

# Théorie des langages : THL

## CM 6

Uli Fahrenberg

EPITA Rennes

S5 2024

# Aperçu

# Programme du cours

- ① Langages rationnels, automates finis
- ③ Langages algébriques, grammaires hors-contexte, automates à pile
  - TP 1 : flex
  - QCM 1 : langages rationnels
- ⑤ Parsage LL
  - TP 2 : parsage LL
- ⑦ **Parsage LR, partie 1**
  - QCM 2 : parsage LL
- ⑧ Parsage LR, partie 2
- ⑨ Introduction flex & bison
- ⑩ TP 3, 4 : flex & bison

# Parsage

# Re : parsage LL(1)

- ① entrée : une grammaire hors contexte  $G = (N, \Sigma, P, S)$ 
  - si-dessous,  $V = N \cup \Sigma$
  - éliminer récursion à gauche dans  $G$  ; factoriser  $G$  à gauche
- ② calculer NULL
  - $\text{NULL} = \{A \in N \mid A \Rightarrow^* \varepsilon\}$
- ③ construire la table FIRST
  - $\text{FIRST}(A) = \{a \in \Sigma \mid \exists w \in V^* : A \Rightarrow^* aw\}$
- ④ construire la table FOLLOW
  - $\text{FOLLOW}(A) = \{a \in \Sigma \mid \exists B \in N, \alpha, \beta \in V^* : B \Rightarrow^* \alpha A a \beta\}$
- ⑤ construire la TABLE de parsage :
  - ① pour chaque production  $X \rightarrow w$  ( $n$ ) :
    - ① pour chaque  $a \in \text{FIRST}(w)$  :  $\text{TABLE}(X, a) += \{n\}$
    - ② si  $w \in \text{NULL}$  ou  $w = \varepsilon$  :
      - pour chaque  $a \in \text{FOLLOW}(X)$  :  $\text{TABLE}(X, a) += \{n\}$

## Re : exemple

$$Z \rightarrow XYZ \quad (1)$$

$$\quad | c \quad (2)$$

$$X \rightarrow a \quad (3)$$

$$\quad | Y \quad (4)$$

$$Y \rightarrow b \quad (5)$$

$$\quad | \varepsilon \quad (6)$$

NULL =

## Re : exemple

$$Z \rightarrow XYZ \quad (1)$$

$$| c \quad (2)$$

$$X \rightarrow a \quad (3)$$

$$| Y \quad (4)$$

$$Y \rightarrow b \quad (5)$$

$$| \varepsilon \quad (6)$$

$$\text{NULL} = \{X, Y\}$$

A	FIRST(A)
X	
Y	
Z	

## Re : exemple

$$Z \rightarrow XYZ \quad (1)$$

$$\quad | c \quad (2)$$

$$X \rightarrow a \quad (3)$$

$$\quad | Y \quad (4)$$

$$Y \rightarrow b \quad (5)$$

$$\quad | \varepsilon \quad (6)$$

$$\text{NULL} = \{X, Y\}$$

A	FIRST(A)
X	a, b
Y	b
Z	c, a, b

A	FOLLOW(A)
X	
Y	
Z	



## Re : exemple

$$Z \rightarrow XYZ \quad (1)$$

$$| c \quad (2)$$

$$X \rightarrow a \quad (3)$$

$$| Y \quad (4)$$

$$Y \rightarrow b \quad (5)$$

$$| \varepsilon \quad (6)$$

$$\text{NULL} = \{X, Y\}$$

A	FIRST(A)
X	a, b
Y	b
Z	c, a, b

A	FOLLOW(A)
X	a, b, c
Y	a, b, c
Z	(rien)

