Théorie des langages : THL

CM 9

Uli Fahrenberg

EPITA Rennes

S5 2024

Aperçu

Programme du cours

1. Langages rationnels, automates finis
   - TP 1 : `flex`
3. Langages algébriques, grammaires hors-contexte, automates à pile
4. Parsage LL
   - TP 2 : LL
6. Parsage LR
9. Conclusion
   - TP 3 : `bison`
   - TP 4 : `flex` & `bison`

Résumé du cours

Hiérarchie de Chomsky

type	langages	grammaires	automates
4	finis	à choix finis	finis acycliques
	$\subsetneq \cap$	$\Downarrow$	$\Downarrow$
3	rationnels	régulières	finis
	$\subsetneq \cap$	$\Downarrow$	$\Downarrow$
2	algébriques	hors-contexte	à pile
	$\subsetneq \cap$	$\Downarrow$	
1	contextuels	contextuelles	linéairement bornés
	$\subsetneq \cap$	$\Downarrow$	$\Downarrow$
0	récursivement énumerables	syntagmatiques	de Turing

Hiérarchie de Chomsky

type	langages	grammaires	automates
4	finis	à choix finis	finis acycliques
	⊊∩	⇓	⇓
3	rationnels	régulières	finis
	⊊∩	⇓	⇓
2	algébriques	hors-contexte	à pile
	⊊∩	⇓	
1	contextuels	contextuelles	linéairement bornés
	⊊∩	⇓	⇓
0	récursivement énumerables	syntagmatiques	de Turing

