ALF

Parser Bottom-Up

Bibliographie pour aujourd'hui

ALF
language crunch

Keith Cooper, Linda Torczon, *Engineering a Compiler*

- Chapitre 3
  - 3.4

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

- Chapitre 4
  - 4.5
  - 4.6
  - 4.7.4

Contenu

- LR
- SLR
- LALR

Limor Fried

- Américain
- MIT
- Adafruit
- STEM

Slides

Partie de slides sont écrie par

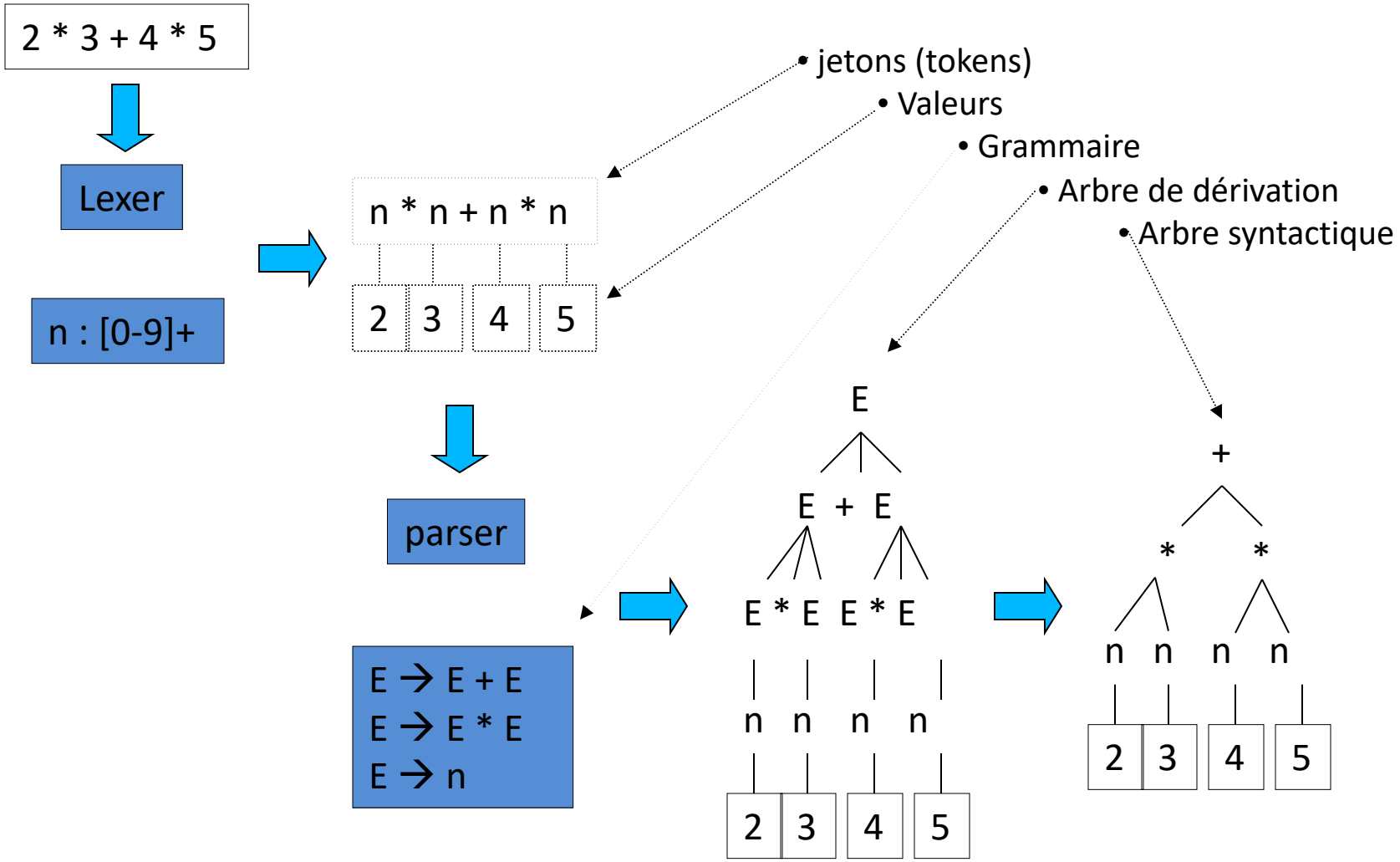
Bogdan Nitulescu

Notation BNF

ALF
language crunch

```
RFC 2616                        HTTP/1.1                        June 1999

       HTTP-date    = rfc1123-date | rfc850-date | asctime-date

       rfc1123-date = wkday "," SP date1 SP time SP "GMT“
       rfc850-date  = weekday "," SP date2 SP time SP "GMT“
       asctime-date = wkday SP date3 SP time SP 4DIGIT

       date1        = 2DIGIT SP month SP 4DIGIT
                      ; day month year (e.g., 02 Jun 1982)
       date2        = 2DIGIT "-" month "-" 2DIGIT
                      ; day-month-year (e.g., 02-Jun-82)
       date3        = month SP ( 2DIGIT | ( SP 1DIGIT ))
                      ; month day (e.g., Jun  2)
       time         = 2DIGIT ":" 2DIGIT ":" 2DIGIT
                      ; 00:00:00 - 23:59:59

       wkday        = "Mon" | "Tue" | "Wed“
                    | "Thu" | "Fri" | "Sat" | "Sun“
       weekday      = "Monday" | "Tuesday" | "Wednesday“
                    | "Thursday" | "Friday" | "Saturday" | "Sunday“
       month        = "Jan" | "Feb" | "Mar" | "Apr“
                    | "May" | "Jun" | "Jul" | "Aug"
                    | "Sep" | "Oct" | "Nov" | "Dec"
```

Arbre de dérivation / syntactique



Types d'analyse syntactique

ALF
language crunch

- Descendent (top-down)
  - Avec backtracking
  - Prédictive
    - Descendent récursive, LL avec un tableau
- Ascendant (bottom-up)
  - Avec backtracking
  - Shift-reduce
    - LR(0),SLR,LALR, LR canonique

Parser LL, LR

ALF
language crunch

- Nous devrions éviter backtracking
- Une grammaire qui permet le parser déterministe
  - LL(k) lit left-to-right, dérivation left
  - LR(k) lit left-to-right, dérivation right
  - K – lookahead (combien de tokens sont lus)
- LL(k) < LR(k)
- L'algorithme est indépendant du langage, la grammaire dépend du langage

Parser ascendant

Un analyseur vers le haut, ou shift-reduce commence à « feuille » et construit vers le haut de l'arbre de dérivation

Étapes de réduction suivent une dérivation droite dans l'ordre inverse

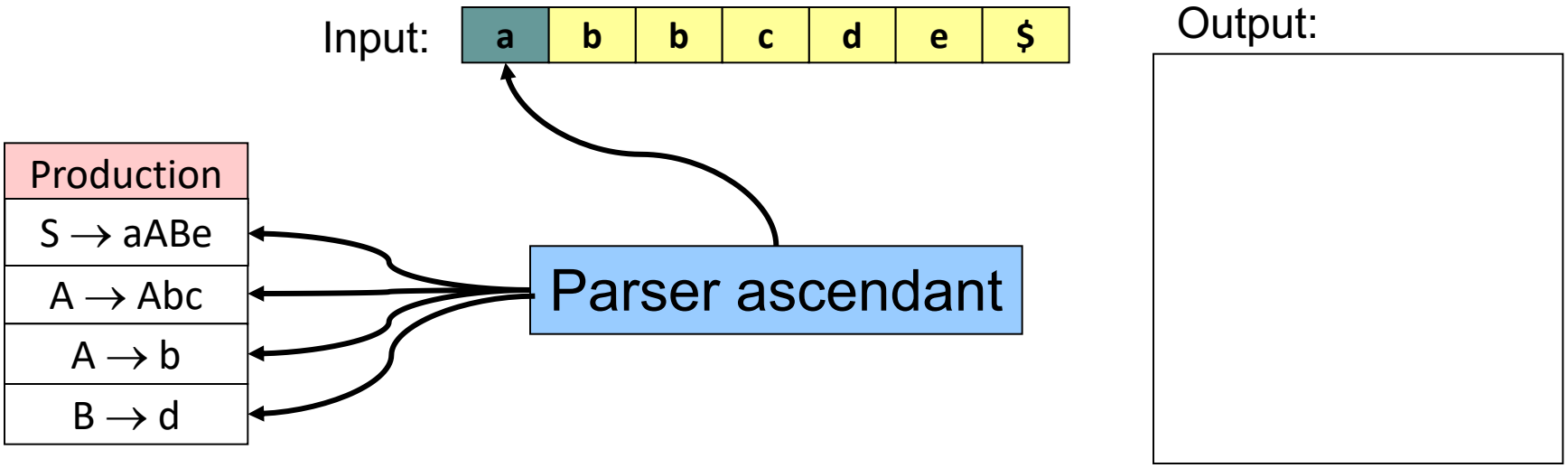
GRAMMAIRE:

$S \rightarrow aABe$
$A \rightarrow Abc \mid b$
$B \rightarrow d$

Nous analysons la chaîne d'entrée abbcde.

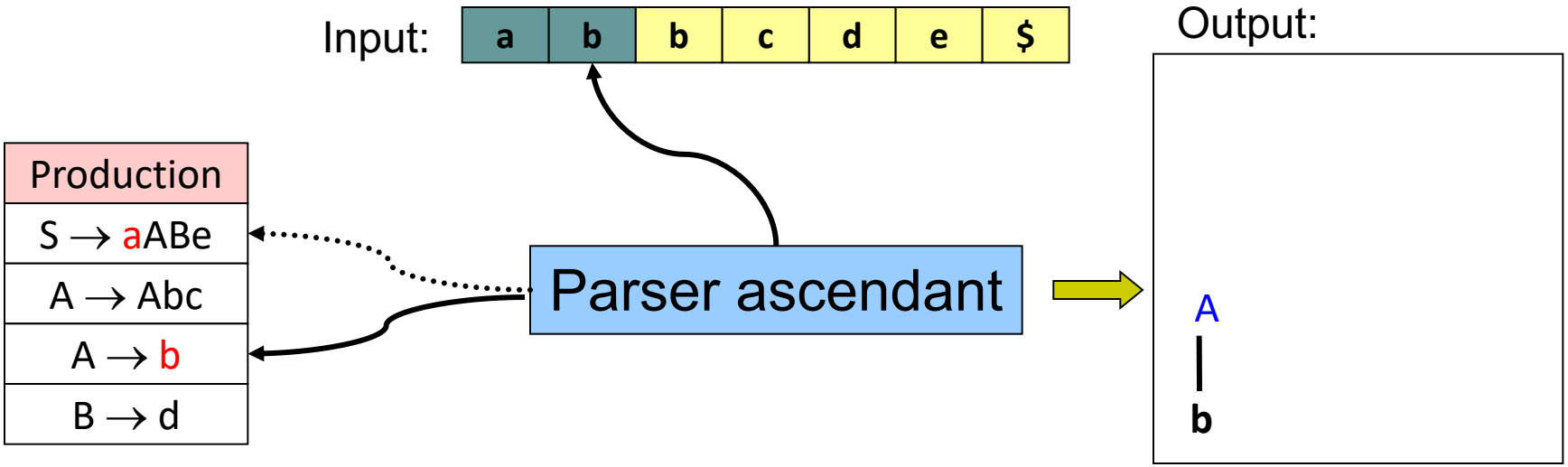
(Aho,Sethi,Ullman, pp. 195)

Exemple de parser ascendant



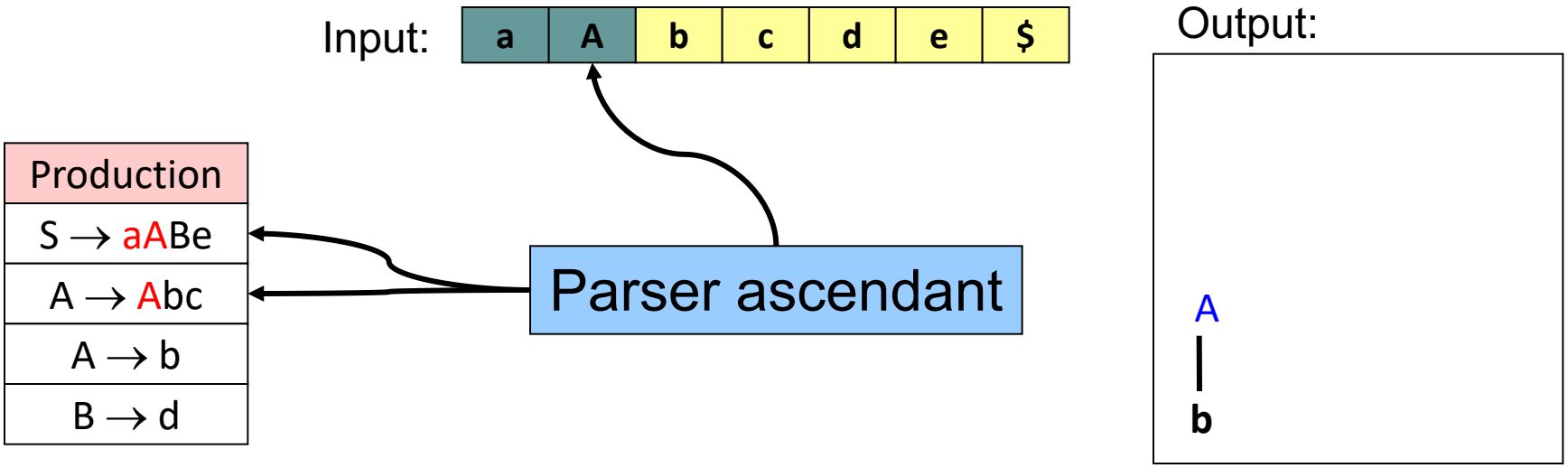
(Aho,Sethi,Ullman, pp. 195)