TABLE	a	b	c
X			
Y			
Z			

## Re : exemple

$$Z \rightarrow XYZ \quad (1)$$

$$\quad | c \quad (2)$$

$$X \rightarrow a \quad (3)$$

$$\quad | Y \quad (4)$$

$$Y \rightarrow b \quad (5)$$

$$\quad | \varepsilon \quad (6)$$

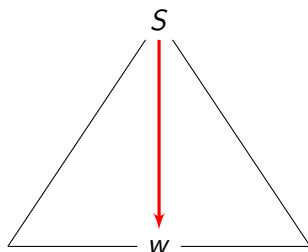
$$\text{NULL} = \{X, Y\}$$

A	FIRST(A)
X	a, b
Y	b
Z	c, a, b

A	FOLLOW(A)
X	a, b, c
Y	a, b, c
Z	(rien)

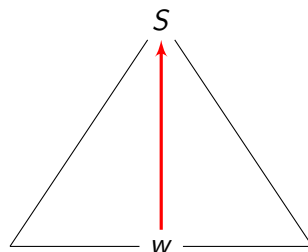
TABLE	a	b	c
X	3, 4	4	4
Y	6	5, 6	6
Z	1	1	1, 2

# Re : approches passage



descendante

↑  
LL



ascendante

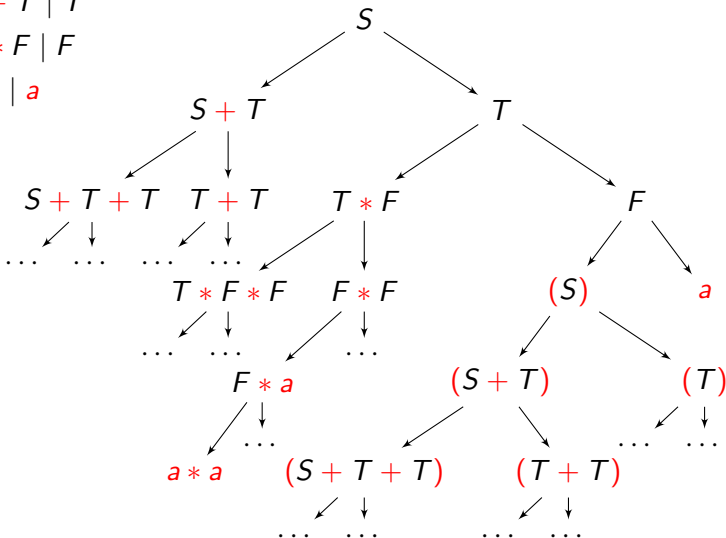
↑  
LR

# Plus précisément

$$S \rightarrow S + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (S) \mid a$$

