CSC4251_4252 : Grammaire Algébrique

Pascal Hennequin, J. Paul Gibson, Denis Conan

Novembre 2024

Sommaire

1 Grammaire I

(rappel) Exemple de 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 ;
$\langle SujetVerbeCompl \rangle$	:=	$\langle GroupNom \rangle$ 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. »

1 Grammaire II

Grammaire algébrique (*Context free*)

$G = (V_T, V_N, S, P)$ avec
- V_T l'ensemble des symboles terminaux ou constantes ou *tokens*
- V_N l'ensemble des symboles non-terminaux ou variables
- V_T et V_N sont finis et disjoints
- $S \in V_N$ le symbole de départ ou axiome
- $P \subset V_N \times (V_T \cup V_N)*$ l'ensemble des règles de production : par exemple, $v := s_1 s_2 \ldots s_n$

- Le membre gauche v est un symbole non-terminal
- Le membre droit est une concaténation de symboles s_i terminaux ou non-terminaux
- Les productions peuvent être regroupées en utilisant l'alternative

$$\begin{array}{l} v := s_1 s_2 \ldots s_n \ ; \\ v := s'_1 s'_2 \ldots s'_p \ ; \end{array} \quad \equiv \quad v := s_1 s_2 \ldots s_n \mid s'_1 s'_2 \ldots s'_p \ ;$$

1 Grammaire III

Utilisations

En **production**

- En partant de l'axiome, substitutions d'un membre gauche d'une production par un membre droit
- On génère l'ensemble des phrases valides pour la grammaire
- Par exemple :

  $\langle Phrase \rangle$

  $\rightsquigarrow \langle SujetVerbeCompl \rangle$ POINT

  $\rightsquigarrow \langle GroupeNom \rangle$ VERBE $\langle GroupeNom \rangle$ POINT

  $\rightsquigarrow$ NOM_P VERBE ARTICLE NOM_C POINT

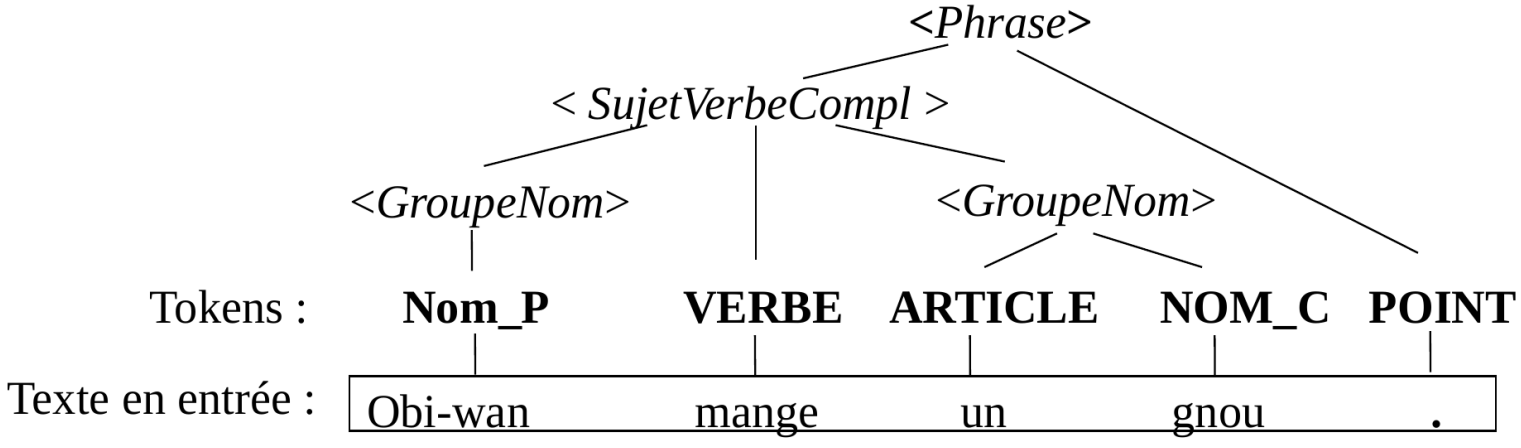
  $\rightsquigarrow$ Bart tue un lapin .

