



ISEN CIR ³

Théorie des langages

3. Grammaires formelles

- Les langages de programmation sont conçu sur la base de la théorie des langages formels.
- La **lexique** est l'ensemble des éléments (mots, lettres, symboles) de ces langages
- La **syntaxe** (ou la grammaire) est l'ensemble des règles qui régissent leur ordonnancement.
- La **sémantique** est le sens donné aux mots ou aux phrases du langage.

- Une grammaire est l'ensemble des règles de bases pour générer les mots (ou les phrases) d'un langage.
- Elle prend en général la forme de règles de réécriture. On remplace un élément par un ensemble d'autres éléments.
- Toute phrase générée est obtenue à partir d'un symbole racine (ex. un mot, une phrase, un programme, ...)

- Considérons la phrase :
« Il est sympa »

Peut on construire une grammaire permettant de produire cette phrase ainsi que d'autres phrases similaires ??

Un ensemble de mots

{il, elle, est, devient, reste, sympa, rapide }

Des constituants ou des éléments grammaticaux

PHRASE, PRONOM, VERBE, COMPLEMENT}

Un point de départ

Nous souhaitons former une PHRASE

Et un ensemble de règles de remplacement

PHRASE → PRONOM VERBE COMPLEMENT

PRONOM → il ou elle

VERBE → est ou devient ou reste

COMPLEMENT → sympa ou rapide

- Une grammaire **G** est défini par 4 éléments (V_T, V_N, P, S):
 - l'ensemble des symboles terminaux V_T
 - l'ensemble des non-terminaux V_N
 - l'ensemble des règles de productions **P** (règles grammaticales)
 - le symbole de départ **S** (non-terminal), nous parlons aussi d'**axiome**
- Diverses notations:
 - Terminaux: les minuscules, chiffres, symboles en italiques
 - Non-terminaux: majuscules: A, B, C, S ; entre <crochets>
 - Chaque chaîne reconnue par la grammaire est une **phrase**
 - L'ensemble des phrases est le **langage** de **G** noté **L(G)**

- $V_T = \{ \text{il, elle, est, devient, reste, sympa, rapide} \}$
- $V_N = \{ \text{PHRASE, PRONOM, VERBE, COMPLEMENT} \}$
- $S = \text{PHRASE}$
- $P = \{$
 - PHRASE \rightarrow PRONOM VERBE COMPLEMENT
 - PRONOM \rightarrow il | elle
 - VERBE \rightarrow est | devient | reste
 - COMPLEMENT \rightarrow sympa | rapide $\}$

Une grammaire des expressions arithmétiques $\mathbf{G} = (V_T, V_N, P, S)$:

$V_T = \{ +, -, *, /, (,), \text{nombre} \}$, avec $\text{nombre} = \{0, 1, 2, 3, 4, \dots\}$ (nombres naturels)

$V_N = \{ \text{EXPRESSION} \}$

$S = \text{EXPRESSION}$

$P = \{$
 EXPRESSION : EXPRESSION + EXPRESSION
 | EXPRESSION - EXPRESSION
 | EXPRESSION * EXPRESSION
 | EXPRESSION / EXPRESSION
 | (EXPRESSION)
 | nombre
}

- Une **dérivation** est un mot composé uniquement de symboles terminaux (lettres de l'alphabet) obtenu à partir de **S** par une série de productions.

$$S \Rightarrow \psi_1 \Rightarrow \psi_2 \Rightarrow \psi_3 \Rightarrow m$$

- La dérivation est notée : $S \Rightarrow^* w$
- Définition:
Etant donnée une grammaire **G**, on note **L(G)** le langage généré par **G** et défini par $\{m \in V_T^* / S \Rightarrow^* m\}$

