7. Théorie des langages

Utilisation

* spécification de langages de programmation
* compilation
* recherche de motifs dans les éditeurs de texte
* dans les programmes
* dans une base de données
* sur le web
* compression de textes
* preuves de programmes
* codage et décodage
* cryptographie
* décodage du génôme
* calculabilité
* et aussi : linguistique, sciences cognitives, ...

Mots

* un **symbole** ou une lettre est un caractère donné : a, 0, 1, α, +, ...
* un **alphabet** Σ est un ensemble (fini) de symboles
  * Exemples : {0,1}, {a,b}, {0,...,9}, caractères Unicode
* un **mot** est une suite finie de symboles
  * Exemples : 01011100, abbaaba, (5+7)/3, ~/SI/TP10/exo3.sh
  * class Hello { public static void main(String [] args) {...} }
  * le source d'une page HTML
* **Concaténation** de mots
  * Si u = abba et v = baaa, alors u.v= abbabaaa

Rappels sur les mots

* le mot vide ε est le mot de longueur 0 (à distinguer du caractère ε) qui vérifie :
  $u.\varepsilon = \varepsilon.u = u$
* l'ensemble des mots sur l'alphabet Σ se note Σ^*
  * si $\Sigma=\{0,1\}$, $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$

* On appelle langage sur Σ toute partie de Σ^*
* Exemples de langages
  * l'ensemble des programmes écrits en C
  * les représentations binaires des int de Java
  * le langage de Dyck (les mots bien parenthésés)

Vocabulaire

* la longueur $|m|$ d'un mot m est son nombre de lettres et $|\varepsilon| = 0$
  * Exemple $|0\,1\,0\,0| = 3$
  * représentation binaire b_n de n : $|b_n| = \lfloor \log_2(n) \rfloor + 1$

* le nombre d'occurrences d'une lettre a dans un mot m est notée $|m|_a$
  * $|0\,1\,0\,0|_0 = 2$
  * $|0\,1\,0\,0|_1 = 1$
* le ième caractère d'un mot m se note $m[i]$, $1 \leq i \leq |m|$

Vocabulaire

* u est un facteur de m s'il existe v et w tels que m = v u w.
  * si v = ε, u est préfixe de m,
  * si w = ε, u est suffixe de m.
  * Si v≠ε et w≠ε, u est facteur propre,
  * un préfixe ou un suffixe u est propre si u≠ε et u≠m.
* on obtient un sous-mot d'un mot donné m en effaçant une ou plusieurs de ses lettres.
* le miroir d'un mot m est le mot obtenu en inversant le sens de lecture
* Les palindromes sont les mots identiques à leur miroir (ex: elle, bob...).

Ordres sur les mots

* Soit $(A, <_{alph})$ un alphabet totalement et strictement ordonné,
  * $\leq_{alph}$ est l'ordre non strict associé à $<_{alph}$.
* Voici la définition de quelques ordres :
  * ordre préfixe si un mot est préfixe d'un autre
    - (B) $\varepsilon \leq \varepsilon$, pour tout a de A : $\varepsilon \leq a$
    - (I) si $u \leq v$ alors, pour tout a de A : $u \leq v\,a$ et $a\,u \leq a\,v$
  * ordre lexicographique (c'est celui du dictionnaire)
    - (B) $\varepsilon \leq_{lex} \varepsilon$, et
      pour tout $(a,b) \subseteq A2$ tel que $a \leq_{alph} b$: $\varepsilon \leq_{lex} a$ et $a \leq_{lex} b$
    - (I) - si $u \leq_{lex} v$ et $|v| \leq |u|$ alors $u\,a \leq_{lex} v$ pour tout a de A
      - si $u \leq_{lex} v$ alors $a\,u \leq_{lex a} v\,a$ et $u \leq_{lex} v\,a$ pour tout a de A
  * ordre hiérarchique (aussi appelé militaire)
    * tri des mots sur la longueur puis à longueur donnée, tri lexicographique

Opérations sur les langages

* Soit L et M deux langages donnés. On dispose tout d'abord des opérations ensemblistes (union, intersection, différences, complémentation, produit cartésien, ...)
  * Définition du produit de concaténation de L et M :
    *
    $$L.M = \{w = uv \,/\, u \in L, v \in M\}$$
  * Définition récursive de la puissance de L :
    * (B) $L0 = \{\varepsilon\}$
    * (I) pour tout $i>0$, $L_i = L.L_{i-1}$

$$L^n = \{u_1.u_2 \ldots u_n, u_i \in L_i\}$$

* Définition de la fermeture de Kleene, notée *
* Remarques
  - on note $L+ = \bigcup_{i \geq 0} L_i$
  - si $L=\varnothing$, $L0 = \{\varepsilon\}$
  - L'ensemble des mots $\Sigma*$ est bien la fermeture de Kleene de l'alphabet Σ.

