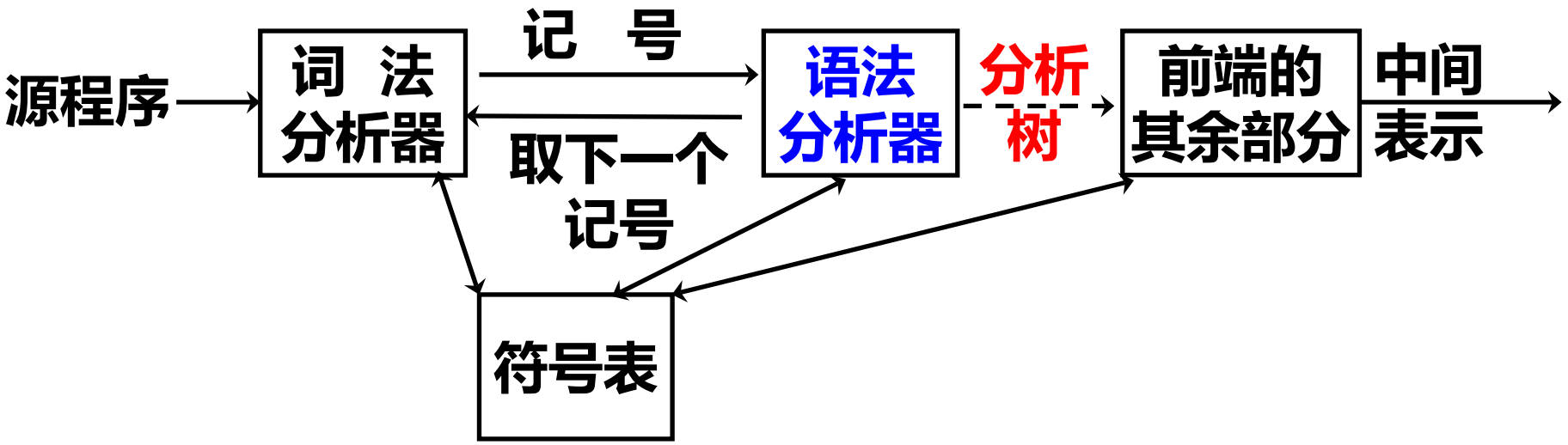
中国科学技术大学
University of Science and Technology of China

USTC, CHINA
ADSLAB
先进数据系统实验室

《编译原理与技术》
语法分析Ⅳ

计算机科学与技术学院
李 诚
2021-09-22

主要内容



- LR(k)分析技术
  - LR分析器的简单模型
    - action, goto函数
  - 简单的LR方法（简称SLR）
    - 活前缀，识别活前缀的DFA/NFA，SLR算法
  - 规范的LR方法
  - 向前看的LR方法（简称LALR）

语法分析的主要方法

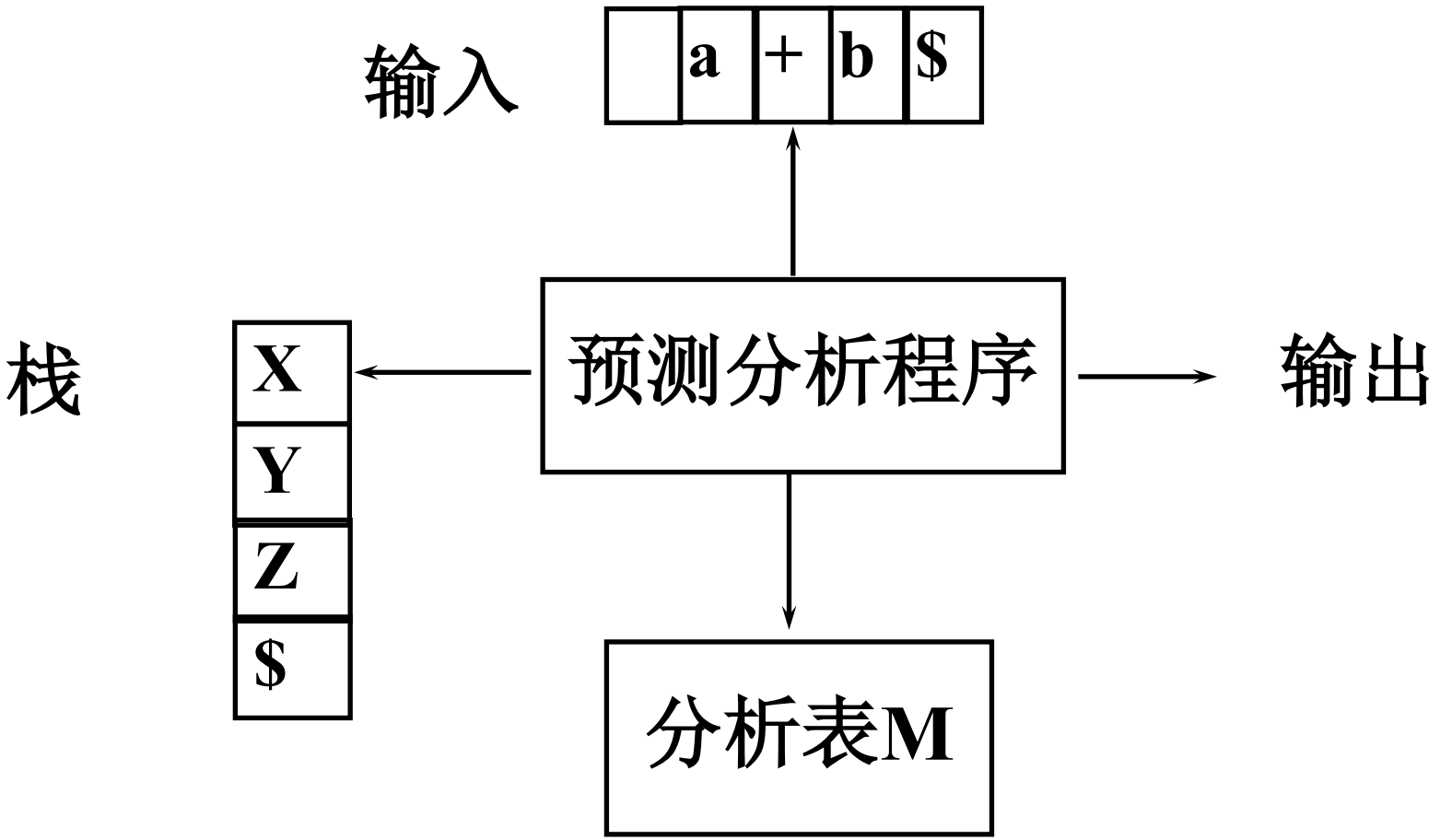
❑自顶向下（Top-down）

- ❖针对输入串，从文法的开始符号出发，尝试根据产生式规则推导（derive）出该输入串。
- ❖LL(1)文法及非递归预测分析方法
- ❖left-to-right scan + leftmost derivation

❑自底向上（Bottom-up）

- ❖针对输入串，尝试根据产生式规则归约（reduce）到文法的开始符号。
- ❖LR(k)文法及其分析器
- ❖left-to-right scan + rightmost derivation

复习：LL(1)非递归分析

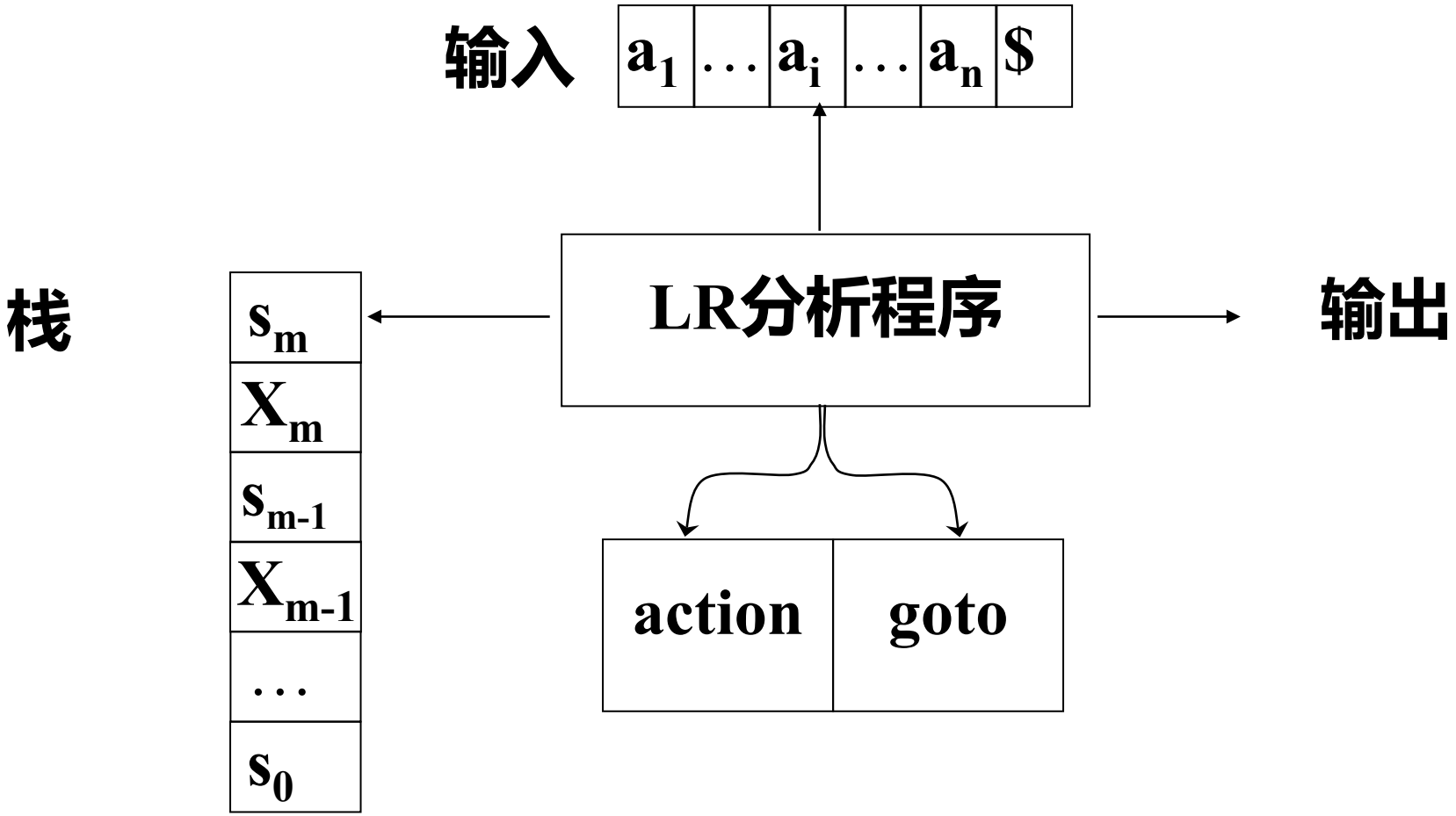


复习：LL(1)非递归分析

❑行：非终结符；列：终结符 或$；单元：产生式

非终结符	输入符号					
	id	+	*	(	)	$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow$ id			$F \rightarrow (E)$		

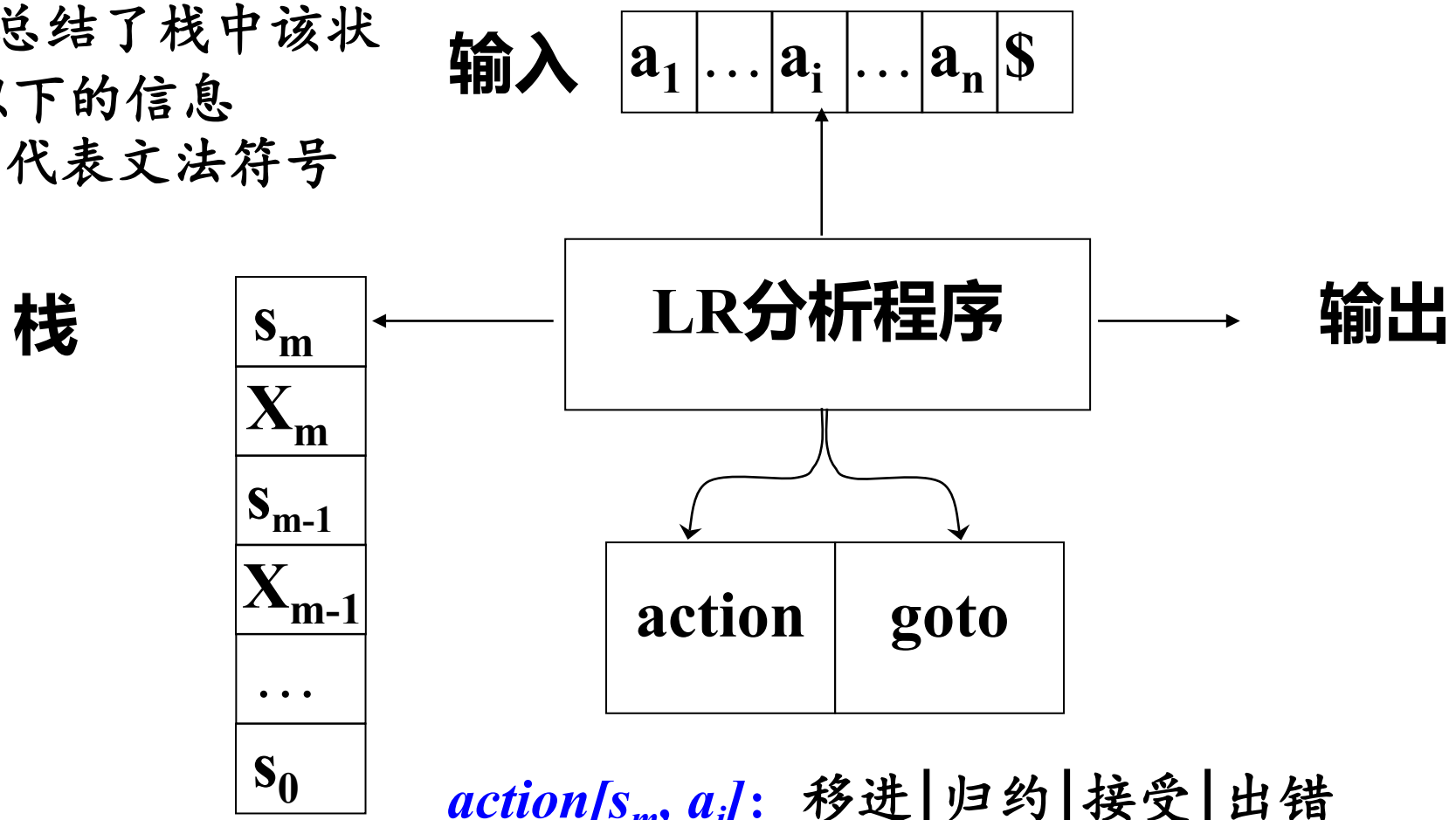
LR分析器



LR分析器

s_j：总结了栈中该状态以下的信息

X_i：代表文法符号



$action[s_m, a_i]$：移进|归约|接受|出错

$goto[s_{m-r}, A]=s_j$：移进A和s_j（归约后使用）

LR分析算法：举例

例 (1) $E \to E + T$ (2) $E \to T$

(3) $T \to T * F$ (4) $T \to F$

(5) $F \to (E)$ (6) $F \to$ id

si 移进当前输入符号和状态*i*

rj 按第*j*个产生式进行归约

acc 接受

状态	动 作 action						转 移 goto		
	id	+	*	(	)	$	*E*	*T*	*F*
0	*s*5			*s*4			1	2	3
1		*s*6				*acc*			
2		*r*2	*s*7		*r*2	*r*2			
3		*r*4	*r*4		*r*4	*r*4			
4	*s*5			*s*4			8	2	3
5		*r*6	*r*6		*r*6	*r*6			
6	*s*5			*s*4				9	3

LR分析算法：举例

栈	输　入	动　作
0	**id * id + id $**	

LR分析算法：举例

栈	输　入	动　作
0	id * id + id $	移进 (查action表)

LR分析算法：举例

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	

LR分析算法：举例

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约

1. 查action[5, *]=>归约
2. 执行归约(F → α):
- 从栈中弹出|α|个<状态, 符号>对
- 查goto[0, F] =>3
- 将(F, 3)压入栈

LR分析算法：举例

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 *F* 3	* id + id $	

LR分析算法：举例

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约

LR分析算法：举例

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	

LR分析算法：举例

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进

LR分析算法：举例

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	

LR分析算法：举例

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进

LR分析算法：举例

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	

LR分析算法：举例

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	按$F \rightarrow$ id归约

LR分析算法：举例

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	按$F \rightarrow$ id归约
0 T 2 * 7 F 10	+ id $	

LR分析算法：举例

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	按$F \rightarrow$ id归约
0 T 2 * 7 F 10	+ id $	按$T \rightarrow T*F$归约

LR分析算法：举例

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	按$F \rightarrow$ id归约
0 T 2 * 7 F 10	+ id $	按$T \rightarrow T*F$归约
. . .	. . .	. . .

