Automates & Langages

Sandrine Julia

L3 Info
L3 Math-Info
L3 Sc.&Techn.

2024/25

Organisation

https://upinfo.univ-cotedazur.fr/~julia/AL
+ site Moodle

12 semaines d'enseignement
1h30 cours hebdo + 2h TD hebdo + 2h TP 1 semaine/2

Cours : lundi 10h15-11h45 amphi Géologie

O. Baldellon et S. Julia en alternance :

TD g.1 : lundi 13h15-15h15 salle M.1.8
TD g.2 : lundi 15h30-17h30 salle M.1.8

TP g.1 et 2 : lundi 13h15-15h15 salle PV 301
TP g.3 et 4 : lundi 15h30-17h30 salle PV 301

Evaluation :
(30 * partiel + 20 * évaluation TP + 50 * exam final) / 100
1h30 1h30

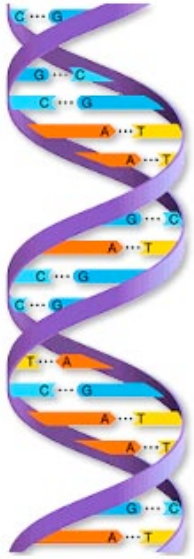
Bibliographie

à la BU !

- Introduction à la calculabilité
  P.Wolper, Dunod, 3e édit°, 2006
- Logique et automates
  P.Bellot, J.Sakarovitch, Ellipses, 1998
- Langages formels, calculabilité et complexité
  O.Carton, Vuibert, 2008
- Introduction to Automata theory, Languages, and Computation
  J.Hopcroft, J.Ullman, Addison-Wesley, 1979
- A Second Course in Formal Languages and Automata Theory
  J.Shallit, Cambridge Univ.Press, 2009
- Introduction to the Theory of Computation
  M.Sipser, PWS publishing company, 2014

Simulateurs :
Awali, vcsn, JFLAP, Visual Automata Simulator, AMoRE

Utilisation (plutôt pratique)



- codage de l'information
- spécification des langages de programmation
- compilation, analyseurs lexicaux
- recherche de motifs :
  - dans un texte
  - dans une BD
  - sur le web
- systèmes d'exploitation, éditeurs
- compression de textes
- électronique des ordinateurs
- model-checking
- images de synthèse
- cryptographie
- biologie, génétique
- linguistique (TALN)
- sciences cognitives
- ...

Utilité théorique

Spécification des langages de programmation
Modélisation
Vérification (model-checking)
Complexité

Calculabilité :

2012 : ANNEE TURING !

A.Church
1903-1995

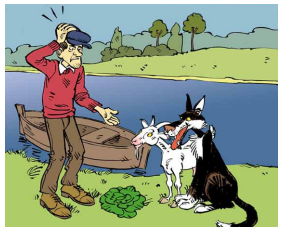
A.Turing
1912-1954

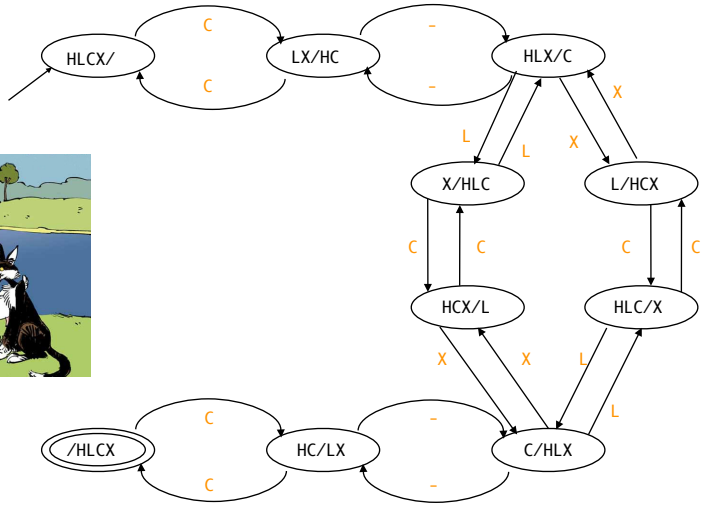
Où se situent les limites de l'informatique actuelle ?

- les mots représentent toutes les instances d'un problème
- un langage regroupe les instances positives du problème
- un automate (ou une autre machine) correspond à un programme

Les solutions d'un problème vues comme un langage

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



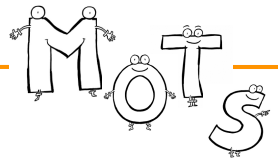


Contenu du cours

- Langages rationnels (ou réguliers)
- Automates finis
- Expressions régulières
- Grammaires régulières
- Grammaires hors-contexte (ou algébriques)
- Automates à pile
- Langages hors-contexte (ou algébriques)
- Grammaires contextuelles
- Machines de Turing
- Automates cellulaires

