#### Bottom-up Parsing

- 从左到右对 Input Terminal 扫描.
- 根据输入的 Input 选择相应的产生式模拟最右推导的逆(也可看成 自下而上建立语法树,由叶索根).
- 无回溯 (Non-backtracking).

#### 归约 (reduction)

- 是推导关系的逆,
- 最左归约是最右推导关系的逆.
- 将寻找文法符号串中的一个子串,该子串和某一个产生式 RHS 的 文法符号串一致,用该产生式 LHS 的非终结符替换该串后得到的 文法符号串满足归约关系:

$$\alpha \gamma \beta \Leftarrow \alpha A \beta \text{ if } A \rightarrow \gamma \in P$$

#### Example

```
XL grammar
T = \{ ID, PLUS, TIMES, \}
      LP, RP);
N = {exp, term, fac};
S = \exp;
P:
1/ exp -> exp PLUS term
       l term
3/ term -> term TIMES fac
        | fac
5/ fac -> ID
       | LP exp RP
```

# 最左归约 ID PLUS ID TIMES ID ID PLUS ID TIMES ID 1 ← fac PLUS ID TIMES ID ID PLUS ID TIMES ID 2 ← term PLUS ID TIMES ID ID PLUS ID TIMES ID 3 ← exp PLUS ID TIMES ID ID PLUS ID TIMES ID

● 在第5如果用以下方式归约将出错:
5 ← exp PLUS term TIMES II
6' ← exp TIMES ID

#### Example

#### XL grammar

2 | term

3/ term -> term TIMES fac 4/ | fac

#### 最左归约

	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
1		

6 exp PLUS term TIMES fac
7 exp PLUS term

ID PLUS ID IIMES II

ID PLUS ID TIMES ID

问题

在第5如果用以下方式归约将出错:

 $6' \underset{rm}{\Leftarrow} expT$ 

#### Example

#### XL grammar T = { ID, PLUS, TIMES, LP, RP); N = {exp, term, fac}; S = exp;P: 1/ exp -> exp PLUS term l term 3/ term -> term TIMES fac | fac 5/ fac -> ID 6/ | LP exp RP

最	最左归约		
		ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
1	← rm	fac PLUS ID TIMES ID	ID PLUS ID TIMES ID

#### Example

#### XL grammar $T = \{ ID, PLUS, TIMES, \}$ LP, RP); N = {exp, term, fac}; S = exp;P: 1/ exp -> exp PLUS term l term 3/ term -> term TIMES fac | fac 5/ fac -> ID | LP exp RP

ī	最左归约			
1			ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
ı	1	← rm	fac PLUS ID TIMES ID	ID PLUS ID TIMES ID
ı	2	← rm	term PLUS ID TIMES ID	ID PLUS ID TIMES ID
ı				
н				

#### Example

#### XL grammar $T = \{ ID, PLUS, TIMES, \}$ LP, RP); N = {exp, term, fac}; S = exp;P: 1/ exp -> exp PLUS term l term 3/ term -> term TIMES fac 4/ | fac 5/ fac -> ID 6/ | LP exp RP

最	最左归约			
		ID PLUS ID TIMES ID	ID PLUS ID TIMES ID	
1	← rm	fac PLUS ID TIMES ID	ID PLUS ID TIMES ID	
2	← rm	term PLUS ID TIMES ID	ID PLUS ID TIMES ID	
3		exp PLUS <u>ID</u> TIMES ID	ID PLUS ID TIMES ID	
4		exp PLUS fac TIMES ID		

#### Example

#### XL grammar $T = \{ ID, PLUS, TIMES, \}$ LP, RP); N = {exp, term, fac}; S = exp;P: 1/ exp -> exp PLUS term | term 3/ term -> term TIMES fac 4/ | fac 5/ fac -> ID 6/ | LP exp RP

最左归约				
		ID PLUS ID TIMES ID	ID PLUS ID TIMES ID	
1	← rm	fac PLUS ID TIMES ID	ID PLUS ID TIMES ID	
2	← rm	term PLUS ID TIMES ID	ID PLUS ID TIMES ID	
3	← rm	exp PLUS <u>ID</u> TIMES ID	ID PLUS ID TIMES ID	
4	← rm	exp PLUS fac TIMES ID	ID PLUS ID ITMES ID	

#### Example

# XL grammar

3/ term -> term TIMES fac 4/ | fac

#### 最左归约

	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
1 ¢		ID PLUS ID TIMES ID
2 ¢		ID PLUS ID TIMES ID
3 <b>←</b>		ID PLUS ID TIMES ID
4 ¢		ID PLUS ID ITMES ID
5 ¢		ID PLUS ID TIMES <b>ID</b>
6 ¢		
7 <		

果用以下方式归约将出错:

o' ← exp TIMES ID

#### Example

#### 

| LP exp RP

# □D PLUS ID TIMES ID ID PLUS ID TIMES ID 1 ← fac PLUS ID TIMES ID ID PLUS ID TIMES ID 2 ← term PLUS ID TIMES ID ID PLUS ID TIMES ID 3 ← exp PLUS ID TIMES ID ID PLUS ID TIMES ID 4 ← exp PLUS fac TIMES ID ID PLUS ID ITMES ID 5 ← exp PLUS term TIMES ID ID PLUS ID TIMES ID 6 ← exp PLUS term TIMES fac ID PLUS ID TIMES ID

问题

5/ fac -> ID

5  $\Leftarrow$  expPLUS term TIMES ID

exp 11MES 1D

NOT match any H

- 7/100 -

#### Example

```
XL grammar
T = { ID, PLUS, TIMES,
    LP, RP};
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2    | term
```

4/ | fac 5/ fac -> ID 6/ | LP exp RP

3/ term -> term TIMES fac

#### 最左归约

最左归约				
	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID		
1 <	m fac PLUS ID TIMES ID	ID PLUS ID TIMES ID		
2 <	term PLUS ID TIMES ID	ID PLUS ID TIMES ID		
3 ¢	exp PLUS <u>ID</u> TIMES ID	ID PLUS ID TIMES ID		
4 <	= exp PLUS <u>fac</u> TIMES ID	ID PLUS ID ITMES ID		
5 ¢	exp PLUS term TIMES ID	ID PLUS ID TIMES <b>ID</b>		
6 ¢	= exp PLUS <u>term TIMES fac</u>	ID PLUS ID TIMES ID		
7 <	= exp PLUS term	ID PLUS ID TIMES ID		
8 <				

#### 问题

● 在第5如果用以下方式归约将出错:

5 \( \Leftrightarrow \text{exp PLUS term TIMES II} \)
6' \( \Leftrightarrow \text{exp TIMES ID} \)

Not match any RHS of productions!

#### Example

#### 

| LP exp RP

#### 最左归约

40.2		
	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
	m fac PLUS ID TIMES ID	ID PLUS ID TIMES ID
2 <		ID PLUS ID TIMES ID
3 ¢		ID PLUS ID TIMES ID
4 <	= exp PLUS <u>fac</u> TIMES ID	ID PLUS ID ITMES ID
5 ¢	exp PLUS term TIMES ID	ID PLUS ID TIMES <b>ID</b>
6 <	= exp PLUS <u>term TIMES fac</u>	ID PLUS ID TIMES ID
7 <	= exp PLUS term	ID PLUS ID TIMES ID
8 <	$=$ $\exp$	ID PLUS ID TIMES ID

#### 问题

● 在第6如果用以下方式归约将出错:

5 \( \infty \) exp PLUS term TIMES II
6' \( \infty \) exp TIMES ID

Not match any RHS of productions!

#### Example

#### XL grammar

3/ term -> term TIMES fac 4/ | fac

#### 最左归约

ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
1 $\leftarrow$ fac PLUS ID TIMES II	ID PLUS ID TIMES ID
2 $\leftarrow$ term PLUS ID TIMES	ID ID PLUS ID TIMES ID
3 $\Leftarrow$ exp PLUS <u>ID</u> TIMES I	D ID PLUS ID TIMES ID
4 \(\phi\) exp PLUS fac TIMES:	ID ID PLUS ID ITMES ID
5 $\Leftarrow$ expPLUS term TIME	S <u>ID</u> ID PLUS ID TIMES <b>ID</b>
6 $\Leftarrow$ expPLUS term TIME	S fac ID PLUS ID TIMES ID
7 \( \infty \text{exp PLUS term} \)	ID PLUS ID TIMES ID
8 ← exp	ID PLUS ID TIMES ID

#### 问题

● 在第⑤如果用以下方式归约将出错:

5 \( \infty \text{exp PLUS term TIMES ID} \)
6' \( \infty \text{exp TIMES ID} \)

Not match any RHS of productions!

最左归约: ᢤ ID PLUS ID TIMES ID

ID

PLUS

TD

'IMES

ID)





最左归约: <del>\* fac PLUS ID TIMES ID</del> fac





最左归约: ⇐ term PLUS ID TIMES ID fac





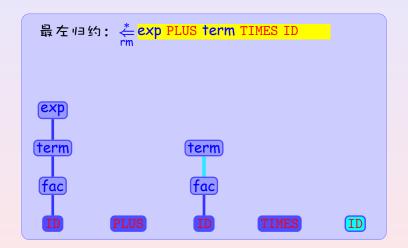




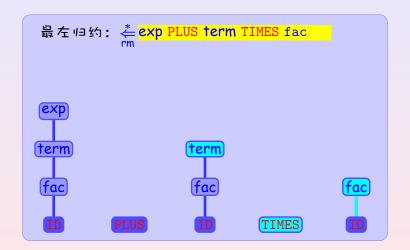
最左归约: ⇐ exp PLUS fac TIMES ID (term) fac fac





