

Langages hors-contexte

Damien Nouvel



Plan

1. Origines
2. Définitions
3. Dérivations
4. Simplification

Langages réguliers et grammaires

- ▶ Les langages **réguliers** (ou rationnels) reconnaissent
 - Des mots issus de lexiques
 - Des formes normales (chiffres, dates, etc.)

⇒ Ils sont **insuffisants** pour

- Les formes récursives
- Les structures de type $\{a^n b^n\}$
- La recherche de la syntaxe dans le langage naturel
- L'analyse de programmes

⇒ La machine de Turing peut faire mieux que ça !

Historique des grammaires

- ▶ Hiérarchie de **Chomsky** (1956)
 - Analyse du langage **naturel** (morphologie, syntaxe)
 - Grammaire formelles
 - Type 0 : grammaires générales (machine de Turing)
 - Type 1 : grammaires contextuelles (automates lin. bornés)
 - Type 2 : grammaires **hors-contexte** (automates à pile)
 - Type 3 : grammaires **régulières** (automates à états finis)
- ▶ Langages de programmation **artificiels** (compilation)
 - Compilateurs mot-à-mot (assembleur)
 - Expressions mathématiques : **arbres** d'analyse
 - Compilateurs modernes
 - Analyse **lexicale** (scanner)
 - Analyse **syntaxique** (parser)

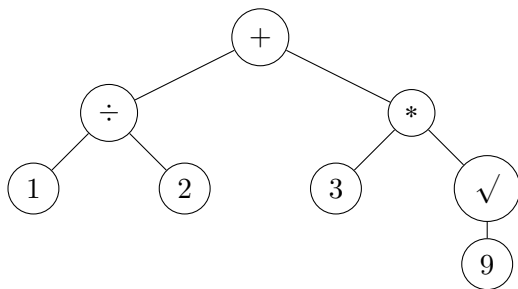
⇒ Utilisation des méthodes linguistiques pour la programmation

Plan

1. Origines
2. Définitions
3. Dérivations
4. Simplification

Arbre d'expression

- ▶ Analyse de l'expression mathématique $\frac{1}{2} + 3 * \sqrt{9}$
 - Priorité des opérateurs : $(\frac{1}{2} + (3 * \sqrt{9}))$
 - Arbre d'expression :



⇒ Chaque branche est aussi une expression valide : $\frac{1}{2}$, $3 * \sqrt{9}$

Backus-Naur Form

- ▶ Langage de programmation Algol 1960 (J. Backus et P. Naur)
- ▶ Règles de description du langage

- Pour la programmation

```

<ifstruct> ::= "if" "(" <test> ")" "{" <block> "}"
<test> ::= <var> "==" <num> | <var> "&&" <var> | ...
<block> ::= <inst> | <inst> <block>
  
```

⇒ **Vérification** de la syntaxe des programmes

⇒ **Récursivité** dans les règles

- Pour la linguistique

```

<E> ::= <GN> <GV>
<GN> ::= <DET> <NC> | <DET> <ADJ> <NC>
<GV> ::= "marche" | "dort"
<DET> ::= "le"
<NC> ::= "chien" | "chat"
<ADJ> ::= "petit" | "gros"
  
```

Définition des grammaires hors-contexte

⇒ **Reconnaissance / génération** de langages

▶ Quadruplet $G = (T, N, R, S)$

- T : symboles terminaux

⇒ Les **mots** possibles des énoncés

- N : symboles non-terminaux

⇒ Groupes de mots intéressants (GN, GV, etc.)

- $R \subset N \times (N \cup T)^*$: règles

⇒ Déterminent la composition des groupes de mots

⇒ **Notations**

- Une règle s'écrit $A \rightarrow \alpha$ avec $A \in N$ et $\alpha \in (N \cup T)^*$

- Des règles $A \rightarrow \alpha$ et $A \rightarrow \beta$ s'écrivent $A \rightarrow \alpha|\beta$

- $S \in N$: axiome (symbole de départ)

⇒ Représente l'énoncé

Exemple

► Expressions mathématiques

- $N = \{S, E\}$ et $T = \{+, *, \div, \sqrt{}, (,), 1, 2, 3, \dots\}$

- Règles :

$$S \rightarrow E$$

$$E \rightarrow E + E$$

$$E \rightarrow E * E$$

$$E \rightarrow E \div E$$

$$E \rightarrow (E)$$

$$E \rightarrow \sqrt{E}$$

$$E \rightarrow 1|2|3\dots$$

► Une **dérivation** possible :

$$S \rightarrow E \rightarrow E + E \rightarrow E \div E + E \dots \rightarrow 1 \div 2 + 3 * \sqrt{9}$$

