北京大学信息科学技术学院

2022年春季学期《编译原理》

第 4 章 语法分析（4）

Syntax Analysis (4)

【对应教材 4.5-4.6】

内容提要

- 语法分析简介
- 上下文无关文法
- 文法的设计方法
- 自顶向下的语法分析
- **自底向上的语法分析**
  - 简单LR分析：LR(0), SLR
  - **更强大的LR分析：LR(1), LALR**
  - **二义性文法的使用**

SLR的原理：活前缀（1）

- LR(0)自动机刻画了可能出现在语法分析栈中的文法符号串。
  - 直接把项当作状态，可以构造得到一个NFA/DFA。

- 活前缀 (Viable Prefix, 也译作可行前缀):
  - 可以出现在移进-归约语法分析器栈中的右句型前缀
  - 定义：某个右句型的前缀，且没有越过该句型的句柄的右端。

- 有效项：
  - 如果存在$S=>\alpha Aw=>\alpha\beta_1\beta_2 w$，那么我们说项$A\rightarrow\beta_1\bullet\beta_2$对$\alpha\beta_1$有效。

SLR的原理：活前缀（2）

- 如果$A \rightarrow \beta_1 \cdot \beta_2$对$\alpha\beta_1$有效，
  - 当β_2不等于空，表示句柄尚未出现在栈中，应该移进；
  - 如果β_2等于空，表示句柄出现在栈中，应该归约。

- 如果某个时刻存在两个有效项要求执行不同的动作，那么就存在冲突。
  - 冲突实际上表示可行前缀可能是两个最右句型的前缀，第一个包含了句柄，而另一个尚未包含句柄
  - SLR解决冲突的思想：假如要按照$A \rightarrow \beta$进行归约，那么A后面跟着的下一个输入符号在FOLLOW(A)中时才可以归约。

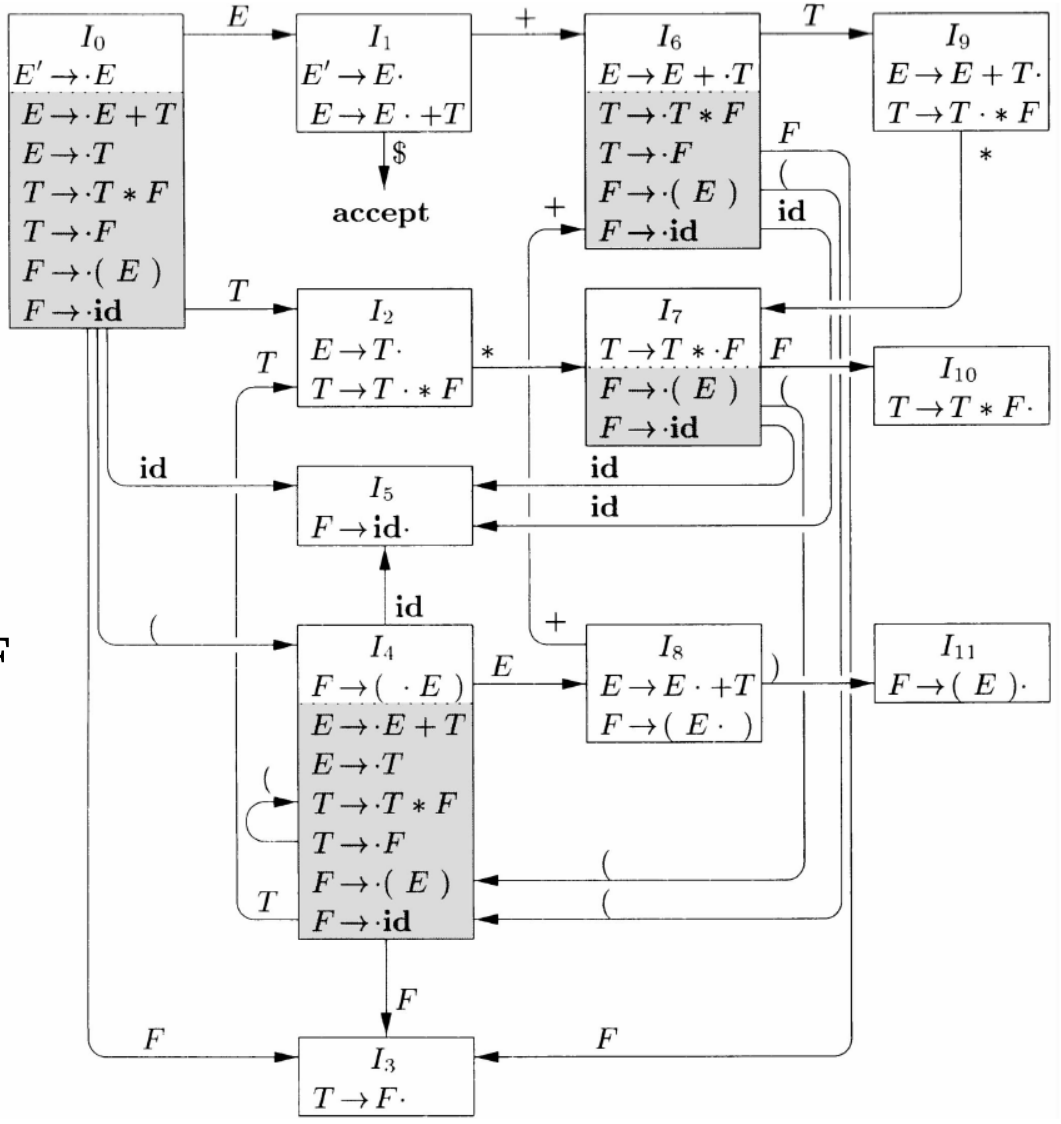
SLR的原理：活前缀（3）

- 如果$A \rightarrow \beta_1 \cdot \beta_2$对$\alpha\beta_1$有效，
  - A？ $\alpha\beta_1$ ？

- 如果在文法G的LR(0)自动机中，从初始状态出发，沿着标号为γ的路径到达一个状态，那么
  - γ是活前缀
  - 这个状态对应的项集就是γ的有效项的集合。

活前缀/有效项的例子

- 活前缀E+T*
- 对应的LR(0)项
  - T→T*•F
  - F→•(E)
  - F→•id
- 对应的最右推导
  - E'=>E=>E+T => E+T*F
  - E' => E => E+T => E+T*F => E+T*(E)
  - E' => E => E+T => E+T*F => E+T*id



