第5章 自底向上的分析

一、自顶向下分析方法

二、自底向上分析方法

自底向上分析法是如何工作的?

G[S]={

S→gAhBkC

A→ab

B→cd

C→ef

}

用自底向上法分析符号串（如串gabhcdkef）

自底向上的分析

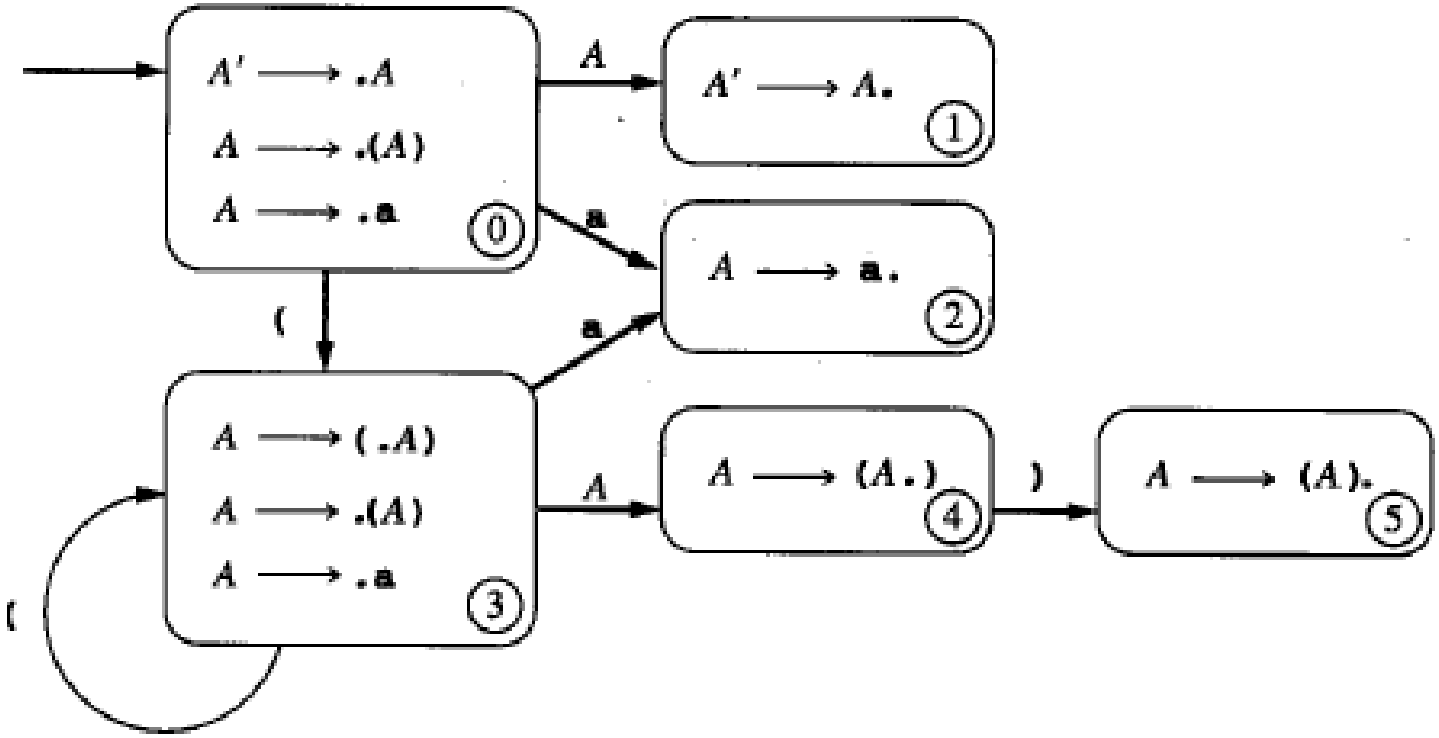
问题:

G[A’]:

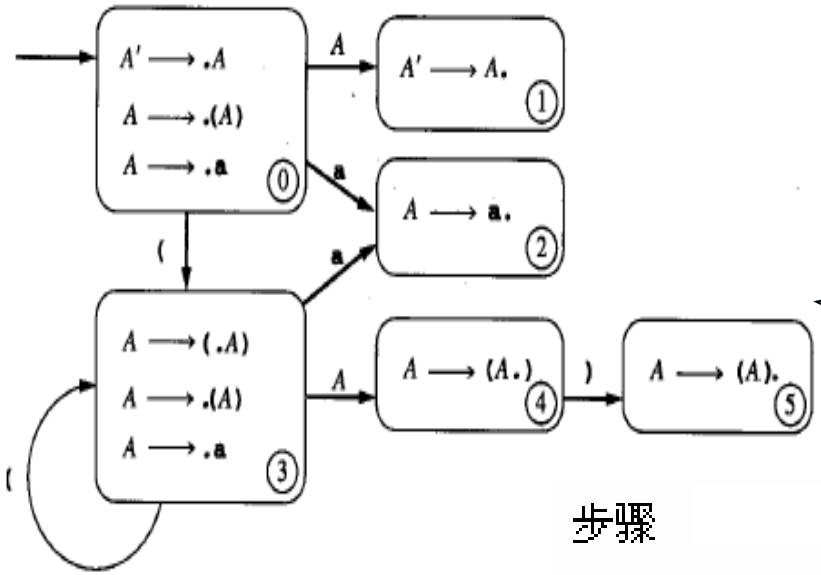
$A' \rightarrow A$

$A \rightarrow (A) \mid a$

分析串：((a))



分析串： ((a))



LR(0)分析法

步骤	分析栈	输入
1	$ 0	((a)) $
2	$ 0 (3	(a)) $
3	$ 0 (3 (3	a)) $
4	$ 0 (3 (3 a 2	)) $
5	$ 0 (3 (3 *A* 4	)) $
6	$ 0 (3 (3 *A* 4) 5	) $
7	$ 0 (3 *A* 4	) $
8	$ 0 (3 *A* 4) 5	$
9	$ 0 *A* 1	$

LR(0)分析在计算机上的实现

(1)LR(0)的存储结构

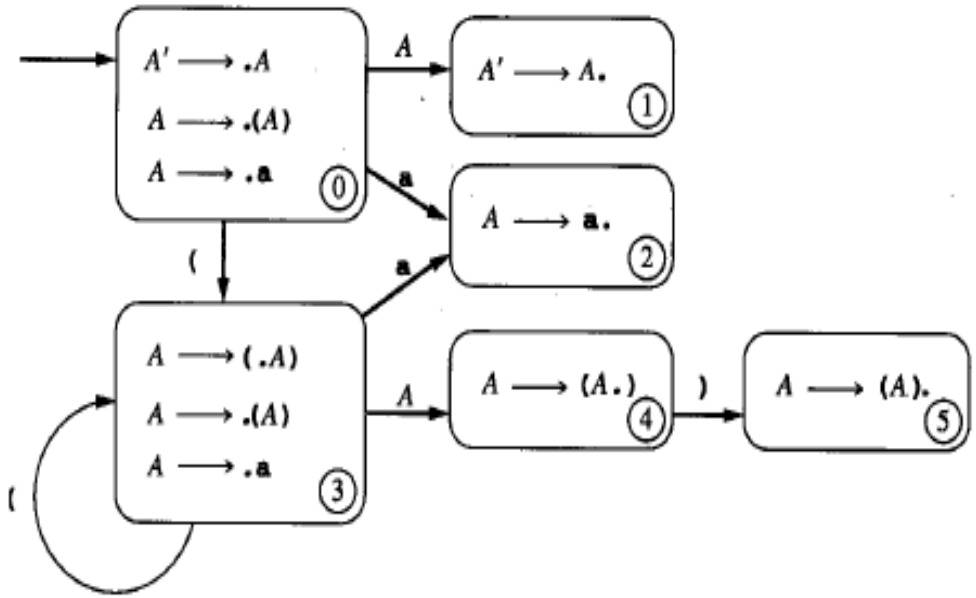
(2)LR(0)的分析算法

LR(0)项DFA的存储结构

（1）邻接矩阵

（2）链接表

（3）新二维表存储——LR(0)分析表



状　态	动　作	规　则	输　入			Goto
			(	a	)	*A*
0	移进		3	2		1
1	归约	$A' \rightarrow A$				
2	归约	$A \rightarrow a$				
3	移进		3	2		4
4	移进				5	
5	归约	$A \rightarrow (A)$				

