Automates & Langages

Sandrine Julia

Licence 3 Info/MI/BIM 2011/12

Organisation

http://deptinfo.unice.fr/~julia/AL

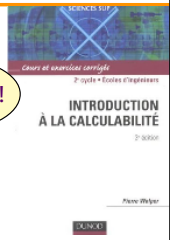
1,5h de cours et 2h de TD hebdomadaires sur 12 semaines + 6 TP de 2h

Cours :	S. Julia	lundi 10h30-12h	salle P.4.2
TD gr.1 :	F. Guingne/E. Formenti	mercredi 8h-10h	salle M.2.7
TD gr.2 :	S. Julia	mercredi 10h15-12h15	salle M.2.7
TP gr.A :	S. Julia	mercredi 8h-10h	salle PV.312
TP gr.B :	J.-V. Millo	mercredi 10h15-12h15	salle PV.312
TP gr.C :	E. Formenti	mercredi 16h45-18h45	salle PV.312

Bibliographie

- Introduction à la calculabilité
  Wolper, InterEditions, 3e éd., 2006.
- Logique et automates
  Bellot/Sakarovitch, Ellipses, 1998.
- Introduction to Automata theory, Languages, and Computation
  Hopcroft, Ullman, Addison-Wesley, 1979.
- A Second Course in Formal Languages and Automata Theory
  Jeffrey Shallit, Cambridge Univ.Press, 2009.
- Introduction to the theory of computation
  Sipser, PWS publishing company, 1997.
- Simulateur JFLAP : http://www.jflap.org/

à la BU !



Utilisation (plutôt pratique)

- spécification des langages de programmation
- compilation
- recherche de motifs
  - dans un texte
  - dans une base de données
  - sur le web
- systèmes d'exploitation, éditeurs
- compression de textes
- cryptographie
- électronique des ordinateurs
- codage pour la transmission
- images de synthèse
- biologie, génétique
- linguistique (TALN)
- sciences cognitives
- ...

Utilité théorique

Spécification de programmes
Vérification
Complexité

Calculabilité :

A. Church
1903-1995

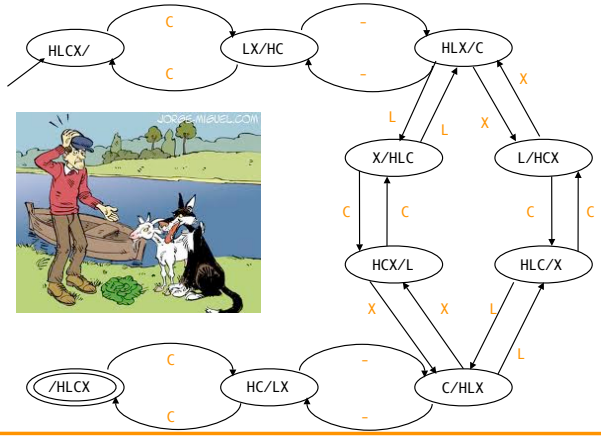
A. Turing
1912-1954

Où se situent les limites de l'informatique ?

- les mots représentent les instances d'un problème
- un langage représentent les instances positives du problème
- un automate ou autre machine de Turing représentent le programme.

Les solutions d'un problème vues comme un langage

Problème du loup, de la chèvre et des choux ...



Contenu du cours

- Automates finis
- Langages rationnels
- Expressions régulières
- Grammaires
- Automates à pile
- Langages non-contextuels
- Machines de Turing

et plus particulièrement en TP :

- Reconnaissance de motifs
- Compression de texte
- Analyseur lexicaux
- Automates cellulaires

1 - Automates finis déterministes

Matériel de base

- symbole, lettre
- alphabet Σ fini
- mot
- langage
- mot vide noté ε (ou υ ou même λ) : l'unique mot à aucune lettre !
- longueur d'un mot m notée $|m|$
  représentation binaire b_n de n : $|b_n|=\lfloor \log_2(n) \rfloor+1$
- nombre d'occurrences du symbole « a » dans m notée $|m|_a$
- i^e caractère du mot m noté $m[i]$
- préfixe (propre)
- suffixe (propre)
- facteur (propre)
- sous-mot
- miroir
- ordres sur les mots : préfixe, lexicographique, hiérarchique

Opérations sur les langages

Soit L et M deux langages donnés.

