A}) = (Q, \delta^t, F, I)$ avec :

$$\delta^t(p, a) = \{q \mid p \in \delta(q, a)\}.$$

Algorithme de Brzozowski

$$R(\mathcal{L}(\mathcal{A})) = \det(\mathrm{tr}(\det(\mathrm{tr}(\mathcal{A})))).$$

Congruence

Congruence

Soit $\mathcal{A} = (Q, \delta, \{i\}, F)$ un automate déterministe. Une relation d'équivalence $\sim$ sur Q est une congruence si :

- elle est compatible avec δ $\quad \forall p, q \quad p \sim q \Rightarrow \forall a \in \Sigma,\ \delta(p, a) \sim \delta(q, a)$
- elle sature F $\quad \forall p, q \quad p \sim q \Rightarrow p \in F$ ssi $q \in F$

Automate quotient

Soit $\mathcal{A} = (Q, \delta, \{i\}, F)$ un automate déterministe et $\sim$ une relation d'équivalence sur Q. Le quotient de $\mathcal{A}$ par $\sim$ est l'automate

$$\mathcal{A}/\sim = (Q/\sim, \delta_\sim, \{[i]\}, \{[f] \mid f \in F\}) \quad \text{où } \delta_\sim([p], a) = [\delta(p, a)].$$

Rq : $\mathcal{A}/\sim$ est déterministe et complet.

Proposition [Car, p. 47]

Soit $\mathcal{A} = (Q, \delta, \{i\}, F)$ un automate déterministe et $\sim$ une congruence sur Q. Alors $\mathcal{L}(\mathcal{A}/\sim) = \mathcal{L}(\mathcal{A})$.

Congruence de Nerode

Congruence de Nerode

Soit $\mathcal{A} = (Q, \delta, \{i\}, F)$. L'équivalence sur Q définie par

$$\begin{aligned} p \equiv q \quad & \text{ssi} \quad L_p = L_q \\ & \text{ssi} \quad \forall w \in \Sigma^*,\ \delta(p, w) \in F \Leftrightarrow \delta(q, w) \in F \end{aligned}$$

est appelée congruence de Nerode.

Rq : le quotient $\mathcal{A}/\equiv$ est isomorphe à $R(L)$.

Conséquences

Soit $L \in \mathrm{Rec}(\Sigma^*)$.

- On peut calculer l'automate minimal de L à partir d'un automate déterministe pour L en utilisant l'équivalence de Nerode.
- L'égalité de langages reconnaissables peut être décidée en comparant leurs automates minimaux respectifs.

Algorithme de Moore

Principe : Calcul de la congruence de Nerode, par raffinements successifs.

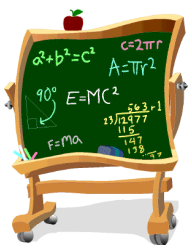
Soit $\mathcal{A} = (Q, \delta, \{i\}, F)$ un automate déterministe.
On définit la congruence $\sim_i$ sur Q par :

$$p \sim_0 q \quad \text{ssi} \quad (p \in F \Leftrightarrow q \in F)$$
$$p \sim_{i+1} q \quad \text{ssi} \quad p \sim_i q \text{ et } \forall a \in \Sigma,\ \delta(p,a) \sim_i \delta(q,a)$$

Proposition [Aut, p.61],[Saka, p.124]

Soit $\mathcal{A} = (Q, \delta, \{i\}, F)$ un automate déterministe complet.

- $p \sim_i q$ ssi $L_p \cap \Sigma^{\leq i} = L_q \cap \Sigma^{\leq i}$
- Si $\sim_{i+1} = \sim_i$ alors $\sim_i = \equiv$
- $\equiv = \sim_{|Q|-2}$.



Optimisation : algorithme d'Hopcroft, utilisant diviser pour régner.
Calcul de l'équivalence de Nerode en $\mathcal{O}(n \log(n))$. [BBC, p.321]

Recherche de motif

[Saka, p.163]

But : recherche d'un mot fini $w \in \Sigma^*$ dans un texte t sur Σ.
Applications : grep, recherche de séquences ADN

- Entrée : $w = w_1 \cdots w_m$ et $t = t_1 \cdots t_n$
- Sortie : k tel que $w_1 \cdots w_m = t_{k+1} \cdots t_{k+m}$

Algorithme naïf

```
i:=1 ; j:=1 ;
tant que i ≤ m et j ≤ n faire
   si t[j] = w[i] alors i:=i+1 ; j:=j+1
   sinon j:=j-i+2 ; i:=1
si i > m alors
   retourner 'occurrence à la position j-m'
sinon
   retourner 'pas d'occurrence'
```

Recherche de motif : Algorithme de Morris et Pratt

Bord

Soit $v \in \Sigma^*$. Le bord de v est le plus long sous-mot strict de v qui est à la fois préfixe et suffixe de v.

