La génération des langages

1

Origine des grammaires

- Tentatives de formalisation du langage naturel
- <u>But</u> : décrire précisément les règles permettant de construire des phrases syntaxiquement correctes d 'une langue
- Échec de la linguistique mais réussite pour des langues plus simples = langages informatiques

2

Exemple pour la linguistique

- Phrase :Sujet Verbe
- Sujet : Pronom
- Pronom : il | elle
- Verbe : dort | écoute

Règles

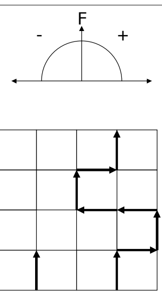
Symboles terminaux

- *Avec ces 4 règles, on peut alors construire les phrases:*
  - il écoute
  - il dort
  - elle écoute
  - elle dort

3

Images de synthèse

- Remplacer (réécrire) un motif
  - Base : F+F+F+F
  - Règle : F→F+F-F-FF+F+F-F
- A essayer sur du papier quadrillé…
  (3 applications de la règle)
  ou à programmer
- C'est l 'exemple de LOGO

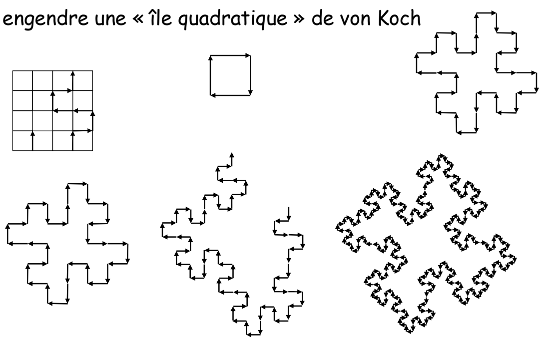


F→F +F-F -FF +F +F-F

4

F+F+F+F; F→F+F-F-FF+F+F-F

engendre une « île quadratique » de von Koch

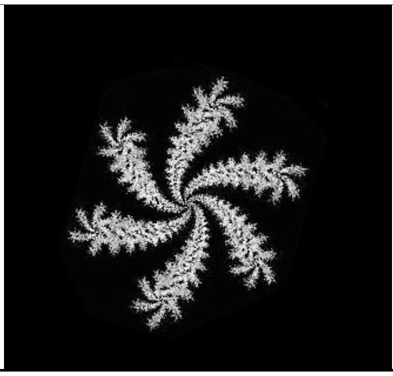


5

Courbe de von Koch

6

Courbe de von Koch



7

En informatique

- DecimalNumeral:
  0
  NonZeroDigit Digits_{opt}
- Digits:
  Digit
  Digits Digit
- Digit:
  0
  NonZeroDigit
- NonZeroDigit: one of
  1 2 3 4 5 6 7 8 9

définition d'un décimal java

8

Grammaire informatique

- Ensemble de règles de la forme
  - Digit:
    0
    NonZeroDigit
- Décrit la manière de construire le langage
- Inversement, un automate nous permet de reconnaître les mots du langage

9

Forme de Backus-Naur BNF

- Description analytique d'une grammaire informatique
- Utile à l 'analyse syntaxique (1ere étape de compilation)
- Catégories syntaxiques : suite de mots commençant par une majuscule sans espace
  - **OpérateurAdditif, NonZeroDigit, Digit**
- Alternatives : Une barre verticale sépare les alternatives
  - **Digit: 0|NonZeroDigit**
- Mots clés : en gras
  - **class, float, switch, boolean**
- Éléments optionnels : Les crochets encadrent les éléments optionnels
  - **DecimalNumeral: 0| NonZeroDigit [Digits]**
- Éléments répétés : encadrés par des accolades
  - **Identificateur : Lettre {Lettre | Chiffre}**

10

Flottants JAVA BNF

- FloatingPointLiteral:
  Digits . [Digits] [ExponentPart] [FloatTypeSuffix] |
  . Digits [ExponentPart] [FloatTypeSuffix] |
  Digits ExponentPart [FloatTypeSuffix] |
  Digits [ExponentPart] FloatTypeSuffix
- ExponentPart: ExponentIndicator SignedInteger
- ExponentIndicator: e | E
- SignedInteger: [Sign] Digits
- Sign: +| -
- FloatTypeSuffix: f | F | d | D

11

Les grammaires formelles

- Principe de base : ensemble de règles qui engendrent les mots d'un langage
- sortes de règles de réécriture
  - Une suite de symboles peut être remplacée par une nouvelle suite de symboles
  - Les mots engendrés sont ceux obtenus en appliquant les règles à partir d'un symbole de départ