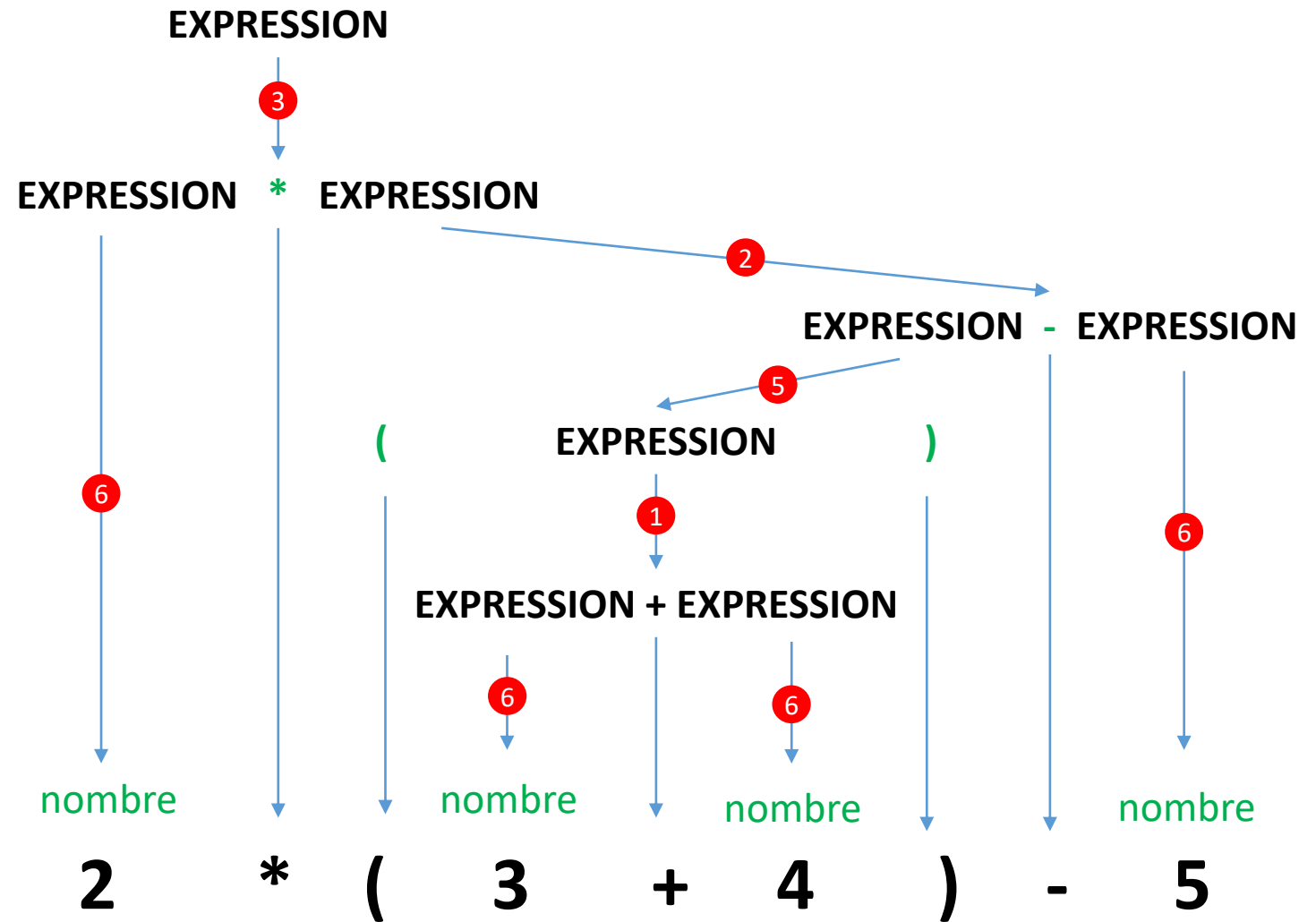
- Un **arbre de dérivation** (ou arbre syntaxique) pour un mot **m** est un arbre :
 - Dont les nœuds sont étiquetés par des **non-terminaux**
 - Dont les feuilles sont étiquetées par des **terminaux** (ou ϵ)
 - Dont la racine est étiquetée par l'**axiome**
 - Chaque nœud correspond l'application d'une règle de production
 - La concaténation des feuilles de gauche à droite forme le mot **m**

Arbre de dérivation : Exemple


EXPRESSION : EXPRESSION + EXPRESSION
 | EXPRESSION - EXPRESSION
 | EXPRESSION * EXPRESSION
 | EXPRESSION / EXPRESSION
 | (EXPRESSION) 5
 | nombre 6