Exemple de parser ascendant



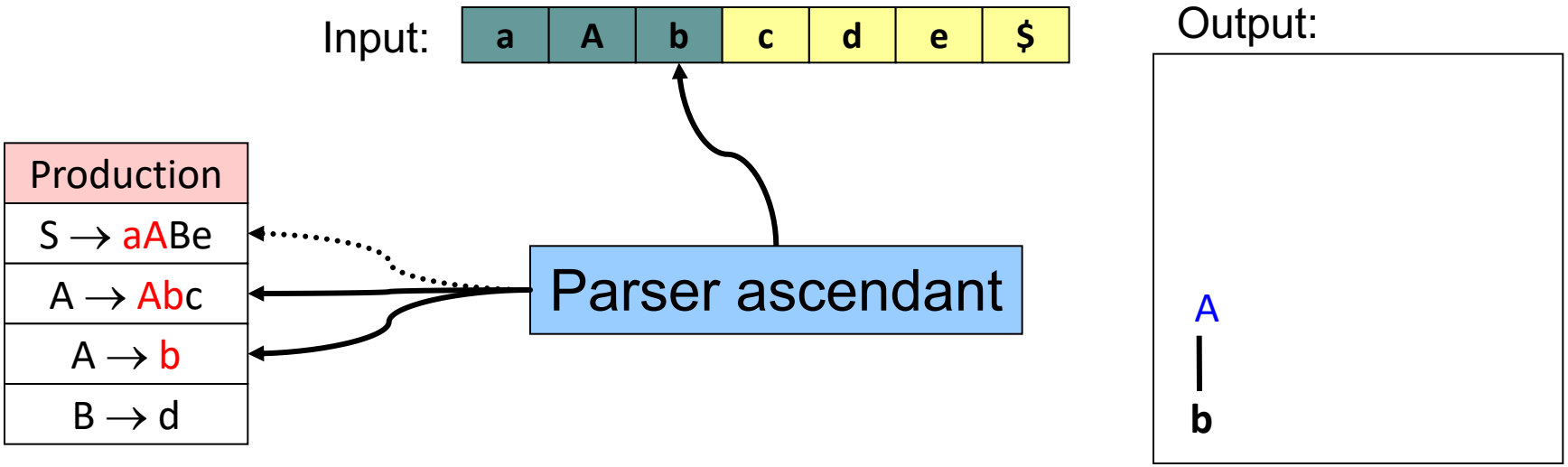
(Aho,Sethi,Ullman, pp. 195)

Exemple de parser ascendant



(Aho,Sethi,Ullman, pp. 195)

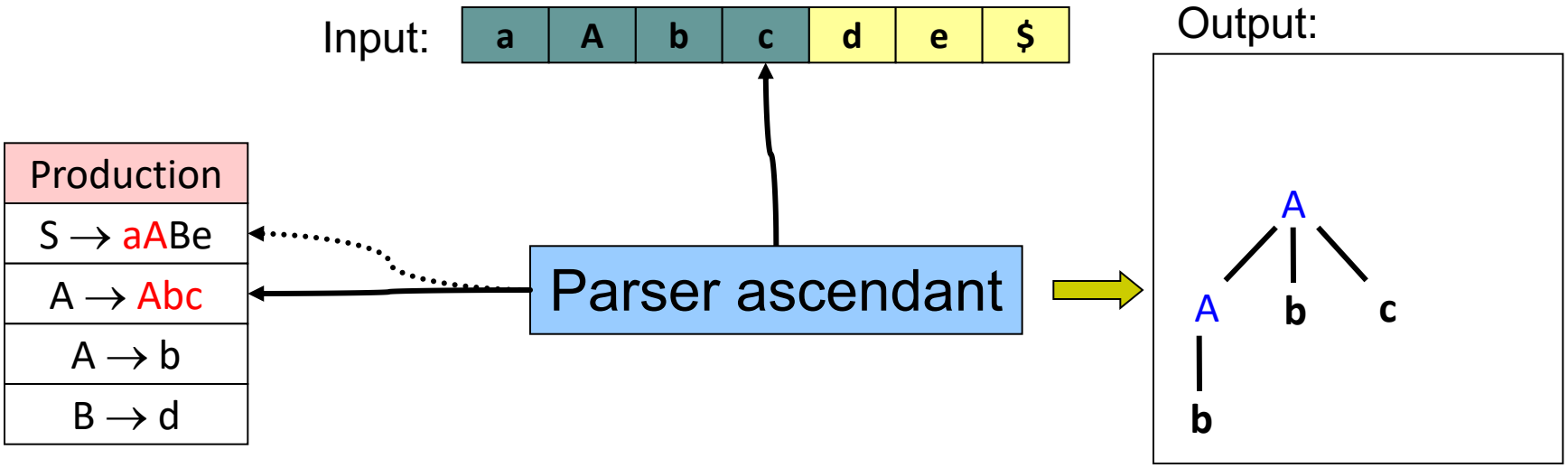
Exemple de parser ascendant



Nous ne pouvons pas réduire, le parser s'arrêterait.

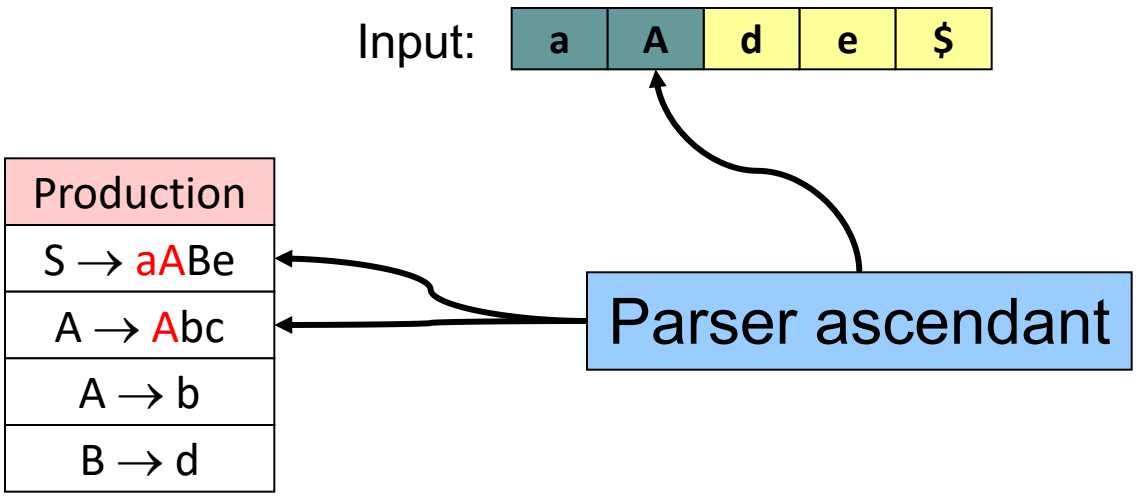
(Aho,Sethi,Ullman, pp. 195)

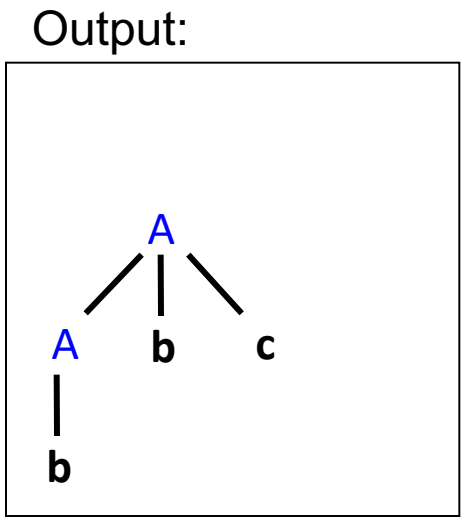
Exemple de parser ascendant



(Aho,Sethi,Ullman, pp. 195)

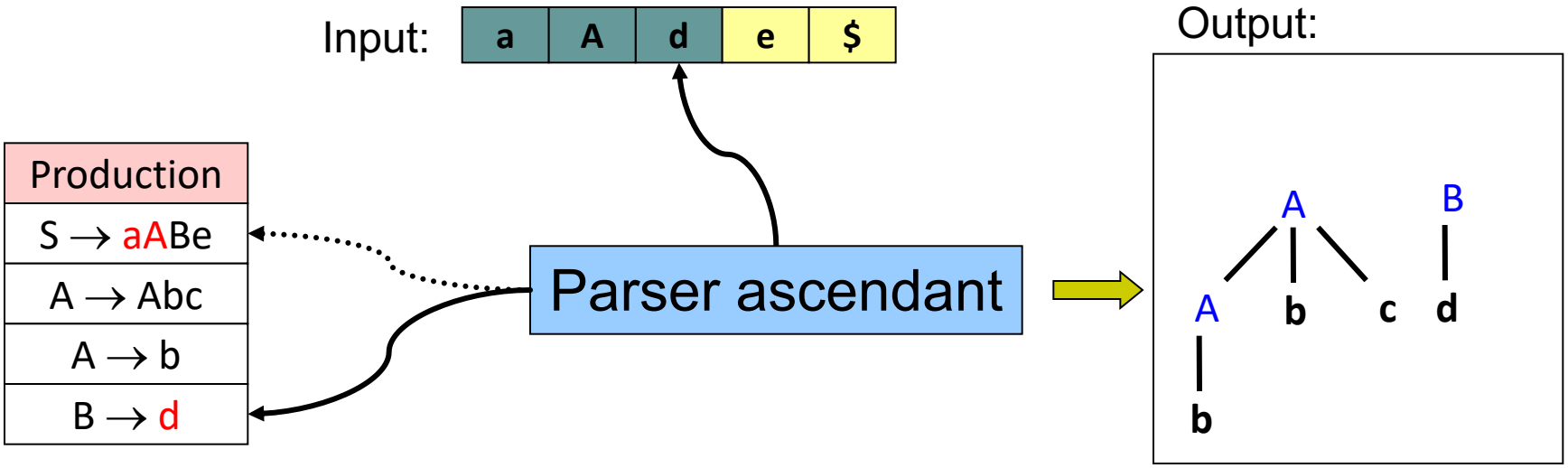
Exemple de parser ascendant





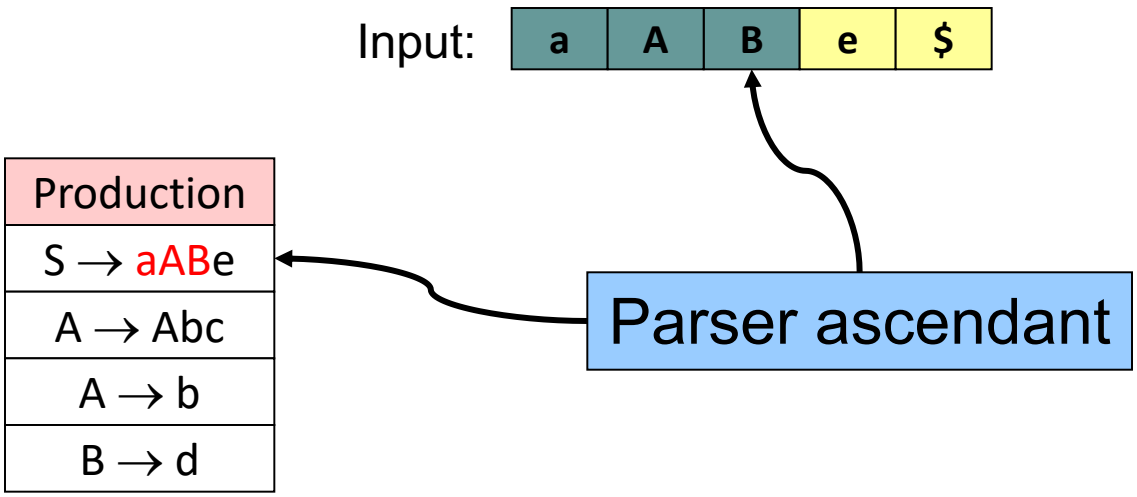
(Aho,Sethi,Ullman, pp. 195)

Exemple de parser ascendant

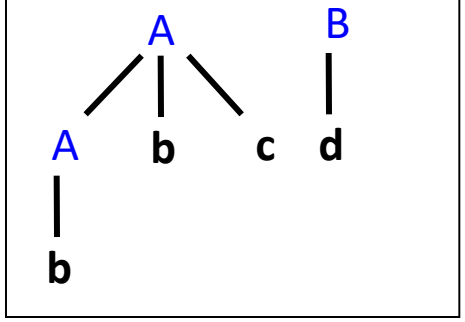


(Aho,Sethi,Ullman, pp. 195)