12

Définition

- Une grammaire $G=(N,T,R,S)$
  - N : ensemble des symboles non terminaux
  - T : ensemble des symboles terminaux
  - $R\subseteq(N\times(N\cup T)^*)$: ensemble fini de règles de réécriture, les productions
  - $S\in N$: symbole de départ également appelé axiome
- Les mots engendrés sont ceux obtenus en appliquant les règles à partir du symbole de départ et qui ne contiennent plus que des symboles terminaux
- **Exemple** :

$G=(N=\{S,A,B\},T=\{0,1\},R=\{S\to ASB;\ S\to\varepsilon;\ A\to 0;\ B\to 1\},S)$

13

Conventions d'écriture

- Les non terminaux sont représentés par des majuscules
- Les terminaux sont représentés par des minuscules
- Les productions $(X, \alpha)\in R$ sont notées $X\to\alpha$
- L'axiome est le plus souvent noté S (start)
- Les symboles de $N\cup T$ sont appelés les symboles grammaticaux.
- Il sont représentés par les lettres minuscules grecques : $\alpha,\beta,\gamma,\ldots$
- Si $X\to\alpha_1, X\to\alpha_2,\ldots, X\to\alpha_k\in R$ avec X comme partie gauche, on peut écrire

$$X\to\alpha_1|\alpha_2|\ldots|\alpha_k$$

14

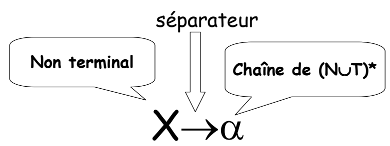
Observations

- Les terminaux sont les symboles de base à partir desquels les mots sont formés; on les appelle des unités lexicales
- Les non terminaux sont des variables syntaxiques qui dénotent un ensemble de chaînes qui aident à la spécification du langage
- Exemple :
  - Lettre $\to$ A|B|C|…|Z|a|b|…|z
  - Chiffre $\to$ 0|1|…|9
  - Identificateur $\to$ Lettre {Lettre | Chiffre}
- Les terminaux sont les lettres et les chiffres
- Les non terminaux sont {Lettre, Chiffre, Identificateur} qui aident à la compréhension du langage

15

Observations

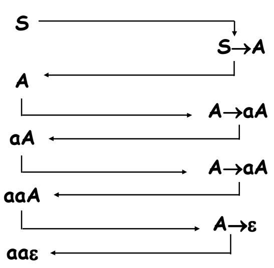
- Les productions spécifient la manière dont les terminaux et les non terminaux peuvent être combinés pour former des chaînes.
- Chaque production $X\to\alpha$ consiste en



16

Exemple

- $G=(N,T,R,S)$
  - $N=\{S,A,B\}$
  - $T=\{a,b\}$
  - $R=\{S\to A|B,\ A\to aA|\varepsilon,\ B\to bB|\varepsilon\}$
- Définit une grammaire



Partant de S on a pu engendrer le mot aa

17

Une vieille connaissance

- $G=(N,T,R,S)$
  - $N=\{S\}$
  - $T=\{a,b\}$
  - $R=\{S\to\varepsilon,\ S\to aSb\}$
- Définit une grammaire pour $\{a^nb^n : n\geq 0\}$ non rationnel

S	$\to$aSb	$\to$aaSbb	$\to$aaaSbbb	$\to$aaaaSbbbb	$\to$aaaaaSbbbbb
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
ε	ab	aabb	aaabbb	aaaabbbb	

18

Dérivation directe

- Un mot m de $(N \cup T)^*$ se dérive directement en un mot m' de $(N \cup T)^*$
  ($m \rightarrow m'$) si :
  - $m=uXv$ pour $X \in N$ et $u,v \in (N \cup T)^*$
  - $m'=uwv$ pour $u,v,w \in (N \cup T)^*$
  - Et s'il existe une production $X \rightarrow w$ dans R
- formalise le fait d'appliquer une fois une production en réécrivant un non terminal en accord avec une production ayant pour membre gauche ce non terminal
- **Exemple**: 3 dérivations directes pour la grammaire de règles $S \rightarrow \varepsilon | aSb$
  - $S \rightarrow aSb$
  - $aSb \rightarrow aaSbb$
  - $aaaSbbb \rightarrow aaabbb$