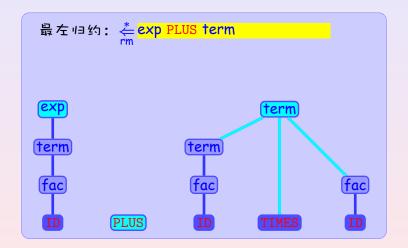




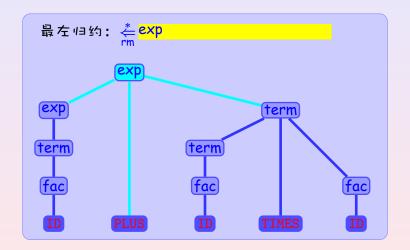






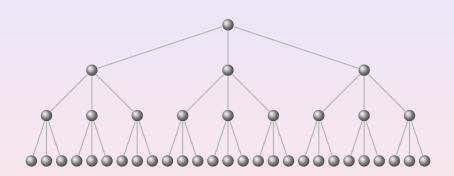


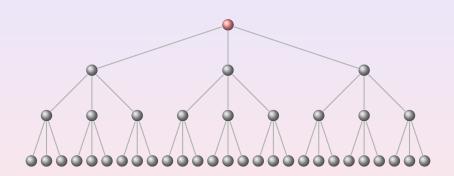


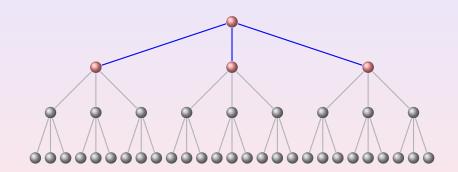


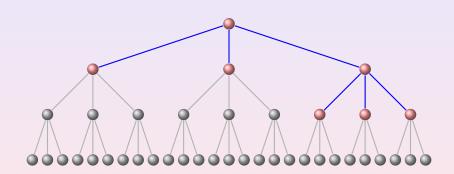


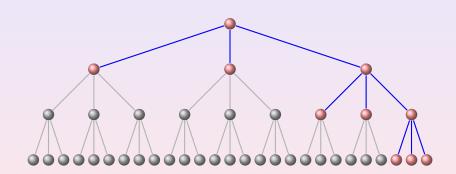


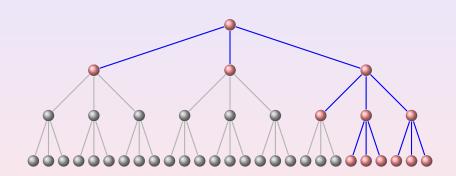


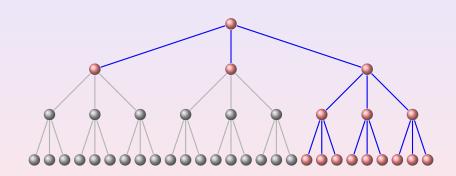


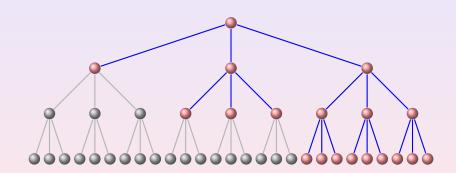


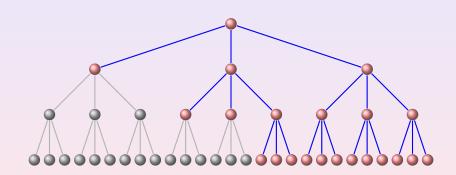


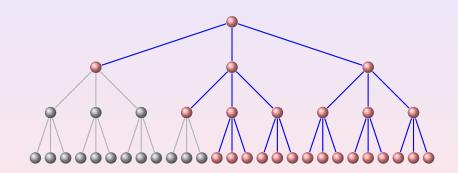


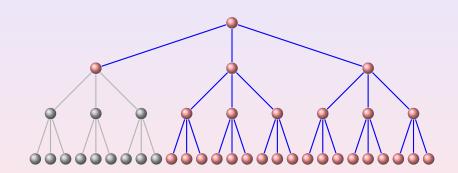


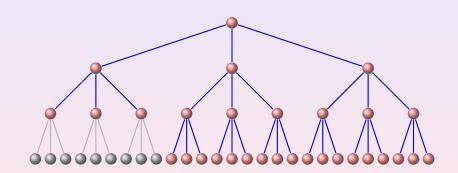


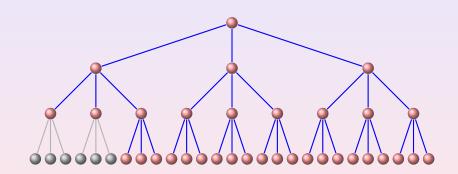


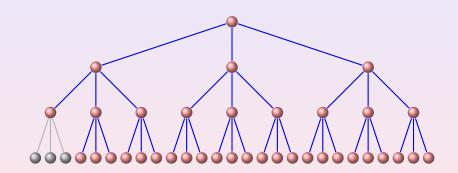


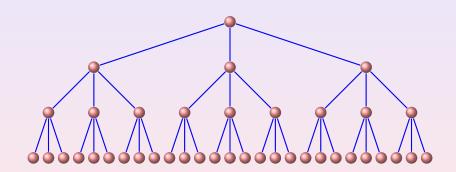




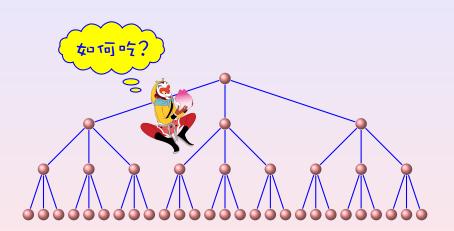


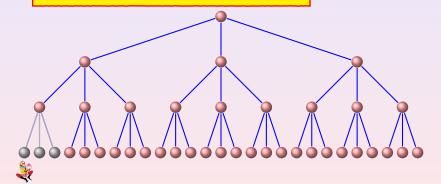


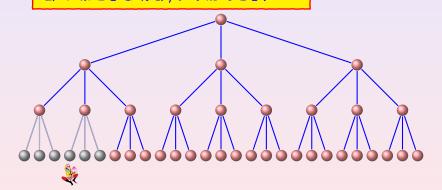


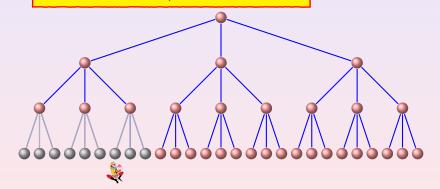


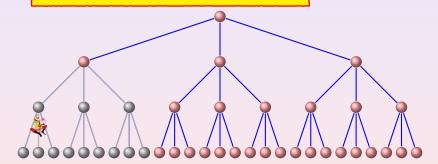
裁枝过程 --- 吃桃子的法则

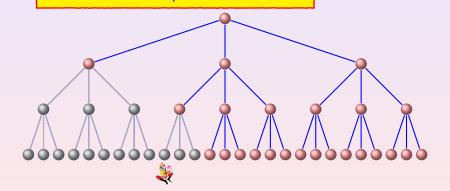


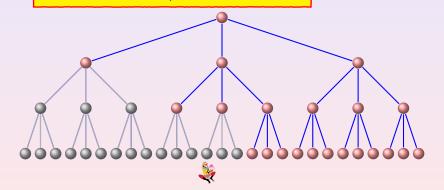


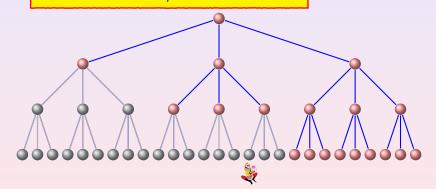


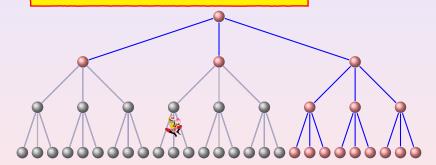


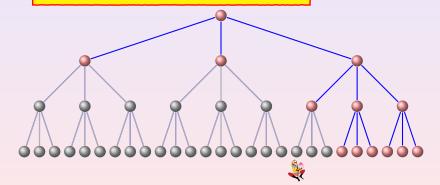


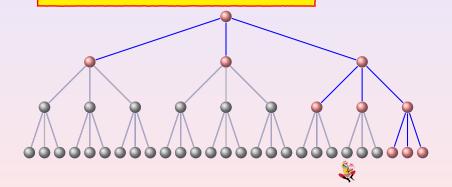


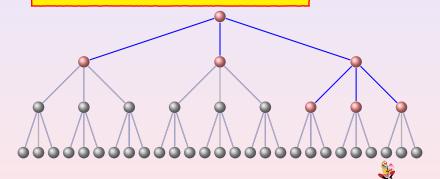


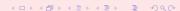


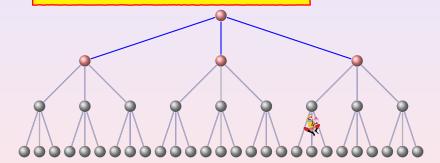


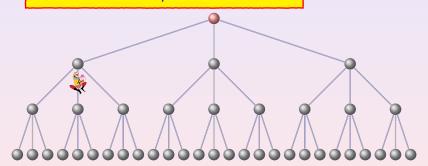


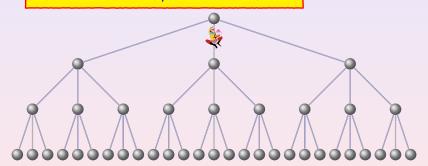












#### 现实的困难

- 树的情况未知,看不到树的宗整结构,只能看到已吃完的树干和当前的树叶。
- 在吃完一组叶结点后如何动作,向前或向上,且不能有丝毫的错误(无回溯)。

# 解决问题的思路

• 保留已吃过的树干作为下次吃法的判断依据.

### 相关性质

- 每组儿子一定是产生式右边的文法符号串,吃过一组后应知道其父亲。
- 已吃过的树干从左到右排列一定是对树的``垂直+水平'' 遍历所经历的叶结点序列。

## 最左归约的性质

- 被归约的对象一定是某一产生式 RHS 的文法符号串  $\alpha$ ,将  $\alpha$  看成是  $T \cup N$  为字母表的一个正则表达式, $\beta \in (T \cup N)^*$  是要归约的对象,则在  $\beta$  中寻找  $\alpha$  子串可以用自动机解决。
- 被归约对象之后一定是终结符号串:

过程: <u>αβw</u>

- 不确定性:
  - exp PLUS term TIMES ID 可以归约的对象有:
    - 1) exp PLUS term TIMES ID
    - 2) exp PLUS term TIMES ID
    - 3) exp PLUS term TIMES ID

但是只有(1)是唯一一个正确的.