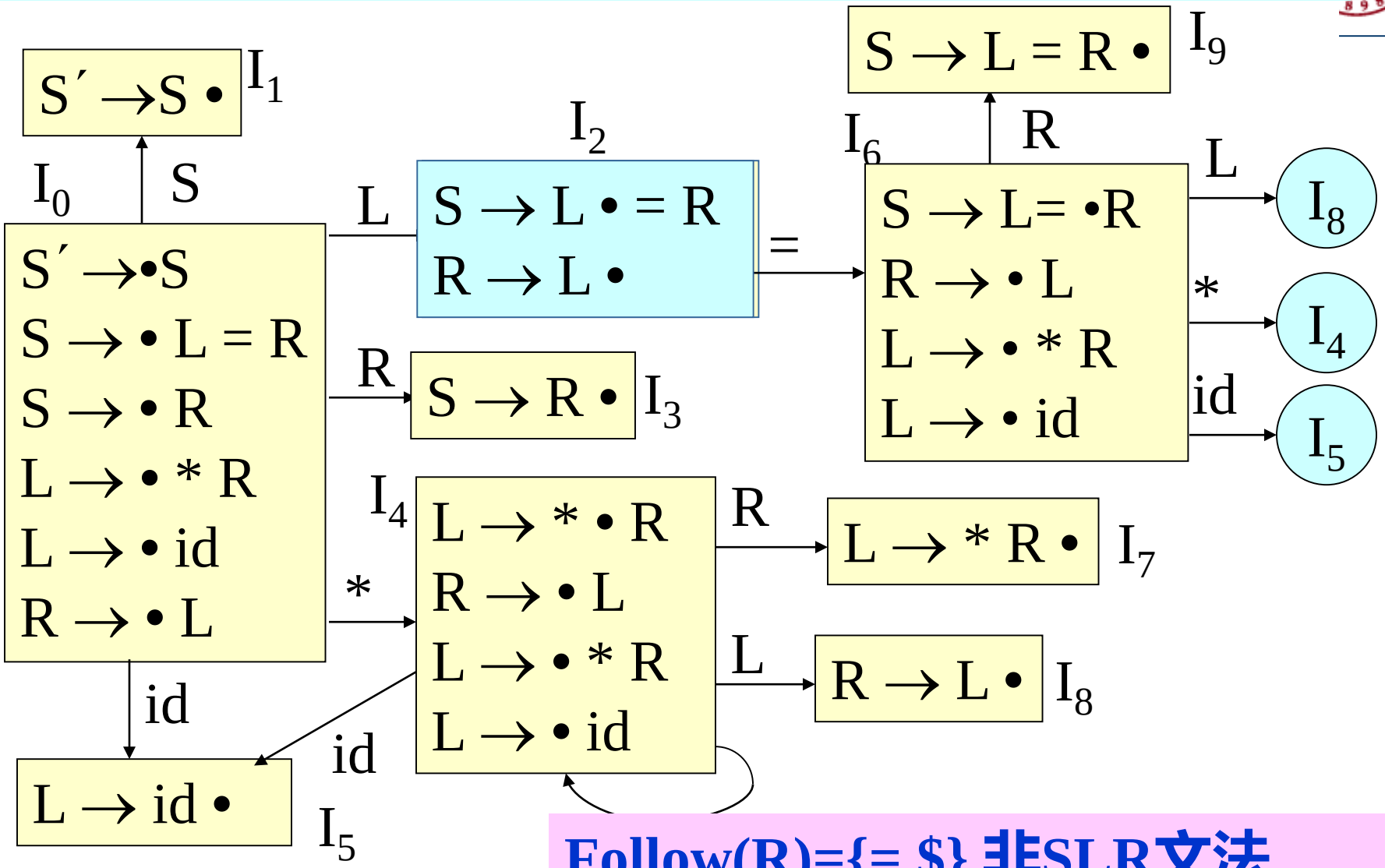
SLR语法分析器的弱点（1）

- SLR技术解决冲突的方法：
  - 项集中包含[A→α•]时，按照A→α进行归约的条件是下一个输入符号a可以在某个句型中跟在A之后。
    - 如果对于a还有其它的移进/归约操作，则出现冲突
  - 假设此时栈中的符号串为βα，
    - 如果βAa不可能是最右句型的前缀，那么即使a在某个句型中跟在A之后，仍然不应该按照A→α归约。
    - 进行归约的条件更加严格可以降低冲突的可能性

SLR语法分析器的弱点（2）

- [A→α•]出现在项集中的条件：
  - 首先[A→• α]出现在某个项集中，然后逐步读入/归约到α中的符号，点不断后移，到达末端。
  - 而[A→• α]出现的条件是B→β•Aγ出现在项中
    - 先按照A→α归约，然后将B→β•Aγ中的点后移到A之后。
  - 显然，在按照A→α归约时要求下一个输入符号是γ的第一个符号。
  - 但是从LR(0)项集中不能确定这个信息。

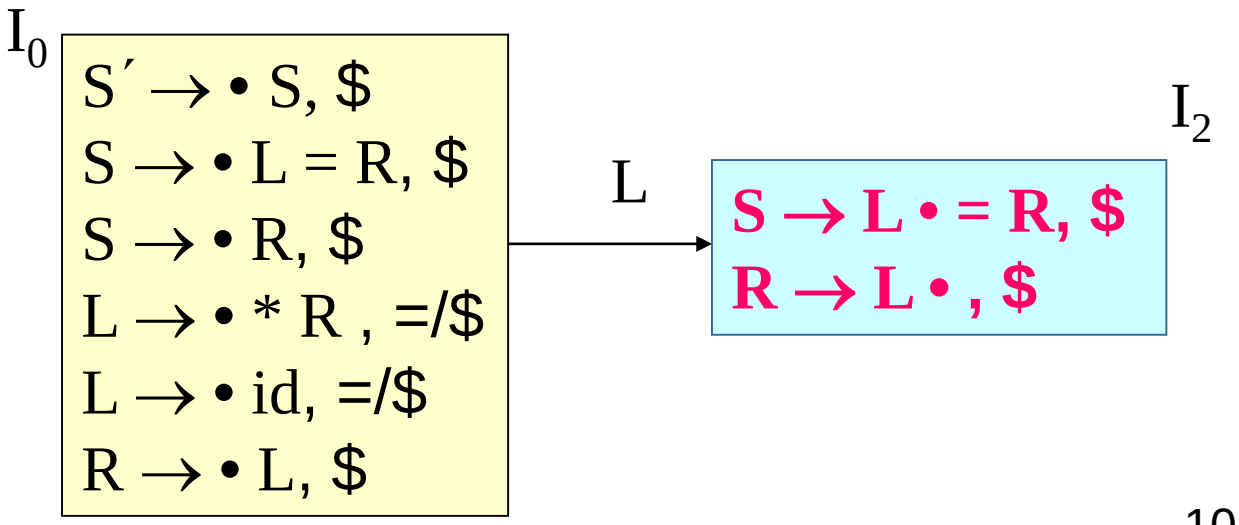
非SLR文法： (0) S′→S (1) S →L = R (2) S → R
(3) L →* R (4) L →id (5) R →L



Follow(R)={=,$},非SLR文法。

LR(1)分析

- 问题：栈中已识别出一个L，此时，a是“=”，不能用“R→L”进行归约，因为不存在“R=”的规范句型。
- 解决方法：添加搜索符号串。
  - 使每个项中含有更多的展望信息，使得能确切知道何时能进行归约。



LR(k)项

- **LR(k) 项**

  形式：$[A\rightarrow\alpha\bullet\beta,\ a_1a_2\ldots a_k]$

  - $\beta\neq\varepsilon$时，是移进项，$a_1a_2\ldots a_k$不起作用。
  - 对归约项$[A\rightarrow\alpha\bullet,\ a_1a_2\ldots a_k]$，仅当前输入符号串开始的前k个符号是$a_1a_2\ldots a_k$时，才能用$A\rightarrow\alpha$进行归约。
  - $a_1a_2\ldots a_k$称为向前搜索符号串。

LR(1)分析表的构造

- **LR(1)有效项：**

  如果存在一个规范推导

  $$S \Rightarrow^*_{rm} \delta Aw \Rightarrow_{rm} \delta\alpha\beta w$$

  则称一个LR(1)项[A→α•β,a]对活前缀γ=δα是有效的。

  - 其中w的第一个符号为a, 或w=ε而a为$。

例：考虑文法G： S→CC　C→cC | d

因为有规范淮导

$$S \Rightarrow^*_{rm} ccCcd \Rightarrow_{rm} cccCcd$$

故项[C→c•C,c] 对活前缀ccc是有效的。

如何推导LR(1)的有效项

- 若项[A→α•Bβ,a] 对活前缀δα是有效的，则存在一个规范推导

$$S \Rightarrow^*_{rm} \delta Aax \Rightarrow_{rm} \delta\alpha\, B\beta ax$$

- 假定$\beta ax \Rightarrow^*_{rm} by$，则对每一个形如B→γ的产生式，我们有规范推导

$$S \Rightarrow^*_{rm} \delta\alpha Bby \Rightarrow_{rm} \delta\alpha\gamma\, by$$

- 从而项[B→ • γ, b] 对于活前缀δα也是有效的。
- 注意到b或者是从β推出的第一个终结符号，或者$\beta \Rightarrow^*_{rm} \varepsilon$而b＝a。这两种可能性结合在一起，则 b∈first(βa)。