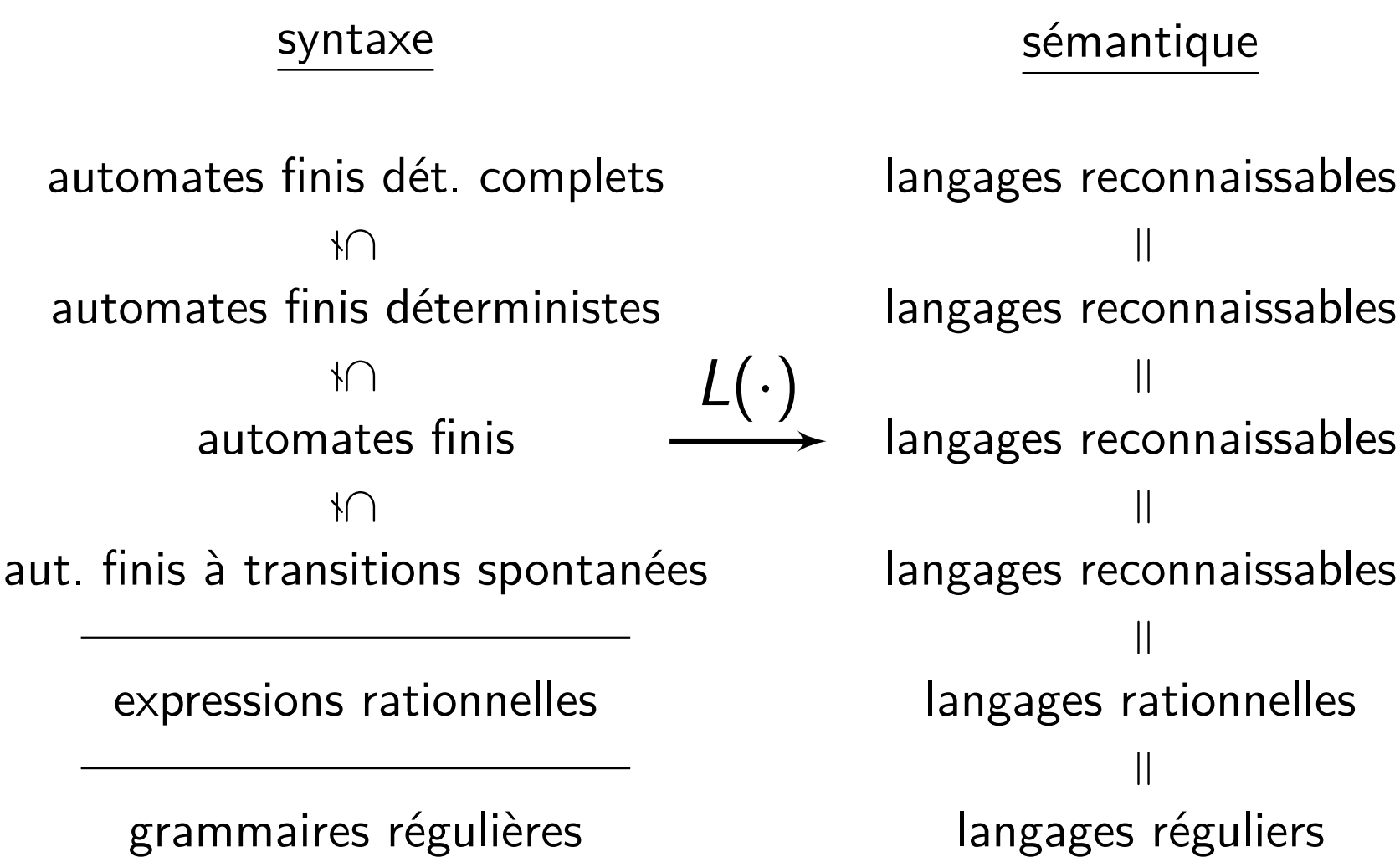
Dans le poly

1. Langages rationnels, automates finis
   - chapitre 2 sauf 2.3.2, 2.3.3, 2.3.4, 2.3.5, 2.4.4
   - chapitre 3 sauf 3.1.3
   - chapitre 4 sauf 4.1.3, 4.2.1, 4.3, 4.4
3. Langages algébriques, grammaires hors-contexte, automates à pile
   - chapitre 6 sauf 6.3.1
   - 9.2.2
   - Sipser 2.2
4. Parsage LL
   - chapitre 7
   - section 8.1
6. Parsage LR
   - section 8.2