Avec $\text{nombre} \in \{0, 1, 2, 3, 4, \dots\}$

$$2 * (3 + 4) - 5$$



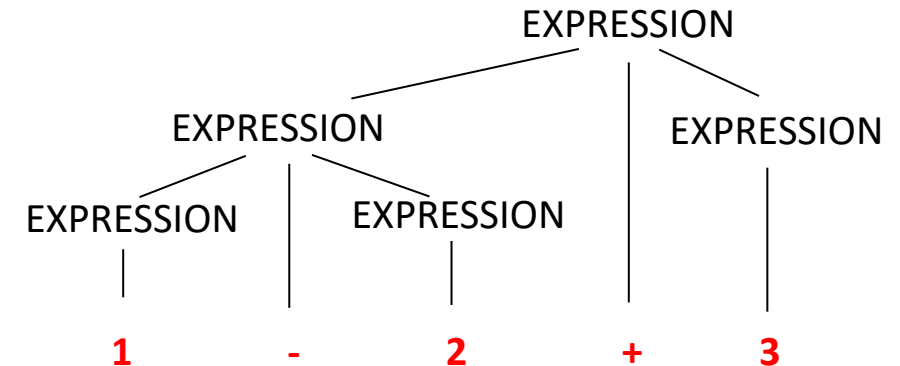
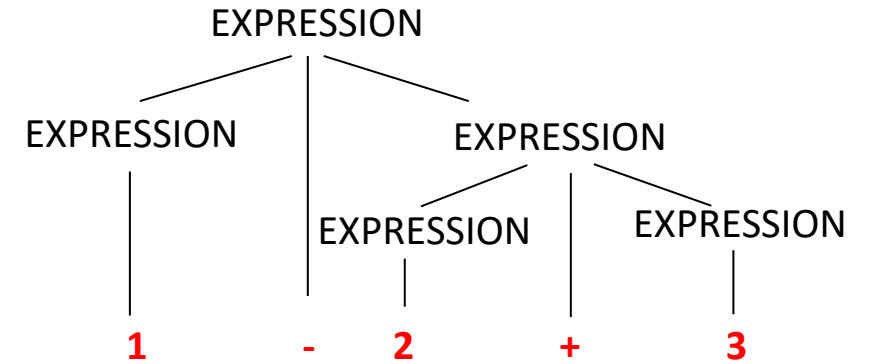
- Définition : Une grammaire est dite **ambiguë** s'il existe une phrase (un mot) du langage $L(G)$ qui a plusieurs arbres de dérivation (arbre syntaxique).
- **A éviter** quand on veut générer un analyseur.
- Exemple d'ambiguïté : les « else pendants » du Pascal

Soit $G = \{$
Instruction : **if** '(' Expr ')' Instruction **else** Instruction
 | **if** '(' Expr ')' Instruction
Instruction: ...
Expr:

if (x>10) **then** **if** (y<10) a = 1 **else** a = 0

EXPRESSION : EXPRESSION + EXPRESSION
 EXPRESSION - EXPRESSION
 EXPRESSION * EXPRESSION
 EXPRESSION / EXPRESSION
 (EXPRESSION)
 nombre

Avec $\text{nombre} \in \{0, 1, 2, 3, 4, \dots\}$

Exemple d'ambiguïté



La grammaire est elle ambiguë ?

$1 - 2 + 3$ (nombre - nombre + nombre)

