

# 第八周编译原理笔记

laisg

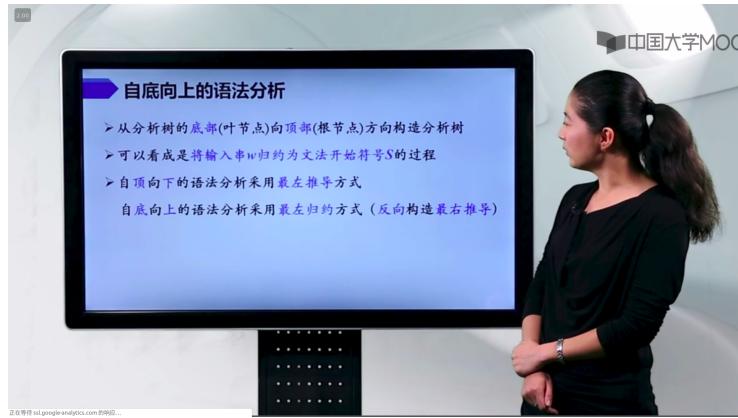
2019 年 4 月 16 日

## 1 预测分析中的错误恢复（重点在同步集合）

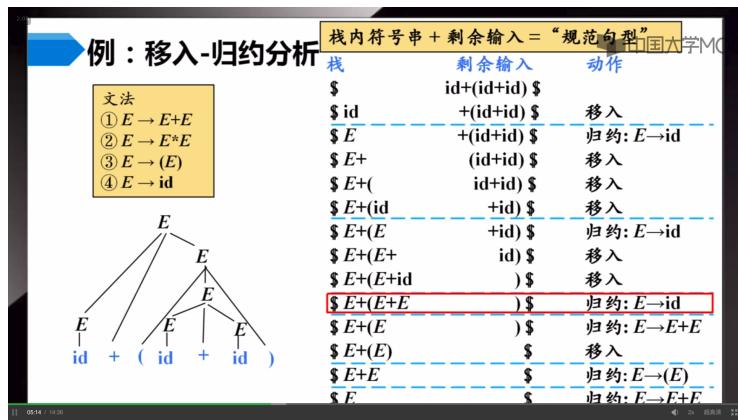


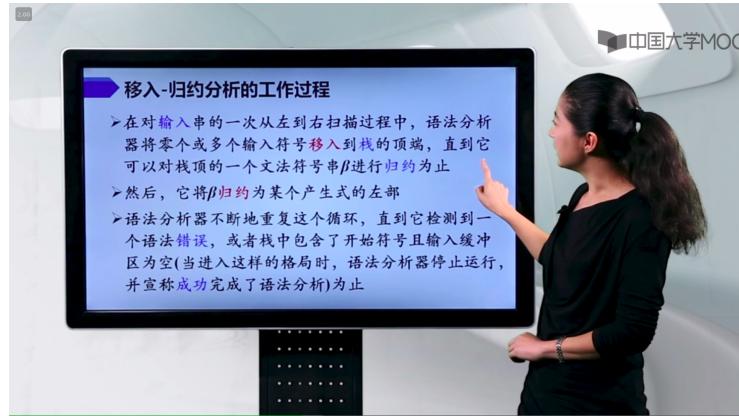
# Part I

## bottom-up analysis

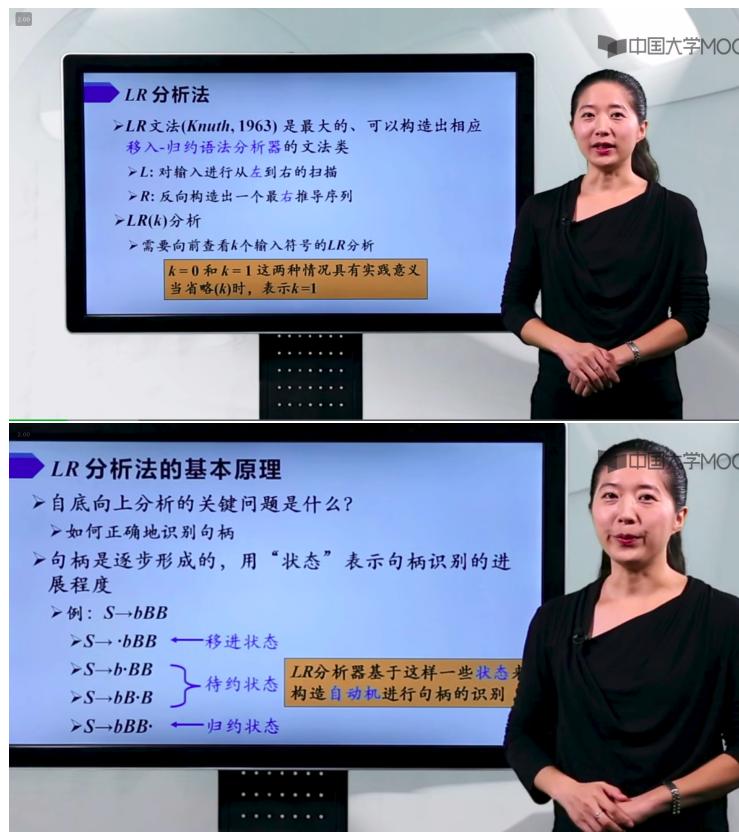


## 2 移入规约分析





### 3 LR 分析法



**LR 分析表的结构**

状态	ACTION			GOTO	
	a	b	\$	S	B
0	s3	s4		1	2
1			acc		
2	s3	s4			5
3	s3	s4			6
4	r3	r3	r3		
5	r1	r1	r1		
6	r2	r2	r2		

sn: 将符号 $a$ 、状态 $n$ 压入栈  
rn: 用第 $n$ 个产生式进行归约

**LR 分析算法**

```

输入: 字符串 $w$ 和LR语法分析表。该表描述了文法 $G$ 的ACTION函数和GOTO函数。
输出: 如果 $w$ 在 $L(G)$ 中，则输出 $w$ 的自底向上语法分析过程中的归约步骤；否则给出一个错误指示。
方法: 初始时，语法分析器栈中的内容为初始状态 $s_0$ ，输入缓冲区中的内容为 $w\$$ 。然后，语法分析器执行下面的程序：
令 $a$ 为 $w\$$ 的第一个字符;
while(1){/* 永远重复 */
    令 $s$ 是栈顶的状态;
    if (ACTION[s, a] = st) {
        将 $t$ 压入栈中;
        令 $a$ 为下一个输入字符;
    } else if (ACTION[s, a] = 归约 $A \rightarrow \beta$ ) {
        从栈中弹出 $|\beta|$ 个符号;
        将GOTO[t, A]压入栈中;