LR分析算法：举例

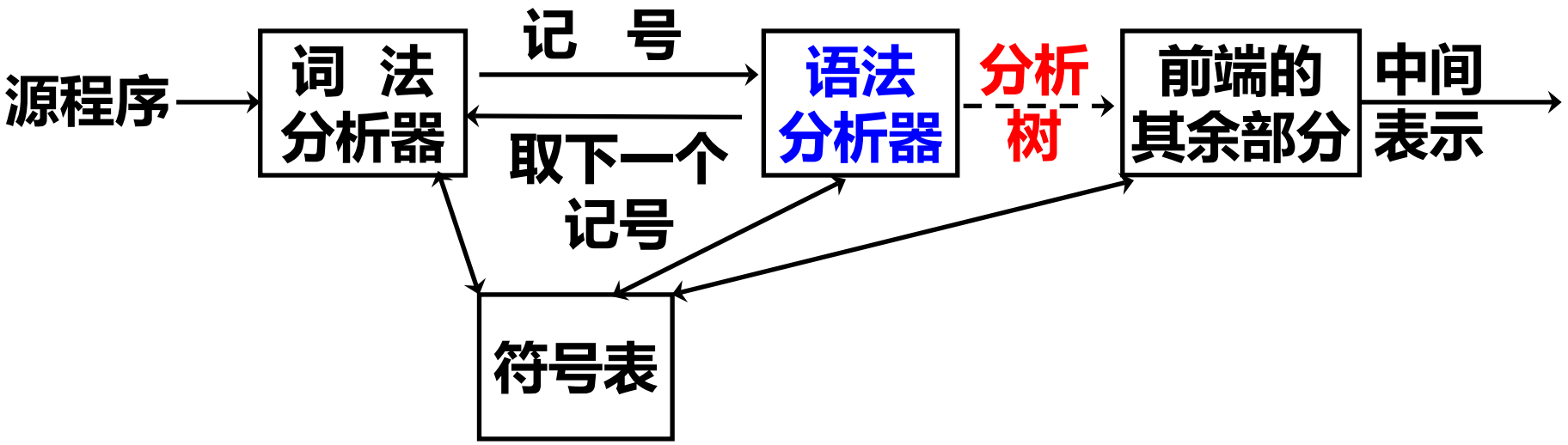
栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	按$F \rightarrow$ id归约
0 T 2 * 7 F 10	+ id $	按$T \rightarrow T*F$归约
. . .	. . .	. . .
0 E 1	$	

LR分析算法：举例

栈	输　入	动　作
0	id * id + id $	移进
0 id 5	* id + id $	按$F \rightarrow$ id归约
0 F 3	* id + id $	按$T \rightarrow F$归约
0 T 2	* id + id $	移进
0 T 2 * 7	id + id $	移进
0 T 2 * 7 id 5	+ id $	按$F \rightarrow$ id归约
0 T 2 * 7 F 10	+ id $	按$T \rightarrow T*F$归约
. . .	. . .	. . .
0 E 1	$	接受

- 关键在于构造LR分析表
  - 计算所有可能的状态
    - 每一个状态描述了语法分析过程中所处的位置
    - 可确定正在分析的产生式集合
    - 可确定句柄形成的中间步骤
  - 明确状态之前的跳转关系
  - 明确状态与输入之间对应的移进或者归约操作

主要内容



- LR(k)分析技术
  - LR分析器的简单模型
  - 简单的LR方法（简称SLR）
    - 活前缀，识别活前缀的DFA/NFA，SLR算法
  - 规范的LR方法
  - 向前看的LR方法（简称LALR）

LR语法分析器的格局

- LR语法分析的每一步都形成一个格局config

$$(s_0X_1s_1X_2s_2\ldots X_ms_m\ ,\quad a_ia_{i+1}\ldots a_n\$)$$

栈的内容　　尚未处理的输入

  - 代表最右句型$X_1X_2\ldots X_ma_ia_{i+1}\ldots a_n$
  - $X_1X_2\ldots X_m$是最右句型的一个前缀

LR语法分析器的格局

- LR语法分析的每一步都形成一个格局config

$$(s_0X_1s_1X_2s_2\ldots X_ms_m,\quad a_ia_{i+1}\ldots a_n\$)$$

栈的内容　　尚未处理的输入

  - 代表最右句型$X_1X_2\ldots X_ma_ia_{i+1}\ldots a_n$
  - $X_1X_2\ldots X_m$是最右句型的一个前缀
  - 每一个前缀都对应一个状态，因此，找出所有可能在栈里出现的前缀，就可以确定所有的状态

LR语法分析器的格局

- LR语法分析的每一步都形成一个格局config

$$(s_0X_1s_1X_2s_2\ldots X_ms_m,\quad a_ia_{i+1}\ldots a_n\$)$$

栈的内容　　尚未处理的输入

  - 代表最右句型$X_1X_2\ldots X_ma_ia_{i+1}\ldots a_n$
  - $X_1X_2\ldots X_m$是最右句型的一个前缀
  - 每一个前缀都对应一个状态，因此，找出所有可能在栈里出现的前缀，就可以确定所有的状态
  - 状态之间的转换 <==> 前缀之间的转换

LR语法分析器的格局

❑LR语法分析的每一步都形成一个格局config

$(s_0X_1s_1X_2s_2\ldots X_ms_m,\quad a_ia_{i+1}\ldots a_n\$)$

栈的内容 **尚未处理的输入**

- ❖代表最右句型$X_1X_2\ldots X_ma_ia_{i+1}\ldots a_n$
- ❖$X_1X_2\ldots X_m$是最右句型的一个前缀
- ❖每一个前缀都对应一个状态，因此，找出所有可能在栈里出现的前缀，就可以确定所有的状态
- ❖状态之间的转换 <==> 前缀之间的转换
- ❖**在栈顶为s，下一个字符为a的格局下，前缀为p**
  - ➢**何时移进？当p包含句柄的一部分且存在p′= pa**
  - ➢**何时归约？当p包含整个句柄时**

LR分析: 基本概念

- **活前缀或可行前缀 (viable prefix):**
  - 最右句型的前缀，该前缀不超过最右句柄的右端

$$S \Rightarrow^*_{rm} \gamma A w \Rightarrow_{rm} \gamma \beta w$$

  - $\gamma\beta$的任何前缀（包括ε和$\gamma\beta$本身）都是活前缀
  - 都出现在栈顶

活前缀

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

栈中可能出现的串：

a
ab
aA
aAb
$aAbc$
aAd
aAB
$aABe$
S

活前缀：
最右句型的前缀，该前缀不超过最右句柄的右端

$$S \Rightarrow^*_{rm} \gamma A w \Rightarrow_{rm} \gamma \beta w$$

$\gamma\beta$的任何前缀（包括ε和$\gamma\beta$本身）都是一个活前缀。

活前缀与句柄的关系

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

栈中可能出现的串：

a
ab ← 出现句柄（对应$A \rightarrow b$）
aA
aAb
$aAbc$ ← 出现句柄（对应$A \rightarrow Abc$）
aAd ← 出现句柄（对应$B \rightarrow d$）
aAB
$aABe$ ← 出现句柄（对应$S \rightarrow aABe$）
S

- **活前缀已含有句柄，表明产生式A→β的右部β已出现在栈顶。**

活前缀与句柄的关系

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

栈中可能出现的串：

a
ab
aA ← 出现产生式$A \rightarrow Abc$右端的一部分，期望从输入串中看到bc
aAb ← 出现产生式A→Abc右端的一部分，期望从输入串中看到c
$aAbc$
aAd
aAB ← 出现产生式S→aABe的右端一部分，期望从输入串中看到e
$aABe$
S

- 活前缀已含有句柄，表明产生式A→β的右部β已出现在栈顶。
- 活前缀只含句柄的一部分符号如β_1表明$A \rightarrow \beta_1\beta_2$的右部子串$\beta_1$已出现在栈顶，当前期待从输入串中看到$\beta_2$推出的符号。

LR分析方法的特点

- 栈中的文法符号总是形成一个活前缀
- 分析表的转移函数本质上是识别活前缀的DFA

下表蓝色部分构成识别活前缀DFA的状态转换表

状态	动作						转移		
	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

LR分析方法的特点

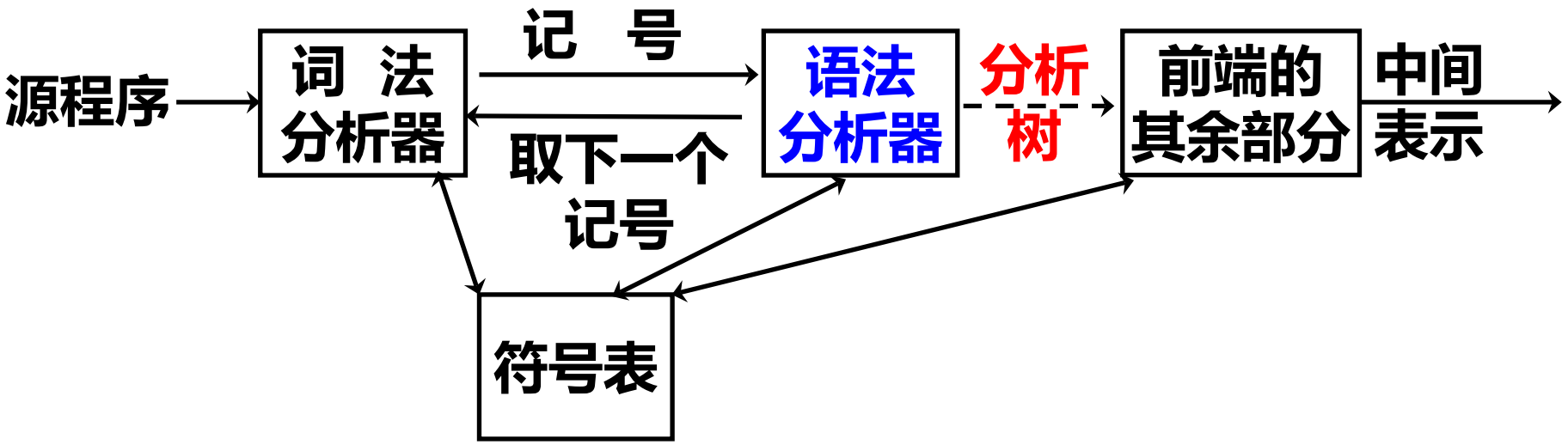
- 栈中的文法符号总是形成一个活前缀
- 分析表的转移函数本质上是识别活前缀的DFA
- 栈顶的状态符号包含确定句柄所需的一切信息

栈	输　入	动　作
0	id * id + id $	移进
…	…	…
0 *T* 2 * 7	id + id $	移进
0 *T* 2 * 7 id 5	+ id $	按$F \rightarrow$ id归约
0 *T* 2 * 7 *F* 10	+ id $	按$T \rightarrow T*F$归约

LR分析方法的特点

- 栈中的文法符号总是形成一个活前缀
- 分析表的转移函数本质上是识别活前缀的DFA
- 栈顶的状态符号包含确定句柄所需的一切信息
- 是已知的最一般的无回溯的移进-归约方法
- 能分析的文法类是预测分析法能分析的文法类的真超集
- 能及时发现语法错误
- 手工构造分析表的工作量太大

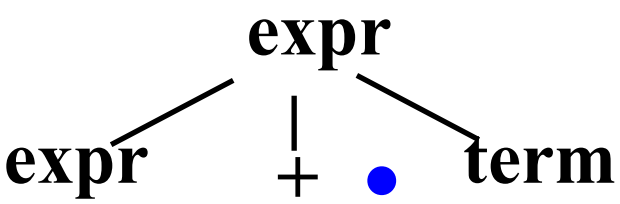
主要内容



- LR(k)分析技术
  - LR分析器的简单模型
  - 简单的LR方法（简称SLR）
    - 活前缀，识别活前缀的DFA/NFA，SLR算法
  - 规范的LR方法
  - 向前看的LR方法（简称LALR）

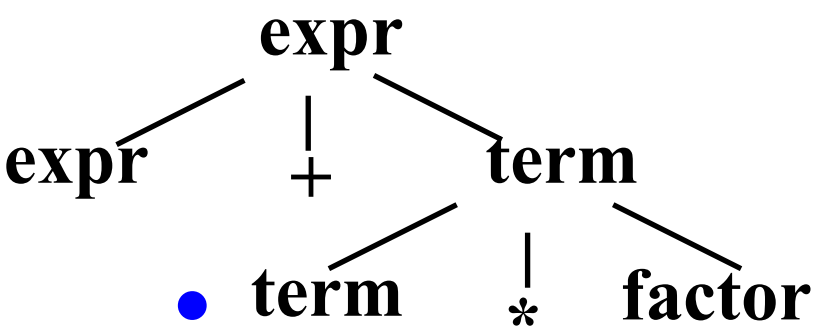
SLR分析表的构造

- SLR (Simple LR)
- LR(0)项目（简称项目）
  - 在右部的某个地方加点的产生式
  - 加点的目的是用来表示分析过程中的状态



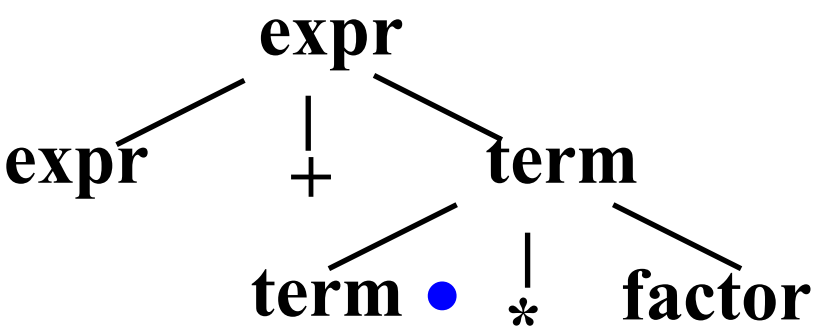
SLR分析表的构造

- SLR (Simple LR)
- LR(0)项目（简称项目）
  - 在右部的某个地方加点的产生式
  - 加点的目的是用来表示分析过程中的状态



SLR分析表的构造

- SLR (Simple LR)
- LR(0)项目（简称项目）
  - 在右部的某个地方加点的产生式
  - 加点的目的是用来表示分析过程中的状态



SLR分析表的构造

- SLR (Simple LR)
- LR(0)项目（简称项目）
  - 在右部的某个地方加点的产生式
  - 加点的目的是用来表示分析过程中的状态

项代表了一个可能的前缀

- 例 $A \to XYZ$对应有四个项目

$$A \to \cdot XYZ$$
$$A \to X \cdot YZ$$
$$A \to XY \cdot Z$$
$$A \to XYZ \cdot$$

点的左边代表历史信息，点的右边代表展望信息。

- 例 $A \to \varepsilon$只有一个项目和它对应

$$A \to \cdot$$

SLR分析表的构造

- 从文法构造识别活前缀的DFA
- 从上述DFA构造分析表

1. 拓（增）广文法（augmented grammar）

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid \mathbf{id}$$

1. 拓（增）广文法（augmented grammar）

当且仅当分析器使用$E' \to E$归约时，宣告分析成功

$$E' \to E$$

$$E \to E + T \mid T$$

$$T \to T * F \mid F$$

$$F \to (E) \mid \mathbf{id}$$

2. 构造LR(0)项目集规范族

项集族是若干可能前缀的集合，对应DFA的状态

I_0:

$E' \rightarrow \cdot E$

2. 构造LR(0)项目集规范族

I_0:

$E' \rightarrow \cdot E$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

求项目集的闭包closure(I)
闭包函数closure(I)
1、I的每个项目均加入closure(I)
2、如果A→α·Bβ在 closure(I)中，且B→γ是产生式，那么如果项目B→ ·γ还不在closure(I)中的话，那么把它加入。

2. 构造LR(0)项目集规范族

I_0:

$E' \rightarrow \cdot E$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

求项目集的闭包closure(I)
闭包函数closure(I)
1、I的每个项目均加入closure(I)
2、如果A→α·Bβ在 closure(I)中，且B→γ是产生式，那么如果项目B→ ·γ还不在closure(I)中的话，那么把它加入。

2. 构造LR(0)项目集规范族

I_0:

$E' \rightarrow \cdot E$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow \cdot \mathrm{id}$

求项目集的闭包closure(I)
闭包函数closure(I)
1、I的每个项目均加入closure(I)
2、如果A→α·Bβ在 closure(I)中，且B→γ是产生式，那么如果项目B→ ·γ还不在closure(I)中的话，那么把它加入。

2. 构造LR(0)项目集规范族

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot \mathrm{id}$

核心项目:初始项目($E' \to \cdot E$)或者点不在最左边的项

非核心项目：不是初始项，且点在最左边

可以通过对核心项目求闭包来获得
为节省存储空间，可省去

2. 构造LR(0)项目集规范族

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E$

$F \to \cdot id$

E

I_1:

$E' \to E \cdot$

$E \to E \cdot + T$

$I_1 := goto(I_0, E)$

求项目集I和文法符号X的I′ = goto(I, X)
(当输入为X时离开I状态后的转换)
1、I′ = ∅
2、对于I中的每一项A→α·Xβ,
I′ = I′ ∪ closure(A→αX·β)

2. 构造LR(0)项目集规范族

I_0:

$E' \rightarrow \cdot E$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow \cdot \mathrm{id}$

E

I_1:

$E' \rightarrow E \cdot$

$E \rightarrow E \cdot + T$

T

I_2:

$E \rightarrow T \cdot$

$T \rightarrow T \cdot * F$

2. 构造LR(0)项目集规范族

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot \mathrm{id}$

E → I_1:

$E' \to E \cdot$

$E \to E \cdot + T$

T → I_2:

$E \to T \cdot$

$T \to T \cdot * F$

F → I_3:

$T \to F \cdot$

2. 构造LR(0)项目集规范族

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot \mathrm{id}$

$\xrightarrow{(}$

I_4:

$F \to (\cdot E)$

2. 构造LR(0)项目集规范族

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot \mathbf{id}$

$\xrightarrow{(}$

I_4:

$F \to (\cdot E)$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

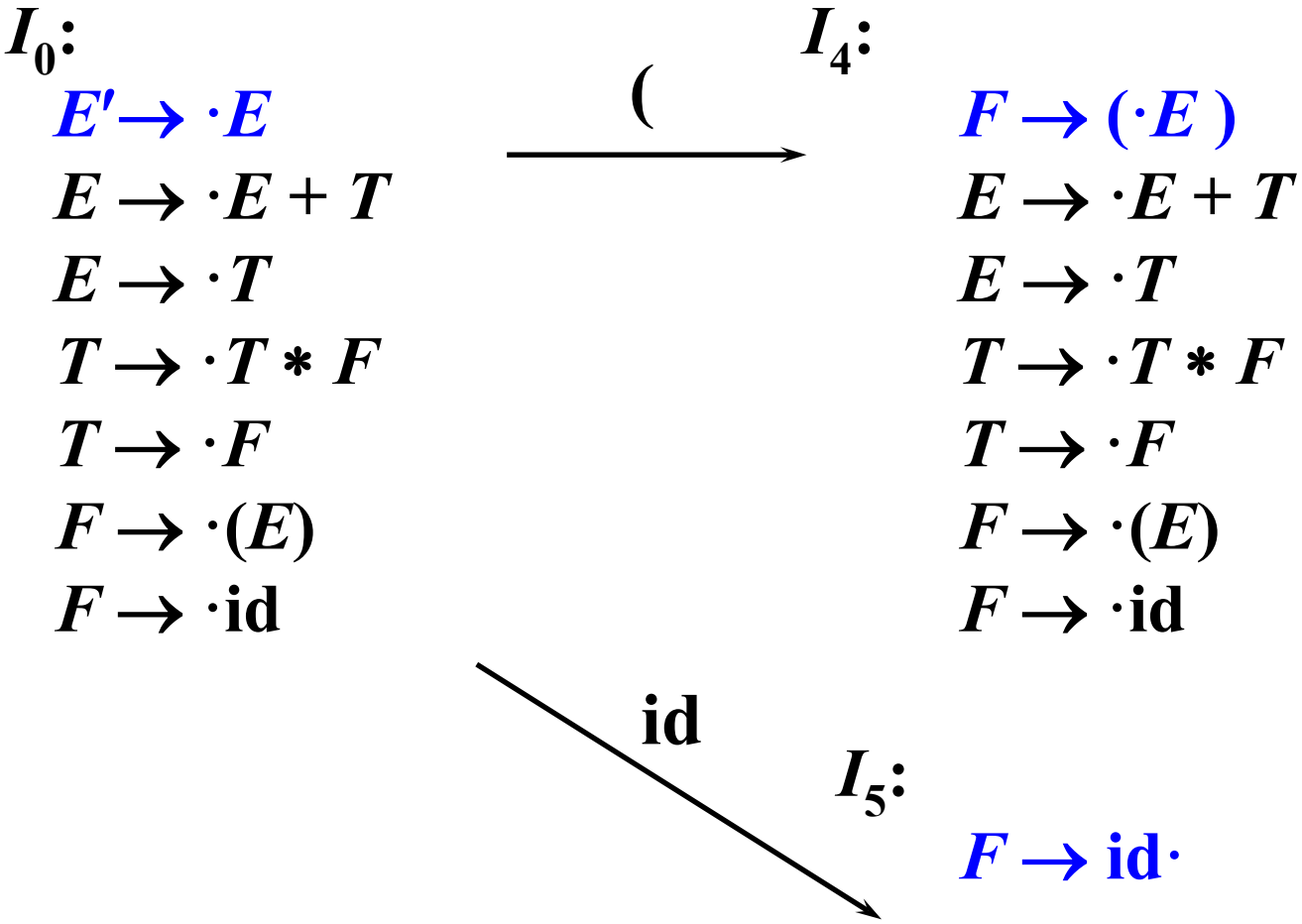
$T \to \cdot F$

$F \to \cdot (E)$

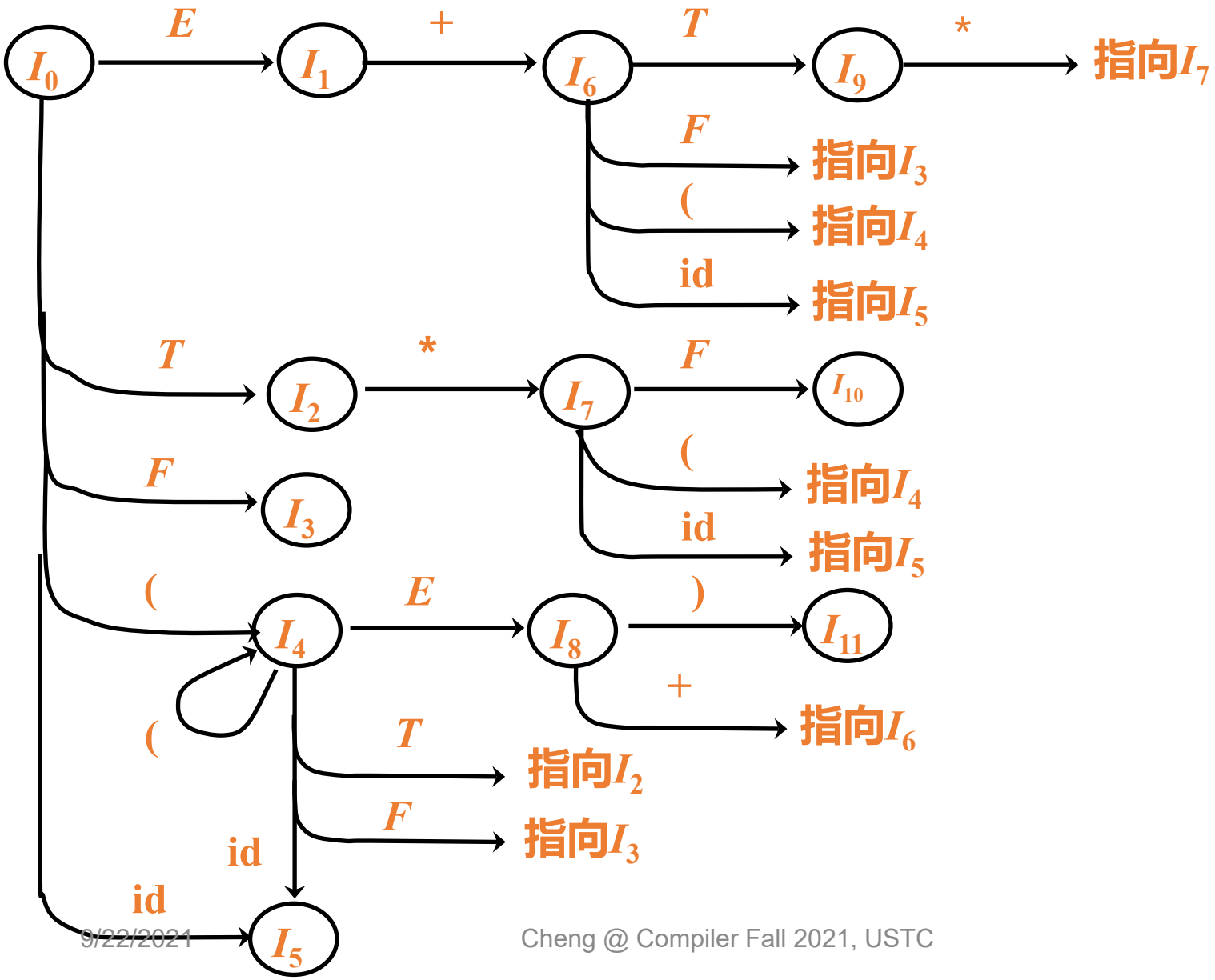
$F \to \cdot \mathbf{id}$

构造识别活前缀的DFA

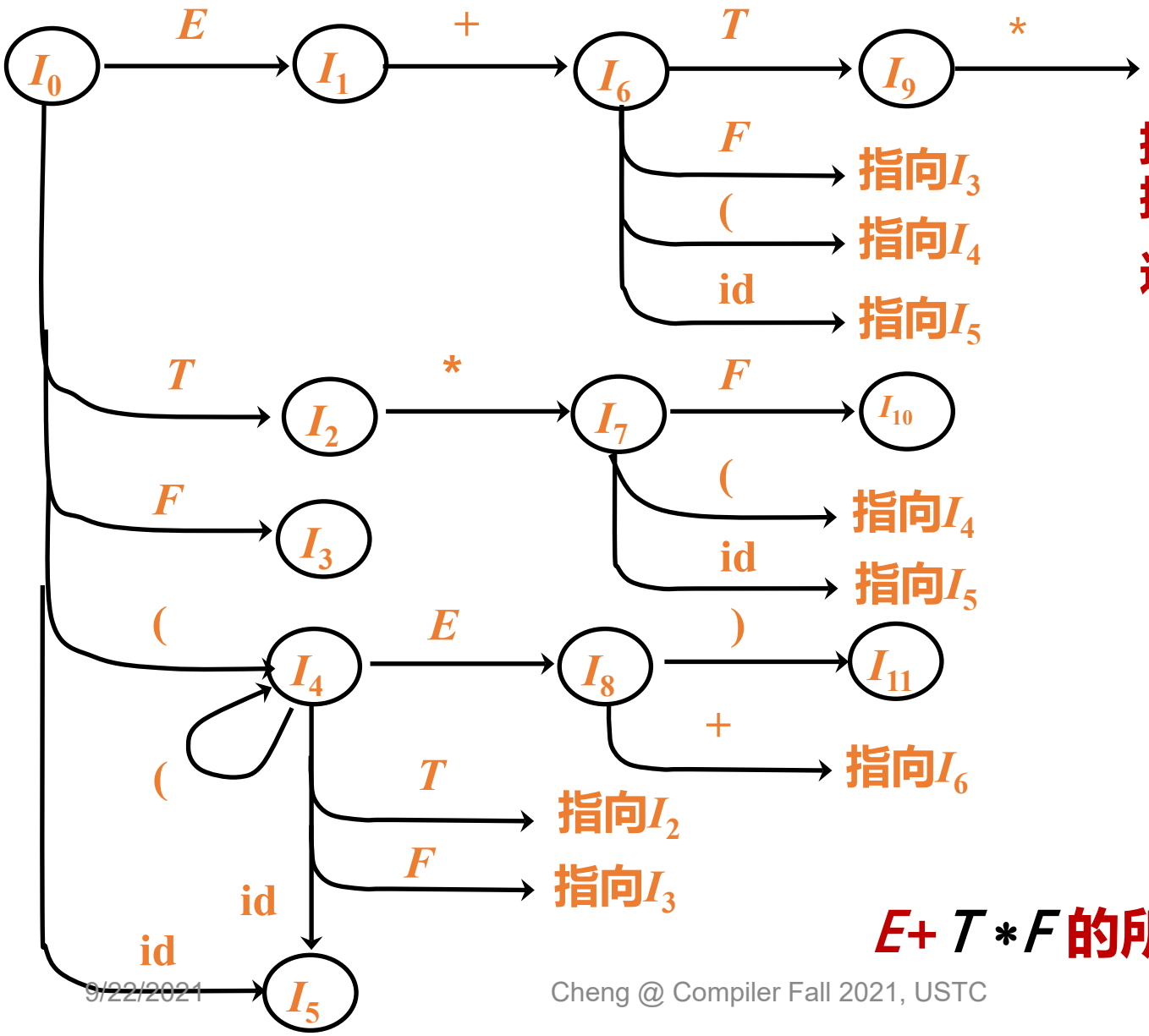
2. 构造LR(0)项目集规范族



构造识别活前缀的DFA



构造识别活前缀的DFA



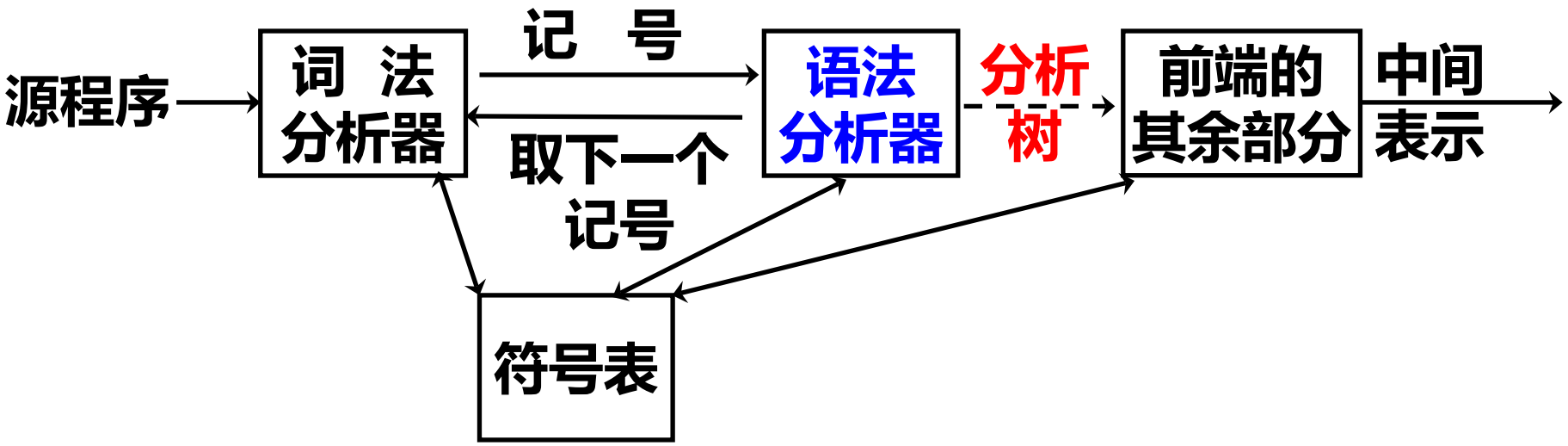
把所有状态都作为接受状态

这是一个DFA

$$E' \Rightarrow E$$
$$\Rightarrow E+T$$
$$\Rightarrow E+T*F$$
$$\Rightarrow E+T*\text{id}$$
$$\Rightarrow E+T*F*\text{id}$$

$E+T*F$ 的所有前缀都可接受

主要内容



- LR(k)分析技术
  - LR分析器的简单模型
  - 简单的LR方法（简称SLR）
    - 活前缀，识别活前缀的DFA/NFA，SLR算法
  - 规范的LR方法
  - 向前看的LR方法（简称LALR）

SLR分析表的构造

- 从文法构造识别活前缀的DFA
- 从上述DFA构造分析表

从DFA构造SLR分析表

- 状态 i 从 I_i 构造，它的 $action$ 函数如下确定：
  - 如果 $[A \to \alpha \cdot a\beta]$ 在 I_i 中，并且 $goto(I_i, a) = I_j$，那么置 $action[i, a]$ 为 sj
  - 如果 $[A \to \alpha \cdot]$ 在 I_i 中，那么对 $FOLLOW(A)$ 中的所有 a，置 $action[i, a]$ 为 rj，j 是产生式 $A \to \alpha$ 的编号
  - 如果 $[S' \to S \cdot]$ 在 I_i 中，那么置 $action[i, \$]$ 为接受 acc
  - 上面的 a 是终结符

- 如果出现动作冲突，那么该文法就不是SLR(1)文法

从DFA构造SLR分析表

- 状态 i 从 I_i 构造，它的*action*函数如下确定：
  - 此处省略，参见上页
- 使用下面规则构造状态i的*goto*函数：
  - 对所有的非终结符A，如果$goto(I_i, A) = I_j$，那么 $goto[i, A] = j$

从DFA构造SLR分析表

- 状态 i 从 I_i 构造，它的*action*函数如下确定：
  - 此处省略，参见上页
- 使用下面规则构造状态i的*goto*函数：
  - 此处省略，参见上页
- 分析器的初始状态是包含$[S' \rightarrow \cdot S]$的项目集对应的状态

不能由上面两步定义的条目都置为error

SLR分析器

例 (1) $E \to E + T$ (2) $E \to T$

(3) $T \to T * F$ (4) $T \to E$

(5) $F \to (E)$ (6) $F \to \text{id}$

si 移进当前输入符号和状态*i*

rj 按第*j*个产生式进行归约

acc 接受

状态	动 作 action						转 移 goto		
	id	+	*	(	)	$	*E*	*T*	*F*
0	*s*5			*s*4			1	2	3
1		*s*6				*acc*			
2		*r*2	*s*7		*r*2	*r*2			
3		*r*4	*r*4		*r*4	*r*4			
4	*s*5			*s*4			8	2	3
5		*r*6	*r*6		*r*6	*r*6			
6	*s*5			*s*4				9	3

SLR(1)文法

- 一个上下文无关文法G，通过上述算法构造出SLR语法分析表，且表项中没有移进/归约或者归约/归约冲突，那么G就是SLR(1)文法。
- 1代表了当看到某个产生式右部时，只需要再向前看1个符号就可决定是否用该式进行归约。
- 通常可以省略1，写作SLR文法

判定满足SLR文法输入串

- 依据上述SLR(1)分析表
- 参照slide 9 -25的分析方法
  - 文法符号栈
  - 输入缓冲区
  - 选择的行为
    - 移进、归约、接受、报错

❑例 I_2:

(2)$E \rightarrow T\cdot$

(3)$T \rightarrow T\cdot * F$

❖归约：因为FOLLOW(E) = {\$, +,)},
所以 $action[2, \$]=action[2, +]=action[2,)]=r2$

❖移进：因为圆点在中间，且点后面是终结符，
所以，$action[2, *] = s7$

活前缀的概念-revisit

- LR(0)自动机刻画了可能出现在文法符号栈中的所有串；
- 栈中的内容一定是某个最右句型的前缀；
- 但是不是所有前缀都会出现在栈中。

$$E \Rightarrow^*_{rm} F * \mathbf{id} \Rightarrow_{rm} (\mathrm{E}) * \mathbf{id}$$

活前缀的概念-revisit

- LR(0)自动机刻画了可能出现在文法符号栈中的所有串；
- 栈中的内容一定是某个最右句型的前缀；
- 但是不是所有前缀都会出现在栈中。

$$E \Rightarrow^*_{rm} F * \mathbf{id} \Rightarrow_{rm} (E) * \mathbf{id}$$

- 栈中只能出现(, (*E*, (*E*)，而不会出现(*E*)*
  - 因为看到*时，(*E*)是句柄，会被归约成为*F*

活前缀的概念-revisit

- 活前缀或可行前缀（viable prefix）：
  - 最右句型的前缀，该前缀不超过最右句柄的右端

$$S \Rightarrow^*_{rm} \gamma A w \Rightarrow_{rm} \gamma \beta w$$

  - $\gamma\beta$的任何前缀（包括ε和$\gamma\beta$本身）都是活前缀
  - 都出现在栈顶

LR(0) 自动机能够识别活前缀
SLR(1)分析器正是基于这个事实

有效项目

- 如果$S' \Rightarrow^*_{rm} \alpha Aw \Rightarrow_{rm} \alpha\beta_1\beta_2 w$，那么就说项目$A \rightarrow \beta_1 \cdot \beta_2$对活前缀$\alpha\beta_1$是有效的

有效项目

❑如果$S' \Rightarrow^*_{rm} \alpha Aw \Rightarrow_{rm} \alpha\beta_1\beta_2 w$，那么就说项目$A \to \beta_1 \cdot \beta_2$对活前缀$\alpha\beta_1$是有效的

❖一个项目可能对好几个活前缀都是有效的

$E' \to E$　　$E \to E + T \mid T$

$T \to T * F \mid F$　　$F \to (E) \mid \mathrm{id}$

项$E \to \cdot E+T$对ε和(这两个活前缀都有效

$E' \Rightarrow E \Rightarrow E+T$　　(α，β_1都为空)

$E' \Rightarrow E \Rightarrow (E) \Rightarrow (E+T)$　　(α="("，β_1为空)

该DFA读过ε和(后到达不同的状态，
那么项目$E \to \cdot E+T$就出现在对应的不同项目集中

有效项目

- 如果$S' \Rightarrow^*_{rm} \alpha Aw \Rightarrow_{rm} \alpha\beta_1\beta_2 w$，那么就说项目$A \rightarrow \beta_1 \cdot \beta_2$对活前缀$\alpha\beta_1$是有效的
  - 一个项目可能对好几个活前缀都是有效的
    - 如果$\beta_2 \neq \varepsilon$，应该移进
    - 如果$\beta_2 = \varepsilon$，应该用产生式$A \rightarrow \beta_1$归约

有效项目

- 如果$S' \Rightarrow^*_{rm} \alpha Aw \Rightarrow_{rm} \alpha\beta_1\beta_2 w$，那么就说项目$A \rightarrow \beta_1 \cdot \beta_2$对活前缀$\alpha\beta_1$是有效的
  - 一个项目可能对好几个活前缀都是有效的
  - 一个活前缀可能有多个有效项目