状　态	动　作	规　则	输　入			Goto
			(	a	)	*A*
0	移进		3	2		1
1	归约	$A' \rightarrow A$				
2	归约	$A \rightarrow$ a				
3	移进		3	2		4
4	移进				5	
5	归约	$A \rightarrow (A)$				

下面讲述LR(0)分析算法是如何工作的。例子：用LR(0)分析方法分析上述文法的一串((a))。

步骤	分析栈	输入	动作
1	$ 0	((a)) $	移进
2	$ 0 (3	(a)) $	移进
3	$ 0 (3 (3	a)) $	移进
4	$ 0 (3 (3 a 2	)) $	用$A \rightarrow$ a归约
5	$ 0 (3 (3 *A* 4	)) $	移进
6	$ 0 (3 (3 *A* 4) 5	) $	用$A \rightarrow$ (*A*) 归约
7	$ 0 (3 *A* 4	) $	移进
8	$ 0 (3 *A* 4) 5	$	用$A \rightarrow$ (*A*) 归约
9	$ 0 *A* 1	$	接受

一个新例子

G[E]:

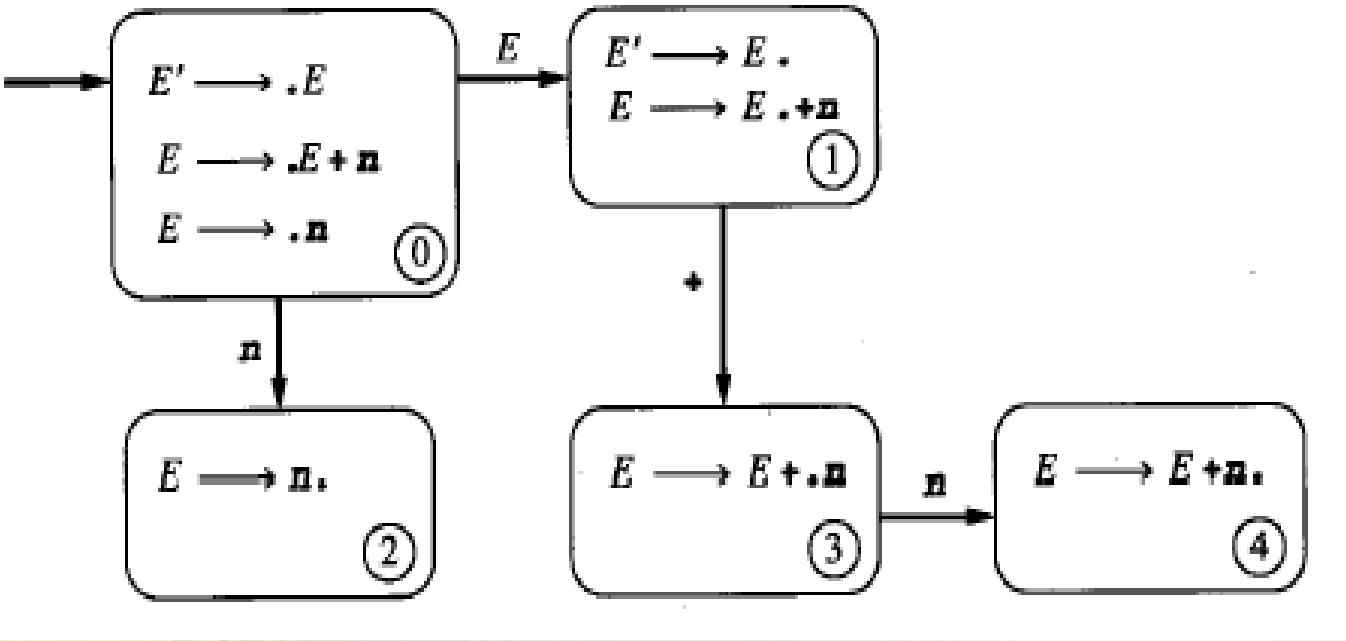
$E \rightarrow E+n \mid n$

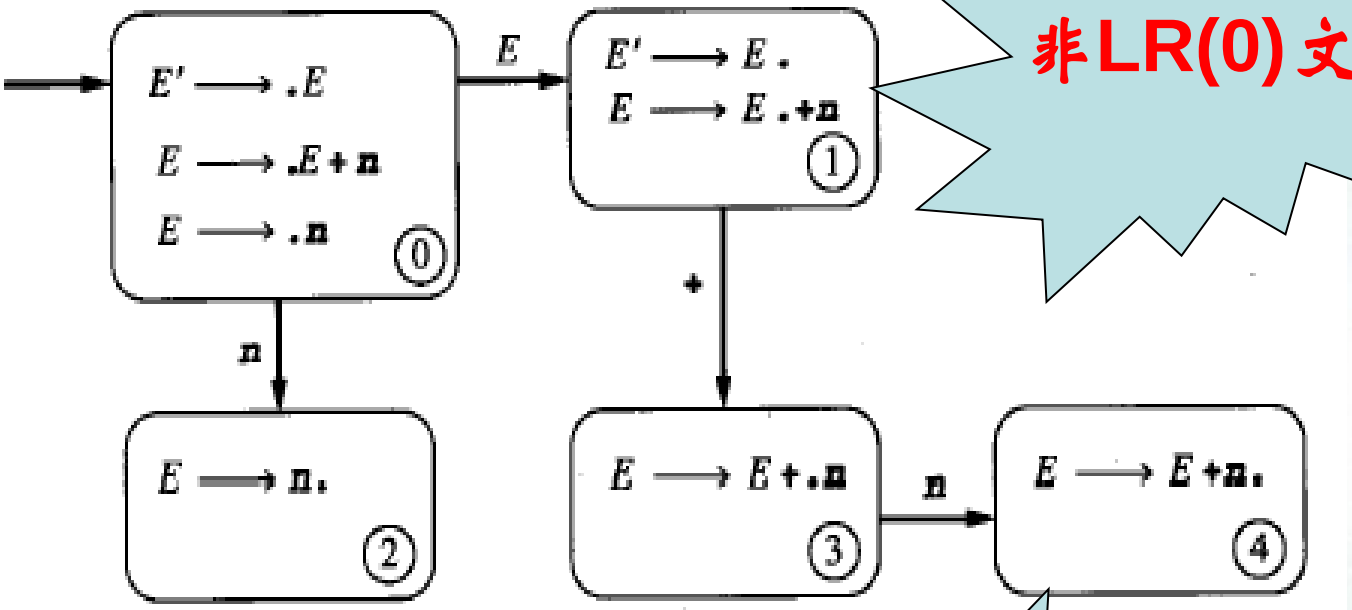
试用自底向上分析方法分析输入串:n+n+n

(1)文法的扩充

E’→E

E→E+n | n


E' ⟶ .E
E ⟶ .E + n
E ⟶ .n
0
E
E' ⟶ E .
E ⟶ E .+n
1
n
+
E ⟶ n.
2
E ⟶ E + .n
3
n
E ⟶ E +n.
4