Exercices

- ▶ Définissez des grammaires qui reconnaissent
 - Un code postal
 - Le langage des dates au format $jj/mm/aaaa$
 - Une plaque d'immatriculation française
 - Un nombre entier naturel (positif)
 - Un nombre réel
 - Le langage régulier ab^*cd^*
 - Le langage $\{a^n c^n, n > 0\}$
 - Les palindrômes
 - Une expression de parenthèses $()$ et crochets $[]$
 - Le langage $\{a^n bc^m d, n > 0, m \geq n\}$

Plan

1. Origines
2. Définitions
3. Dérivations
4. Simplification

Dérivation

- ▶ Opérations qui **gènèrent** le langage pour une grammaire
- ▶ Un mot $\alpha \in (N \cup T)^*$ se **dérive** en un mot $\beta \in (N \cup T)^*$ si
 - α se décompose en $\alpha_1 A \alpha_2$ avec $A \in N$
 - β se décompose en $\alpha_1 \gamma \alpha_2$ avec $\gamma \in (N \cup T)^*$
 - $A \rightarrow \gamma \in R$ (c'est une règle)
- ▶ Exemple : $E + E \div E \rightarrow E + E * E \div E$
 - $\alpha_1 = E +$
 - $\alpha_2 = \div E$
 - $A = E$
 - $\gamma = E * E$
 - $E \rightarrow E * E \in R$

Suite de dérivations

► Par **transitivité**

- Chaîne de dérivations $\alpha \rightarrow \beta \cdots \rightarrow \gamma = \alpha \xrightarrow{*} \gamma$
- Fermeture transitive, clôture (cf étoile de Kleene)
- Si $\gamma \in (N \cup T)^*$ alors γ est une **proto-phrase** de G

► **Ordre** des dérivations

- Possibilité d'analyses pour $1 + 2 + 3$
 - Dérivation gauche : réécrit le non-terminal le plus à gauche
 $E \rightarrow E + E \rightarrow 1 + E \rightarrow 1 + E + E \rightarrow 1 + 2 + E \rightarrow 1 + 2 + 3$
 - Dérivation droite : réécrit le non-terminal le plus à droite
 $E \rightarrow E + E \rightarrow E + 3 \rightarrow E + E + 3 \rightarrow E + 2 + 3 \rightarrow 1 + 2 + 3$

⇒ Dérivations différentes ...même résultat ?

⇒ Pas toujours (par ex. associativité, priorité des opérateurs)

Langage généré

- ▶ Soit G une grammaire, alors le langage **généré** par G est

$$L(G) = \{m \in T^* \mid S \xrightarrow{*} m\}$$

⇒ Sous-ensemble de T^*

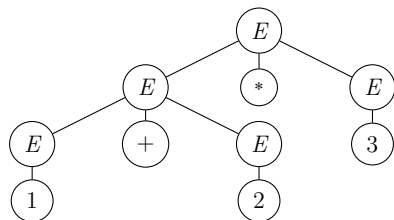
⇒ Pas nécessairement fini

Arbre de dérivation

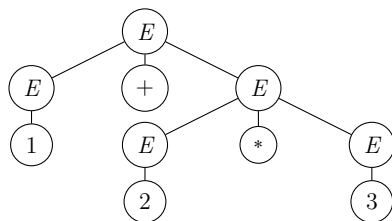
- ▶ Représentation graphique de la dérivation
 - **Racine** : symbole initial = S
 - **Nœud** : symbole non-terminal $\in N$
 - **Feuille** : symbole terminal $\in T$
 - Relation **parent-enfants** : dérivation (règle)

⇒ Structure de l'analyse

- ▶ Dérivations (et analyses) de $1 + 2 * 3$



dérivation gauche



dérivation droite

Plan

1. Origines
2. Définitions
3. Dérivations
4. Simplification

Simplifier une grammaire

- ⇒ Supprimer les éléments **inutiles** de la grammaire
- Symboles **improductifs**
 - A est improductif s'il n'y a pas de $m \in T^*$ tel que $A \xrightarrow{*} m$
 - Symboles **inaccessibles**
 - A est inaccessible s'il n'y a pas de α et β tels que $S \xrightarrow{*} \alpha A \beta$
 - ϵ -productions
 - Une ϵ -production est une dérivation telle que $A \xrightarrow{*} \epsilon$
 - Production simple
 - $A \rightarrow B$ est une production simple si $A \in N$ et $B \in N$
- ⇒ Pour toute grammaire, il existe une grammaire équivalente sans symboles improductifs ni inaccessibles, sans ϵ -productions ni productions simples