On définit $\beta : \{0, \cdots, m\} \to \{-1, \cdots, m-1\}$ par :

$$\beta(0) = -1 \qquad \beta(i) = |\text{bord}(w_1 \cdots w_i)|$$

Algorithme de Morris et Pratt [BBC, p.340]

```
i:=1 ; j:=1 ;
tant que i ≤ m et j ≤ n faire
   si i ≥ 1 et t[j] ≠ w[i] alors i:=1 + β(i-1)
   sinon i:=i+1 ; j:=j+1
si i > m alors
   retourner 'occurrence à la position j-m'
sinon
   retourner 'pas d'occurrence'
```

Recherche de motif : Automate du motif

Automate d'un motif

L'automate du motif w est $\mathcal{A}_w = (\text{Pref}(w), \delta, \{\varepsilon\}, \{w\})$ où :

$$\delta(u, a) = \begin{cases} ua & \text{si } ua \text{ préfixe de } w \\ \text{bord}(ua) & \text{sinon.} \end{cases}$$

Proposition

L'automate du motif w est l'automate minimal de $\Sigma^* w$: $\mathcal{A}_w = R(\Sigma^* w)$.

Analyse lexicale

[HU, p.45]

Première étape de la chaîne de compilation.
But : Transformer une suite de caractères en une suite de mots, les lexèmes.

L'analyseur lexical reconnaît les composants élémentaires de la syntaxe.

Quelques expressions rationnelles pour l'analyse lexicale de `C` par `flex`:

mots clefs $E_{\texttt{if}} = \texttt{if} \quad E_{\texttt{int}} = \texttt{int}$

symboles $E_{=} = \texttt{=} \quad E_{+} = \texttt{+} \quad E_{;} = \texttt{;}$

blancs $E_{\texttt{space}} = \texttt{[\textbackslash n\textbackslash t]+}$

identifiants $E_{\texttt{id}} = [a-zA-Z][0-9a-zA-Z]*$

entiers $E_{\texttt{ic}} = \texttt{[1-9] [0-9]* | 0[0-1]*}$

```
int a = 12;        'int' 'space' 'id' 'space' '=' 'space' 'ic' ';' 'space'
int b = 3 + a;     'int' 'space' 'id' 'space' '=' 'space' 'ic' '+' 'space' 'id' ';'
```

Deux principes :

- Plus long préfixe : `12` ne constitue qu'un lexème '`ic`'.
- Priorités : `int` pourrait être un identifiant, mais '`int`' est prioritaire.

Analyse lexicale : Algorithme naïf

Principe : simuler un automate déterministe pour le langage $L(\sum_i E_i)$.
$\mathcal{A} = (Q, \delta, \{q_0\}, F)$; $f : F \to E_1, \cdots, E_p$; $w = a_1 \cdots a_n$

Analyse lexicale naïve

```
q:=q0 ; i:=1 ;
tant que i ≤ n faire
   qf:=⊥;
   tant que i ≤ n et δ(q,ai) ≠ ∅
      si q ∈ F alors
         qf:=q ; j:=i ;
      i:=i+1 ;
   si qf=⊥ alors
      retourner 'échec';
   afficher f(qf) ; i:=j ;
retourner 'succès';
```

Analyse lexicale : Programmation dynamique

Principe : trouver un facteur maximal à partir d'un état q de l'automate depuis la position i du mot w en utilisant d'autres calculs pour q' et $i+1$.

$$T(q,i) = \begin{cases} (\bot, 0) & \text{si } \delta(q,a_i) = \emptyset \text{ et } q \notin F \\ (q,i) & \text{si } \delta(q,a_i) = \emptyset \text{ et } q \in F \\ (q,i) & \text{si } \delta(q,a_i) = q',\ T(q',i+1) = (\bot,0) \text{ et } q \in F \\ T(q',i+1) & \text{sinon.} \end{cases}$$

$$T(q,n+1) = \begin{cases} (\bot, 0) & \text{si } q \notin F \\ (q,n+1) & \text{sinon.} \end{cases}$$

Analyse lexicale

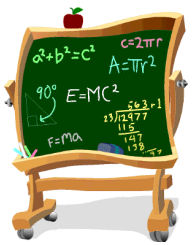
```
i:=1 ;
tant que i ≤ n faire
   (qf,i):=T(q0,i);
   si qf=⊥ alors
      retourner 'échec';
   afficher f(qf) ;
retourner 'succès';
```

Classification

Problème de séparation par automate.
Étant donnés $S, T \subseteq \Sigma^*$ finis, et $k \in \mathbb{N}$, existe-t-il un automate déterministe $\mathcal{A}$ à k états tel que $S \subseteq \mathcal{L}(\mathcal{A})$ et $T \cap \mathcal{L}(\mathcal{A}) = \emptyset$?

Théorème [FB, Ch.9]

Le problème de séparation par automate est NP-complet.



Preuve de la NP-difficulté par réduction de SAT.

Robert W. Floyd et Richard Biegel. The language of machines: An Introduction to Computability and Formal Languages. Computer Science Press, 1994.