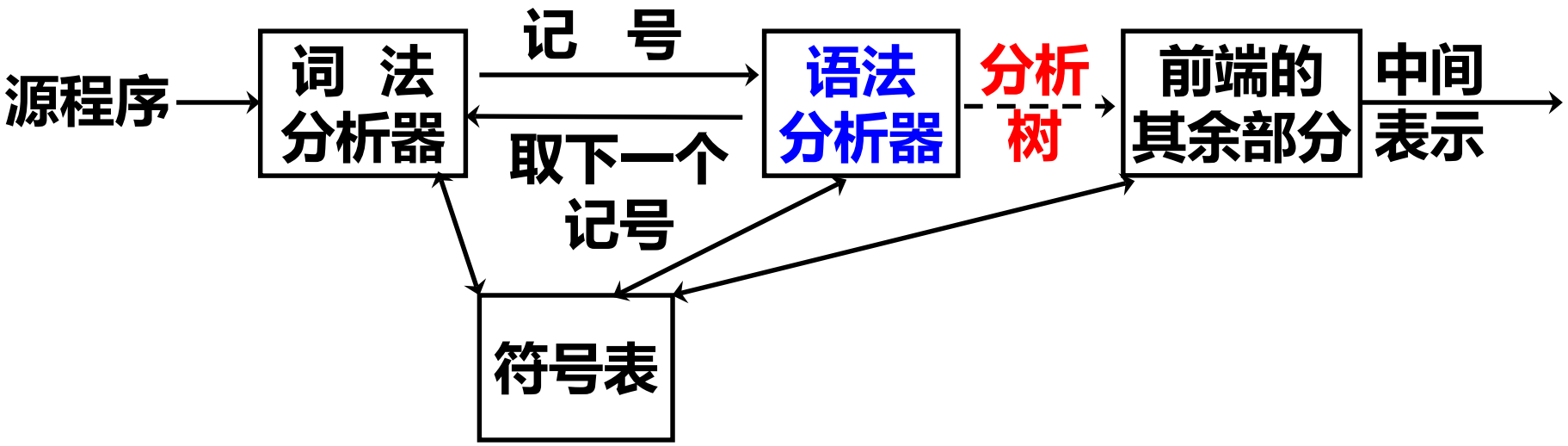
中国科学技术大学
University of Science and Technology of China