一个活前缀γ的有效项目集就是
从这个DFA的初态出发，沿着标记为γ的路径到达的那个项目集（状态）

- 例 串$E + T *$是活前缀，读完它后，DFA处于状态I_7

I_7: $T \to T * \cdot F$, $F \to \cdot(E)$, $F \to \cdot \mathrm{id}$

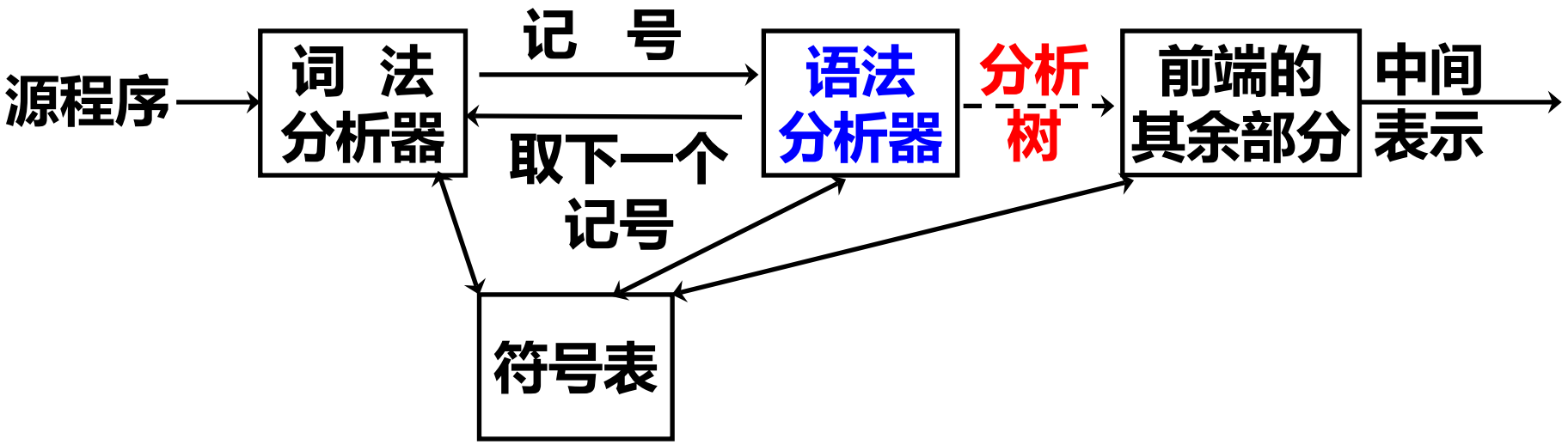
$E' \Rightarrow E \Rightarrow E+T \Rightarrow E+T * F \Rightarrow E+T * \mathrm{id} \Rightarrow E+T * F * \mathrm{id}$

$E' \Rightarrow E \Rightarrow E+T \Rightarrow E+T * F \Rightarrow E+T * (E) \Rightarrow E+T * \mathrm{id}$

$E' \Rightarrow E \Rightarrow E+T \Rightarrow E+T * F \Rightarrow E+T * \mathrm{id}$

包含活前缀的最右推导，且
I_7中所有的项目对该活前缀是有效的

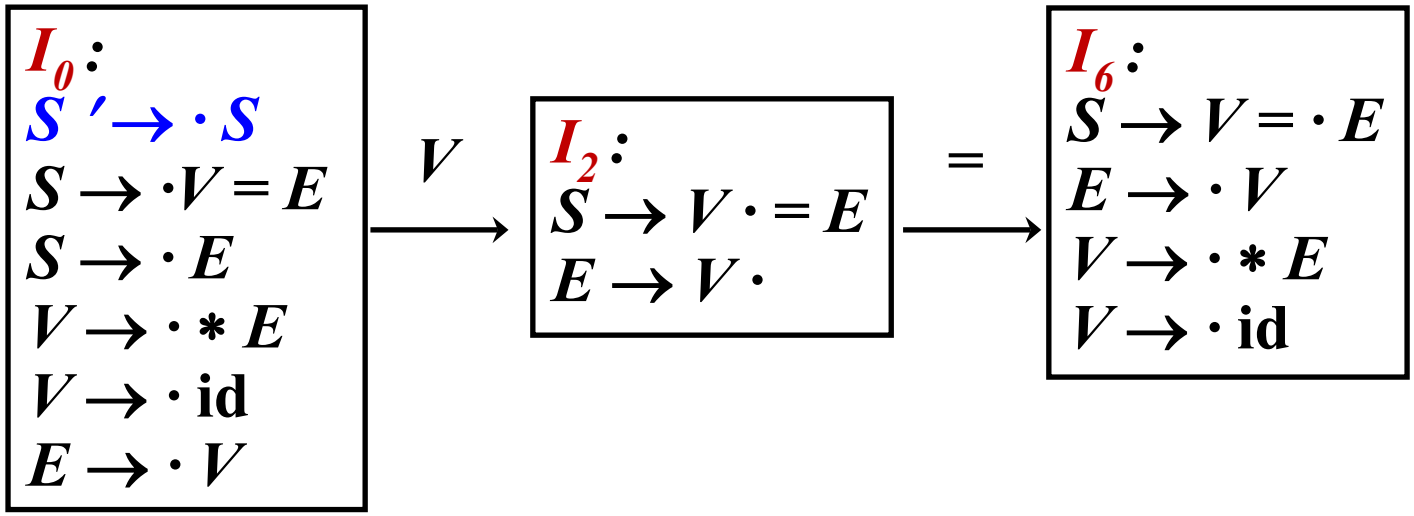
主要内容



- ❑LR(k)分析技术
  - ❖LR分析器的简单模型
  - ❖简单的LR方法（简称SLR）
    - ➢活前缀，识别活前缀的DFA/NFA，SLR算法
  - ❖规范的LR方法
  - ❖向前看的LR方法（简称LALR）

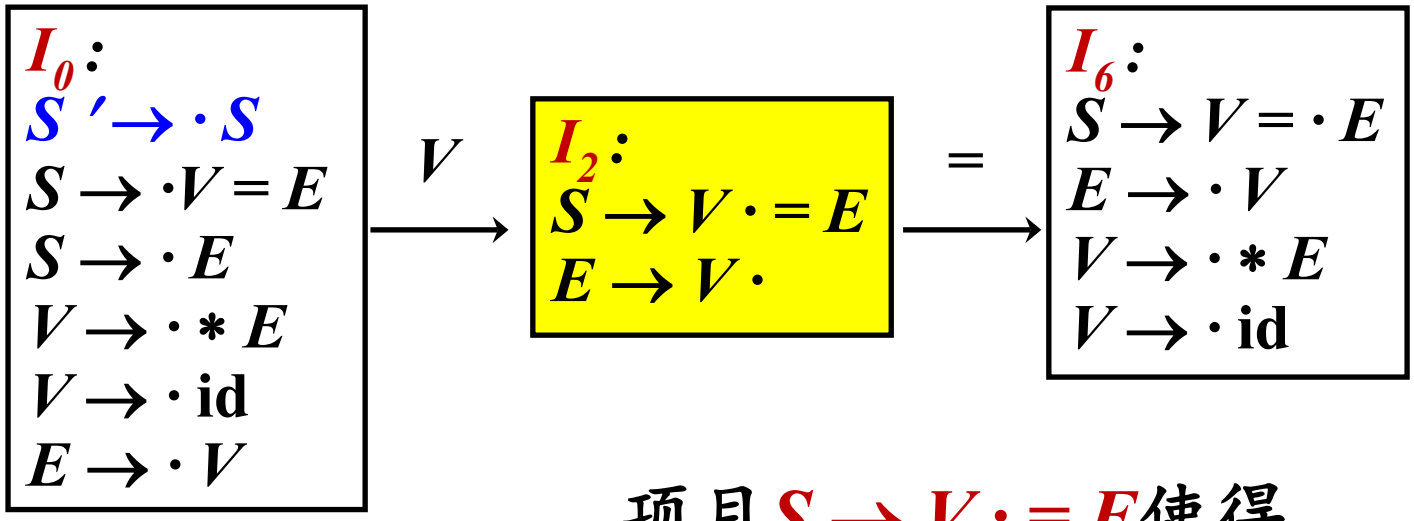
SLR(1)文法的描述能力有限

$S \to V = E$
$S \to E$
$V \to * E$
$V \to \text{id}$
$E \to V$



SLR(1)文法的描述能力有限

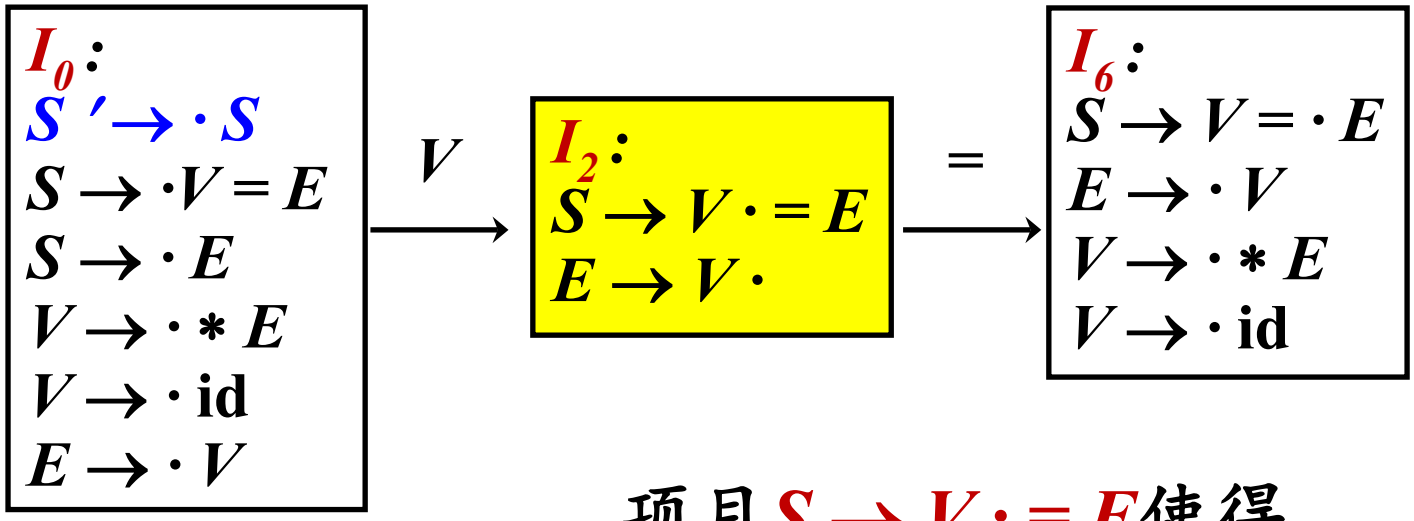
$S \to V = E$
$S \to E$
$V \to * E$
$V \to \text{id}$
$E \to V$



项目$S \to V \cdot = E$使得
action[2, =] = s6

SLR(1)文法的描述能力有限

$S \to V = E$
$S \to E$
$V \to * E$
$V \to \text{id}$
$E \to V$



项目$S \to V \cdot = E$使得
action[2, =] = s6

项目$E \to V \cdot$使得
action[2, =] = r5
因为Follow(E)={=, $}

产生移进-归约冲突，但该文法不是二义的。

规范的LR分析

- 目标：在识别活前缀DFA的状态中，增加信息，排除一些不正确的归约操作

规范的LR分析

- 目标：在识别活前缀DFA的**状态**中，**增加信息**，**排除一些**不正确的**归约**操作
- 方法：添加了前向搜索符
  - 一个项目$A \rightarrow \alpha \cdot \beta$，如果最终用这个产生式进行归约之后，期望看见的符号是a，则这个加点项的前向搜索符是a。
  - 上述项目可以写成：$A \rightarrow \alpha \cdot \beta,\ a$

规范的LR分析

- 目标：在识别活前缀DFA的**状态**中，**增加信息**，**排除一些**不正确的**归约**操作
- 方法：添加了前向搜索符
  - 一个项目$A \rightarrow \alpha \cdot \beta$，如果最终用这个产生式进行归约之后，期望看见的符号是a，则这个加点项的前向搜索符是a。
  - 上述项目可以写成：$A \rightarrow \alpha \cdot \beta, a$
- 与SLR(1)分析的区别
  - 项目集的定义发生了改变：LR(0) => LR(1)
  - closure(I) 和GOTO函数需要修改

规范的LR分析

- **LR(1)项目：**

$$[A \rightarrow \alpha \cdot \beta, a]$$

  - 当项目由两个分量组成，第一分量为SLR中的项，第二分量为搜索符（向前看符号）
  - LR(1)中的1代表了搜索符a的长度

规范的LR分析

❑LR(1)项目：

$$[A \rightarrow \alpha \cdot \beta, a]$$

- ❖当项目由两个分量组成，第一分量为SLR中的项，第二分量为搜索符（向前看符号）
- ❖LR(1)中的1代表了搜索符a的长度

❑使用注意事项：

- ❖当β不为空时，a不起作用
- ❖当β为空时，如果下一个输入符号是a，将按照$A \rightarrow \alpha$进行归约
  - ➢a的集合是FOLLOW(A)的子集

规范的LR分析

❑LR(1)项目：

$$[A \rightarrow \alpha \cdot \beta, a]$$

- ❖当项目由两个分量组成，第一分量为SLR中的项，第二分量为搜索符（向前看符号）
- ❖LR(1)中的1代表了搜索符a的长度

❑LR(1)项目$[A \rightarrow \alpha \cdot \beta, a]$对活前缀$\gamma$有效：

- ❖如果存在着推导$S \Rightarrow^*_{rm} \delta A w \Rightarrow_{rm} \delta\alpha\beta w$，其中：
  - ➢$\gamma = \delta\alpha$;
  - ➢a是w的第一个符号，或者w是ε且a是\$

规范的LR分析：举例

❑例 $S \to BB$

$B \to bB \mid a$

LR(1)项目$[A \to \alpha \cdot \beta, a]$对活前缀$\gamma$有效：
存在着推导$S \Rightarrow^*_{rm} \delta Aw \Rightarrow_{rm} \delta\alpha\beta w$，其中：
$\gamma = \delta\alpha$;
a是w的第一个符号，或者w是ε且a是\$

从最右推导$S \Rightarrow^*_{rm} bbBba \Rightarrow_{rm} bbbBba$看出：

$[B \to b \cdot B, b]$对活前缀$\gamma = bbb$是有效的

规范的LR分析：举例

❑例 $S \to BB$

$B \to bB \mid a$

LR(1)项目$[A \to \alpha \cdot \beta, a]$对活前缀$\gamma$有效：
存在着推导$S \Rightarrow^*_{rm} \delta Aw \Rightarrow_{rm} \delta\alpha\beta w$，其中：
$\gamma = \delta\alpha$;
a是w的第一个符号，或者w是ε且a是\$

从最右推导$S \Rightarrow^*_{rm} bbBba \Rightarrow_{rm} bbbBba$看出：

令$A = B,\ \alpha = b,\ \beta = B,\ \delta = bb,\ \gamma = \delta\alpha = bbb,\ w = ba$

$[B \to b \cdot B, b]$对活前缀$\gamma = bbb$是有效的

规范的LR分析—步骤

- 构造LR(1)项目集规范族
  - 也就是构造识别活前缀的DFA
- 构造规范的LR分析表
  - 状态之间的转换关系

构造LR(1)项目集规范族

❑基础运算1：计算闭包CLOSURE(I)

- ❖I中的任何项目都属于CLOSURE(I)
- ❖若有项目 $[A\rightarrow\alpha\cdot B\beta, a]$在CLOSURE(I)中，而$B\rightarrow\gamma$是文法中的产生式，$b$是FIRST($\beta a$)中的元素，则$[B\rightarrow\cdot\gamma, b]$也属于CLOSURE(I)

保证在用$B\rightarrow\gamma$进行归约后，

- 出现的输入字符b是句柄$\alpha B\beta$中B的后继符号
- 或者是$\alpha B\beta$归约为A后可能出现的终结符。

构造LR(1)项目集规范族

- 基础运算2：通过GOTO(I,X)算CLOSURE(J)
  - 将J置为空集
  - 若有项目 $[A\rightarrow\alpha\cdot X\beta, a]$在I中，那么将项目 $[A\rightarrow\alpha X\cdot\beta, a]$放入J中
  - 计算并返回CLOSURE(J)

注意：GOTO(I,X)中的X可以是终结符或非终结符

构造LR(1)项目集规范族

❑具体算法

- ❖初始项目集I_0：

  I_0 = CLOSURE(*[S'→·S, $]*)　将$作为向前的搜索符

- ❖设C为最终返回的项目集族，初始为C={I_0}
- ❖重复以下步骤
  - ➢对C中的任意项目集I，重复
    - 对每一个文法符号X(终结符或非终结符)
      - 如果GOTO(I,X) ≠ ∅ 且 GOTO(I,X) ∉ C，那么将GOTO(I,X)放入C
      - 注：上述GOTO(I,X)是上一页ppt中计算闭包的GOTO
  - ➢当C中项目集不再增加为止