1 - Langages rationnels

Matériel de base



- symbole : lettre, chiffre, caractère, signe ...
- alphabet : ensemble fini de lettres
- mot : concaténation finie de lettres
- langage : ensemble de mots
- mot vide noté ε : l'unique mot sans aucune lettre !
- longueur d'un mot m : $|m|$
  - représentation binaire de n non nul: $|bin(n)| = \lfloor \log_2(n) \rfloor + 1$
  - mot vide : $|\varepsilon| = 0$
- nombre d'occurrences de la lettre a dans le mot m : $|m|_a$
- le i^e caractère du mot m : $m[i]$

- préfixe, suffixe, facteur, sous-mot (propre ou pas)
- miroir d'un mot m : m^R
- ordres sur les mots :
  - préfixe, suffixe ...
  - lexicographique (généralisation ordre alphabétique)
  - hiérarchique (ou militaire)

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 :
  $L \,.\, M = \{ w = uv \;/\; u \in L,\ v \in M \}$
- quotient gauche de M par L
  $L^{-1} M = \{ w \;/\; \text{il existe } u \in L \text{ et } v \in M \text{ tels que } v = uw \}$
- puissance de L :
  (B) $L^0 = \{ \varepsilon \}$
  (I) pour tout entier $i>0$, $L^i = L \,.\, L^{i-1}$
- fermeture de Kleene du langage L : $L^* = \cup_{i \geq 0} L^i$

appelée "opération étoile"

  - on note $L^+ = \cup_{i>0} L^i$
  - par convention : $L^0 = \{ \varepsilon \}$ (même si $L = \varnothing$)
  - 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

Exemples

- $K = \{ 0,1,2,3,4 \}$
- $L = \{ 5,6,7,8,9 \}$
- $M = \{ 0,1 \}$

- $K \cup L = \{ 0,1,2,3,4,5,6,7,8,9 \}$
- $M.L = \{ 05,06,07,08,09,15,16,17,18,19 \}$
- $M.M = M^2 = \{ 00,01,10,11 \}$
- $K^{-1} \{ 01,2,345,5678,9 \} = \{ 1,\varepsilon,45 \}$
- $M^{-1} K = \{ \varepsilon \}$
- $M^{-1} L = \varnothing$

- $(K \cup L)^* \setminus \{\varepsilon\}$: ensemble des représentations décimales des entiers (avec éventuellement des 0 inutiles en tête)

Ensemble des langages rationnels*

- plus petit ensemble contenant les langages finis sur un alphabet Σ 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 (entiers, hexadécimaux, identificateurs, commentaires ...)

* on dit aussi réguliers (regular en anglais)

Expressions régulières (E.R.)

- formalisme simplifié pour décrire les langages rationnels
- définition inductive :

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

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

Unix a son propre formalisme pour les exp. régulières que **grep, find, awk ...** utilisent

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

Exemple : $(((0 + \varepsilon).(1 . 0)^*).(1 + \varepsilon))$ qui peut se simplifier en
$(0 + \varepsilon)(1\ 0)^*(1 + \varepsilon)$

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

Correspondance entre langage rationnel (ensemble) 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

Le langage $L(\alpha)$ est décrit ou dénoté par l'expression régulière α

Langage rationnel

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

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

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

Automate fini (A.F.)

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 (les flèches)

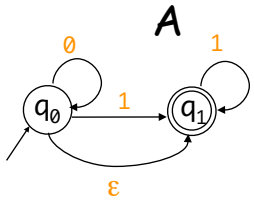
$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times Q$

- q_0 est l'état initial
- F est l'ensemble des états finals (F sous-ensemble de Q)

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

Exemple

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



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

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

$0^*\ 1^*$

Automates finis déterministes (A.F.D.)

Un automate fini est déterministe si et seulement si la relation δ est une fonction de transition :

$$\delta : (Q \times \Sigma) \rightarrow Q$$

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

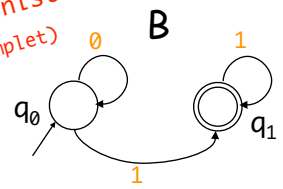
Un automate fini déterministe est complet si et seulement si δ est une fonction définie sur l'ensemble $(Q \times \Sigma)$ tout entier

- de chaque état et pour chaque lettre de l'alphabet Σ, il part exactement une flèche étiquetée par cette lettre.

Exemple d'A.F.D

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

B déterministe (non-complet)



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

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

q_0 est l'état initial
$F = \{q_1\}$

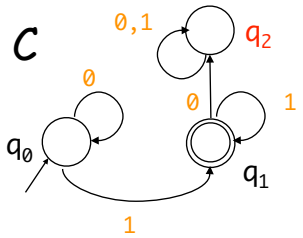
B accepte les mots du langage L(B) décrit par l'expression régulière :

$0^* 1 1^*$ aussi noté $0^* 1^+$

Exemple d'A.F.D complet

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

C déterministe et complet !



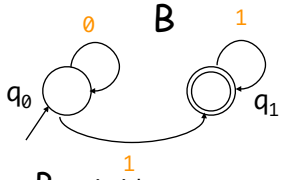
$\Sigma = \{0,1\}$

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

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

q_0 est l'état initial

$F = \{q_1\}$



C reconnaît le langage L(C) = L(B) décrit par 0^*1^+
c'est la version complétée de l'automate déterministe B précédent.

