



ALF

Grammaires

## **Keith Cooper, Linda Torczon, *Engineering a Compiler***

- Chapitre 3
  - 3.1
  - 3.2

## **Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, *Compilers: Principles, Techniques, and Tools (2nd Edition)***

- Chapitre 4
  - 4.2
  - 4.3

- Grammaires indépendantes du contexte
- L'arbre du parse
- ANTLR





- Américain
- FORTRAN
- Forme Backus-Naur
- Turing Prix

- Une manière formelle de décrire une langue
- Une langue peut avoir plusieurs grammaires
- Hiérarchie de Chomsky

# Hiérarchie de Chomsky

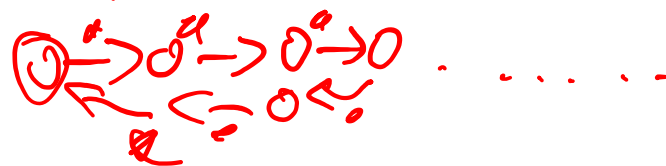
Grammaire	Description
Type 0	Récursivement énumérable <i>ROCKET SCIENCE</i>
Type 1	Dépendante du contexte <i>ALEXA, SIRI</i>
Type 2	<b>Indépendant du contexte</b> <i>! ALF</i>
Type 3	Grammaire régulière <i>Regex</i>

$aaa \dots lll \dots$   
 $n = n$

$a + l +$   
 $\frac{\quad}{n = m?}$

$a \{ \_ \} e \{ \_ \}$   
 $n$

$aaalll$



- 1 Terminaux

- jetons (tokens)
- a, b, c, d, e, f

- 2 Non-terminaux

- S, V, N, E ..

- 3 Symbole de début

- S

- 4 Productions

- $N \rightarrow \dots$

GIP

– REGLES

(REGLE)

# Exemple

---

**S**     $\rightarrow$     **A**

**S**     $\rightarrow$     **B**

**A**     $\rightarrow$     **a**

**A**     $\rightarrow$     **aA**

**B**     $\rightarrow$     **b**

**B**     $\rightarrow$     **bB**



# Exemple

---

## Token

numero:  $[0-9]^+$

sign:  $[\+ \- \* \/]$

## Production

$E \rightarrow E \text{ sign } E$

$E \rightarrow -E$

$E \rightarrow \text{numero}$

# Exemple

---

## Token

numero:  $[0-9]^+$

sign:  $[\+ \- \* \/]$

## Production

$E \rightarrow E \text{ sign } E \mid -E \mid \text{numero}$

# Exemple

---

## Token

numero:  $[0-9]^+$

sign:  $[\+ \- \* \ /]$

## Production

E	->	E sign E
		-E
		numero

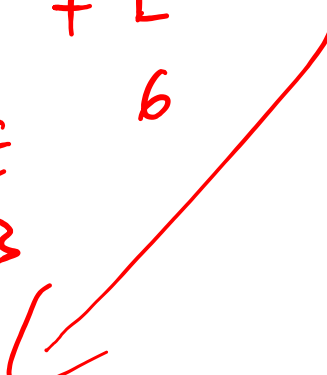
- Remplacement d'un non terminal par sa production
  - $1+2-3+6$
- $E \Rightarrow 1 + E$ 
  - $\Rightarrow 1 + 2 - E$
  - $\Rightarrow 1 + 2 - 3 + E$
  - $\Rightarrow 1 + 2 - 3 + 6$

# Dérivation de gauche

- 1+2-3+6
- $E \Rightarrow 1 + E$ 
  - $\Rightarrow 1 + 2 - E$
  - $\Rightarrow 1 + 2 - 3 + E$
  - $\Rightarrow 1 + 2 - 3 + 6$

