

Leçon 32 : Langages rationnels et automates finis. Exemples et Applications.

Niveau : MP1

Prérequis : définition inductive, définition de base sur les langages

I - Langages réguliers

I.1 - Langages

Def 1 : Un langage sur un alphabet Σ est un ensemble de mots de Σ .

Exemple 2 : $\Sigma = \{0, 1\}$; \emptyset , $\{\epsilon, 101, 11\}$, $\{\epsilon\}$, $\{\text{mots de taille paire sur } \Sigma\}$ sont des langages sur Σ .

Def 3 : On prolonge les définitions ensemblistes d'union, d'intersection et de complémentaire sur les langages. On définit :

- * la concaténation de L_1 et L_2 : $L_1 \cdot L_2 = \{u \cdot v \mid u \in L_1, v \in L_2\}$
- * On définit inductivement L^n par $L^0 = \{\epsilon\}$ et $L^n = L \cdot L^{n-1}$
- * l'étoile de Kleene par $L^* = \bigcup_{n \geq 0} L^n$

I.2) Langages réguliers

Def 4 : On définit par induction l'ensemble des expressions régulières sur Σ par :

- $\emptyset, \epsilon, a \in \Sigma$ sont des expressions régulières
- $+, \cdot$ sont des constructeurs binaires
- $*$ est un constructeur unaire

Exemple 5 : $(0+1)^* \cdot 1 \cdot 0$ est une expression régulière.

Def 6 : Le langage $L(r)$ associé à une expression régulière r est défini inductivement par :

- $L(\emptyset) = \emptyset$; $L(\epsilon) = \{\epsilon\}$; $L(a) = \{a\} \forall a \in \Sigma$
- $L(r_1 + r_2) = L(r_1) \cup L(r_2)$
- $L(r_1 \cdot r_2) = L(r_1) \cdot L(r_2)$
- $L(r^*) = (L(r))^*$ On appelle $L(r)$ un langage régulier.

Exemple 7 : $L((0+1)^* \cdot 1 \cdot 0)$ est l'ensemble des nbr binaires multiples

de 2 est par de 4.

II - Automates

II.1) Automates non déterministes

Def 8 : Un automate fini non déterministe (NFA) est un 5-uplet $(Q, \Sigma, q_0, F, \delta)$ où :

- Q est un ensemble fini d'états
- Σ est un alphabet fini
- $q_0 \in Q$ est l'état initial
- $F \subset Q$ est l'ensemble des états finaux
- $\delta : Q \times \{\Sigma \times \epsilon\} \rightarrow \mathcal{P}(Q)$ une fonction de transition

Remarque 9 : $\delta(q, \epsilon)$ est une ϵ -transition.

Représentation 10 : les états sont des cercles, les transitions des flèches étiquetées. Une flèche entrante dans l'état initial, sortante pour $q \in F$.

Exemple 11 :
Exercice 12 : Donner l'automate $(Q, \Sigma, q_0, F, \delta)$ associé.

Définition 13 : Soit $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ un NFA. On appelle ϵ -fermeture de l'état q , noté $E(q)$, l'ensemble $\bigcup_{n \geq 0} \delta_{\epsilon}^n(\{q\})$ où

$$\delta_{\epsilon} : \mathcal{P}(Q) \rightarrow \mathcal{P}(Q)$$

$$A \mapsto \bigcup_{q \in A} \delta(q, \epsilon)$$

intuition : états accessibles depuis q en empruntant 0 ou plus ϵ -transitions.

Def 14 : On définit δ^* sur $Q \times \Sigma^*$ par induction sur le mot par :

$$* \delta^*(q, \epsilon) = E(q)$$

$$* \delta^*(q, wa) = E\left(\bigcup_{q' \in \delta^*(q, w)} \delta(q', a)\right) \text{ pour } w \in \Sigma^*, a \in \Sigma.$$

Exemple 15 : $\delta^*(0, 10) = \{0, 2\}$ sur l'exemple 11.

Def 16 : $w \in \Sigma^*$ est accepté par \mathcal{A} si $F \cap \delta^*(q_0, w) \neq \emptyset$. On note $L(\mathcal{A})$ le langage des mots reconnus par \mathcal{A} . On dit que $L(\mathcal{A})$ est reconnaissable.

Exemple 17 : Pour l'exemple 11, $L(\mathcal{A}) = \{w \in \{0, 1\}^* \mid w \text{ termine par } 10\}$

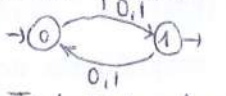
Def 18 : Soit \mathcal{A}_1 et \mathcal{A}_2 deux NFA sur Σ . On dit que \mathcal{A}_1 et \mathcal{A}_2 sont équivalents si $L(\mathcal{A}_1) = L(\mathcal{A}_2)$.

II.2 - Automates déterministes

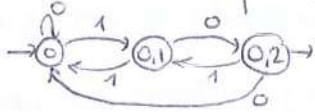
Def 19: Soit A un NFA. Si $\forall q \in Q, \forall a \in \Sigma, | \delta(q, a) | \leq 1$ et $\delta(q, \epsilon) = \emptyset$ alors on dit que A est un automate déterministe. ($\forall w \in \Sigma^*, \forall q \in Q, | \delta^*(q, w) | = 1$).

Def 20: Soit L un langage. S'il existe A un DFA tq $L(A) = L$, alors L est un langage rationnel.

Remarque 21: On peut, dans la définition d'un DFA, remplacer δ par une fonction partielle $\delta: Q \times \Sigma \rightarrow Q$.

Exemple 22:  DFA reconnaissant les mots de longueur impaire.

