Université J.F. CHAMPOLLION Albi

Théorie des Langages

Licence d'informatique S6

Thierry Montaut

Plan

Théorie des Langages

1. Mots et Langages.
2. Automates finis déterministes et non déterministes :
   AFD - langage d'un AFD - AFN - Construction d'automates
   Théorème de Kleene - Equivalence d'automates - minimisation
3. Langages algébriques :
   Grammaires formelles - automates à pile - Analyse syntaxique
4. Machines de Turing et calculabilité
   Machines de Turing - fonctions MT calculables - Thèse de Church

Pour toute question : thierry.montaut@univ-jfc.fr

Introduction

Maintenant que vous connaissez des langages de différents paradigmes (impératifs : C ou Python, Orientés objets ; Java (ou encore Python) ; fonctionnels (Caml)), vous pourriez être tentés d'écrire votre propre langage de programmation (cela ne doit pas être si difficile puisqu'il en naît tous les ans). Les plus ambitieux d'entre-vous pourraient même être tentés d'écrire "le langage ultime", capable d'écrire tous les programmes, résolvant tous les problèmes, calculant toutes les fonctions.

1. Dans le premier, il vous faudrait alors écrire un compilateur pour votre langage. C'est-à-dire un programme qui, étant donné un programme P, écrit dans votre langage, est capable de décider s'il est syntaxiquement correct ou s'il comporte des erreurs de syntaxe, puis s'il est correct de le traduire en un programme exécutable (en langage machine).
2. Dans le second cas, il faudra peut-être d'abord vous demander ce qu'est un problème **effectivement** résoluble par un ordinateur. Ce qu'est une fonction **effectivement** calculable sur machine.

1. Au début du XXème siècle, donc avant même la construction du premier ordinateur, les logiciens Kurt Gödel, Alonzo Church et Alan Turing ont cherché à donner un sens précis au concept de fonction effectivement calculable et aux limites des langages effectivement décidables.
2. Depuis plusieurs modèles de description des langages ont été proposés, chacun avec ses intérêts et ses limitations propres.
3. Le but de ce cours est de présenter ses différents modèles et leurs propriétés ainsi que les outils théoriques adaptés à l'étude des langages et de la notion de fonction effectivement calculable.

A l'issue de ce cours, vous devriez disposer des bagages théoriques suffisants pour suivre un cours de compilation et écrire votre propre compilateur (ce qui se fait en Master d'informatique).

Nous étudierons successivement :

1. Les langages formels : Définition et propriétés théoriques des langages et des opérations entre langages.
2. Les automates finis : Un modèle très efficace pour la reconnaissance pratique des langages.
3. Les grammaires formelles : Définitions "pédagogique" des langages, analyse lexicale, applications à la compilation.
4. Les machines de Turing : Le concept de base de la théorie de la calculabilité.

Alphabets et Mots

Définition

Un alphabet est un ensemble fini. On appelle symbole ou lettre les éléments de l'alphabet.

Exemples :

$$\Sigma = \{0, 1\}, \quad \Sigma = \{a, b\}, \quad \Sigma = \{a, b, ..., z, \%, \#\}$$

$$\Sigma = \{0, 1, +, *\}, \Sigma = \{let, fun, ->, =, ;;\}$$

Sont des exemples d'alphabets courants.
Sauf cas particulier, Nous construirons tout ce cours sur l'alphabet "minimal" : $\Sigma = \{a, b\}$, qui permet de tout illustrer avec un minimum de lettres.

Alphabets et Mots

Définition

Un mot sur l'alphabet Σ est une suite finie de lettres appartenant à l'alphabet Σ.

Définition

Le mot vide (contenant 0 lettres) est noté ε .

Définition

L'ensemble des mots sur Σ est noté Σ^*.

Définition

Si $w \in \Sigma^*$ est un mot sur l'alphabet Σ, on appelle longueur du mot et on note $|w|$ son nombre de lettres.

Bien sûr,

Proposition

Deux mots u et v sont égaux ssi ils ont même longueur et si toutes leurs lettres sont égales.