非LR(0)文法

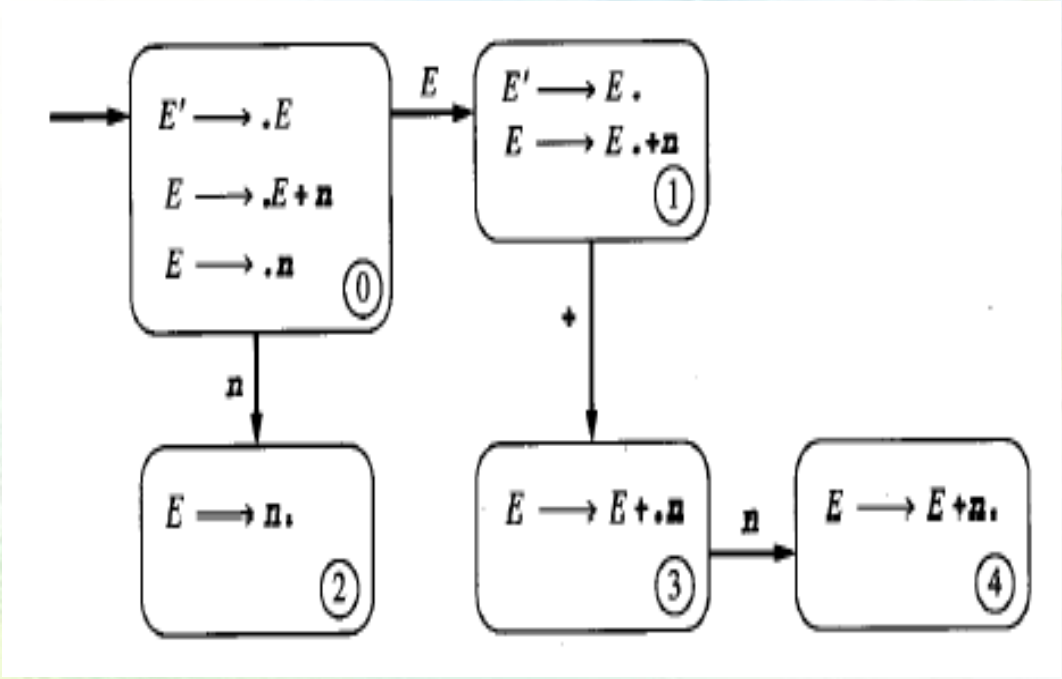
输入串：n+n+n

无法分析下去！

如何解决？

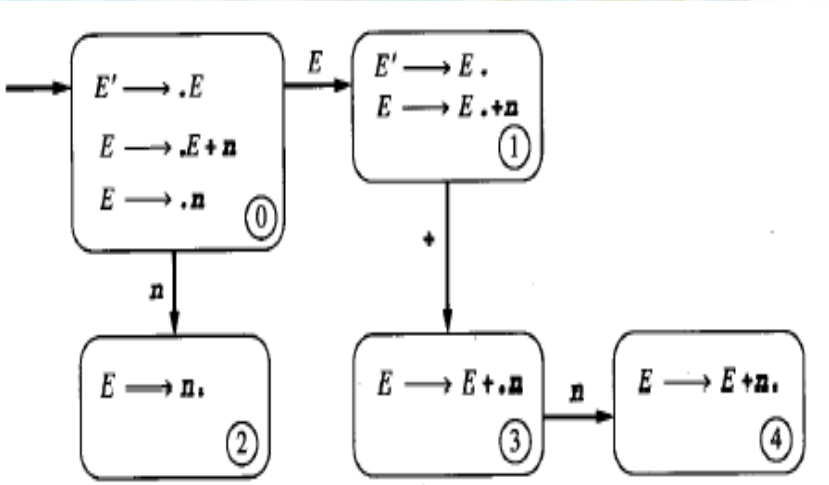
SLR(1)分析

- 简单LR(1)分析
- SLR(1)分析还是使用LR(0)项目集DFA。
- 只是在构造分析表时才考虑超前查看的符号。



- 重点考察状态1
- Follow(*E'*) = {$}
- E→E.+n 分隔符后的符号是终结符号+,因此在状态1时,输入串当前符号为+则移进,而$则规约.

- SLR(1)分析表如下表5-5：



状　态	输　入			Goto
	n	+	$	E
0	s2			1
1		s3	接受	
2		r(E→n)	r(E→n)	
3	s4			
4		r(E→E+n)	r(E→E+n)	

注意：移进由表项中的字母s指出，归约由字母r指出。

状　态	输　入			Goto
	n	+	$	*E*
0	s2			1
1		s3	接受	
2		r(*E*→n)	r(*E*→n)	
3	s4			
4		r(*E*→*E*+n)	r(*E*→*E*+n)	

步骤	分析栈	输入	动作
1	$ 0	n + n + n $	移进2
2	$ 0 n 2	+ n + n $	用*E*→n归约
3	$ 0 *E* 1	+ n + n $	移进3
4	$ 0 *E* 1 + 3	n + n $	移进4
5	$ 0 *E* 1 + 3 n 4	+ n $	用*E*→*E* + n 归约
6	$ 0 *E* 1	+ n $	移进3
7	$ 0 *E* 1 + 3	n $	移进4
8	$ 0 *E* 1 + 3 n 4	$	用*E*→*E* + n 归约
9	$ 0 *E* 1	$	接受

SLR(1)分析法的新问题1

例子：

if语句的文法：

statement → *if-stmt* | *other*

if-stmt → if (*exp*) *statement*| if (*exp*)
statement else *statement*

exp → 0 | 1

试用SLR(1)分析方法进行输入串的分析。

- 步骤：

（1）为了处理上的简单，我们先做文法的简化

$S \rightarrow I \mid other$

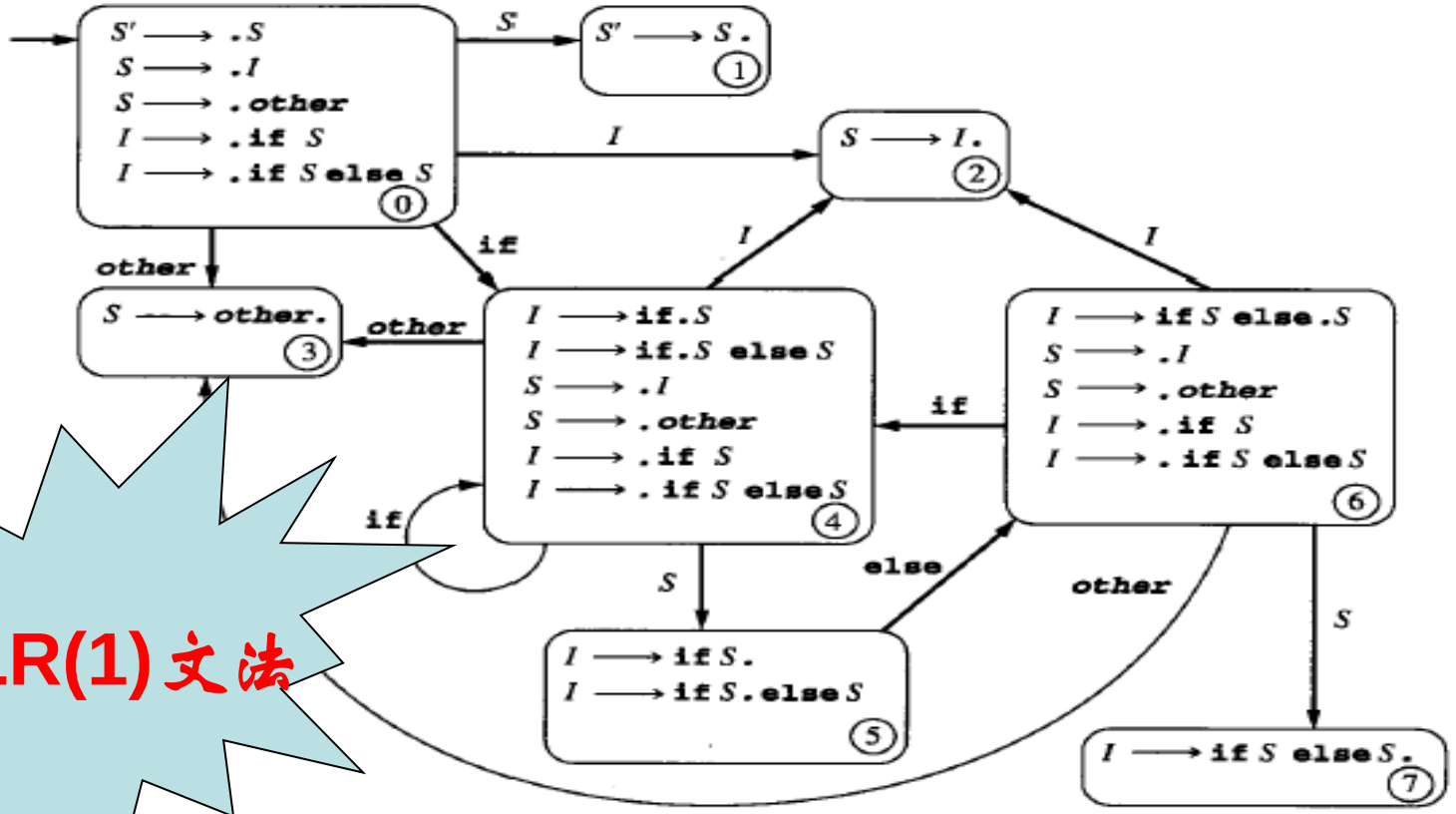
$I \rightarrow if\ S \mid if\ S\ else\ S$