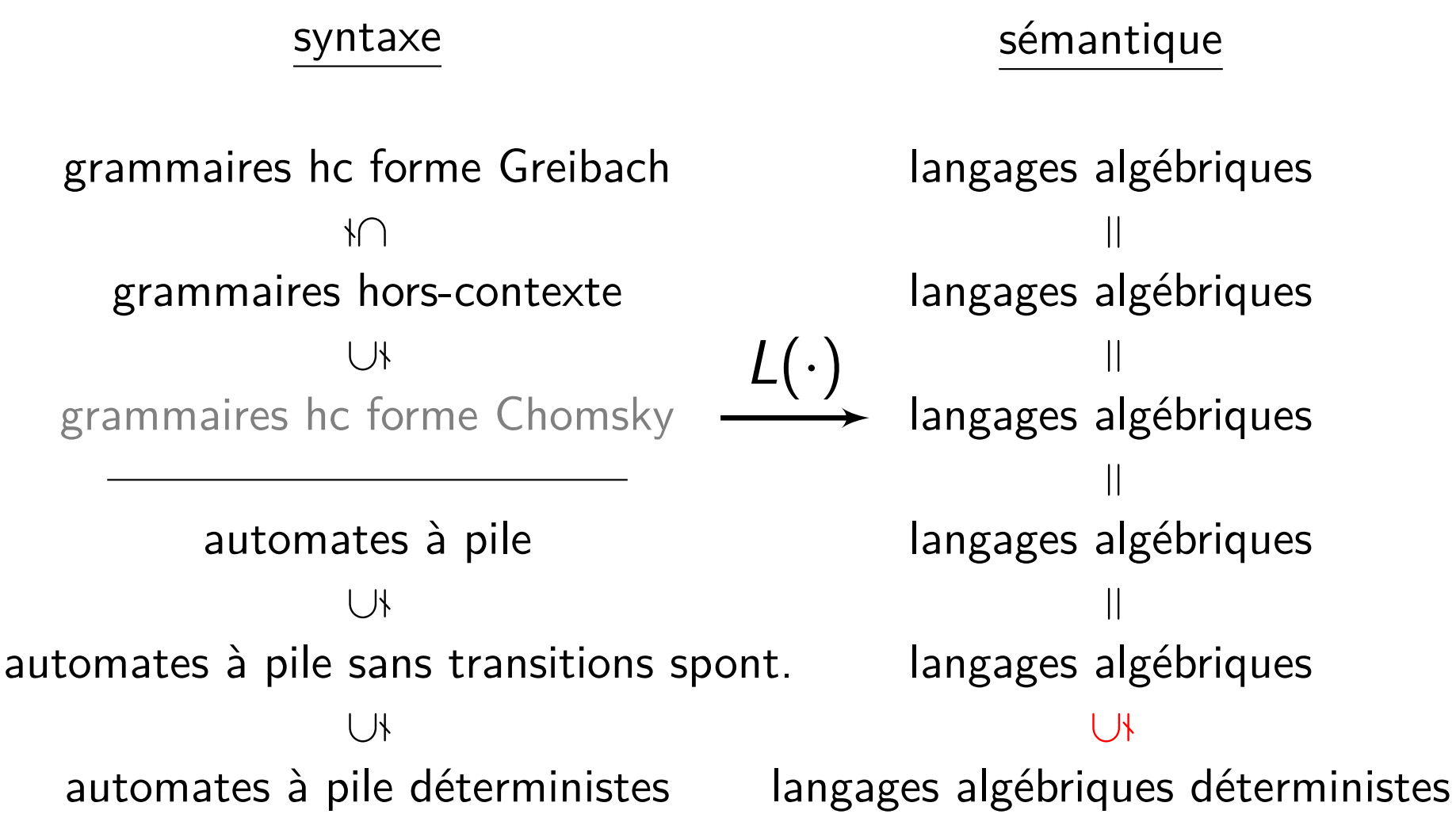
Rationnel vs. algébrique

langages rationnels	langages algébriques
grammaires régulières :	grammaires hors contexte :
– linéaire à droite : $N \to \Sigma N \mid \Sigma \mid \varepsilon$	– $N \to (N \cup \Sigma)^*$
– linéaire à gauche : $N \to N\Sigma \mid \Sigma \mid \varepsilon$	– Greibach : $N \to \Sigma NN \mid \Sigma N \mid \Sigma \mid \varepsilon$
automates finis	automates à pile
– déterministes / non-déterministes	– pas de déterminisation
décidabilité :	décidabilité :
– appartenance ($w \in L$) ✓	– appartenance ($w \in L$) ✓
– vacuité ($L = \emptyset$) ✓	– vacuité ($L = \emptyset$) ✓
– universalité ($L = \Sigma^*$) ✓	– universalité ($L = \Sigma^*$) ✗