        输出产生式 $A \rightarrow \beta$ ;
    } else if (ACTION[s, a] = 接受)break; /* 语法分析完成*/
    else 调用错误恢复例程;
}

```

### 3.1 LR(0) 分析 && 增广文法

**LR(0) 项目**

右部某位置标有圆点的产生式称为相应文法的一个LR(0)项目（简称为项目）

例： $A \rightarrow a_1 \cdot a_2$

例： $S \rightarrow bBB$

- $S \rightarrow \cdot bBB$  ← 移进项目
- $S \rightarrow b \cdot BB$
- $S \rightarrow bB \cdot B$  } 待约项目
- $S \rightarrow BBB \cdot$  ← 归约项目

项目描述了句柄识别的状态

产生式 $A \rightarrow \epsilon$ 只生成一个项目 $A \rightarrow \cdot$

增广文法 (Augmented Grammar)

如果  $G$  是一个以  $S$  为开始符号的文法，则  $G$  的增广文法  $G'$  就是在  $G$  中加上新开始符号  $S'$  和产生式  $S' \rightarrow S$  而得到的文法。

例

1) $E \rightarrow E + T$	6) $E' \rightarrow E$
2) $E \rightarrow T$	7) $E \rightarrow \cdot$
3) $T \rightarrow T * F$	8) $T \rightarrow \cdot$
4) $T \rightarrow F$	9) $T \rightarrow T * F$
5) $F \rightarrow (E)$	10) $T \rightarrow \cdot F$
6) $F \rightarrow \text{id}$	11) $F \rightarrow (E)$