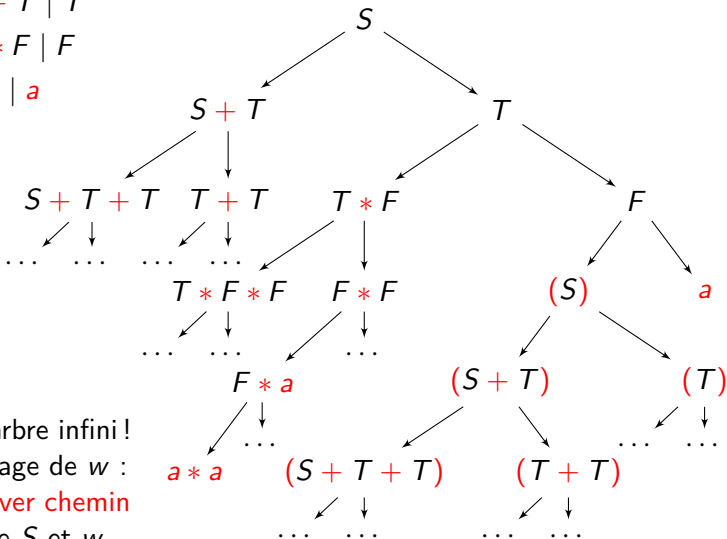


# Plus précisément

$$S \rightarrow S + T \mid T$$

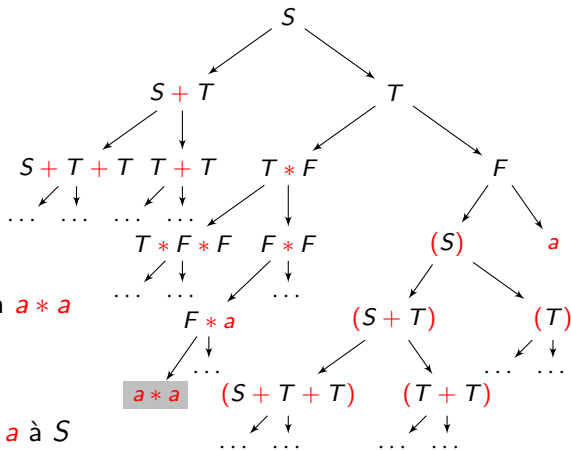
$$T \rightarrow T * F \mid F$$

$$F \rightarrow (S) \mid a$$



- un arbre infini !
- parsage de  $w$  :  
trouver chemin  
entre  $S$  et  $w$

# Approches parsing



Approche descendante :

- trouver chemin de  $S$  à  $a * a$
- difficile ici !

Approche ascendante :

- trouver chemin de  $a * a$  à  $S$
- ici, facile !
  - l'inverse de l'arbre est ( presque ) **déterministe**
  - ( pas toujours le cas ... )

# Parsage ascendant : the basics

```
function BULRP( $\alpha$ )  
  if  $\alpha = S$  then  
    return True  
  for  $i \leftarrow 1$  to  $|\alpha|$  do  
    for  $j \leftarrow i$  to  $|\alpha|$  do  
      for  $A \in N$  do  
        if  $A \rightarrow \alpha_i \dots \alpha_j$  then  
          if BULRP( $\alpha_1 \dots \alpha_{i-1} A \alpha_{j+1} \dots \alpha_n$ ) then  
            return True  
  return False
```

- très simple ☺
- plein de retours en arrière
- inutilisable?

# Parsage ascendant : the basics

```
function BULRP( $\alpha$ )  
  if  $\alpha = S$  then  
    return True  
  for  $i \leftarrow 1$  to  $|\alpha|$  do  
    for  $j \leftarrow i$  to  $|\alpha|$  do                                ▷ décalage / SHIFT  
      for  $A \in N$  do  
        if  $A \rightarrow \alpha_i \dots \alpha_j$  then                            ▷ réduction / REDUCE  
          if BULRP( $\alpha_1 \dots \alpha_{i-1} A \alpha_{j+1} \dots \alpha_n$ ) then  
            return True  
  return False
```

- très simple 😊
- plein de retours en arrière
- inutilisable?



# Exemple

$S \rightarrow (S) \quad (1)$   
 $\quad \quad | n \quad \quad (2)$

- utiliser une **pile** pour stocker les entrées décalés

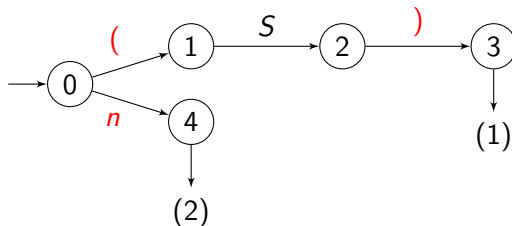
entrée	pile	action
$((n))\$$	$\perp$	décaler
$(n))\$$	$\perp($	décaler
$n))\$$	$\perp(($	décaler
$)\$$	$\perp((n$	réduire 2
$)\$$	$\perp((S$	décaler
$)\$$	$\perp((S)$	réduire 1
$)\$$	$\perp(S$	décaler
$\$$	$\perp(S)$	réduire 1
$\$$	$\perp$	PROFIT

# Automate de parsing

$$S \rightarrow (S) \quad (1)$$

$$| n \quad (2)$$

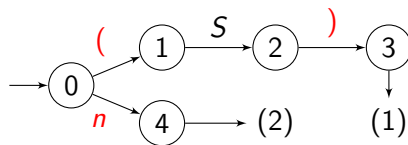
- reconnaissance de motifs par **automate fini déterministe**



- ( en fait un **transducteur fini** )