- 正确的被归约的对象称为白柄 (handle),即它能保证被归约后一定还保持着最右句型。 ■
  - 1) exp PLUS term TIMES  $\underline{ID} \Leftarrow \exp PLUS term TIMES \underline{fac}$
  - 2) exp PLUS term TIMES ID = exp PLUS exp TIMES ID
  - 3) exp PLUS term TIMES ID exp TIMES ID

只有 (1) 是 (1)

- 如果文法是没有二义性的,则一个最右句型的句柄是唯一的.
- 寻找句柄是解决自度向上分型的关键.

page-break

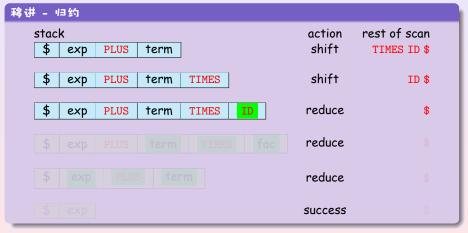
移进	- 4	ョ約							
stack							action	rest of scan	
	\$	exp	PLUS	term			shift	TIMES ID \$	
							shift		
							reduce		
							reduce		
							reduce		
	\$	ехр					success	\$	



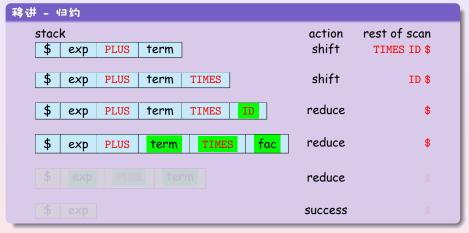


移讲	移进 - 归约									
stack							action	rest of scan		
	\$	exp	PLUS	term			shift	TIMES ID \$		
	\$	exp	PLUS	term	TIMES		shift	ID\$		
							reduce			
							reduce			
							reduce			
	\$	exp					success	\$		



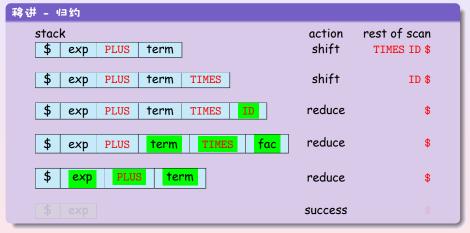






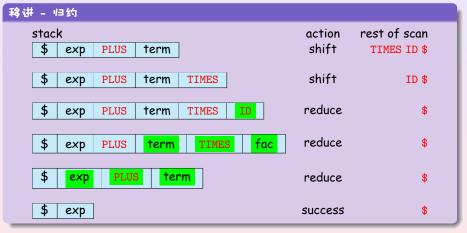
















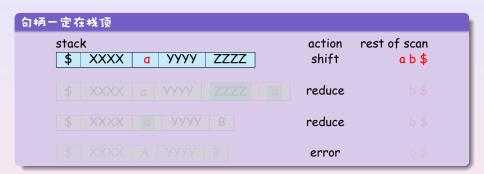
#### 分析栈

- 保留已经归约的句型.
- 自左向右扫描.
- "stack" + "rest of scan" = 最右句型。
- 句柄总是在栈顶形成。
- 五回溯.

## 问题

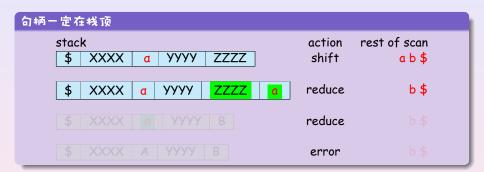
- 在归约无误的前提下, 句柄是否能保证总是在栈顶形成.
- 如何不回看栈中元素,仅根据栈顶的状态和当前的输入就能 够正确地做出移进或归约的操作

# 句柄一定在栈顶 (1/2)

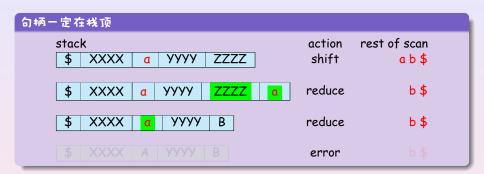




## 台柄一定在栈顶 (1/2)



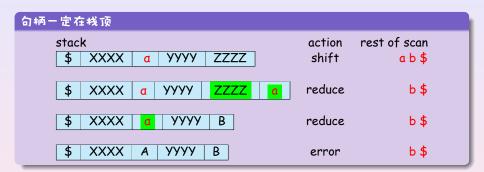
## 句柄一定在栈顶 (1/2)



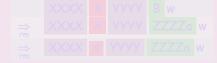
这与最右维导矛盾:



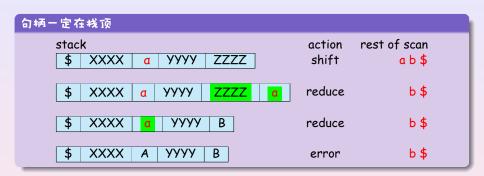
## 句柄一定在栈顶 (1/2)



这与最右推导矛盾:



## 白柄一定在栈顶 (1/2)



#### 这与最右推导矛盾:

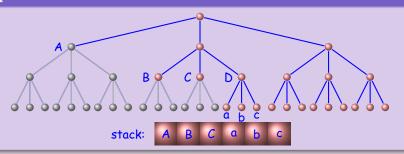


### 句柄一定在栈顶 (2/2)

### 原因

- 栈内的元素一定是已经归约到不能再归约的对象,否则,由于是句柄, 在新的移进前还要归约。
- 句柄的最后一个文法符号一定是栈顶元素。
- 最右归约过程是自应向上从左到右的裁核过程。

### 裁枝



### 如何判断句柄已经在栈顶形成

# 活前缀(viable prefix)

- 最右句型中句柄之前(含句柄)的文法符号串。
- 最右句型的一个前缀,该前缀不含有句柄以后的文法符号.
- 移进 归约分析栈中的符号串.

### Example

Sentential form: exp PLUS termTIMES ID w

### 活前缀的性质

- 任一活前缀都可以在其后增加终结符而形成一个右句型。
- 活前缀中出现在前的文法符号的辈分一定不低于后面的.
- 活前缀中的任一文法符号一定是最右推导中从未被展开的符号.
- 如果分析器能识别活前缀并且能预测形成活前缀的所有可能的最 右维导就能进行无回溯的分析。

### Example

Sentential form: expPLUS termTIMES <u>ID</u> w

可以形成最右句型的 w:

在以下讨论中用+,\*,(和)分别表示终结符PLUS, TIMES, LP和RP.

### 自动机所接受的文法符号串: 清前編

### 相关性质

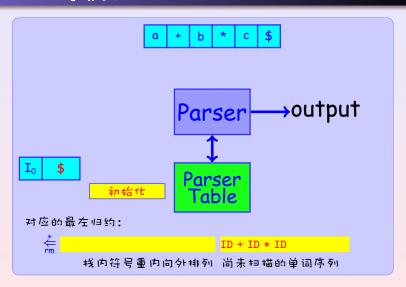
- 活前缀是语法树由父亲到其最左儿子垂直向下偏历或具有相同父亲结点从最左端开始自左向右水平偏历所经过的节点序列
- 自动机从开始状态出发的任何一条路经与上述偏历一一对应
- 即某一文法符号串是活前缀,当且仅当,它是自动机从开始状态到某一个或多个状态所经历的边的序列。
- 形成某一活前缀的所有可能的最右维导与自动机在接受该 活前缀后所到达的状态集一致。
- 自动机从开始状态出发到达任何一个状态,其状态对应的项目一定能最终形成句柄。
- 该自动机称为识别活前缀的自动机(简称前缀自动机)。

### 活前缀与有效项目

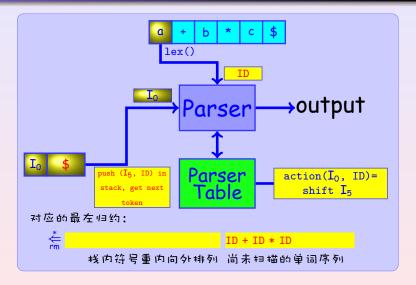
#### 定义与定理

- 设  $\alpha\beta_1$  是一个活前缀,设  $A \to \beta_1 \bullet \beta_2$  是一个以  $\beta_1$  为前缀的项目,称  $A \to \beta_1 \bullet \beta_2$  是  $\alpha\beta_1$  的一个有效项目 iff  $S' \overset{*}{\Rightarrow} \alpha\beta_1\beta_2$ w.
- 表示活前缀  $\alpha \beta_1$  中的后缀  $\beta_1$  可以形成  $A \rightarrow \beta_1 \beta_2$  的句柄.
- β1 ≠ ε,表示当前活前缀中还没有句柄中的一个符号。
- β<sub>2</sub> = ε,表示句柄已经形成。
- 所有的有效项目集合涵盖了从文法开始符号最右推出该活前缀的所有可能。
- 任何一个活前缀的有效项目就是自动机从开始状态出发在接受该 活前缀后所能到达的状态对应的项目。
- 分析就是根据状态对应项目和当前的输入选择唯一的移进-归约操作。