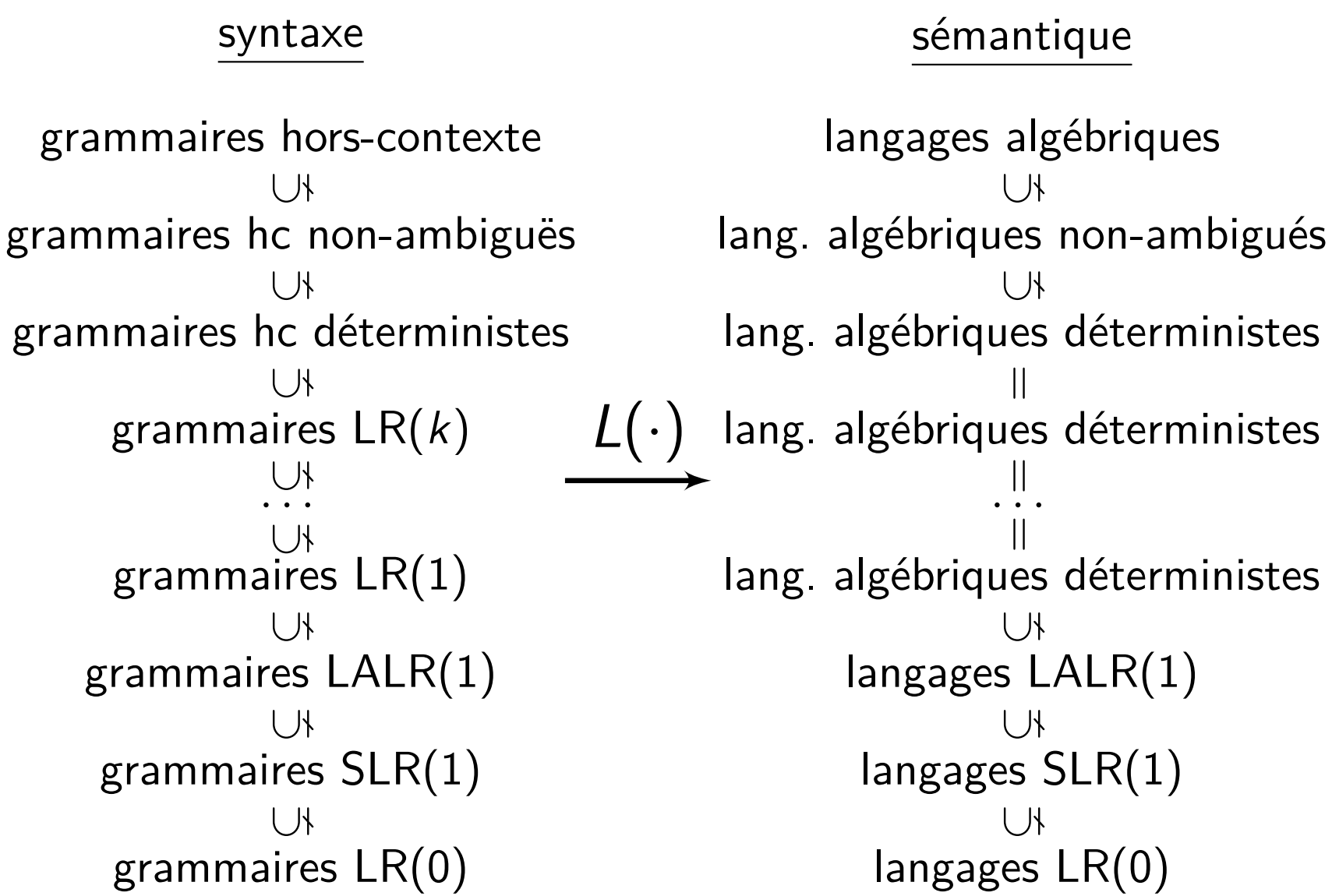
Zoom sur type 3 (régulier / rationnel)



Zoom sur type 2 (hors contexte / algébrique)