Dérivation en k étapes

- m, mot de $(N \cup T)^*$ se dérive en m', mot de $(N \cup T)^*$ ($m \rightarrow^* m'$) si
  - Il existe k un entier
  - $m_0, m_1, \ldots, m_k$ des mots de $(N \cup T)^*$ tels que
  - m_{i+1} se dérive directement de m_i , $0<i<k+1$
  - $m_0=m$ et $m_k=m'$
- formalise le fait d'appliquer successivement k productions
- **Exemple** : pour la grammaire dont la règle est $S \rightarrow \varepsilon | aSb$
  - $S \rightarrow^* aaaSbbb$ est une dérivation en 3 étapes
  - $S \rightarrow^* aaabbb$ est une dérivation en 4 étapes

Remarque

- Dans la définition rien ne dit que les mots qui se dérivent directement les uns des autres soient uniques
- <u>Exemple</u> : $S \rightarrow \varepsilon | aSb | ab$, ab a 2 dérivations différentes :
  - $S \rightarrow ab$ (directe)
  - $S \rightarrow aSb \rightarrow ab$ (indirecte)

Mots engendrés

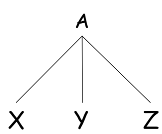
- Les mots engendrés par une grammaire $G=(N,T,R,S)$ sont les mots $m \in T^*$ (uniquement composés de symboles terminaux) qui peuvent être dérivés depuis l'axiome :
  $$S \rightarrow^* m$$
- Exemple : pour la grammaire de règle $S \rightarrow \varepsilon | aSb$
  aaabbb est un mot engendré par la grammaire

Langage engendré

- La grammaire G engendre un langage
  $$L(G)=\{m \in T^* : S \rightarrow^* m\}$$
- ensemble des mots engendrés par G en dérivant l'axiome
- Pour les grammaires de la forme $X \rightarrow \alpha$, $X \in N$ et $\alpha \in (N \cup T)^*$ (grammaire algébrique), on engendre des langages algébriques.
- <u>Exemple</u> : $G=(N,T,R,S)$
  - $N=\{S\}$
  - $T=\{a,b\}$
  - $R=\{S \rightarrow \varepsilon, S \rightarrow aSb\}$
- Définit une grammaire du langage $\{a^n b^n : n \geq 0\}$

Arbre syntaxique

- Un arbre syntaxique illustre graphiquement la manière dont l'axiome se dérive en une chaîne du langage
- Si le non terminal A définit la production $A \rightarrow XYZ$, un arbre syntaxique possède 1 nœud interne et trois fils étiquetés X, Y et Z de gauche à droite



Arbre syntaxique (définition)

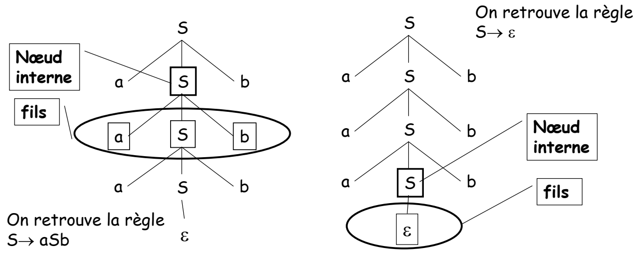
- La racine est l 'axiome
- Chaque feuille est soit ε soit un terminal
- Chaque nœud interne est un non terminal
- Si A est l'étiquette d 'un nœud interne de fils (de gauche à droite) $X_1, X_2, \ldots, X_n$ alors

$$A \to X_1 X_2 \ldots X_n$$

est une production de la grammaire

Arbre syntaxique: $\{a^n b^n : n \geq 0\}$

- $S \to \varepsilon | aSb$, arbre syntaxique pour aaabbb

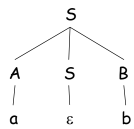


Avec l'arbre syntaxique d'un mot dont la dérivation contient toutes les règles, on peut retrouver l'ensemble des règles de la grammaire

Arbre & dérivations

- Dérivations $\to$ arbre = représentation graphique de la dérivation
- Dans la dérivation ainsi obtenue, on fait disparaître les choix de l'ordre d'application des règles

- Pour la grammaire
  1. $S \to ASB$
  2. $S \to \varepsilon$
  3. $A \to a$
  4. $B \to b$
- On peut appliquer les règles selon différents ordres



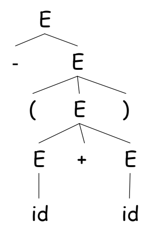
- $S \to ASB \to aSB \to aB \to ab$
  - 1;3;2;4

  Ou

- $S \to ASB \to AB \to Ab \to ab$
  - 1;2;4;3

Dérivation➔ arbre

- A partir d'une dérivation, on peut construire l'arbre syntaxique :
- Grammaire
  - $E \to E+E|E*E|(E)|-E|id$
- Mot engendré
  - -(id+id)