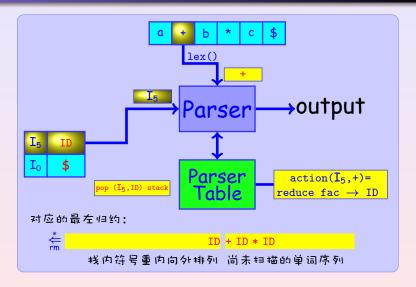
page-break

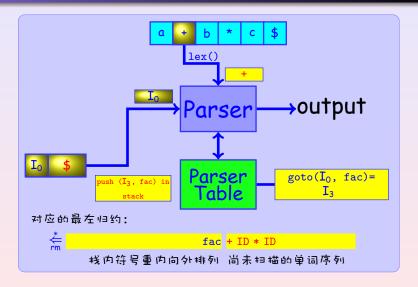


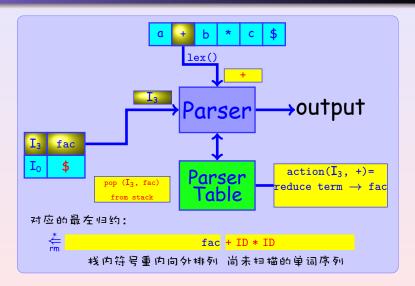




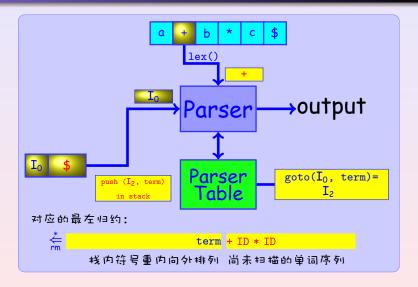


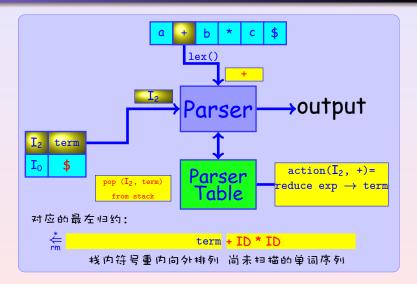




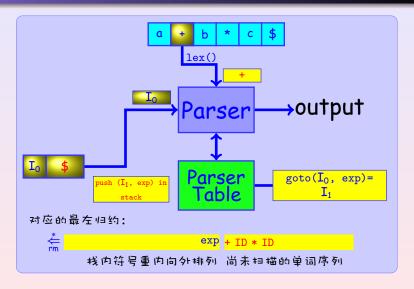






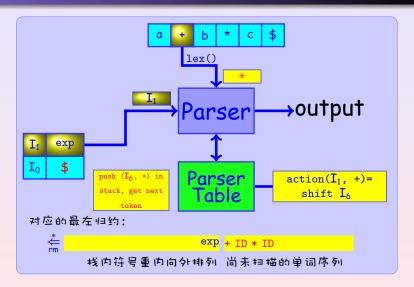




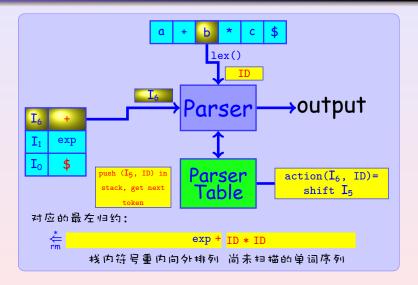




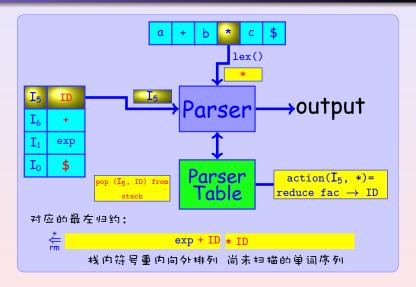


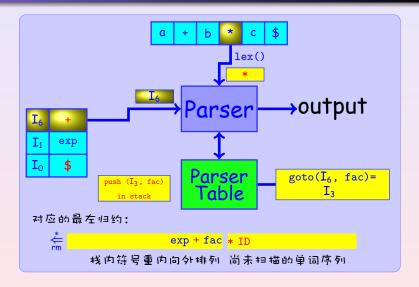


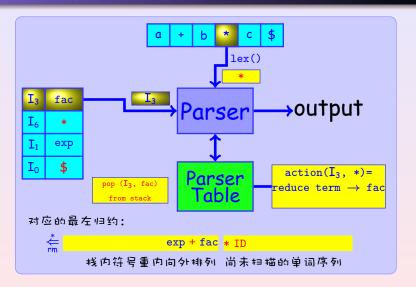




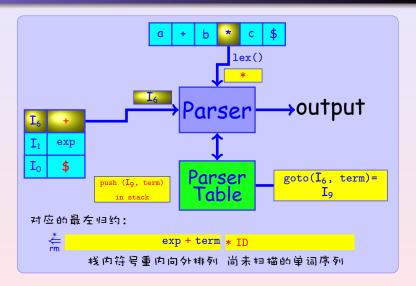




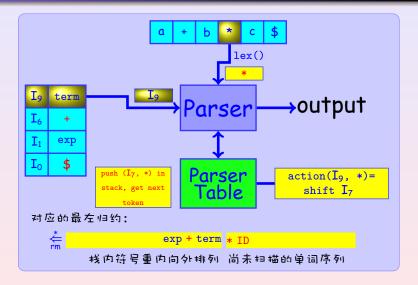




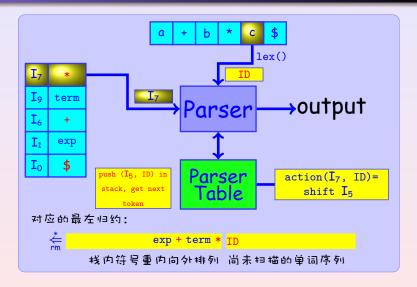


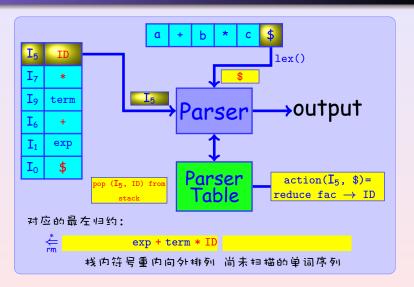




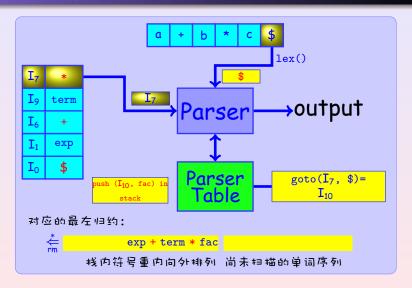




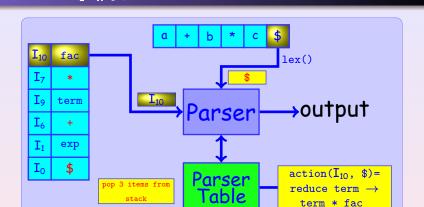








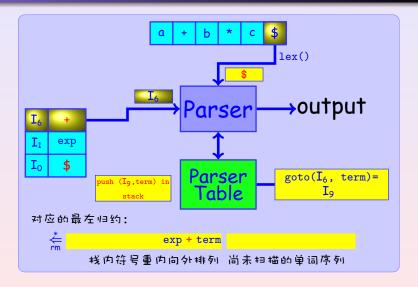


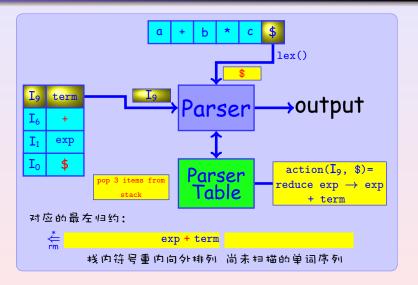


对应的最左归约:

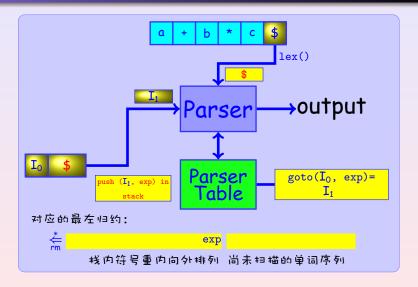
( 移进归约

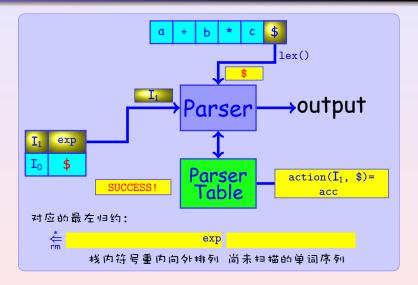












#### XL 语言的 SLR 分析表

#### XL 语言文法

1/exp 
$$\rightarrow$$
 exp+term 2/exp  $\rightarrow$  term 3/term  $\rightarrow$  term\* fac 4/term  $\rightarrow$  fac 5/fac  $\rightarrow$  ID 6/fac  $\rightarrow$  (exp)

			ac	tion				goto	
状态	ID	+	*	(	)	\$	exp	term	fac
0	<i>s</i> 5			<i>s</i> 4			1	2	3
1		<b>s</b> 6				acc			
2		r2	<i>s</i> 7		r2	r2			
3		r4	r4		r4	r4			
4	<i>s</i> 5			s4			8	2	3
5		r5	r5		r5	r5			
6	<i>s</i> 5			<i>s</i> 4				9	3
7	<i>s</i> 5			s4					10
8		<b>s</b> 6			s11				
9		r1	<i>s</i> 7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

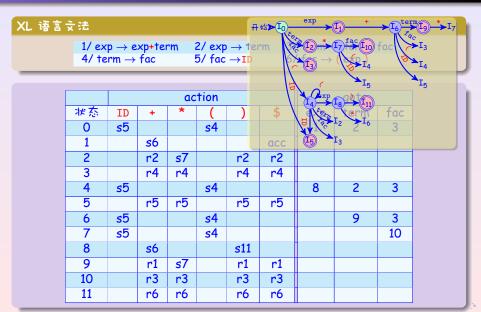
### XL 语言的 SLR 分析表

#### XL 语言文法

1/ exp 
$$\rightarrow$$
 exp+term 2/ exp  $\rightarrow$  term 3/ term  $\rightarrow$  term\* fac 4/ term  $\rightarrow$  fac 5/ fac  $\rightarrow$ ID 6/ fac  $\rightarrow$  ( exp )

			ac	tion				goto		
状态	ID	+	*	(	)	\$	exp	term	fac	
0	<i>s</i> 5			s4			1	2	3	
1		<b>s</b> 6				acc				
2		r2	<i>s</i> 7		r2	r2				
3		r4	r4		r4	r4				
4	<i>s</i> 5			s <b>4</b>	C C	12 E	exp	2 +	3 ter	* T
5		r5	r5		r5 <sup>#</sup>	70 to		fac @	T.	c 17
6	<i>s</i> 5			s <b>4</b>		(3		I7 10	\$/>	<b>I</b> 3
7	<i>s</i> 5			s <b>4</b>		\ \	<u>*</u>	<b>→</b> I <sub>4</sub>	10	1 <sub>4</sub>
8		<b>s</b> 6			<b>s11</b>	17		<b>1</b> <sub>15</sub>		I <sub>5</sub>
9		r1	<i>s</i> 7		r1	r1	X xp			
10		r3	r3		r3	r3	te <sub>r</sub>	18 × (11)		
11		r6	r6		r6	r6		$I_2 \longrightarrow I_6$		