Zoom sur LR

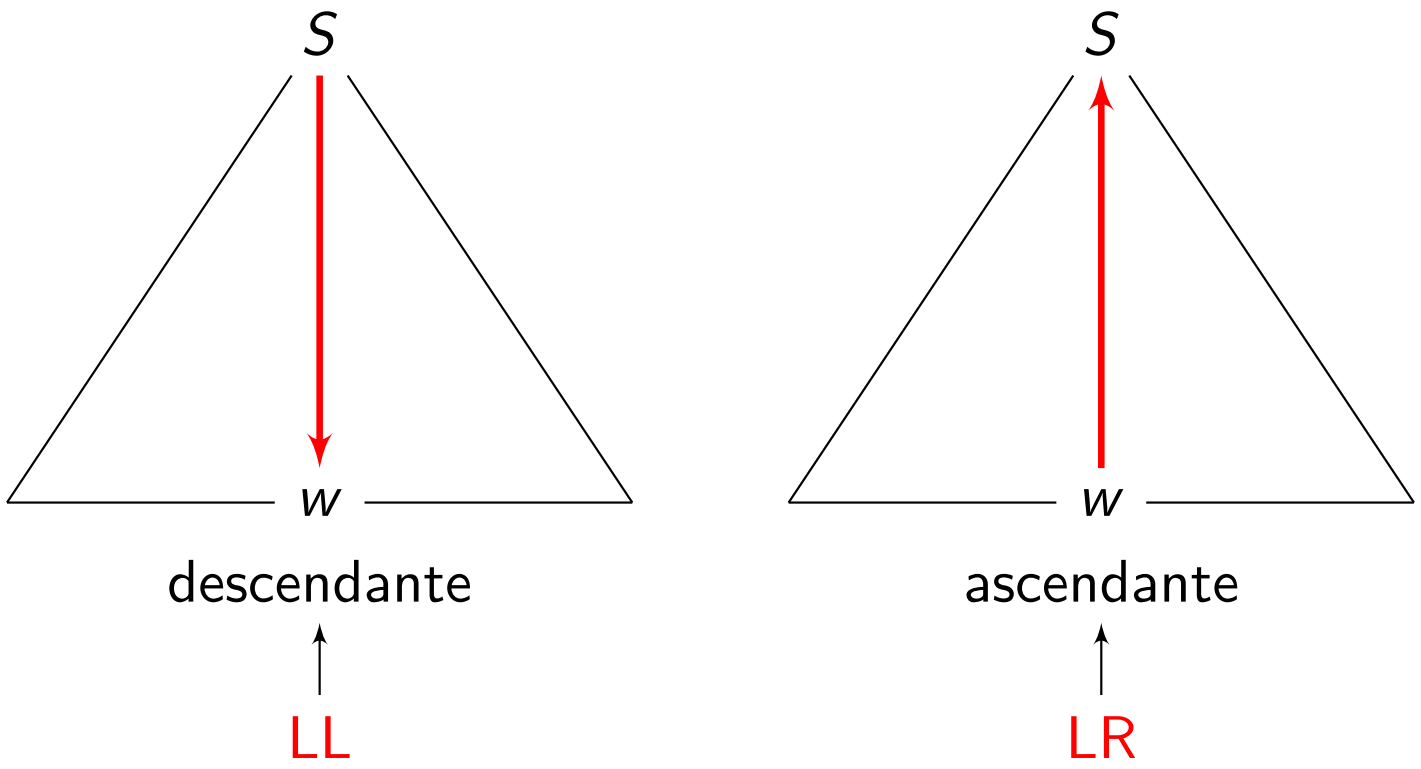

syntaxe
sémantique
grammaires hors-contexte
langages algébriques
grammaires hc non-ambiguës
lang. algébriques non-ambigués
grammaires hc déterministes
lang. algébriques déterministes
grammaires LR(k)
L(·)
lang. algébriques déterministes
grammaires LR(1)
lang. algébriques déterministes
grammaires LALR(1)
langages LALR(1)
grammaires SLR(1)
langages SLR(1)
grammaires LR(0)
langages LR(0)

Zoom sur LL

syntaxe		sémantique
grammaires hors-contexte		langages algébriques
$\cup\!\!\!\!\!\!+$		$\cup\!\!\!\!\!\!+$
grammaires hc non-ambiguës		lang. algébriques non-ambigués
$\cup\!\!\!\!\!\!+$		$\cup\!\!\!\!\!\!+$
grammaires hc déterministes		lang. algébriques déterministes
$\cup\!\!\!\!\!\!+$		$\cup\!\!\!\!\!\!+$
grammaires LL(k)	$\xrightarrow{L(\cdot)}$	langages LL(k)
$\cup\!\!\!\!\!\!+$		$\cup\!\!\!\!\!\!+$
$\cdots$		$\cdots$
$\cup\!\!\!\!\!\!+$		$\cup\!\!\!\!\!\!+$
grammaires LL(2)		langages LL(2)
$\cup\!\!\!\!\!\!+$		$\cup\!\!\!\!\!\!+$
grammaires LL(1)		langages LL(1)

Parsage

Parsage



Parsage LL(1) : approche descendante

0. 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
1. calculer NULL
   - $\text{NULL} = \{A \in N \mid A \Rightarrow^* \varepsilon\}$
2. construire la table FIRST
   - $\text{FIRST}(A) = \{a \in \Sigma \mid \exists w \in V^* : A \Rightarrow^* aw\}$
3. construire la table FOLLOW
   - $\text{FOLLOW}(A) = \{a \in \Sigma \mid \exists B \in N, \alpha, \beta \in V^* : B \Rightarrow^* \alpha A a \beta\}$
4. construire la TABLE de parsage :
   1. pour chaque production $X \to w \quad (n)$:
      1. pour chaque $a \in \text{FIRST}(w)$: TABLE(X, a) += $\{n\}$
      2. si $w \in \text{NULL}$ ou $w = \varepsilon$:
         – pour chaque $a \in \text{FOLLOW}(X)$: TABLE(X, a) += $\{n\}$

Re : parsage ascendant : the basics

function BULRP(α)
  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 $A \rightarrow \alpha_i \ldots \alpha_j$ **then** ▷ réduction / REDUCE
          if BULRP($\alpha_1 \ldots \alpha_{i-1} A \alpha_{j+1} \ldots \alpha_n$) **then**
            return True
  return False