LR(1)项集的构造

- 设I是G的一个LR(1)项集，closure(I)是从I出发用下面三个规则构造的项集：

  1. 每一个I中的项都属于closure(I)。

  2. 若项 [A→α•Bβ, a] 属于closure(I) 且 B→γ ∈ P, 则对任何b∈FIRST(βa), 把 [B→• γ, b] 加到 closure(I)中。

  3. 重复执行(2)直到closure(I)不再增大为止。

- 设I是G的一个LR(1)项集，X是一个文法符号，定义

  $$GOTO(I, X) = closure(J),$$

  其中J={[A→ α X•β,a] |当[A→ α •Xβ,a]∈I时}

LR(1)项集的CLOSURE算法

```
SetOfItems CLOSURE(I) {
        repeat
                for ( I中的每个项 [A → α·Bβ, a] )
                        for ( G′中的每个产生式 B → γ )
                                for ( FIRST(βa)中的每个终结符号 b )
                                        将 [B → ·γ, b] 加入到集合I中;
        until不能向I中加入更多的项;
        return I;
}
```

注意添加[B→.γ,b]时，b的取值范围。

LR(1)项集的GOTO算法

SetOfItems GOTO(I, X) {
    将 J 初始化为空集；
    for (I 中的每个项 $[A \to \alpha \cdot X \beta, a]$)
        将项 $[A \to \alpha X \cdot \beta, a]$ 加入到集合 J 中；
    return CLOSURE(J);
}

和LR(0)项集的GOTO算法基本相同

LR(1)项集族的主构造算法

```
void items(G') {
        将 C 初始化为{CLOSURE({[S' → ·S, $]})}
        repeat
                for ( C 中的每个项集 I )
                        for ( 每个文法符号 X )
                                if ( GOTO(I, X) 非空且不在 C 中)
                                        将 GOTO(I, X) 加入 C 中;
        until 不再有新的项集加入到 C 中;
}
```

和LR(0)项集族的构造算法相同

LR(1)项集及规范族的构造过程

例4_9 G(S′):　(0) S′→S　(1) S →CC

(2) C →cC　(3) C →d

若[A →α•Bβ,a] ∈I

且B → γ ∈ P

则[B→• γ, b] ∈I

其中b ∈FIRST(βa)

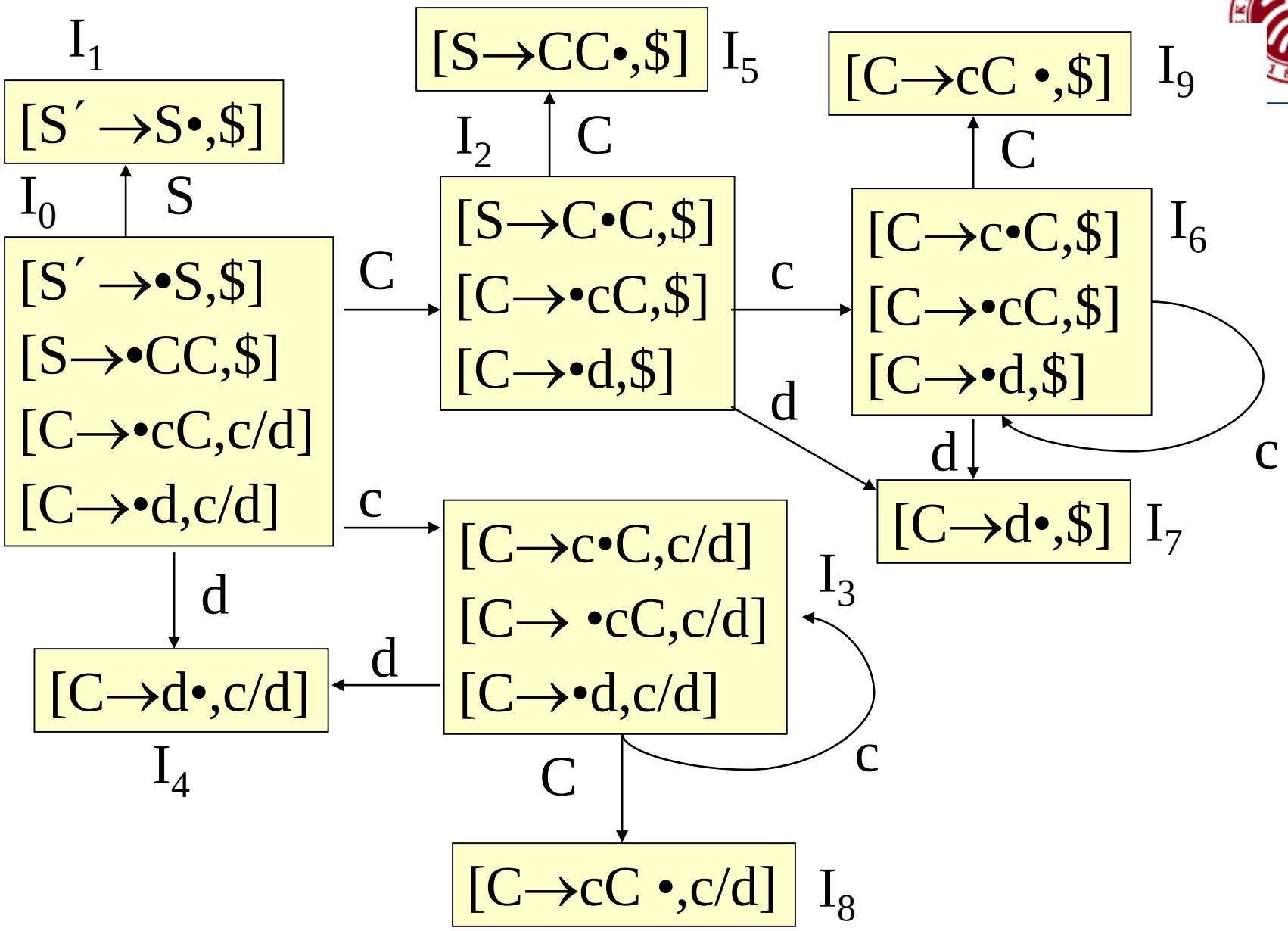
I_0

[S′ →•S, $]

[S →•CC,$]

[C →•cC,c/d]

[C →•d,c/d]


I1
[S′→S•,$]
I0
S
[S′→•S,$]
[S→•CC,$]
[C→•cC,c/d]
[C→•d,c/d]
C
[S→C•C,$]
[C→•cC,$]
[C→•d,$]
I2
C
[S→CC•,$]
I5
c
[C→c•C,$]
[C→•cC,$]
[C→•d,$]
I6
C
[C→cC •,$]
I9
d
[C→d•,$]
I7
d
c
c
[C→c•C,c/d]
[C→ •cC,c/d]
[C→•d,c/d]
I3
d
[C→d•,c/d]
I4
C
[C→cC •,c/d]
I8
c