（2）画出LR(0)项DFA图

S' → S

S → I | other

I → if S | if S else S



非SLR(1)文法

- 状态5： 对于规约项： Follow (I) = {$, else }
- 对于移进项的下一个符号为 else
- 问题：移进-归约冲突
- 解决方法:只做移进不做归约
- ——消除二义性的新方法

为了在SLR(1)分析表中描述方便，我们对文法规则进行编号：

(1) $S \rightarrow I$　　(2) $S \rightarrow other$

(3) $I \rightarrow$ if S　　(4) $I \rightarrow$ if S else S

[该分析表已删除了分析冲突部分]

状　态	输　入				Goto	
	if	Else	other	$	S	I
0	s4		s3		1	2
1				接受		
2		r1		r1		
3		r2		r2		
4	s4		s3		5	2
5		s6		r3		
6	s4		s3		7	2
7		r4		r4		

SLR(1)分析法的新问题2

文法：

stmt → call-stmt | assign-stmt

call-stmt → identifier

assign-stmt →var := exp

var → var [exp] | identifier

exp → var | number

试采用SLR(1)分析方法进行分析。

- 步骤：

- （1）先简化文法：

  S→ id | V := E

  V→ id

  E→ V | n

(2)文法的扩充：

  S’ →S

  S→id

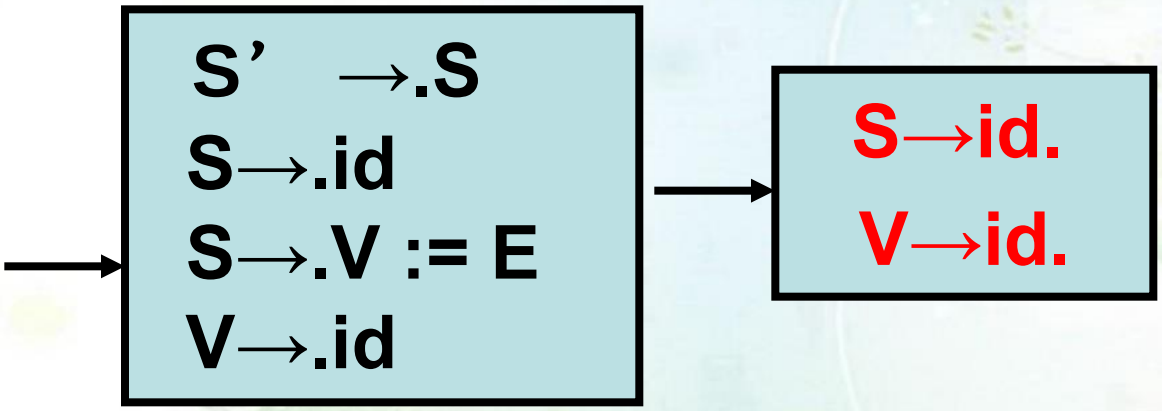
  S→V := E

  V→id

  E→V

  E→n

(3)构造DFA



Follow (S)={$} 和 Follow (V)={:=, $ }

如何解决？ 问题的根源在哪里？

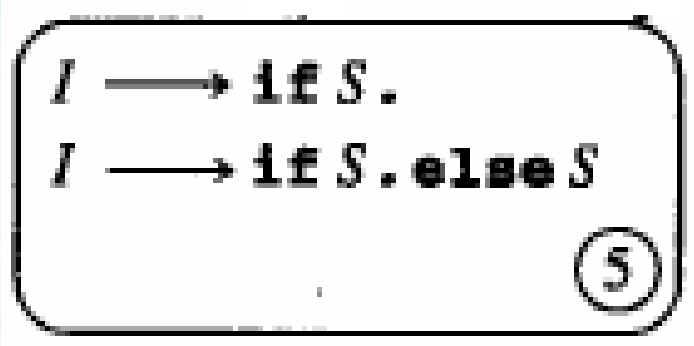
SLR(1)分析法的缺陷在于构造LR(0)项DFA时不考虑先行符号，而在构造分析表的时候才加以考虑。

总 结

(1)

S → I | other

I → if S | if S else S



Follow (*I*) = {$, else }

(2)

S→ id | V := E

V→ id

E→ V | n

S→id.

V→id.

SLR(1)文法(SLR(1) grammar)

- 当且仅当对于任何状态s，以下的两个条件：
- 1) 对于在s中的任何项目$A\rightarrow\alpha.X\beta$，当$X$是一个终结符，且$X$在Follow ($B$) 中时，$s$中没有完整的项目$B\rightarrow\gamma.$。 [移进-归约冲突]
- 2) 对于在s中的任何两个完整项目$A\rightarrow\alpha.$和$B\rightarrow\beta.$，Follow(A)∩ Follow(B)为空。[归约-归约冲突]

  均满足时，文法为SLR(1)文法。

SLR(1)分析法的总结

- SLR(1)分析是LR(0)分析的一个简单但有效的扩展，
- SLR(1)分析的能力足以处理几乎所有实际的语言结构。
- 某些情况， SLR(1)分析能力并不太强

解决方法一

- 超前查看K个符号——SLR(k)文法
- 当$k>1$时，SLR(k)分析比SLR(1)分析更强大，但由于分析表的大小将按k的指数倍增长，所以它又要复杂许多。
- 上例不是简单SLR(1)文法而是SLR(2)文法。