### XL 语言的 SLR 分析表



# 分析过程

stack	symbols	input	action
0		ID+ID*ID\$	shift
	ID	+ID*ID\$	reduce by fac→ID
	fac	+ID*ID\$	reduce by term→fac
	term	+ID*ID\$	reduce by exp→term
	exp	+ID*ID\$	shift
	exp+	ID*ID\$	shift
	exp+ID	*ID\$	reduce by $fac \rightarrow ID$
	exp+fac	*ID\$	reduce by term→fac
	exp+term	*ID\$	shift
	exp+term*	ID\$	shift
	exp+term*ID	\$	reduce by fac→ID
	exp+term*fac	\$	reduce by term→term*fac
	exp+term	\$	reduce by exp→exp*term
	exp	\$	accept

# 分析过程

10

11

r3

n6 n6

r3

stack	Sy	mbol	s			inpu	t a	ction			
0					ID+1	D*ID		hift			
	ID						\$ r				
	fa										
	te										
	ex										
	ex										
	ex						\$ r				
0163	<u> </u>			a	ction			П	goto		ĵ.
0169	状态	ID	+	*	(	)	\$	exp	term	fac	
01697	0	<i>s</i> 5	s6		s4		acc	1	2	3	
01697	2		r2	<i>s</i> 7		r2	r2				
01697	3		r4	r4	- 4	r4	r4		2	2	m*fac
0169	4 5	<i>s</i> 5	r5	r5	s4	r5	r5	8	2	3	term
01	6	<i>s</i> 5	. 0	. 0	s4	. 0	. 0		9	3	
	7 8	<i>s</i> 5	s6		s4	s11				10	
	9		r1	s7		r1	r1				

r3

**~**4

r3

**~**4

# 分析过程

9

10

11

r1

r3

**n**6

s7

r3

**~**4

stack	S	ymbol	S			inpu	†	ac	tion			
0					ID+1	[D*ID	\$	sh	ift			
05	I	D			+]	[D*ID	\$	re	duce	by fac	$\rightarrow$ ID	
	f						\$					ıc
	t						\$					rm
01	e:						\$					
016	e:		+									
	e:						\$					
0163				а	ction		_			goto		1
0169	状态	ID	+	*	(	)	\$		exp	term	fac	
01697	0	<i>s</i> 5			s4				1	2	3	
01697	2		s6 r2	s7		r2	ac					-
01697	3		r4	r4		r4	r					m*fac
0169	4	1 <i>s</i> 5			s4				8	2	3	term
01	5	r5 r5			s4	r5	rţ	)		9	3	
	7				s4					7	10	
	8	so s6				s11						

r1

r3

**~**4

r1

r3

**~**4

stack	Sy	mbol	S			inpu	t	ac	tion			
0					ID+I	D*ID	\$	sh	ift			
05	ID	)			+1	D*ID	\$			by fac		
03	fa	ıC			+1	D*ID	\$	re	duce	by ter	m→fa	.c
02	te						\$					m
01	ex						\$		ift			
016	ex				ID*ID\$							
0165	ex				*ID\$			reduce by fac $\rightarrow$ ID				
0163		C			action					goto		î
0169	状态				(	)	5	5	exp	term	fac	
01697	0	0 s5			s4				1	2	3	

4				ac	tion				goto	
	状态	ID	+	*	(	)	\$	exp	term	fac
7	0	<i>s</i> 5			s4			1	2	3
7	1		s6				acc			
	2		r2	s7		r2	r2			
	3		r4	r4		r4	r4			
	4	<i>s</i> 5			s4			8	2	3
	5		r5	r5		r5	r5			
	6	<i>s</i> 5			s4				9	3
	7	<i>s</i> 5			s4					10
	8		s6			s11				
	9		r1	s7		r1	r1			
	10		r3	r3		r3	r3			
	11					/	/			

- 56/100 -

stack	sy	mbol	S			inpu	†   (	action				
0					ID+I	D*ID		shift				
05	ID	)			+]	D*ID		reduce				
03	fa	С			+]	D*ID		reduce				
02	te	rm			+]	D*ID	\$ I	reduce	by exp	→ter	m	
01	ex						\$ .	shift				
016	ex				ID*ID\$							
0165	ex				*ID\$			reduce by fac→ID				
0163				00	action				goto		S.	
0169	状态	ID	+	*	(	)	\$	exp	term	fac		
01697	0	0 s5						1	2	3		

				ac	tion				goto	
	状态	ID	+	*	(	)	\$	exp	term	fac
	0	<i>s</i> 5			s4			1	2	3
	1		<i>s</i> 6				acc			
	2		r2	<i>s</i> 7		r2	r2			
	3		r4	r4		r4	r4			
	4	<i>s</i> 5			s4			8	2	3
	5		r5	r5		r5	r5			
	6	<i>s</i> 5			<i>s</i> 4				9	3
١	7	<i>s</i> 5			s4					10
	8		<i>s</i> 6			s11				
	9		r1	s7		r1	r1			
	10		r3	r3		r3	r3			
	4.4		/	. /		./	. /			

stack	CV	mbol	c			inpu	+ 0	ction			
	эу	וטטווו	3								
0					ID+I	D*ID	S	hift			
05	ID	)			+1	D*ID	\$ r	educe	by fac-	$\rightarrow$ ID	
03	fa	С			+1	D*ID			by ter		
02	te	rm			+1	D*ID	\$ r	educe	by exp	→ter	m
01	ex	exp				D*ID	s	hift			
016	ех						\$ 5	hift			
0165	ex						\$ re				
0163											2
					ction				goto		
0169	状态	*			(	)	\$	exp	term	fac	
01697	0				s4			1	2	3	
01607	1	1 s6					acc				

2 r2 s7 r2 r2 r4 r4 r4 r4 4 3 *s*5 s4 8 2 5 r5 r5 r5 r5 6 *s*5 s4 9 3 *s*5 s4 10 8 s11 s6 9 r1 s7 r1 r1 10 r3 r3 r3 r3 11 **~**4 **~**4 **~**4 **~**4

stack	sy	mbol	S			inpu	t a	ction			
0					ID+I	D*ID	\$ <b>S</b>	hift			
05	ID	)			+]	D*ID	\$ r	educe	by fac	$\rightarrow$ ID	
03	fa	ıC			+]	D*ID	\$ r	educe	by ter	$\mathtt{m}{ ightarrow}\mathtt{fa}$	.C
02	te	rm			+]	D*ID	\$ r	educe	by exp	→ter	m
01	ex	р			+]	D*ID	\$ 5	hift			
016	ex	p+			I	D*ID	\$ 5	hift			
0165	ex						\$ r	reduce by fac→ID			
0163		_			ction			П	ooto.		ŝ
0169	状态	ID	+	*	CHON	· ·	\$	avn	goto term	fac	
01697		0 s5				)	Ф	exp 1	2	3	
01037	3	33	-,		s4			-	_	3	

				- 1			4			
4				ac	tion				goto	
	状态	ID	+	*	(	)	\$	exp	term	fac
7	0	s5			s4			1	2	3
7	1		<i>s</i> 6				acc			
_	2		r2	s7		r2	r2			
	3		r4	r4		r4	r4			
	4	s5			s4			8	2	3
	5		r5	r5		r5	r5			
	6	<i>s</i> 5			s4				9	3
-	7	<i>s</i> 5			s4					10
	8		s6			s11				
	9		r1	s7		r1	r1			
	10		r3	r3		r3	r3			
	11		n4	<b>n</b> 4		n4	n4			

stack	symbols	input	action			
0		ID+ID*ID\$	shift			
05	ID	+ID*ID\$	reduce by fac→ID			
03	fac	+ID*ID\$	reduce by term→fac			
02	term	+ID*ID\$	reduce by exp→term			
01	exp	+ID*ID\$	shift			
016	exp+	ID*ID\$	shift			
0165	exp+ID	*ID\$	reduce by fac→ID			
0163	_	goto				

							4				
				ac	tion		goto				
	状态	ID	+	*	(	)	\$	exp	term	fac	
	0	<i>s</i> 5			s4			1	2	3	
	1		<i>s</i> 6				acc				
	2		r2	s7		r2	r2				
	3		r4	r4		r4	r4				
	4	<i>s</i> 5			s4			8	2	3	
	5		r5	r5		r5	r5				
	6	<i>s</i> 5			s4				9	3	
-	7	<i>s</i> 5			s4					10	
	8		s6			s11					
	9		r1	s7		r1	r1				
	10		r3	r3		r3	r3				
	11		n6	n6		n6	n6				

	0000000000	0000000	00000	000000		tion	00000	000000	00000000		000000	0000000	
τ	祈讨和	状态	TD	+	ac	( ) \$			0.45	goto exp term fac			
		الا الا الا	ID s5	+		s4	)	P	exp	term 2	3		
		1	3.7	s6		37		acc	1		3		
	stack	2		r2	s7		r2	r2					
	0	3		r4	r4		r4	r4					
	05	4	<i>s</i> 5			s4			8	2	3		
		5		r5	r5	- 4	r5	r5		_	_		
	03	6 7	<i>s</i> 5			s4 s4				9	3 10	;	
	02	8	SO	s6		54	s11				10	1	
	01	9		r1	s7		r1	r1					
	016	10		r3	r3		r3	r3					
	0165	11		r6	r6		r6	r6					
		1											
	0163	ez	rp+fa	C			*ID						
		ez						\$ 5	hift				
		ez						\$ 5					
		016975 exp+term*ID			D	\$			reduce by fac→ID reduce by term→term*fa				
					ac								
		ez						\$ 0					

分析过和	action goto												
					(	)	\$	exp term		fac			
0 <i>s</i> 5					s4			1	2	3			
	1		<i>s</i> 6				acc						
stack	2		r2	s7		r2	r2						
0	3	<i>s</i> 5	r4	r4	s4	r4	r4	8	2	3			
05	5	50	r5	r5	54	r5	r5	0	۷	3			
03	6	<i>s</i> 5	13	13	s4	13	13		9	3			
	7	<i>s</i> 5			s4					10	1		
02	8		s6			s11					1		
01	9		r1	s7		r1	r1						
016	10		r3	r3		r3	r3						
0165	11		r6	r6		r6	r6						
0163		exp+fa	.c			*ID	\$ r	reduce by term→fac					
0169	•	- exp+te	rm		*ID\$ shift								
01697	(				ID\$ shift								
01697	016975 exp+term*ID 0169710 exp+term*fac						\$ r	reduce by fac→ID reduce by term→term*fac					
01697													
0169	69 exp+term \$ r					reduce by exp→exp*term							
01	(							accept					