En **reconnaissance, *parsing***

- Réductions d'une phrase en remplaçant des membres droits par les membre gauches
- La phrase est valide si les réductions terminent sur l'axiome

2 Arbre Syntaxique

- **Décrit la reconnaissance d'une phrase donnée conformément à une grammaire**
- **Arbre étiqueté :**
  - Feuilles = *tokens* = mots de V_T
  - Nœud = membre gauche de production = symbole de V_N
  - Fils = membres droits de la production
  - Racine = axiome



3 Où est l'étoile ?

- **Problème !**
  - Les grammaires algébriques contiennent la concaténation et l'alternative mais pas la fermeture de Kleene
  - Selon Chomsky, les langages rationnels sont algébriques
  - Comment réaliser l'étoile avec une grammaire algébrique ?
- **Exercice**
  - Soit L un langage décrit par le symbole L, comment définir le langage $L*$ avec une grammaire algébrique ?
    - N.B. : si L est rationnel, $L*$ est rationnel et doit donc être algébrique

3.1 Où est l'étoile ? (solutions)

Solution(s)

- « La récursivité contient l'étoile »

Récursif Droit	Récursif Gauche
Lstar := ϵ | L Lstar ;	Lstar := ϵ | Lstar L ;

ou aussi !

Lstar := ϵ | L | Lstar Lstar

- N.B. : dans les outils, le mot vide ϵ s'écrira « /* mot vide */ »

3.1.1 Solution Bis

- **Solution(s)**
  - « La récursivité contient l'étoile »

Récursif Droit	Récursif Gauche
Lstar := ϵ	Lstar := ϵ
| L Lstar	| Lstar L
;	;

ou aussi !

Lstar := ϵ | L | Lstar Lstar

$\Longleftarrow$ **très mauvaise solution pour nous (cf. ambiguïté plus tard)**

  - Les 2 solutions (récursivités droit/gauche) seront necessaires pour nous, mais on s'interdit la solution « quadratique »

4 Exemple

Ébauche d'un langage informatique

- Vocabulaire terminal

```
TYPE  = int | char | float
IDENT = [a-zA-Z] [a-zA-Z0-9]*
INT   = .? [0-9]+
FLOAT = .? [0-9]+ ( . [0-9]* )?
```

- Règles de production

$$
\begin{array}{lll}
\langle \textit{Program} \rangle & := & \langle \textit{Definitions} \rangle \ \langle \textit{Functions} \rangle \ ; \\
\langle \textit{Definitions} \rangle & := & \langle \textit{TypeVar} \rangle \mid \langle \textit{Definitions} \rangle \ \langle \textit{TypeVar} \rangle \ ; \\
\langle \textit{TypeVar} \rangle & := & \textbf{TYPE} \ \langle \textit{VarList} \rangle \ \textbf{';'} \ ; \\
\langle \textit{VarList} \rangle & := & \langle \textit{Var} \rangle \mid \langle \textit{VarList} \rangle \ \textbf{','} \ \langle \textit{Var} \rangle \ ; \\
\langle \textit{Var} \rangle & := & \langle \textit{ScalarVar} \rangle \mid \langle \textit{ArrayVar} \rangle \ ; \\
\langle \textit{ScalarVar} \rangle & := & \textbf{IDENT} \mid \textbf{IDENT '='} \ \langle \textit{Number} \rangle \ ; \\
\langle \textit{ArrayVar} \rangle & := & \text{IDENT } \textbf{'[' INT ']'} \mid \langle \textit{ArrayVar} \rangle \ \textbf{'[' INT ']'} \ ; \\
\langle \textit{Number} \rangle & := & \textbf{INT} \mid \textbf{FLOAT} \ ;
\end{array}
$$

- Exemple de production

```
int i, j, i42=-1 ;
int a[-42], b[3][1][4] ;
float pi=3 ;
int yy=0.3 ;
```

5 Grammaire régulière I

- **Une grammaire algébrique peut décrire un langage rationnel**
  - Indécidable dans le cas général (non-déterministe)
  - Mais certaines productions simples génèrent des langages rationnels
- **Grammaire Linéaire Gauche (resp. Droite)**
  - Toute production contient au plus un symbole non-terminal en membre droit
  - Ce symbole non-terminal est le premier (resp. dernier) symbole en membre droit

Linéaire Gauche	Linéaire Droit
$v_i := v_j t_1 t_2 \ldots t_n$;	$v_i := t_1 t_2 \ldots t_n v_j$;

- **Les grammaires linéaires gauches (resp. droite) engendrent les langages rationnels**

5 Grammaire régulière II

Exemples

1) $V_N = \{S, A\}$, $V_T = \{\mathrm{a}, \mathrm{b}\}$
Expressions régulières pour A et S ?