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
29/09/2018

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

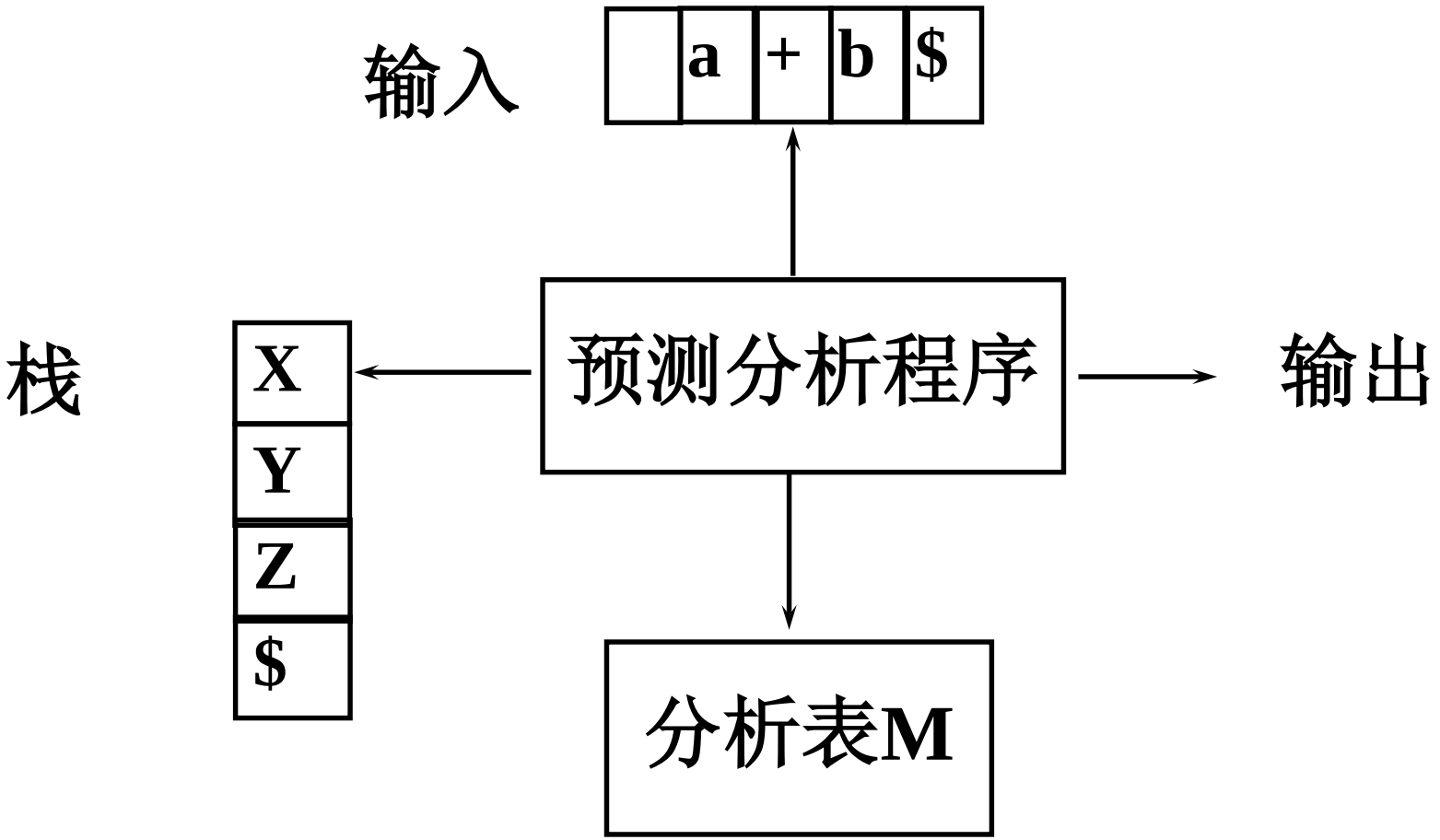
❑自顶向下（Top-down）

- ❖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

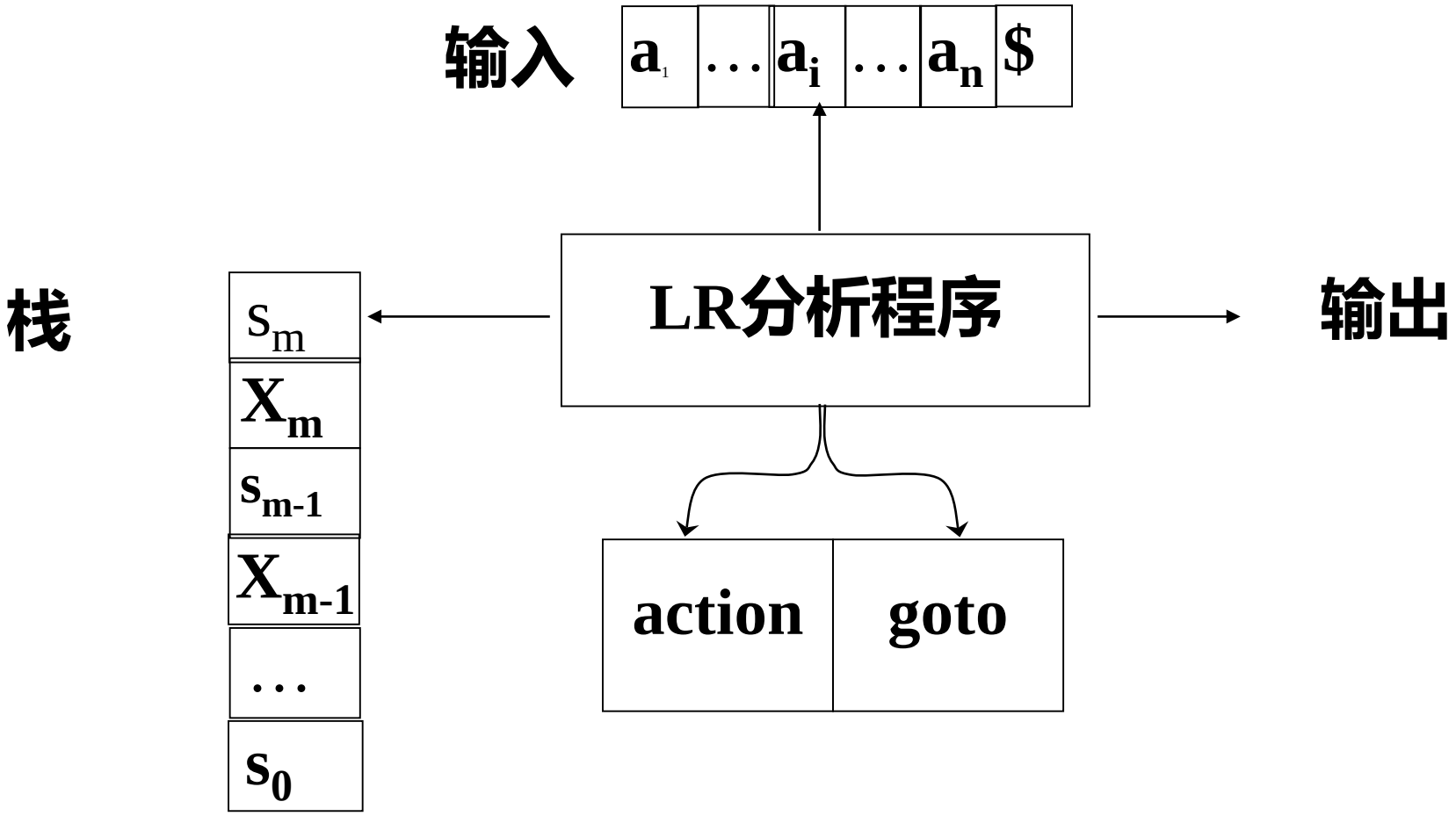


复习：LL(1)非递归分析

❑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			$F \rightarrow (E)$		

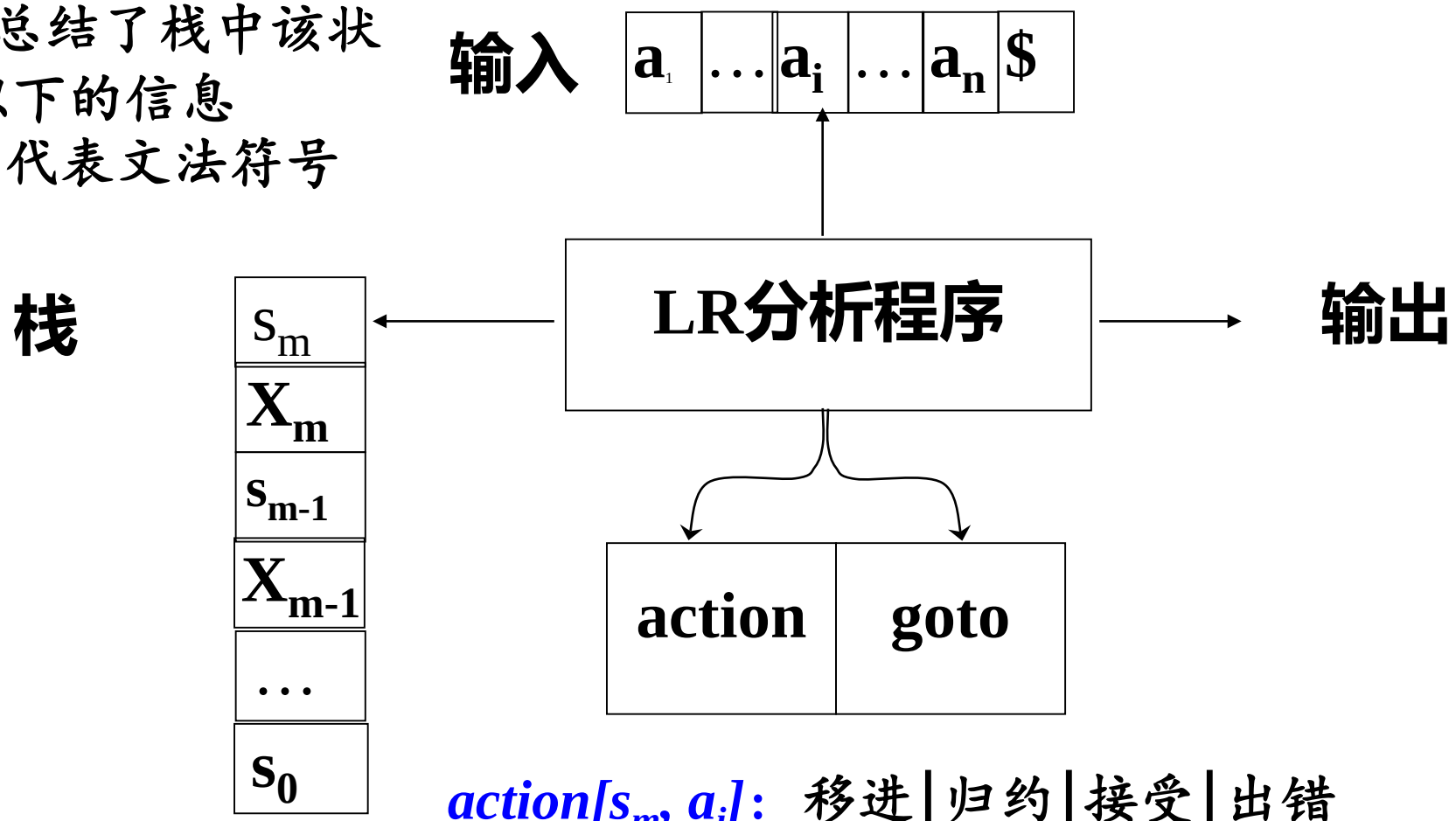
LR分析器



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) $E \rightarrow E + T$ (2) $E \rightarrow T$