Élimination des symboles improductifs

► Calcul des symboles productifs

- Soit $P_0 = \emptyset$ et $i = 1$
- Soit $P_1 = \{A \in N, \exists \alpha \in T^*, A \rightarrow \alpha \in R\}$
- Tant que $P_i \neq P_{i-1}$
 - $P_{i+1} = P_i \cup \{A \in N, \exists \alpha \in (T \cup P_i)^*, A \rightarrow \alpha \in R\}$
 - $i \leftarrow i + 1$

⇒ Les symboles de $N \setminus P$ sont improductifs

⇒ Enlever ces symboles et les règles dans lesquels ils figurent

Élimination des symboles inaccessibles

► Calcul des symboles accessibles

- Soit $C_0 = \emptyset$, $C_1 = \{S\}$ et $i = 1$
- Tant que $C_i \neq C_{i-1}$
 - $C_{i+1} = C_i \cup \{A \in N, \exists \alpha, \beta \in (N \cup T)^*, X \in C_i, X \rightarrow \alpha A \beta \in R\}$

⇒ Les symboles de $N \setminus C$ sont inaccessibles

⇒ Enlever ces symboles et les règles dans lesquels ils figurent

Élimination des ϵ -productions

► Calcul des symboles annulables

- Soit $U_0 = \emptyset$ et $i = 1$
- Soit $U_1 = \{A \in N, A \rightarrow \epsilon \in R\}$
- Tant que $P_i \neq P_{i-1}$
 - $U_{i+1} = U_i \cup \{A \in N, \exists \alpha \in (U_i)^*, A \rightarrow \alpha \in R\}$
 - $i \leftarrow i + 1$

⇒ Les symboles de U sont annulables

► Modification de la grammaire

- Remplacer les règles $A \rightarrow \alpha X \beta$ où $X \in U$ par $A \rightarrow \alpha X \beta | \alpha \beta$
(avec combinaisons possibles de X dans les règles)
- Supprimer toutes les règles $A \rightarrow \epsilon$ (sauf pour S)
- Supprimer toutes les règles $A \rightarrow A$

⇒ Grammaire équivalente à ϵ près

Équivalences et productions simples

- ▶ Productions simples, dérivations et classes d'équivalences
 - Production simple : toute règle $A \rightarrow B$ avec $B \in N$
 - Soit la relation \geq telle que $A \geq B$ si $A \xrightarrow{*} B$
 - Soit la relation \approx telle que $A \approx B$ si $A \geq B$ et $B \geq A$
 - **Classes d'équivalences**
 - Si $A \approx B$, tout ce qui est dérivé de A peut l'être de B
 - Relation réflexive, symétrique et transitive
 - L'ensemble des classes est une **partition** de N
- ▶ Modification de la grammaire
 - On conserve les productions non-simples
 - Pour chaque classe d'équivalence
 - \Rightarrow Choisir un symbole qui remplace tous les autres
 - \Rightarrow Pour chaque dérivation $A \xrightarrow{*} B$
 - Pour chaque $B \rightarrow \beta$, ajouter $A \rightarrow \beta$

Exemple : simplification de grammaire

► Grammaire

1. $S \rightarrow T|U$
2. $U \rightarrow aYb|V$
3. $V \rightarrow W$
4. $X \rightarrow W|a$
5. $Y \rightarrow Z$
6. $Z \rightarrow c|\epsilon$

► Étapes

- Symboles productifs :
- Symboles accessibles :
- ϵ -productions :
- Productions simples :

Exemple : simplification de grammaire

► Grammaire

1. $S \rightarrow T|U$
2. $U \rightarrow aYb|V$
3. $V \rightarrow W$
4. $X \rightarrow W|a$
5. $Y \rightarrow Z$
6. $Z \rightarrow c|\epsilon$

► Étapes

- Symboles productifs : $\{X, Z, Y, U, S\} \Rightarrow$ retirer T, V et W
- Symboles accessibles :
- ϵ -productions :
- Productions simples :

Exemple : simplification de grammaire

► Grammaire

1. $S \rightarrow U$
2. $U \rightarrow aYb$
- 3.
4. $X \rightarrow a$
5. $Y \rightarrow Z$
6. $Z \rightarrow c|\epsilon$

► Étapes

- Symboles productifs : $\{X, Z, Y, U, S\} \Rightarrow$ retirer T, V et W
- Symboles accessibles :
- ϵ -productions :
- Productions simples :