S	:=	**b** A | **a** S
A	:=	**b** A | **a**

2) $V_N = \{S\}$, $V_T = \{\mathrm{LETTRE}, \mathrm{CHIFFRE}\}$
LETTRE=[a-z], CHIFFRE=[0-9]
Expression régulière pour S ?

S	:=	LETTRE | S LETTRE | S CHIFFRE ;

3) $V_N = \{S\}$, $V_T = \{\mathrm{a}, \mathrm{b}\}$
Linéaire ? Régulier ? Expression régulière pour S ?

S	:=	**a b** | **a** S **b** ;

5.1 Grammaire régulière (solutions)

1)

$$\begin{array}{lll} S & := & \mathbf{b}\ A \mid \mathbf{a}\ S \\ A & := & \mathbf{b}\ A \mid \mathbf{a} \end{array}$$

A = b*a

S = a*bA = a*b+a

$L(S) = \{ba, aba, aaba, abba, aaaba...\}$

2)

$$S := \text{LETTRE} \mid S\ \text{LETTRE} \mid S\ \text{CHIFFRE}\ ;$$

S = [a-zA-Z] [a-zA-Z0-9]*

3)

$$S := \mathbf{a}\ \mathbf{b} \mid \mathbf{a}\ S\ \mathbf{b}\ ;$$

Linéaire « milieu », ni droite ni gauche

Non régulier, $L(S) = \{a^n b^n, n > 0\}$

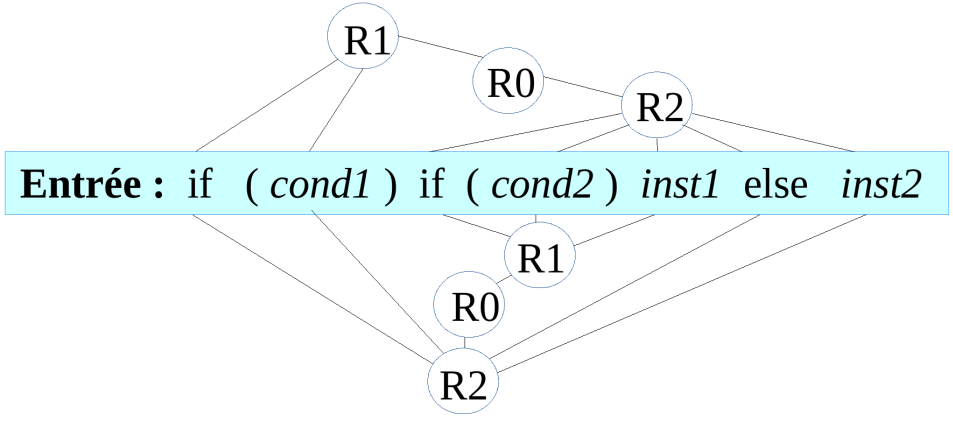
Linéaire « et droite et gauche »

$$\begin{array}{lll} S & := & \mathbf{a}\ \mathbf{b}\ ; \\ S & := & \mathbf{a}\ ; \\ T & := & S\ \mathbf{b}\ ; \end{array}$$

6 Grammaire ambiguë I

- **Plusieurs arbres de syntaxe possibles pour une entrée**
  - Ambiguë $\Longrightarrow$ indéterminisme dans l'analyse syntaxique
  - En général, arbres différents $\Longrightarrow$ sémantiques différentes
- **Exemple : Instruction IF**

(R0)	⟨*Inst*⟩	:=	⟨*Inst_if*⟩ | ... ;
(R1)	⟨*Inst_if*⟩	:=	"if" "(" ⟨*Condition*⟩ ")" ⟨*Inst*⟩ ;
(R2)		|	"if" "(" ⟨*Condition*⟩ ")" ⟨*Inst*⟩ ⟨*else*⟩ ⟨*Inst*⟩ ;