构造LR(1)分析表

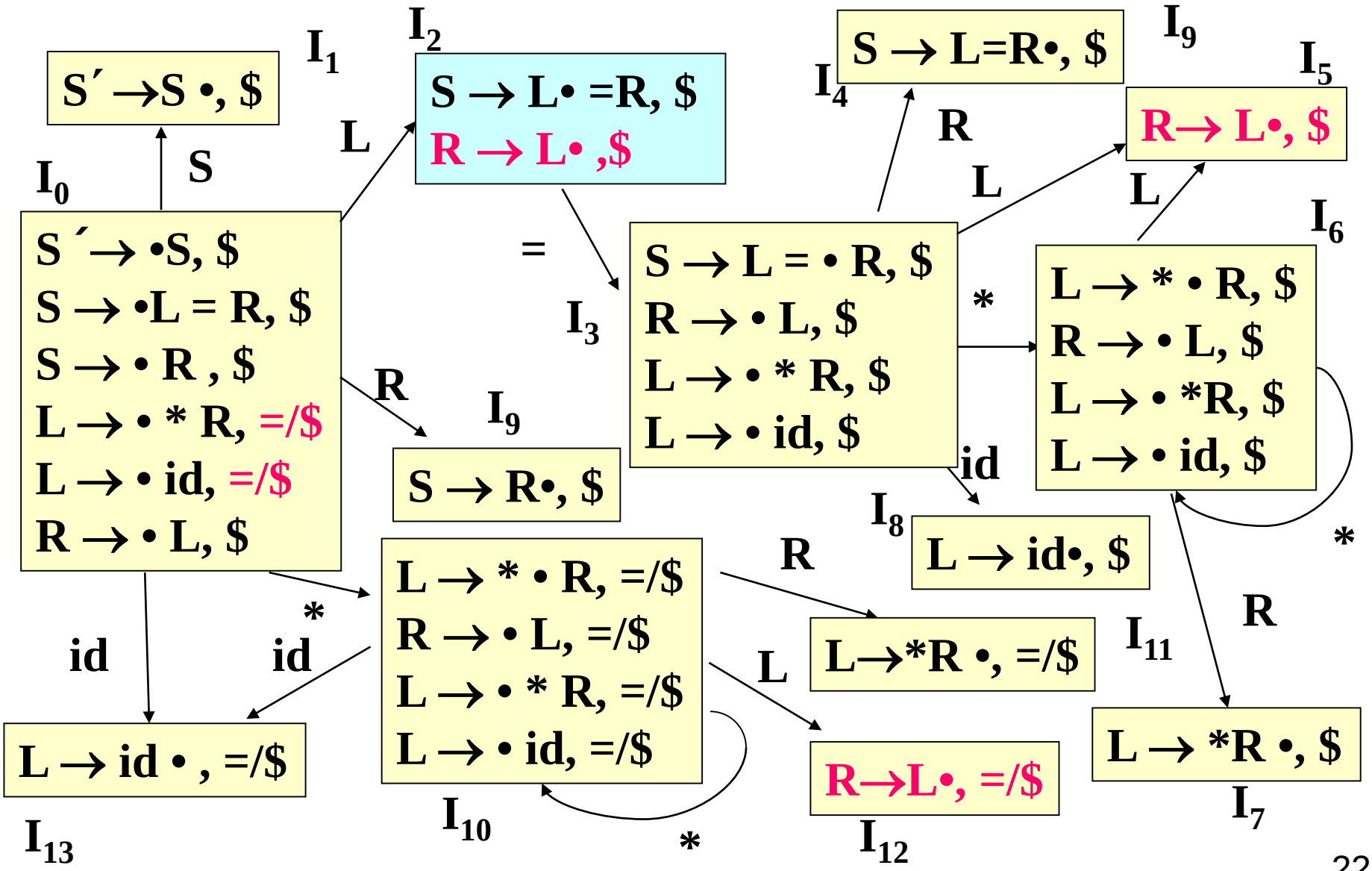
- 根据DFA构造LR(1)分析表M：
  - DFA中的每个状态对应分析表中的一行
  - 对于DFA中的每一个从状态i到状态j的转移
    - 如果转移符号为终结符a：在相应的表项 M[i, a] 中填写移进动作 S_j
    - 如果转移符号为非终结符A：在相应的表项 M[i, A] 中填写要转移到的状态 j
  - 对于包含归约项[$A \rightarrow \alpha \bullet , a$]的状态 i
    - 在相应的表项 M[i, a] 中填写归约动作 r_k，其中 k 是产生式$A \rightarrow \alpha$的编号。

如果按照上面的步骤填写的分析表中没有冲突（即每个单元格中只包含一个动作），那么得到的就是合法的LR(1)分析表

例：LR(1)分析表

	c	d	$	S	C
0	S_3	S_4		1	2
1			acc		
2	S_6	S_7			5
3	S_3	S_4			8
4	r_3	r_3			
5			r_1		
6	S_6	S_7			9
7			r_3		
8	r_2	r_2			
9			r_2		