Théorème 23: Tout NFA est équivalent à un DFA.

Exemple 24:  Le DFA équivalent au NFA de l'exemple 11.

Remarque 25: Dans le pire des cas, déterminer un NFA crée un nbr exponentiel d'états.

Def 26: On dit qu'un DFA A est complet si $\forall q \in Q, \forall a \in \Sigma, | \delta(q, a) | = 1$.

Théorème 24: Tout DFA est équivalent à un DFA complet.

III - Vien automate fini et langages réguliers

III.1 - Equivalence

Théorème 25: (Kleene): Pour $L \subset \Sigma^*, L$ régulier $\Leftrightarrow L$ rationnel.

preuve: \Rightarrow Construction de Thompson

\Leftarrow : Algorithme d'élimination des états (Brozowski)

Corollaire: Les langages réguliers sont clos par complémentaire et intersection.

dem: en passant par les automates.

Remarque 26: L'algorithme de Thompson nous fournit un NFA avec beaucoup d' ϵ -transitions. On voudrait les éviter.

Développement 1: Construction de Thompson

Def 27: Soit $L \subset \Sigma^*$ un langage. On définit:

- $P(L) = \{ a \in \Sigma \mid a \cdot \Sigma^* \cap L \neq \emptyset \}$ // lettres commençant L

- $D(L) = \{ a \in \Sigma \mid \Sigma^* \cdot a \cap L \neq \emptyset \}$ // lettres terminant L

- $F(L) = \{ (a, b) \in \Sigma^2 \mid \Sigma^* \cdot ab \cdot \Sigma^* \cap L \neq \emptyset \}$ // facteurs de taille 2 de L

Remarque 28: L est local si il est déterminé par $P(L)$, $D(L)$ et $F(L)$.

Def 29: Soit $L \subset \Sigma^*$. On définit Loc_L le NFA $(Q_L, q_0, F_L, \delta_L)$ par:

- $Q_L = \{ q_a \mid a \in \Sigma \} \cup \{ q_0 \}$

- $F_L = \{ q_a \mid a \in D(L) \} \cup \{ q_0 \text{ si } \epsilon \in L \}$

- $\delta_L = \begin{cases} \delta_L(q_0, a) = \{ q_a \mid a \in P(L) \} \\ \delta_L(q_a, b) = \{ q_b \mid (ab) \in F(L) \} \end{cases}$

Théorème 30: Si L est local, alors $L = L(Loc_L)$

Algorithme 31: (Bertie Sethi): Entrée: r une expression régulière
Sortie: Un NFA sans ϵ -transition qui reconnaît $L(r)$.

- 1) Linéariser r en r' (en marquant chaque lettre pour la rendre unique)
- 2) Construire $Loc_{L(r')}$ (en calculant $P(L(r')), D(L(r')), \dots$)
- 3) Effacer les indices des symboles sur les transitions de $Loc_{L(r')}$

III.2 - Conséquences

Remarque 32: La première conséquence est de vérifier l'appartenance d'un mot à un langage régulier.

Théorème 33: (Lemme de l'étoile)

Soit L régulier. Alors $\exists n \in \mathbb{N}$ tq $\forall w \in L, \exists x, y, z: w = x \cdot y \cdot z$ et

i) $|x| \leq n$

ii) $|y| \geq 1$

iii) $\forall i \in \mathbb{N}, x y^i z \in L$.

preuve: On prend A le DFA tq $L(A) = L$, n son nombre d'état

Remarque 34: Ce lemme est surtout utilisé pour prouver la non rationalité d'un langage.

Exemple 35: Montrer que $\{a^n b^n \mid n \in \mathbb{N}\}$ n'est pas rationnel.

IV - Application

IV.1) Analyse lexicale

Les expressions régulières et les DFA interviennent dans la compilation des langages de programmation, au niveau de l'analyse lexicale.

Principe 36: Définir un ensemble d'expressions régulières appelées lexèmes et qu'on souhaite reconnaître. Un analyseur lexical génère les automates reconnaissant ces lexèmes et passe le code en parallèle dans ces automates, en gardant l'automate qui reconnaît le plus long préfixe, de plus haute priorité.

Exemple 37:

lexème 1 = $(0\dots9)^+ \cdot ' '(0\dots9)^*$ (flottant)

lexème 2 = $(0\dots9)^+$ (int)

lexème 3 = $\Sigma \setminus ' '$ (variables)

lexème 4 = if (IF)

lexème 5: then (THEN)

On reconnaît le code:

if (1) then 42,5 comme IF INT THEN FLOAT.

IV.2) Expression régulière POSIX

Motivation 38: grep 'reg-exp' fichier renvoie toutes les lignes de fichier dont une sous-chaîne est dans L(reg-exp)

Syntaxe 39: un caractère représente lui-même en général

- $|$ représente le +
- $*$ représente l'étoile de Kleene
- etc... (voir man grep)

IV.3) Reconnaissance de motif dans un texte

On a un texte t et un mot w . On veut savoir si w est un sous-mot de t .

Motivation 40: Ctrl + F

Solution naïve 41: $O(|t| \times |w|)$ (On parcourt t et on compare les lettres une à une).

Autre solution 42: On construit l'automate des motifs, en $O(P(|w|))$ où P est un polynôme, et on détecte ensuite si w est un sous-mot de t en $O(|t|)$

Développement 2: Construction de l'automate des motifs.

IV.4) Automates en architecture des ordinateurs

Une version modifiée des automates (sans état final) est très utilisée en architecture des ordinateurs, servant à représenter un circuit séquentiel.