CSC4251_4252 : Analyse Lexicale

Pascal Hennequin, J. Paul Gibson, Denis Conan

Novembre 2024

Sommaire

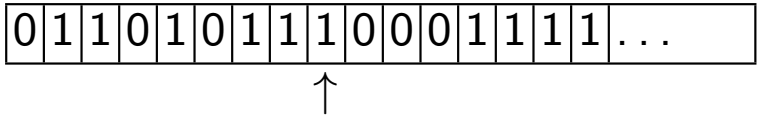
1 De la théorie I

- **Théorie des langages, théorie de la calculabilité**
  - Que peut-on calculer automatiquement ?
  - Que peut-on calculer de manière efficiente ?
  - Systèmes et grammaires formels
- **Hiérarchie de Chomsky-Schützenberger**

Type	Langage / Grammaire	Machine
0	Récursivement énumérable	Machine de Turing
1	Contextuel (*context-sensitive*)	Automate à ressources bornées
2	Algébrique (*context-free*)	Automate à pile
3	Rationnel (*regular*)	Automate fini

1 De la théorie II

Machine de Turing et Calculabilité



- **Bande** : infinie avec tête de lecture
- **Tête** : lecture ou écriture de 0 ou 1,
  et déplacement droite ou gauche
- **Contrôle** :
  - Un état (« variable entière »)
  - Des instructions (« table d'automate ») : [état courant, symbole lu, nouvel état, action=0/1/D/G]

Machine la plus simple pour les calculs les plus généraux

- Thèse de Church-Turing : tout ce qui est calculable automatiquement est calculable avec une machine de Turing
- Il existe des fonctions non calculables
- L'arrêt de la machine de Turing est indécidable
- Théorème de Rice : toute propriété non triviale est indécidable

2 Et en pratique

- **Langage de type 3 = Expressions régulières**
  - Reconnues par automates à états finis
  - Il existe des outils pour reconnaître les expressions régulières en temps $O(n)$
    - Générateurs d'analyseur lexical : lex, flex, JFlex...
    - Aussi : commande `grep`, langage Perl, éditeurs de texte, etc.
- **Langage de type 2 = Grammaires Algébriques**
  - Reconnues par automates à Pile
  - Couvre l'ensemble des langages informatiques
  - Il existe des outils pour faire l'analyse syntaxique en temps $O(n)$
    - `yacc` (Yet Another Compiler Compiler), Bison, CUP, etc.
    - ANTLR, JavaCC, etc.

3 Lexicale *versus* Syntaxique I

- **Séparation usuelle dans les langages naturels**
  - Les mots et leur catégorie (nom, verbe, adjectif...) définis par un dictionnaire
  - Les phrases définies par des règles de construction (grammaire)
- **Séparation théorique dans la hiérarchie de Chomsky**
  - Expressions Régulières *versus* Grammaire algébrique
- **Vocabulaire = Unité lexicale = Lexème**
  - Les mots autorisés dans le langage
- **Catégorie Lexicale = Symbole terminal = Unité lexicale *Token***
  - Ensemble de mots ayant le même comportement dans la grammaire
- **Grammaire**
  - Ensemble de règles définissant les séquences de unités lexicales valides

3 Lexicale *versus* Syntaxique II

- Exemple : langage naturel élémentaire
  - Lexique

POINT	=	.
ARTICLE	=	un |le
VERBE	=	mange |tue
NOM_C	=	lapin |chasseur |gnou
NOM_P	=	Bart |Jeannot |Obi-Wan

  - Syntaxe

$\langle Phrase \rangle$	:=	$\langle SujetVerbeCompl \rangle$ POINT ;
$\langle SujetVerbeCompl \rangle$	:=	$\langle GroupeNom \rangle$ VERBE $\langle GroupeNom \rangle$;
$\langle GroupeNom \rangle$	:=	NOM_P |ARTICLE NOM_C ;

  - Des phrases valides :
    - « Obi-Wan mange un gnou. »
    - « le lapin tue un chasseur. »

4 Analyse lexicale I

- **Reconnaître**
  - Les mots du langage, ou unités lexicales ou lexème
  - Leur catégorie lexicale ou token
- **Qui est lexème ?**
  - Une chaîne de caractères dont la signification n'est pas construite à partir de ces composants.
  - Exemples : mots clés (`for`, `while`, `if`), identificateur (`ma_variable`, `forif`), Opérateurs ($+$, $=$, $++$, $+=$), Ponctuation (`;`, `,`, `.`), Commentaire (`/* coucou */`)
  - Contre-exemple : une constante entière est en général un lexème même si conceptuellement la numération est un processus syntaxique

4 Analyse lexicale II

- **Qui est catégorie lexicale ?**
  - Ensemble de lexèmes non différentiés dans les règles de grammaire
  - Dépend de l'écriture de la grammaire
    - OpBin=$\{+ , - , * , /\}$ ou OpPlus, OpMoins, OpMult, OpDiv ?
- **Unité lexicale** (*Token*)
  - Entier identifiant la catégorie lexicale reconnue (type énuméré)
  - De façon optionnelle :
    - Valeur du lexème dans la catégorie pour les traitements après l'analyse syntaxique
    - Position du lexème dans le fichier source
    - ...