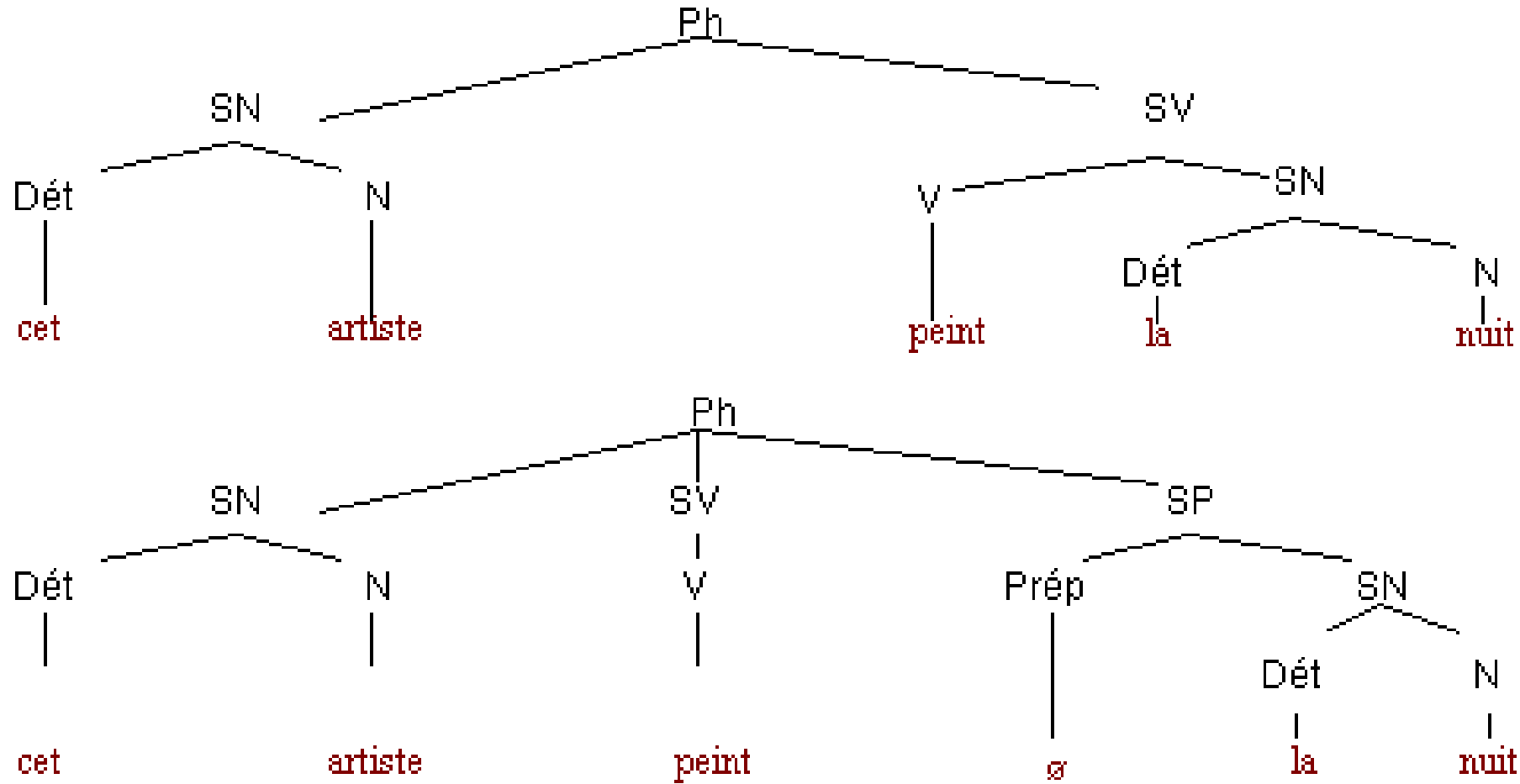
a deux arbres d'analyse.

(Ambiguïté d'associativité)

Nous reviendrons plus tard sur

les méthodes pour lever les ambiguïtés

Autre exemple d'ambiguïté



- Générer des mots de la forme $a^i b^i$ l'alphabet $\{a, b, c\}$
- Exemple de mots du langage : $ab, aabb, aaabbb, aaaabbbb, \dots$

Réponse :

$$S \rightarrow aSb \mid \varepsilon$$

Exemple : pour générer $aabb$, voici l'ensemble des dérivations :

$$S \rightarrow aSb \rightarrow aaSbb \rightarrow aabb$$

- Générer des mots palindromes sur l'alphabet $\{a, b, c\}$
- Exemple de mots du langage : $a, aba, abba, abcba, aaccbccaa, \dots$

Réponse :