解决方法二

在构造DFA的时候就超前考虑一个符号。

$S \rightarrow id \mid V := E$

$V \rightarrow id$

$E \rightarrow V \mid n$

解决方法二

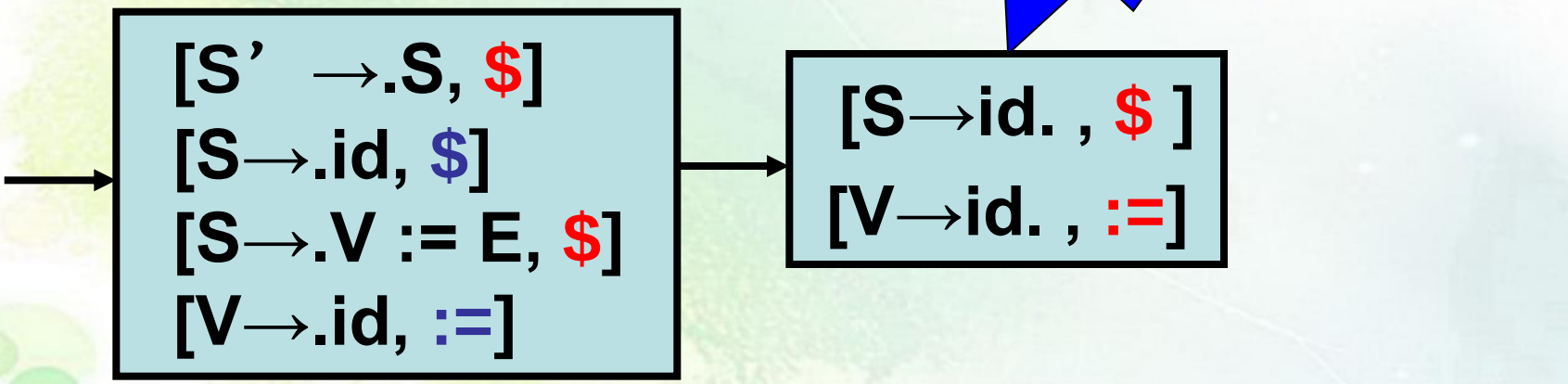
在构造DFA的时候就超前考虑一个符号。

S→ id | V := E

V→ id

E→ V | n

LR(1)分析法

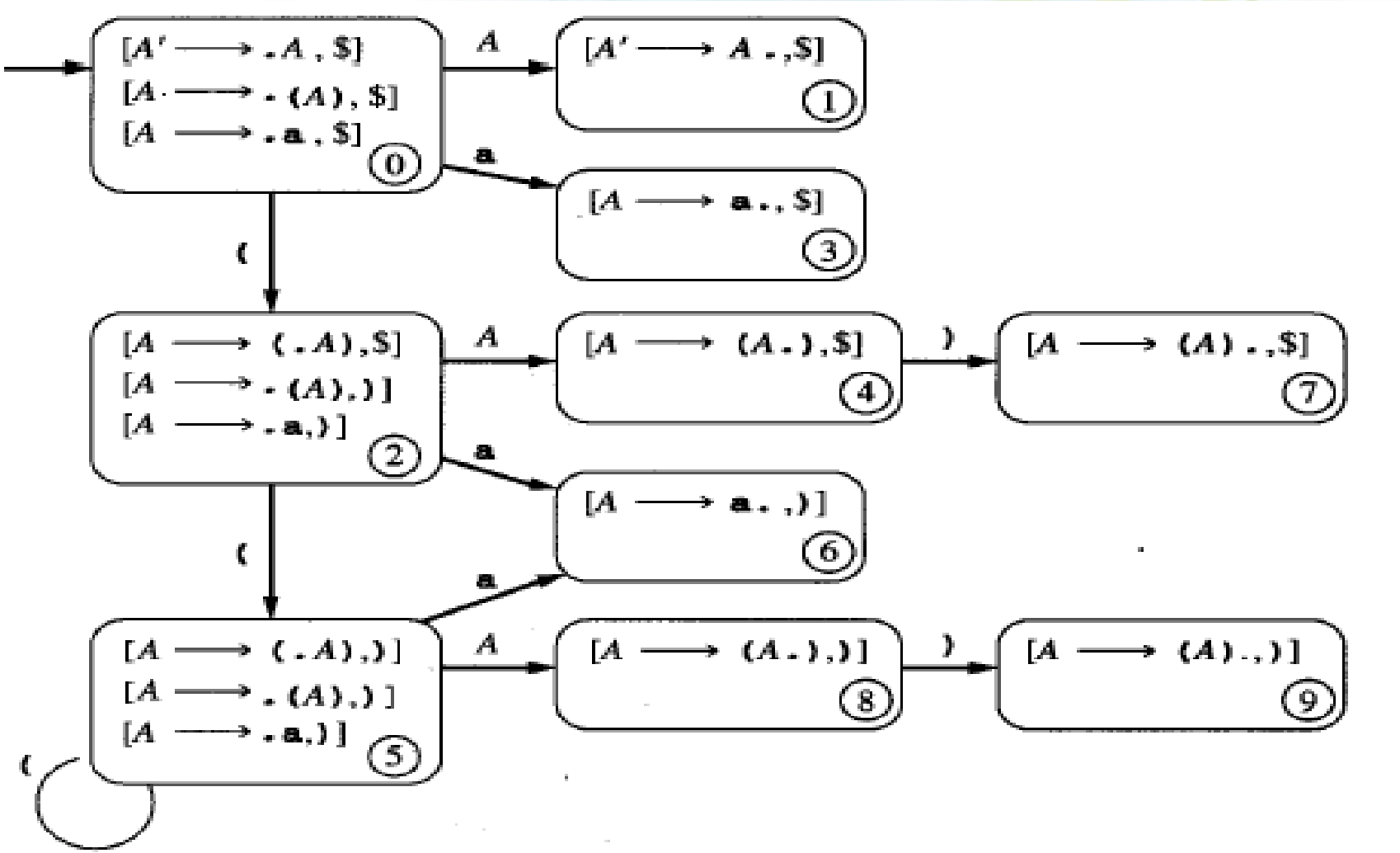


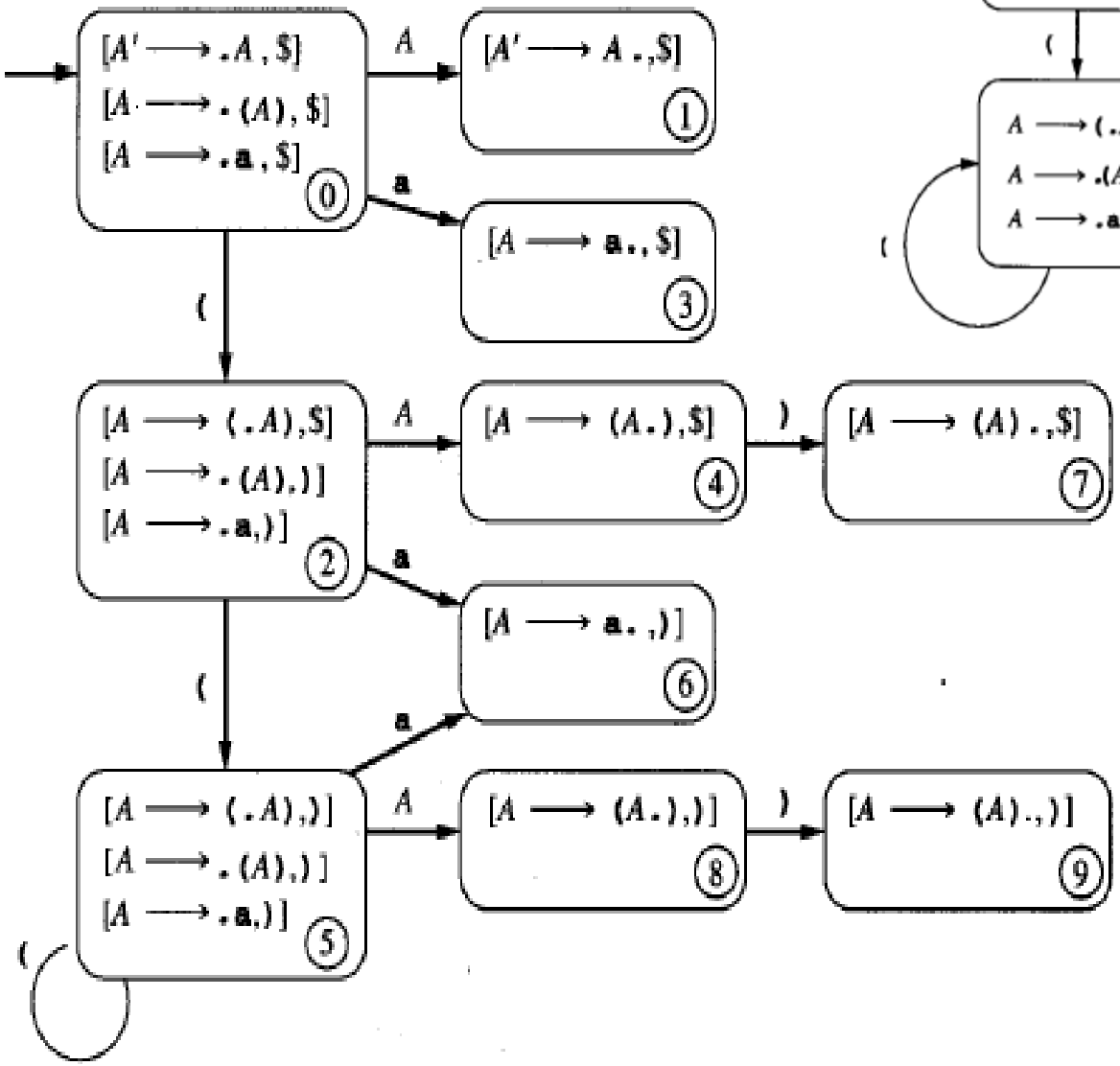
LR(1)分析法之问题发现

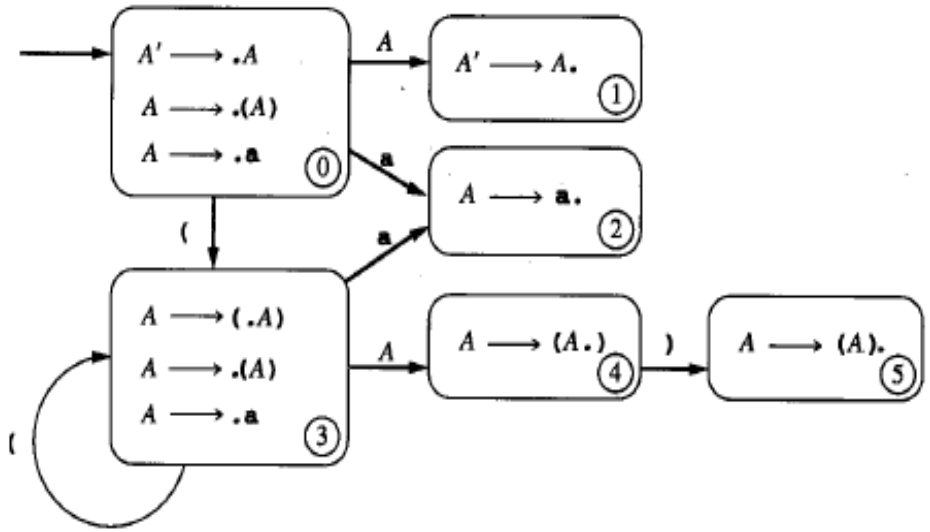
文法：　$A \rightarrow (A) \mid a$

- (1)首先改写为扩充文法

$A' \rightarrow A$　　$A \rightarrow (A) \mid a$

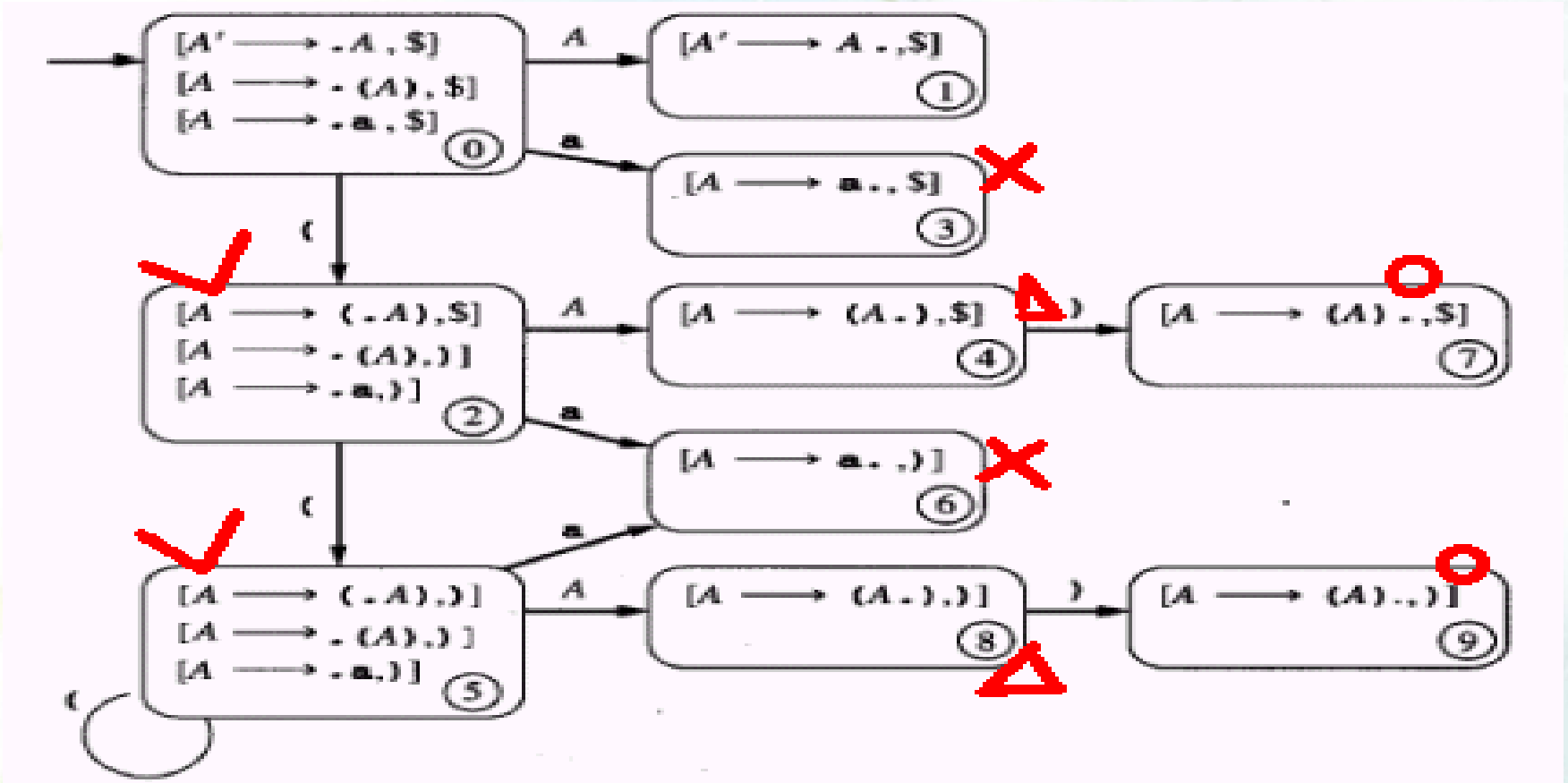

[A' → .A , $]
[A → .(A), $]
[A → .a , $]
0
A
[A' → A .,$]
1
a
[A → a .,$]
3
(
[A → (.A),$]
[A → .(A),)]