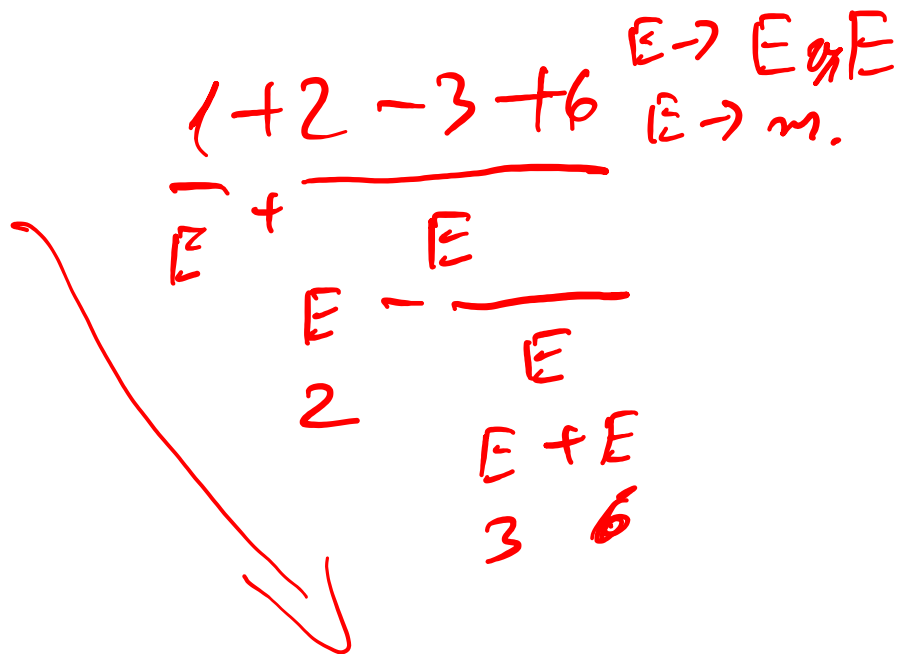
$$\begin{array}{r} 1+2-3+6 \\ \hline E \quad \quad \quad + E \\ \hline E \quad - E \quad \quad 6 \\ E+E \quad - \quad 3 \\ 1+2 \end{array}$$

$E \rightarrow E \text{ on } E$   
 $E \rightarrow m.$



# Dérivation la plus à droite

- $1+2-3+6$
- $E \Rightarrow E + 6$ 
  - $\Rightarrow E - 3 + 6$
  - $\Rightarrow E + 2 - 3 + 6$
  - $\Rightarrow 1 + 2 - 3 + 6$