- opérations ensemblistes

  union, intersection, différences, complémentation, produit cartésien, ...

- produit de concaténation de L et M :

  $L.M = \{w = uv \ / \ u \in L, v \in M\}$

- puissance de L :

  (B) $L^0 = \{\varepsilon\}$

  (I) pour tout $i>0$, $L^i = L.L^{i-1}$

- fermeture de Kleene, notée *

  $L^* = \cup_{i \geq 0} L^i$

- on note $L^+ = \cup_{i>0} L^i$
- même si $L=\varnothing$, $L^0 = \{\varepsilon\}$
- l'ensemble des mots Σ^* est la fermeture de Kleene de l'alphabet Σ
- L^* : plus petit langage contenant ε, L et fermé par concaténation.

Ensemble des langages rationnels*

- plus petit ensemble contenant les langages finis et clos par union, intersection et étoile
- définition inductive :

  (B)

  $\varnothing \in R$

  $\{\varepsilon\} \in R$

  $\{a\} \in R$, pour tout $a \in \Sigma$

  (I)

  si $L, M \in R$ alors

  $L \cup M \in R$

  $L.M \in R$

  $L^* \in R$

- par construction, l'ensemble des langages rationnels est dénombrable

Exemple : les lexèmes des langages de programmation

(les entiers, les hexadécimaux, les identificateurs, les commentaires …)

cf.TD1

* On dit aussi "réguliers" car c'est "regular" en anglais.

Expressions régulières

- formalisme simple pour décrire les langages réguliers
- définition inductive :

  (B)

  $\varnothing \in ER$

  $\varepsilon \in ER$

  $a \in ER$, pour tout $a \in \Sigma$

  (I)

  si $\alpha, \beta \in ER$ alors

  $(\alpha + \beta) \in ER$

  $(\alpha . \beta) \in ER$

  $\alpha^* \in ER$

- par construction, l'ensemble des expressions régulières est dénombrable

Exemple : sur l'alphabet binaire $(((0 + \varepsilon).(1 . 0)^*).(1 + \varepsilon))$

simplifiable en $(0 + \varepsilon)(1\ 0)^*(1 + \varepsilon)$

avec, du plus au moins prioritaire, les opérations : * puis . puis + .

Correspondance entre langage rationnel (ensembles) et expression régulière (formalisme)

(B)
$\varnothing \in ER$
$\varepsilon \in ER$
$a \in ER$, pour tout $a \in \Sigma$

(I)
si $\alpha,\beta \in ER$ alors
$(\alpha + \beta) \in ER$
$(\alpha\ \beta) \in ER$
$(\alpha^*) \in ER$

Expression régulière

Langage

(B)
$L(\varnothing) = \varnothing$
$L(\varepsilon) = \{\varepsilon\}$
$L(a) = \{a\}$ pour tout $a \in \Sigma$

(I)
$L(\alpha + \beta) = L(\alpha) \cup L(\beta)$
$L(\ \alpha\ \beta\ ) = L(\alpha).L(\beta)$
$L(\ \alpha^*\ ) = L(\alpha)^*$

On dit qu'une expression régulière dénote un langage.

Théorème un langage est rationnel si et seulement si il est dénoté par une expression régulière.

Automate fini

Un automate fini A est la donnée d'un quintuplet :

$$(\Sigma,\ Q,\ \delta,\ q_0,\ F)$$

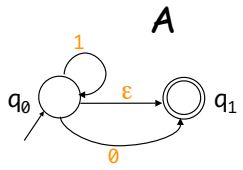
tel que :

- Σ est un alphabet
- Q est un ensemble fini d'états
- δ est un ensemble de règles de transition
  $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times Q$
- q_0 est l'état initial
- F est l'ensemble des états finals
  (un sous-ensemble de Q)

A accepte un mot m s'il existe un chemin de q_0 à un état de F étiqueté par les lettres de m.
L'ensemble des mots acceptés forme le langage $L(A)$ reconnu par A.