Exemple de parser ascendant

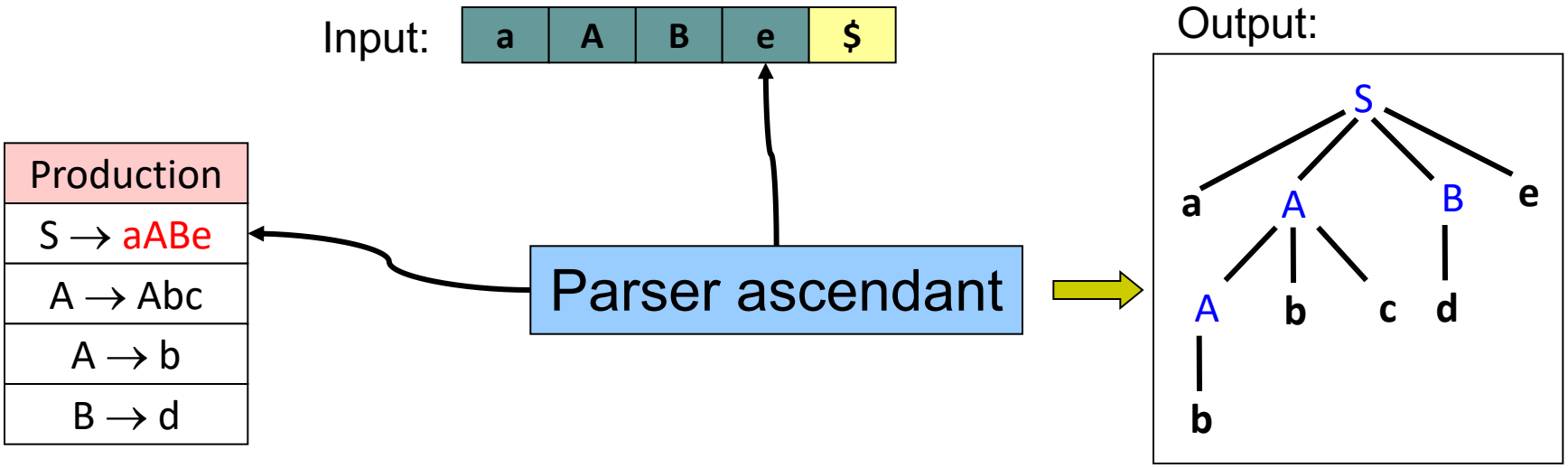


Output:



(Aho,Sethi,Ullman, pp. 195)

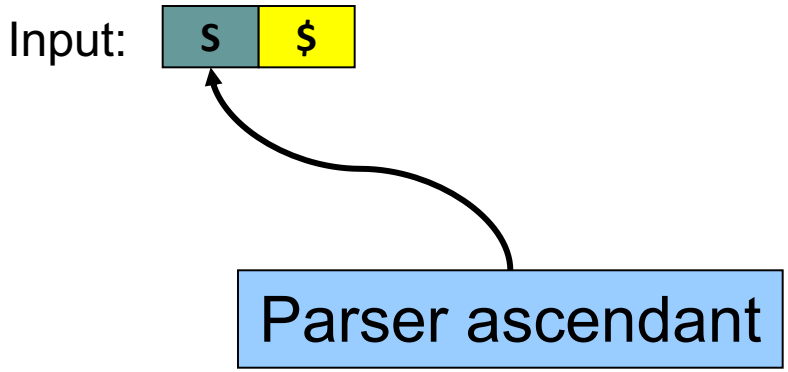
Exemple de parser ascendant



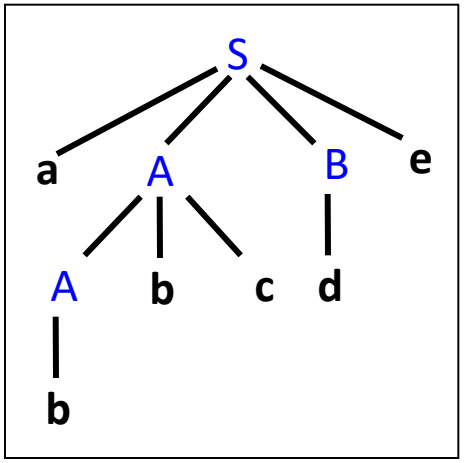
(Aho,Sethi,Ullman, pp. 195)

Exemple de parser ascendant

Production
S → aABe
A → Abc
A → b
B → d



Output:



Parser LR parce que l'entrée este lu « De gauche à droite », et construit « dérivation plus à droite » dans l'ordre inverse.

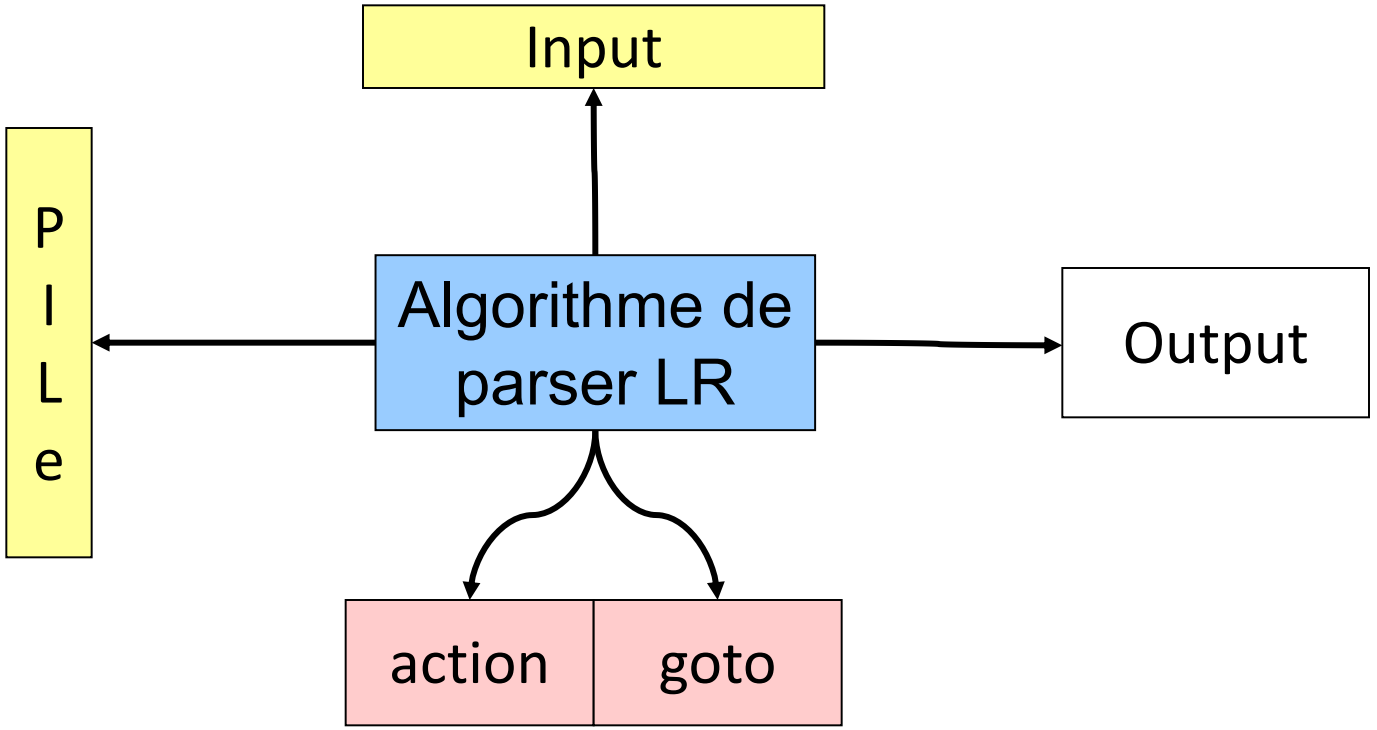
(Aho,Sethi,Ullman, pp. 195)

Exemple de parser ascendant

ALF
language crunch

- Analyse les productions pour le match avec le texte d'entrée
- Backtracking
- Inefficace
- Pouvons-nous faire mieux?

Exemple de parser LR



(Aho,Sethi,Ullman, pp. 217)

Exemple de parser LR

GRAMMAIRE:

(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Il peut être parser avec les tableaux ‘action’ et ‘goto’

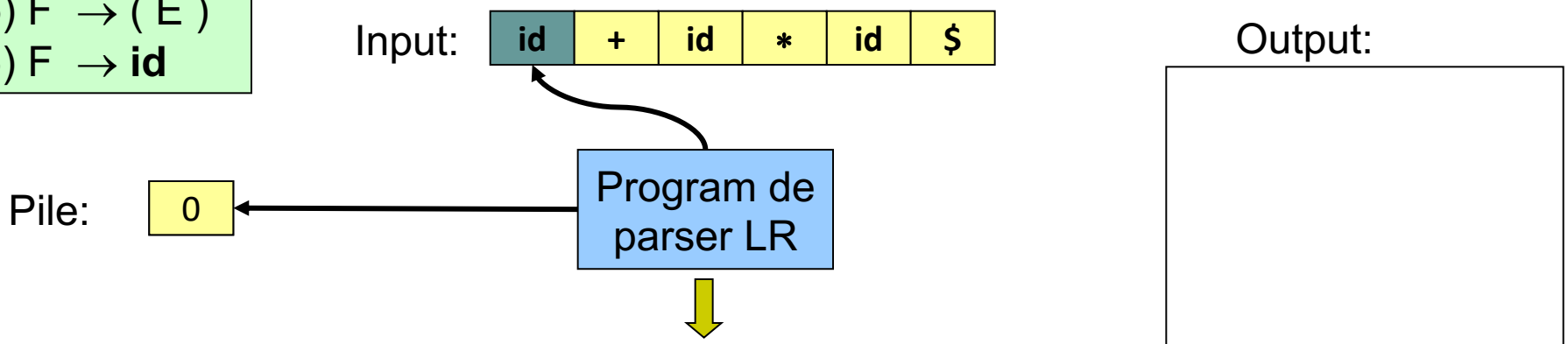
State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

(Aho,Sethi,Ullman, pp. 219)

GRAMMAIRE:

(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR



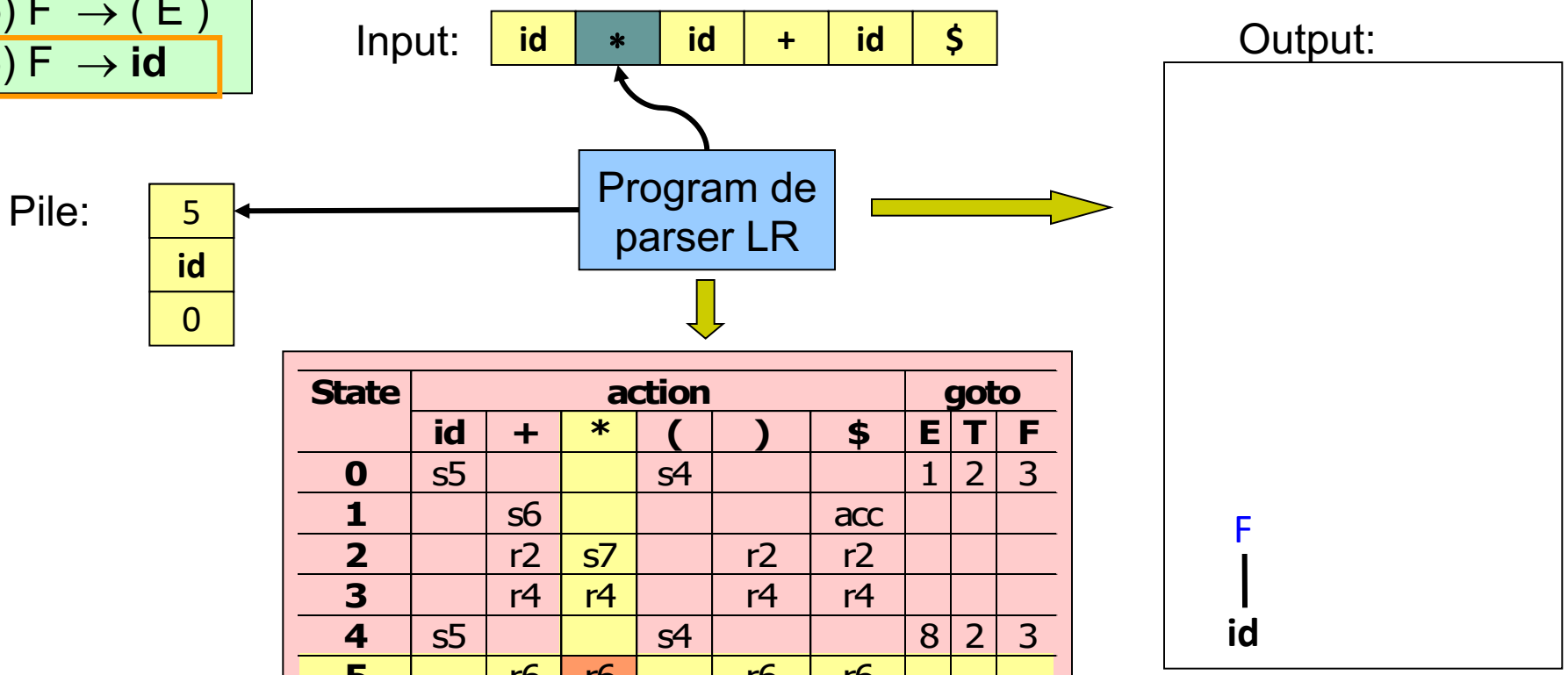
State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR



State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

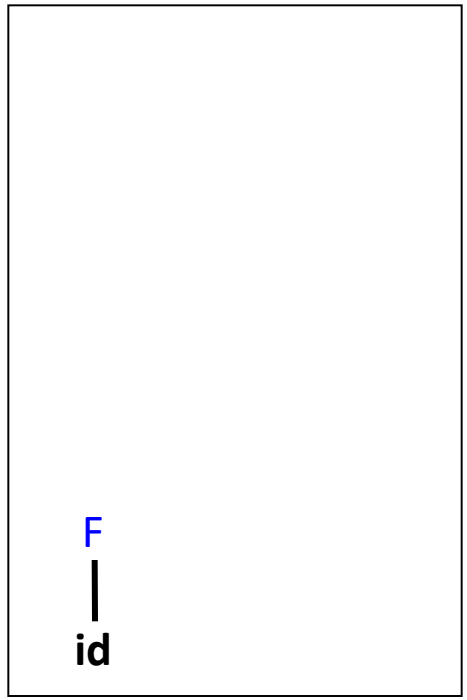
Exemple de parser LR

Input: | **id** | * | **id** | + | **id** | **$** |

Output:

Pile: 0

Program de parser LR



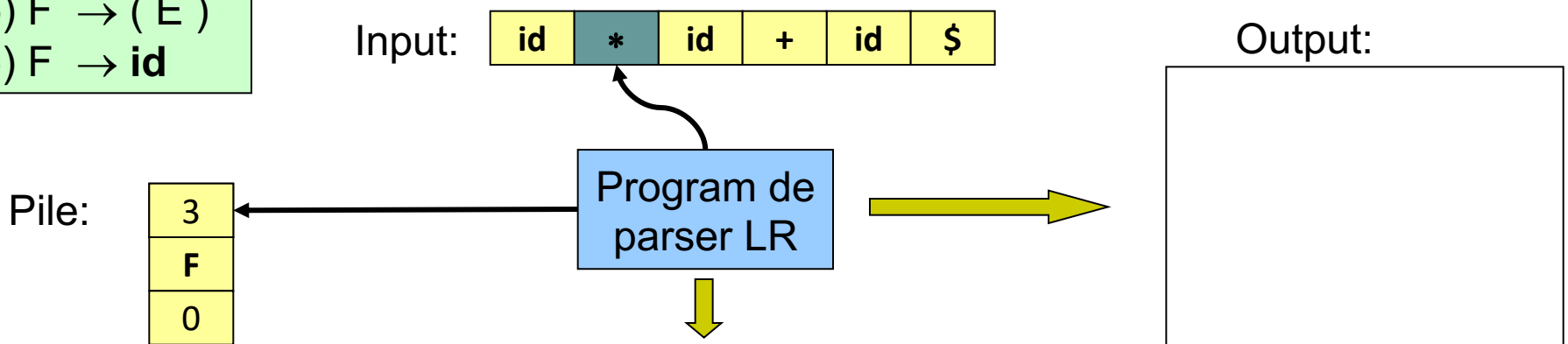
State	action						goto		
	id	**+**	*	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR



State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR

ALF language crunch

Input: | id | * | id | + | id | $ |

Pile: 0

Program de parser LR

Output:

T
|
F
|
id

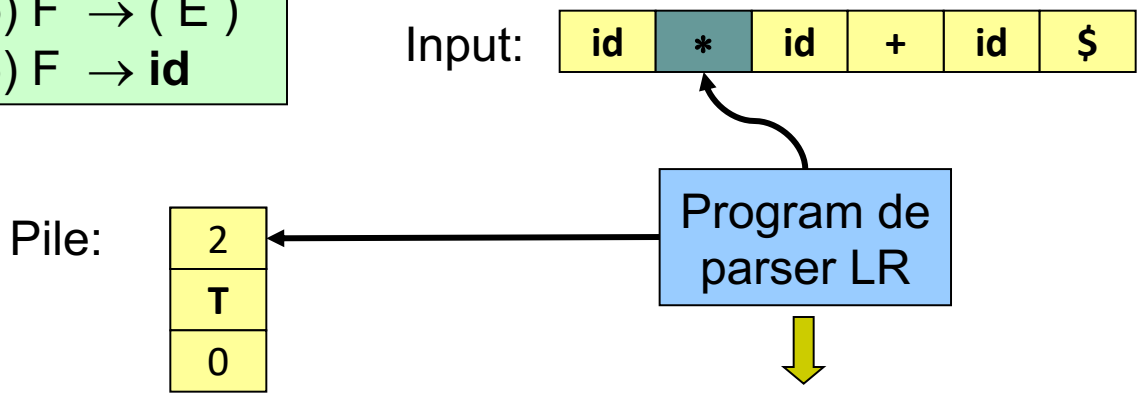
State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

(Aho,Sethi,Ullman, pp. 220)

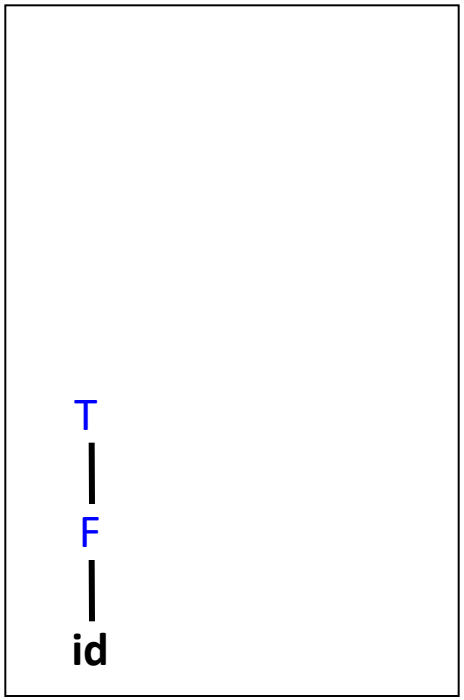
GRAMMAIRE:

(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR



Output:



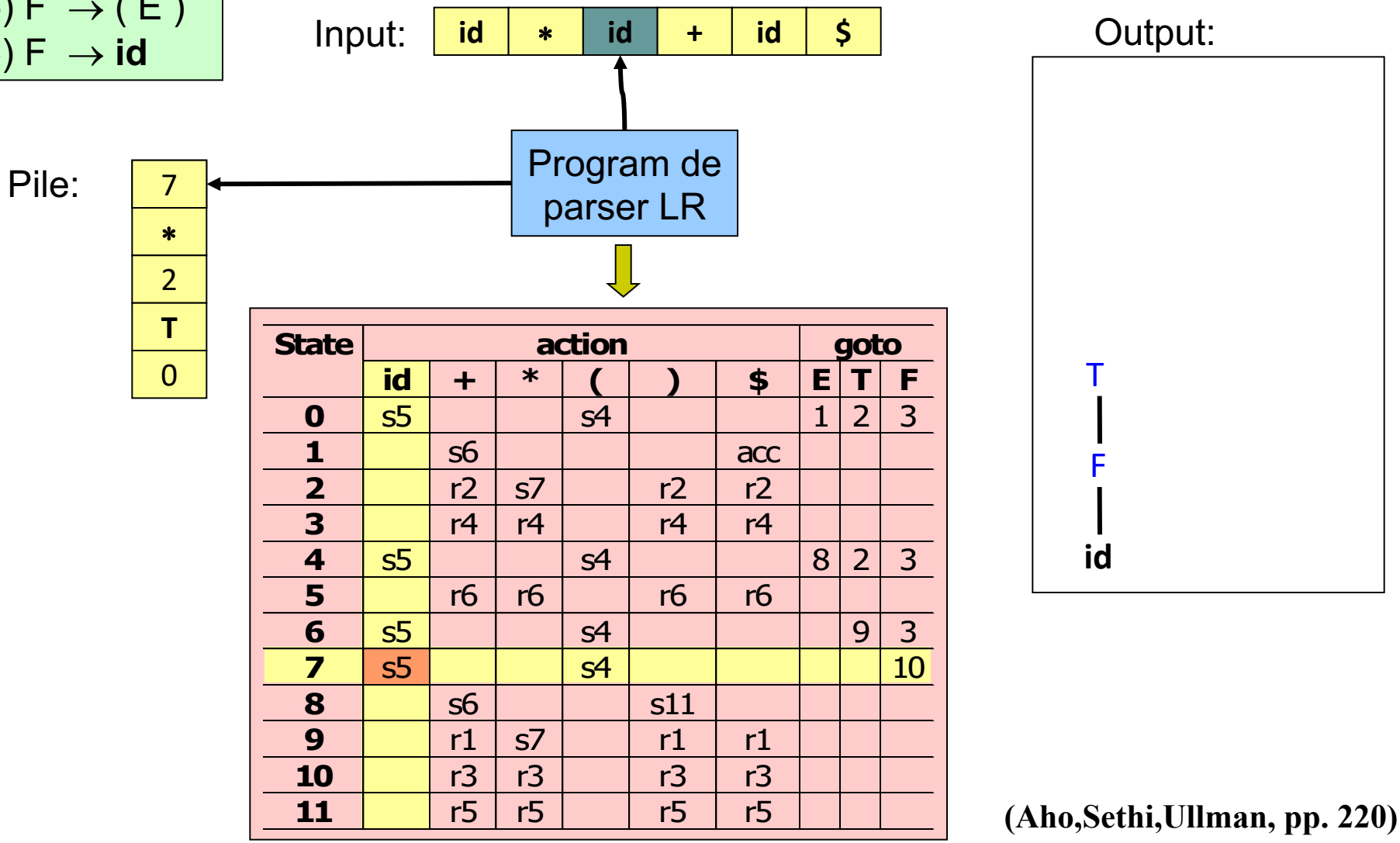
State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR



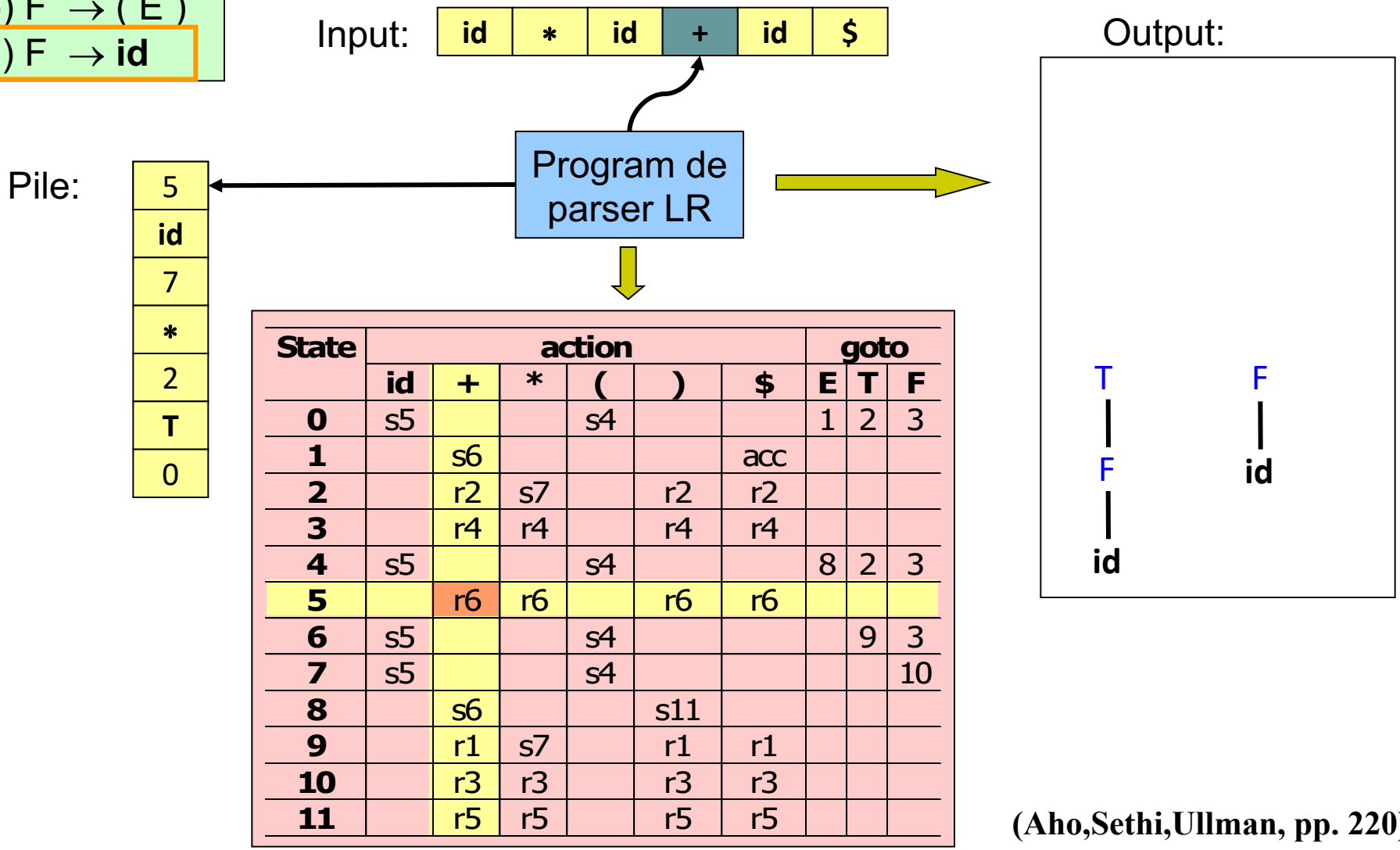
State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR



State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR

Input: | **id** | * | **id** | + | **id** | **$** |

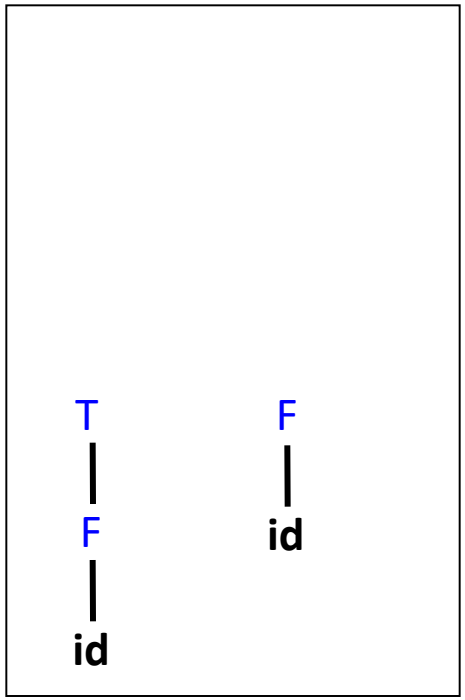
Pile:

7
*
2
T
0

Program de parser LR

State	action						goto		
	id	**+**	*	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Output:



(Aho,Sethi,Ullman, pp. 220)

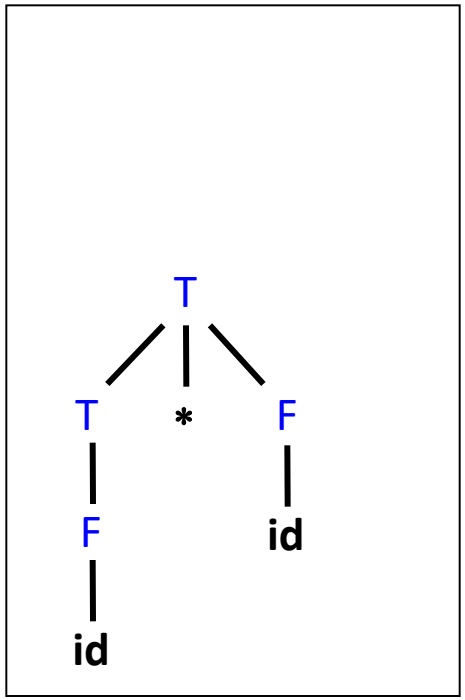
GRAMMAIRE:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR

Input: | id | * | id | + | id | $ |

Output:



Pile:

10
F
7
*
2
T
0

Program de parser LR

State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

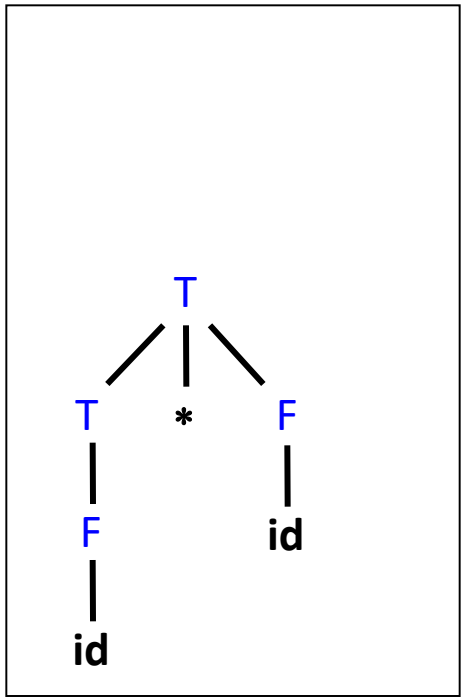
Exemple de parser LR

Input: | id | * | id | + | id | $ |

Pile: 0

Program de parser LR

Output:



State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR

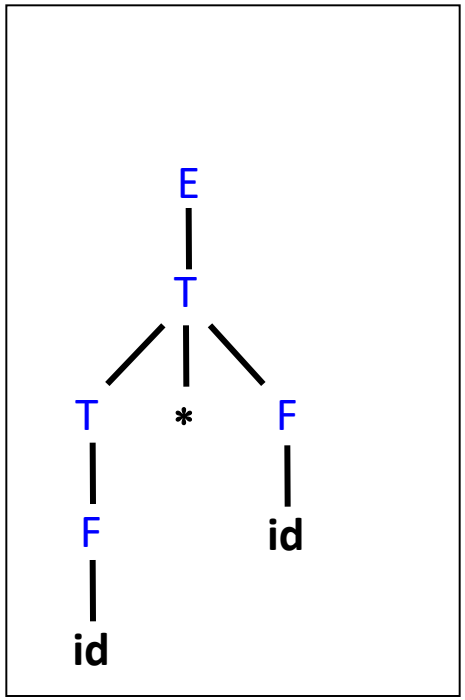
ALF language crunch

Input: | id | * | id | + | id | $ |

Pile: | 2 | T | 0 |

Program de parser LR

Output:



State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR

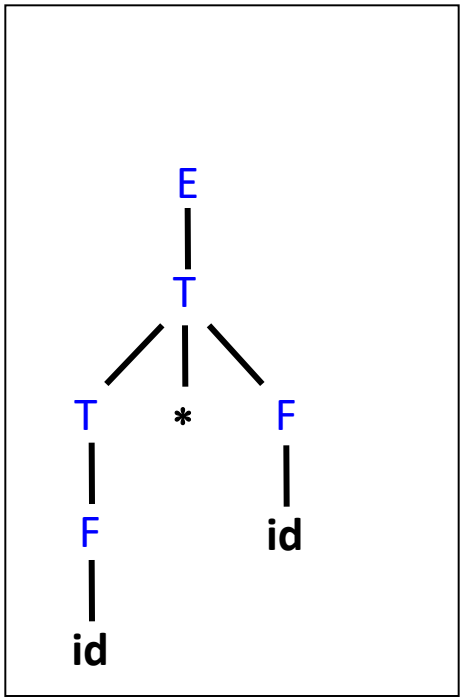
Input: | **id** | * | **id** | + | **id** | **$** |

Pile: 0

Program de parser LR

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Output:



(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR

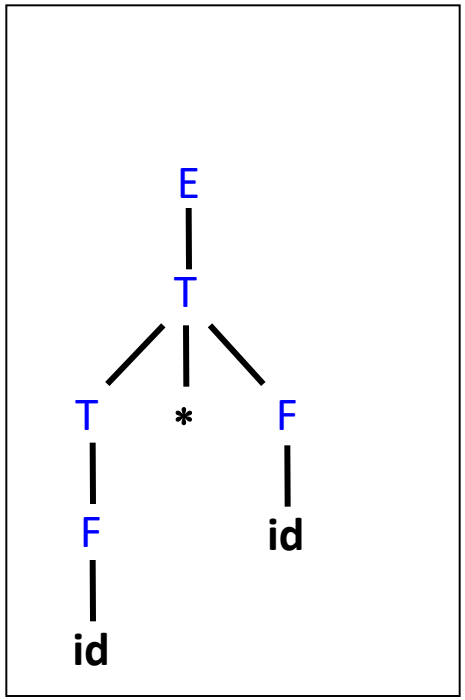
Input: | id | * | id | + | id | $ |

Pile: | 1 | E | 0 |

Program de parser LR

State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Output:



(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR

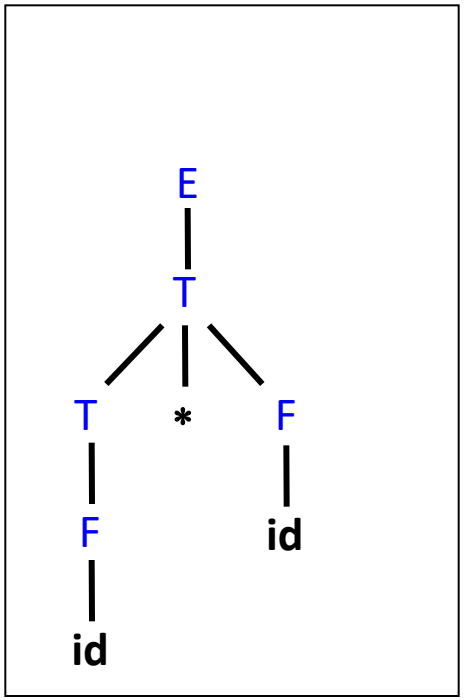
Input: | id | * | id | + | id | $ |

Pile: | 6 | + | 1 | E | 0 |

Program de parser LR

State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Output:



(Aho,Sethi,Ullman, pp. 220)

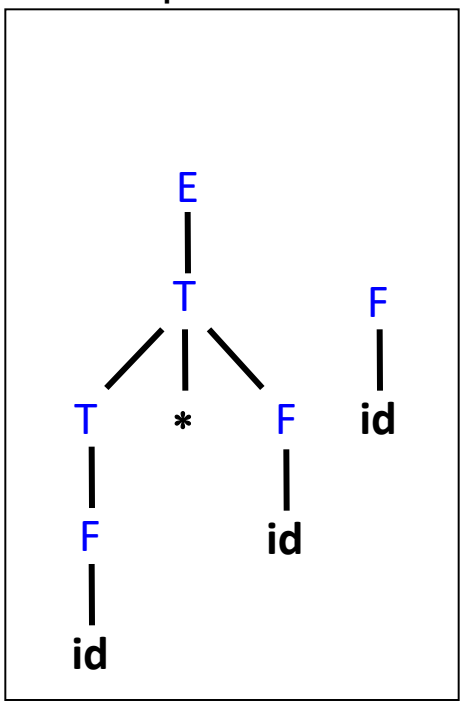
GRAMMAIRE:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR

Input: | **id** | * | **id** | + | **id** | **$** |

Output:



Pile:

5
id
6
+
1
E
0

Program de parser LR

State	action						goto		
	id	+	*	(	)	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR

Input: | **id** | * | **id** | + | **id** | **$** |

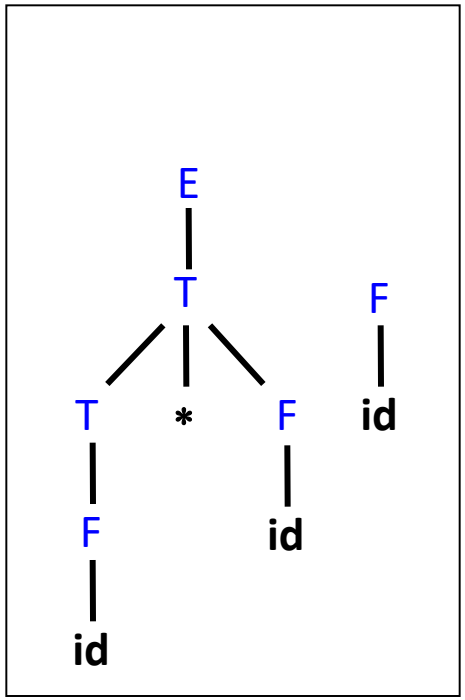
Pile:

6
+
1
E
0

Program de parser LR

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Output:



(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR

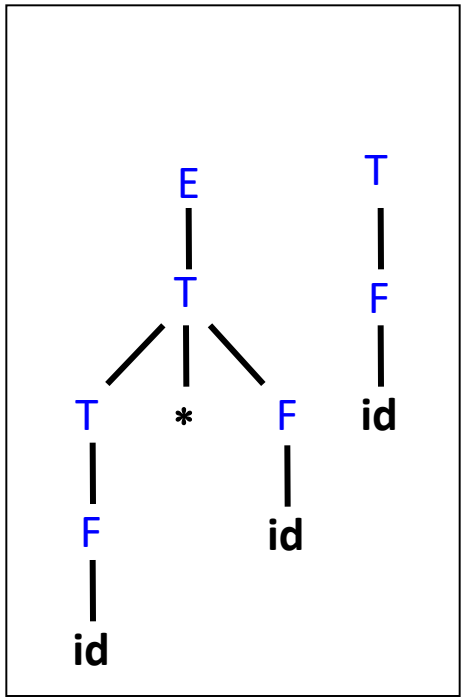
Input: | id | * | id | + | id | $ |

Pile: | 3 | F | 6 | + | 1 | E | 0 |

Program de parser LR

State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Output:



(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR

ALF language crunch

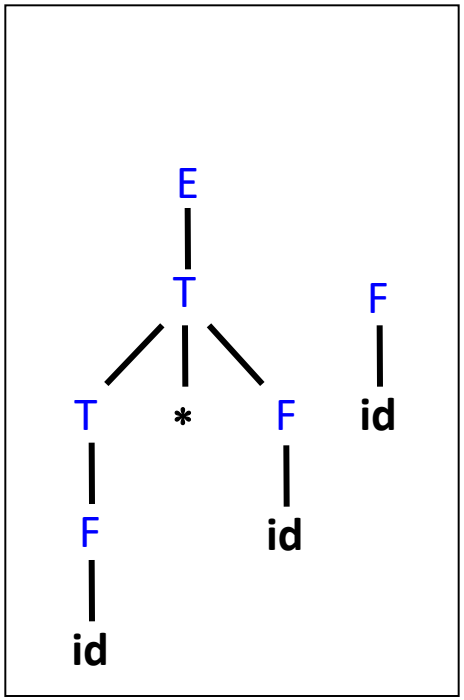
Input: | **id** | * | **id** | + | **id** | **$** |

Pile: | 6 | + | 1 | E | 0 |

Program de parser LR

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Output:



(Aho,Sethi,Ullman, pp. 220)

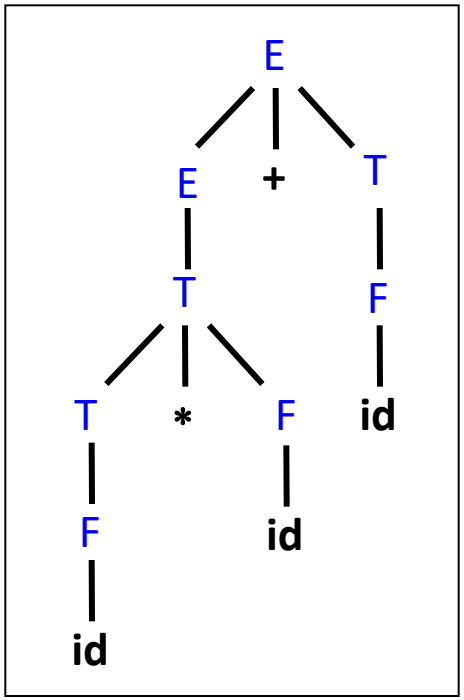
GRAMMAIRE:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR

Input: | **id** | * | **id** | + | **id** | **$** |

Output:



Pile:

9
T
6
+
1
E
0

Program de parser LR

State	**action**						**goto**		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR

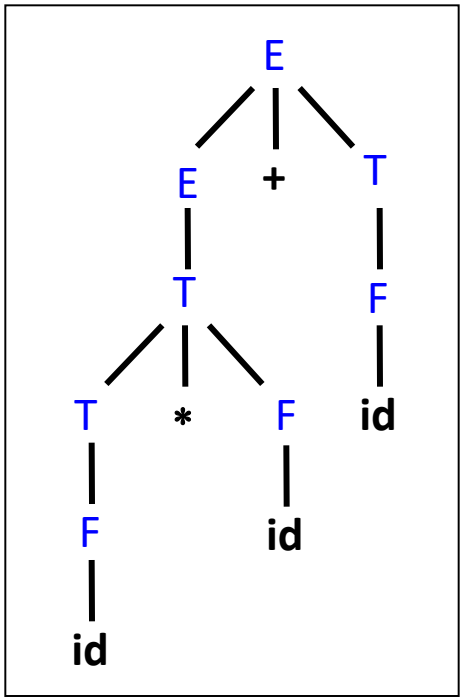
Input: | **id** | * | **id** | + | **id** | **$** |

Pile: 0

Program de parser LR

State	action						goto		
	id	+	*	(	)	$	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Output:



(Aho,Sethi,Ullman, pp. 220)

GRAMMAIRE:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Exemple de parser LR

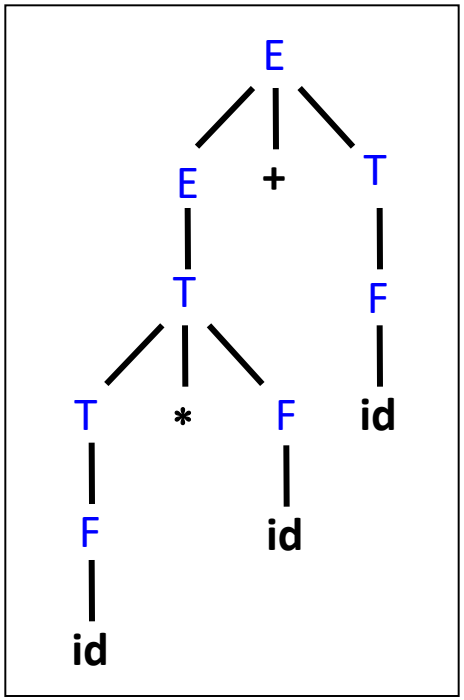
Input: | id | * | id | + | id | $ |

Pile: | 1 | E | 0 |

Program de parser LR

State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Output:



(Aho,Sethi,Ullman, pp. 220)

Construction de tableaux de parser

- Tous les analyseurs utilisent le même algorithme
- La différence est dans l'action et goto
- Simple LR (SLR) se poursuit moins la grammaire, mais il est le plus facile à mettre en œuvre
- Canonique LR allez sur la plupart de grammaire, mais il est plus difficile à mettre en œuvre.
- LR Lookahead (LALR) va sur la plupart des constructions syntaxiques utilisées dans les langages de programmation, mais produire des tableaux beaucoup plus petites que Canonique LR.

Shift-Reduce

ALF
language crunch

- Actions:
  - Une séquence de shift et reduce
- Pile
  - Règles et jetons
  - Etat
- Etat suivante
  - Pile et input

Actions Shift-Reduce

- Shift: push lookahead

pile	input	action
(	1+2+(3+4))+5	shift 1
(1	+2+(3+4))+5	

- Reduce: pop β, push X

pile	input	action
(S+E	+(3+4))+5	reduce S → S+ E
(S	+(3+4))+5	

Analyse Shift-Reduce

ALF
language crunch

$S \rightarrow S + E \mid E$
$E \rightarrow num \mid (S)$

derivation	pile	input	action
(1+2+(3+4))+5		(1+2+(3+4))+5	shift
(1+2+(3+4))+5	(	1+2+(3+4))+5	shift
(1+2+(3+4))+5	(1	+2+(3+4))+5	reduce E→num
(E+2+(3+4))+5	(E	+2+(3+4))+5	reduce S→E
(S+2+(3+4))+5	(S	+2+(3+4))+5	shift
(S+2+(3+4))+5	(S+	2+(3+4))+5	shift
(S+2+(3+4))+5	(S+2	+(3+4))+5	reduce E→num
(S+E+(3+4))+5	(S+E	+(3+4))+5	reduce S→S+E
(S+(3+4))+5	(S	+(3+4))+5	shift
(S+(3+4))+5	(S+	(3+4))+5	shift
(S+(3+4))+5	(S+(	3+4))+5	shift
(S+(3+4))+5	(S+(3	+4))+5	reduce E→ num
…			

Sélection de l'action

ALF
language crunch

- Comment sélectionnons-nous l'action?
  - shift ou reduce?
- Quelle règle s'appliquons-nous?
  - Éviter le blocage
  - La pile peut être réduite de plusieurs façons

Sélection de l'action

- Etat current:
  - Pile β
  - Symbole look-ahead b
  - existe la production X → γ, et la pile est de forme $\beta = \alpha\gamma$
- Le parser devrait:
  - Shift b sur la pile, βb ?
  - Reduce avec la production X → γ, si la pile est $\beta = \alpha\gamma$, et transformer la pile en αX ?
- Décision fondée su b et le préfix α
  - α est différent pour différentes productions, parce que le côté droit de productions (les γ) peuvent avoir des longueurs différentes

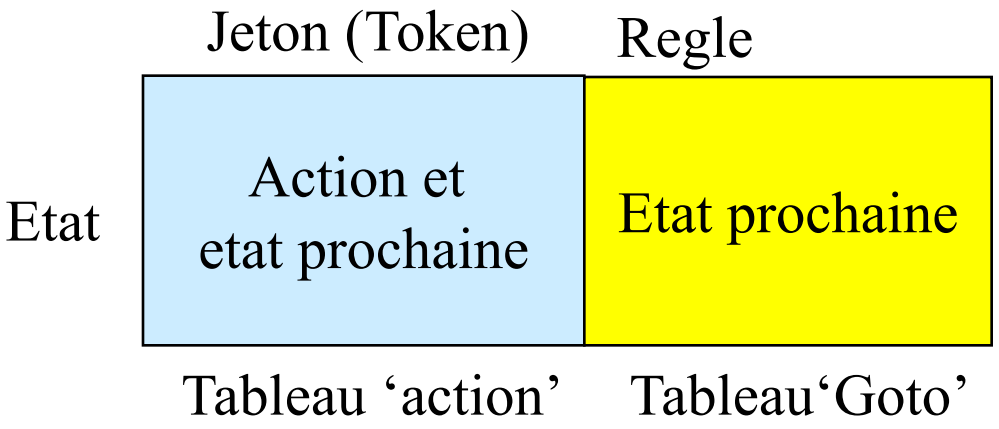
Algorithme de parser LR

GRAMMAIRE:

(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Tableau de parser LR



- Algorithme: état curent S si token C au input
  - Si Action[S,C] = s(S') shift et aller sur S':
    - push(C), push(S')
  - Si Action[S,C] = X→ α reduce:
    - pop(2*|α|), S'= top(), push(X), push(Goto[S',X])

Element LR(0)

ALF
language crunch

Règle

E → num | (S)

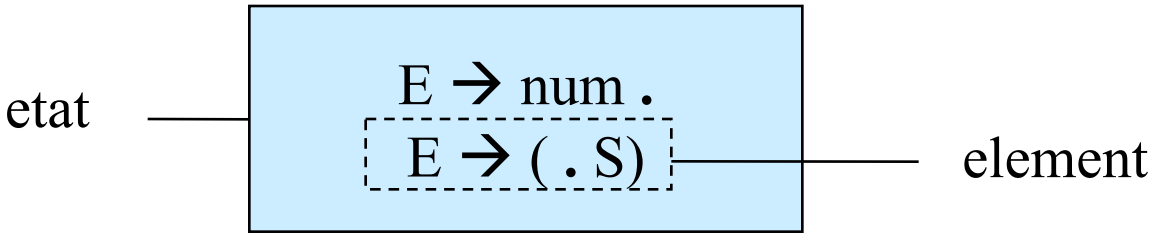
Deux elements LR(0)

E → num .

E → (. S)

Exemple de etat LR(0)

- Un element LR(0) est une production avec un “.” dans la partie droite

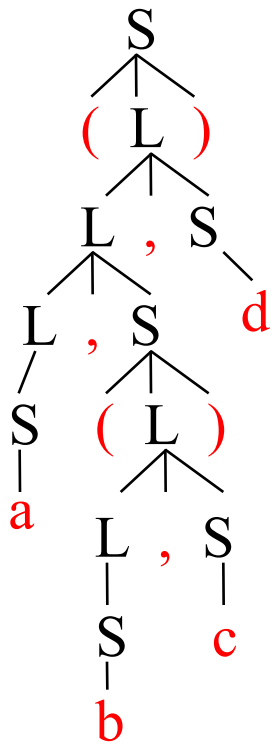


- Les éléments avant “.” sont sur la pile
- Les elements apres “.” nous dire ce que nous pouvions lire suivante

Grammaire LR(0)

- Grammaire de liste
  - S → (L) | id
  - L → S | L,S
- Exemple:
  - (a,b,c)
  - ((a,b), (c,d), (e,f))
  - (a, (b,c,d), ((f,g)))

Arbre de derivation pour
(a, (b,c), d)



Etat de start et Closure

ALF
language crunch

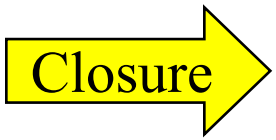
- Etat de start
  - Production S' → S
  - Etat start: Closure (S' → . S)
- Closure pour l'ensable des éléments LR(0):
  - Closure (S) = S
  - Pour toutes les éléments en S:
    - X → α . Y β
    - Toutes les éléments LR(0) de forme Y → . γ, pour toutes les production Y → γ de la grammaire

Exemple

S → (L) | id
L → S | L,S

Elément LR(0) “start”

