CS30017 编译

第五讲：自底向上解析

徐 辉

xuh@fudan.edu.cn

主要内容

一、问题定义

二、SLR文法

三、LR(1)文法

四、LALR文法

一、问题定义

自底向上解析

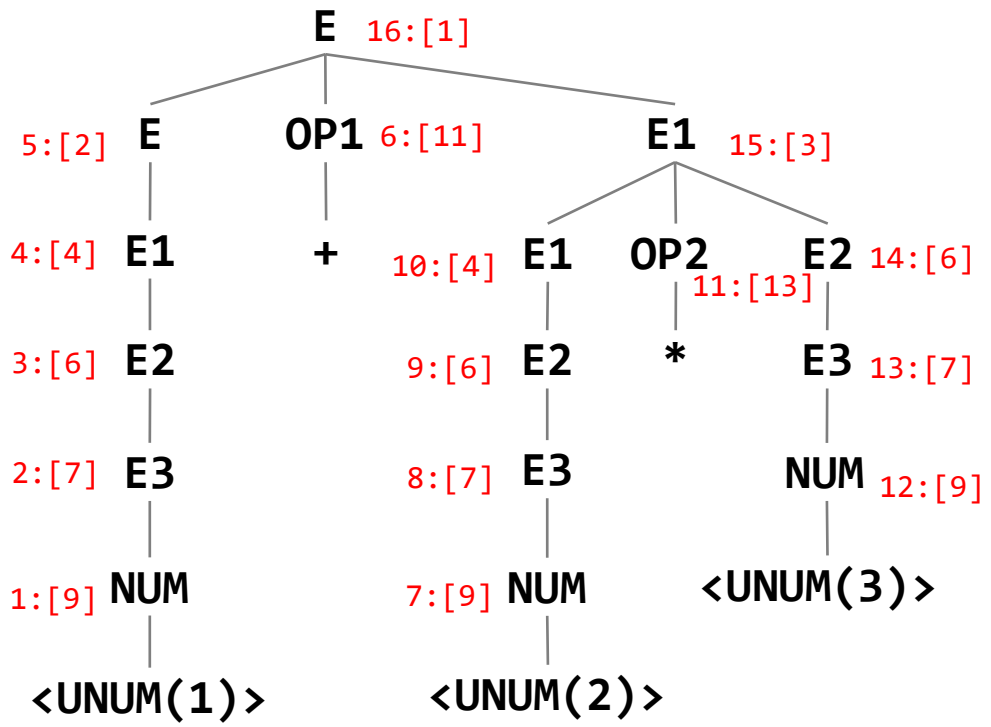
- 已知一套CFG语法规则和待解析的句子
- 从句子开始（自左至右）逐步应用规则合并规约
- 两种基本操作：
  - 移进：读入下一个字符
  - 规约：应用语法规则规约已读入字符
- 解析成功：将整个句子规约为语法规则的开始符号
- 如无二义性问题，则规约方式唯一

自底向上解析示例

语法规则：

```
[1] E → E OP1 E1
[2] E → E1
[3] E1 → E1 OP2 E2
[4] E1 → E2
[5] E2 → E3 OP3 E2
[6] E2 → E3
[7] E3 → NUM
[8] E3 → '(' E ')'
[9] NUM → <UNUM>
[10] NUM → '-' <UNUM>
[11] OP1 → '+'
[12] OP1 → '-'
[13] OP2 → '*'
[14] OP2 → '/'
[15] OP3 → '^'
```

词元流： <UNUM(1)>'+'<UNUM(2)>'*'<UNUM(3)>



挑战：如何选取恰当操作

- 可能存在多种选择：
  - 移进（shift）或规约（reduce）
  - 多种规约方式

```
s[0] = <UNUM(1)> ◦ '+' <UNUM(2)> '*' <UNUM(3)>
```

当前位置

步骤	方式一	方式二	方式三	方式四
6		E → E ◦ OP1 E1	结束	
5		E → E1 ◦		E1 → E1 ◦ OP2 E2
4		E1 → E2 ◦		
3	E2 → E3 ◦ OP3 E2	E2 → E3 ◦		
2	E3 → NUM ◦			
1	NUM → <UNUM> ◦			

二、SLR文法

SLR文法

- Simple Left-to-Right，Rightmost，前瞻一个字符
- 基本要求：同一个状态只有一种可选操作
  - 不存在既可移进，又可规约的情况
  - 同一个状态不能存在两个规约选项

语法增强：加入辅助规则

- 辅助规则：加入一条初始规则S->E
- 解析成功：句柄为E∘，且下一个字符是结束符<eof>
- 句柄（handle）：产生式的右部可规约的部分

```
[1] E → ∘ E OP1 E1
[1] E → E ∘ OP1 E1
[1] E → E OP1 ∘ E1
[1] E → E OP1 E1 ∘
[2] E → ∘ E1
[2] E → E1 ∘
[3] E1 → ∘ E1 OP2 E2
[3] E1 → E1 ∘ OP2 E2
[3] E1 → E1 OP2 ∘ E2
[3] E1 → E1 OP2 E2 ∘
...
```

语法增强

```
[0] S → ∘ E
[0] S → E ∘
[1] E → ∘ E OP1 E1
[1] E → E ∘ OP1 E1
[1] E → E OP1 ∘ E1
[1] E → E OP1 E1 ∘
[2] E → ∘ E1
[2] E → E1 ∘
[3] E1 → ∘ E1 OP2 E2
[3] E1 → E1 ∘ OP2 E2
[3] E1 → E1 OP2 ∘ E2
[3] E1 → E1 OP2 E2 ∘
...
```

构建LR(0)有穷自动机：项目集规范族

```
[0] S → ∘ E
[1] E → ∘ E OP1 E1
[2] E → ∘ E1
...
```

分析项目集

While (Q has changed) //仅包含当前项目
  for each item $[A \to \beta \circ C\delta] \in Q$
    for each production $[C \to \lambda] \in G$
      if $[C \to \circ \lambda] \notin Q$
        $Q \leftarrow Q \cup [C \to \circ \lambda]$

应用

```
[0] S → E
[1] E → E OP1 E1
[2] E → E1
[3] E1 → E1 OP2 E2
[4] E1 → E2
[5] E2 → E3 OP3 E2
[6] E2 → E3
[7] E3 → NUM
[8] E3 → '(' E ')'
[9] NUM → <UNUM>
[10] NUM → '-' <UNUM>
[11] OP1 → '+'
[12] OP1 → '-'
[13] OP2 → '*'
[14] OP2 → '/'
[15] OP3 → '^'
```

分析项目集

S0

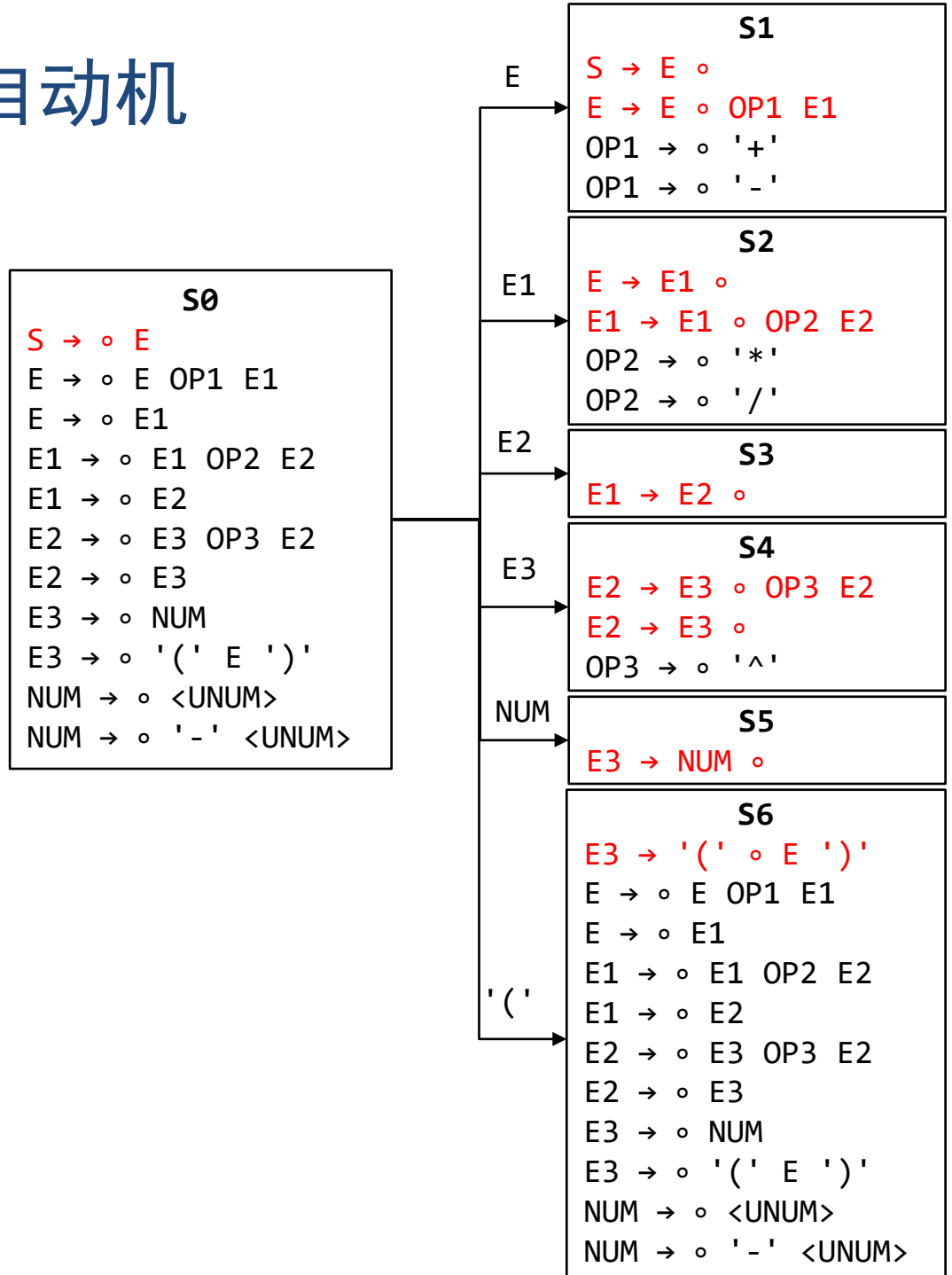
```
S → ∘ E
E → ∘ E OP1 E1
E → ∘ E1
E1 → ∘ E1 OP2 E2
E1 → ∘ E2
E2 → ∘ E3 OP3 E2
E2 → ∘ E3
E3 → ∘ NUM
E3 → ∘ '(' E ')'
NUM → ∘ <UNUM>
NUM → ∘ '-' <UNUM>
```