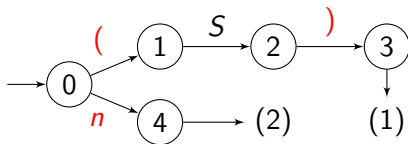
# Optimisations

$$S \rightarrow (S) \quad (1)$$

$$| \textcolor{red}{n} \quad (2)$$


entrée	pile	action
$((\textcolor{red}{n}))\$$	$\perp$	décaler
$(\textcolor{red}{n}))\$$	$\perp ($	décaler
$n))\$$	$\perp (($	décaler
$))\$$	$\perp ((\textcolor{red}{n}$	réduire 2
$))\$$	$\perp ((S$	décaler
$)\$$	$\perp ((S)$	réduire 1
$)\$$	$\perp (S$	décaler
$\$$	$\perp (S)$	réduire 1
$\$$	$\perp$	PROFIT

# Optimisations

 $S \rightarrow (S) \quad (1)$ 
 $| \quad n \quad (2)$ 


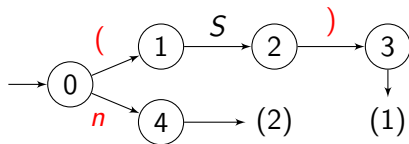
- stocker l'état actuel dans la pile

entrée	pile	action
$((n))\$$	$\perp$	décaler
$(n))\$$	$\perp (1$	décaler
$n))\$$	$\perp (1 (1$	décaler
$) )\$$	$\perp (1 (1 n_4$	réduire 2
$) )\$$	$\perp (1 (1 S_2$	décaler
$) \$$	$\perp (1 (1 S_2)_3$	réduire 1
$) \$$	$\perp (1 S_2$	décaler
$\$$	$\perp (1 S_2)_3$	réduire 1
$\$$	$\perp$	PROFIT

# Optimisations

$$S \rightarrow (S) \quad (1)$$

$$| n \quad (2)$$



- stocker l'état actuel dans la pile
- en fait, plus besoin d'empiler des symboles !

entrée	pile	action
$((n))\$$	$\perp$	décaler
$(n))\$$	$\perp 1$	décaler
$n))\$$	$\perp 11$	décaler
$) ) \$$	$\perp 114$	réduire 2
$) ) \$$	$\perp 112$	décaler
$) \$$	$\perp 1123$	réduire 1
$) \$$	$\perp 12$	décaler
$\$$	$\perp 123$	réduire 1
$\$$	$\perp$	PROFIT

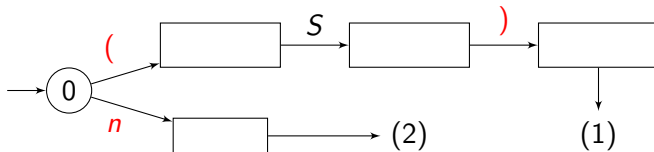
# Parsage LR

# Construire l'automate de parsage

- états : productions partiellement achevées

$$S \rightarrow (S) \quad (1)$$

$$| n \quad (2)$$

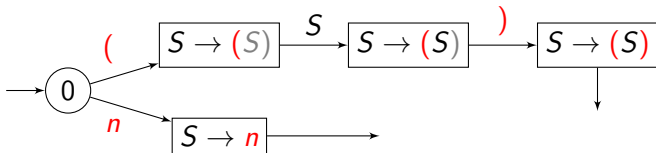


# Construire l'automate de parcage

- états : productions partiellement achevées

$$S \rightarrow (S) \quad (1)$$

$$| n \quad (2)$$



## Définition (8.8)

Soit  $G$  une grammaire hors-contexte. Une **production pointée** de  $G$  est une paire  $(A, \alpha \bullet \beta)$  telle que  $A \rightarrow \alpha \beta$  est une production de  $G$ .

