Théorie des langages : THL

CM 7

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`
5. Parsage LL
7. Parsage LR, partie 1
   - TP 2 : parsage LL
8. Parsage LR, partie 2
9. Parsage LR, partie 3
10. Introduction `flex` & `bison`
11. TP 3, 4 : `flex` & `bison`

Re : parsage ascendant : the basics

function $\text{BULRP}(\alpha)$
  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 $A \to \alpha_i \dots \alpha_j$ **then** ▷ réduction / REDUCE
          return $\text{BULRP}(\alpha_1 \dots \alpha_{i-1} A \alpha_{j+1} \dots \alpha_n)$
  return False

Définition (8.8)

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

Re : automate de parsage LR(0)

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 $\mathrm{cl}(\mathcal{I})$ t.q. $\mathcal{I} \subseteq \mathrm{cl}(\mathcal{I})$ et

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

Définition

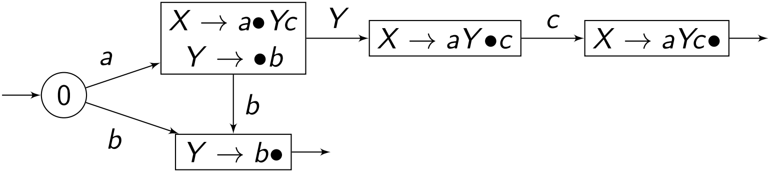
L'automate de parsage LR(0) d'une grammaire hors-contexte G est l'automate fini déterministe (Q, q_0, F, δ) avec

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

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

Re : exemple

$$X \to aYc \qquad (1)$$
$$Y \to b \qquad (2)$$



Dans le poly

1. Langages rationnels — 2.1, 2.2, 2.3.1, 2.4, 3.1.1, 3.1.2, 3.2
2. Automates finis — 4.1, 4.2.2
3. Langages algébriques, grammaires hors-contexte, automates à pile — 5.1, 5.2.3, 5.2.4, 5.3.6, 6.2, plus Sipser 2.2
4. Parsage LL — 7, 8.1
5. Parsage LR — 8.2

Parsage LR(0)

Algorithme de parsage

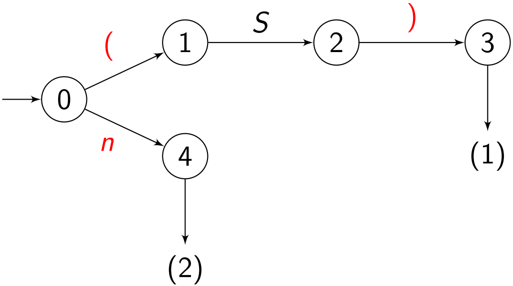
1. empiler q_0
2. `repeat`
    1. $q \leftarrow$ état en haut de la pile
    2. si $q =$ état final $X \to w\bullet$: REDUCE
        1. dépiler $|w|$ états
        2. $q' \leftarrow$ état en haut de la pile
        3. empiler $\delta(q', X)$
    3. sinon : SHIFT
        1. $a \leftarrow \text{next}(\text{input})$
        2. empiler $\delta(q, a)$
3. `until` $q =$ état final $Z \to S\$\bullet$ (✓) ou échec (✗)