核心项

非核心项

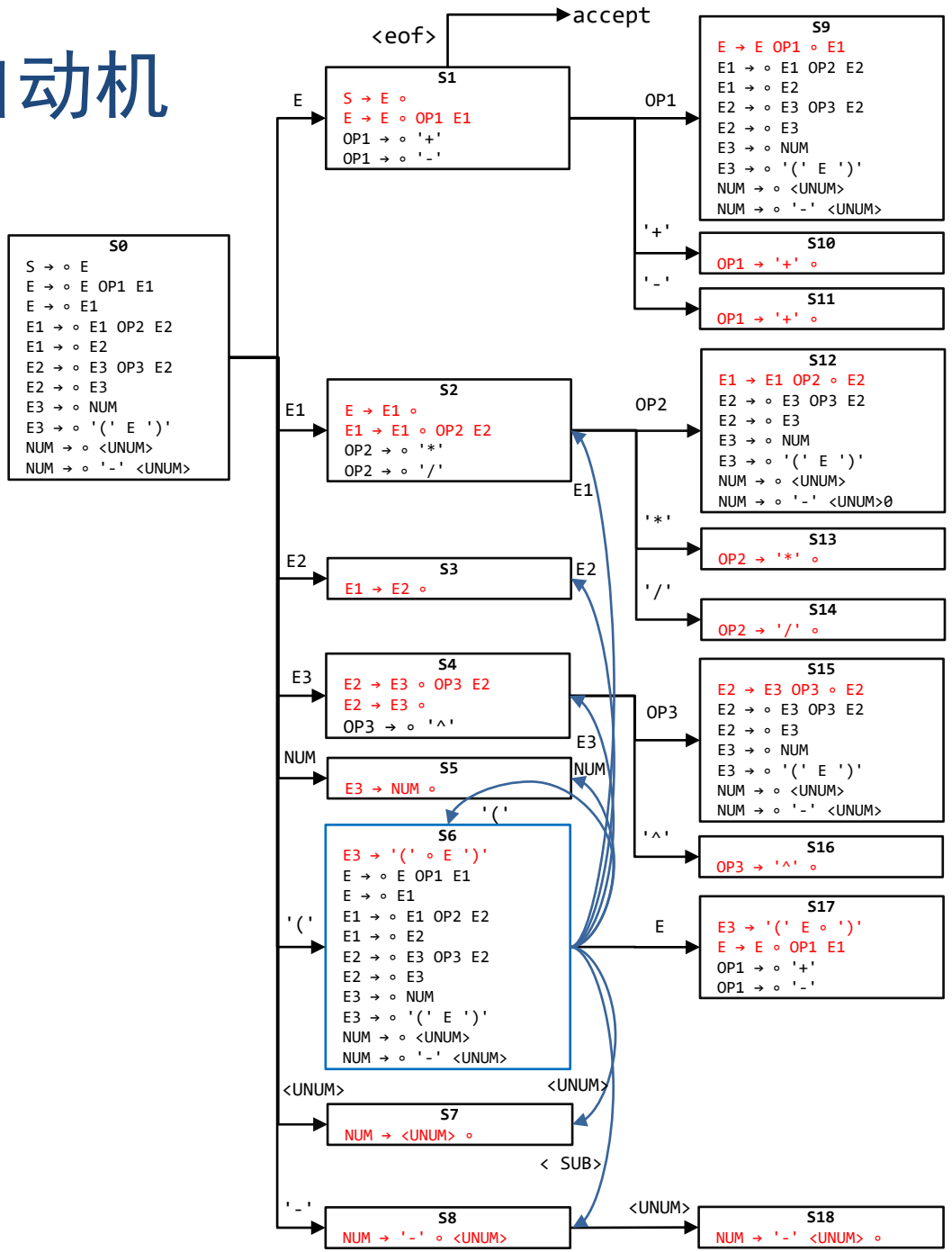
构建LR(0)有穷自动机

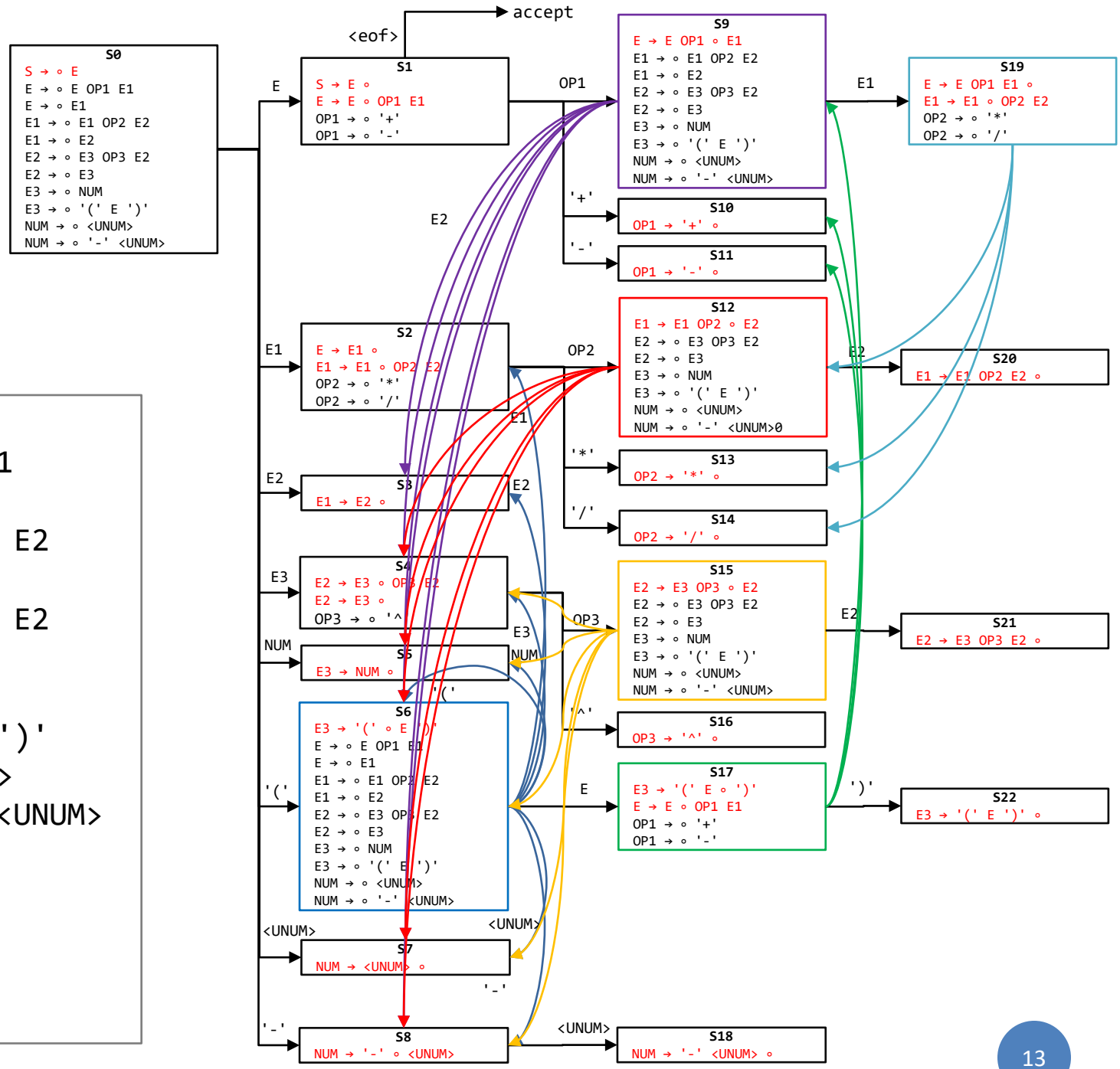
```
[0] S → ∘ E
[1] E → E OP1 E1
[2] E → E1
[3] E1 → E1 OP2 E2
[4] E1 → E2
[5] E2 → E3 OP3 E2
[6] E2 → E3
[7] E3 → NUM
[8] E3 → '(' E ')'
[9] NUM → <UNUM>
[10] NUM → '-' <UNUM>
[11] OP1 → '+'
[12] OP1 → '-'
[13] OP2 → '*'
[14] OP2 → '/'
[15] OP3 → '^'
```