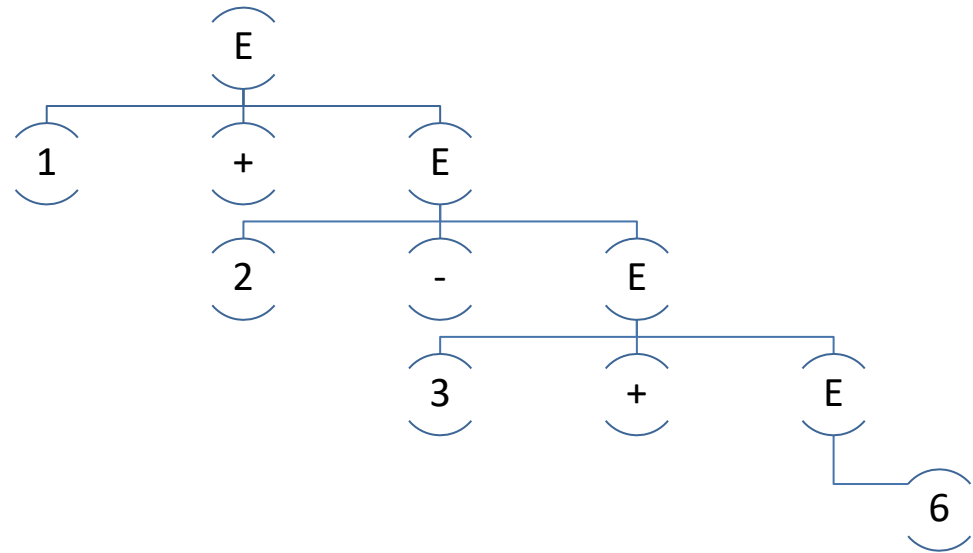
# L'arbre du parse

- $1+2-3+6$
- $E \Rightarrow 1 + E$

–  $\Rightarrow 1 + 2 - E$

–  $\Rightarrow 1 + 2 - 3 + E$

–  $\Rightarrow 1 + 2 - 3 + 6$



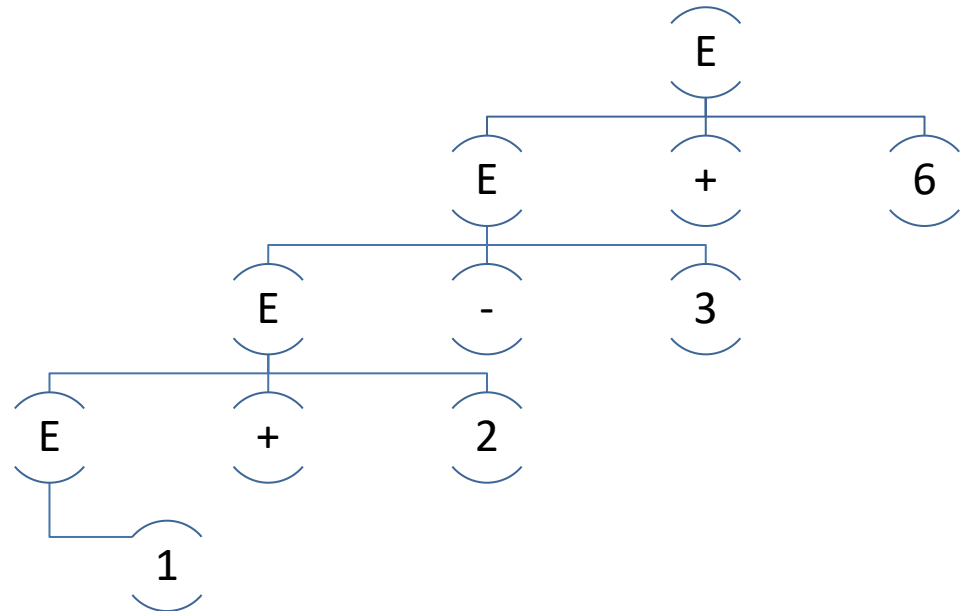
# L'arbre du parse

- $1+2-3+6$
- $E \Rightarrow E + 6$

~~-~~  $\Rightarrow E - 3 + 6$

~~-~~  $\Rightarrow E + 2 - 3 + 6$

~~-~~  $\Rightarrow 1 + 2 - 3 + 6$





# L'arbre du parse

- $1+2-3+6$

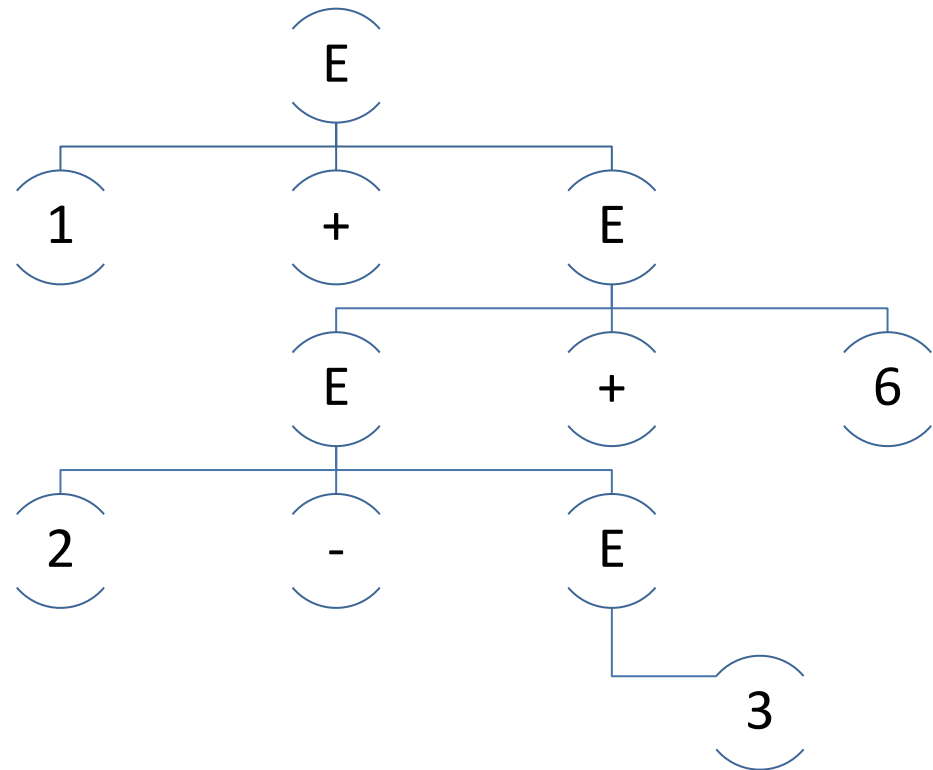
- $E \Rightarrow 1 + E$

- $\Rightarrow 1 + E + 6$

- $\Rightarrow 1 + 2 - E + 6$

- $\Rightarrow 1 + 2 - 3 + 6$

$1+2-3+6$   
      -        
E      E      E  
E+E      E+E  
1  2      3  6



- Grammaires qui produisent plus d'un arbre du parse