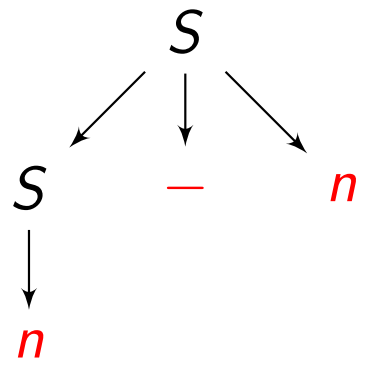
Re : parsage LR(0)

$$
\begin{array}{lll}
Z \to S\$ & & (0) \\
S \to S - n & & (1) \\
\quad \mid n & & (2)
\end{array}
$$

état	action	n	$-$	$\$$	S
0	décaler	1			2
1	réduire 2				
2	décaler		4	3	
3	accepter				
4	décaler	5			
5	réduire 1				

parser $n - n\$$:

entrée	pile	action	
$n - n\$$	$\perp 0$	décaler	
$- n\$$	$\perp 01$	réduire 2	$S \to n$
$- n\$$	$\perp 02$	décaler	
$n\$$	$\perp 024$	décaler	
$\$$	$\perp 0245$	réduire 1	$S \to S - n$
$\$$	$\perp 02$	décaler	
	$\perp 023$	✓	



Re : parsage SLR(1)

1. calculer la table LR(0)
2. si conflits : conditionner l'action par le FOLLOW

Exemple :

$$Z \to S\$ \quad (0)$$
$$S \to n{-}S \quad (1)$$
$$\mid n \quad (2)$$

état	action	n	$-$	$\$$	S
0	décaler	2			1
1	décaler			4	
2	réd. 2, déc.		3		
3	décaler	2			5
4	accepter				
5	réduire 1				

$\Longrightarrow$

état	n	$-$	$\$$	S
0	d.2			d.1
1			d.4	
2		d.3	r.2	
3	d.2			d.5
4	— accepter —			
5			r.1	

Re : parsage LR(1)

- conditionner l'action par le contexte : les symboles qui peuvent suivre

Exemple :

$$
\begin{array}{lll}
Z \to S\$ & & (0) \\
S \to L{=}E & & (1) \\
\quad \mid E & & (2) \\
L \to x & & (3) \\
\quad \mid *E & & (4) \\
E \to L & & (5)
\end{array}
$$

état	productions pointées élargies
0	$Z \to \bullet S\$\ [\varepsilon]$
	$S \to \bullet L{=}E\ [\$],\ S \to \bullet E\ [\$]$
	$L \to \bullet x\ [=],\ L \to \bullet *E\ [=]$
	$E \to \bullet L\ [\$]$
	$L \to \bullet x\ [\$],\ L \to \bullet *E\ [\$]$
1	$Z \to S\bullet\$\ [\varepsilon]$
2	$S \to L\bullet{=}E\ [\$],\ E \to L\bullet\ [\$\checkmark]$

Exemple

$Z \to S\$$ (0)
$S \to L{=}E$ (1)
$\quad\mid E$ (2)
$L \to x$ (3)
$\quad\mid *E$ (4)
$E \to L$ (5)

état	x	$*$	$=$	$\$$	S	L	E
0	d.4	d.5			d.1	d.2	d.3
1				d.6			
2			d.7	r.5			
3				r.2			
4			r.3	r.3			
5	d.4	d.5				d.9	d.8
6			— accepter —				
7	d.12	d.13				d.11	d.10
8			r.4				
9			r.5				
10				r.1			
11				r.5			
12				r.3			
13	d.12	d.13				d.11	d.14
14				r.4			

Parsage LALR(1)

Définition

Deux productions pointées élargies $A \to \alpha \bullet \beta$ [a] et $A \to \alpha' \bullet \beta'$ [b] sont équivalent LALR(1) si $\alpha = \alpha'$ et $\beta = \beta'$.

- les items sont identiques, mais les contextes peuvent être différents

Définition

L'automate LALR(1) d'une grammaire hors-contexte G est le quotient de l'automate LR(1) de G sous équivalence LALR(1).