构建LR(0)有穷自动机

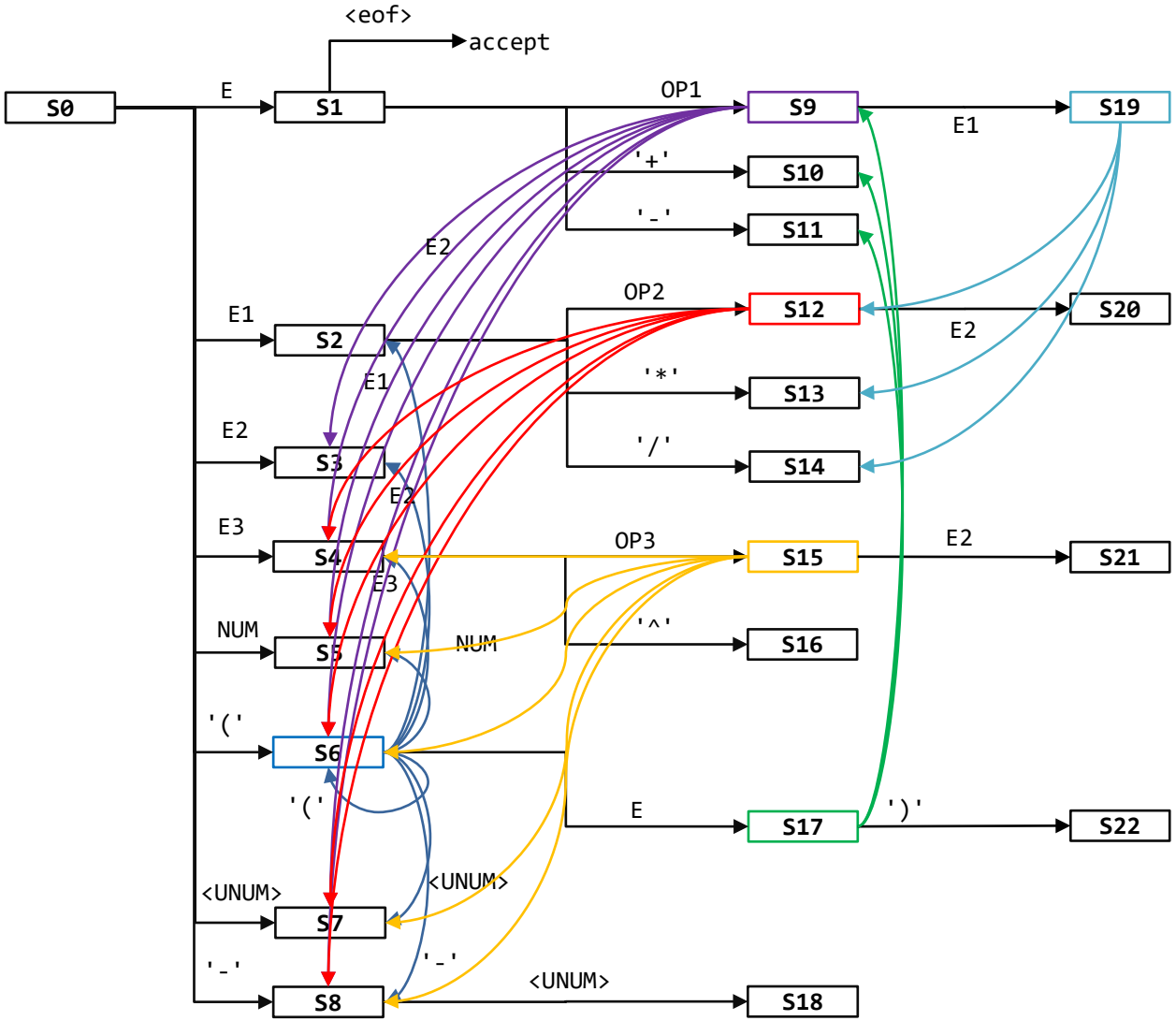
[0] S → ∘ E
[1] E → E OP1 E1
[2] E → E1
[3] E1 → E1 OP2 E2
[4] E1 → E2
[5] E2 → E3 OP3 E2
[6] E2 → E3
[7] E3 → NUM
[8] E3 → '(' E ')'
[9] NUM → <UNUM>
[10] NUM → '-' <UNUM>
[11] OP1 → '+'
[12] OP1 → '-'
[13] OP2 → '*'
[14] OP2 → '/'
[15] OP3 → '^'





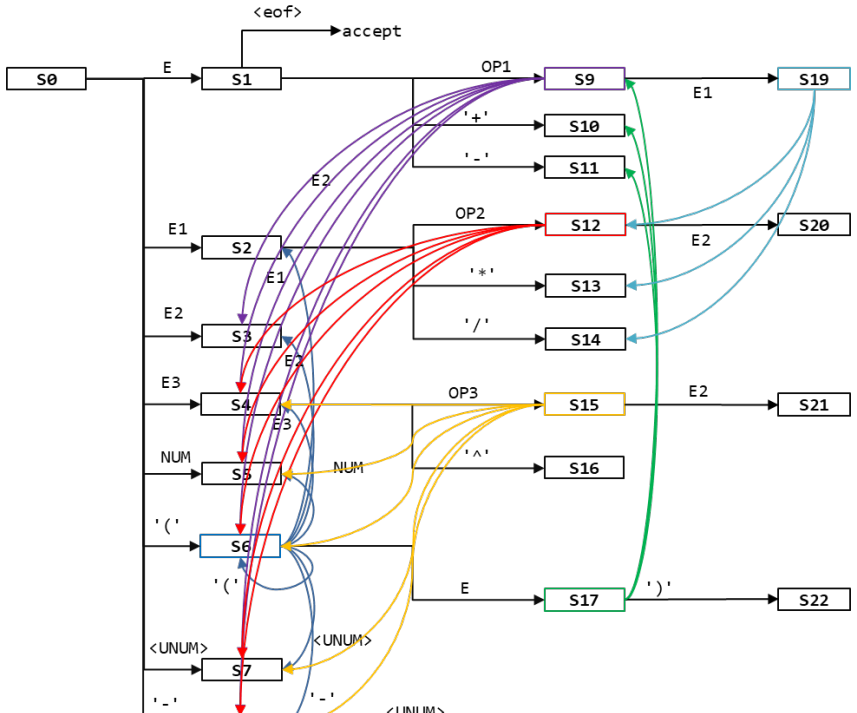
[0] S → ∘ E
[1] E → E OP1 E1
[2] E → E1
[3] E1 → E1 OP2 E2
[4] E1 → E2
[5] E2 → E3 OP3 E2
[6] E2 → E3
[7] E3 → NUM
[8] E3 → '(' E ')'
[9] NUM → <UNUM>
[10] NUM → '-' <UNUM>
[11] OP1 → '+'
[12] OP1 → '-'
[13] OP2 → '*'
[14] OP2 → '/'
[15] OP3 → '^'

LR(0)有穷自动机：状态转移关系



LR(0)有穷自动机的状态转移关系表

规范族	E	E1	E2	E3	OP1	OP2	OP3	NUM	<UNUM>	'+'	'-'	'*'	'/'	'^'	<LP>	<RP>	<eof>
S0	S1	S2	S3	S4				S5	S7		S8				S6		
S1					S9					S10	S11						accept
S2						S12						S13	S14				
S3																	
S4							S15							S16			
S5																	
S6	S17	S2	S3	S4				S5	S7		S8				S6		
...																	
S22																	

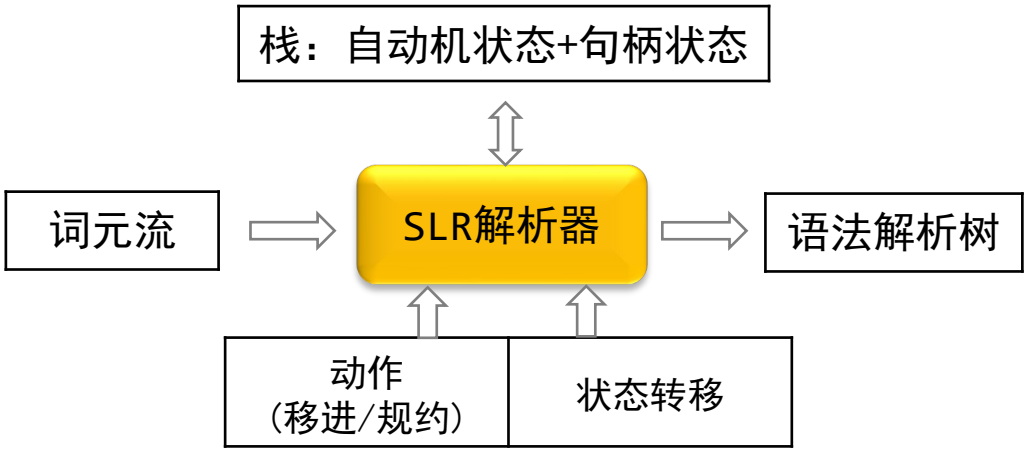


LR(0)有穷自动机的状态转移关系表

规范族	E	E1	E2	E3	OP1	OP2	OP3	NUM	<UNUM>	'+'	'-'	'*'	'/'	'^'	<LP>	<RP>	<eof>
S0	S1	S2	S3	S4				S5	S7		S8				S6		
S1					S9					S10	S11						accept
S2						S12						S13	S14				
S3																	
S4							S15							S16			
S5																	
S6	S17	S2	S3	S4				S5	S7		S8				S6		
S7																	
S8									S18								
S9		S19	S3	S4				S5	S7		S8				S6		
S10																	
S11																	
S12			S20	S4				S5	S7		S8				S6		
S13																	
S14																	
S15			S21	S4				S5	S7		S8				S6		
S16																	
S17					S9					S10	S11					S22	
S18																	
S19						S12						S13	S14				
S20																	
S21																	
S22																	

构建SLR解析器

- 移进条件：如果$A \to \alpha \circ a\beta \in S_i$，并且$Goto(S_i, a) = S_j$，执行$Action(S_i, a) = Shift\ S_j$
- 规约条件：如果$A \to \alpha \circ \in S_i$，$\forall a \in Follow(A)$，执行$Action(S_i, a) = Reduce\ A \to \alpha$