Concaténation

L'unique opération définie entre mots est la concaténation, qui consiste à mettre les mots bout à bout :

Définition

Soit u et v deux mots sur un même alphabet Σ, de longueurs respectives n et p. La concaténation de u et de v est le mot w de longueur $n+p$ défini par :

$$\forall i \in [\![1, n]\!] : w[i] = u[i]$$

$$\forall i \in [\![1, p]\!] : w[n+i] = v[i].$$

La concaténation est notée $u.v$ et souvent appelée "produit" dans ce cours.

Proposition

1. *La concaténation est associative :* $(u.v).w = u.(v.w)$. *On pourra donc noter simplement* $u.v.w$ *sans risque d'ambigüité.*
2. *Le mot vide* ε *est élément neutre pour la concaténation.*
3. *Si l'alphabet a plus d'une lettre, la concaténation n'est pas commutative (en général* $u.v \neq v.u$*).*

Puissance d'un mot

La puissance $k$ième d'un mot est la répétition k fois de ce mot. On peut la définir précisément par récurrence.

Définition

Soit u un mot sur un alphabet Σ.

- $u^0 = \varepsilon$ *(le mot u répété 0 fois)*
- $u^1 = u$ *(le mot u répété 1 fois)*
- $u^{n+1} = u^n.u$

Exemple $(ba)^3 = bababa$.

Proposition

On a alors $u^{m+n} = u^m.u^n = u^n.u^m$

Proposition

L'ensemble des mots Σ^* muni de la concaténation est simplifiable à gauche et à droite. :

- *Si $u.w = v.w$ alors $u = v$ (par simplification à droite)*
- *Si $w.u = w.v$ alors $u = v$ (par simplification à gauche)*

Ordre préfixe

Il existe deux relations d'ordres classiques sur les mots.
Que l'alphabet soit ordonné ou pas, on définit simplement une relation d'ordre partielle sur les mots à l'aide de la concaténation :

Définition

Le mot u est un préfixe du mot w ssi il existe un mot v sur le même alphabet tel que

$$w = u.v$$

On note alors

$$u \sqsubseteq w.$$

Si $u \sqsubseteq w$ et que $u \neq w$, on dit que u est un préfixe stricte de w et on note $u \sqsubset w$.

Proposition

La relation préfixe définit une relation d'ordre partiel sur Σ^*.

Ordre lexicographique

Définition

Si l'alphabet Σ est muni d'une relation d'ordre total notée $\leq$, l'ordre lexicographique définit une relation d'ordre total sur Σ^*, que l'on notera encore $\leq$.
Cet ordre est compatible avec l'ordre préfixe : Si $u \sqsubset v$ alors $u \leq v$

Langages

Définition

Soit Σ un alphabet. On appelle langage sur l'alphabet Σ toute partie de Σ^* : $L \subseteq \Sigma^*$.

L'ensemble de tous les langages sur l'alphabet Σ est $P(\Sigma^*)$ l'ensemble des sous-ensembles de Σ^*.

Exemples : On se fixe l'alphabet : $\Sigma = \{a, b\}$.

1. $\{\varepsilon, a\}$ est un langage de deux mots.
2. $\{a, b, ab, abba\}$ est un langage de quatre mots.
3. $\{\varepsilon\}$ contient un mot.
4. $\emptyset$ n'en contient pas.

Exemples de langages infinis

Nous utiliserons beaucoup les exemples suivants définis sur l'alphabet $\Sigma = \{a, b\}$.

1. Les mots contenant un nombre pair de lettres.
2. Les mots commençant par "a"
3. Les mots ayant un nombre pair de "b"
4. Les mots ayant autant de "a" que de "a"
5. Les mots ne contenant pas la sous-chaîne "abba"
6. etc.

Exemple fondamental

Il n'existe qu'un nombre fini d'éléments lexicaux en Caml.
On appelle Σ ={ let, fun, 1, a, [, and, &, ->, ; ;, ... } l'ensemble de ces éléments lexicaux.
Une expression ou un programme Caml peut alors être vu comme un mot sur cet alphabet :

$$let \;\; f \;\; = \;\; fun \;\; x \;\; -> \;\; x+1 \;\; ;;$$

On appelle « Langage Caml » le langage constitué des mots sur l'alphabet Σ vérifiant les règles syntaxiques caractéristiques du langage.
C'est donc bien un langage au sens ou nous l'entendons dans ce cours...Se demander si un programme est bien écrit, c'est se demander si, en tant que mot, il appartient au « Langage Caml. »

Opérations sur les Langages

Somme de langages

Dans tout ce paragraphe, L_1 et L_2 sont deux langages sur le même alphabet Σ.