Exemple

Automate fini
$A = (\Sigma,\ Q,\ \delta,\ q_0,\ F)$



$\Sigma = \{0,1\}$
$Q = \{q_0,q_1\}$
$\delta = \{(q_0,0,q_1),(q_0,1,q_0),(q_0,\varepsilon,q_1)\}$
q_0 est l'état initial
$F = \{q_1\}$

Ici, A reconnaît le langage $L(A)$ décrit par l'expression régulière :

$$1^*\ (0 + \varepsilon)$$

Automates finis déterministes

Un automate fini est déterministe (A.F.D) ssi δ est une fonction de transition :

$$\delta :\ Q \times \Sigma \to Q$$

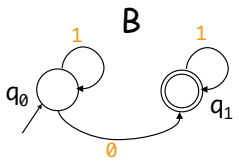
- d'un état donné, il part au plus une seule flèche étiquetée par une lettre donnée.
- on n'autorise pas les ε-transitions

Un automate fini déterministe est complet ssi δ est une fonction totale sur $Q \times \Sigma$.

- de chaque état, il part exactement une flèche étiquetée par chacune des lettres de l'alphabet Σ.

Exemple d'A.F.D

$B = (\Sigma, Q, \delta, q_0, F)$



$\Sigma = \{0,1\}$
$Q = \{q_0, q_1\}$

$\delta = \{(q_0,0,q_1),(q_0,1,q_0),(q_1,1,q_1)\}$

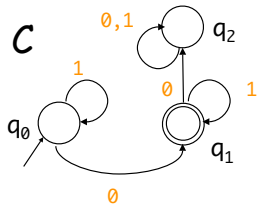
q_0 est l'état initial
$F = \{q_1\}$

B accepte les mots du langage L(B) décrit par l'expression régulière :

$$1^* 0\ 1^*$$

Exemple d'A.F.D complet

$C = (\Sigma, Q, \delta, q_0, F)$



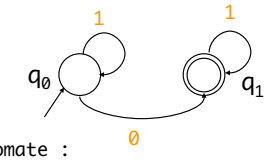
$\Sigma = \{0,1\}$

$Q = \{q_0, q_1, q_2\}$

$\delta = \{(q_0,0,q_1),(q_0,1,q_0),(q_1,0,q_2),(q_1,1,q_1),(q_2,0,q_2),(q_2,1,q_2)\}$

q_0 est l'état initial

$F = \{q_1\}$



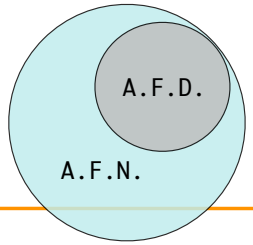
C accepte les mots du langage L(*C*) décrit par $1^*0\ 1^*$, c'est la version complétée de cet automate :

Non-déterminisme

Dans un automate fini non-déterministe (A.F.N.) il peut y avoir le choix entre plusieurs chemins lors de la lecture d'un mot.

Pour qu'un mot soit accepté, il suffit que ses lettres étiquettent un chemin d'un état initial à un état final (même s'il y en a d'autres ne menant pas à un état final, ou bien s'arrêtant en cours de route).

Notez qu'un A.F.D. c'est aussi un A.F.N. !



Equivalence A.F.N. et A.F.D.

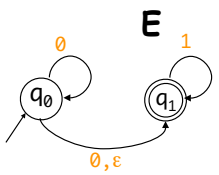
Théorème : si un langage est reconnu par un automate fini, alors il est également reconnu par un automate fini déterministe.

- si l'automate fini du départ *A* est déterministe, c'est évident
- si l'automate de départ n'est pas déterministe, on se propose de construire un automate fini déterministe B qui intègre tous les choix existant dans l'automate de départ (cf. algorithme de déterminisation)
- il resterait à prouver formellement que le nouvel automate B accepte exactement les mots acceptés par *A*.