SLR解析表

规范族	GOTO								Action (Shift-Reduce)								
	E	E1	E2	E3	OP1	OP2	OP3	NUM	<UNUM>	'+'	'-'	'*'	'/'	'^'	<LP>	<RP>	<eof>
S0	S1	S2	S3	S4				S5	S7		S8				S6		
S1					S9					S10	S11						acc
S2						S12				R[2]	R[2]	S13	S14			R[2]	R[2]
S3										R[4]	R[4]	R[4]	R[4]			R[4]	R[4]
S4							S15			R[6]	R[6]	R[6]	R[6]	S16		R[6]	R[6]
S5										R[7]	R[7]	R[7]	R[7]	R[7]		R[7]	R[7]
S6								S5	S7		S8				S6		
S7										R[9]	R[9]	R[9]	R[9]	R[9]		R[9]	R[9]
S8									S18								
S9								S5	S7		S8				S6		
S10									R[11]		R[11]				R[11]		
S11									R[12]		R[12]				R[12]		
S12								S5	S7		S8				S6		
S13									R[13]		R[13]				R[13]		
S14									R[14]		R[14]				R[14]		
S15								S5	S7		S8				S6		
S16									R[15]		R[15]				R[15]		
S17										S10	S11					S22	
S18										R[10]	R[10]			R[10]		R[10]	R[10]
S19						S12				R[1]	R[1]	S13	S14			R[1]	R[1]
S20										R[3]	R[3]	R[3]	R[3]			R[3]	R[3]
S21										R[5]	R[5]					R[5]	R[5]
S22										R[8]	R[8]	R[8]	R[8]	R[8]		R[8]	R[8]

S2: E → E1 ∘
S3: E1 → E2 ∘
S4: E2 → E3 ∘
S5: E3 → NUM ∘

[0] S → ∘ E
[1] E → E OP1 E1
[2] E → E1
[3] E1 → E1 OP2 E2
[4] E1 → E2
[5] E2 → E3 OP3 E2
[6] E2 → E3
[7] E3 → NUM
[8] E3 → '(' E ')'
[9] NUM → <UNUM>
[10] NUM → '-' <UNUM>
[11] OP1 → '+'
[12] OP1 → '-'
[13] OP2 → '*'
[14] OP2 → '/'
[15] OP3 → '^'

SLR查表解析应用示例

规范族	GOTO								Action (Shift-Reduce)								
	E	E1	E2	E3	OP1	OP2	OP3	NUM	<UNUM>	'+'	'-'	'*'	'/'	'^'	<LP>	<RP>	<eof>
S0	S1	S2	S3	S4				S5	S7		S8				S6		
S1					S9					S10	S11						acc
S2						S12				R[2]	R[2]	S13	S14			R[2]	R[2]
S3										R[4]	R[4]	R[4]	R[4]			R[4]	R[4]
S4							S15			R[6]	R[6]	R[6]	R[6]	S16		R[6]	R[6]
S5										R[7]	R[7]	R[7]	R[7]	R[7]		R[7]	R[7]
S6	S17	S2	S3	S4				S5	S7		S8				S6		
S7										R[9]	R[9]	R[9]	R[9]	R[9]		R[9]	R[9]
S8									S18								
S9		S19	S3	S4				S5	S7		S8				S6		
S10									R[11]		R[11]				R[11]		

状态栈	符号栈	待读入词元	操作
S0		<UNUM>'*'<UNUM><eof>	shift <UNUM>, goto S7
S0,S7	<UNUM>	'*'<UNUM><eof>	Reduce [9], back to S0, goto S5
S0,S5	NUM	'*'<UNUM><eof>	Reduce [7], back to S0, goto S4
S0,S4	E3	'*'<UNUM><eof>	Reduce [6], back to S0, goto S3
S0,S3	E2	'*'<UNUM><eof>	Reduce [4], back to S0, goto S2
S0,S2	E1	'*'<UNUM><eof>	Shift '*', goto S13
S0,S2,S13	E1 '*'	<UNUM><eof>	Reduce [13], back to S2, goto S12
S0,S2,S12	E1 OP2	<UNUM><eof>	

SLR查表解析应用示例

状态栈	符号栈	待读入词元	操作
S0		<UNUM>'*'<UNUM><eof>	shift <UNUM>, goto S7
S0,S7	<UNUM>	'*'<UNUM><eof>	Reduce [9], back to S0, goto S5
S0,S5	NUM	'*'<UNUM><eof>	Reduce [7], back to S0, goto S4
S0,S4	E3	'*'<UNUM><eof>	Reduce [6], back to S0, goto S3
S0,S3	E2	'*'<UNUM><eof>	Reduce [4], back to S0, goto S2
S0,S2	E1	'*'<UNUM><eof>	Shift '*', goto S13
S0,S2,S13	E1 '*'	<UNUM><eof>	Reduce [13], back to S2, goto S12
S0,S2,S12	E1 OP2	<UNUM><eof>	Shift <UNUM>, goto S7
S0,S2,S12,S7	E1 OP2 <UNUM>	<eof>	Reduce [9], back to S12, goto S5
S0,S2,S12,S5	E1 OP2 NUM	<eof>	Reduce [7], back to S12, goto S4
S0,S2,S12,S4	E1 OP2 E3	<eof>	Reduce [6], back to S12, goto S20
S0,S2,S12,S20	E1 OP2 E2	<eof>	Reduce [3], back to S0, goto S2
S0,S2	E1	<eof>	Reduce [2], back to s0, goto S1
S0,S1	E	<eof>	accept

练习

- 下列语法属于（多选题）：
  a) SLR
  b) LL(1)

```
[1] T → TA
[2] T → A
[3] A → a
```

练习

- 为下列语法规则构造SLR解析表

```
[1] REGEX   → REGEX '|' CONCAT
[2] REGEX   → CONCAT
[3] CONCAT  → CONCAT CLOSURE
[4] CONCAT  → CLOSURE
[5] CLOSURE → CLOSURE '*'
[6] CLOSURE → ITEM
[7] ITEM    → '(' REGEX ')'
[8] ITEM    → <CHAR>
```