构造LR(1)项目集族：举例

$S' \rightarrow \cdot S,\ \$$ I_0

步骤一：从初始项开始

构造LR(1)项目集族：举例

$S' \rightarrow \cdot S,\ \$$ I_0

$S \rightarrow \cdot BB$

步骤二：计算非核心项目的第一个分量

构造LR(1)项目集族：举例

$S' \rightarrow \cdot S,\ \$$ I_0

$S \rightarrow \cdot BB,\ \$$

步骤三：通过FIRST(ε$) 计算非核心项目的第二个分量

构造LR(1)项目集族：举例

$S' \rightarrow \cdot S,\ \$$ I_0

$S \rightarrow \cdot BB,\ \$$

$B \rightarrow \cdot bB$

$B \rightarrow \cdot a$

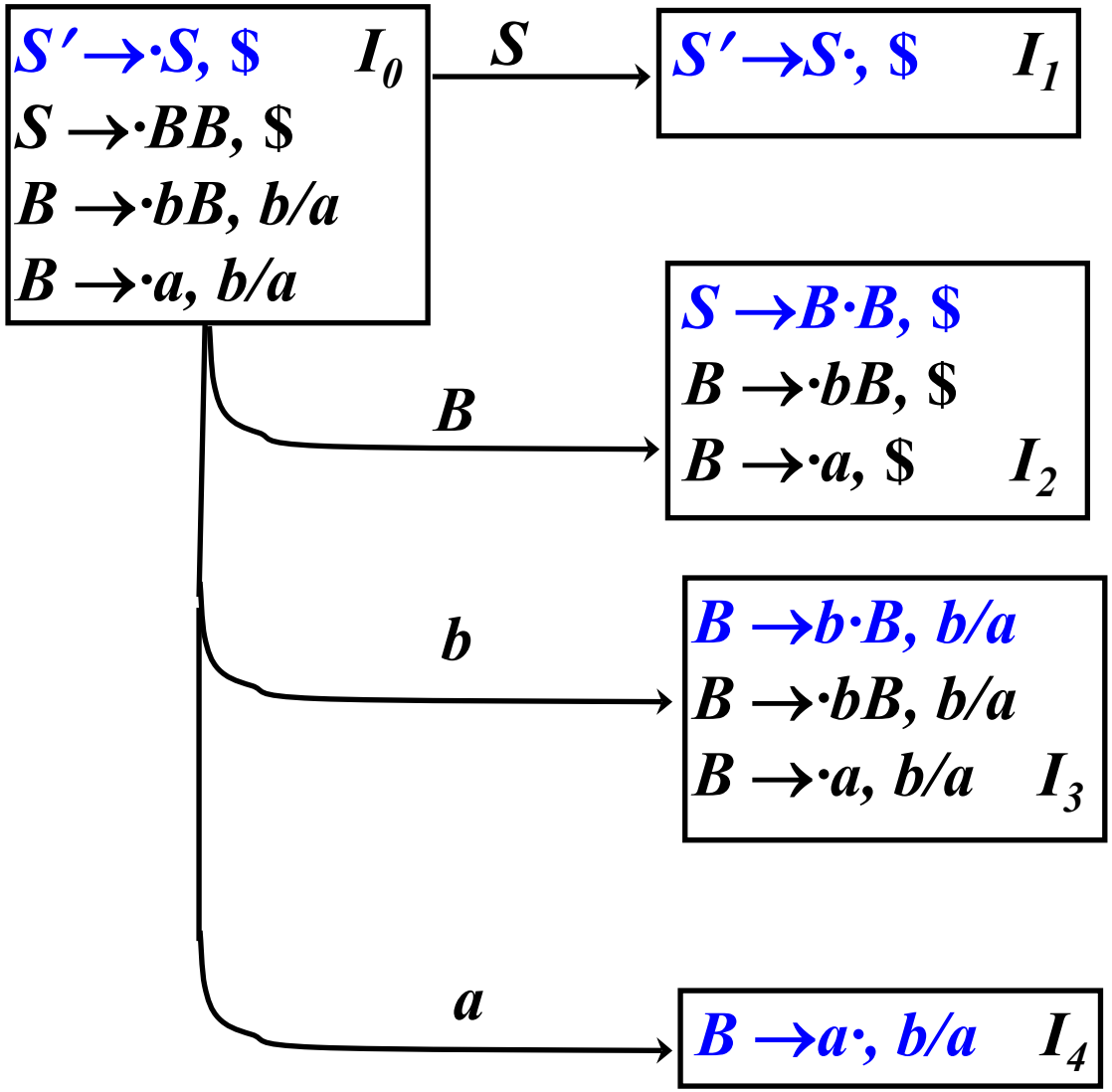
步骤二：计算非核心项目的第一个分量

构造LR(1)项目集族：举例

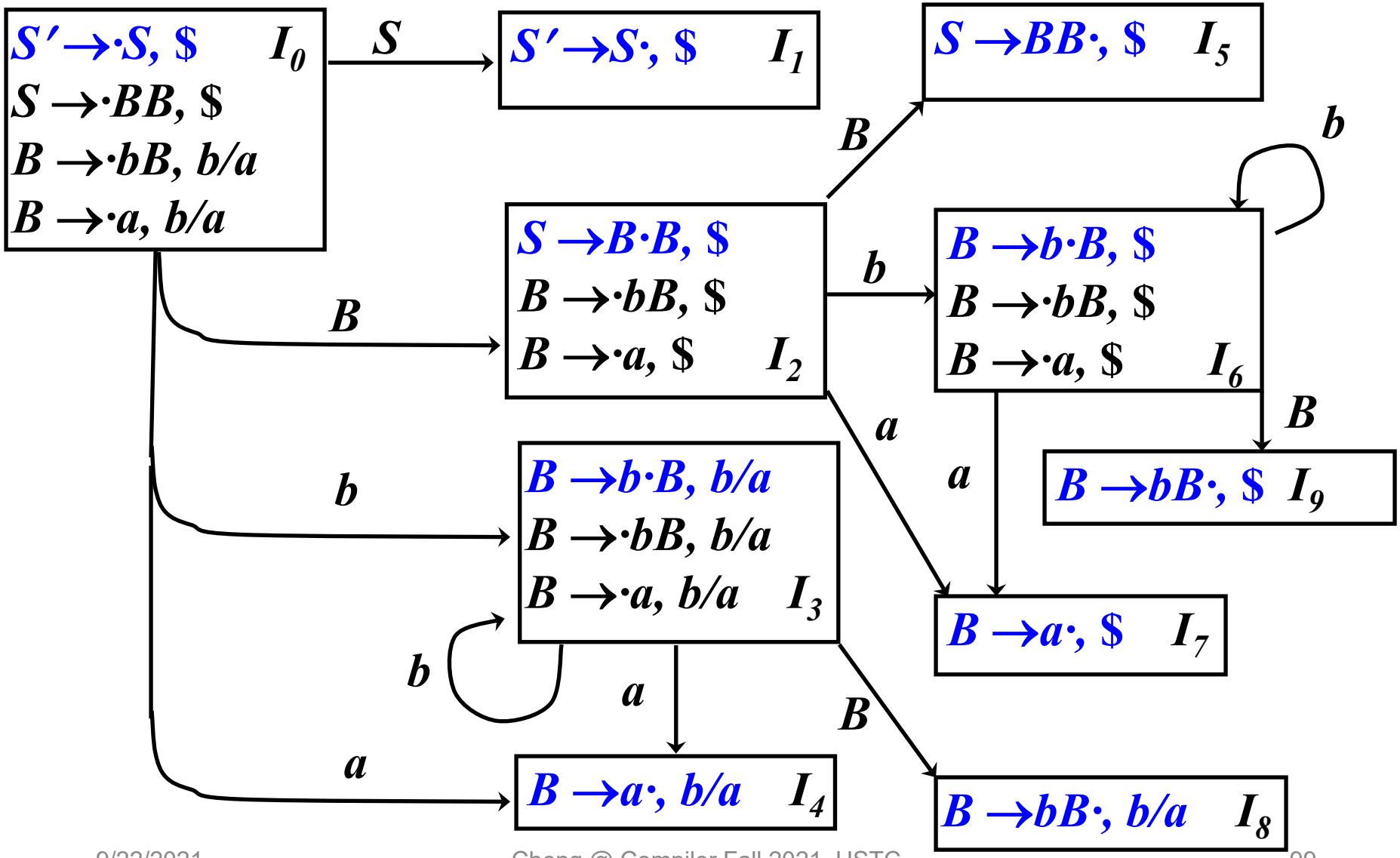
$S' \rightarrow \cdot S,\ \$$ I_0
$S \rightarrow \cdot BB,\ \$$
$B \rightarrow \cdot bB,\ b/a$
$B \rightarrow \cdot a,\ b/a$

步骤三：通过FIRST(B$) 计算非核心项目的第二个分量

构造LR(1)项目集族：举例



构造LR(1)项目集族：举例



构造规范的LR分析表

- 构造识别拓广文法G′活前缀的DFA
  - 基于LR(1)项目族来构造
- 状态i的$action$函数如下确定：
  - 如果$[A \to \alpha \cdot a\beta, b]$在$I_i$中，且$goto(I_i, a) = I_j$，那么置$action[i, a]$为$sj$ (此时，不看b)
  - 如果$[A \to \alpha \cdot, a]$在I_i中，且$A \neq S'$，那么置$action[i, a]$为rj (此时，不再看FOLLOW(A))
  - 如果$[S' \to S \cdot, \$]$在$I_i$中，那么置$action[i, \$] = acc$

如果上述构造出现了冲突，那么文法就不是LR(1)的

构造规范的LR分析表

- 构造识别拓广文法G′活前缀的DFA
- 状态i的$action$函数如下确定：
  - 参见上页ppt
- 状态i的$goto$函数如下确定：
  - 如果$goto(I_i, A) = I_j$, 那么$goto[i, A] = j$

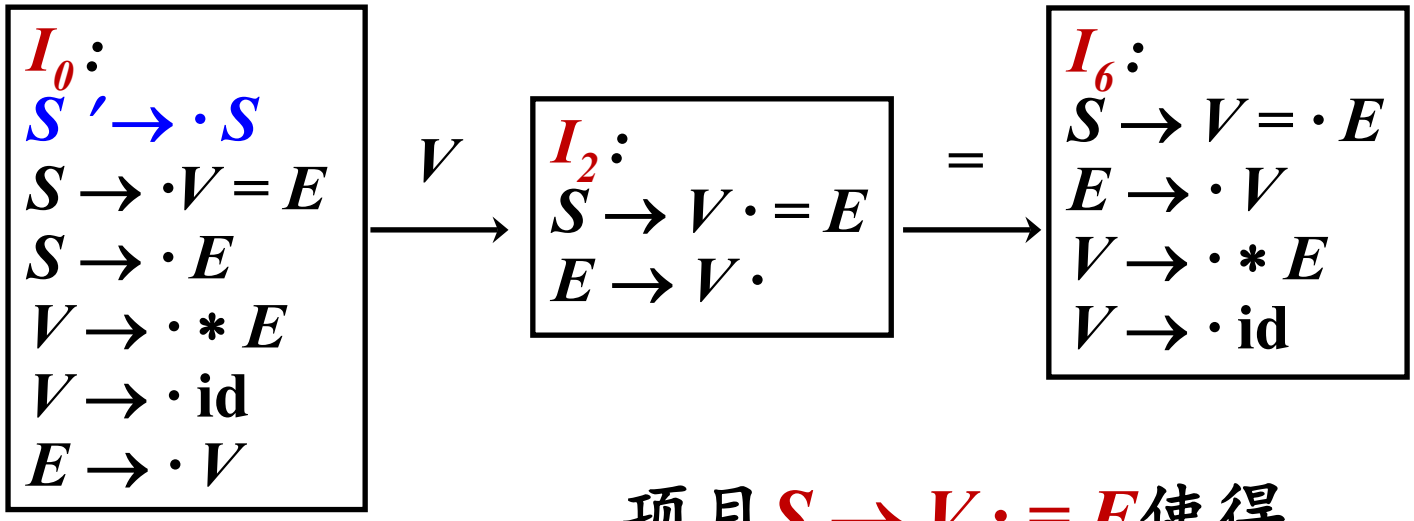
构造规范的LR分析表

- 构造识别拓广文法G′活前缀的DFA
- 状态i的$action$函数如下确定：
  - 参见上页ppt
- 状态i的$goto$函数如下确定：
  - 如果$goto(I_i, A) = I_j$，那么$goto[i, A] = j$
- 分析器的初始状态是包含$[S' \rightarrow \cdot S, \$]$的项目集对应的状态

用上面规则未能定义的所有条目都置为error

非SLR(1)文法-revisit

$S \to V = E$
$S \to E$
$V \to * E$
$V \to \text{id}$
$E \to V$



项目$S \to V \cdot = E$使得
action[2, =] = s6

项目$E \to V \cdot$使得
action[2, =] = r5
因为Follow(E)={=, $}

产生移进-归约冲突，但该文法不是二义的。

非SLR(1)但是LR(1)文法

$S \rightarrow V = E$
$S \rightarrow E$
$V \rightarrow * E$
$V \rightarrow \mathbf{id}$
$E \rightarrow V$

I_0:
$S' \rightarrow \cdot S,\ \$$

非SLR(1)但是LR(1)文法

$S \to V = E$
$S \to E$
$V \to * E$
$V \to \text{id}$
$E \to V$

I_0:
$S' \to \cdot S,\ \$$

计算闭包：

定义里：　$[A \to \alpha \cdot B\ \beta,\ a]$

这里：　$[S' \to \varepsilon \cdot S\ \varepsilon,\ \$]$

FIRST(βa)

⇩

FIRST(ε$)={$}

非SLR(1)但是LR(1)文法

$S \to V = E$
$S \to E$
$V \to * E$
$V \to \text{id}$
$E \to V$

I_0:
$S' \to \cdot S,\ \$$
$S \to \cdot V = E,\ \$$
$S \to \cdot E,\ \$$

计算闭包：

定义里：　$[A \to \alpha \cdot B\ \beta,\ a]$

这里：　$[S' \to \varepsilon \cdot S\ \varepsilon,\ \$]$

FIRST(βa)

⇩

FIRST(ε$)={$}

非SLR(1)但是LR(1)文法

$S \to V = E$
$S \to E$
$V \to * E$
$V \to \text{id}$
$E \to V$

I_0:
$S' \to \cdot S, \$$
$S \to \cdot V = E, \$$
$S \to \cdot E, \$$

计算闭包：

定义里：　$[A \to \alpha \cdot B\ \beta, a]$

这里：　$[S \to \varepsilon \cdot V = E, \$]$

FIRST(βa)

⇩

FIRST(= E$)={=}

非SLR(1)但是LR(1)文法

$S \to V = E$
$S \to E$
$V \to * E$
$V \to \text{id}$
$E \to V$

I_0:
$S' \to \cdot S, \$$
$S \to \cdot V = E, \$$
$S \to \cdot E, \$$
$V \to \cdot * E, =$
$V \to \cdot \text{id}, =$

计算闭包：

定义里：$[A \to \alpha \cdot B\ \beta, a]$

这里：$[S \to \varepsilon \cdot V = E, \$]$

FIRST(βa)

⇩

FIRST(= E$)={=}

非SLR(1)但是LR(1)文法

$S \to V = E$
$S \to E$
$V \to * E$
$V \to \text{id}$
$E \to V$

I_0:
$S' \to \cdot S, \$$
$S \to \cdot V = E, \$$
$S \to \cdot E, \$$
$V \to \cdot * E, =$
$V \to \cdot \text{id}, =$

计算闭包：

定义里：$[A \to \alpha \cdot B\ \beta, a]$

这里：$[S \to \varepsilon \cdot V = E, \$]$

FIRST(βa)

⇩

FIRST(= E$)={=}

非SLR(1)但是LR(1)文法

$S \to V = E$
$S \to E$
$V \to * E$
$V \to \text{id}$
$E \to V$

I_0:
$S' \to \cdot S,\ \$$
$S \to \cdot V = E,\ \$$
$S \to \cdot E,\ \$$
$V \to \cdot * E,\ =$
$V \to \cdot \text{id},\ =$
$E \to \cdot V,\ \$$

计算闭包：

定义里：$[A \to \alpha \cdot B\ \beta,\ a]$

这里：$[S \to \varepsilon \cdot E\ \varepsilon,\ \$]$

FIRST(βa)

⇩

FIRST(ε$)={$}

非SLR(1)但是LR(1)文法

$S \to V = E$
$S \to E$
$V \to * E$
$V \to \text{id}$
$E \to V$

I_0:
$S' \to \cdot S, \$$
$S \to \cdot V = E, \$$
$S \to \cdot E, \$$
$V \to \cdot * E, =$
$V \to \cdot \text{id}, =$
$E \to \cdot V, \$$
$V \to \cdot * E, \$$
$V \to \cdot \text{id}, \$$

非SLR(1)但是LR(1)文法

$S \rightarrow V = E$
$S \rightarrow E$
$V \rightarrow * E$
$V \rightarrow \mathrm{id}$
$E \rightarrow V$

I_0:
$S' \rightarrow \cdot S, \$$
$S \rightarrow \cdot V = E, \$$
$S \rightarrow \cdot E, \$$
$V \rightarrow \cdot * E, =/\$$
$V \rightarrow \cdot \mathrm{id}, =/\$$
$E \rightarrow \cdot V, \$$

可通过合并搜索符简化

非SLR(1)但是LR(1)文法

$S \to V = E$
$S \to E$
$V \to * E$
$V \to \text{id}$
$E \to V$

I_0:
$S' \to \cdot S, \$$
$S \to \cdot V = E, \$$
$S \to \cdot E, \$$
$V \to \cdot * E, =/\$$
$V \to \cdot \text{id}, =/\$$
$E \to \cdot V, \$$

$V \longrightarrow$

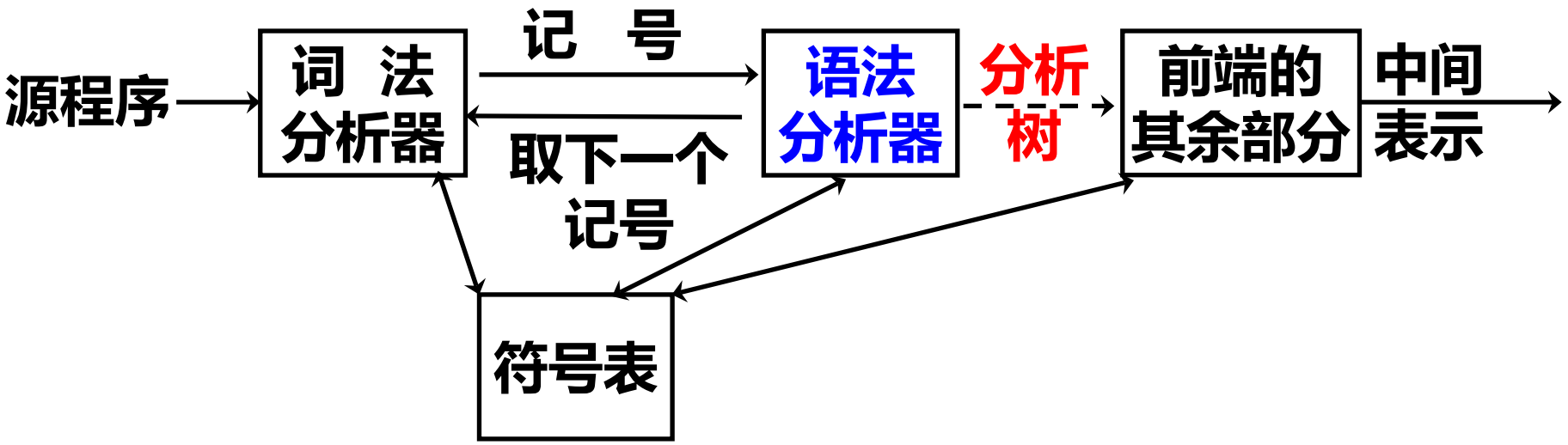
I_2:
$S \to V \cdot = E, \$$
$E \to V \cdot, \$$

项目$[S \to V \cdot = E, \$]$使得
action[2, =] = s6

项目$[E \to V \cdot, \$]$使得
action[2, $] = r5
因为{$}是Follow(E)={=, $}的真子集

每一个SLR(1)文法都是LR(1)的

主要内容

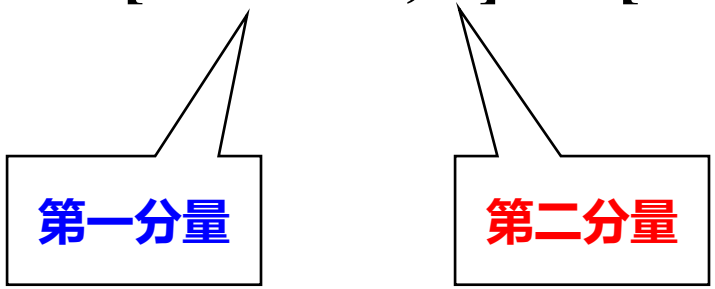


- LR(k)分析技术
  - LR分析器的简单模型
  - 简单的LR方法（简称SLR）
    - 活前缀，识别活前缀的DFA/NFA，SLR算法
  - 规范的LR方法
  - 向前看的LR方法（简称LALR）