$$L^* = \bigcup_{i \geq 0} L_i$$

Exemples

* Soient L = {0, 1, 2, 3, 4} et M = {5, 6,7,8,9}
  * L ∪ M = {0,1,2,3,4,5,6,7,8,9}
  * L^2 = {00,01,02,03,04,10,11,12,13,14,20,21,22,23,24, 30,31,32,33,34, 40,41,42,43,44}
  * L.M = {05,06,07,08,09,15,16,17,18,19,25,26,27,28,29,35,36,37,38,39,...}
* $(L \cup M)^+ = (L \cup M)^* \setminus \{\varepsilon\}$ est l'ensemble de toutes les représentations décimales, y compris celles avec des 0 inutiles en tête.

Propriétés

Si K, L et M sont des langages, on a :

* $(K^*)^* = K^*$
* $L \cup \varnothing = \varnothing \cup L = L$
* $K \cup L = L \cup K$
* $K \cup (L \cup M) = (K \cup L) \cup M$
* $L.\varnothing = \varnothing.L = \varnothing$
* $L.\{\varepsilon\} = \{\varepsilon\}.L = L$
* $K (L \cup M) = K.L \cup K.M$
* $(K \cup L) . M = K.M \cup L.M$
* $(K.L) . M = K.(L.M)$
* $(K \cup L)^*. K = (K^*.L)^* K+$

Dénombrabilité

Soit A = { a0, a1, a2, a3, ... } un alphabet ordonné.

* L'ensemble des mots sur A est dénombrable lorsque A est fini.
  * Preuve
    * on trie les éléments de A* par ordre hiérarchique et, triés ainsi, on les indice par les entiers naturels
    * on a ainsi mis A* en bijection avec IN et donc prouvé que A* est dénombrable.

* L'ensemble des langages sur A n'est pas dénombrable.
  * Preuve (cf. vue suivante)
    * supposons par l'absurde que l'ensemble des langages est dénombrable ; cela revient à les indicer par les entiers naturels
    * on associe à chaque langage sa fonction caractéristique sur A*
    * l'argument de la diagonale de Cantor nous assure de l'existence d'un langage absent de la liste
    * on en déduit que l'ensemble des langages sur A* n'est pas dénombrable.

Illustration de la preuve

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{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	...
⋮																

On en déduit que le langage L tel que : $a_i \in L$ ssi $a_i \notin L_i$, ici $L = \{a_1, a_2, a_4, \dots\}$
ne figure pas dans la liste qui est donc incomplète.
Pour n'importe quelle liste, on pourrait exhiber un langage qui n'y appartient pas

Conclusion l'ensemble des langages sur A n'est pas dénombrable.

L'ensemble des langages sur un ensemble fini n'est pas dénombrable

Théorème. Il n'existe pas de bijection d'un ensemble E dans l'ensemble de ses parties P(E).

* Preuve. Raisonnons par l'absurde et supposons que f soit une bijection de E dans P(E). Pour tout élément x de E on peut se demander si x est un élément de f(x) ou pas. Considérons justement la partie X de E constitué des éléments x de E tels que x ne soit pas élément de f(x). $X=\{x \in E$ tels que $x \notin f(x)$

* Puisque f est surjective, on peut trouver un élément a de E tel que f(a)=X.

  * Mais alors, par définition de X, $a \in X=f(a)$ ssi $a \notin X$, d'où la contradiction : a est élément de E sans être ni dans X ni dans son complémentaire !

* **Corollaire.** P(A*) n'est pas dénombrable, puisque A* est dénombrable et qu'il n'existe pas de bijection de P(A*) dans A*.

Langages réguliers

Soit A un alphabet. On définit inductivement l'ensemble R des **langages réguliers sur A** (aussi dits langages rationnels)

* **Base**
  * $\emptyset \in R$
  * $\{\varepsilon\} \in R$
  * $\{a\} \in R$, pour tout $a \in A$
* **Induction.** si $L, M \in R$ alors
  * $L \cup M \in R$
  * $L.M \in R$
  * $L^* \in R$
* **Remarque.** R est le plus petit ensemble contenant les langages finis et clos par union, concaténation et opération *.

Exemples de langages réguliers

* l'ensemble des octets
* l'ensemble de toutes les représentations décimales
  y compris celles sans des 0 inutiles en tête
* les nombres du langage C
  les hexadécimaux, les octaux et les décimaux (à virgule ou pas, avec ou sans exposant, signés ou pas)
* les identificateurs
  ceux de Java, C, Pascal...
* Remarque
  * il existe des langages non réguliers (voir plus loin)
  * Le langage de Dyck n'est pas un langage régulier.
  * les langages de programmation ne sont pas réguliers.