思考

- LL(1)和SLR哪个语法的表达能力更强？
  - 如果一个语法是SLR，是否一定是LL(1)？
  - 如果一个语法是LL(1)，是否一定是SLR？

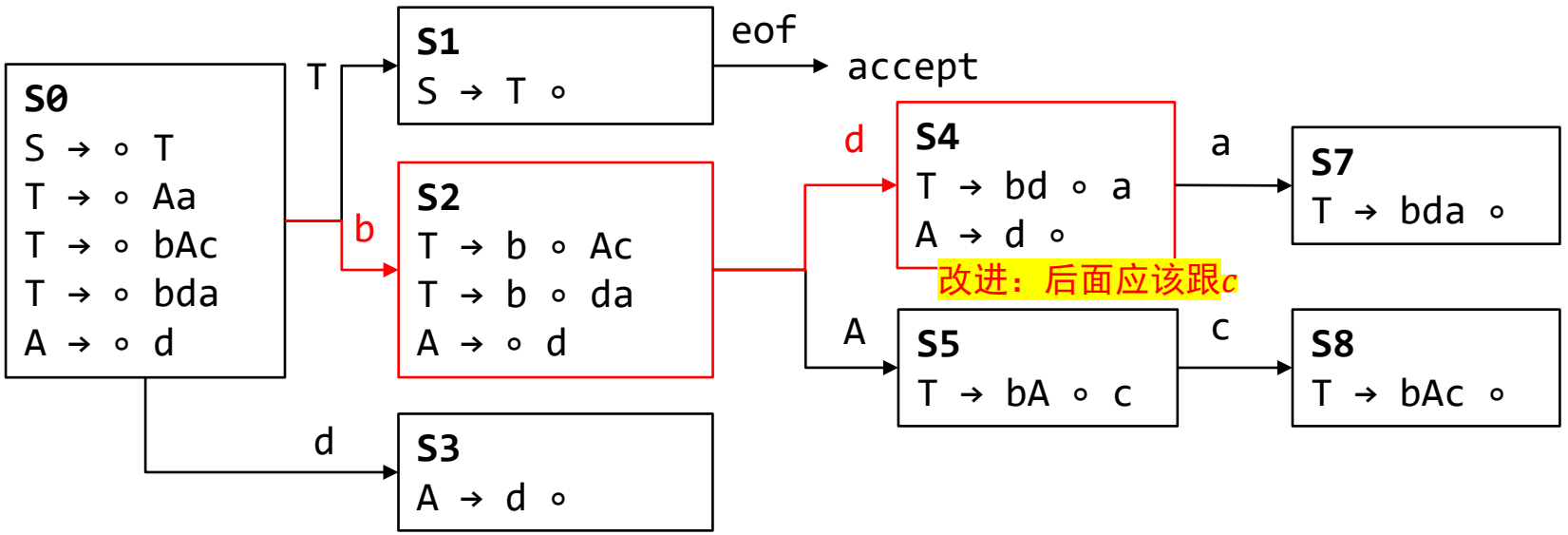
三、LR(1)文法

示例：SLR移进-规约冲突

- 解析字符串“bda”时存在移进-规约冲突
  - S4下一个字符为a，可移进
  - Follow(A) = {a,c}，可规约

语法规则：

```
[1] T → Aa
[2] T → bAc
[3] T → bda
[4] A → d
```

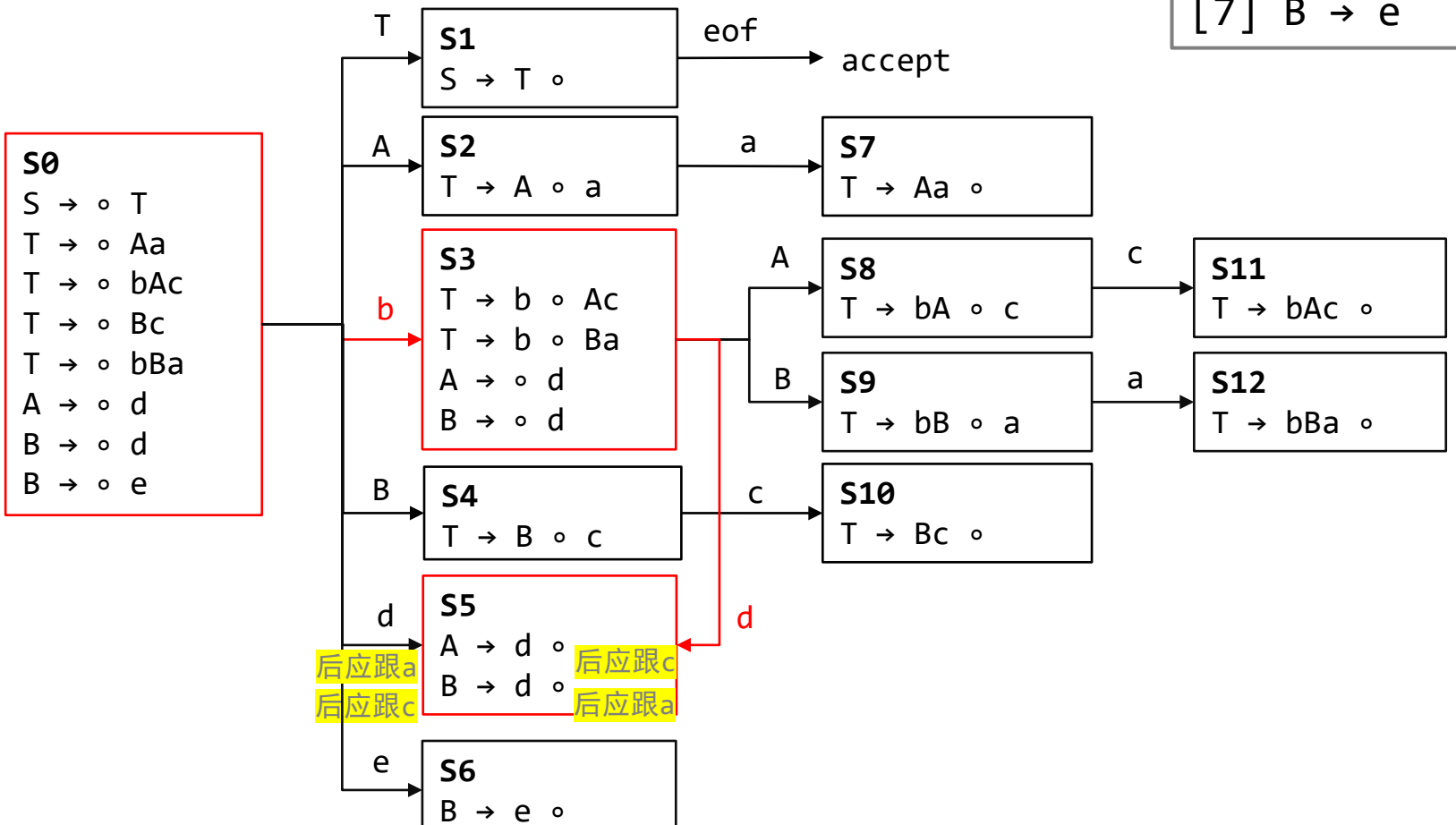


LR(0)有穷自动机

示例：SLR规约-规约冲突

- 解析“bda”时存在规约-规约冲突
  - Follow(A) = Follow(B) = {a,c}
- 解析da、dc等其它句子时存在同样的问题

[1] T → Aa
[2] T → bAc
[3] T → Bc
[4] T → bBa
[5] A → d
[6] B → d
[7] B → e

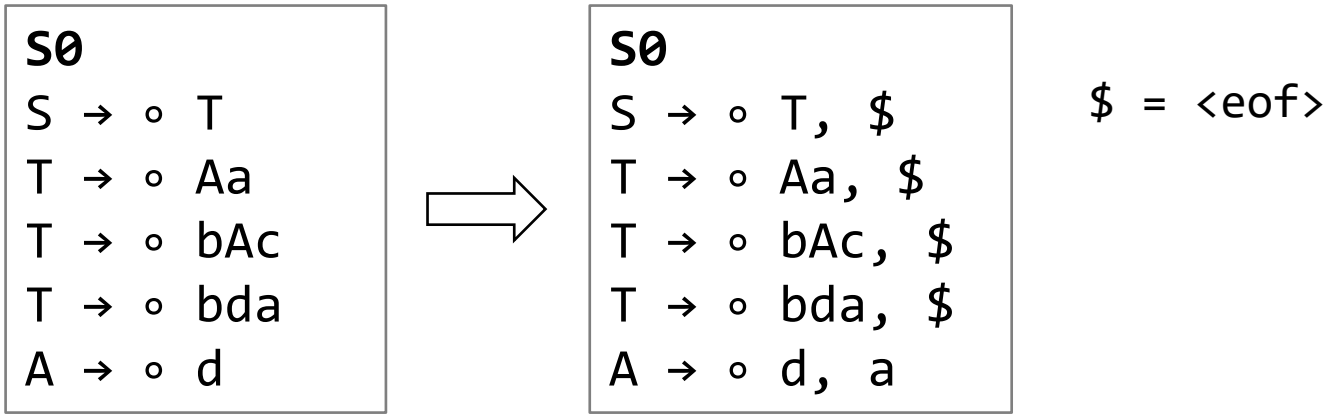


SLR的局限性：解析表可能存在冲突

- 原因：SLR表达能力太弱
  - 移进-规约冲突
  - 规约-规约冲突
- 增强表达能力：
  - LR(1)>LALR>SLR：项目集构造时考虑规约上下文的Follow信息
  - 通用CFG解析算法：GLR(Generalized LR)、CYK

如果存在冲突怎么办？

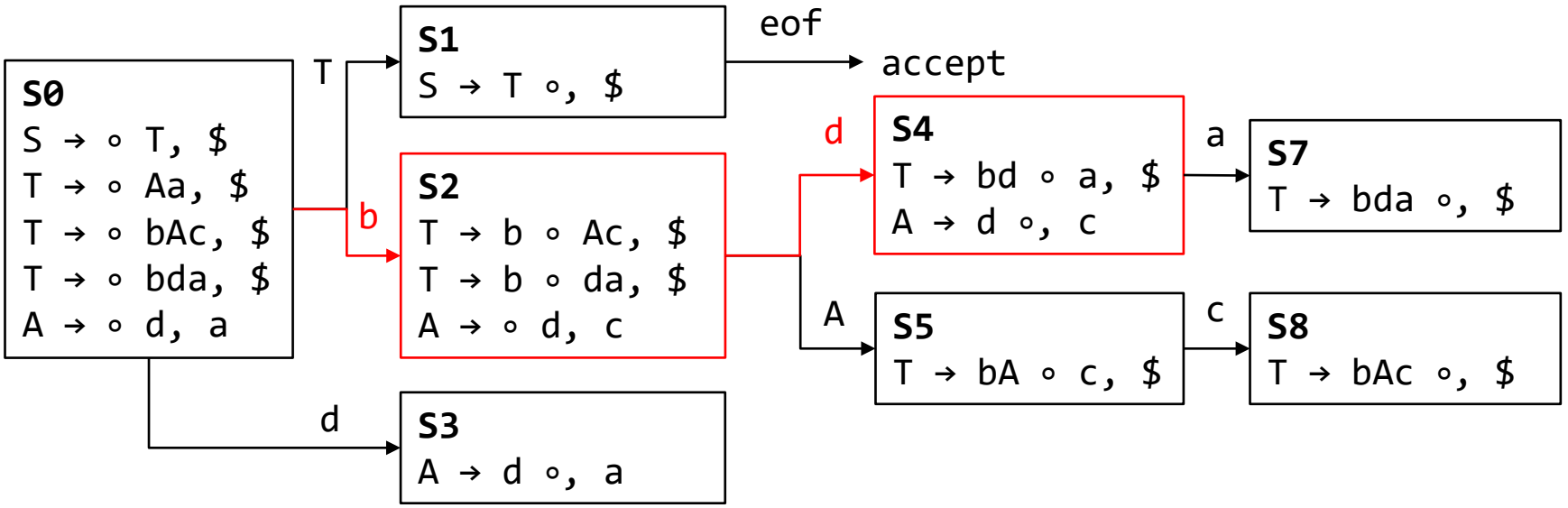
- 进一步细化SLR解析表：提前考虑Follow信息
- LR(1)项目集：记录每条项目对应的Follow字符信息