E	->	E sign E
		-E
		numero

# Exemple de Ambiguïté

- Grammaires qui produisent plus d'un arbre du parse

TOKEN

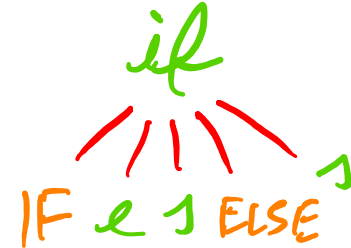
RULE

if -> IF expression statements

| IF expression statements ELSE statements

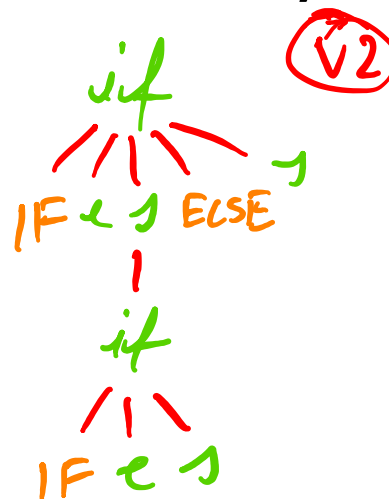
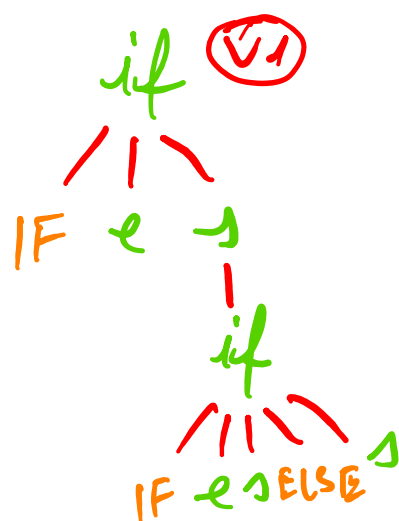
# Exemple de Ambiguïté

```
if (a == 0)
    console.log ('zero');
else
    console.log ('not zero');
```



# Exemple de Ambiguïté

```
if (a == 0)
  if (e == 0)
    console.log ('zero');
  else
    ? console.log ('not zero');
```



# Exemple de Ambiguïté

---

```
if (a == 0)
    if (e == 0)
        console.log ('zero');
else
    console.log ('not zero');
```