引入这个新的开始产生式的目的是使得文法开始符号仅出现在一个产生式的左边，从而使得分析器只有一个接受状态。

### 文法中的项目

①  $S' \rightarrow S$  ②  $S \rightarrow vI;T$  ③  $I \rightarrow I,i$  ④  $I \rightarrow i$  ⑤  $T \rightarrow r$

(2)  $S \rightarrow vI;T$   
(3)  $S \rightarrow vI;T$  (7)  $I \rightarrow I,i$   
(4)  $S \rightarrow vI;T$  (8)  $I \rightarrow I,i$   
(0)  $S' \rightarrow S$  (5)  $S \rightarrow vI;T$  (9)  $I \rightarrow I,i$  (11)  $I \rightarrow i$  (13)  $T \rightarrow r$   
(1)  $S' \rightarrow S$  (6)  $S \rightarrow vI;T$  (10)  $I \rightarrow I,i$  (12)  $I \rightarrow i$  (14)  $T \rightarrow r$

后继项目 (Successive Item)

同属于一个产生式的项目，但圆点的位置只相差一个，则称后者是前者的后继项目。

$A \rightarrow \alpha X \beta$  的后继项目是  $A \rightarrow \alpha X \beta$

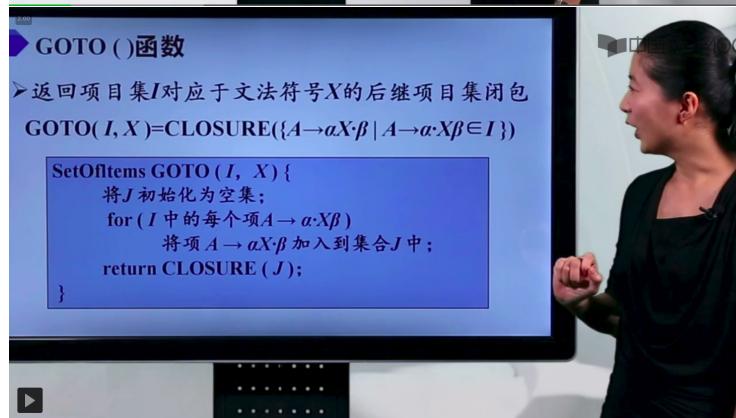
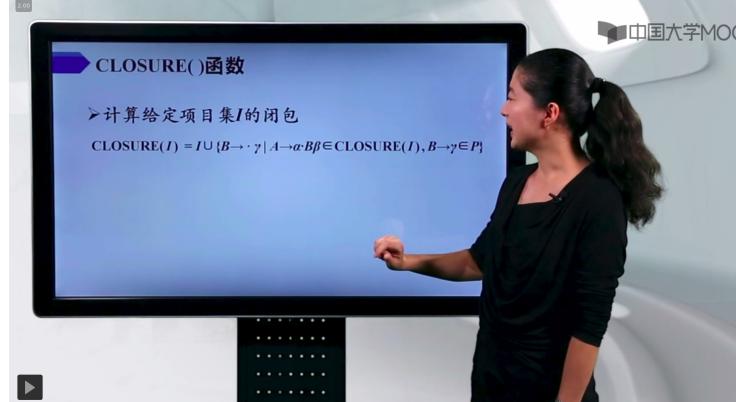
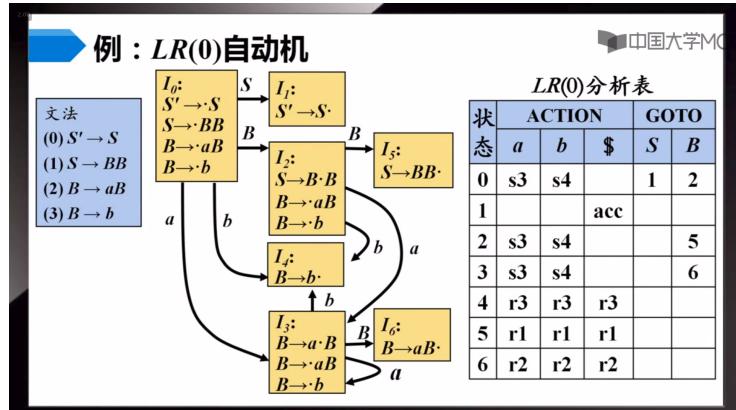
### 文法中的项目