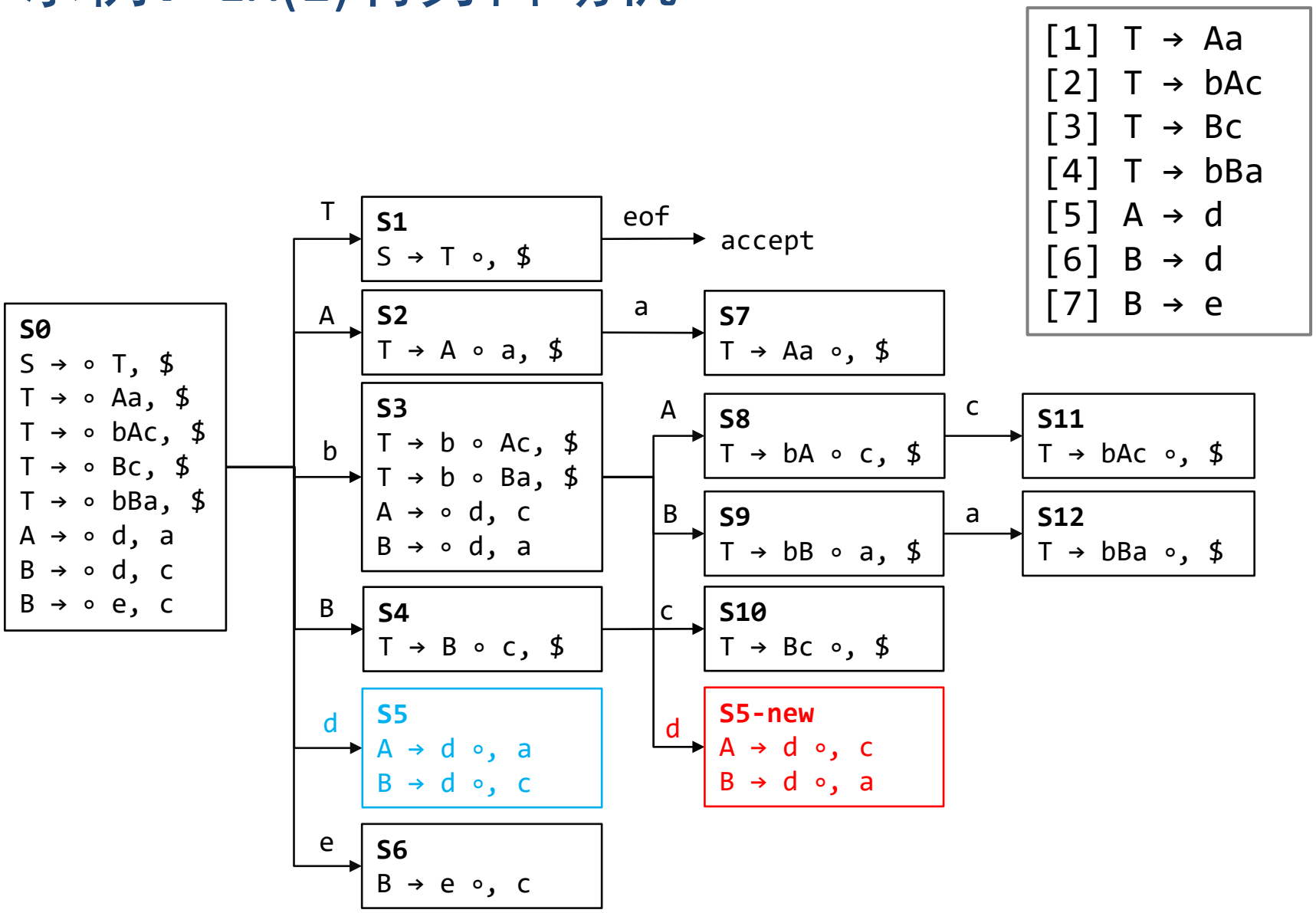
LR(0)规范族　　LR(1)规范族

示例：LR(1)有穷自动机

[1] T → Aa
[2] T → bAc
[3] T → bda
[4] A → d



示例：LR(1)有穷自动机


[1] T → Aa
[2] T → bAc
[3] T → Bc
[4] T → bBa
[5] A → d
[6] B → d
[7] B → e
S0
S → ∘ T, $
T → ∘ Aa, $
T → ∘ bAc, $
T → ∘ Bc, $
T → ∘ bBa, $
A → ∘ d, a
B → ∘ d, c
B → ∘ e, c
T
S1
S → T ∘, $
eof
accept
A
S2
T → A ∘ a, $
a
S7
T → Aa ∘, $
b
S3
T → b ∘ Ac, $
T → b ∘ Ba, $
A → ∘ d, c
B → ∘ d, a
A
S8
T → bA ∘ c, $
c
S11
T → bAc ∘, $
B
S9
T → bB ∘ a, $
a
S12
T → bBa ∘, $
B
S4
T → B ∘ c, $
c
S10
T → Bc ∘, $
d
S5
A → d ∘, a
B → d ∘, c
d
S5-new
A → d ∘, c
B → d ∘, a
e
S6
B → e ∘, c

问题

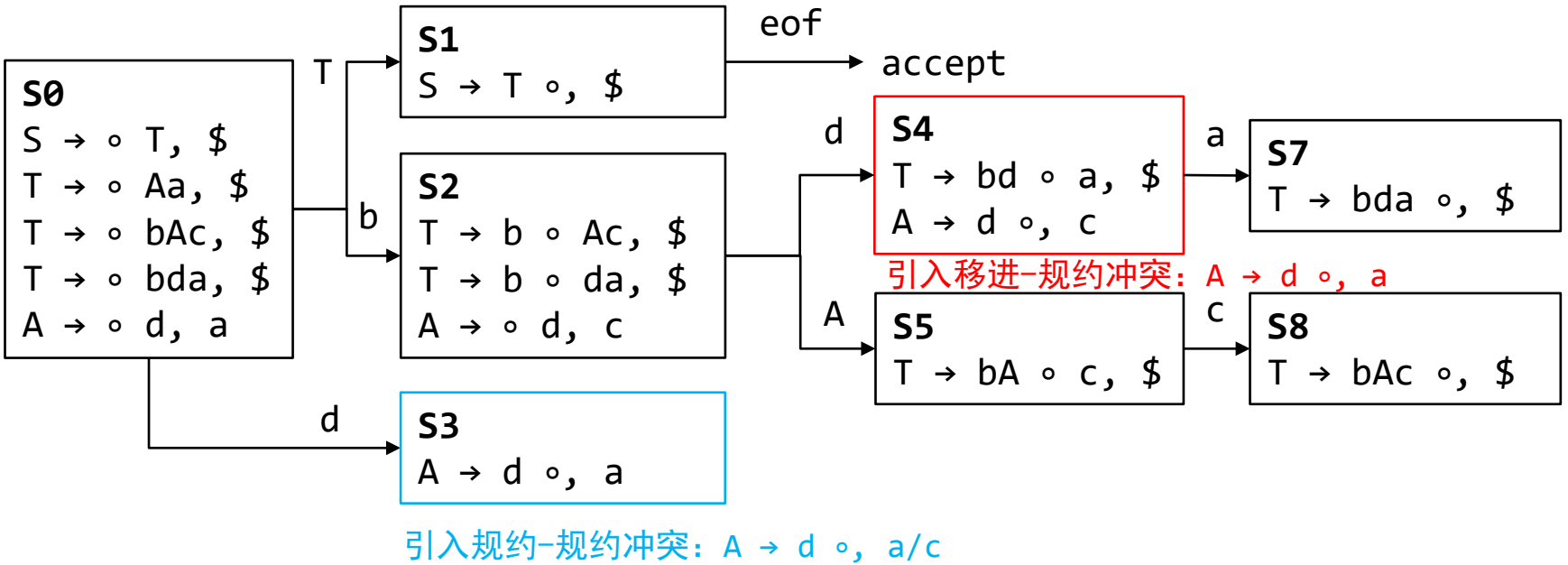
- 以下两套语法规则是否属于LL(1)?

```
[1] T → bAc
[2] T → bda
[3] T → Aa
[4] A → d
```

```
[1] T → Aa
[2] T → bAc
[3] T → Bc
[4] T → bBa
[5] A → d
[6] B → d
[7] B → e
```

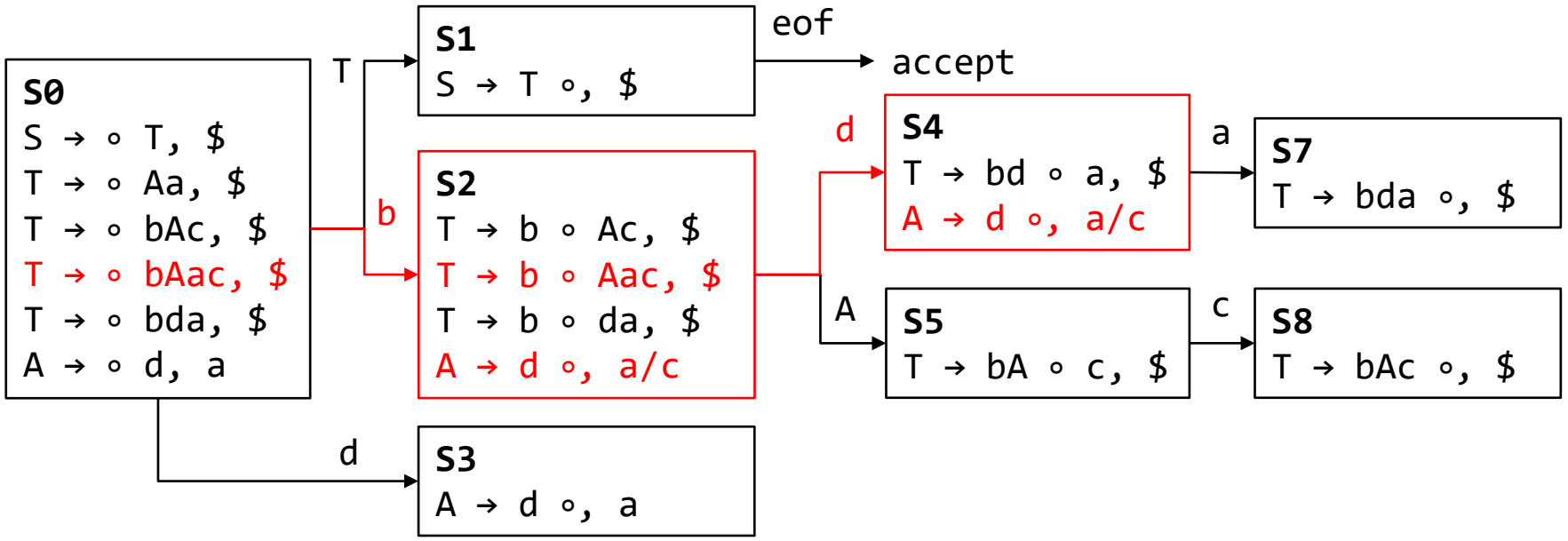
CFG是否可能非LR(1)?