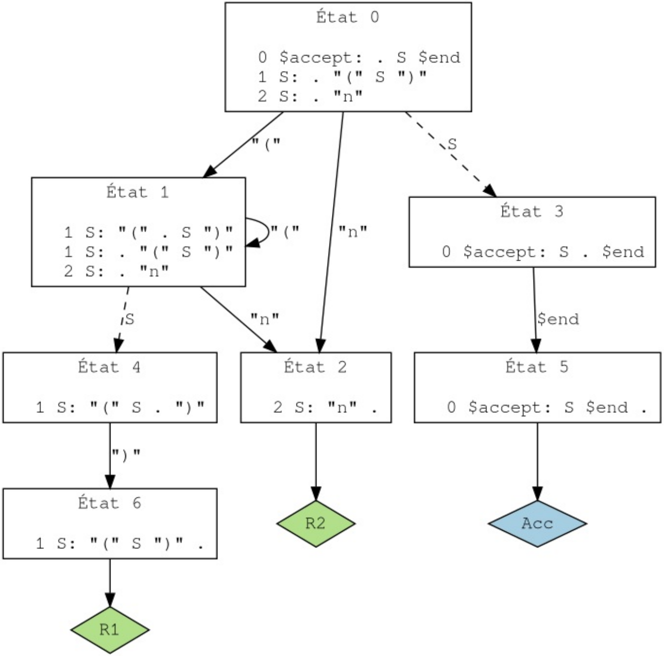
Algorithme de parsage

1. empiler q_0
2. `repeat`
   1. $q \leftarrow$ état en haut de la pile
   2. si $q =$ état final $X \to w\bullet$: REDUCE
      1. dépiler $|w|$ états
      2. $q' \leftarrow$ état en haut de la pile
      3. empiler $\delta(q', X)$ $\longleftarrow$ possible ✗
   3. sinon : SHIFT
      1. $a \leftarrow \text{next}(\text{input})$ $\longleftarrow$ possible ✗
      2. empiler $\delta(q, a)$ $\longleftarrow$ possible ✗
3. `until` $q =$ état final $Z \to S\$\bullet$ (✓) ou échec (✗)

Exemple

$$\begin{aligned} S &\to (S) && (1) \\ &\mid n && (2) \end{aligned}$$

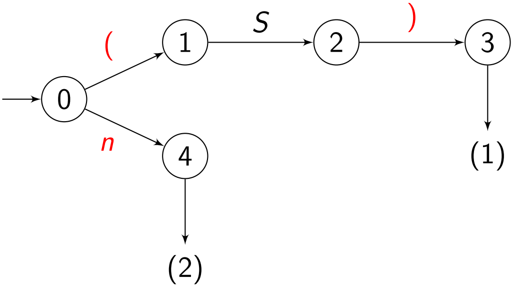




État 0
0 $accept: . S $end
1 S: . "(" S ")"
2 S: . "n"
"("
S
"n"
État 1
1 S: "(" . S ")"
1 S: . "(" S ")"
2 S: . "n"
"("
État 3
0 $accept: S . $end
S
"n"
$end
État 4
1 S: "(" S . ")"
État 2
2 S: "n" .
État 5
0 $accept: S $end .
")"
État 6
1 S: "(" S ")" .
R2
Acc
R1

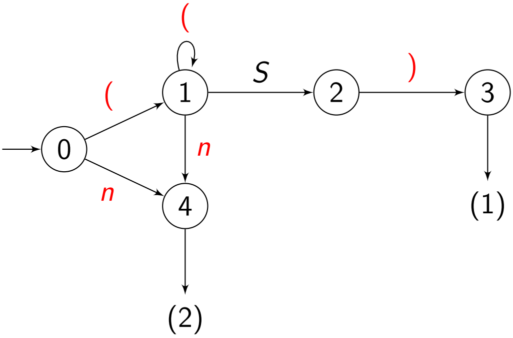
Exemple : re

$$S \to (S) \quad (1)$$
$$\mid n \quad (2)$$



Exemple : re

$$
\begin{aligned}
S &\to (S) \qquad (1)\\
&\mid n \qquad (2)
\end{aligned}
$$



Exemple : re re

$$
\begin{aligned}
Z &\to S\$ && (0) \\
S &\to (S) && (1) \\
&\mid n && (2)
\end{aligned}
$$

$$\longrightarrow \boxed{Z \to \bullet S\$}$$

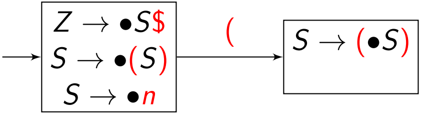
Exemple : re re

$$
\begin{aligned}
Z &\to S\$ && (0)\\
S &\to (S) && (1)\\
&\mid n && (2)
\end{aligned}
$$

$$
\longrightarrow \boxed{\begin{array}{c} Z \to \bullet S\$ \\ S \to \bullet (S) \\ S \to \bullet n \end{array}}
$$

Exemple : re re

$$
\begin{aligned}
Z &\to S\$ && (0)\\
S &\to (S) && (1)\\
&\mid n && (2)
\end{aligned}
$$