Exemple : simplification de grammaire

► Grammaire

1. $S \rightarrow U$
2. $U \rightarrow aYb$
- 3.
4. $X \rightarrow a$
5. $Y \rightarrow Z$
6. $Z \rightarrow c|\epsilon$

► Étapes

- Symboles productifs : $\{X, Z, Y, U, S\} \Rightarrow$ retirer T, V et W
- Symboles accessibles : $\{S, U, Y, Z\} \Rightarrow$ retirer X
- ϵ -productions :
- Productions simples :

Exemple : simplification de grammaire

► Grammaire

1. $S \rightarrow U$
2. $U \rightarrow aYb$
- 3.
- 4.
5. $Y \rightarrow Z$
6. $Z \rightarrow c|\epsilon$

► Étapes

- Symboles productifs : $\{X, Z, Y, U, S\} \Rightarrow$ retirer T, V et W
- Symboles accessibles : $\{S, U, Y, Z\} \Rightarrow$ retirer X
- ϵ -productions :
- Productions simples :

Exemple : simplification de grammaire

► Grammaire

1. $S \rightarrow U$
2. $U \rightarrow aYb$
- 3.
- 4.
5. $Y \rightarrow Z$
6. $Z \rightarrow c|\epsilon$

► Étapes

- Symboles productifs : $\{X, Z, Y, U, S\} \Rightarrow$ retirer T, V et W
- Symboles accessibles : $\{S, U, Y, Z\} \Rightarrow$ retirer X
- ϵ -productions : $\{Z, Y\} \Rightarrow$ modifier 6, 2
- Productions simples :

Exemple : simplification de grammaire

► Grammaire

1. $S \rightarrow U$
2. $U \rightarrow aYb|ab$
- 3.
- 4.
5. $Y \rightarrow Z$
6. $Z \rightarrow c$

► Étapes

- Symboles productifs : $\{X, Z, Y, U, S\} \Rightarrow$ retirer T, V et W
- Symboles accessibles : $\{S, U, Y, Z\} \Rightarrow$ retirer X
- ϵ -productions : $\{Z, Y\} \Rightarrow$ modifier 6, 2
- Productions simples :

Exemple : simplification de grammaire

► Grammaire

1. $S \rightarrow U$
2. $U \rightarrow aYb|ab$
- 3.
- 4.
5. $Y \rightarrow Z$
6. $Z \rightarrow c$

► Étapes

- Symboles productifs : $\{X, Z, Y, U, S\} \Rightarrow$ retirer T, V et W
- Symboles accessibles : $\{S, U, Y, Z\} \Rightarrow$ retirer X
- ϵ -productions : $\{Z, Y\} \Rightarrow$ modifier 6, 2
- Productions simples : $S \rightarrow U$ et $Y \rightarrow Z \Rightarrow$ modifier 1, 2, 5, 6

Exemple : simplification de grammaire

► Grammaire

1. $S \rightarrow aYb|ab$
- 2.
- 3.
- 4.
5. $Y \rightarrow c$
- 6.

► Étapes

- Symboles productifs : $\{X, Z, Y, U, S\} \Rightarrow$ retirer T, V et W
- Symboles accessibles : $\{S, U, Y, Z\} \Rightarrow$ retirer X
- ϵ -productions : $\{Z, Y\} \Rightarrow$ modifier 6, 2
- Productions simples : $S \rightarrow U$ et $Y \rightarrow Z \Rightarrow$ modifier 1, 2, 5, 6

Exercice : simplification de grammaire

► Réduire les grammaires suivantes

• G_1

- $S \rightarrow bSc|bTc|a|\epsilon$
- $T \rightarrow U$
- $U \rightarrow bUc|T$
- $V \rightarrow U|bc$

• G_2

- $S \rightarrow UXT$
- $T \rightarrow b$
- $U \rightarrow aV|aXTXb$
- $V \rightarrow cV|aWT$
- $W \rightarrow V$
- $X \rightarrow ab|\epsilon$
- $Y \rightarrow cZ$
- $Z \rightarrow aa$

Formes normales

- ▶ Forme normale de **Chomsky** : toutes règles de la forme
 - $A \rightarrow BC$ avec $A, B, C \in N$
 - $A \rightarrow a$ avec $a \in T$
 - ▶ Forme normale de **Greibach** : toutes règles de la forme
 - $A \rightarrow a\alpha$ avec $a \in T, \alpha \in N^*$
- ⇒ Pour tout langage hors-contexte il existe une grammaire en forme normale de Chomsky et une grammaire en forme normale de Greibach qui le génèrent