[1] T → Aa
[2] T → bAc
[3] T → bda
[4] A → d

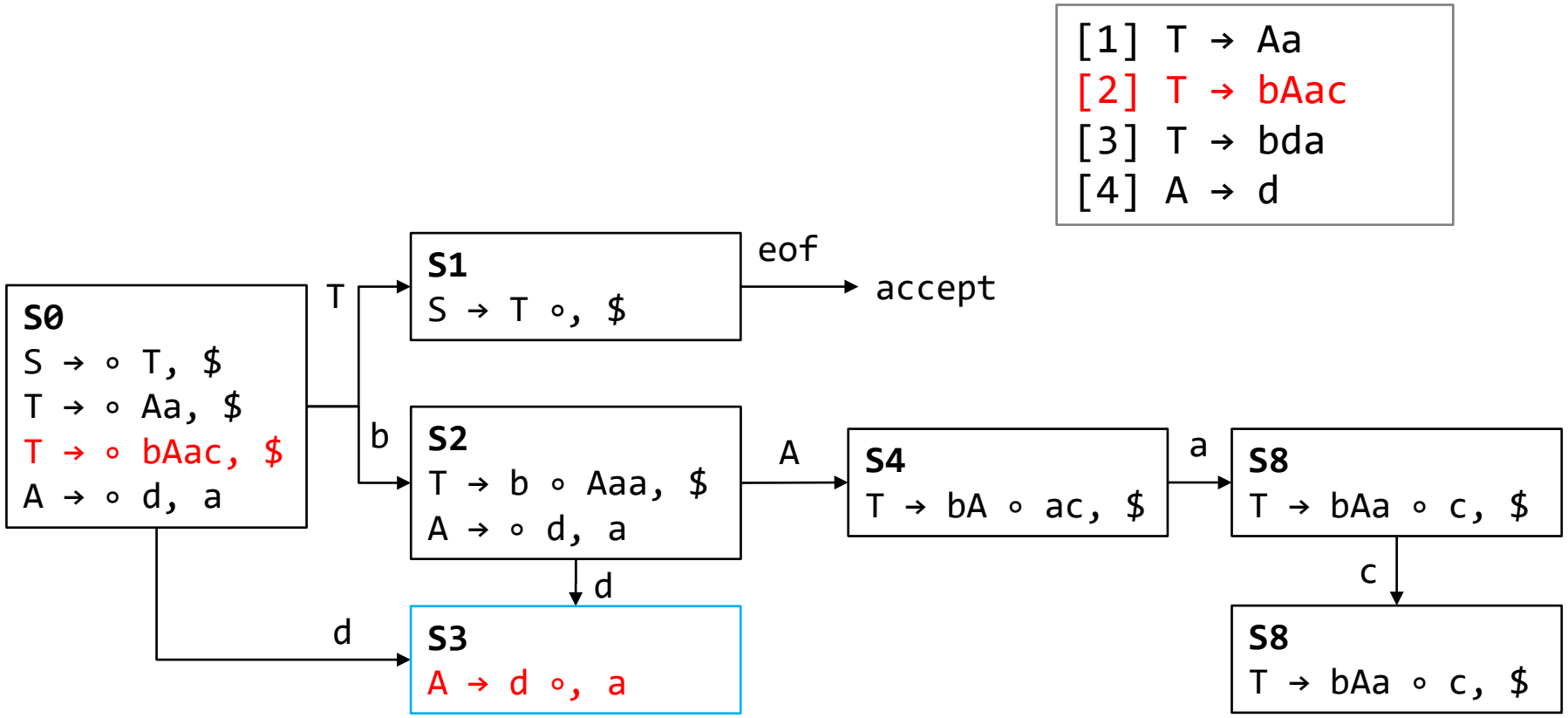


构造非LR(1)：移进-规约冲突

[1] T → Aa
[2] T → bAc
[3] T → bAac
[4] T → bda
[5] A → d



构造非LR(1)：规约-规约冲突



四、LALR文法

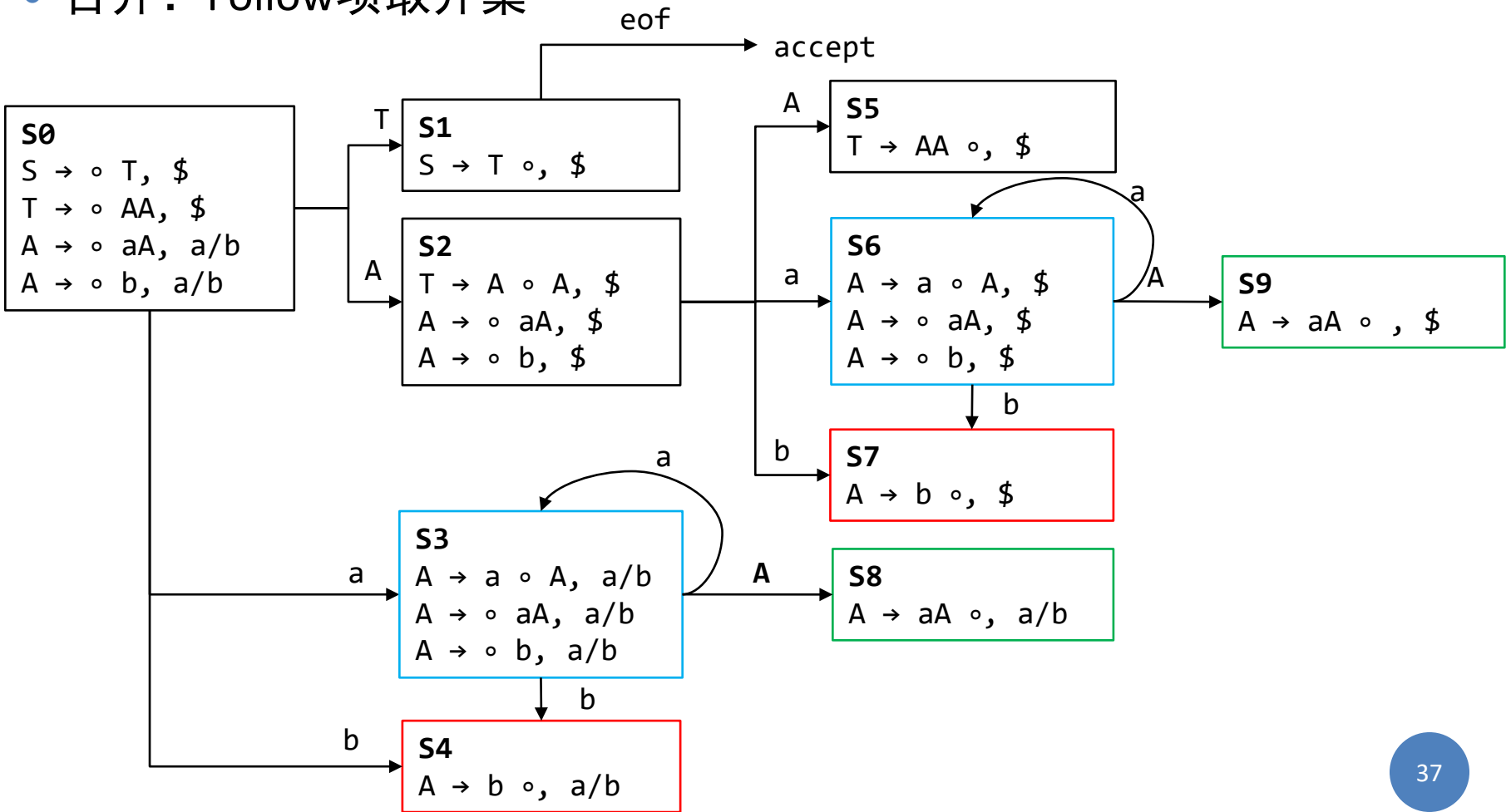
LR(1)的问题

- 表达能力有限：并非所有CFG语法都属于LR(1)
- LR(1)的项目集数量可能远多于LR(0)
  - 折中思路LALR（Lookahead LR）：精简项目集族
  - 有穷自动机构造时考虑Follow信息
  - 合并核心项完全相同的规范集