Exemple, re

$Z \to S\$$ (0)

$S \to L{=}E$ (1)

$\quad\mid E$ (2)

$L \to x$ (3)

$\quad\mid *E$ (4)

$E \to L$ (5)

état	x	$*$	$=$	$\$$	S	L	E
0	d.4	d.5			d.1	d.2	d.3
1				d.6			
2			d.7	r.5			
3				r.2			
4			r.3	r.3			
5	d.4	d.5				d.9	d.8
6			— accepter —				
7	d.12	d.13				d.11	d.10
8			r.4				
9			r.5				
10				r.1			
11				r.5			
12				r.3			
13	d.12	d.13				d.11	d.14
14				r.4			

Exemple, re

$Z \to S\$$ (0)

$S \to L{=}E$ (1)

$\quad | \; E$ (2)

$L \to x$ (3)

$\quad | \; *E$ (4)

$E \to L$ (5)

état	x	$*$	$=$	$\$$	S	L	E
0	d.4	d.5			d.1	d.2	d.3
1				d.6			
2			d.7	r.5			
3				r.2			
4			r.3	r.3			
5	d.4	d.5				d.9	d.8
6			— accepter —				
7	d.12	d.13				d.11	d.10
8			r.4	r.4			
9			r.5	r.5			
10				r.1			

Résolution de conflits

Exemple :

$$
\begin{array}{lll}
Z \to E\$ & & (0) \\
E \to E{+}E & & (1) \\
\quad \mid E{*}E & & (2) \\
\quad \mid n & & (3)
\end{array}
$$

état	$+$	$*$	n	\$	E
0			d.2		g.1
1	d.4	d.5		d.3	
2	r.3	r.3		r.3	
3			— accepter —		
4			d.2		g.6
5			d.2		g.7
6	d.4	d.5			
	r.1	r.1		r.1	
7	d.4	d.5			
	r.2	r.2		r.2	

- une grammaire ambiguë
- donc pas LR(k) pour n'importe quel k

Résolution de conflits

Exemple :

$$
\begin{array}{lll}
Z \to E\$ & & (0) \\
E \to E{+}E & & (1) \\
\quad \mid E{*}E & & (2) \\
\quad \mid n & & (3)
\end{array}
$$

état	$+$	$*$	n	\$	E
0			d.2		g.1
1	d.4	d.5		d.3	
2	r.3	r.3		r.3	
3			— accepter —		
4			d.2		g.6
5			d.2		g.7
6	d.4	d.5			
	r.1	r.1		r.1	
7	d.4	d.5			
	r.2	r.2		r.2	

- une grammaire ambiguë
- donc pas LR(k) pour n'importe quel k
- associativité : d.4 $\Rightarrow n + (n + n)$; r.1 $\Rightarrow (n + n) + n$
- priorité : d.5 $\Rightarrow n * (n + n)$; r.1 $\Rightarrow (n * n) + n$

Résolution de conflits

Exemple :

$$
\begin{aligned}
Z &\to E\$ && (0)\\
E &\to E{+}E && (1)\\
&\mid E{*}E && (2)\\
&\mid n && (3)
\end{aligned}
$$

état	$+$	$*$	n	\$	E
0			d.2		g.1
1	d.4	d.5		d.3	
2	r.3	r.3		r.3	
3			— accepter —		
4			d.2		g.6
5			d.2		g.7
6	d.4	d.5			
	r.1	r.1		r.1	
7	d.4	d.5			
	r.2	r.2		r.2	

- une grammaire ambiguë
- donc pas LR(k) pour n'importe quel k
- associativité : d.4 $\Rightarrow n+(n+n)$; r.1 $\Rightarrow (n+n)+n$
- priorité : d.5 $\Rightarrow n*(n+n)$; r.1 $\Rightarrow (n*n)+n$
- solution : règles de priorité
- ici : r.1 $>$ d.4, r.2 $>$ d.5, r.2 $>$ d.4, d.5 $>$ r.1 $\Leftarrow$!

Parsage LR généralisé

- *embrace non-determinism!*
- parsage GLR : en cas de conflit, suivre tous les chemins en parallel
- « parsage parallel », « parsage Tomita »
- implémenter l'automate (non-déterministe) de parsage sans déterminisation
- états : productions pointées, pas de clôture
- algorithme en temps exponentiel, pas linéaire
- optimisation : partager préfixes et suffixes de piles


That's all Folks!