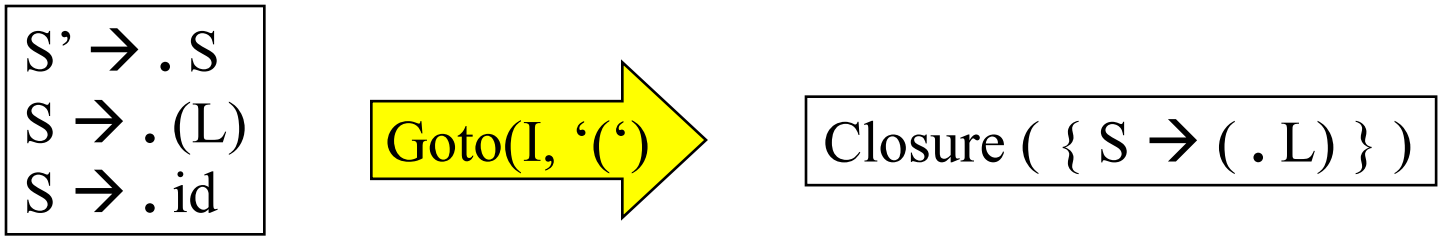
S’ → . S



S’ → . S
S → . (L)
S → . id

Goto

- Transitions entre les états (ensembles des éléments LR(0))



Questions

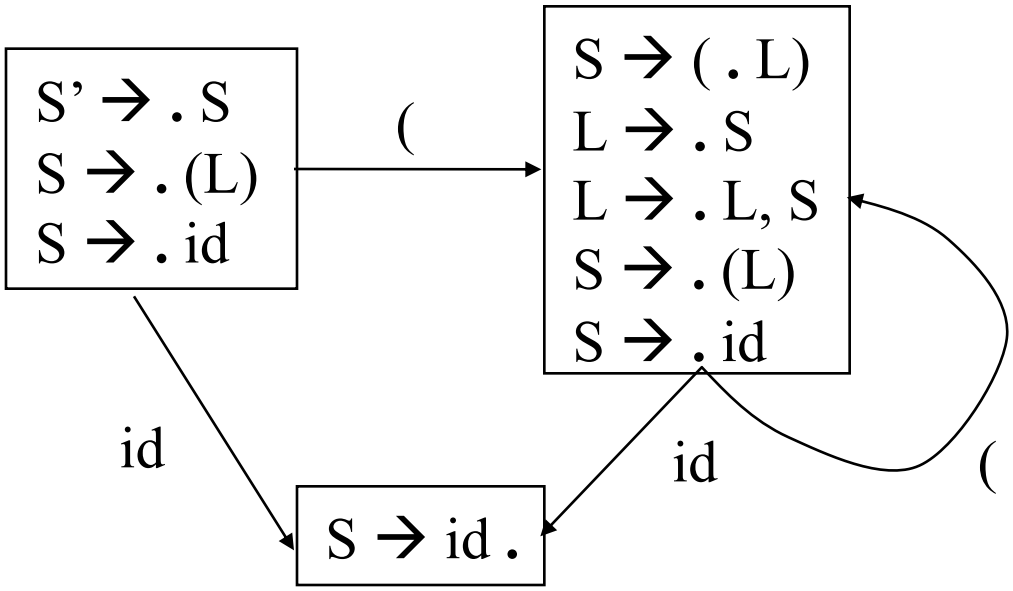
ALF
language crunch

E' → E
E → E + T | T
T → T * F | F
F → (E) | id

I = { [E' → . E]}, Closure (I) = ??

I = { [E' → E .], [E → E . + T] }, Goto(I,+) = ??

Goto: tokens

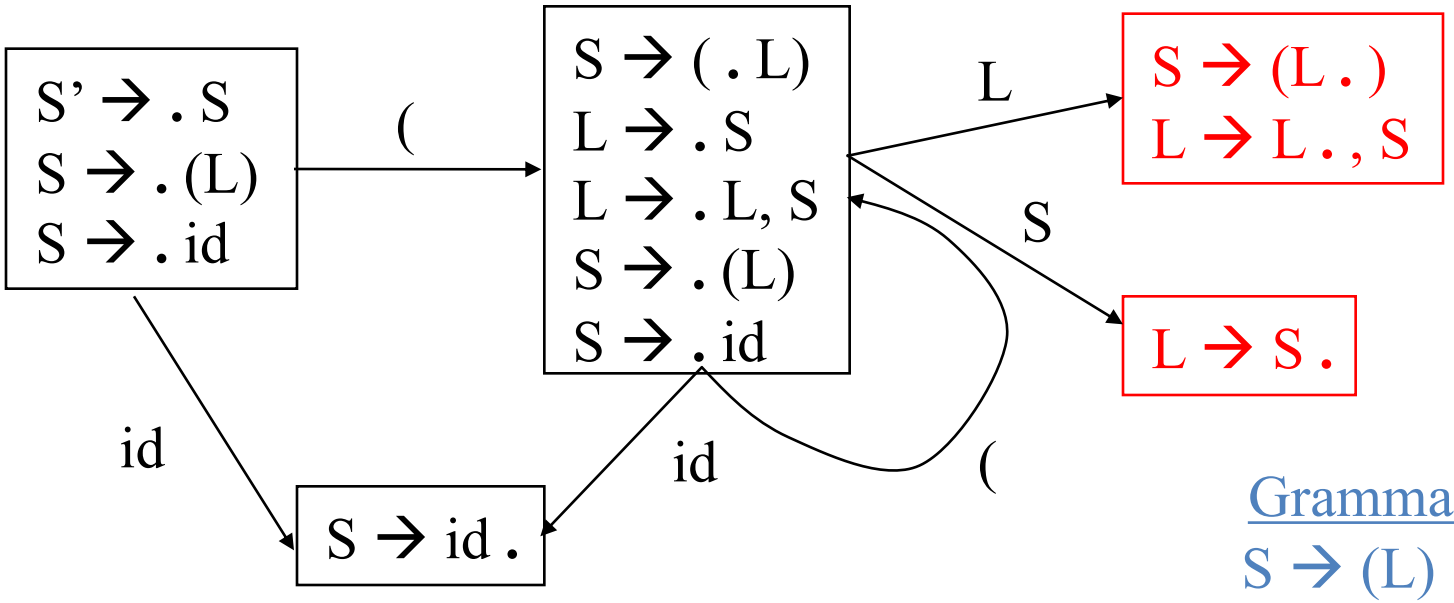


Grammaire

S → (L) | id

L → S | L,S

Goto: règles

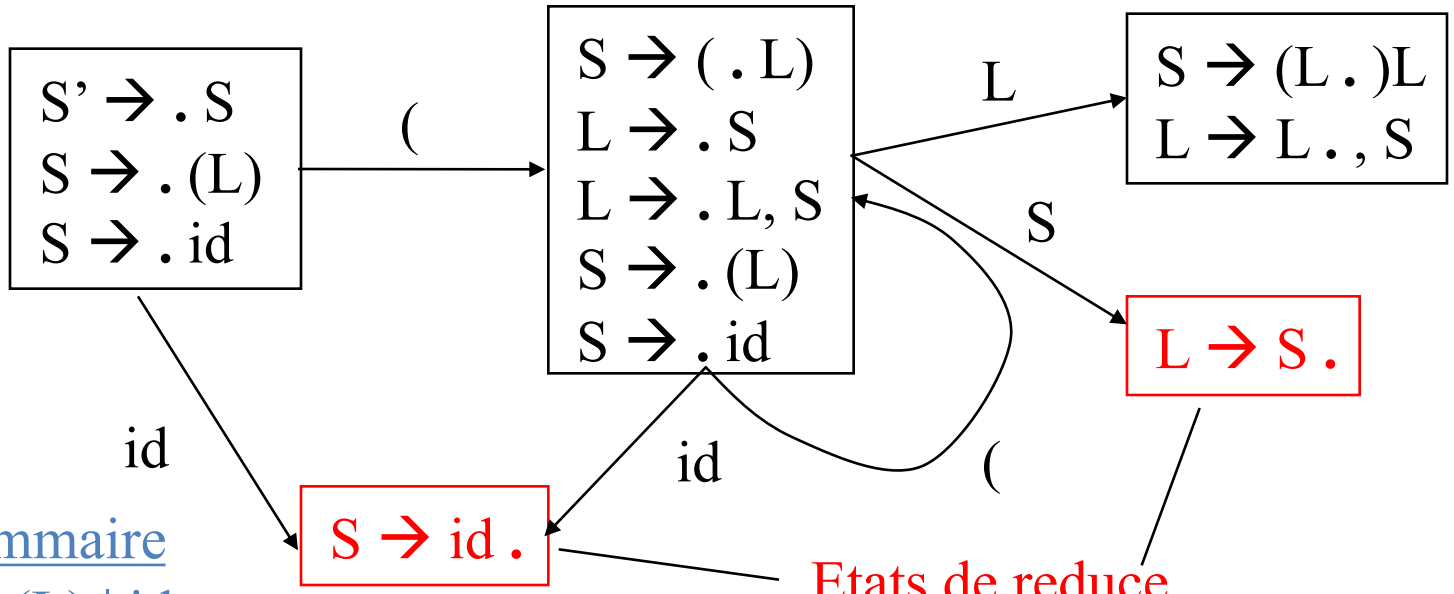


Grammaire

S → (L) | id

L → S | L,S

Reduce



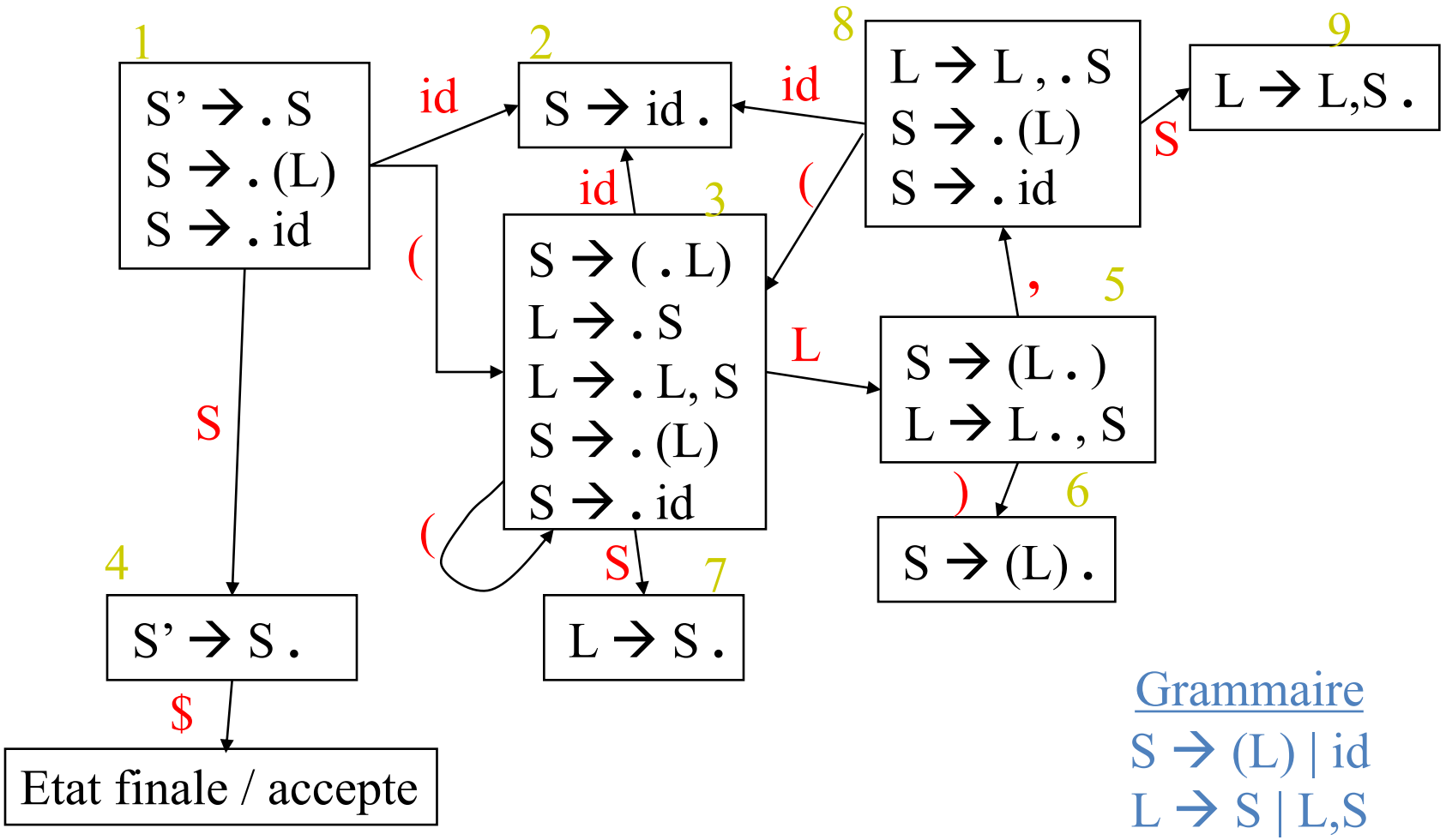
Grammaire

S → (L) | id

L → S | L,S

Etats de reduce
(le point e a la fin!)