Définition

On appelle somme des langages L_1 et L_2 leur union :

$$L_1 + L_2 = \{w \in \Sigma^* \text{ tel que } w \in L_1 \text{ ou } w \in L_2\}$$

Proposition

La somme est commutative, associative et admet le langage vide $\emptyset$ comme élément neutre.

Intersection

Définition

L'intersection des langages L_1 et L_2 est :

$$L_1 \cap L_2 = \{w \in \Sigma^* \text{ tel que } w \in L_1 \text{ et } w \in L_2\}$$

Proposition

1. *L'intersection est commutative, associative et admet le langage Σ^* comme élément neutre.*
2. *Somme et intersection sont distributives l'une par rapport à l'autre.*

$$L + (L_1 \cap L_2) = L + L_1 \cap L + L_2$$

$$L \cap (L_1 + L_2) = (L \cap L_1) + (L \cap L_2)$$

Différence

Définition

La différence ensembliste " L_1 privé de L_2 " est :

$$L_1 \setminus L_2 = \{w \in \Sigma^* \text{ tel que } w \in L_1 \text{ et } w \notin L_2\}$$

Définition

Le complémentaire du langage L est :

$$\overline{L} = \{w \in \Sigma^* \text{ tel que } w \notin L\}$$

Produit de concaténation de Langages

Définition

On appelle produit (de concaténation) des langages L_1 et L_2 le langage :

$$L_1.L_2 = \{w \in \Sigma^* \text{ tel que } \exists u \in L_1, \exists v \in L_2 \text{ tels que } w = u.v\}$$

Donc les mots constitués d'un mot de L_1 suivi d'un mot de L_2.

Proposition

Le produit de langages est une opération associative, distributive par rapport à l'union, d'élément neutre $\{\varepsilon\}$.

- Attention : Le produit n'est pas distributif par rapport à l'intersection !
  On a bien
  $$L.(L_1 \cap L_2) \subseteq L.L_1 \cap L.L_2$$
  Mais la réciproque est fausse en général. Nous le montrerons en TD.
- Remarque : $L.\{\varepsilon\} = L$ mais $L.\emptyset = \emptyset$ Ce qui prouve la différence entre les deux langages.

Puissance d'un langage

Définition

Soit $L \subset \Sigma^*$, on définit par récurrence les puissances du langage L par :

- $L^0 = \{\varepsilon\}$
- $L^1 = L$
- $L^2 = L.L$
- $\forall n \geq 1, L^n = L^{n-1}.L$

Remarque : $L^2 = L.L$ contient bien tous les w^2 où w est un mot de L mais pas seulement :
Ex : Si $L = \{a, a^2\}$, $L^2 = \{a^2, a^3, a^4\}$

Exemple :

Si $L = \{a, da\}$

- $L^0 = \{\varepsilon\}$
- $L^2 = \{aa, ada, daa, dada\}$
- $L^3 = \{aaa, aada, adaa, adada, daaa, daada, dadaa, dadada\}$

Langage plus et langage étoile

Définition

Soit $L \subset \Sigma^*$, on définit les langages

$$L^+ = L + L^2 + ... + L^p + ..$$

Les mots obtenus en concaténant des mots de L (au moins 1)

$$L^* = \{\varepsilon\} + L + L^2 + ... + L^p + ..$$

Les mots obtenus en concaténant des mots de L (y compris 0 mots)

Respectivement nommés "langage plus" et "langage étoile" de *L*.

Remarque :
La notation est cohérente avec Σ^*.
Soit Σ un alphabet, alors Σ est aussi le langage sur Σ constitué de tous les mots d'une lettre. Alors Σ^2 est l'ensemble des mots de 2 lettres sur cet alphabet, Σ^3 celui des mots de 3 lettres et finalement Σ^* le langage constitué de tous les mots de taille quelconque.

Proposition

- *Si* $L_1 \subseteq L_2$ *alors pour tout entier p :*

$$L_1^p \subseteq L_2^p \text{ et } L_1^* \subseteq L_2^*.$$

- $(L^*)^* = L^*$
- $L^+ = L.L*$
- $L^* = \varepsilon + L^+$
- $\varepsilon \in L$, *si et seulement si* $L^* = L^+$.