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

(5) $F \rightarrow (E)$ (6) $F \rightarrow \mathrm{id}$

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
. . .	. . .	. . .

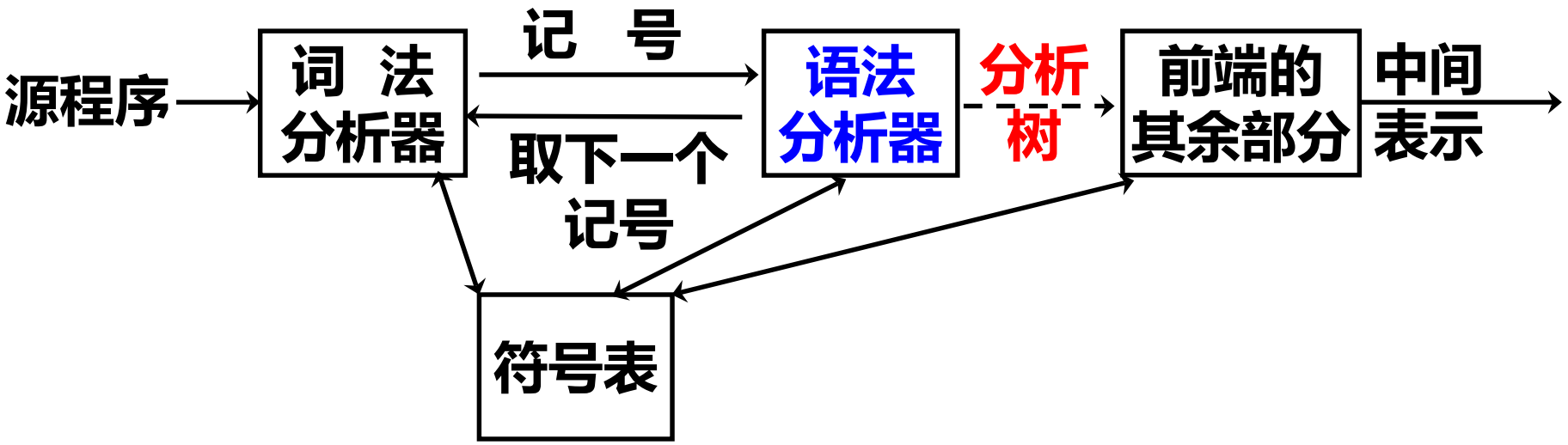
LR分析算法：举例

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

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）

- 关键在于构造LR分析表
  - 计算所有可能的状态
  - 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

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不是句柄且存在p′= pa
  - ➢何时归约？当p为句柄时