$$S \rightarrow aSa \mid bSb \mid cSc \mid a \mid b \mid c \mid \varepsilon$$

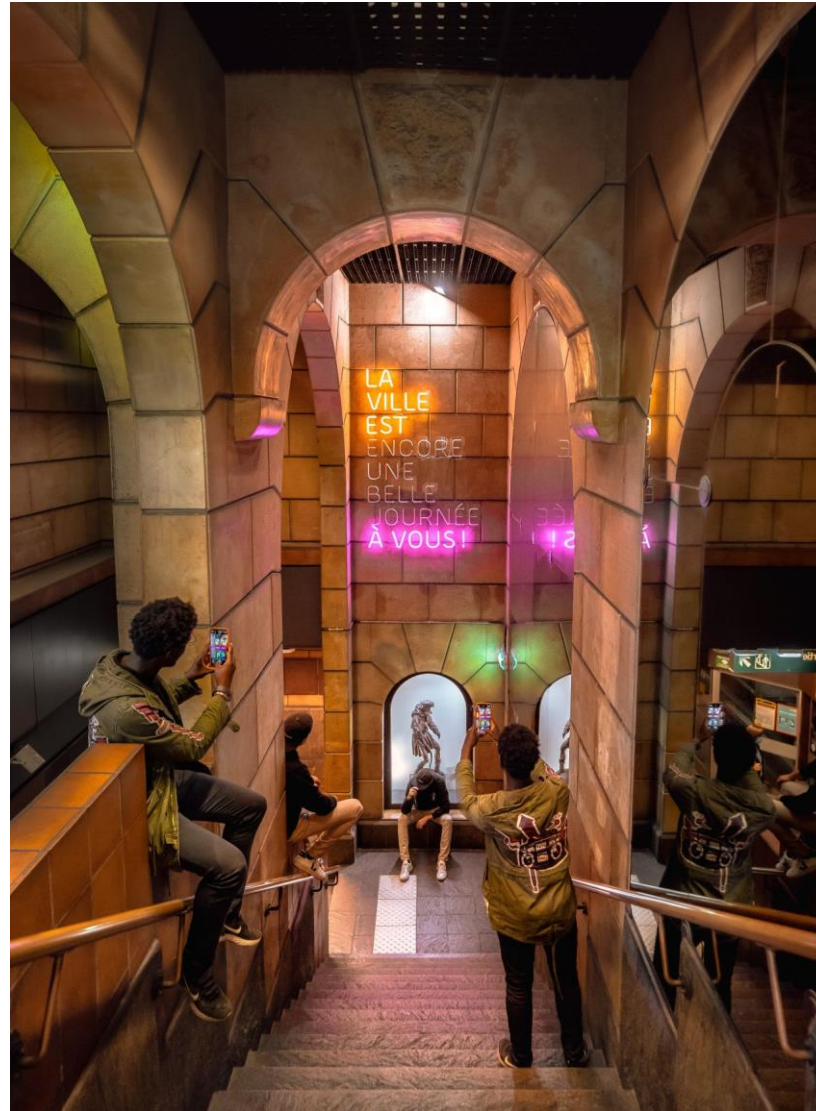
Exemple : pour générer $abcba$, voici l'ensemble des dérivations :

$$S \rightarrow aSa \rightarrow abSba \rightarrow abcba$$

- Grammaire permettant de générer des nombres binaires ayant un nombre pair de **1** et un nombre pair de **0**
- Exemple de mots : le mot vide , 00, 1010, 0011, 0110, 11100001, ...

Réponse :

$$S \rightarrow 0A \mid 1B \mid \varepsilon$$
$$A \rightarrow 0S \mid 1C$$
$$B \rightarrow 0C \mid 1S$$
$$C \rightarrow 0B \mid 1A$$



Installation artistique à
métro République à
base de tubes au néon

Combien de phrases
possible ?

Ecrire une grammaire
permettant de les
générer.

- **Type 0** : Aucune restriction dans les règles de production.
- **Type 1** : Dans toutes les productions, la longueur de la partie gauche doit être inférieure ou égale à la partie droite.
- **Type 2** : Dans toutes les productions, la partie gauche est composée d'un seul non terminal.
- **Type 3** : Dans toutes les productions, la partie gauche est composée d'un seul non-terminal. La partie droite est soit un terminal, soit une paire terminal/non-terminal, en respectant la même position du non-terminal pour toutes les règles (à gauche ou à droite).

Exemples pour chaque type

Type 3

Grammaire régulière

$$S \rightarrow 0A \mid 1B \mid \varepsilon$$
$$A \rightarrow 0S \mid 1C$$
$$B \rightarrow 0C \mid 1S$$
$$C \rightarrow 0B \mid 1A$$

Type 2

Grammaire algébrique
(ou à contexte libre)

$$\begin{aligned} S &\rightarrow aSa \\ &\mid bSb \\ &\mid cSc \\ &\mid a \mid b \mid c \mid \varepsilon \end{aligned}$$

Type 1

Grammaire contextuelle
(à contexte lié)

$$S \rightarrow aA \mid bSb$$
$$aA \rightarrow aA \mid bSb$$
$$cSc \rightarrow aAa \mid bSb$$
$$aA \rightarrow aA \mid bSb$$
$$\mid a \mid b \mid c \mid \varepsilon$$

Ne seront pas étudiées
dans ce cours

Type 0

Grammaire générale
(sans restrictions)

$$S \rightarrow aA \mid bSb$$
$$aA \rightarrow aA \mid bbA$$
$$baAb \rightarrow \varepsilon$$

Ne seront pas étudiées
dans ce cours