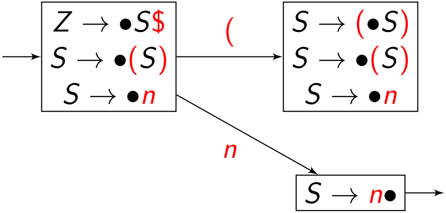
Exemple : re re

$$
\begin{aligned}
Z &\to S\$ && (0) \\
S &\to (S) && (1) \\
&\mid n && (2)
\end{aligned}
$$

$$
\begin{tikzcd}
\arrow[r] & \boxed{\begin{array}{c} Z \to \bullet S\$ \\ S \to \bullet(S) \\ S \to \bullet n \end{array}} \arrow[r, "("] & \boxed{\begin{array}{c} S \to (\bullet S) \\ S \to \bullet(S) \\ S \to \bullet n \end{array}}
\end{tikzcd}
$$

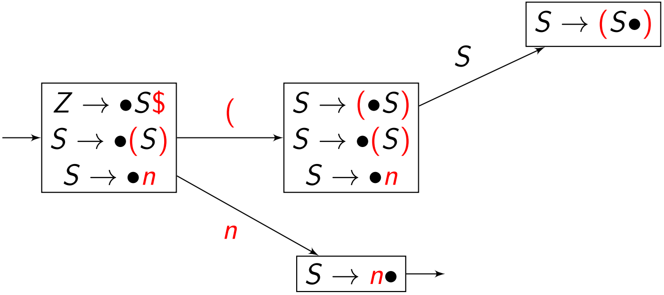
Exemple : re re

$$
\begin{aligned}
Z &\to S\$ && (0)\\
S &\to (S) && (1)\\
&\mid n && (2)
\end{aligned}
$$



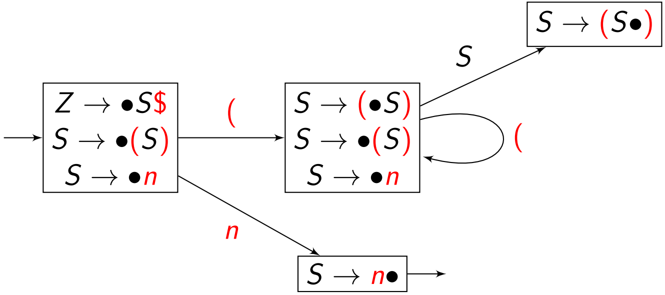
Exemple : re re

$$Z \to S\$ \qquad (0)$$
$$S \to (S) \qquad (1)$$
$$\mid n \qquad (2)$$



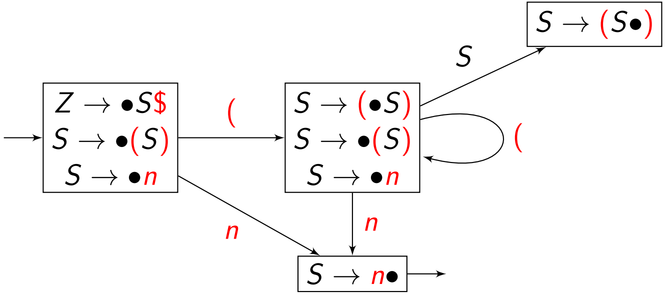
Exemple : re re

$$
\begin{aligned}
Z &\to S\$ && (0) \\
S &\to (S) && (1) \\
&\mid n && (2)
\end{aligned}
$$



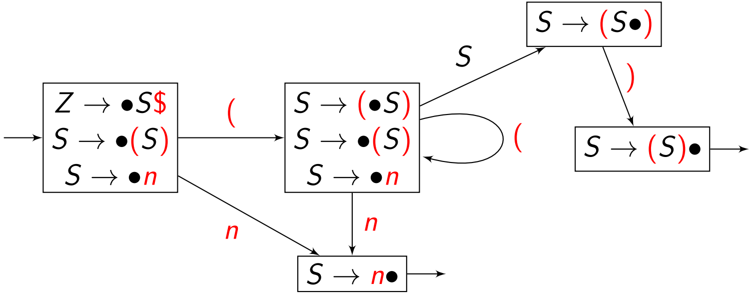
Exemple : re re

$$
\begin{aligned}
Z &\to S\$ && (0)\\
S &\to (S) && (1)\\
&\mid n && (2)
\end{aligned}
$$