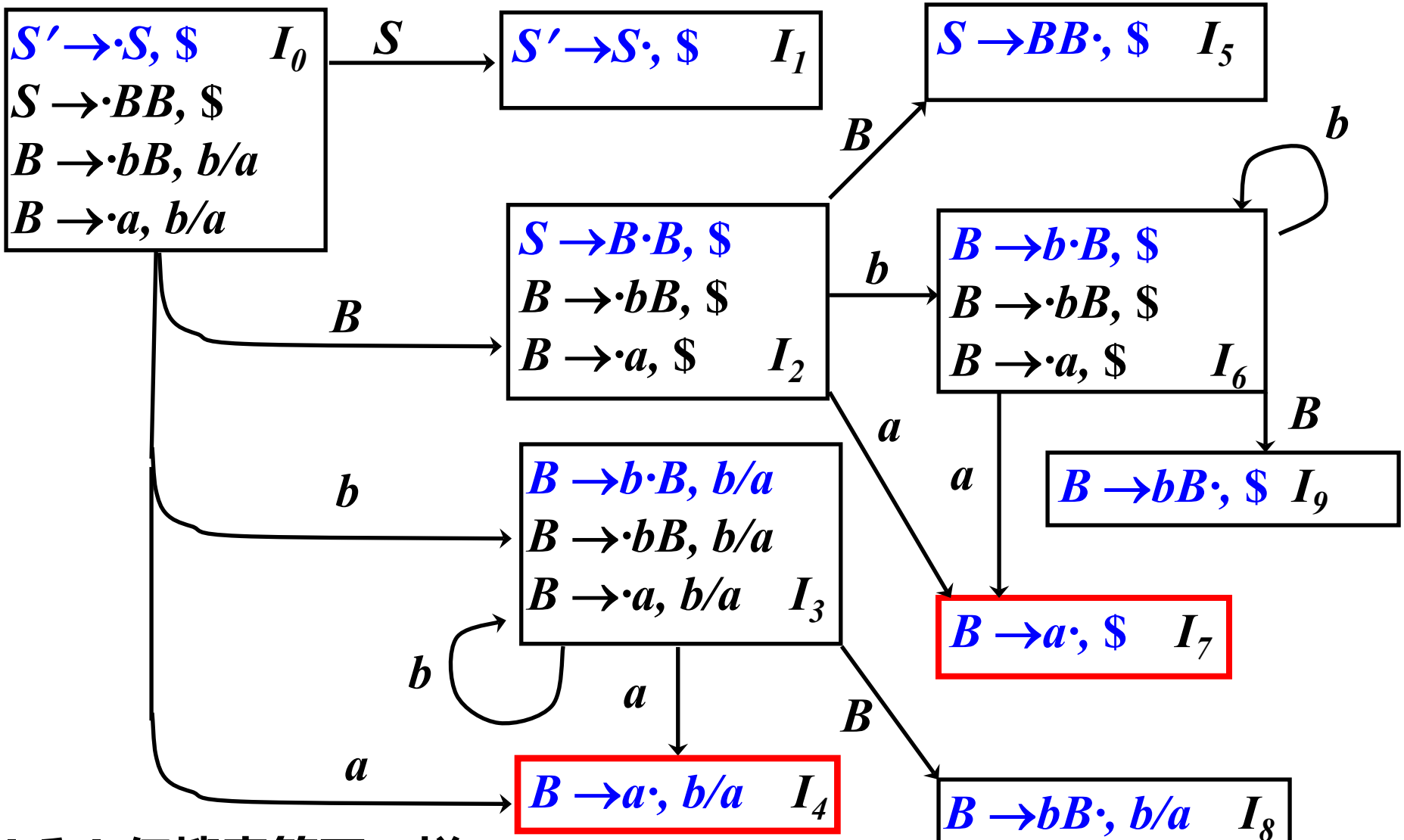
LALR分析方法

- 研究LALR的原因
  - 规范LR分析表的状态数偏多
- LALR特点
  - LALR和SLR的分析表有同样多的状态，比规范LR分析表要小得多
  - LALR的能力介于SLR和规范LR之间
  - LALR的能力在很多情况下已经够用
- LALR分析表构造方法
  - 通过合并规范LR(1)项目集来得到

- 合并识别 LR(1)文法的活前缀的DFA中的相同核心项目集(同心项目集，注意：不是项)
- 同心的LR(1)项目集
  - 核心：项目集中第一分量的集合
  - 略去搜索符后它们是相同的集合
  - 例： [B →·bB, $] 与 [B →·bB, b/a]