4 Analyse lexicale III

- **Analyseur Lexical (*lexer*, *tokenizer*)**
  - **Entrée** : Texte ASCII, ou Unicode, ou éventuellement binaire
  - **Sortie** : une séquence de symboles terminaux (*tokens*) qui servira d'entrée à un analyseur syntaxique
- Exemple :
  - Entrée : Alpha $+=$ 32 $\times$ bêta $+$ 2 /* comment */
  - Sortie : 1 3 2 4 1 4 2 avec Catégories $=$ {Ident., Littéral, Affect., OpBin.}
  - Ignore : blancs, commentaires, etc.

5 Spécification du Vocabulaire

- **Par énumération**
  - Vocabulaire fini
  - Structure de dictionnaire

$$\langle Séparateur \rangle : \text{‘,’} \mid \text{‘.’} \mid \text{‘:’} \mid \text{‘;’} ;$$
$$\langle Opérateur \rangle : \text{‘+’} \mid \text{‘-’} \mid \text{‘*’} \mid \text{‘/’} ;$$

- **Par règle de production récursive**
  - Vocabulaire infini
  - Reconnaissance complexe
  - Par exemple, BNF

$$\langle Entier \rangle := \langle Chiffre \rangle \mid \langle Entier \rangle \langle Chiffre \rangle ;$$
$$\langle Nom \rangle := \langle Lettre \rangle \mid \langle Nom \rangle \mid \langle Nom \rangle \langle Lettre \rangle ;$$
$$\langle Chiffre \rangle := \text{‘0’} \mid \text{‘1’} \mid \ldots \mid \text{‘9’} ;$$
$$\langle Lettre \rangle := \text{‘a’} \mid \text{‘b’} \mid \ldots \mid \text{‘Z’} ;$$

- **Par expression régulière**
  - Vocabulaire infini
  - Formalisme simple
  - Reconnaissance automatisable et efficace

$$\langle Entier \rangle = [0\text{-}9]+$$
$$\langle Nom \rangle = [a\text{-}za\text{–}Z][\_a\text{-}zA\text{-}Z]*$$

6 Expression régulière I

Définition

Constantes	ϵ Caractère	Mot vide singleton
Trois opérateurs	$r_1 \mid r_2$ $r_1 r_2$ $r*$	Alternative ou union Concaténation (implicite) Fermeture de Kleene (répété $N \geq 0$ fois)
Parenthèses	()	Gestion des priorités

Exemple

- « a* a (b | c)* a* » sur $\Sigma = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$
  - Mots commençant par au moins un « a », suivi de « b » ou « c » en nombre quelconque, et éventuellement terminé par des « a »
  - Soit « a », « aa », « ab », « ac », « aaa », « aab », « aac », « aba », « abb », « abc », « abba », etc.

6 Expression régulière II

- **Classes de caractères** : reconnaît un symbole parmi un ensemble

.	Un caractère quelconque sauf fin de ligne
[xyzT]	Un caractère de l'énumération, soit x | y | z | T
[...A-F...]	Idem avec intervalle
[^...]	Un caractère hors de l'énumération

Attention ! L'écriture « A-F » est déconseillée car dépendante du jeu de caractères.

- **Répétitions**

r **?**	r répétée 0 ou 1 fois, soit $r \mid \epsilon$
r+	r répétée $n > 0$ fois, soit r r*
r{**k**}	r répétée k fois
r{**k,l**}	r répétée entre k et l fois

- **Délimiteurs de contexte**

^r	r en début de ligne
$r$$	r en fin de ligne

6 Expression régulière III

- **Exemples**

`[aeiouy]`	Une voyelle
`[A-Za-z]+`	Un mot alphabétique non vide
`^[ \t]+$`	Une ligne blanche
`//.*`	Un commentaire C++
`\"[^"]*\"`	Une constante chaîne
`[0-9]+ (\.[0-9]*)? ([eE][+-]?[0-9]+)?`	Une constante flottante
`0 | [1-9] ( [0-9_]* [0-9] )?`	Une constante décimale JAVA