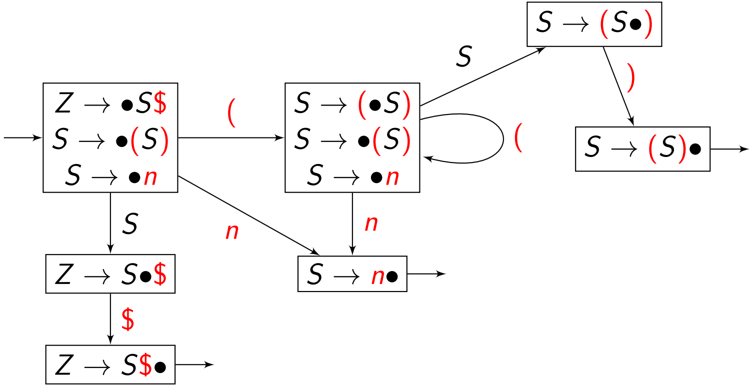
Exemple : re re

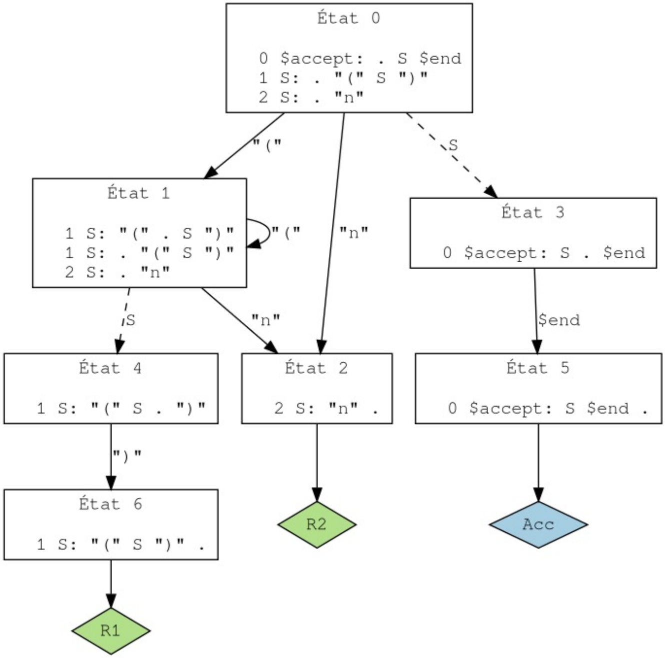
$$
\begin{aligned}
Z &\to S\$ && (0)\\
S &\to (S) && (1)\\
&\mid n && (2)
\end{aligned}
$$



Exemple : re re

$$
\begin{array}{rcll}
Z & \to & S\$ & (0) \\
S & \to & (S) & (1) \\
 & | & n & (2)
\end{array}
$$





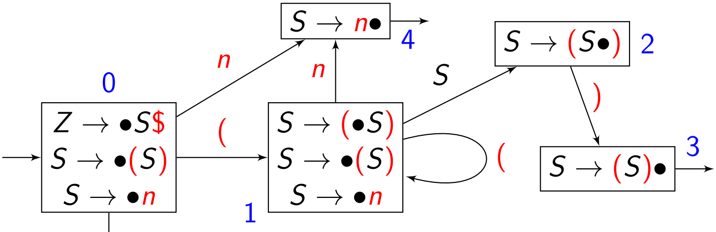
État 0
0 $accept: . S $end
1 S: . "(" S ")"
2 S: . "n"
"("
"n"
S
État 1
1 S: "(" . S ")"
1 S: . "(" S ")"
2 S: . "n"
"("
État 3
0 $accept: S . $end
S
"n"
$end
État 4
1 S: "(" S . ")"
État 2
2 S: "n" .
État 5
0 $accept: S $end .
")"
État 6
1 S: "(" S ")" .
R2
Acc
R1

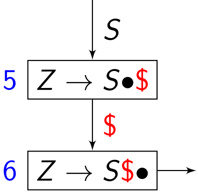
Exemple : table de parsage

$Z \to S\$ \quad (0)$

$S \to (S) \quad (1)$

$\quad\ \mid n \quad (2)$





state	action	(	n	)	\$	S
0	shift	1	4			5
1	shift	1	4			2
2	shift			3		
3	reduce 1					
4	reduce 2					
5	shift				6	
6	accept					