Expression régulière

Définition

On définit une expression régulière sur l'alphabet Σ récursivement de la manière suivante :

- ε *est une expression régulière,*
- *Toute lettre* $x \in \Sigma$*, est une expression régulière,*
- *Si* E *est une expression régulière alors* (E) *est une expression régulière,*
- *Si* $E1$ *et* $E2$ *sont des expressions régulières alors* $E1 + E2$ *est une expression régulière,*
- *Si* $E1$ *et* $E2$ *sont des expressions régulières alors* $E1.E2$ *est une expression régulière,*
- *Si* E *est une expression régulière alors* E^* *est une expression régulière.*

Dit autrement une expression régulière est toute expression "bien formée", construite à partir d'ε, des lettres et des opérations somme, produit et étoile.
On vérifie à l'aide de cette définition récursive que

$$(b^*.a)^* + b.a^* + a$$

est une expression régulière sur $\Sigma = \{a, b\}$.

Langage régulier

Définition

Un langage est dit régulier ssi il peut être décrit par une expression régulière.

Exemple : Le langage L des mots sur $\Sigma = \{a, b\}$ contenant exactement un "a" est régulier car il peut être décrit par l'expression :

$$L = b^*.a.b^*$$

Complément sur la notion de langage décidable

- On rappelle qu'un ensemble de cardinal infini est dit dénombrable s'il possède autant d'éléments que $\mathbb{N}$. et qu'il est non dénombrable s'il en possède au moins autant que $\mathbb{R}$.
- On montre en mathématiques que si E est un ensemble infini dénombrable, alors l'ensemble de ses parties $P(E)$ est un ensemble infini NON dénombrable.
- Soit Σ un alphabet non vide. Il possède donc au moins une lettre notée a. Alors Σ^*, l'ensemble de ses mots est infini (il contient tous les $a^k, k \in \mathbb{N},$) et dénombrable (les mots peuvent être ordonnées dans l'ordre lexicographique donc peuvent être numérotés dans l'ordre, donc mis en bijection avec $\mathbb{N}$. D'après le rappel $P(\Sigma^*)$ qui est l'ensemble des langages sur l'alphabet Σ est alors infini non dénombrable.

Dit autrement, dès que l'alphabet Σ contient au moins UNE lettre, il y a autant de mots sur cet alphabet qu'il y a d'entiers dans $\mathbb{N}$ et autant de langages sur cet alphabet qu'il y a de nombres réels.

Complément sur la notion de langage décidable

- La spécification d'un langage L par une propriété caractéristique de ses éléments ne fournit pas nécessairement un moyen pratique pour déterminer si un mot donné appartient à L ou pas.
- Bien sûr dans le cas d'un langage fini on peut énumérer tous ses éléments.
- Dans le cas général, on cherche à décrire une méthode effective, un algorithme qui pour chaque mot permet de savoir en un temps fini si un mot donné appartient à L : C'est ce qu'on appelle le problème de décision du langage L.

Complément sur la notion de langage décidable

Définition

Un langage L est dit décidable s'il existe une méthode effective de résolution du problème de décision du langage L c'est-à-dire en d'autres termes un algorithme permettant en un temps fini de dire si oui ou non un mot donné w appartient au langage L.

Nous décrirons dans la suite plusieurs types de tels procédés effectifs de décision. Il existe donc des langages décidables. Mais...

Théorème

L'ensemble des langages décidables est dénombrable.
Il existe donc un ensemble infini non dénombrable de langages qui ne le sont pas.

- Donc les langages pour lesquels il existe un procédé pratique de décision sont au milieu de ceux qu'on ne peut décider (donc décrire) comme $\mathbb{N}$ est au milieu de $\mathbb{R}$, ce qui doit nous rendre humble.
- Idée de la preuve : L'ensemble des éléments syntaxiques du langage Caml est fini. Il existe donc un nombre infini mais dénombrable de programmes en Caml.
  Si un langage est décidable il existe un algorithme de décision de ce langage, algorithme qu'on peut coder en Caml. Il ne peut donc exister qu'un ensemble dénombrable de tels langages.
- Je dis cela pour votre culture, mais soyez rassurés, on ne vous demandera rien sur les langages décidables...cette année.