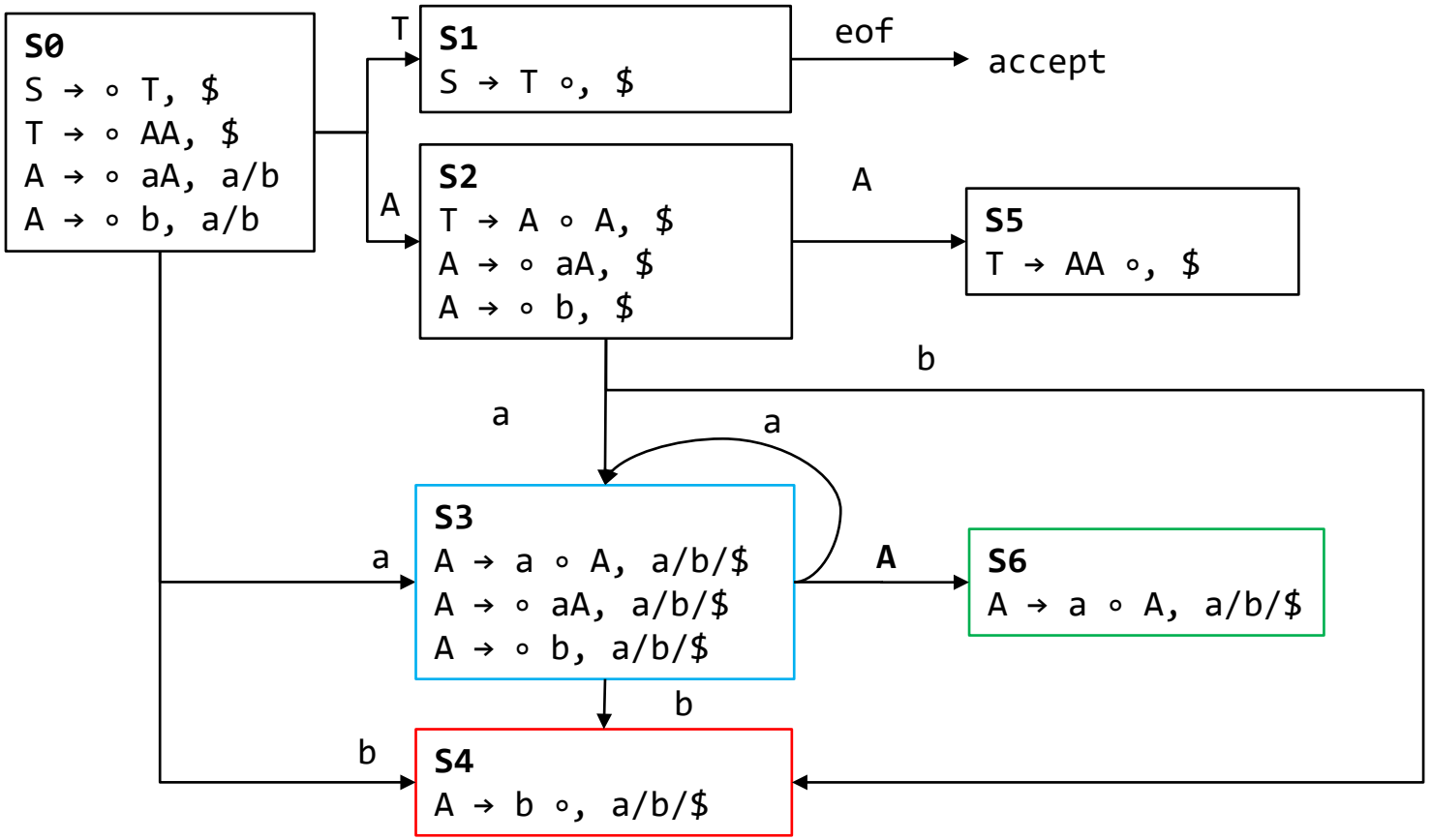
LALR举例

- 核心项相同的项目集：
  - S3和S6、S4和S7、S8和S9
- 合并：Follow项取并集

```
[0] S → T
[1] T → AA
[2] A → aA
[3] A → b
```



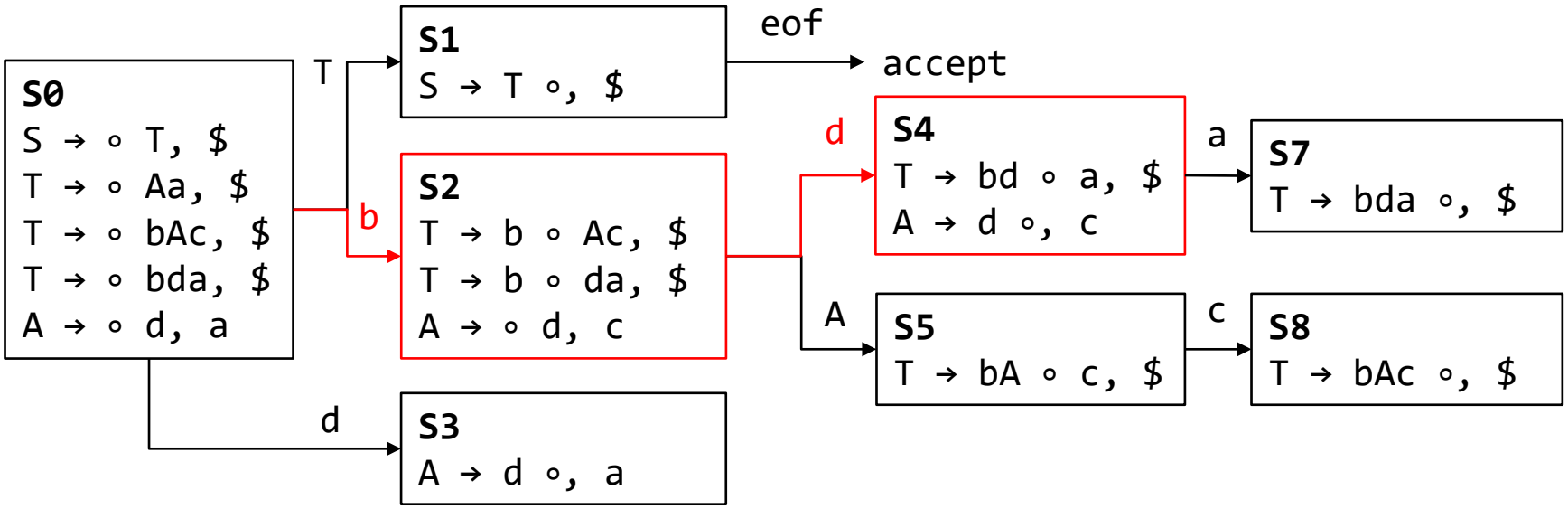
示例：LALR有穷自动机



LALR?

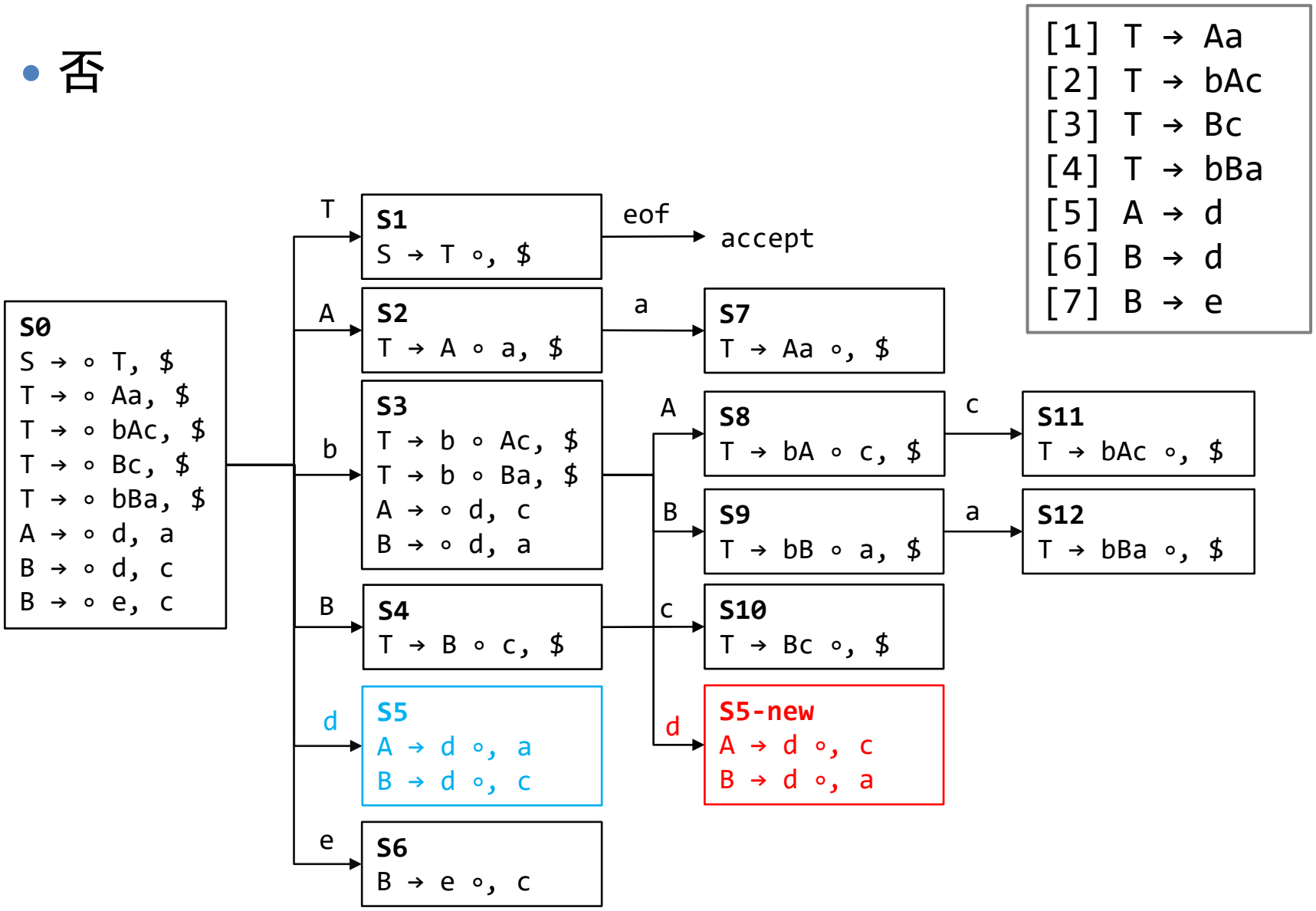
- 是

```
[1] T → bAc
[2] T → bda
[3] T → Aa
[4] A → d
```



LALR?

- 否


[1] T → Aa
[2] T → bAc
[3] T → Bc
[4] T → bBa
[5] A → d
[6] B → d
[7] B → e
S0
S → ∘ T, $
T → ∘ Aa, $
T → ∘ bAc, $
T → ∘ Bc, $
T → ∘ bBa, $
A → ∘ d, a
B → ∘ d, c
B → ∘ e, c
T
A
b
B
d
e
S1
S → T ∘, $
eof
accept
S2
T → A ∘ a, $
a
S7
T → Aa ∘, $
S3
T → b ∘ Ac, $
T → b ∘ Ba, $
A → ∘ d, c
B → ∘ d, a
S4
T → B ∘ c, $
S5
A → d ∘, a
B → d ∘, c
S6
B → e ∘, c
A
B
c
d
S8
T → bA ∘ c, $
S9
T → bB ∘ a, $
S10
T → Bc ∘, $
S5-new
A → d ∘, c
B → d ∘, a
c
a
S11
T → bAc ∘, $
S12
T → bBa ∘, $

通用CFG解析算法

- GLR算法：允许解析表单元格有冲突，广度优先搜索
- CYK算法：基于动态规划思想，非预测解析

练习

- 下列语法属于（多选题）：
  a) LL(1)
  b) SLR
  c) LALR
  d) LR(1)

```
[1] REGEX   → REGEX '|' CONCAT
[2] REGEX   → CONCAT
[3] CONCAT  → CONCAT CLOSURE
[4] CONCAT  → CLOSURE
[5] CLOSURE → CLOSURE '*'
[6] CLOSURE → ITEM
[7] ITEM    → '(' REGEX ')'
[8] ITEM    → <CHAR>
```

思考

- LL(1)语法一定是LR(1)吗？为什么？
- LL(1)一定是LALR(1)吗？为什么？