Autre exemple

$$Z \to S\$ \qquad (0)$$
$$S \to S - n \qquad (1)$$
$$\mid n \qquad (2)$$

état	productions pointées
0	$Z \to \bullet S\$$, $S \to \bullet S - n$, $S \to \bullet n$
1	$S \to n\bullet$
2	$Z \to S\bullet\$$, $S \to S\bullet - n$
3	$Z \to S\$\bullet$
4	$S \to S - \bullet n$
5	$S \to S - n\bullet$

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

Autre exemple

$$Z \to S\$ \qquad (0)$$
$$S \to S - n \qquad (1)$$
$$\mid n \qquad (2)$$

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

Autre exemple

$$Z \to S\$ \qquad (0)$$
$$S \to S - n \qquad (1)$$
$$\mid n \qquad (2)$$

parser $n - n\$$:

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

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

Autre exemple

$$Z \to S\$ \qquad (0)$$
$$S \to S - n \qquad (1)$$
$$\mid n \qquad (2)$$

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	
	$\perp 023$	✓	

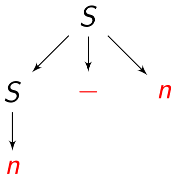
Autre exemple

$$Z \to S\$ \qquad (0)$$
$$S \to S - n \qquad (1)$$
$$\mid n \qquad (2)$$

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	
	$\perp 023$	✓	



Parsage LR(0)

- lire l'entrée de gauche à droite (L)
- approche ascendant
- construire une dérivation droite (R)
- pas de regard avant (0)

Parsage SLR(1)

Encore un exemple

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

état	productions pointées
0	$Z \to \bullet S\$$, $S \to \bullet n{-}S$, $S \to \bullet n$
1	$Z \to S \bullet \$$
2	$S \to n \bullet {-}S$, $S \to n\bullet$
3	$S \to n{-}\bullet S$, $S \to \bullet n{-}S$, $S \to \bullet n$
4	$Z \to S\$\bullet$
5	$S \to n{-}S\bullet$

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

Encore un exemple

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

état	productions pointées
0	$Z \to \bullet S\$,\ S \to \bullet n{-}S,\ S \to \bullet n$
1	$Z \to S\bullet\$$
2	$S \to n\bullet{-}S,\ S \to n\bullet$
3	$S \to n{-}\bullet S,\ S \to \bullet n{-}S,\ S \to \bullet n$
4	$Z \to S\$\bullet$
5	$S \to n{-}S\bullet$

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

conflit SHIFT/REDUCE

Conflits

- SHIFT/REDUCE
  - faut réduire avec $Y \to u$
  - ou décaler en attendant v ?

$$X \to u \bullet v$$
$$Y \to u \bullet$$

- REDUCE/REDUCE
  - faut réduire avec $X \to u$
  - ou réduire avec $Y \to u$?

$$X \to u$$
$$Y \to u$$

- utiliser FOLLOW pour résoudre
- (et pourquoi des conflits SHIFT/SHIFT n'existent pas ?)

Re : FOLLOW

Calculer des terminaux qui peuvent suivre un symbole dans une dérivation :

Définition

Soit $x \in V$, alors $\text{FOLLOW}(x) \subseteq \Sigma$ est défini par
$\text{FOLLOW}(x) = \{a \in \Sigma \mid \exists B \in N, \alpha, \beta \in V^* : B \Rightarrow^* \alpha x a \beta\}$.

Algorithme :

1. pour chaque $x \in V$: $\text{FOLLOW}(x) = \emptyset$
2. répéter jusqu'au point fixe :
   1. pour chaque $B \to \alpha x \beta \gamma$ avec $\beta \in \text{NULL}^*$:
      1. si $\gamma \notin \text{NULL}^*$: $\text{FOLLOW}(x)$ += $\text{FIRST}(\gamma)$
      2. si $\gamma \in \text{NULL}^*$: $\text{FOLLOW}(x)$ += $\text{FOLLOW}(B)$

Simple LR(1)

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

Exemple :

$$
\begin{array}{rcll}
Z & \to & S\$ & (0) \\
S & \to & n{-}S & (1) \\
 & | & n & (2)
\end{array}
$$

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	

Simple LR(1)

1. calculer la table LR(0)
2. si conflits : conditionner l'action par le FOLLOW
3. passer du type état $\to$ action $\to$ entrée
   au type état $\to$ entrée $\to$ action


That's all Folks!