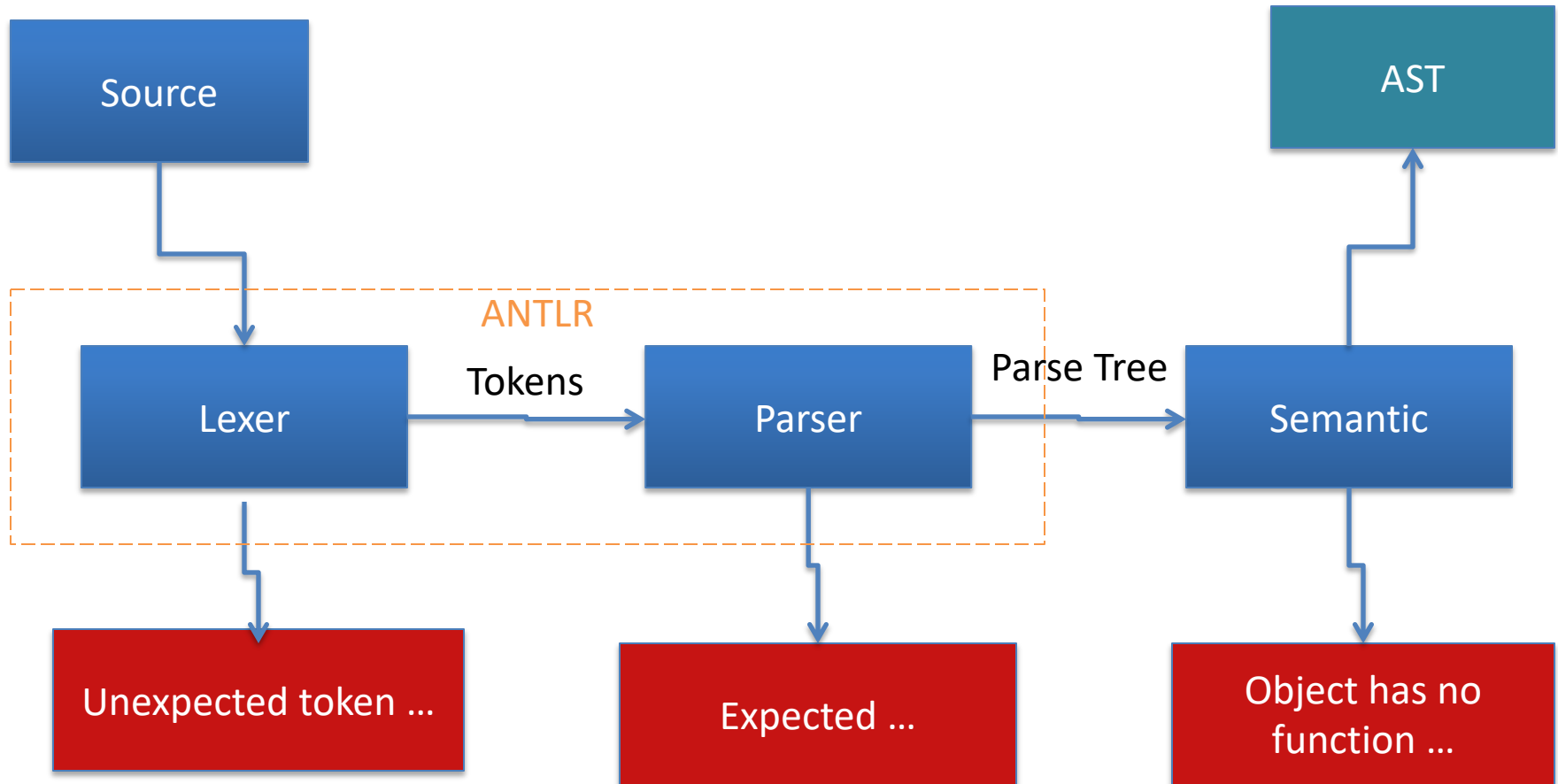
- Grammaire de parenthèses
- Expression avec multiplication et division
- Expression avec parenthèses
- Vecteur
- Déclaration de variable
- Déclaration de fonction