回顾：(0) S´→S　(1) S →L = R　(2) S → R
(3) L →* R　(4) L →id　(5) R → L

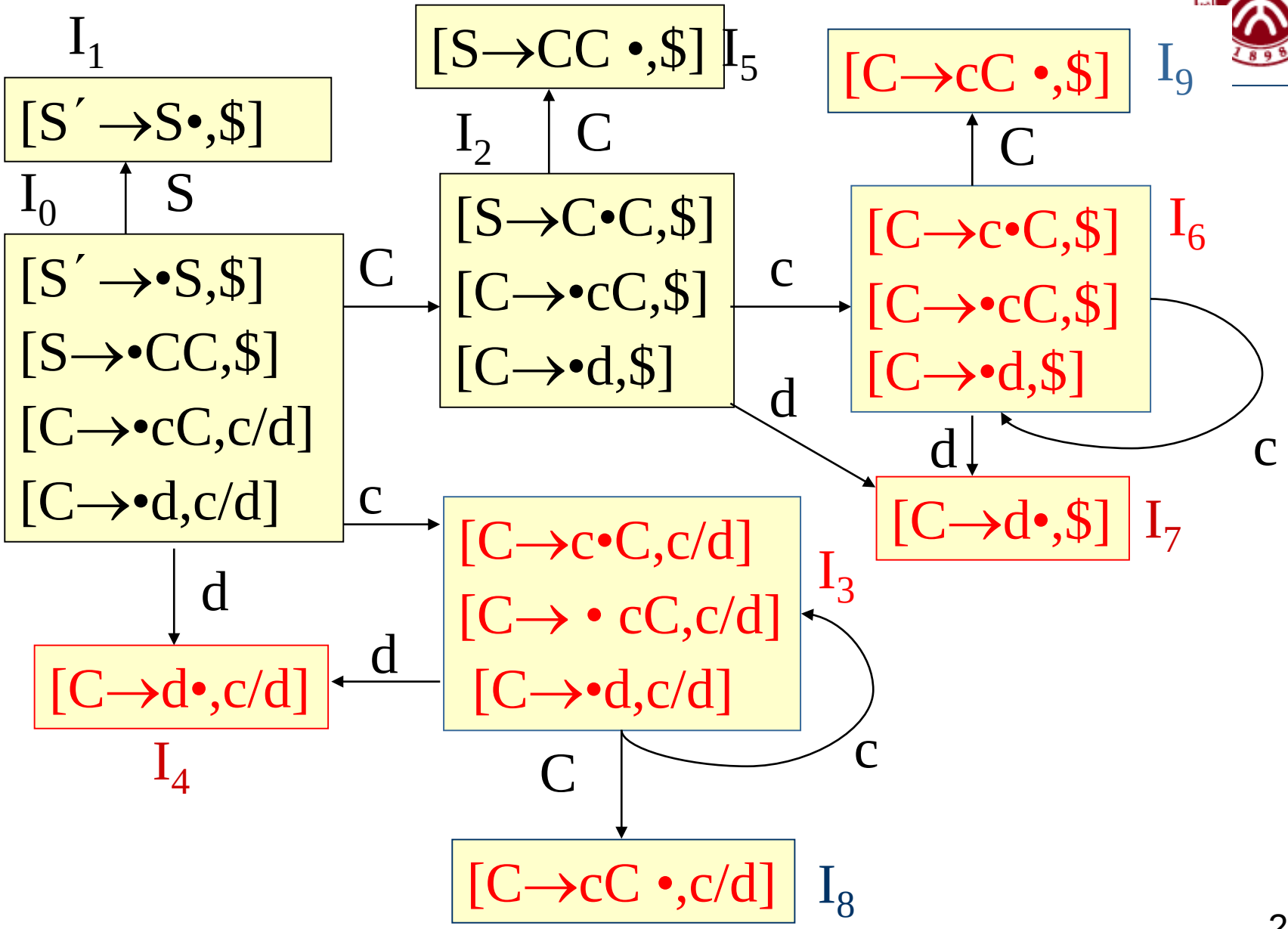


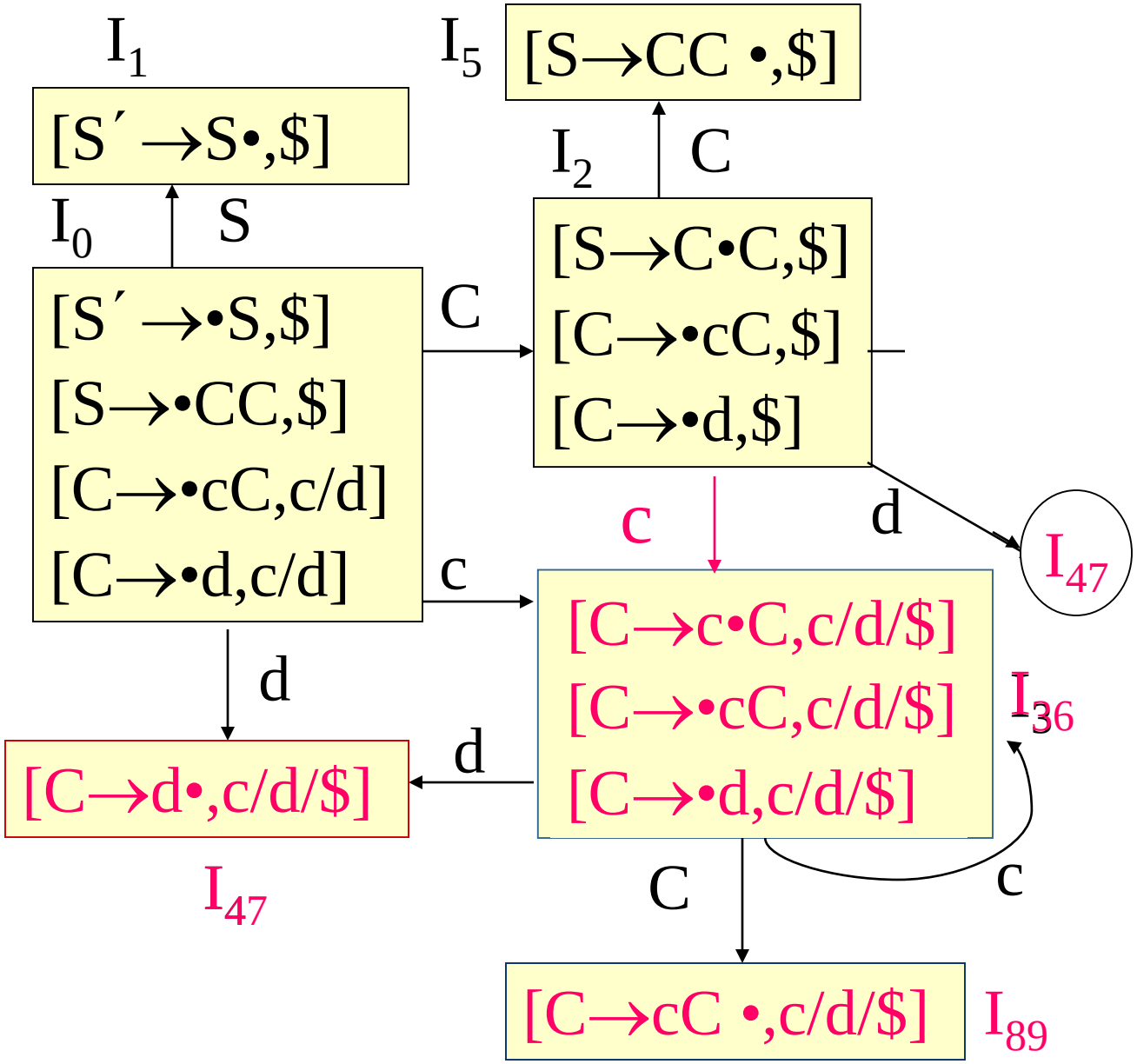
LALR文法

- LR(1)分析表常常会含有比较多的状态，在实际使用中不是很常用。

- 同心集：如果两个LR(1)项集去掉搜索符之后是相同的，则称这两个项集具有相同的核心(core)。

- 合并同心集（合并搜索符串）后，像构造LR(1)分析表一样构造出的LR分析表称作是**LALR(1)分析表**。


I1
[S´→S•,$]
I0
S
[S´→•S,$]
[S→•CC,$]
[C→•cC,c/d]
[C→•d,c/d]
C
c
d
I2
[S→C•C,$]
[C→•cC,$]
[C→•d,$]
[S→CC •,$]
I5
C
c
d
I6
[C→c•C,$]
[C→•cC,$]
[C→•d,$]
[C→cC •,$]
I9
C
d
c
I7
[C→d•,$]
[C→c•C,c/d]
[C→ • cC,c/d]
[C→•d,c/d]
I3
d
c
C
[C→d•,c/d]
I4
[C→cC •,c/d]
I8


I1
[S′→S•,$]
I0
S
[S′→•S,$]
[S→•CC,$]
[C→•cC,c/d]
[C→•d,c/d]
I5
[S→CC •,$]
I2
C
[S→C•C,$]
[C→•cC,$]
[C→•d,$]
C
c
d
I47
c
[C→c•C,c/d/$]
[C→•cC,c/d/$]
[C→•d,c/d/$]
I36
d
[C→d•,c/d/$]
I47
d
C
c
[C→cC •,c/d/$]
I89

对应的LALR 分析表

	c	d	$	S	C
0	S_3	S_4		1	2
1			acc		
2	S_6	S_7			5
3(6)	S_3	S_4			8
4(7)	r_3	r_3	r_3		
5			r_1		
~~6~~	~~S_6~~	~~S_7~~			~~9~~
~~7~~			~~r_3~~		
8(9)	r_2	r_2	r_2		
~~9~~			~~r_2~~		

LALR分析表的高效构造算法

- LALR的项集可以看作是在LR(0)项集上添加了适当的向前看符号。
- 基本方法
  - 只使用内核项来表示LR(0)或LR(1)项集。只使用[S'→•S]或[S'→•S,$]，以及点不在最左端的项
  - 使用传播和自发生成的过程来生成向前看符号，得到LALR(1)内核项
  - 使用CLOSURE函数，求出内核的闭包。然后得到LALR分析表

详细算法参见龙书第4.7.5节

LALR(1)的讨论

1. goto(I, X) 仅仅依赖于I的核心，因此 LR(1)项集合并后的转换函数 goto(I, X) 随自身的合并而得到。
2. 动作action应当进行修改，反映各被合并集合的既定动作。
3. 把同心的项集合并的时候，有可能导致冲突，这种冲突不会是移进-归约冲突；但可能引起归约-归约冲突。