	000000000	0000	0000	00000	00000	00000	000000	00000	0000	000	0000000	0000000	000000	0000000 000	
4	祈过和						ction					goto			
		状态			+	*	(	)	\$		exp	term	fac		
0 s5					s4				1	2	3				
ſ	at a da	1			s6				ac						
	stack	2			r2	s7		r2	rá						
	0	3		<i>s</i> 5	r4	r4	s4	r4	r	+	8	2	3		
	05	5		50	r5	r5	54	r5	r!		0	2	3		
	03	ě		<i>s</i> 5	13	13	s4	13	١,			9	3		
		7	,	<i>s</i> 5			s4						10		
	02	8	3		s6			s11						1	
	01	9			r1	<i>s</i> 7		r1	r:						
	016	10			r3	r3		r3	r:						
	0165	1	1		r6	r6		r6	r	<b>5</b>					
	0163				_			TD	φ.	maduaa buu oo C					
				p+fa							reduce by term→fac				
	0169		ex	p+te	rm			*ID	\$	sr	nift				
	01697		ex	p+te	rm*			ID	\$	shift					
		•				D					reduce by fac→ID				
	01697 <u>10</u> ex								\$					erm*fac	
									\$						
									ф	reduce by exp→exp*term				) · GCIII	
									ф						

	action aoto												
4	计析过利	al es '	_		ı	a(	ction				goto		
		**	_	ID	+	^	s4	)	\$	exp	term 2	fac 3	
		0		<i>s</i> 5	s6		54		acc	1	2	3	
ſ	stack	2			r2	s7		r2	r2				
		3			r4	r4		r4	r4				
	0	4		s5			s4			8	2	3	
	05	5			r5	r5		r5	r5				
	03	6	)	s5			s4				9	3	
	02	7		s5			s4					10	
		8			<i>s</i> 6			s11					1
	01	9		r1 s7				r1	r1				
	016	10			r3	r3		r3	r3				
	0165	11	L		r6	r6		r6	r6				
	0163		ex	- p+fa	.c			*ID	\$ r	reduce	by ter	m→fa	ıc
	0169			p+te				*ID		shift	,		
	01697			- p+te				ID	\$ 5	shift			
	016975	5		-	rm*I	D			\$ r	reduce	by fac-	$\rightarrow$ ID	
	016971	10		p+term*fac									erm*fac
									\$ r				*term

	action goto												0000000 0	000	
7	祈讨和	*	态	ID	+	*	(	)	\$		exp	term	fac		
			0	<i>s</i> 5			s4				1	2	3		
	1 1		1		s6				acc						_
	stack		2		r2	s7		r2	r2						
	0		3 4	s5	r4	r4	s4	r4	r4		8	2	3		
	05		5	50	r5	r5	54	r5	r5		0		3		
	03		6	<i>s</i> 5	10	1.5	s4	1.5	13			9	3	1.	
	02		7	<i>s</i> 5			s4						10		
			8		<i>s</i> 6			s11						1	
	01		9		r1	s7		r1	r1						
	016		.0 !1		r3 r6	r3 r6		r3 r6	r3 r6						
	0165	Ľ.	. 1		1.0	10		10	10			,			
	0163		ex	p+fa	.c			*ID	\$	red	uce	by teri	n→fa	ac	
	0169		ex	p+te	rm			*ID	\$	shif	†				
	01697		ex	p+te	rm*			ID	\$	shif	†				
	01697	5	ex	p+te	rm*I	D			\$	red	uce	by fac-	$\rightarrow$ ID		
	01697	<u>10</u>	ex	p+te	rm*f	ac								erm*fac	
			ex						\$			by exp			
			ex												

0000000000000000000000000000000000000											0000000 000		
T	计析过利	-l /s -	_			ac	tion				goto		
		**	(3)	ID	+	^	(	)	\$	exp	term 2	fac 3	
		0		s5	<i>s</i> 6		s4		222	1	2	3	
	stack	2			r2	s7		r2	acc r2				
		3			r4	r4		r4	r4				
	0	4		s5	17	17	s4	17	17	8	2	3	
	05	5		30	r5	r5	3 1	r5	r5		_	J	
	03	6		s5			s4	. •			9	3	
		7		s5			s4				-	10	
	02	8						s11					1
	01	9		r1	s7		r1	r1					
	016	10			r3	r3		r3	r3				
	0165	11			r6	r6		r6	r6				
							*ID\$ reduce by term—fa						
	0163		ex	p+fa	.C			*ID			<b>by</b> teri	n→fa	.C
	0169		ex	p+te	rm			*ID	\$ sł	nift			
	01697		ex	- p+te	rm*			ID	\$ s	nift			
	016975	5	exp+term*ID			D			\$ re	educe	by fac-	$\rightarrow$ ID	
	016971	-								rm*fac			
	0169	exp+term		\$ reduce by exp→exp*term									
	01	exp						\$ ac			Ī		

00	00000000000000000000000000000000000000													
T	`析过和		_				ction					goto		
			态	ID	+	*	(	)	\$		exp	term	fac	
		(	,	<i>s</i> 5	46		s4				1	2	3	-
	stack	2	2		s6 r2	s7		r2	ac r2	-				
		3			r4	r4		r4	r4					
	0	2		<i>s</i> 5			s4				8	2	3	
	05	Ę	5		r5	r5		r5	r5	5				1
	03	6	ó	<i>s</i> 5			s4					9	3	;
	02	7	7	<i>s</i> 5			s4						10	
	01	8			s6	-7		s11						1
		1			r1 r3	<i>s</i> 7		r1 r3	r1 r3					
	016	1			r6	r6		r6	r6					
	0165		_		10	10		10						
	0163		ex	p+fa	.C			*ID	\$	rec	duce 1	by terr	n→fa	ic
	0169		ex	p+te	rm			*ID	\$	shi	ft			
	01697		ex	p+te	rm*			ID	\$	shi	ft			
	016975	5	ex	p+te	rm*I	D	· ·			rec	duce I	by fac-	$\rightarrow$ ID	
	016971	0	ex	p+te	rm*f	ac								erm*fac
	0169		ex	p+te	rm					reduce by exp→exp*term				
	01		ex	- :p							cept	•	·	

stack	symbols	input	action
0	·	ID+ID*ID\$	shift
05	ID	+ID*ID\$	reduce by fac→ID
03	fac	+ID*ID\$	reduce by term→fac
02	term	+ID*ID\$	reduce by exp→term
01	exp	+ID*ID\$	shift
016	exp+	ID*ID\$	shift
0165	exp+ID	*ID\$	reduce by fac→ID
0163	exp+fac	*ID\$	reduce by term→fac
0169	exp+term	*ID\$	shift
01697	exp+term*	ID\$	shift
016975	exp+term*ID	\$	reduce by fac→ID
01697 <u>10</u>	exp+term*fac	\$	reduce by term→term*fac
0169	exp+term	\$	reduce by exp→exp*term
01	exp	\$	accept

page-break

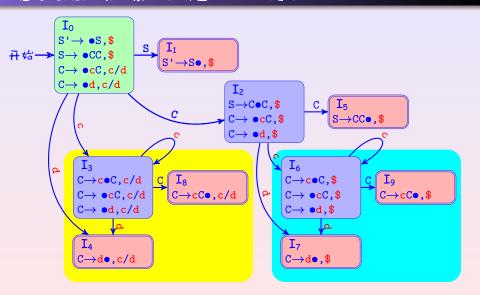
## LALR(LookAhead LR) 文法.

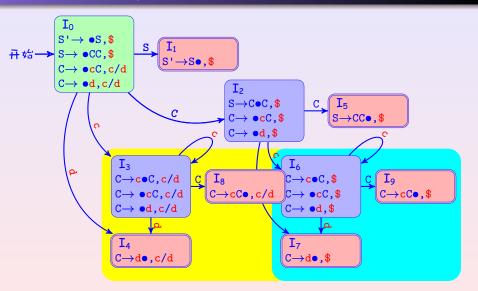
#### 解决 LR(1) 分析器状态过大问题的一个实际解决方案

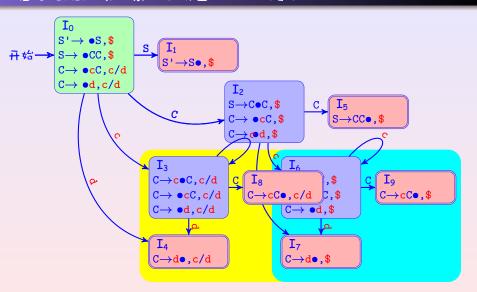
- LR(1) 规范项目集构造中 ε-closure 求出的项目集的第一分量和 LR(0) 的 一致.
- LR(1) DFA的 Dtrans 函数仅和项目集的第一分量有关,因此,它和LR(0)的也是一致的。
- 因此, LR(1) 的规范项目集是 LR(0) 的项目集加上不同的后确符号.