- $\alpha$  : partie achevée,  $\beta$  : ce qui reste à trouver
- états : productions pointées
- transitions étiquetées dans  $V$
- états finaux : productions de type  $A \rightarrow w \bullet$

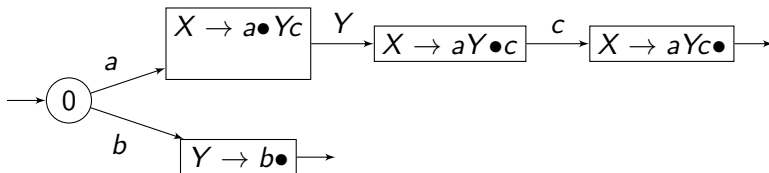


# But Wait !

- plus compliqué que ça :

$$X \rightarrow aYc \quad (1)$$

$$Y \rightarrow b \quad (2)$$

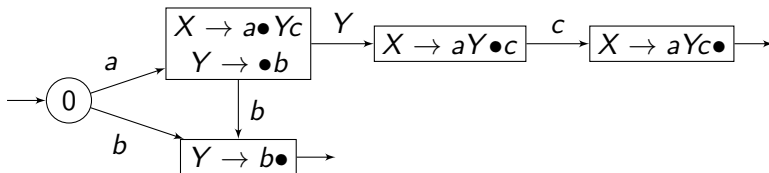


# But Wait !

- plus compliqué que ça :

$$X \rightarrow aYc \quad (1)$$

$$Y \rightarrow b \quad (2)$$



# Clôture

## Définition (8.10)

Soit  $G$  une grammaire hc et  $\mathcal{I}$  un ensemble de productions pointées de  $G$ . La **clôture** de  $\mathcal{I}$  est le plus petit ensemble  $\text{cl}(\mathcal{I})$  t.q.  $\mathcal{I} \subseteq \text{cl}(\mathcal{I})$  et

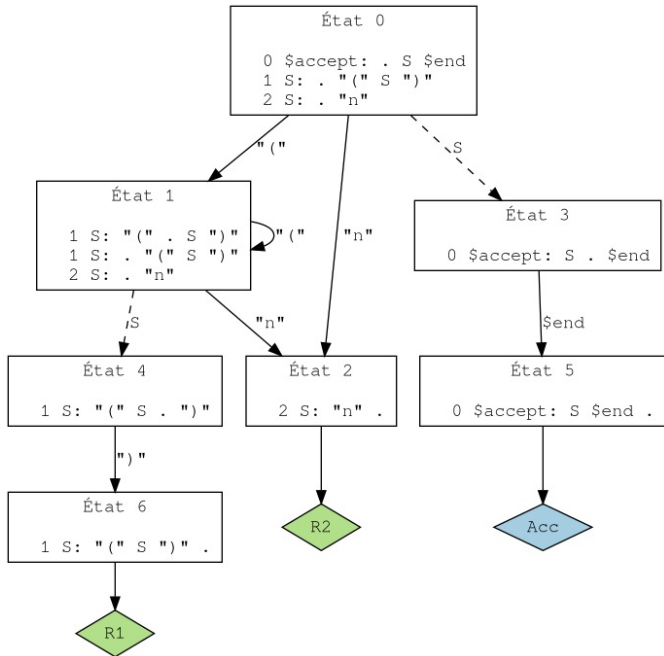
- si  $(A, \alpha \bullet B \beta) \in \text{cl}(\mathcal{I})$  et  $B \rightarrow \gamma$  est une production de  $G$ , alors  $(B, \bullet \gamma) \in \text{cl}(\mathcal{I})$ .

## Définition

L'**automate de passage LR(0)** d'une grammaire hors-contexte  $G$  est l'automate fini déterministe  $(Q, q_0, F, \delta)$  avec

- $Q = \{\text{cl}(\mathcal{I}) \mid \mathcal{I} \text{ ensemble de productions pointées de } G\}$  ;
- $q_0 = \text{cl}(\{(Z, \bullet S \$)\})$  ;
- $F = \{q \in Q \mid \exists \text{ production } X \rightarrow w \text{ de } G \text{ t.q. } (X, w \bullet) \in q\}$
- et  $\delta : Q \times V \rightarrow Q$  donnée par

$$\delta(q, \beta) = \text{cl}(\{(X, \alpha \beta \bullet \gamma) \mid (X, \alpha \bullet \beta \gamma) \in q\}).$$





*That's all Folks!*