I_k:{[A→α•,u_1] [B→β•aγ,b] } a∩u_1=∅

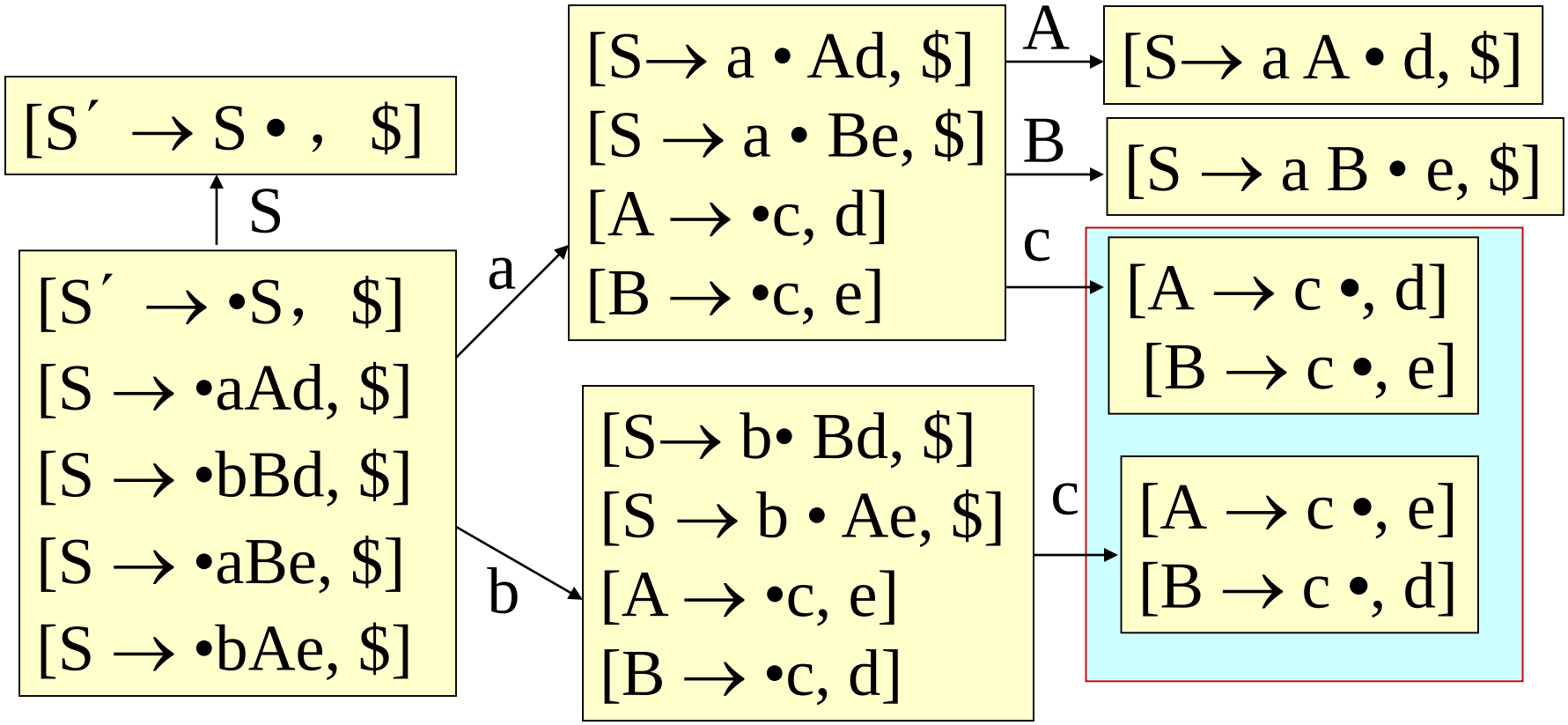
I_j:{[A→α•,u_2] [B→β•aγ,c] } a∩u_2=∅

I_{kj}:{[A→α•,u_1/u_2] [B→β•aγ,b/c] } a∩{u_1, u_2}=∅

非LALR(1)的LR(1)文法

例： $S' \rightarrow S$　　$S \rightarrow aAd \mid bBd \mid aBe \mid bAe$

$A \rightarrow c$　　$B \rightarrow c$



LR(1)和LALR(1)在分析上的差别

例4_9 G(S′):　(0) S′→S　(1) S →CC

(2) C →cC　(3) C →d

输入：ccd$

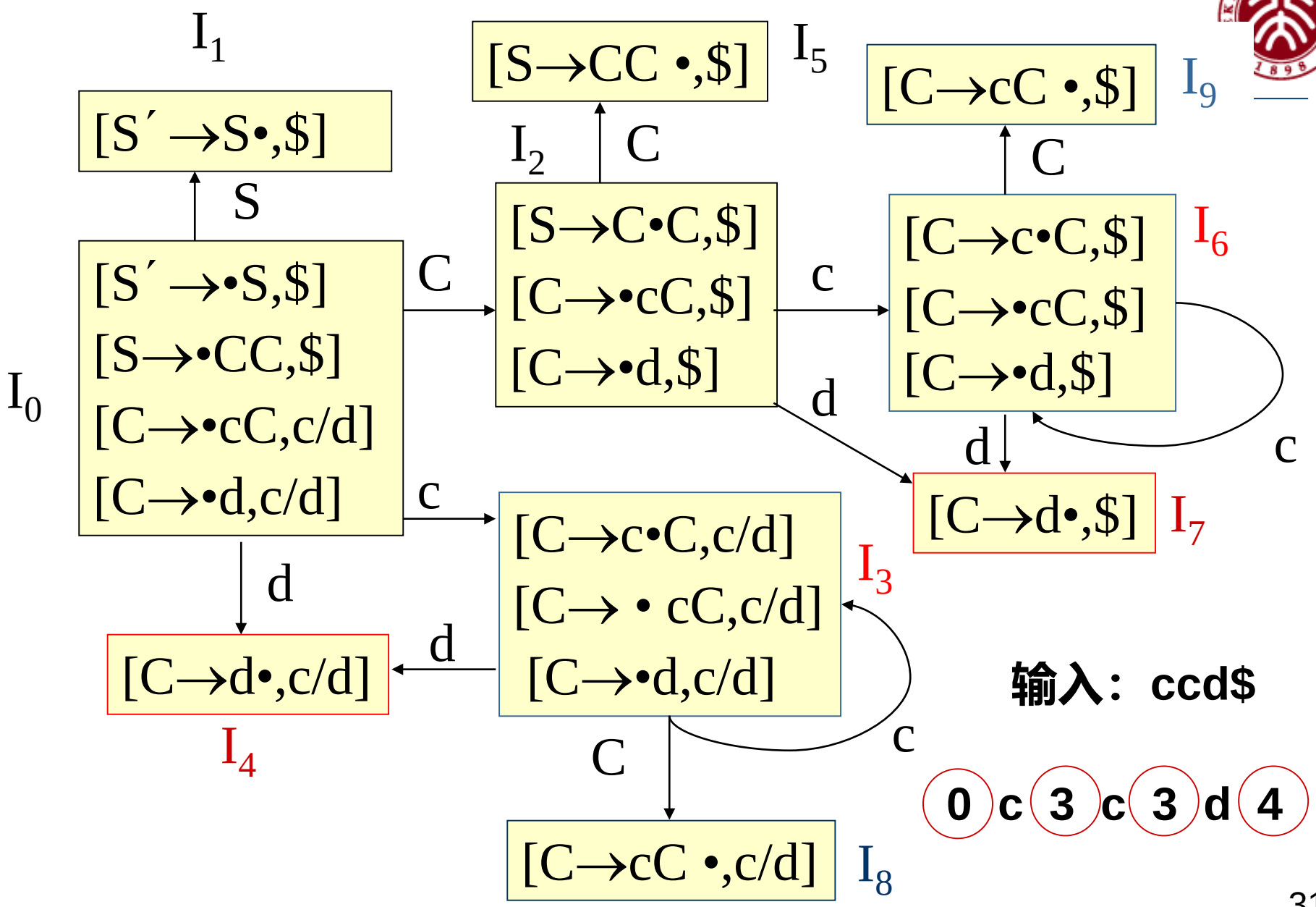
LR:　0 c 3 c 3 d 4　　报错

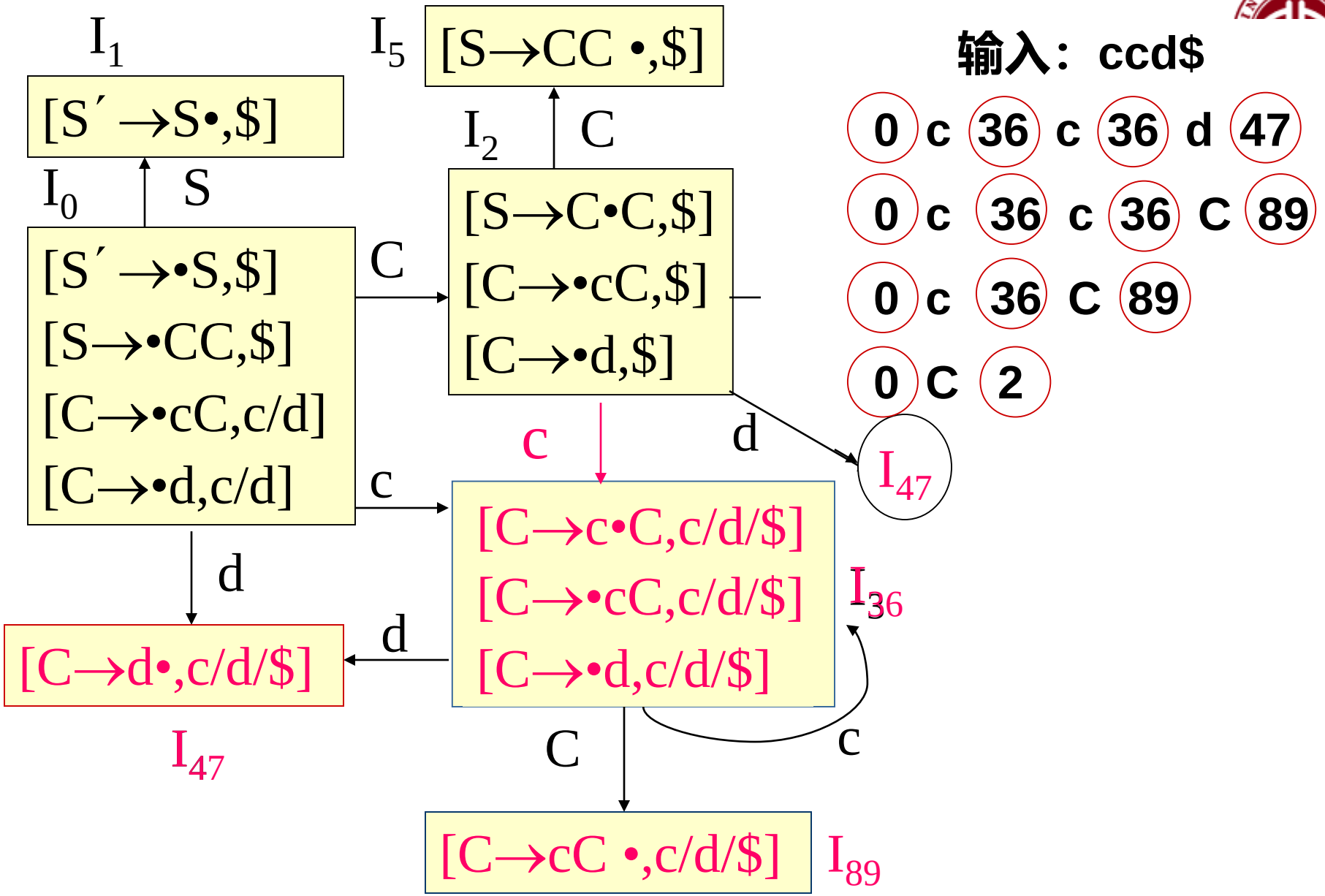
LALR:　0 c 36 c 36 d 47

0 c 36 c 36 C 89

0 c 36 C 89

0 C 2　　报错


I1
[S´→S•,$]
S
I0
[S´→•S,$]
[S→•CC,$]
[C→•cC,c/d]
[C→•d,c/d]
C
c
d
I2
[S→C•C,$]
[C→•cC,$]
[C→•d,$]
I5
[S→CC •,$]
C
c
d
I6
[C→c•C,$]
[C→•cC,$]
[C→•d,$]
I9
[C→cC •,$]
C
c
d
I7
[C→d•,$]