[A → .a,)]
2
A
[A → (A.),$]
4
)
[A → (A) .,$]
7
a
[A → a.,)]
6
(
a
[A → (.A),)]
[A → .(A),)]
[A → .a,)]
5
A
[A → (A.),)]
8
)
[A → (A).,)]
9
(


[A' ⟶ .A , $]
[A ⟶ .(A), $]
[A ⟶ .a , $]
0
A
[A' ⟶ A .,$]
1
a
[A ⟶ a., $]
3
(
[A ⟶ (.A),$]
[A ⟶ .(A),)]
[A ⟶ .a,)]
2
A
[A ⟶ (A.),$]
4
)
[A ⟶ (A) .,$]
7
a
[A ⟶ a. ,)]
6
(
a
[A ⟶ (.A),)]
[A ⟶ .(A),)]
[A ⟶ .a,)]
5
A
[A ⟶ (A.),)]
8
)
[A ⟶ (A).,)]
9


A' ⟶ .A
A ⟶ .(A)
A ⟶ .a
0
A
A' ⟶ A.
1
a
A ⟶ a.
2
(
A ⟶ (.A)
A ⟶ .(A)
A ⟶ .a
3
a
A
A ⟶ (A.)
4
)
A ⟶ (A).
5
(

对比效果

LR(1)分析法之问题



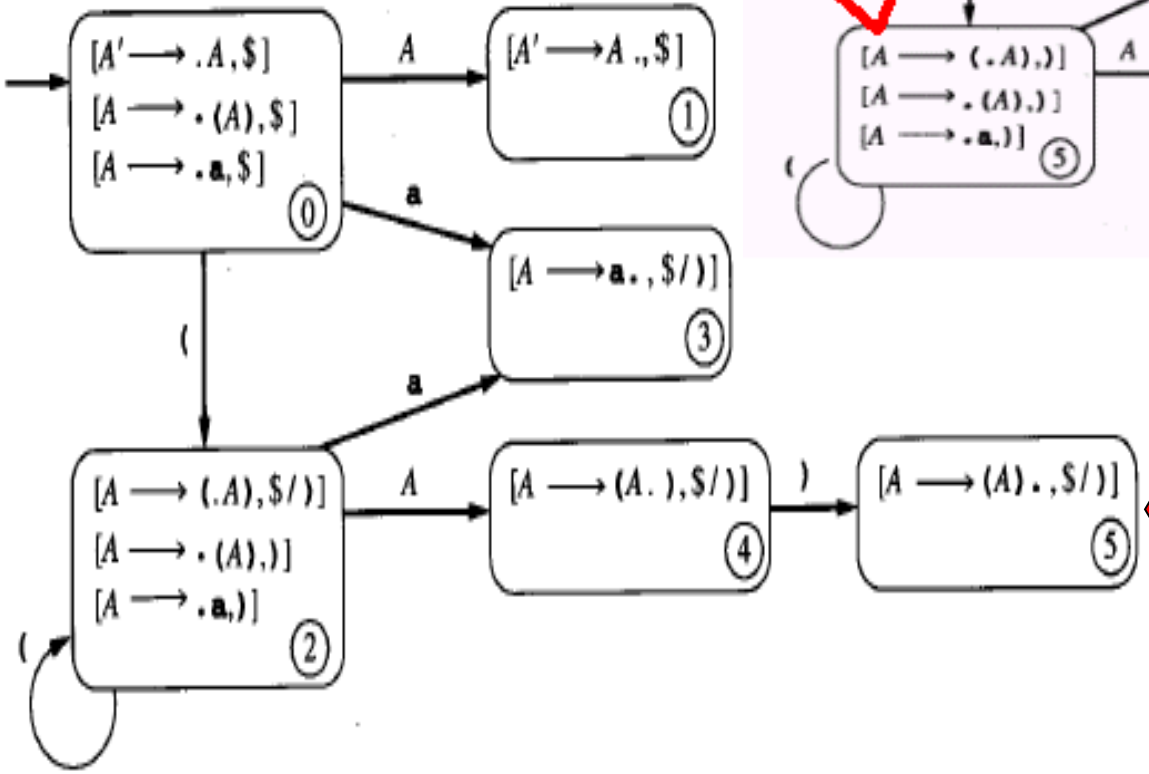
状态2和5、状态4和8、状态7和9、状态3和6，只是先行的符号不一样,而核心是一样的。