Expressions régulières

* Un langage régulier peut être défini comme étant un ensemble de mots sur un alphabet donné et ayant telle ou telle propriété.
* Une telle définition ensembliste ne peut pas nous apprendre directement si un langage est régulier ou pas.
* L'idée est d'élaborer un formalisme pour décrire les langages réguliers et eux seuls.
* Ce formalisme est celui des expressions régulières. Un langage décrit par une expression régulière sera forcément un langage régulier.

Syntaxe des expressions régulières

Soit A un alphabet.. On définit inductivement la syntaxe des expressions régulières par

* **Base**
  * $\emptyset \in ER$
  * $\varepsilon \in ER$
  * $a \in ER$, pour tout $a \in A$
* **Induction**. si $E, F \in ER$ alors
  * $(E + F) \in ER$
  * $(E.F) \in ER$
  * $(E^*) \in ER$
* Remarque. On peut enlever les parenthèses superflues en supposant que * est prioritaire sur . elle-même prioritaire sur +.

Sémantique des expressions régulières

Expression régulière

* (B)
  * $\varnothing \in ER$
  * $\varepsilon \in ER$
  * $a \in ER$, pour tout $a \in A$
* (I) si $E, F \in ER$ alors
  * $(E+F) \in ER$
  * $(E.F) \in ER$
  * $(E^*) \in ER$

Langage associé à une expression régulière

* (B)
  * $L(\varnothing) = \varnothing$
  * $L(\varepsilon) = \{\varepsilon\}$
  * $L(a) = \{a\}$ pour tout $a \in A$
* (I)
  * $L(E+F) = L(E) \cup L(F)$
  * $L(E.F) = L(E) . L(F)$
  * $L(E^*) = (L(E))^*$

On dit qu'une expression régulière dénote ou décrit un langage.

Exemples

Soit A l'alphabet { a,b,c}

* (a+b+c)* décrit l'ensemble de tous les mots sur A
* a (a+b+c)* dénote l'ensemble des mots qui commencent par la lettre a
* abc (abc)* décrit l'ensemble des répétitions non vides du motif abc
* (a+b+ε) c* décrit l'ensemble des mots qui commencent par la lettre a ou b ou rien et sont suivis d'un nombre positif ou nul de c
* (a+ε) (ba) *(b+ε) décrit l'ensemble des suites alternées de a et de b.

Equivalence ER/LR

Théorème

Un langage est régulier si et seulement si il est dénoté par une expression régulière.

Démonstration de l'équivalence ER/LR

* (⇒) On part de la définition inductive des langages réguliers.
    * On vérifie à la main que chaque élément de la base est décrit par une expression régulière.
    * Par hypothèse d'induction, on suppose que les langages réguliers L et M sont décrits par une expression régulière.
    * On s'assure que par chacune des opérations de l'étape inductive, on obtient un langage décrit par expression régulière.

Démonstration de l'équivalence ER/LR

* (⇐) On part de la définition inductive des expressions régulières.
    * On vérifie à la main que le langage décrit par chaque élément de la base est bien régulier.
    * Par hypothèse d'induction, on suppose que les expressions E et F décrivent des langages réguliers.
    * On s'assure que par chacune des opérations de l'étape inductive, on obtient une expression régulière qui décrit un langage régulier.

Qui utilise les expressions régulières ?

* Les langages du shell (zsh, ksh...) et Unix/linux utilisent la notion d'expression régulière.
  - la commande grep signifie même globally search for regular expression and print
  - find, awk, sed sont d'autres commandes du shell qui permettent de manipuler des expressions régulières

* Emacs a lui aussi ses propres expressions régulières

* Lex d'Unix (et Flex de GNU) est un générateur automatique d'analyseurs lexicaux, c'est-à-dire :

  * il prend en entrée les expressions régulières de plusieurs petits langages réguliers constitutifs des programmes et appelés lexèmes.

  * il retourne un programme en C capable d'associer à chaque mot d'un texte son lexème d'appartenance.

* ...

Langages non réguliers

* Lemme de l'étoile (pumping lemma en anglais ou théorème de gonflement). On démontrera en utilisant les automates le théorème suivant

  * Théorème. Soit L un langage régulier. Il existe un entier n tel que tout mot de L de longueur >=n peut s'écrire sous la forme uxv de façon que l'on ait les propriétés suivantes
    * |ux| <=n
    * x différent du mot vide
    * u(x)* v est inclus dans L.

* Application : le langage $\{a^n b^n, n>=0\}$ n'est pas rationnel.