$E \to -E \to -(E) \to -(E+E) \to -(id+E) \to -(id+id)$

Analyse

Données : G une grammaire et m un mot

Calcul : trouver (s'il en existe) les dérivations de G qui engendrent m

Grammaire
- $E \to E+E|E*E|(E)|-E|id$
- Mot engendré
  - -(id+id)

- -(id+id)
- -(id+id)
- -(id+id)
- -(id+id)

$E \to -E$

$E \to (E)$

$E \to E+E$

$E \to id$

$E \to -E \to -(E) \to -(E+E) \to -(id+E) \to -(id+id)$

Attention

- Pour une grammaire donnée, on peut avoir plus d'une dérivation qui engendre un même mot
  - Soit avec des arbres syntaxiques différents
  - Soit au sein du même arbre syntaxique
- On passe ainsi de l'arbre syntaxique aux dérivations

Exemple

- Grammaire

  $E \rightarrow E+E|E*E|(E)|-E|id$
- Mot engendré

  -(id+id)

$E \rightarrow -E \rightarrow -(E) \rightarrow -(E+E) \rightarrow -(id+E) \rightarrow -(id+id)$

$E \rightarrow -E \rightarrow -(E) \rightarrow -(E+E) \rightarrow -(E+id) \rightarrow -(id+id)$

deux dérivations pour un même arbre syntaxique

31

Exemple

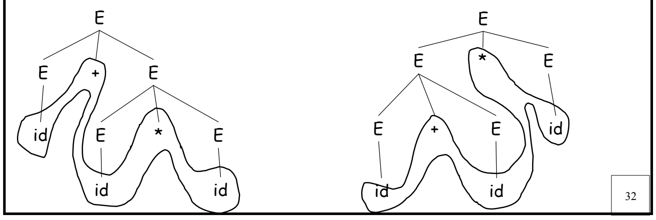
- Grammaire

$E \rightarrow E+E|E*E|(E)|-E|id$
- Mot engendré id+id*id

Deux arbres syntaxiques différents engendrant le même mot

$E \rightarrow E+E \rightarrow id+E \rightarrow id+E*E \rightarrow id+id*E \rightarrow id+id*id$

$E \rightarrow E*E \rightarrow E*id \rightarrow E+E*id \rightarrow id+E*id \rightarrow id+id*id$



32

Dérivations gauches et droites

- Comme on peut trouver plusieurs dérivations à partir d 'un même arbre syntaxique, on parle alors
  - De <u>dérivation gauche</u>
  - Chaque étape d'une dérivation gauche s 'écrit

    $wA\gamma \rightarrow w\delta\gamma$
  - De <u>dérivation droite</u>
  - Chaque étape d'une dérivation droite s'écrit

    $\gamma Aw \rightarrow \gamma\delta w$
  - Pour w un mot formé de terminaux
  - γ une chaîne de symboles grammaticaux (de $(N \cup T)^*$)
  - et $A \rightarrow \delta$ est la production utilisée

33

Dérivation gauche resp droite

- $S \rightarrow aAS|a$
- $A \rightarrow SbA|SS|ba$
- $S \rightarrow aAS$

  $\rightarrow aSSS$

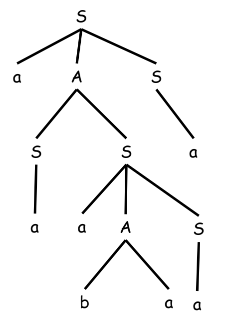
  $\rightarrow aaSS$

  $\rightarrow aaaASS$

  $\rightarrow aaabaSS$

  $\rightarrow aaabaaS$

  $\rightarrow aaabaaa$



- $S \rightarrow aAS$

  $\rightarrow aAa$

  $\rightarrow aSSa$

  $\rightarrow aSaASa$

  $\rightarrow aSaAaa$

  $\rightarrow aSabaaa$

  $\rightarrow aaabaaa$

34

Ambiguïté

- Problème
  - G une grammaire
  - G est-elle ambiguë ?
- Pour le résoudre, il suffit de trouver un mot qui admet au moins deux dérivations gauches (resp. droites) différentes
- Dans le contexte de la compilation,
  - Soit on essaye d'éviter les grammaires ambiguës,
  - Soit on ajoute des règles pour résoudre les problèmes de conflit liés à l'ambiguïté de la grammaire.
- Pour qu'il y ait unicité de l'analyse

35