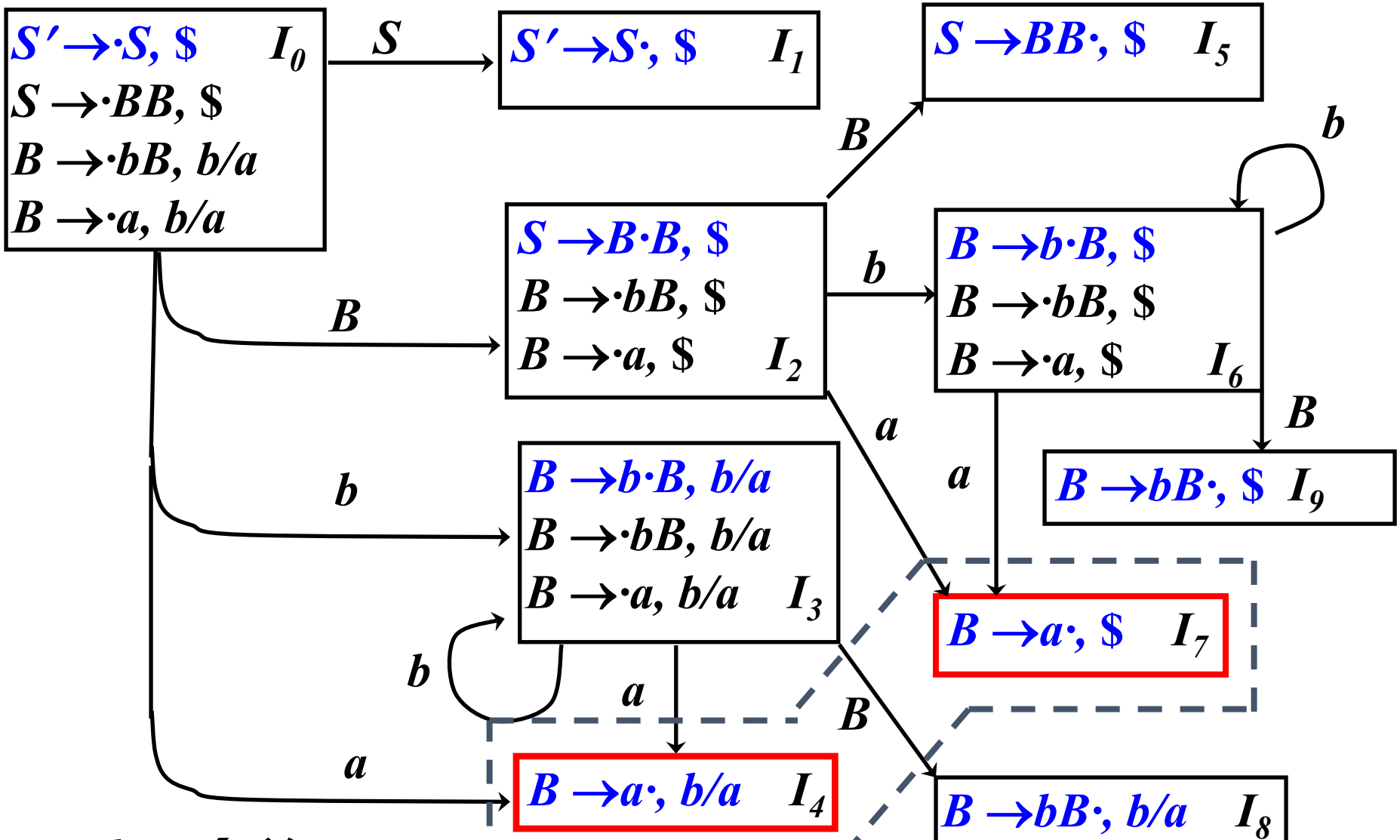


识别活前缀的DFA



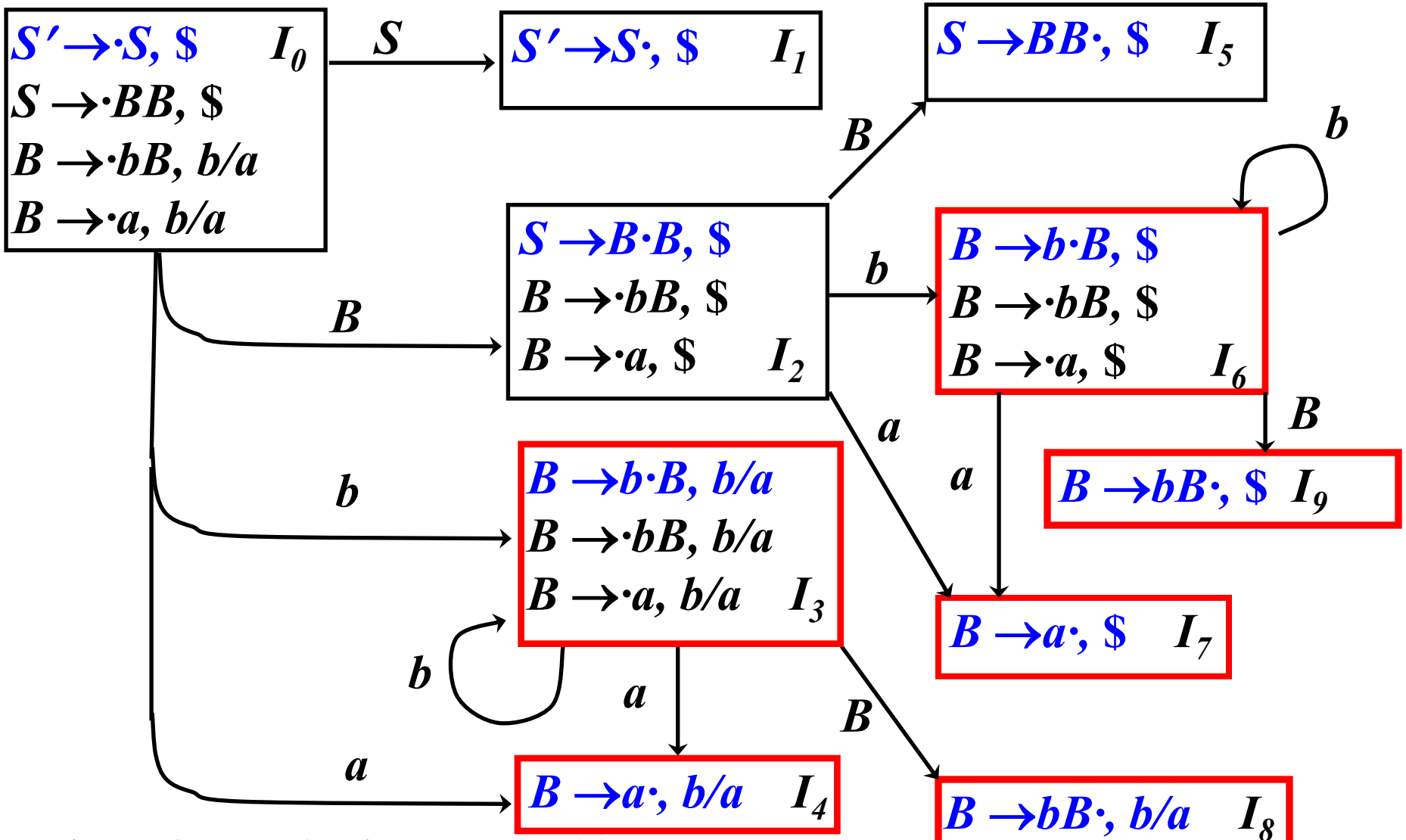
I_4和I_7仅搜索符不一样

合并同心项目集



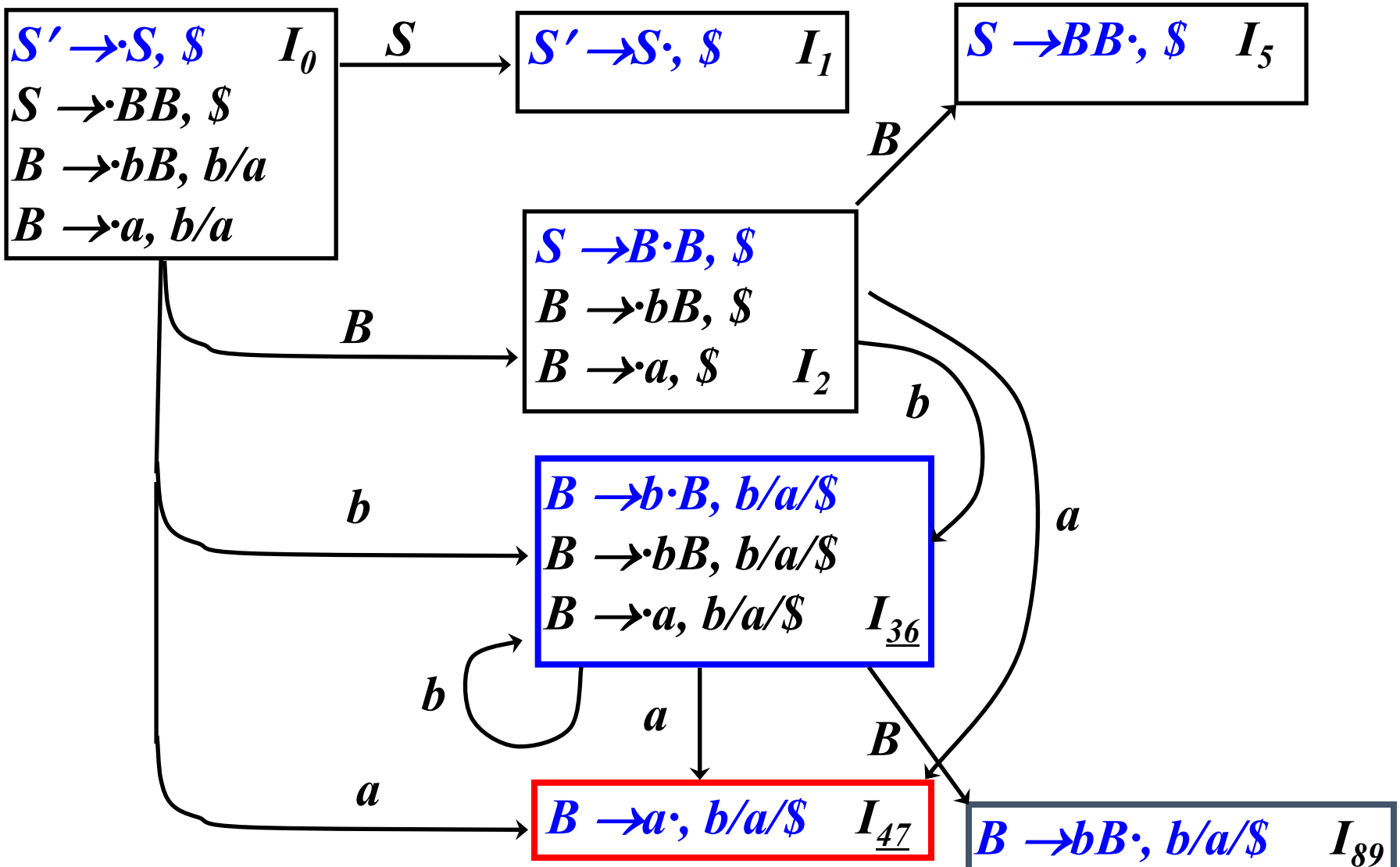
I_4和I_7合并

合并同心项目集

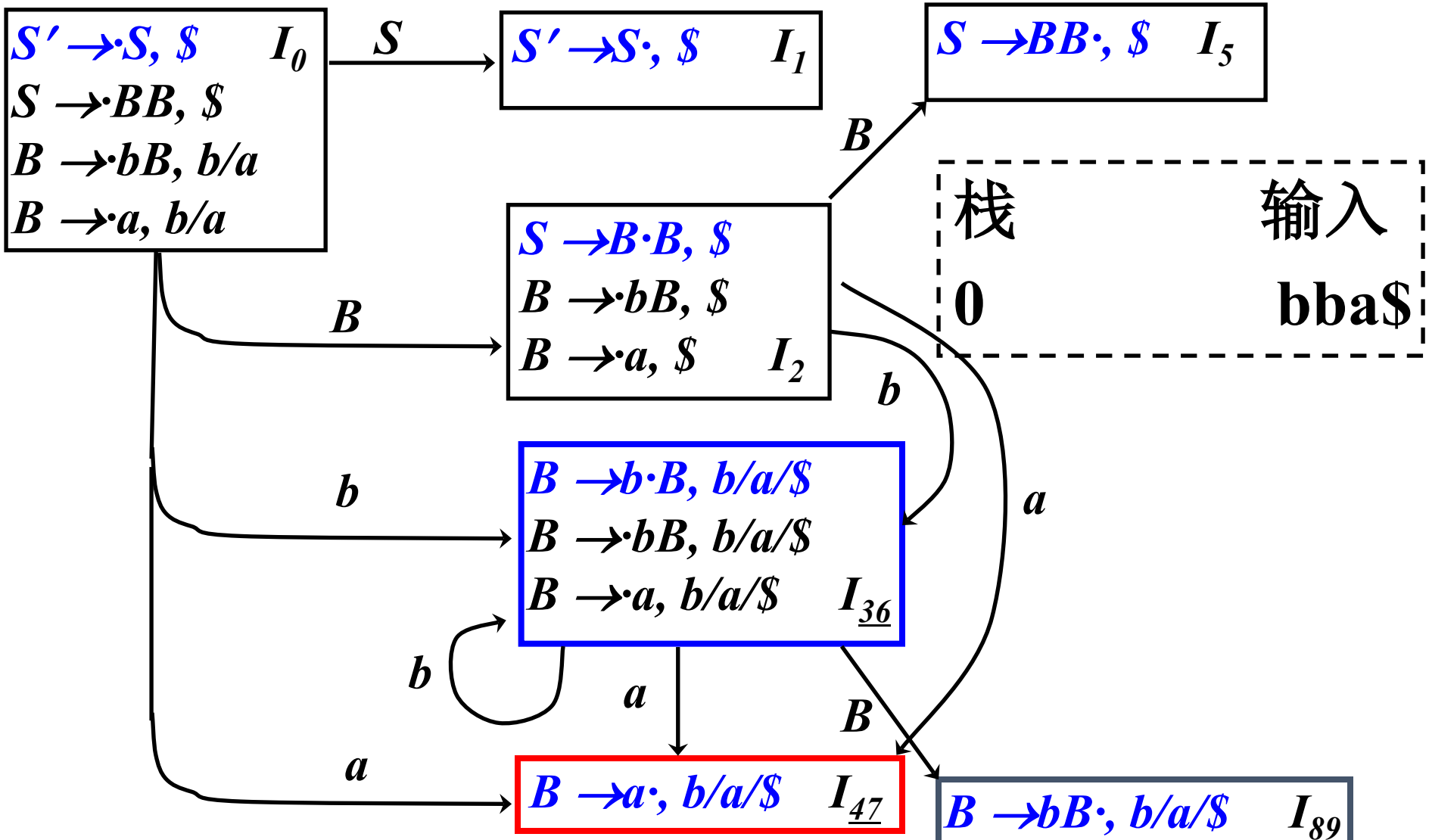


有三组同心集，都合并

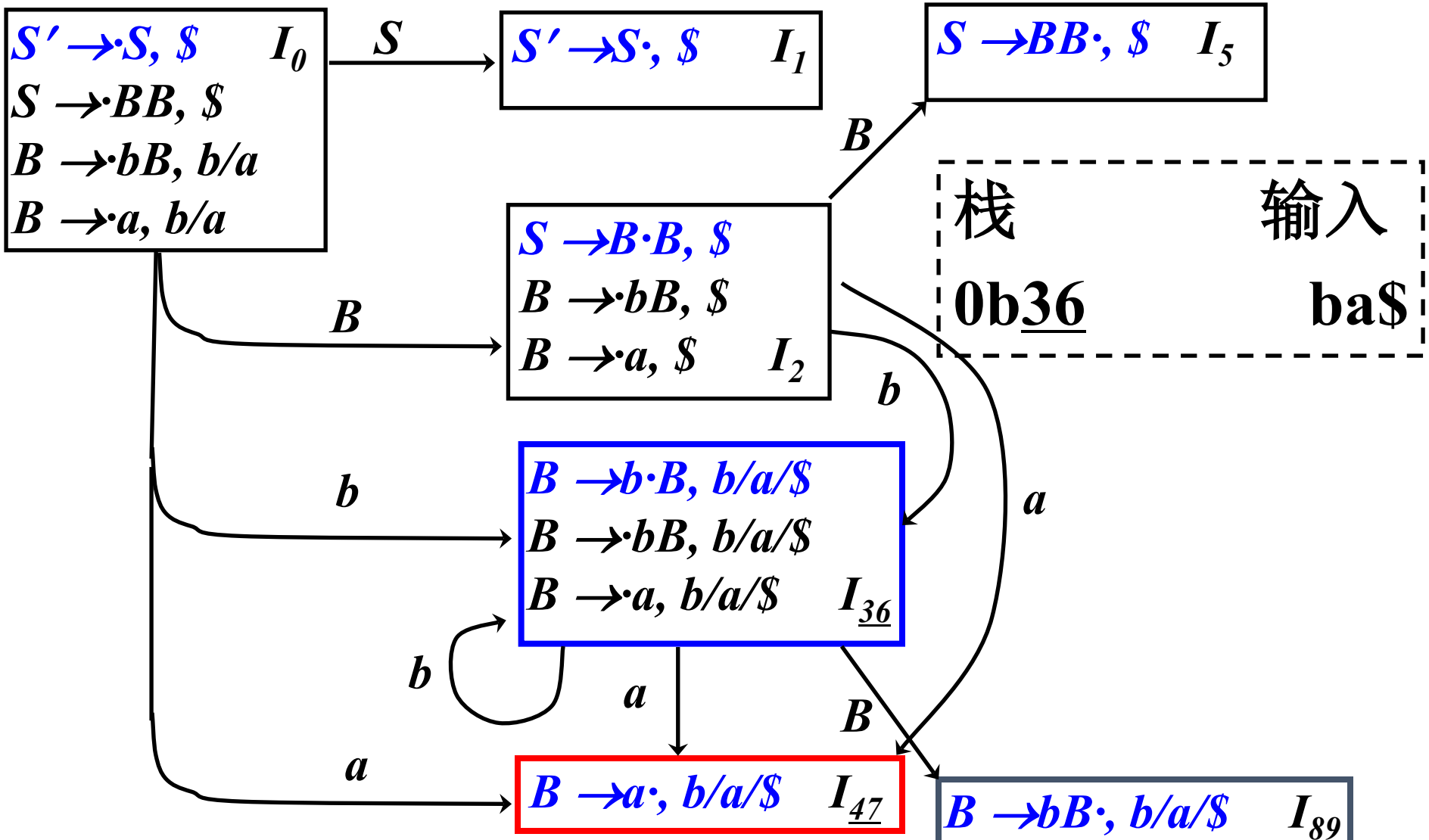
合并同心项目集


S′ →·S, $ I0
S →·BB, $
B →·bB, b/a
B →·a, b/a
S
S′ →S·, $ I1
S →BB·, $ I5
B
S →B·B, $
B →·bB, $
B →·a, $ I2
B
b
b
B →b·B, b/a/$
B →·bB, b/a/$
B →·a, b/a/$ I36
a
b
a
B
a
B →a·, b/a/$ I47
B →bB·, b/a/$ I89

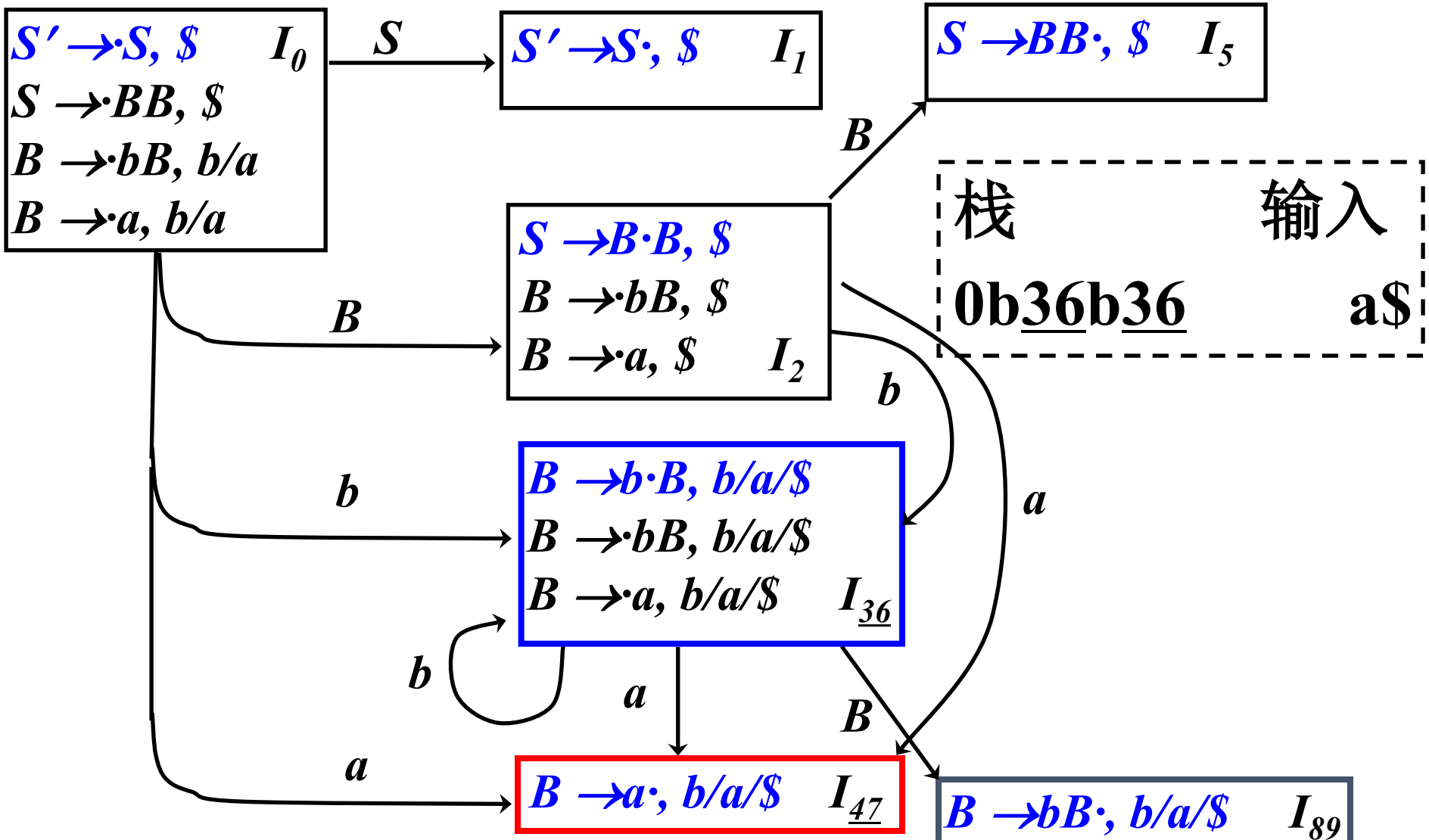
LALR分析：举例


S′ →·S, $ I0
S →·BB, $
B →·bB, b/a
B →·a, b/a
S
S′ →S·, $ I1
B
S →BB·, $ I5
B
S →B·B, $
B →·bB, $
B →·a, $ I2
b
B
b
B →b·B, b/a/$
B →·bB, b/a/$
B →·a, b/a/$ I36
a
b
a
B
a
B →a·, b/a/$ I47
B →bB·, b/a/$ I89
栈 输入
0 bba$

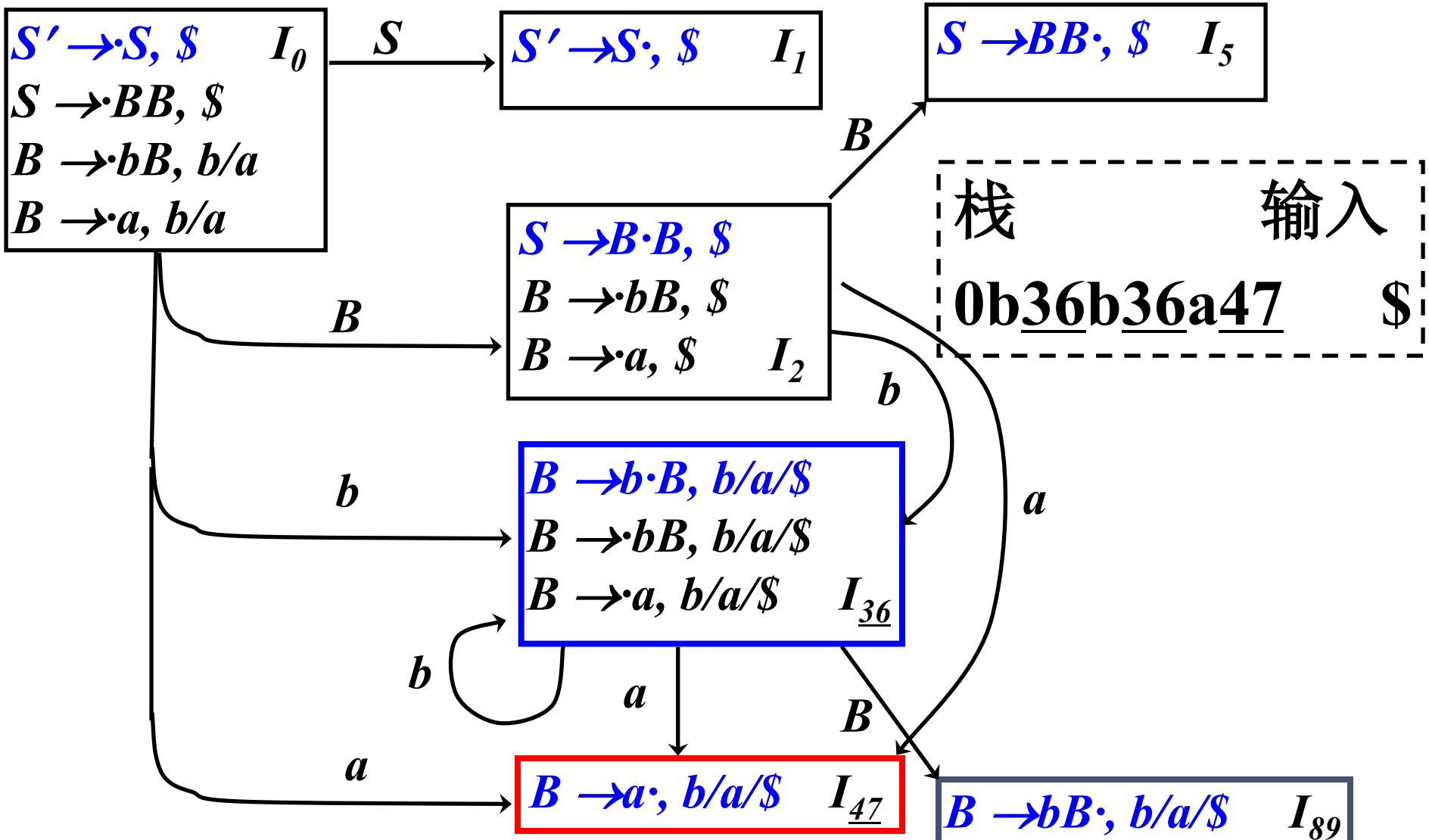
LALR分析：举例


S′ →·S, $ I0
S →·BB, $
B →·bB, b/a
B →·a, b/a
S
S′ →S·, $ I1
S →BB·, $ I5
B
B
S →B·B, $
B →·bB, $
B →·a, $ I2
b
b
B →b·B, b/a/$
B →·bB, b/a/$
B →·a, b/a/$ I36
a
b
a
B
a
B →a·, b/a/$ I47
B →bB·, b/a/$ I89
栈 输入
0b36 ba$

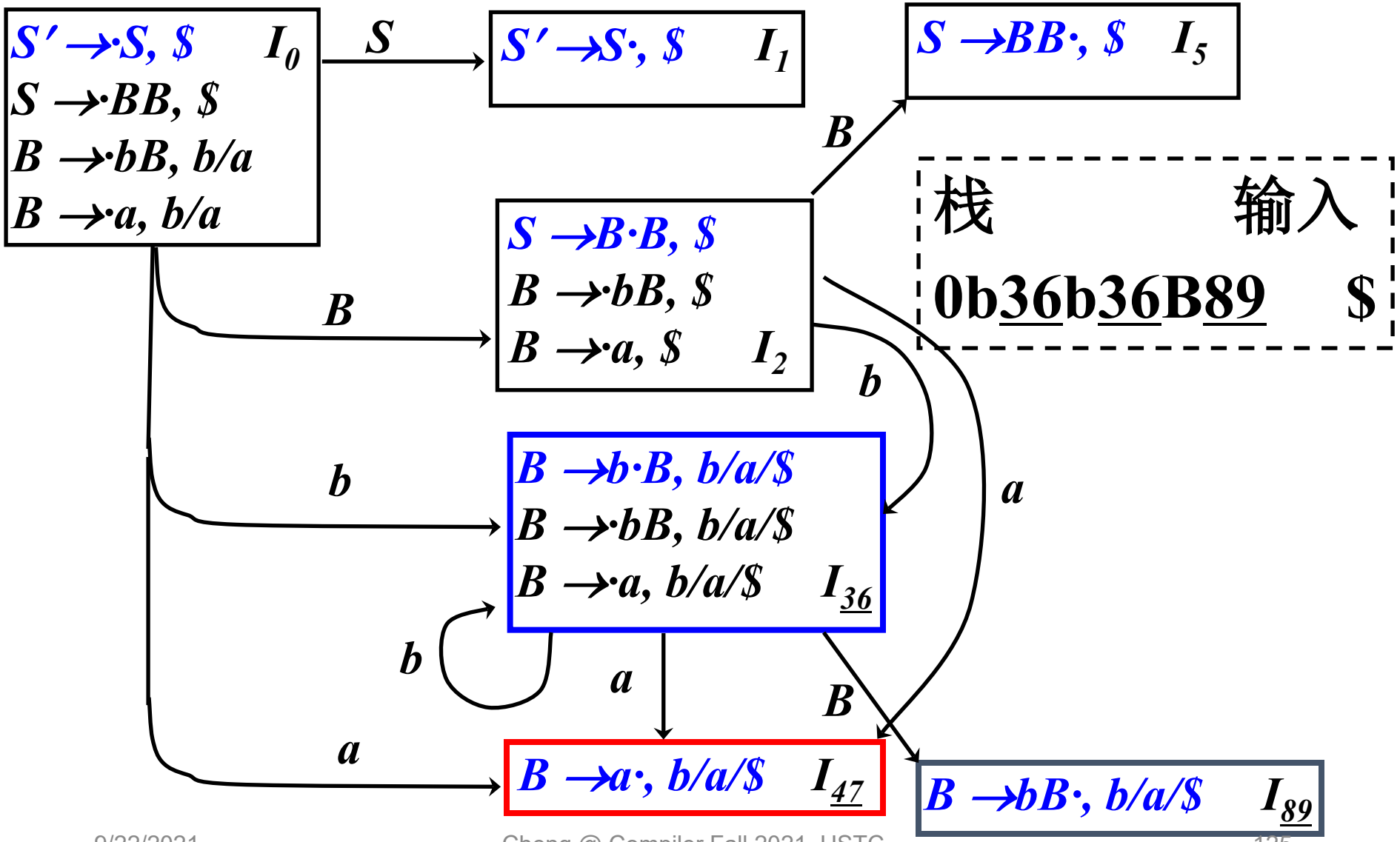
LALR分析：举例


S′ →·S, $ I0
S →·BB, $
B →·bB, b/a
B →·a, b/a
S
S′ →S·, $ I1
S →BB·, $ I5
B
S →B·B, $
B →·bB, $
B →·a, $ I2
B
b
B →b·B, b/a/$
B →·bB, b/a/$
B →·a, b/a/$ I36
b
a
b
a
B
B →a·, b/a/$ I47
B →bB·, b/a/$ I89
栈 输入
0b36b36 a$

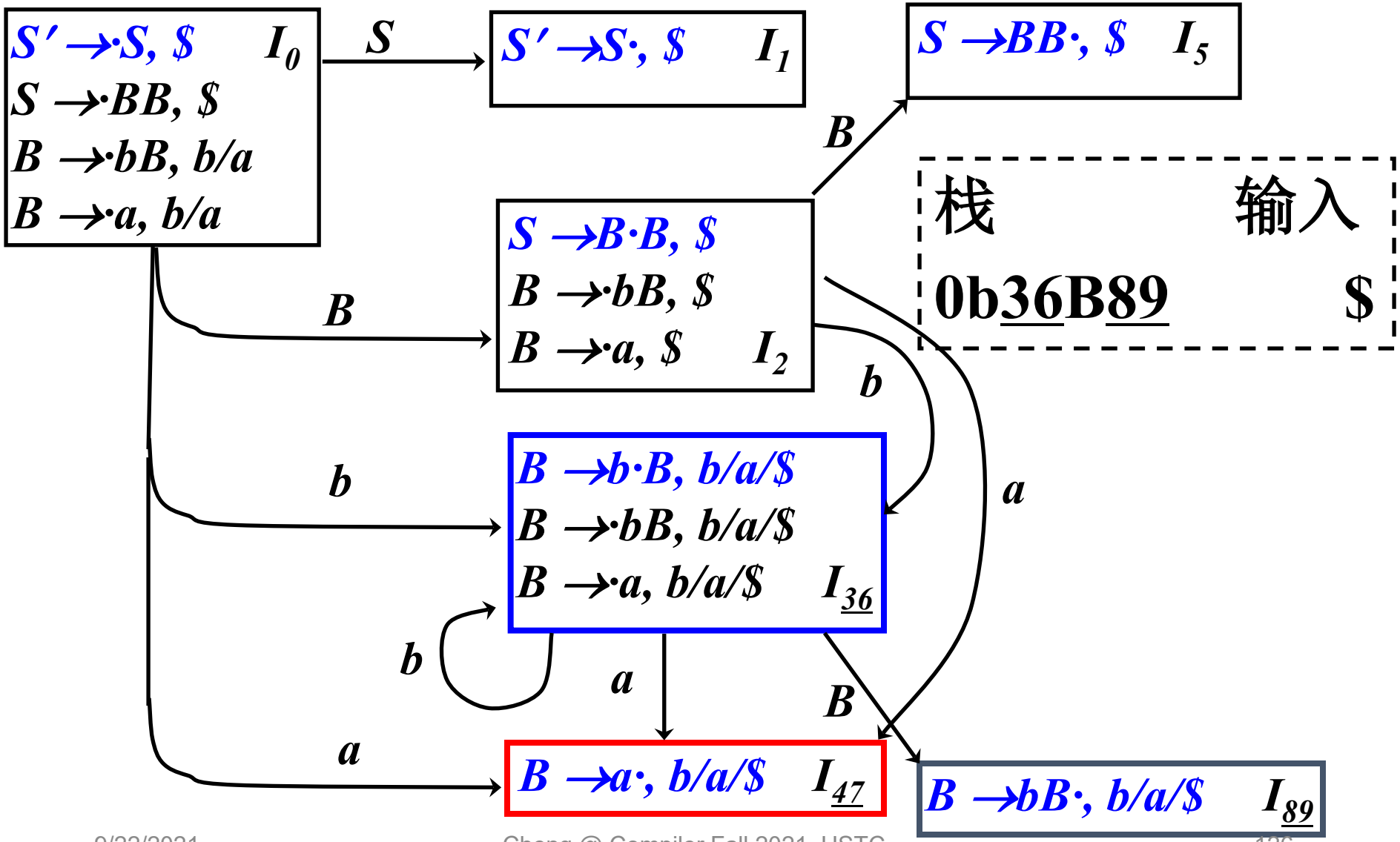
LALR分析：举例


S′ →·S, $ I0
S →·BB, $
B →·bB, b/a
B →·a, b/a
S
S′ →S·, $ I1
S →BB·, $ I5
B
B
S →B·B, $
B →·bB, $
B →·a, $ I2
b
b
B →b·B, b/a/$
B →·bB, b/a/$
B →·a, b/a/$ I36
a
b
a
B
a
B →a·, b/a/$ I47
B →bB·, b/a/$ I89
栈 输入
0b36b36a47 $