6 Grammaire ambiguë II

- **Grammaire d'opérateur** (pas de mot vide, pas de non-terminaux voisins)
  - Généralement ambiguë
    2+3+4 = (2+3)+4 ou 2+(3+4) ?
    a=1 ; a + ++a + ++a = ?
    2-3-4 = (2-3)-4 ou 2-(3-4) ?
    2+3*4 = (2+3)*4 ou 2+(3*4) ?
  - Solutions
    - Réécriture de la grammaire en ajoutant des non-terminaux
    - Gestion de priorités sur les règles dans l'analyse syntaxique

```
⟨Expr⟩   :=   ⟨Expr⟩ '+' ⟨Expr⟩
          |   ⟨Expr⟩ '-' ⟨Expr⟩
          |   ⟨Expr⟩ '*' ⟨Expr⟩
          |   '-' ⟨Expr⟩
          |   '(' ⟨Expr⟩ ')'
          |   INT | SYMBOL ;
```

```
/* Version non ambiguë gauche */
⟨Expr⟩     :=   ⟨Term⟩
            |   ⟨Expr⟩ '+' ⟨Term⟩
            |   ⟨Expr⟩ '-' ⟨Term⟩ ;
⟨Term⟩     :=   ⟨Facteur⟩
            |   ⟨Term⟩ '*' ⟨Facteur⟩ ;
⟨Facteur⟩  :=   '-' ⟨Regle⟩
            |   '(' ⟨Expr⟩ ')'
            |   INT | SYMBOL ;
```

7 Spécification avec BNF

- **Syntaxes Multiples : BNF, EBNF, ABNF**
- **Regroupe grammaire algébrique et expression régulière**
  - Terminaux = chaînes de caractères constantes
  - Répétitions et groupages dans les règles de production
- **Éléments de syntaxe**
  - Définition avec `=` ou `::=` et éventuellement `;` en fin
  - Non-Terminaux entre `<>` ou pas
  - Terminaux entre `" "` ou `' '` ou pas
  - Groupage avec `( )`
  - Répétition Kleene entre `{ }` ou `*terme`
  - Répétition numérique `n*`terme, `n*m`terme, etc.
  - Optionnel (répétition 0 ou 1) entre `[]`
  - Alternative avec `|` ou `/`
  - Concaténation avec `,` ou pas
  - Commentaires `(* ... *)` ou `;` ...

8 Exemple de spécification avec BNF

- Expressions Arithmétiques en BNF/EBNF

⟨*Expression*⟩ := ⟨*Terme*⟩ { (+ | -) ⟨*Terme*⟩ } ;
⟨*Terme*⟩ := ⟨*Facteur*⟩ { (* | /) ⟨*Facteur*⟩ } ;
⟨*Facteur*⟩ := ⟨*Nombre*⟩ | ⟨*Variable*⟩ | “(” ⟨*Expression*⟩ “)” ;
⟨*Nombre*⟩ := [-] ⟨*Chiffre*⟩ {⟨*Chiffre*⟩} ;
⟨*Chiffre*⟩ := 0 | 1 | ... | 9 ;
⟨*Variable*⟩ := ⟨*Lettre*⟩ { ⟨*Lettre*⟩ | ⟨*Chiffre*⟩ | - } ;
⟨*Lettre*⟩ := a | b | ... | Z ;

- ABNF RFC822

```
date-time   =   [ day "," ] date time
day         =   "Mon" / "Tue" / "Wed" / "Thu" / "Fri" / "Sat" / "Sun"
date        =   1*2DIGIT month 2DIGIT    ; dd mm yy
month       =   "Jan" / "Feb" / "Mar" / "Apr" / "May" / "Jun"
                / "Jul" / "Aug" / "Sep" / "Oct" / "Nov" / "Dec"
time        =   hour zone    ; hh :mm :ss zzz
hour        =   2DIGIT " :" 2DIGIT ["  :" 2DIGIT]
zone        =   "GMT" / ..... / ( ("+" / "-") 4DIGIT )
DIGIT       =   ¡any ASCII decimal digit¿    ; ( decimal 48.- 57.)
```