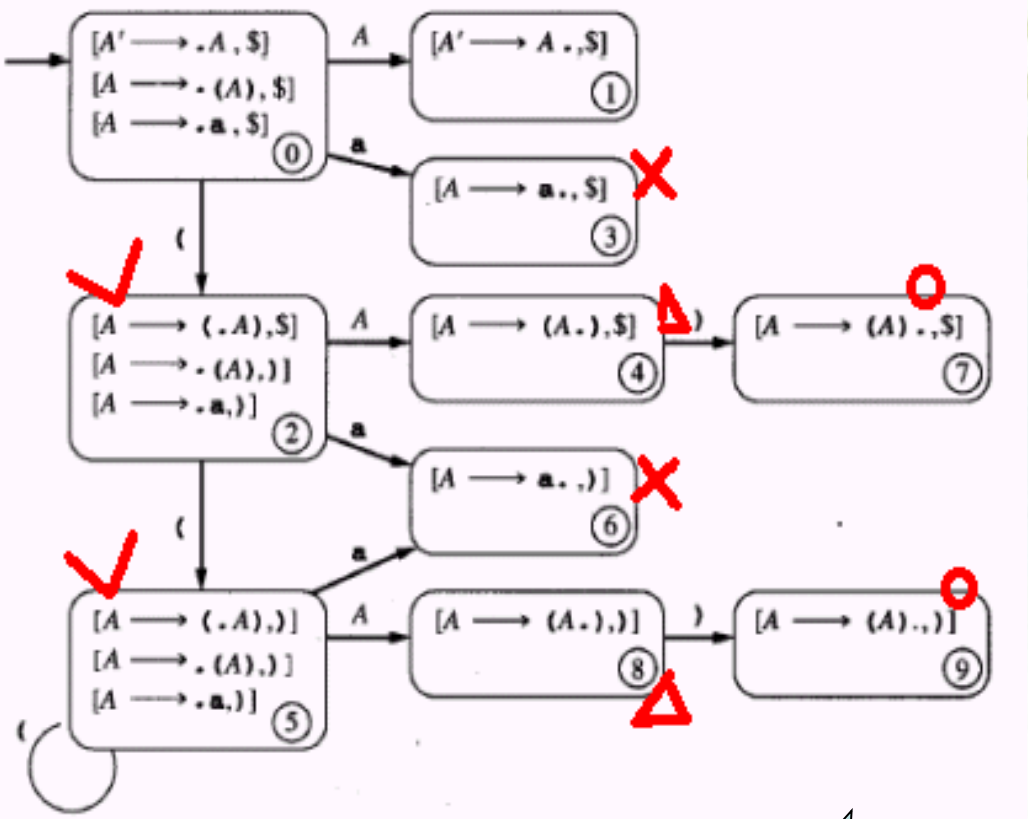
解决方法

- 压缩状态数——数目与LR(0)相仿
- 一个概念：
- 状态的核心(core)是由状态中的所有LR(1)项目的第1个成分组成的LR(0)项目的集合。
- 压缩方法：