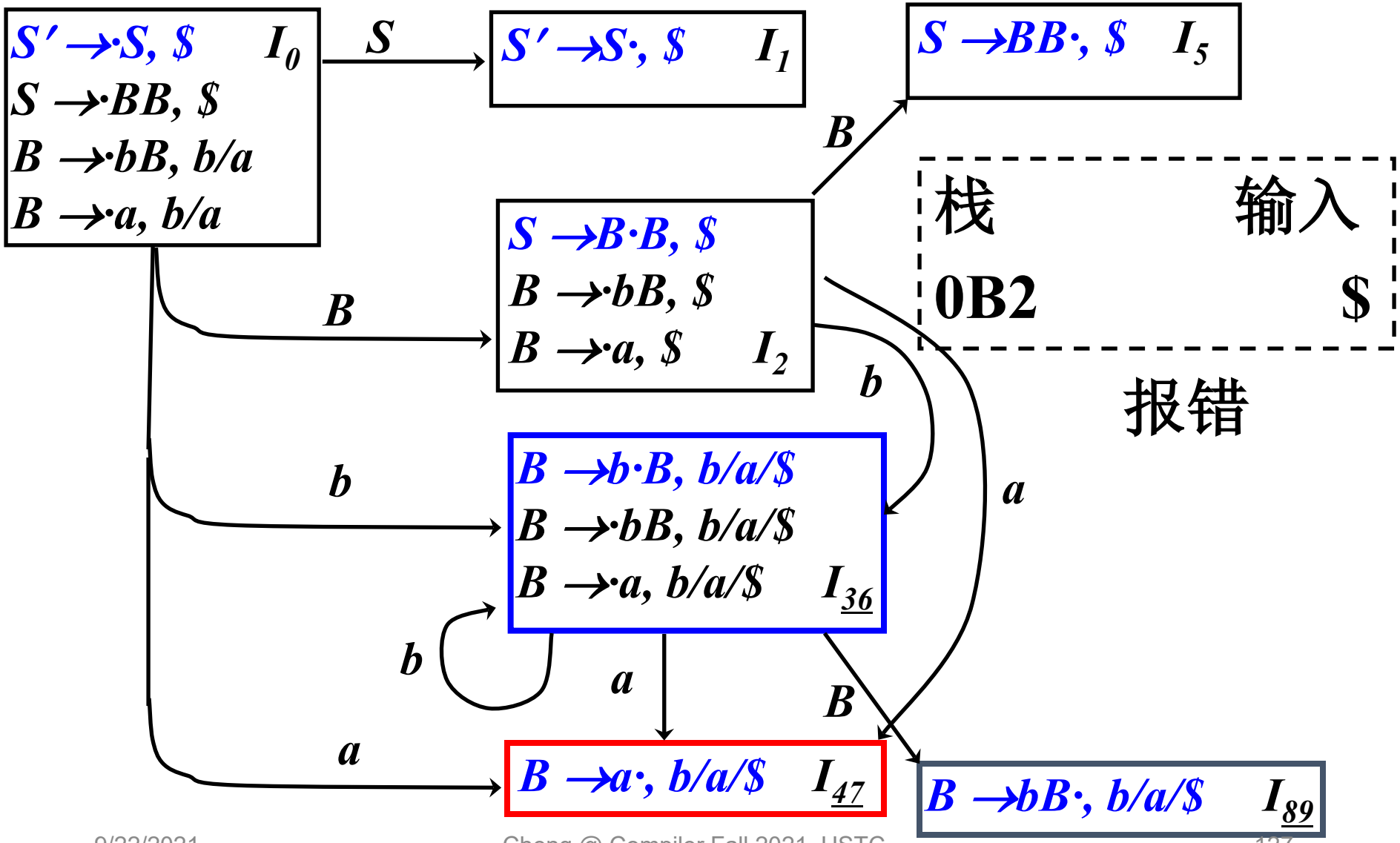
LALR分析：举例


S′ →·S, $ I0
S →·BB, $
B →·bB, b/a
B →·a, b/a
S
S′ →S·, $ I1
S →BB·, $ I5
B
S →B·B, $
B →·bB, $
B →·a, $ I2
B
b
a
B →b·B, b/a/$
B →·bB, b/a/$
B →·a, b/a/$ I36
b
a
B
a
B →a·, b/a/$ I47
B →bB·, b/a/$ I89
栈 输入
0b36b36B89 $

LALR分析：举例


S′ →·S, $ I0
S →·BB, $
B →·bB, b/a
B →·a, b/a
S
S′ →S·, $ I1
S →BB·, $ I5
B
S →B·B, $
B →·bB, $
B →·a, $ I2
B
b
B →b·B, b/a/$
B →·bB, b/a/$
B →·a, b/a/$ I36
b
a
b
a
B
B →a·, b/a/$ I47
B →bB·, b/a/$ I89
栈 输入
0b36B89 $

LALR分析：举例


S′ →·S, $ I0
S →·BB, $
B →·bB, b/a
B →·a, b/a
S
S′ →S·, $ I1
S →BB·, $ I5
B
B
S →B·B, $
B →·bB, $
B →·a, $ I2
b
b
B →b·B, b/a/$
B →·bB, b/a/$
B →·a, b/a/$ I36
a
b
a
B
a
B →a·, b/a/$ I47
B →bB·, b/a/$ I89
栈 输入
0B2 $
报错

2、构造LALR(1)分析表

- 构造LR(1)项目集规范族$C = \{I_0, I_1, \ldots, I_n\}$
- 构造LALR(1)项目集规范族$C' = \{J_0, J_1, \ldots, J_k\}$，其中任意项目集$J_i = I_n \cup I_m \cup \ldots \cup I_t$
  - $I_n, I_m, \ldots, I_t \in C$且具有共同的核心
- 按构造规范LR(1)分析表的方式构造分析表

如没有语法分析动作冲突，那么给定文法就是LALR(1)文法

LALR分析方法

- 合并同心项目集可能会引起冲突
  - 同心集的合并不会引起新的移进–归约冲突

项目集1	项目集2
$[A \rightarrow \alpha \cdot, a]$	$[B \rightarrow \beta \cdot a\gamma, b]$
$[B \rightarrow \beta \cdot a\gamma, c]$	$[A \rightarrow \alpha \cdot, d]$
...	...

如果有移进归约冲突，则合并前就有冲突

❑合并同心项目集可能会引起冲突

- ❖同心集的合并不会引起新的移进–归约冲突
- ❖同心集的合并有可能产生新的归约–归约冲突

$S' \to S$

$S \to aAd \mid bBd \mid aBe \mid bAe$

$A \to c$

$B \to c$

对ac有效的项目集

$A \to c \cdot, d$

$B \to c \cdot, e$

对bc有效的项目集

$A \to c \cdot, e$

$B \to c \cdot, d$

合并同心集后

$A \to c \cdot, d/e$

$B \to c \cdot, d/e$

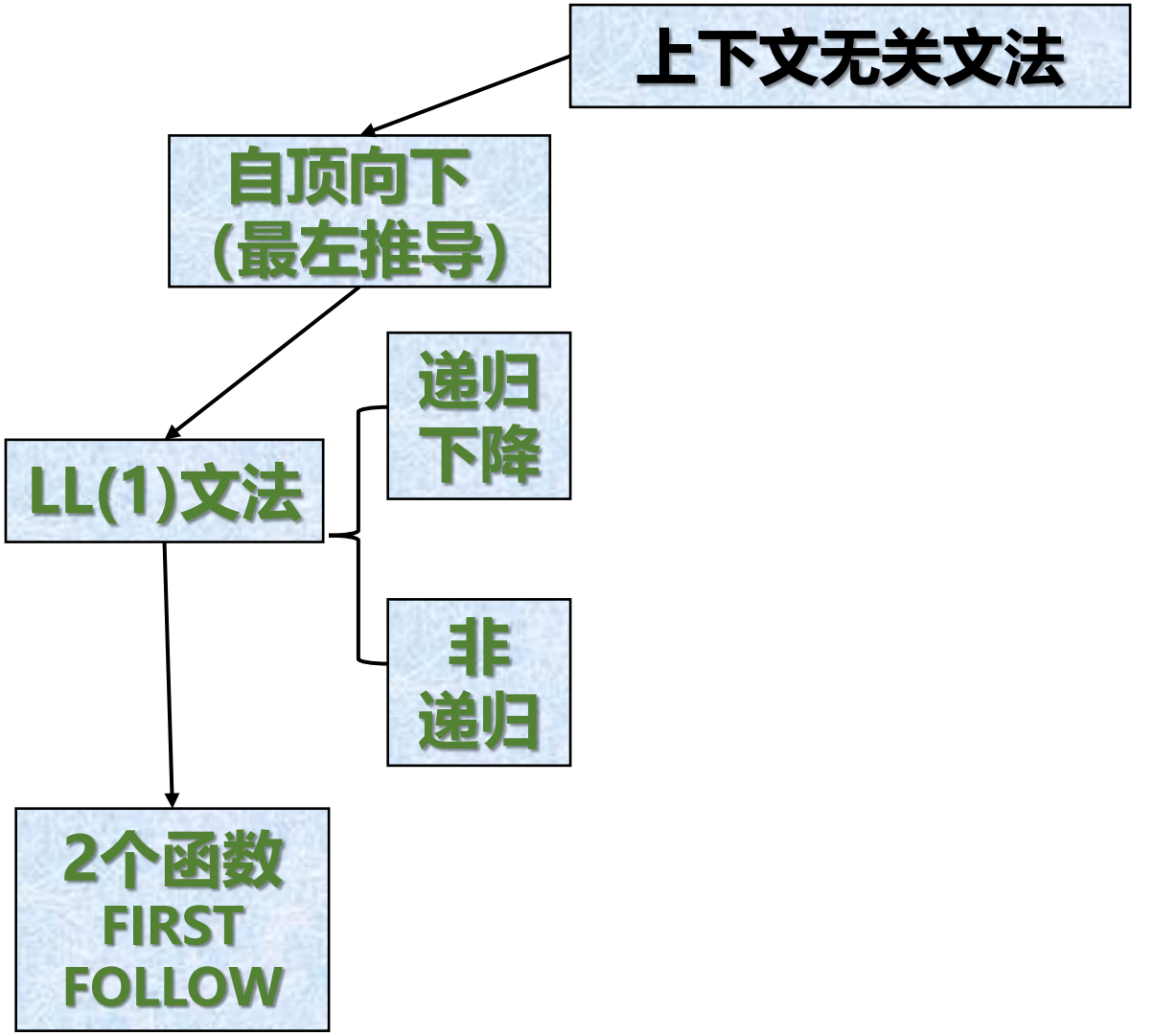
该文法是LR(1)的
但不是LALR(1)的

LR分析法总结

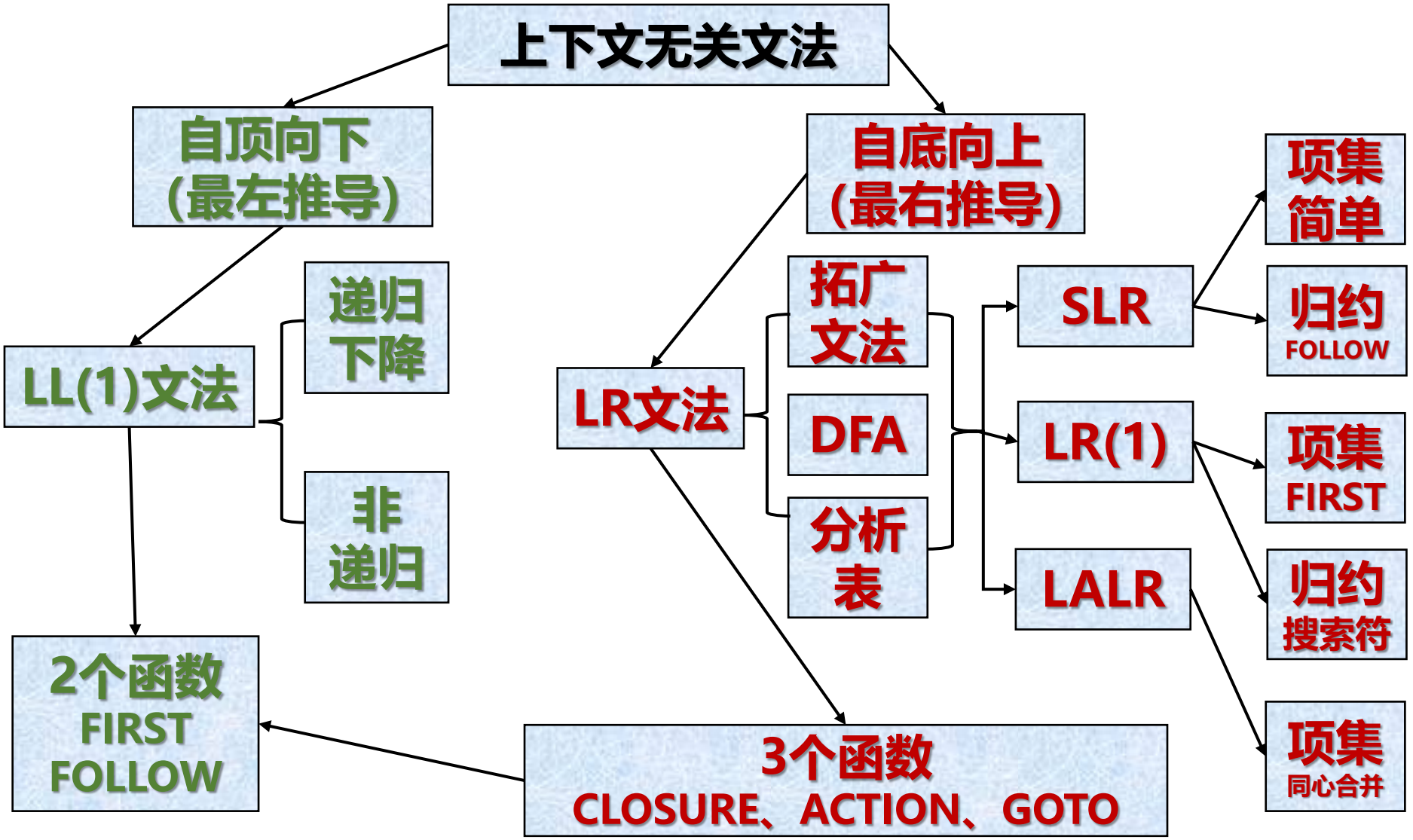
中国科学技术大学 University of Science and Technology of China

		SLR	LALR	LR(1)
初始状态		$[S' \to \cdot S]$	$[S' \to \cdot S, \$]$	$[S' \to \cdot S, \$]$
项目集		LR(0) CLOSURE(I)	合并LR(1)项目集族的同心项目集	LR(1), CLOSURE(I) 搜索符考虑FISRT(βa)
动作	移进	$[A \to \alpha \cdot a\beta] \in I_i$ $GOTO(I_i, a) = I_j$ $ACTION[i, a] = sj$	与LR(1) 一致	$[A \to \alpha \cdot a\beta, b] \in I_i$ $GOTO(I_i, a) = I_j$ $ACTION[i, a] = sj$
	归约	$[A \to \alpha \cdot] \in I_i, A \neq S'$ $a \in FOLLOW(A)$ $ACTION[i, a] = rj$	与LR(1) 一致	$[A \to \alpha \cdot, a] \in I_i$ $A \neq S'$ $ACTION[i, a] = rj$
	接受	$[S' \to S \cdot] \in I_i$ $ACTION[i, \$] = acc$	与LR(1) 一致	$[S' \to S \cdot, \$] \in I_i$ $ACTION[i, \$] = acc$
	出错	空白条目	与LR(1) 一致	空白条目
GOTO		$GOTO(I_i, A) = I_j$ $GOTO[i, A] = j$	与LR(1) 一致	$GOTO(I_i, A) = I_j$ $GOTO[i, A] = j$
状态量		少(几百)	与SLR一样	多(几千)

语法分析技术总结



语法分析技术总结


上下文无关文法
自顶向下
(最左推导)
自底向上
(最右推导)
LL(1)文法
递归
下降
非
递归
LR文法
拓广
文法
DFA
分析
表
SLR
LR(1)
LALR
项集
简单
归约
FOLLOW
项集
FIRST
归约
搜索符
项集
同心合并
2个函数
FIRST
FOLLOW
3个函数
CLOSURE、ACTION、GOTO

LR和LL分析方法的比较

	LR(1)方 法	LL(1)方 法
建立分析树	自 底 而 上	自 顶 而 下
归约or推导	规 范 归 约	最 左 推 导
决定使用产生式的时机	看见产生式整个右部推出的串后(句柄)	看见产生式推出的第一个终结符后
对文法的限制	无	无左递归、无公共左因子
分析表	状态×文法符号，大	非终结符×终结符，小
分析栈	状态栈，信息更多	文法符号栈
确定句柄	根据栈顶状态和下一个符号便可以确定句柄和归约所用产生式	无句柄概念
语法错误	决不会将出错点后的符号移入分析栈	和LR一样，决不会读过出错点而不报错

中国科学技术大学
University of Science and Technology of China
USTC, CHINA
ADSLAB
先进数据系统实验室

《编译原理与技术》
语法分析IV

The Pessimist Sees Difficulty In Every Opportunity.
The Optimist Sees Opportunity In Every Difficulty.

—— ***Winston Churchill***