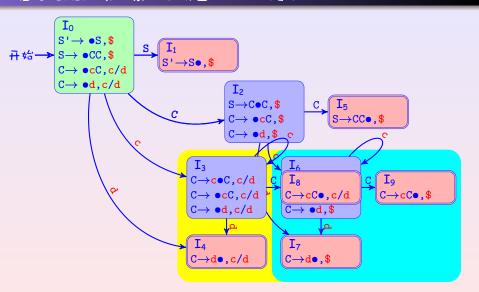
### 核 (core)

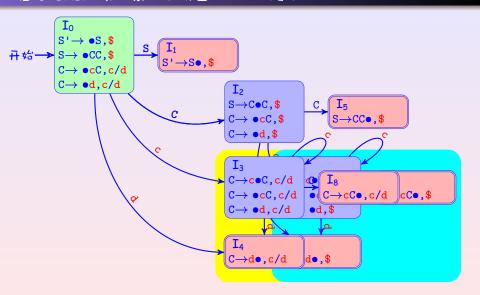
- LR(1) 规范项目集第一分量的集合称为该项目集的核.
- 每个核一定是规范 LR(0) 项目集.
- 设状态 I 和状态 J 对应的规范项目集是相同的核,则 Dtrans(I,X) 和 Dtrans(J,X) 也有相同的核,即核对状态转移函数是相容的.
- 合并 LR(1)的同核项目所得到的自动机称为LALR 自动机,该自动机的状态转换函数和 LR(0)的一致、唯一不同的是项目集增加了后随符号。

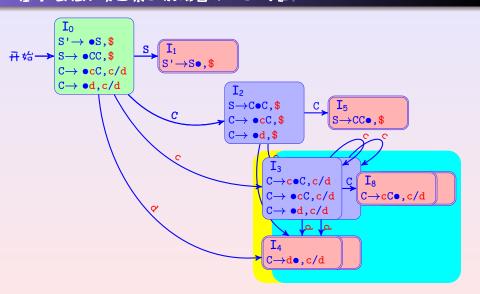


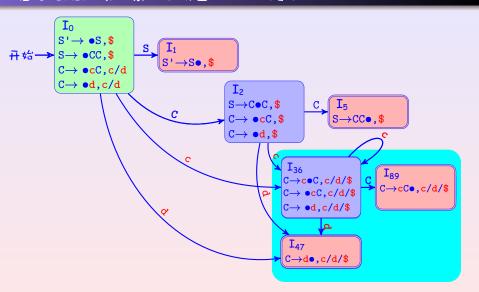












# LALR(1) 分析表

### 文法 G

$$1/S \rightarrow CC$$
  $2/C \rightarrow cC$   $3/C \rightarrow d$ 

		action		ge	oto
状态	С	d	\$	5	С
0	<i>s</i> 3	s4		1	2
1			acc		
2	<i>s</i> 6	<i>s</i> 7			5
36	<i>s</i> 36	s47			89
47	r3	r3	r3		
5			r1		
89	r2	r2	r2		

page-break



## 二义性的处理

- 二义文法 定不是 LR 文法, LR 文法 定不是二义文法.
- 二义文法比其对应的无二义性文法表达更简洁,自然,如: 表达式文法。
- 二义性表现为 LR 分析表中的 S-R 和 R-R 冲突.
- 在保证不破坏识别能力的前提下,通过对冲突项选择适当的操作解消二义性。

### 二义表达式文法 SLR 分析表

### 二义表达式文法

 $1/E \rightarrow E+E$   $2/E \rightarrow E*E$   $3/E \rightarrow (E)$   $4/E \rightarrow id$ 

			Actio	n			Goto
状态	id	+	*	(	)	\$	Е
0	s3			s2			1
1		s4	<i>s</i> 5			асс	
2	s3			s2			6
3		r4	r4		r4	r4	
4	s3			s2			8
5	s3			s2			8
6		s4	<i>s</i> 5		s9		
7		r1 s4	r1 <i>s</i> 5		r1	r1	
8		r2 s4	r2 s5		r2	r2	
9		r3	r2		r3	r3	

4日ト 4回ト 4 重ト 4 重ト

- expression:  $E \rightarrow E+E \mid E*E \mid (E) \mid id$
- 无层次结构.
- LR(0) 规范项目集 I<sub>7</sub> = {E → E+E•, E → E•+E, E → E•\*E}, 面对 ``+'' 和``\*'' 有移讲 归约冲容
- action[I<sub>7</sub>, +] = reduce E → E+E, 表示 ``+'' 左结合.
- action[I7, \*] = shift J, 表示 ``\*'' 运算的优先级别比 ``+'' 要高.
- LR(0) 规范项目集 I<sub>8</sub> = {E → E\*E•, E → E•+E, E → E•\*E}, 面对 ``+'' 和 ``\*'' 有務讲 归约冲突.
- action[I<sub>8</sub>, +] = reduce  $E \rightarrow E*E$ , 表示 ``\*'' 运算的优先级别比 ``+'' 要高
- action[I<sub>8</sub>,\*] = reduce E → E\*E, 表示 ``\*'' 左结合.

### 二义表达式文法 SLR 分析表

### 二义表达式文法

 $1/E \rightarrow E+E$   $2/E \rightarrow E*E$   $3/E \rightarrow (E)$   $4/E \rightarrow id$ 

			Ac	tion			Goto
状态	id	+	*	(	)	\$	E
0	s3			s2			1
1		s4	s5			acc	
2	s3			s2			6
3		r4	r4		r4	r4	
4	s3			s2			7
5	s3			s2			8
6		s4	<i>s</i> 5		s9		
7		r1	<i>s</i> 5		r1	r1	
8		r2	r2		r2	r2	
9		r3	r2		r3	r3	

## 含有一元减的表达式文法

- 表达式文法: E → E-E | E() | -E | id.
- 无层次结构.
- LR(0) 规范项目集 I = {E → -E•, E → E•-E, E → E•()}, 面对 ``-'' 和 ``('' 有移进 归约冲突.
- action[I, -] = reduce  $E \rightarrow -E$ , 表示一元 ``-'' 的优先级别比 二元 ``-'' 的高.
- action[I, (] = shift J, 表示后缀运算 ``()'' 运算的优先级别 比一元 ``-'' 要高.

#### if-then-else 结构

- if-then-else  $\bar{\chi}$   $\stackrel{\cdot}{\approx}$ : S → iSeS | iS | a.
- LR(0) 规范项目集: I = {S → iS eS, S → iS•}, 面对``e''
   有移进 归约冲突
- action[I, e] = shift J, 表示 ``e'' 将挂靠最近的 ``i''.
- action[I, e] = reduce  $S \to iS$ ,将破坏分析器的识别能力,如: ``iaea'' 在移进``ia'' 之后到达状态 I,使用归约操作后,再也无法移进``e''.

#### 出错处理

- 分析表对应的 Action 空白项表示出错,
- 一般处理方法:
  - 如果出错状态含有项目 A → α β,表示希望形成形如``αβ''的句柄,但是当前的错误输入使得分析器不能移进,此时可采用类似 LL 分析的方法,跳过当前的输入直到非终结符 A 的同步符号出现,用 A → αβ 归约继续分析.
  - 如果出错状态含有项目  $A \to \alpha$ •,表示当前的输入不是 A 的 Follow 集元素,处理方法,归约该项目,转入和前者一样的处理。

#### 二义表达式文法出错分析 --- error1

- Input: ``id+\*id'', 分析器在移进 ``id+'' 进入状态 I<sub>4</sub> = {E → E+ E, E → •E+E, E → •E\*E, E → •(E), E → •id}, 面对 ``\*'' 出错,此时的状态希望移进一个表达式成分 ``id'' 或 ``(''.
- 报错:缺少一个运算量.
- 恢复方法: 移进一个虚拟的``id'' 到下一个状态 J={E→ id●}.
- 相关的状态:有移进``id''的状态.

### 二义表达式文法出错分析 --- error2.

- 报错:不平衡出现的右括号.
- 恢复方法: skip ``)'', read next token.
- 相关的输入: ``)''.

#### 二义表达式文法出错分析 --- error3.

- Input: ``id+ id id'', 分析器在移进``id+'' 进入状态 I<sub>3</sub> = {E → id●}, 面对``id'' 出错,状态 I<sub>3</sub> 仅能在面对 Follow(E) = {+,\*,),\$}的元素才能归约,即缺少一个运算符.
- 報错: missing an operator.
- 恢复方法:
  - reduce  $E \rightarrow id$ ;
  - goto state  $I_7 = \{E \rightarrow E + E \bullet, E \rightarrow E \bullet + E, E \rightarrow E \bullet *E\};$
  - reduce  $E \rightarrow E+E$ ;
  - goto state  $I_1 = \{E' \rightarrow E \bullet, E \rightarrow E \bullet + E, E \rightarrow E \bullet *E\};$
  - 移进虚拟运算符号 ``+'', 转移状态到 I<sub>4</sub> = {E → E+ E, E →
     •E+E, E → •E\*E, E → •(E), E → •id}
- 有归约项目的状态面对任何输入都执行归约操作.

#### 二义表达式文法出错分析 --- error4

- Input: ``(id+ id'', 分析器在移进``(id+id'' 进入状态 I<sub>6</sub> = {E → E \*E, E → E +E, E → (E )}, 面对``\$'' 出错, 状态 I<sub>6</sub> 希望移进一个``)'' 形成平衡括号对.
- 报错: missing a right parenthesis.
- 恢复方法:
  - 移进一个虚拟的``)'' 到 state I<sub>9</sub> = {E → (E)•};;
  - reduce  $E \rightarrow (E)$ ;
  - goto state  $I_1 = \{E' \rightarrow E \bullet, E \rightarrow E \bullet + E, E \rightarrow E \bullet *E\}.$
- 相关状态: 6.
- error4 总是在面对 \$ 时发生.