——核心一样就合并，先行符号就做并集

合并结果





LALR(1)项DFA

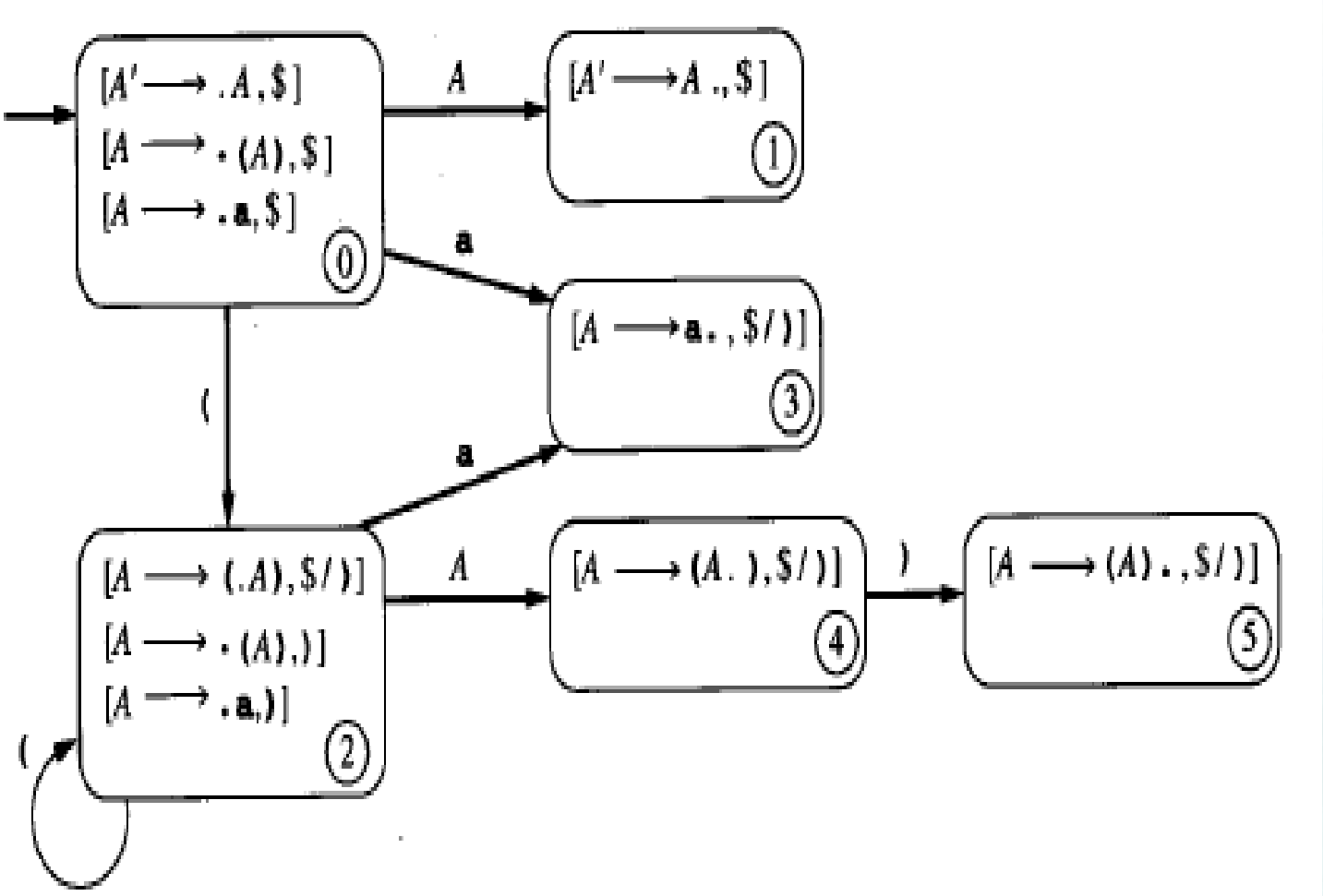
LALR(1)分析过程的步骤

- (1)先构造LALR(1)DFA，
- (2)再构造出LALR(1)分析表，
- (3)最后利用LALR(1)分析表对所要分析的符号串进行分析。

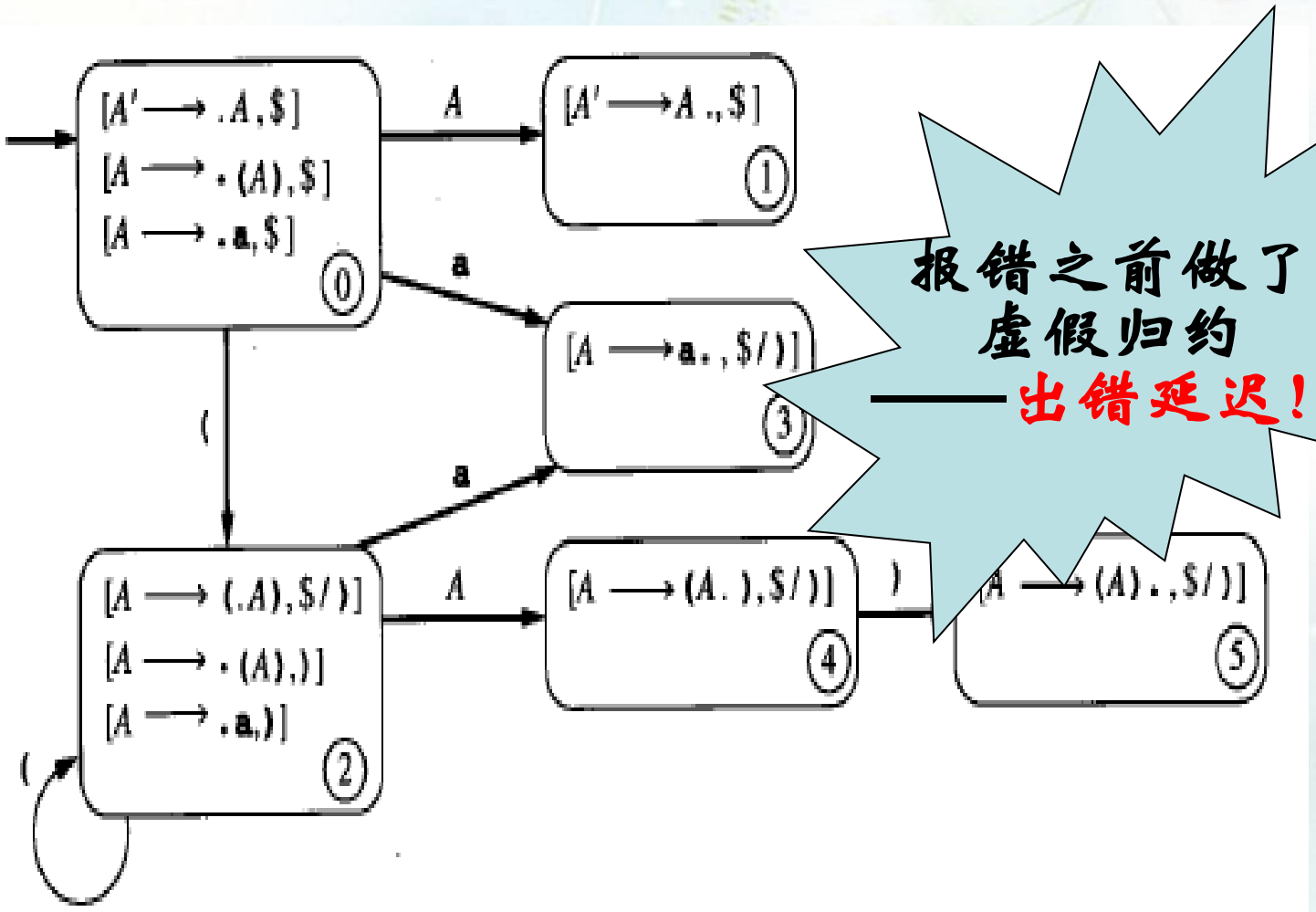
- 例子： 使用LALR(1)分析方法分析符号串(a)

[只用LALR(1)DFA进行分析，而利用LALR(1)分析表的方法请自行完成。]

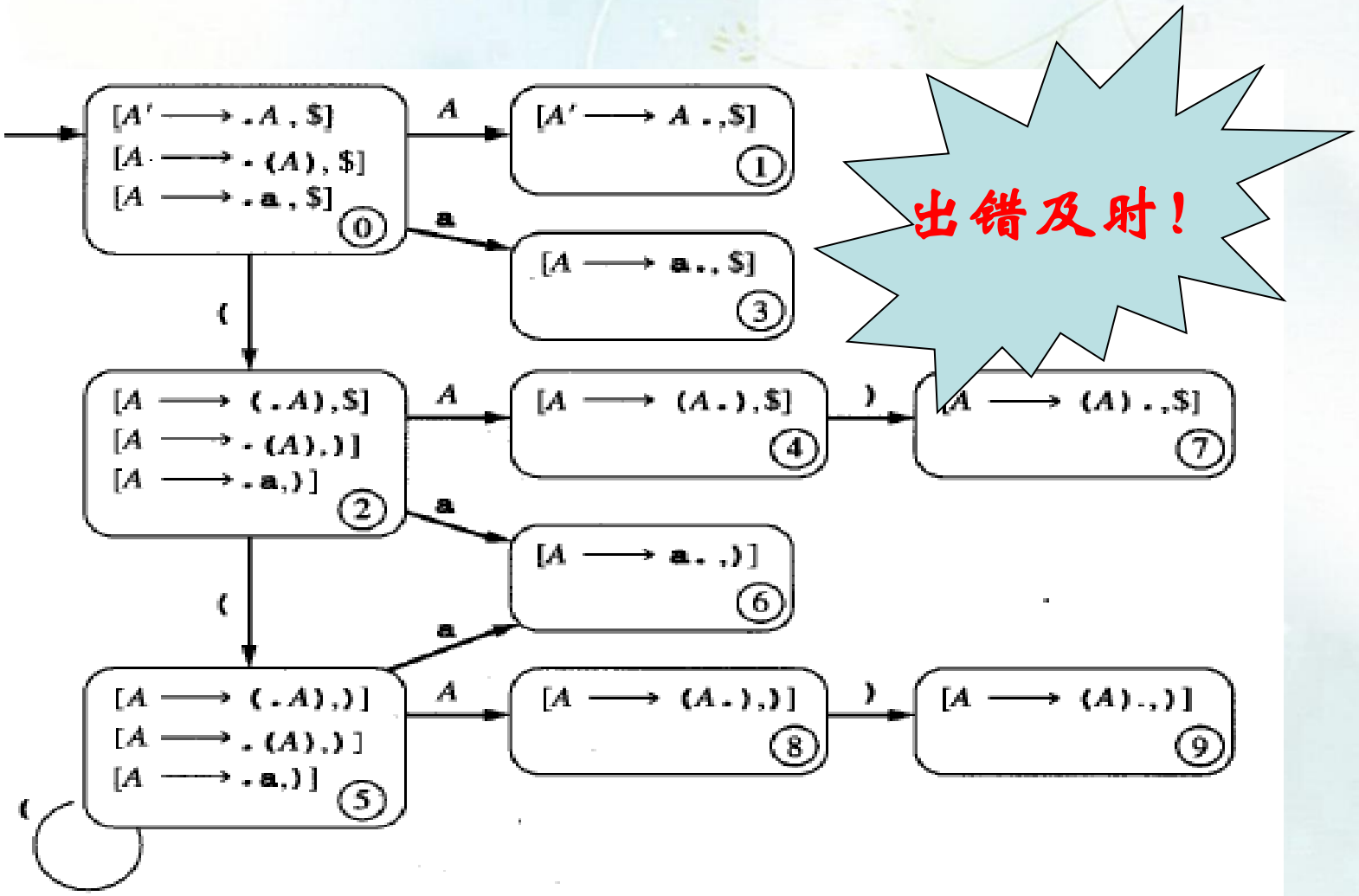
符号串：(a)



用LALR(1)分析方法分析符号串 a)



用LR(1)分析方法分析符号串 a)



自底向上分析方法

- LR(0)分析
- SLR(1) 分析
- LR(1)分析
- LALR(1)分析