Exemple de déterminisation

soit l'AFN $\mathbf{E} = (\Sigma, Q, \delta, q_0, F)$

$\Sigma = \{0,1\}$
$Q = \{ q_0, q_1 \}$
$\delta = \{(q_0,0,q_0), (q_0,0,q_1), (q_0,\varepsilon,q_1), (q_1,1,q_1)\}$
q_0 est l'état initial
$F = \{ q_1 \}$

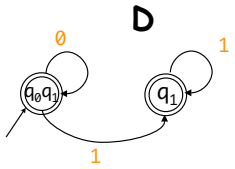


on construit $\mathbf{D} = (\Sigma, Q', \delta', q_0', F')$

$q_0' = \{q_0 q_1\}$ est l'état initial

$Q' = \{\{ q_0 \}, \{ q_0, q_1 \}\}$
$\delta' = \{ (\{ q_0, q_1 \}, 0, \{ q_0, q_1 \}),$
$(\{ q_0, q_1 \}, 1, \{ q_1 \}),$
$(\{ q_1 \}, 1, \{q_1 \})$
$F = \{\{ q_0 q_1 \}, \{q_1 \}\}$

	0	1
$\{q_0,q_1\}$	q_0,q_1	q_1
$\{q_1\}$	-	q_1



Algorithme de déterminisation

1. soit un AFN $A = (\Sigma, Q, \delta, q_0, F)$
   on construit l'automate $B = (\Sigma, Q', \delta', q_0', F')$
   Q' sera inclus dans P(Q)

2. $\delta' \leftarrow \varnothing$
   $q_0' \leftarrow \{q_0\} \cup \{q \in Q$ tels que $(q_0,\varepsilon,q) \in \delta^* \}$

3. pour tout $q' \in Q'$ non encore considéré faire
   pour tout $\sigma \in \Sigma$ faire
   $q'' \leftarrow \{ y \in Q \,/\, \exists\, x \in q'$ tel que $(x,\sigma,y) \in \delta \}$
   si $q'' \neq \varnothing$ alors
   $q'' \leftarrow q'' \cup \{z \in Q \,/\, \exists\, y \in q''$ tel que $(y,\varepsilon,z) \in \delta^* \}$
   $\delta' \leftarrow \delta' \cup \{(q',\sigma,q'')\}$
   $Q' \leftarrow Q' \cup \{ q'' \}$

4. $F' \leftarrow \{q'$ tels que $q' \cap F \neq \varnothing \}$

Clôtures par ε-transitions

Application

En début de compilation, lors de l'analyse lexicale, le flot de caractères est découpé. Chaque morceau appartient à un lexème.

Les générateurs automatiques d'analyseurs lexicaux (Lex, Flex … cf.TP1) utilisent un algorithme pour passer directement d'une expression régulière à un automate fini déterministe complet.

Dénombrabilité

- un alphabet Σ est un ensemble fini de symboles
- l'ensemble des mots Σ^* sur l'alphabet Σ
  ➡ (infini) dénombrable
- l'ensemble des langages sur l'alphabet Σ
  ➡ non-dénombrable

(et aussi :
toutes les fonctions ne sont pas calculables...)

Preuve de non-dénombrabilité

	σ_0	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7	σ_8	σ_9	σ_{10}	σ_{11}	σ_{12}	σ_{13}	σ_{14}	...
L_0	**1**	0	0	0	1	0	0	1	0	0	1	0	0	1	0	...
L_1	1	**0**	0	0	0	0	0	1	0	1	0	0	0	0	0	...
L_2	0	1	**0**	0	1	1	1	0	0	0	0	1	0	0	0	...
L_3	0	1	0	**1**	0	0	0	0	0	0	0	0	0	1	0	...
L_4	0	0	1	0	**0**	0	1	0	0	1	0	0	0	0	0	...
L_5	1	0	1	0	1	**1**	0	0	0	0	0	1	0	1	0	...
⋮																

- soit L le langage tel que :

  $\sigma_i \in L$ ssi $\sigma_i \notin L_i$

- ici $L = \{\sigma_1, \sigma_2, \sigma_4, \ldots\}$
- il ne figure pas dans notre liste donc elle est incomplète
- pour n'importe quelle liste, on peut exhiber un langage qui n'y appartient pas.

⇒ l'ensemble des langages sur Σ n'est pas dénombrable.