LR分析: 基本概念

- **活前缀 (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

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

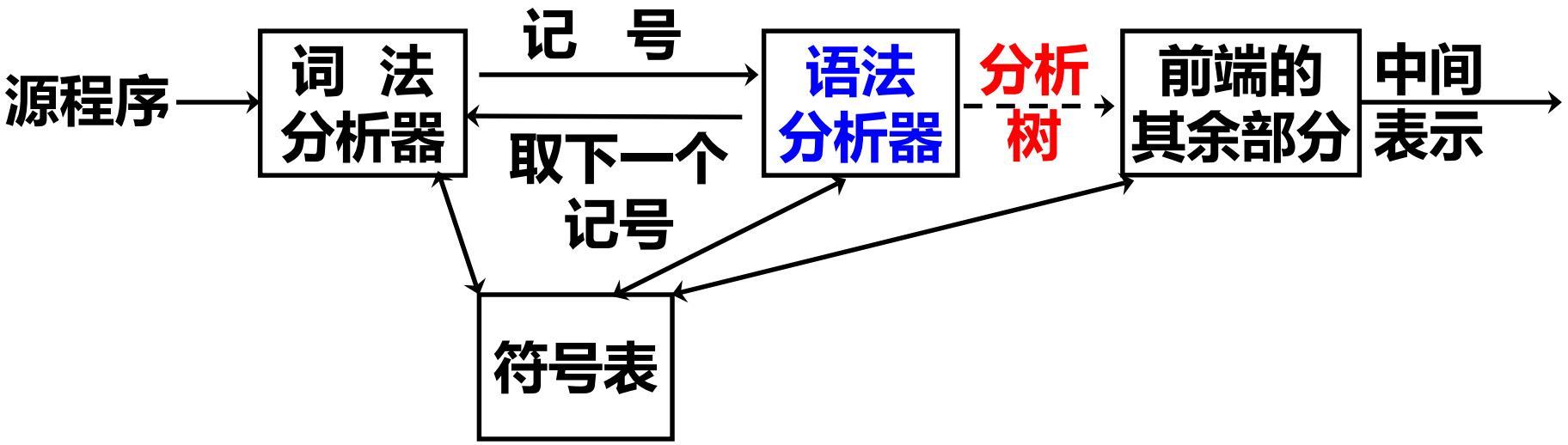
- 栈中的文法符号总是形成一个活前缀
- 分析表的转移函数本质上是识别活前缀的DFA
- 栈顶的状态符号包含确定句柄所需的一切信息

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

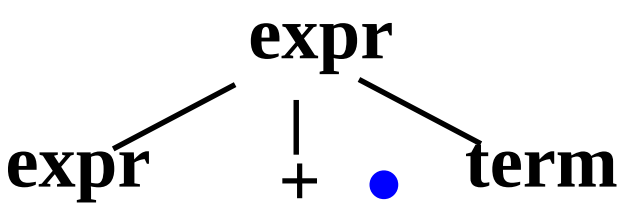
主要内容



- 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）

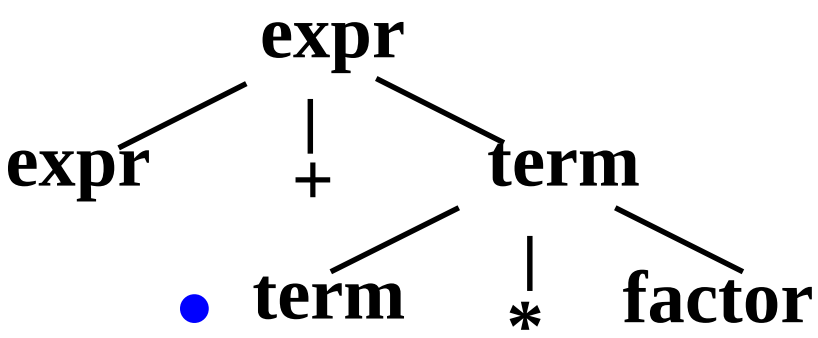
SLR分析表的构造

- 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



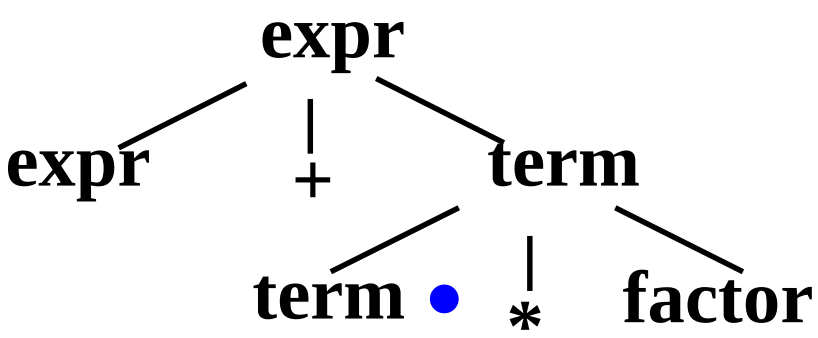
SLR分析表的构造

- 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 $A \rightarrow XYZ$对应有四个项目

$A \rightarrow \cdot XYZ$

$A \rightarrow X \cdot YZ$

$A \rightarrow XY \cdot Z$

$A \rightarrow XYZ \cdot$

点的左边代表历史信息，点的右边代表展望信息。

- 例 $A \rightarrow \varepsilon$只有一个项目和它对应

$A \rightarrow \cdot$

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

$E' \to E$

$E \to E + T \mid T$

$T \to T * F \mid F$

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

构造识别活前缀的DFA

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:

$E' \to \cdot E$ ← 核心项目，初始项+点不在最左边的项

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

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

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:

$E' \rightarrow \cdot E$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

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

$\xrightarrow{E}$ I_1:

$E' \rightarrow E \cdot$

$E \rightarrow E \cdot + T$

$I_1 := goto(I_0, E)$

2. 构造LR(0)项目集规范族

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

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

E

I_1:

$E' \to E \cdot$

$E \to E \cdot + T$

T

I_2:

$E \to T \cdot$

$T \to T \cdot * F$

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:

$F \to (\cdot E)$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

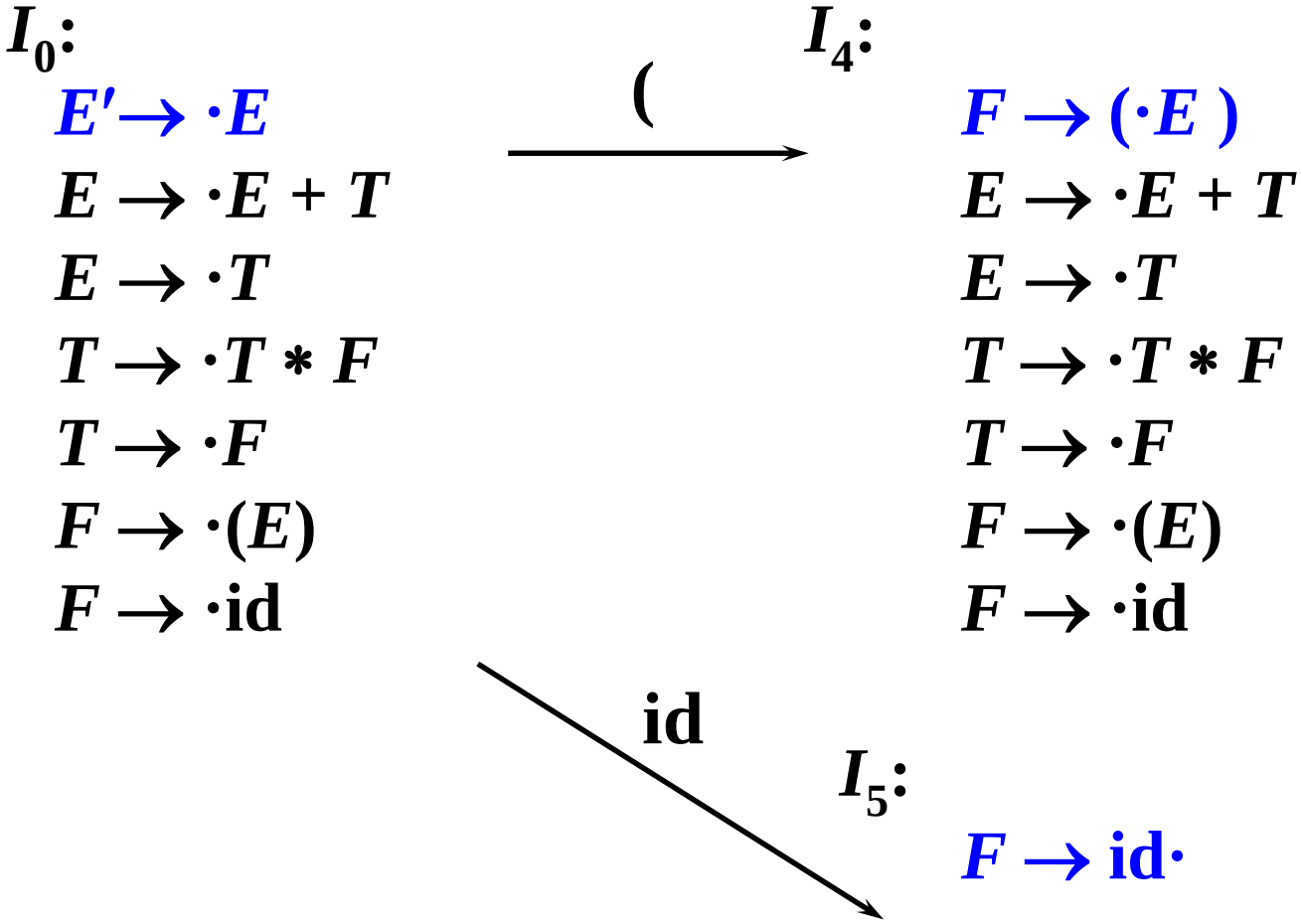
$T \to \cdot F$

$F \to \cdot (E)$

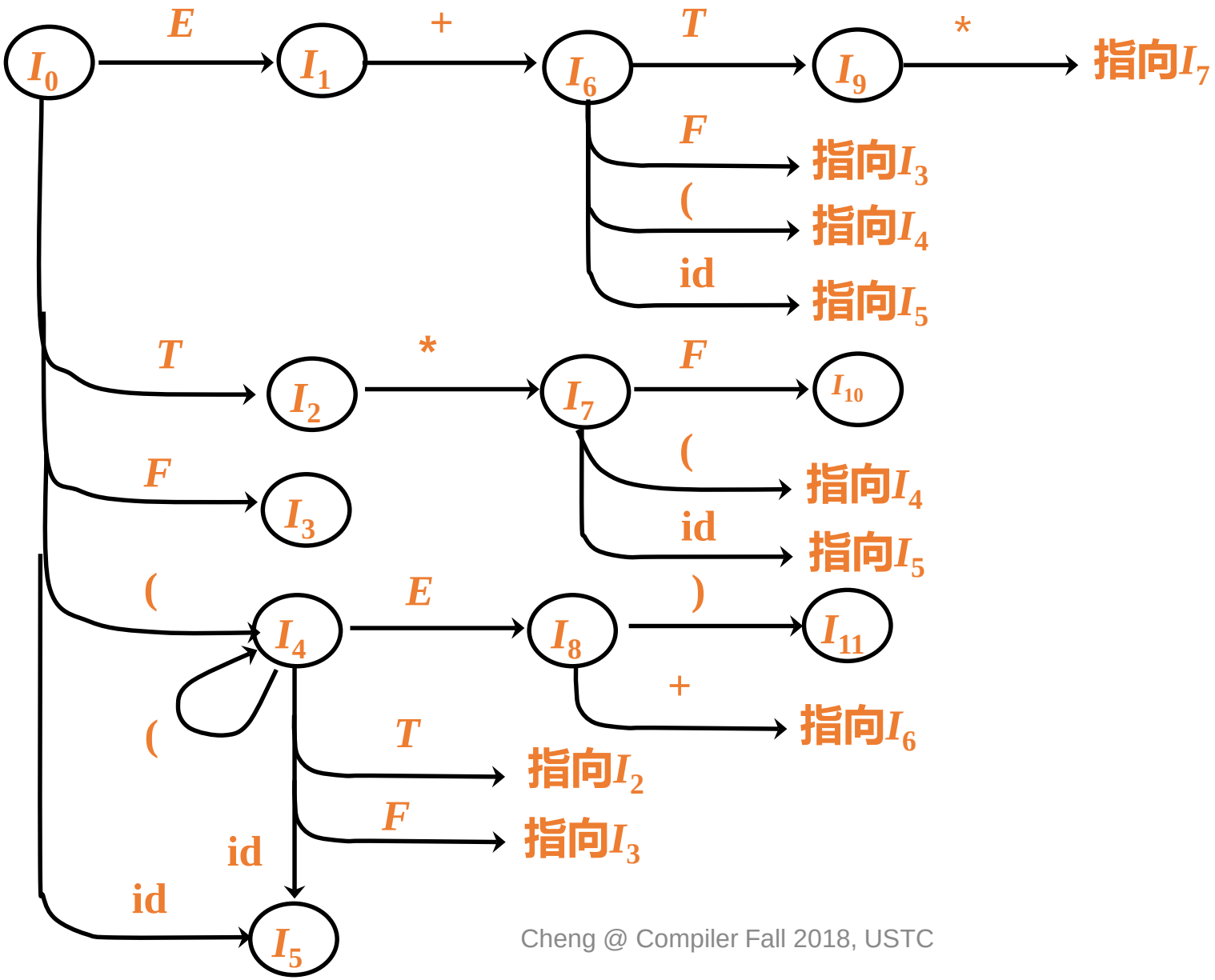
$F \to \cdot \mathbf{id}$

构造识别活前缀的DFA

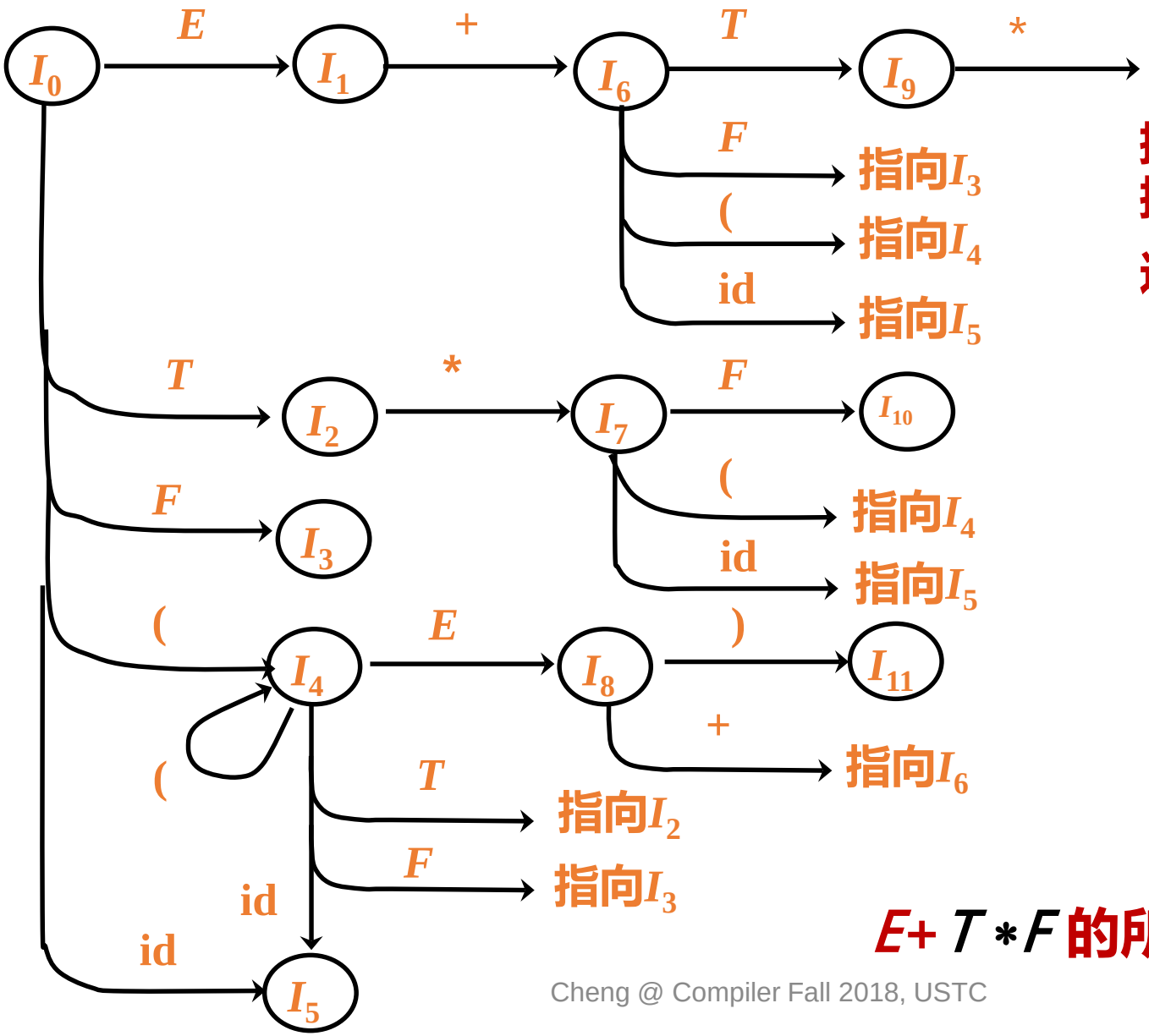
2. 构造LR(0)项目集规范族



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

❑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

$E' \to E$　　$E \to E + T \mid T$

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

$E \to \cdot E+T$ 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

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

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

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

$E' \Rightarrow E \Rightarrow E+T \Rightarrow E+T*F$

包含活前缀的最右推导

构造识别活前缀的NFA

每一个项目一个状态

I_0:

$E' \rightarrow \cdot E$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

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

构造识别活前缀的NFA

每一个项目一个状态

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

$E \to \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

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

$E' \to \cdot E$

构造识别活前缀的NFA

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

$E \to \cdot T$

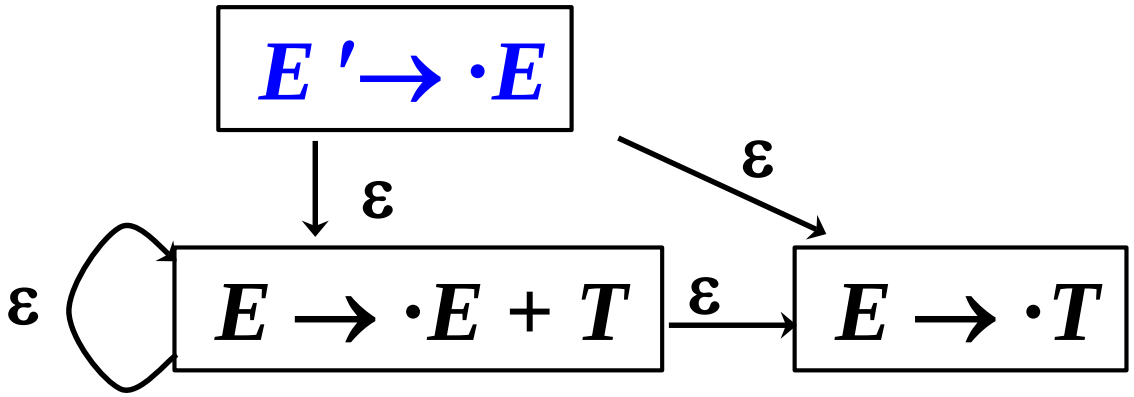
$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

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

每一个项目一个状态



构造识别活前缀的NFA

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

$E \to \cdot T$

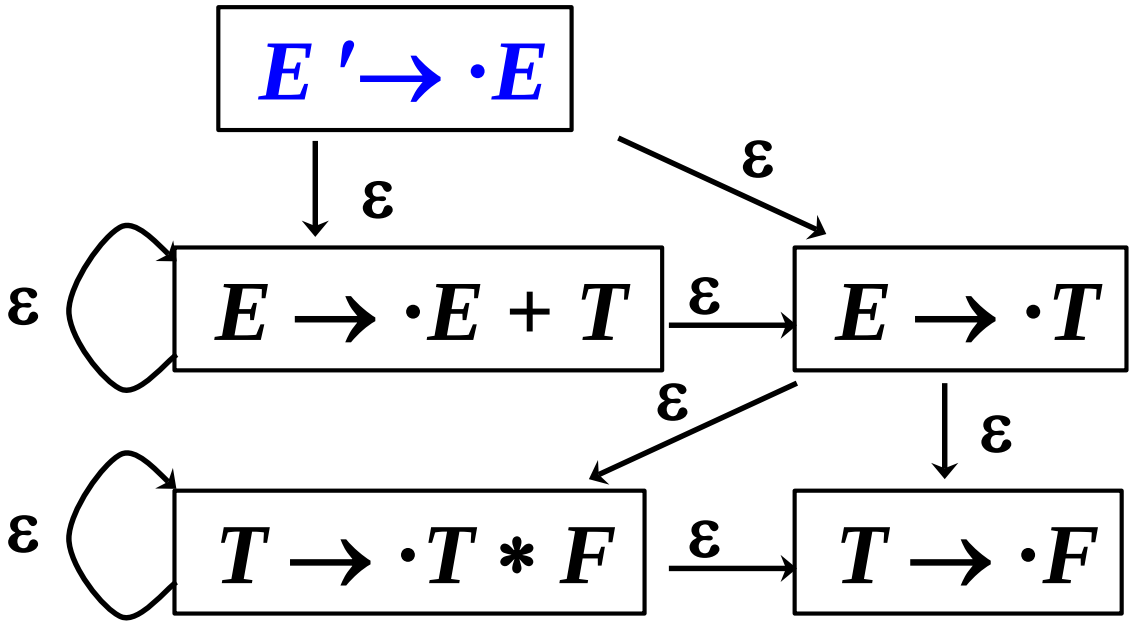
$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot(E)$

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

每一个项目一个状态



构造识别活前缀的NFA

I_0:

$E' \to \cdot E$

$E \to \cdot E + T$

$E \to \cdot T$

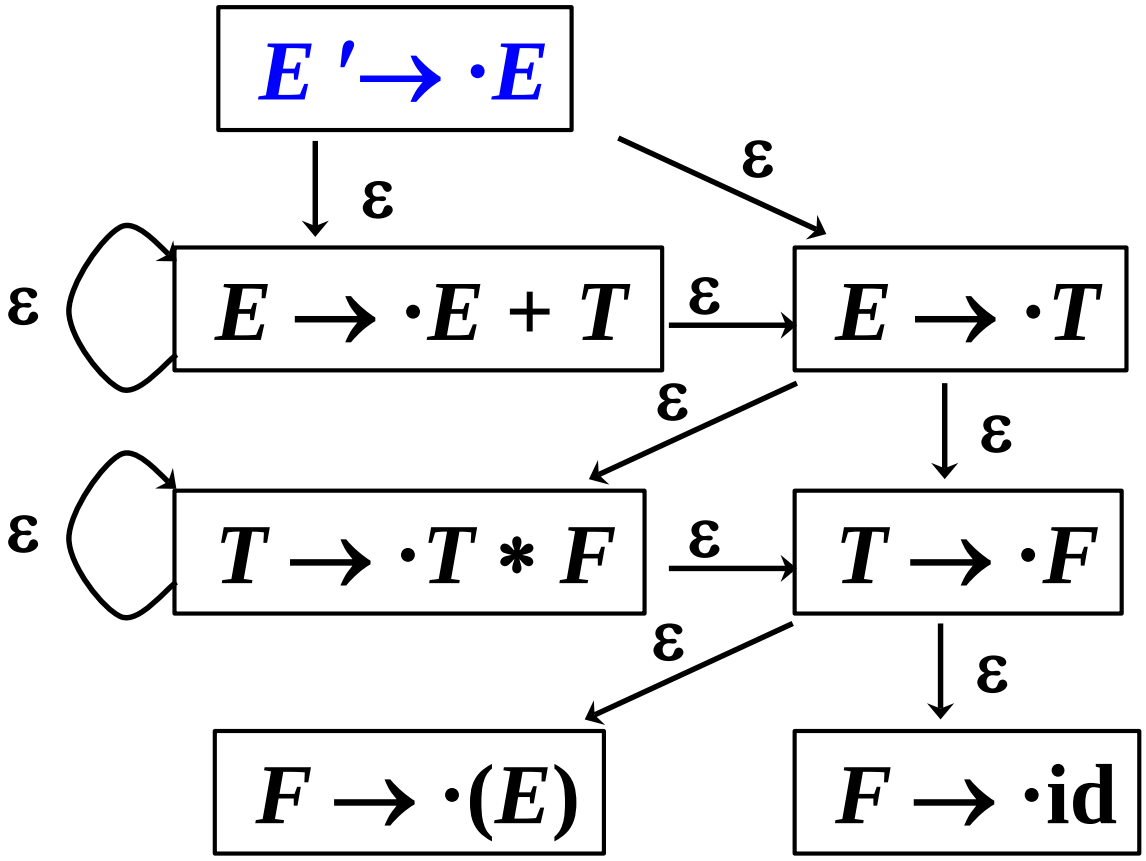
$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot(E)$

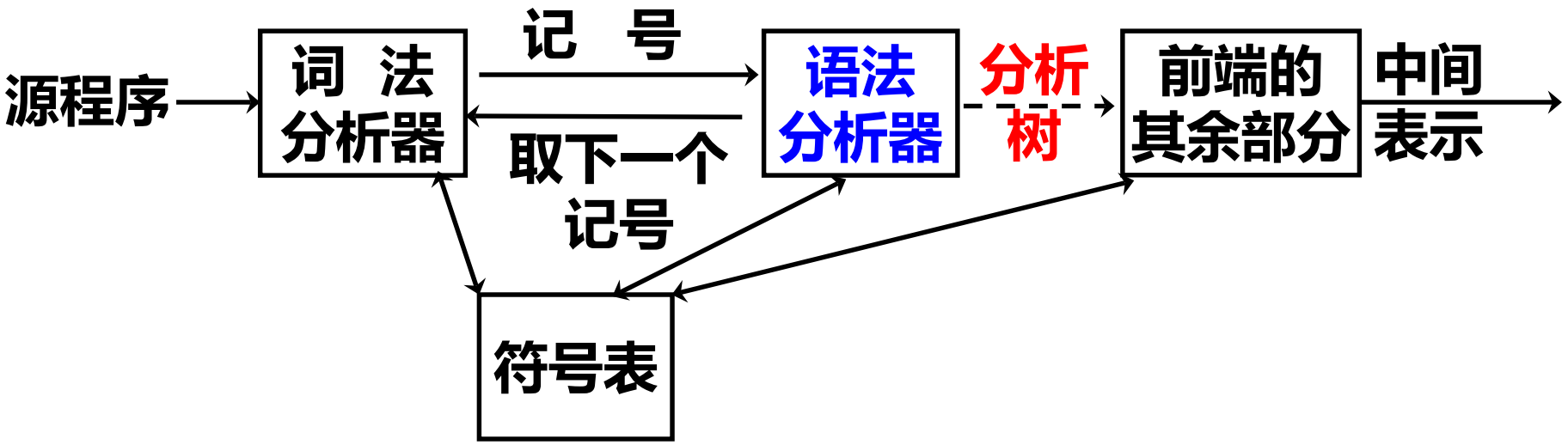
$F \to \cdot \mathrm{id}$

每一个项目一个状态



由NFA通过子集构造法可以得到一个DFA。

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*

- 如果出现动作冲突，那么该文法就不是SLR(1)

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**

❑例 I_2:

$$E \rightarrow T\cdot$$

$$T \rightarrow T\cdot * F$$

❖因为FOLLOW(E) = {\$, +,)}, 所以

action[2, \$]=*action*[2, +]=*action*[2,)]=*r*2

❖*action*[2, *] = *s*7

LR分析器

例 (1) $E \rightarrow E + T$ (2) $E \rightarrow T$

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

(5) $F \rightarrow (E)$ (6) $F \rightarrow$ 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

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***