I3
[C→c•C,c/d]
[C→ • cC,c/d]
[C→•d,c/d]
c
d
C
I4
[C→d•,c/d]
I8
[C→cC •,c/d]
输入：ccd$
0 c 3 c 3 d 4


I1
[S′→S•,$]
I0
S
[S′→•S,$]
[S→•CC,$]
[C→•cC,c/d]
[C→•d,c/d]
I5
[S→CC •,$]
I2
C
[S→C•C,$]
[C→•cC,$]
[C→•d,$]
C
c
d
I47
[C→c•C,c/d/$]
[C→•cC,c/d/$]
[C→•d,c/d/$]
I36
c
d
[C→d•,c/d/$]
I47
C
[C→cC •,c/d/$]
I89
输入：ccd$
0 c 36 c 36 d 47
0 c 36 c 36 C 89
0 c 36 C 89
0 C 2

二义性文法的使用

- 二义性文法都不是LR的。
- 某些二义性文法是有用的
  - 可以简洁地描述某些结构；
  - 隔离某些语法结构，对其进行特殊处理。
- 对于某些二义性文法
  - 可以通过消除二义性规则来保证每个句子只有一棵语法分析树。
  - 且可以在LR分析器中实现这个规则。

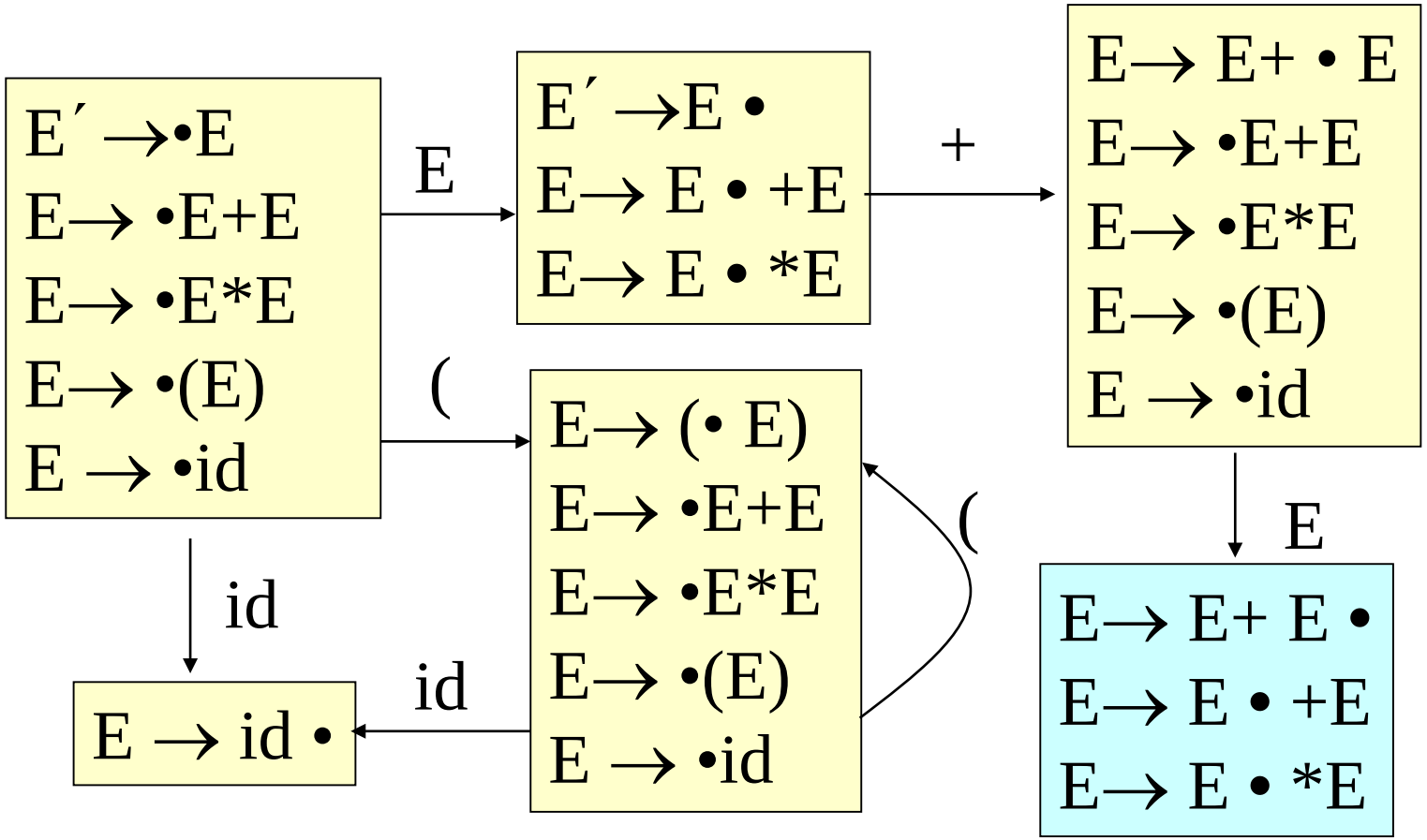
利用优先级/结合性消除冲突

- 二义性文法：**E→E+E | E*E | (E) | id**
- 等价于：**E→E+T | T　　T→T*F | F**
  F→(E)|id
- 二义性文法的优点：
  - 容易修改算符的优先级和结合性。
  - 简洁：如果有多个优先级，那么无二义性文法讲引入太多的非终结符号
  - 高效：不需要处理E→T这样的归约。