Mise sous forme normale de Chomsky

▶ Étapes

- Pour chaque terminal a , créer
 - Un symbole Z_a
 - Une règle $Z_a \rightarrow a$
- Pour chaque règle $A \rightarrow \alpha$ où $|\alpha| > 1$
 - Tout terminal a de α est remplacé par Z_a
- Pour chaque règle $A \rightarrow \alpha$ où $|\alpha| > 2$
 - On décompose : $\alpha = A_1, A_2 \dots A_n$
 - On crée les non-terminaux $Y_1, Y_2 \dots Y_{n-2}$
 - On remplace $A \rightarrow \alpha$ par

$$A \rightarrow A_1 Y_1$$

$$Y_1 \rightarrow A_2 Y_2$$

$$\dots$$

$$Y_{n-2} \rightarrow A_{n-1} A_n$$
- Suppression des ϵ -productions et productions simples

⇒ Forme normale de Chomsky

Récurtivité

- ▶ Symbole **récurtif** : $A \xrightarrow{*} \alpha A \beta$
 - Si $\alpha = \epsilon$, A est récurtif à **gauche**
 - Si $\beta = \epsilon$, A est récurtif à **droite**
 - Si $\xrightarrow{*}$ ne comporte qu'une dérivation : récurtivité **directe**
 - Si $\xrightarrow{*}$ comporte plusieurs dérivation : récurtivité **indirecte**

⇒ Une grammaire récurtive comporte un symbole récurtif

- ▶ Exemple : grammaire indirectement récurtive à gauche
 - $A \rightarrow B$
 - $B \rightarrow CD$
 - $C \rightarrow AE$
- ▶ Suppression de la récurtivité directe à gauche
 - Remplacer toute règle $A \rightarrow Aa|b$
 - $A \rightarrow bA'$
 - $A' \rightarrow aA'|\epsilon$

Ambiguïté et équivalence

► Ambiguïté de grammaires

- Grammaire : un mot est généré par deux arbres de dérivation

⇒ Exemple : rattachement prépositionnel

- $S \rightarrow GN V GN PRP GN$

- $S \rightarrow GN V GN$

- $GN \rightarrow GN PRP GN$

⇒ *Il voit le chat de sa voisine* (le chat de qui?)

⇒ *Il voit le chat de sa fenêtre* (d'où voit-il?)

- Ambiguïté sémantique (plutôt que morpho-syntaxique)

► **Équivalence** de grammaires

- Génèrent le même langage
- Donnent les mêmes arbres d'analyse (équivalence forte)

Autre formalismes de grammaires

- ▶ Insuffisances des grammaires hors-contexte
 - Accords (flexions / morphologie)
 - Anaphores
 - Portée de la coordination
 - Méthodes statistiques

⇒ Non traitées par les grammaires hors-contexte

- ▶ Autres formalismes
 - PCFG (Probabilistic context-free grammar)
 - LFG (Lexical Functional Grammar)
 - HPSG (Head-driven Phrase Structure Grammar)
 - TAG (Tree Adjoining Grammar)
 - CCG (Combinatory Categorical Grammar)
 - Transition-based Parsing
 - ...

Exercices

- ▶ Mettre sous forme normale de Chomsky
 - $S \rightarrow AbA$
 - $A \rightarrow AaA|ca$
- ▶ Soit l'ensemble de symboles non-terminaux :
 $N = \{GN, GV, DET, PREP, NOM, ADJ, ADV\}$
 - Définissez les règles d'une grammaire qui génère des phrases
 - Ajoutez des éléments terminaux et leurs règles
 - Donnez les arbres de dérivation pour les phrases suivantes
 - *le chat mange*
 - *le chat mange la souris*
 - *le chat regarde le bout de fromage*
 - Donnez quelques phrases générées par la grammaire
 - Quel problème rencontre-t-on pour les genres (m/f) ?
 - Modifiez la grammaire pour les phrases interrogatives

TP : SWI-Prolog

```

:- use_rendering(svgtree, [list(false)]).

% Règles non terminales
s(s(X,Y)) --> gn(X), gv(Y).
gn(X) --> np(X).
gn(gn(X,Y)) --> det(X), nc(Y).
gv(gv(X,Y)) --> v(X), gn(Y).

% Règles terminales
np(np(jean)) --> [jean].
det(det(de)) --> [de].
nc(nc(philosophie)) --> [philosophie].
nc(nc(politique)) --> [politique].
v(v(discute)) --> [discute].

% Requête
phrase(s(Tree), [jean, discute, de, politique]).

```