### 二义表达式文法 SLR 分析表

### 二义表达式文法

 $1/E \rightarrow E+E$   $2/E \rightarrow E*E$   $3/E \rightarrow (E)$   $4/E \rightarrow id$ 

			Ac	tion			Goto
状态	id	+	*	(	)	\$	E
0	s3			s2			1
1		s4	<i>s</i> 5			асс	
2	s3			s2			6
3		r4	r4		r4	r4	
4	s3			s2			7
5	s3			s2			8
6		s4	<i>s</i> 5		s9		
7		r1	<i>s</i> 5		r1	r1	
8		r2	r2		r2	r2	
9		r3	r2		r3	r3	

### 二义表达式文法 SLR 分析表。

#### 分析表的出错处理

 $1/E \rightarrow E+E$   $2/E \rightarrow E*E$   $3/E \rightarrow (E)$   $4/E \rightarrow id$ 

			Ac	tion			Goto
状态	id	+	*	(	)	\$	E
0	s3	e1	e1	s2	e2	e1	1
1	e3	s4	<i>s</i> 5	e3	e2	acc	
2	s3	e1	e1	s2	e2	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8
6	e3	s4	<i>s</i> 5	e3	<i>s</i> 9	e4	
7	r1	r1	<i>s</i> 5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r2	r3	r3	r3	

stack	symbols	input	action
0		id+)\$	shift
03		+)\$	reduce by E $ ightarrow$ id
01		+)\$	shift
014		)\$	e2: 多余右括号, skip
014		\$	e1: 缺少运算量
			压id入栈到状态 3
0143		\$	reduce by E $ ightarrow$ id
0147		\$	reduce by E→E+E
01		\$	accept

sta	ck	S	symbols		inpu	†   ·	actior	1		
0					id+)	\$	shift			
						\$				
01						\$	shift			
			E+		)\$ e					kip
		- A. F-L		A	to E	1				
					Ac	tion			Goto	
	米	态	id	+	*	(	)	\$	Е	
	C	)	<b>s</b> 3	e1	e1	s2	e2	e1	1	
	1		e3	s4	<i>s</i> 5	e3	e2	acc		
	2	-	<i>s</i> 3	e1	e1	s2		e1	6	
	3	3	r4	r4	r4	r4	r4	r4		
	4		s3	e1	e1	s2	e2	e1	7	
	6.4	5	s3	e1	e1	s2	e2	e1	8	
	6	)	e3	s4	<i>s</i> 5	e3	<i>s</i> 9	e4		
	7	7	r1	r1	<i>s</i> 5	r1	r1	r1		

r2

r3

r2

r3

8

9

r2

r2

r3

r2

r3

r2

r3

	sta	ck	S	/mbo	ls	inpu <sup>r</sup>	t	action	1					
	0					id+)								
	03		ic	i		+);	\$	reduce by $\mathtt{E}{ ightarrow}\mathtt{id}$						
	01						\$	shift						
							\$			括号, skip				
							-1	th E	.					
						Ac	tion			Goto				
		妆	态	id	+	*	(	)	\$	Е				
		C	)	<i>s</i> 3	e1	e1	s2		e1	1				
		状态 0 1 2		e3	s4	<i>s</i> 5	e3		acc					
		2	2	<i>s</i> 3	e1	e1	s2	e2	e1	6				
		また 0 1		r4	r4	r4	r4		r4					
_		4	ŀ	s3	e1	e1	s2	e2	e1	7				
		5	5	<b>s</b> 3	e1	e1	s2		e1	8				
		6	)	e3	s4	<i>s</i> 5	e3	<i>s</i> 9	e4					
		7	7	r1	r1	<i>s</i> 5	r1	r1	r1					

r2

r3

r2

r3

8

9

r2

r2

r3

r2

r3

r2

r3

stack	symbols	input	action
0		id+)\$	T
03	id		reduce by E $ ightarrow$ id
01	E	+)\$	shift
014		)\$	e2: 多余右括号, skip

				Ac	tion			Goto
	状态	id	+	*	(	)	\$	Е
	0	<i>s</i> 3	e1	e1	s2	e2	e1	1
	1	e3	s4	<i>s</i> 5	e3	e2	acc	
	2	<i>s</i> 3	e1	e1	s2	e2	e1	6
Н	3	r4	r4	r4	r4	r4	r4	
	4	s3	e1	e1	s2	e2	e1	7
	5	<i>s</i> 3	e1	e1	s2	e2	e1	8
	6	e3	s4	<i>s</i> 5	e3	s9	e4	
	7	r1	r1	<i>s</i> 5	r1	r1	r1	
	8	r2	r2	r2	r2	r2	r2	
	9	r3	r3	r2	r3	r3	r3	

stack	symbols	input	action
0		id+)\$	shift
03	id	+)\$	reduce by $\mathtt{E}{ ightarrow}\mathtt{id}$
01	E	+)\$	shift
014	E+	)\$	e2: 多余右括号, skip
014		4	4. FA 1 FE

		Action												
状态	id	+	*	(	)	\$	Е							
0	s3	e1	e1	s2	e2	e1	1							
1	e3	s4	<i>s</i> 5	e3	e2	acc								
2	<i>s</i> 3	e1	e1	s2	e2	e1	6							
3	r4	r4	r4	r4	r4	r4								
4	s3	e1	e1	s2	e2	e1	7							
5	s3	e1	e1	s2	e2	e1	8							
6	e3	s4	<i>s</i> 5	e3	s9	e4								
7	r1	r1	<i>s</i> 5	r1	r1	r1								
8	r2	r2	r2	r2	r2	r2								
9	r3	r3	r2	r3	r3	r3								

45 1											
分析过程						Ac	tio	1		Goto	
	_	*	状态		+	*	(	)	\$	Е	
		0		<b>s</b> 3	e1	e1	Si		e1	1	
		1		e3	s <b>4</b>	<i>s</i> 5	e		acc		
		2		<b>s</b> 3	e1	e1	Si		e1	6	
		3		r4	r4	r4	r		r4		
	sto	4		<i>s</i> 3	e1	e1	Si		e1	7	
	0	5		<b>s</b> 3	e1	e1	Si		e1	8	
		6		e3	s4	<i>s</i> 5	e		e4		
	03	7		r1	r1	<i>s</i> 5	r		r1		
	01	8		r2	r2	r2	r		r2		
	01	9		r3	r3	r2	r	3 r3	r3		kip
								4 5	4 .1	** E	יף איני
	014	4	E-	+			\$		热少证		
								压id	入栈到		
	014						\$				
							\$				
							\$		ot		

分析过程													
73 417 122 75							tio	n			Goto		
	_	*	\$	id	+	*	(	(	)	\$	Е	г	
		0		<i>s</i> 3	e1	e1	S		e2	e1	1		
		1		e3	s4	<i>s</i> 5	e		e2	acc			
		2		<b>s</b> 3	e1	e1	S		e2	e1	6		
		3		r4	r4	r4	r		r4	r4			_
	sto	4		<i>s</i> 3	e1	e1	S		e2	e1	7		
	0	5		s3	e1	e1	S		e2	e1	8		П
		6		e3	s4	<i>s</i> 5	e	3	s9	e4			
	03	7		r1	r1	<i>s</i> 5		1	r1	r1			
	01	8		r2	r2	r2	r		r2	r2			
	01	9		r3	r3	r2	r	3	r3	r3		kip	, I
											** F	P	1
	014	7	E	+			\$	е	1: 37	少证	算量		
								Ī	Eid>	人栈到	状态 3		
	014	13	E	+id			\$	r	educ	e by I	E→id		
	014	17					\$	r	educ				
	01						\$	a	ccep	t			

台

	过程 二											
析过程。						Ac	tio	n			Goto	
	_	*	た	id	+	*	(		)	\$	E	
		0		<i>s</i> 3	e1	e1	S	2	e2	e1	1	
		1		e3	s4	<i>s</i> 5	е	3	e2	acc		
	3			s3	e1	e1			e2	e1	6	
		_		r4	r4	r4	r		r4	r4		
	sto	4		<i>s</i> 3	e1	e1	S		e2	e1	7	
	0	5		<b>s</b> 3	e1	e1	S		e2	e1	8	
		6		e3	s4	s5	e		<i>s</i> 9	e4		
	03	7		r1	r1	<i>s</i> 5	r		r1	r1		
	01	8		r2	r2	r2	r		r2	r2		
	01	9		r3	r3	r2	r	3	r3	r3		κip
	014	1	E	+			\$			少证		
								压id入栈到				
	014		E-	+id			\$	re	educ	e by I	E→id	
	014	17	E-	+E			\$	r	educ	e by I	E→E+E	
	01						\$	a	ccep	†		

/ \											
分析过程。						Ac	tion			Goto	
	_	**	5	id	+	*	(	)	\$	E	
		0		<i>s</i> 3	e1	e1	s2	_	e1	1	
		1		e3	s4	<i>s</i> 5	e3		acc		
		2		<i>s</i> 3	e1	e1	s2		e1	6	
		3		r4	r4	r4	r4		r4		
	ste	4		<i>s</i> 3	e1	e1	s2	_	e1	7	
	0	5		<i>s</i> 3	e1	e1	s2		e1	8	
	_	6		e3	s4	s5	e3		e4		
	03	7		r1	r1	s5	r1		r1		
	01	8		r2	r2	r2	r2		r2		
	01	9		r3	r3	r2	r3	r3	r3		۸ip
	014	4	E	+			\$	e1: #	少证	算量	
	014	43	E	+id			\$	reduc	e by 1	E→id	
	014	<b>47</b>	E	+E			\$	reduc	e by 1	E→E+E	
	01		Ε				\$	accep	†		

1517											
分析过程。						Ac	tio	n		Goto	
	_	** 3	5	id	+	*	(	( )	\$	E	
		0		<i>s</i> 3	e1	e1	S		e1	1	
		1		e3	s4	<i>s</i> 5	e		acc		
		2		<i>s</i> 3	e1	e1	S		e1	6	
		3		r4	r4	r4	r		r4		
	st	4		<i>s</i> 3	e1	e1	S			7	
	0	5		<i>s</i> 3	e1	e1	S			8	
		6		e3	s4	<i>s</i> 5	e				
	03	7		r1	r1	<i>s</i> 5	r		r1		
	01	8		r2	r2	r2	r		r2		
	01	9		r3	r3	r2	r	3 r3	r3		kip
	014	1	E	+			\$		缺少证		
										□状态 3	
	014		E	+id			\$	redu	ce by	$\mathtt{E}{ ightarrow}\mathtt{id}$	
	014	47	E	+E			\$	redu	ce by	E→E+E	
	01		Ε				\$	acce	pt		

#### 相关工具

#### 自下而上的语法分析器住成工具

将含有耐注的上下文无关文法转换以某种程序设计语言识别为输出的识别该文法语法分析器源程序。

#### Examples

- ① yacc(Yet Another Compiler-Compiler): 1975 年由贝尔实验室 Mike Lesk & Eric Schmidt 开发,UNIX 标准实用工具 (utility);
- ② byacc: Berkeley YACC: Robert Corbett, 1989 年, yacc compatible, in Free BSD distribution, DOS version in my CD-ROM;
- 3 bison: Robert Corbett & Richard Stallmen, 1988 年, yacc compatible, in Linux distribution, 最新版本: 2.4. 支持 GLR(Generalized LR) 支法, http://www.gnu.org/software/bison/.
- ◆ CUP: LALR Parser Generator in Java, current version 0.11a, 对应的词法分析器生成工具为: JFLex(http://jflex.de/), http://www2.cs.tum.edu/projects/cup/.
- 🧿 A. Holub LRpars: See CD-ROM, 支持动态显示分析过程.