LR分析方法对二义文法的应用

龙书4.8节

表达式文法：E → E+E | E*E | (E) | id



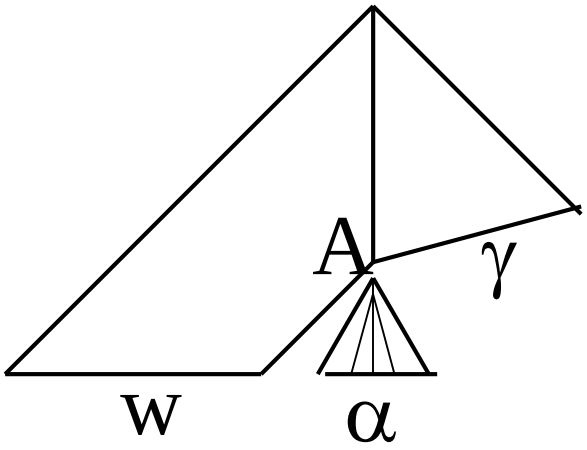
LR分析总结：四种LR分析的对比

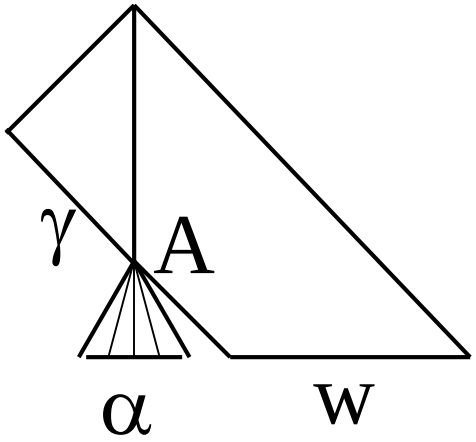
- 如果构造LR(0)的DFA
  - 没有归约冲突就是LR(0)文法
  - 有冲突但是可以通过Follow集合解决冲突就是SLR文法
  - 否则不是SLR文法。
- 如果构造LR(1)的DFA
  - 没有冲突就是LR(1)文法
  - 如果合并同心集之后也没有冲突，那么就是LALR(1)文法。
- 包含关系：LR(0) < SLR < LALR < LR(1)

LR(k)和LL(k)的比较（1）

1. $A \rightarrow \alpha_1 \mid \alpha_2$

LL(k)根据FIRST(α_i)确定使用哪条产生式；而LR(k)是在识别出整个α后，再往后看k个符号，然后确定使用哪条产生式。





LR(k)和LL(k)的比较（2）

2. LL(k)文法都是LR(k)文法。都能用形式化方法实现。

3. 存在非LR的结构

例：$L=\{ww^R \mid w\in\{a,b\}^*\}$

$G[S]: S\rightarrow aSa \mid bSb \mid \varepsilon$

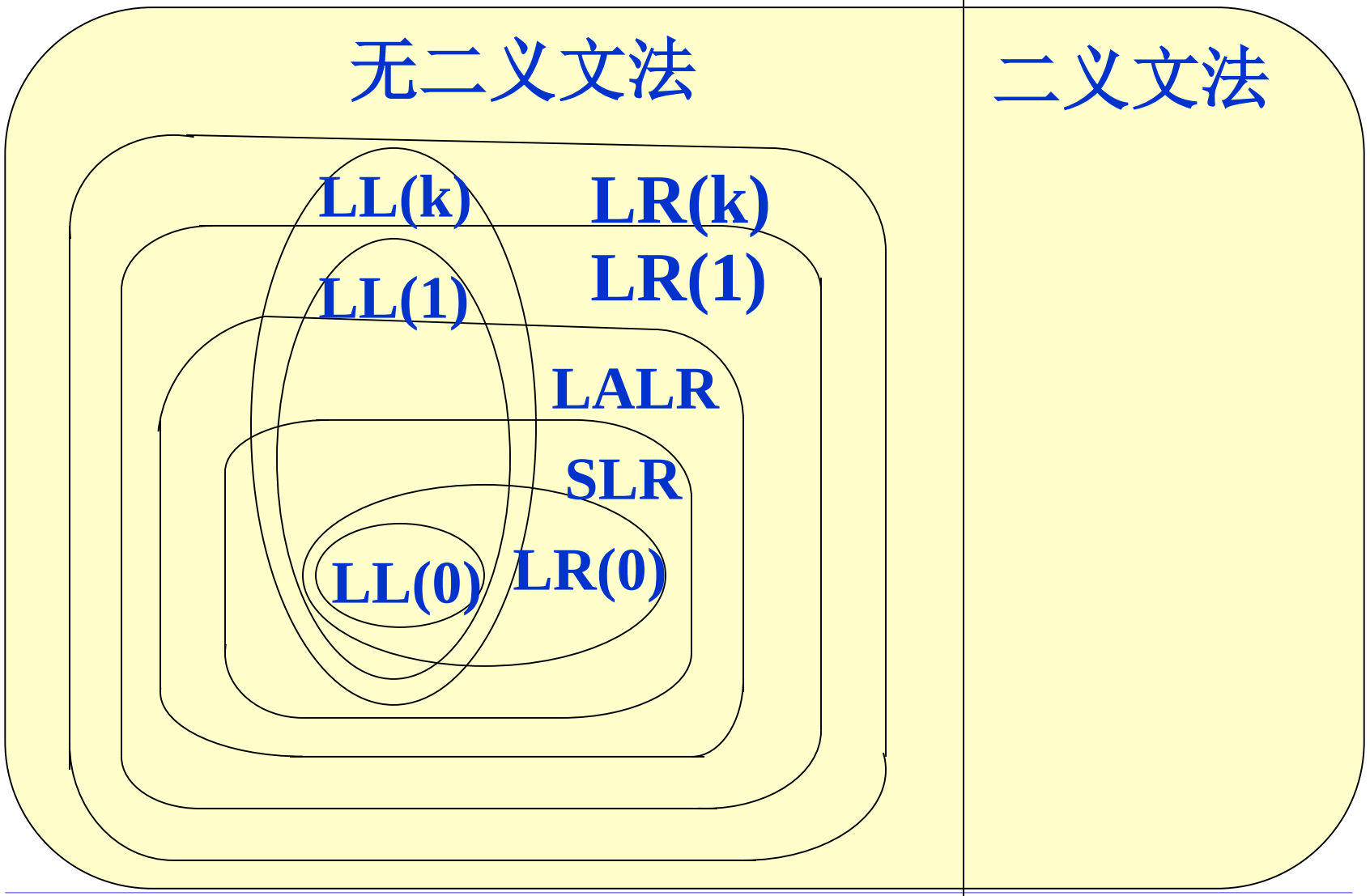
4. LR(k)分析用手工构造是不可能的

- 类Pascal语言的LR(1)分析表，需要数千个状态；
- 由于存在自动生成工具(比如YACC)，LR分析受到广泛重视。

本章小结

- 上下文无关文法
- 自顶向下分析
  - 递归预测分析（递归子程序法）
  - 非递归预测分析——LL(1)
  - 注意：首先消除左递归和提取左公因子。
- 自底向上分析
  - LR分析: SLR(1), LR(1), LALR(1)

文法类的谱系



作业

- LR分析
  - 4.5.1, 4.5.2
- LR(0), SLR
  - 4.6.2, 4.6.3, 4.6.5
- LR(1), LALR(1)
  - 4.7.1, 4.7.4, 4.7.5