Automate LR



Grammaire

S → (L) | id

L → S | L,S

Exemple de parser ((a),b)

dérivation	pile	input	action
((a),b) ←	1	((a),b)	shift, goto 3
((a),b) ←	1(3	(a),b)	shift, goto 3
((a),b) ←	1(3(3	a),b)	shift, goto 2
((a),b) ←	1(3(3a2	),b)	reduce S→id
((S),b) ←	1(3(3(S7	),b)	reduce L→S
((L),b) ←	1(3(3(L5	),b)	shift, goto 6
((L),b) ←	1(3(3L5)6	,b)	reduce S→(L)
(S,b) ←	1(3S7	,b)	reduce L→S
(L,b) ←	1(3L5	,b)	shift, goto 8
(L,b) ←	1(3L5,8	b)	shift, goto 9
(L,b) ←	1(3L5,8b2	)	reduce S→id
(L,S) ←	1(3L8,S9	)	reduce L→L,S
(L) ←	1(3L5	)	shift, goto 6
(L) ←	1(3L5)6		reduce S→(L)
S ←	1S4	$	accepte

S → (L) | id
L → S | L,S

Tableau de parser LR(0)

ALF
language crunch

- Etats = états de Automate LR
- S → S’ avec jeton C:
  - Action[S,C] += Shift(S’)
- S → S’ avec la règle N:
  - Goto[S,N] += S’
- Si S est état de réduction X → β.:
  - Action[S,*] += Reduce(X → β)

Exemple de tableau

ALF language crunch

Etat	Jetons					Regles	
	(	)	id	,	$	S	L	
1		s3		s2			g4	
2		S→id	S→id	S→id	S→id	S→id		
3		s3		s2			g7	g5
4						accept		
5			s6		s8			
6		S→(L)	S→(L)	S→(L)	S→(L)	S→(L)		
7		L→S	L→S	L→S	L→S	L→S		
8		s3		s2			g9	
9		L→L,S	L→L,S	L→L,S	L→L,S	L→L,S		

shift reduce

Limites LR(0)

- Une action pou

OK

$L \rightarrow L , S .$

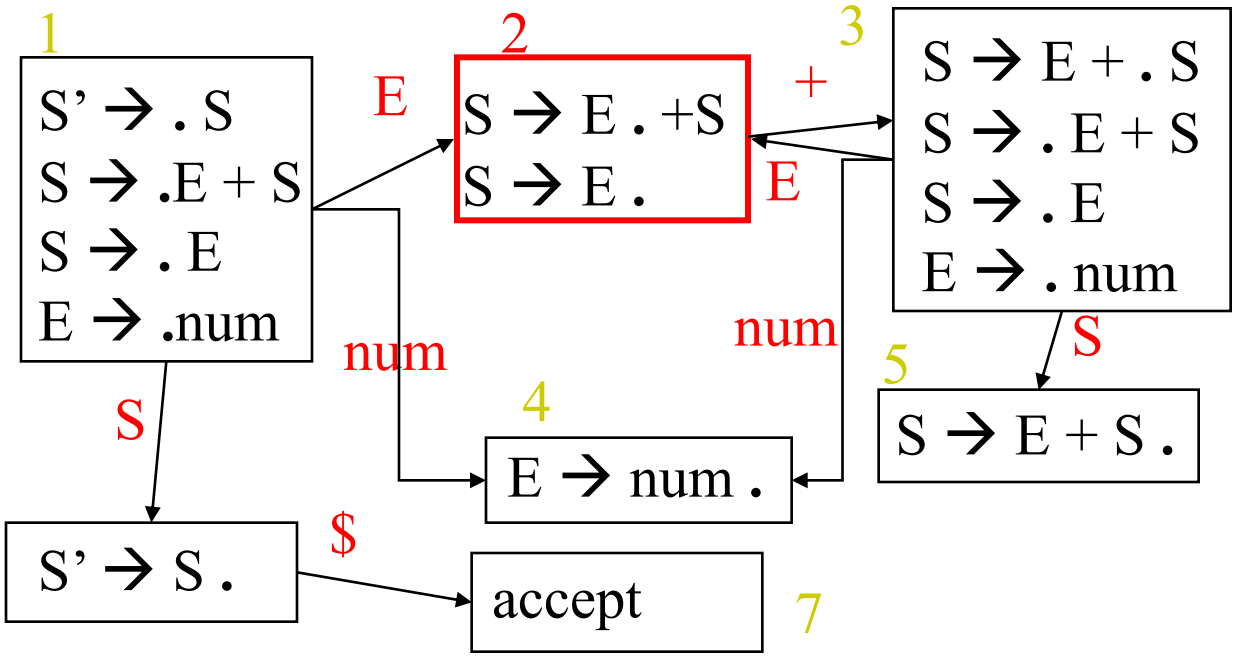
shift/reduce

$L \rightarrow L , S .$
$S \rightarrow S . , L$

reduce/reduce

$L \rightarrow S , L .$
$L \rightarrow S .$

Exemple de shift/reduce



Grammaire

S → E + S | E

E → num

Shift ou reduce etat 2?

	num	+	$	E	S
1	s4			g2	g6
2	S→E	s3/S→E	S→E		

Parser SLR

- SLR ="Simple LR"= extension simple LR(0)
  - Pour X → β, le prochain jeton C
  - reduce seulement si C est de FOLLOW(X)
- Résoudre partial les shift/reduce
  - Identique avec LR(0) sauf 'reduction'
  - Réduction X → β seulement les symboles de FOLLOW(X)

Exemple: FOLLOW(S) = {$}

	num	+	$	E	S
1	s4			g2	g6
2		s3	S→E		

Exemple de parser SLR

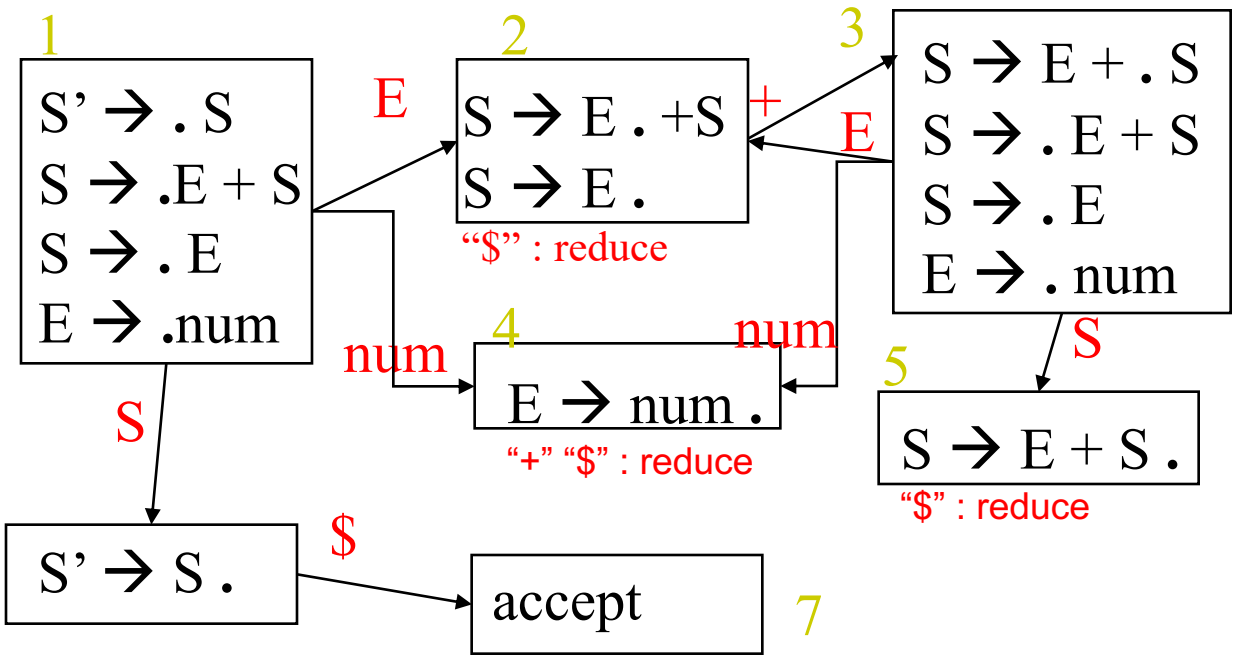
	num	+	$	E	S
1	s4			g2	g6
2		s3	S→E		
3	s4			g2	g5
4		E→num	E→num		
5			S→E+S		
6			s7		
7			accept		

Grammaire

S → E + S | E

E → num

Automate SLR



Grammaire

S → E + S | E

E → num

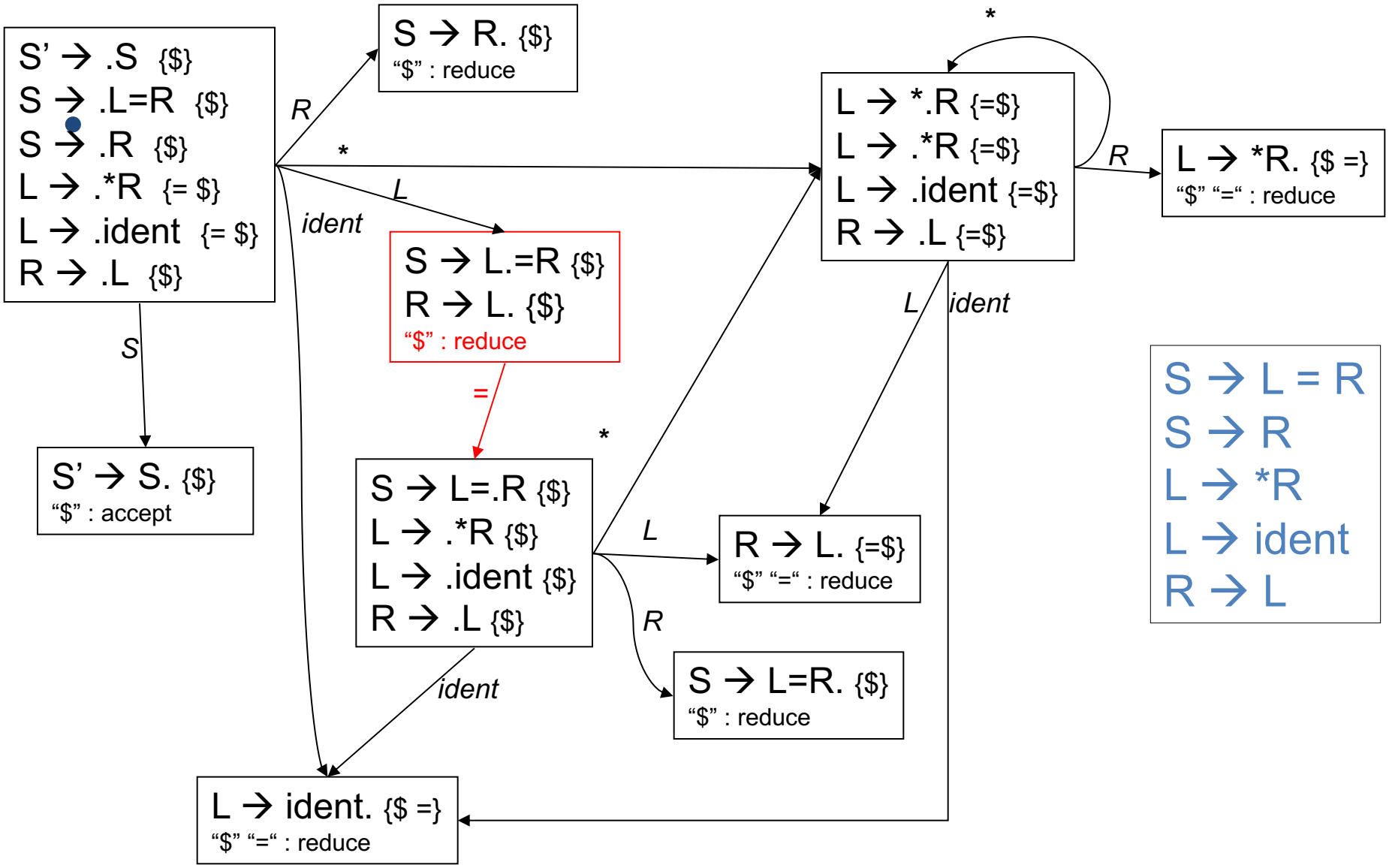
	num	+	$	E	S
1	s4			g2	g6
2		s3	S→E		

LALR(1)

- SLR: Decizia de reduce pentru "X->β" se face doar daca urmatorul simbol e in FOLLOW(X)
- LALR(1): Fiecare productie are un set de simboli care ii pot urma. "X->β , S" se reduce doar daca urmatorul simbol e in S.
- Seturile se pot calcula prin aplicarea recursiva a regulilor:

S' →.S, {$}	B→ α.Aβ c in FIRST(β) A → .Xδ , {c}	B → α.Aβ , {c} c ∈ FOLLOW(β) β →* ε A → .Xδ , {c}	X→ α.Xβ , c X → αX.β , c
S' → .S , {$} S → .L=R S → .R L → .*R L → .ident R → .L	S' → .S , {$} S → .L=R S → .R L → .*R , {=} L → .ident , {=} R → .L	S' → .S , {$} S → .L=R , {$} S → .R , {$} L → .*R , {= $} L → .ident , {= $} R → .L , {$}	L → .ident, {= $} L → ident. , {= $}

DFA pentru LALR



Sujets

ALF
language crunch

- LR
- SLR
- LALR

Questions

ALF
language crunch