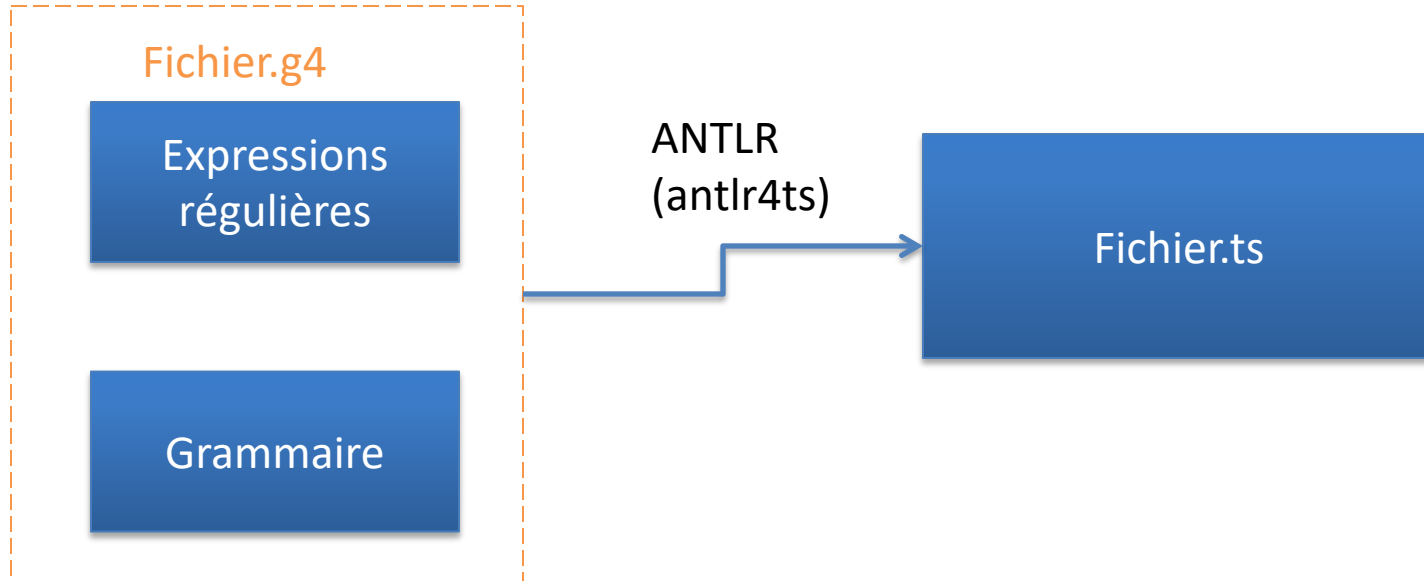
**ANTLR**



# Frontent



# Générateur de lexer/parseur



- Lexer
  - Expressions régulières
  
- Parser
  - Grammaire

# Fichier.l (lex ou flex)

```
grammar Expr;
```

```
NEWLINE : [\r\n]+ ;
```

```
INT : [0-9]+ ;
```

```
prog: (expr NEWLINE)* ;
```

```
expr: expr ('*' | '/') expr
```

```
    | expr ('+' | '-') expr
```

```
    | INT
```

```
    | '(' expr ')'
```

```
;
```

- Grammaires indépendantes du contexte
- L'arbre du parse
- Ambiguïté
- Jison

# Fichier.g4

```
grammar Expr;
```

Nom

```
NEWLINE: [\r\n]+;
```

```
INT: [0-9]+;
```

Expressions Régulières

```
prog: (expr NEWLINE)*;
```

Règles de Grammaire

```
expr:
```

```
1 | expr ( '*' | '/' ) expr # exprMultiplyDivision  
2 | | expr ( '+' | '-' ) expr # exprAddSubtract  
3 | | INT # exprInt  
4 | | '(' expr ')' # exprParantheses;
```

# Nom

grammar Expr;

Nom

Expression Régulières

Règles de Grammaire

# Expressions Régulières

Nom

NEWLINE: `[\r\n]+;`

INT: `[0-9]+;`

Expressions Régulières

Règles de Grammaire



# Règles de Grammaire

Nom

Expressions Régulières

```
prog: (expr NEWLINE)*;                                Règles de Grammaire
expr:
expr ('*' | '/') expr # exprMultiplyDivision
| expr ('+' | '-') expr # exprAddSubtract
| INT # exprInt
| '(' expr ')' # exprParantheses;
```

# Questions

---