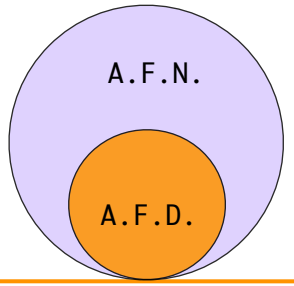
IMPORTANT !!!

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 de l'état initial à un état final (même s'il existe d'autres chemins non réussis).

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



Equivalence A.F.N. / A.F.D.

Théorème
si un langage est reconnu par un automate fini alors
il est aussi 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 D
  qui intègre tous les choix existant dans l'automate de départ
  (cf. algorithme de déterminisation)
- il resterait à prouver formellement que le nouvel automate D accepte exactement les mots acceptés par **A** ...

Algorithme de déterminisation

1. soit un AFN $A = (\Sigma, Q, \delta, q_0, F)$
   on construit l'automate $D = (\Sigma, Q', \delta', q_0', F')$
   Q': les nouveaux états seront issus de $\mathcal{P}(Q)$

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

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 $q' \cap F \neq \varnothing \}$

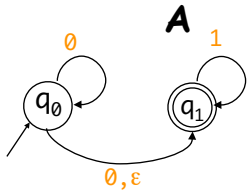
Clôtures par ε-transitions

(δ^* est la généralisation de δ aux mots)

Exemple de « déterminisation »

soit l'AFN : $A = (\Sigma, Q, \delta, q_0, F)$

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



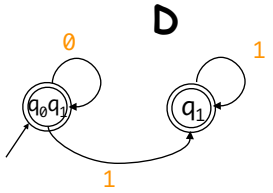
on construit l'AFD : $D = (\Sigma, Q', \delta', q_0', F')$

$q_0' = \{q_0 q_1\}$ nouvel état initial

$Q' = \{\{q_0,q_1\}, \{q_1\}\}$
$\delta' = \{ (\{ q_0,q_1 \},0,\{ q_0,q_1 \}),$
$(\{ q_0,q_1 \},1,\{ q_1 \}),$
$(\{ q_1 \},1,\{q_1\})$
$F' = \{\{ q_0q_1\},\{q_1\}\}$

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

D non-complet !



Application

cf. UE Compilation au S6

En début de compilation, lors de l'analyse lexicale, le flot de caractères est découpé. Chaque morceau appartient à un lexème i.e. un petit langage décrit par une expression régulière.

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

Algorithme : E.R. → A.F.D.

- on indice les n lettres de Σ figurant dans E (de gauche à droite par les entiers de 1 à n)
- on transforme E en E' en remplaçant chaque lettre par q ($q \notin \Sigma$)
- on ajoute q_0 au tout début de E'
- pour tout i de 1 à n faire
  pour tout $\sigma \in \Sigma$ faire
  succ(q_i,σ) ← {q_j suivant q_i dans E' après lecture de σ}
- Q ← {q_0}
- $\delta \leftarrow \varnothing$
- pour tout $q \in Q$ non encore considéré faire
  pour tout $\sigma \in \Sigma$ faire
  q' ← succ (q,σ)
  si q' $\neq \varnothing$ alors
  δ ← ($\delta \cup$ (q,σ, q'))
  Q ← Q $\cup$ {q'}
- F ← { états qui contiennent un symbole final de E' }

Exemple : E.R. → A.F.D.

E : $(a + \varepsilon)(b\ a)^*(b + \varepsilon)$

- on indice les lettres :

$$(a_1 + \varepsilon)(b_2\ a_3)^*(b_4 + \varepsilon)$$

- on remplace les lettres par le symbole q et on met q_0 devant :

E' : $q_0\ (q_1 + \varepsilon)(q_2\ q_3)^*(q_4 + \varepsilon)$

initialisations :	Q ← {{ q_0 }}	$\delta \leftarrow$	$\varnothing$
q_0 ,a:	Q ← Q $\cup$ {q_1}	$\delta \leftarrow$	$\delta \cup$ {(q_0,a,q_1)}
q_0 ,b:	Q ← Q $\cup$ {{q_2,q_4}}	$\delta \leftarrow$	$\delta \cup$ {(q_0,b,{q_2,q_4})}
q_1 ,a:	(rien)		
q_1 ,b:		$\delta \leftarrow$	$\delta \cup$ {(q_1,b,{q_2,q_4})}
{q_2,q_4},a:	Q ← Q $\cup$ {q_3}	$\delta \leftarrow$	$\delta \cup$ {({q_2,q_4},a,q_3)}
{q_2,q_4},b:	(rien)		
q_3 ,a:	(rien)		
q_3 ,b:		$\delta \leftarrow$	$\delta \cup$ {(q_3,b,{q_2,q_4})}

➡ $A = (\Sigma,\ Q=\{q_0, q_1, q_3, \{q_2,q_4\}\}, \{q_0\}, \delta, F=\{q_0, q_1, q_3, \{q_2,q_4\}\})$