①  $S' \rightarrow S$  ②  $S \rightarrow vI;T$  ③  $I \rightarrow I,i$  ④  $I \rightarrow i$  ⑤  $T \rightarrow r$

(2)  $S \rightarrow vI;T$   
(3)  $S \rightarrow vI;T$  (7)  $I \rightarrow I,i$   
(4)  $S \rightarrow vI;T$  (8)  $I \rightarrow I,i$   
(0)  $S' \rightarrow S$  (5)  $S \rightarrow vI;T$  (9)  $I \rightarrow I,i$  (11)  $I \rightarrow i$  (13)  $T \rightarrow r$   
(1)  $S' \rightarrow S$  (6)  $S \rightarrow vI;T$  (10)  $I \rightarrow I,i$  (12)  $I \rightarrow i$  (14)  $T \rightarrow r$

这15个项目中是否会有某些项目是“等价”的？

可以把所有等价的项目组成一个项目集 ( $I$ )，称为项目集 (Item Set)。闭包 (Closure of Item Sets)，每个项目集闭包对应着自动机的一个状态。



▶ 构造LR(0)自动机的状态集

➤ 规范LR(0)项集族(Canonical LR(0) Collection)

$$C = \{I_0\} \cup \{I \mid \exists J \in C, X \in V_N \cup V_T, I = \text{GOTO}(J, X)\}$$

```

void items( G' ) {
    C = { 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在某一轮中没有新的项集被加入到C中;
}

```

▶ LR(0)分析表构造算法

➤ 构造 $G'$ 的规范LR(0)项集族 $C = \{I_0, I_1, \dots, I_n\}$

➤ 令 $I_i$ 对应状态*i*。状态*i*的语法分析动作按照下面的方法决定：

- if  $A \rightarrow a \cdot \alpha \beta \in I_i$  and  $\text{GOTO}(I_i, a) = I_j$  then  $\text{ACTION}[i, a] = s_j$
- if  $A \rightarrow a \cdot B \beta \in I_i$  and  $\text{GOTO}(I_i, B) = I_j$  then  $\text{GOTO}[i, B] = j$
- if  $A \rightarrow a \cdot \in I_i$  且  $A \neq S'$  then for  $\forall a \in V_T \cup \{\$\}$  do  $\text{ACTION}[i, a] = r_j$   
( $j$ 是产生式 $A \rightarrow a$ 的编号)
- if  $S' \rightarrow S \cdot \in I_i$  then  $\text{ACTION}[i, \$] = acc$
- 没有定义的所有条目都设置为“error”

▶ 例：移进/归约冲突和归约/归约冲突

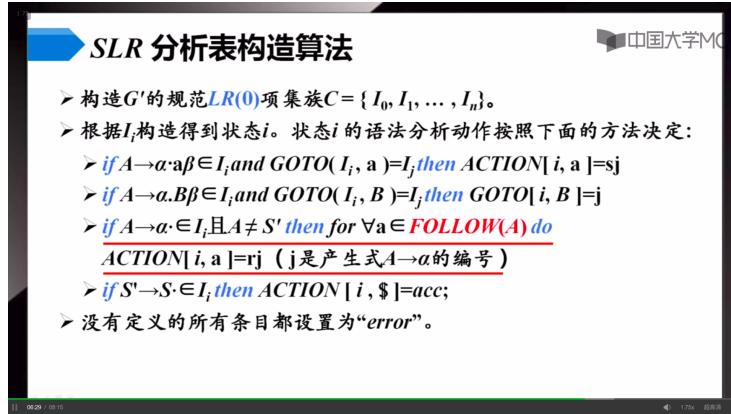
文法

- (0)  $S' \rightarrow T$
- (1)  $T \rightarrow aBd$
- (2)  $T \rightarrow c$
- (3)  $B \rightarrow Tb$
- (4)  $B \rightarrow e$

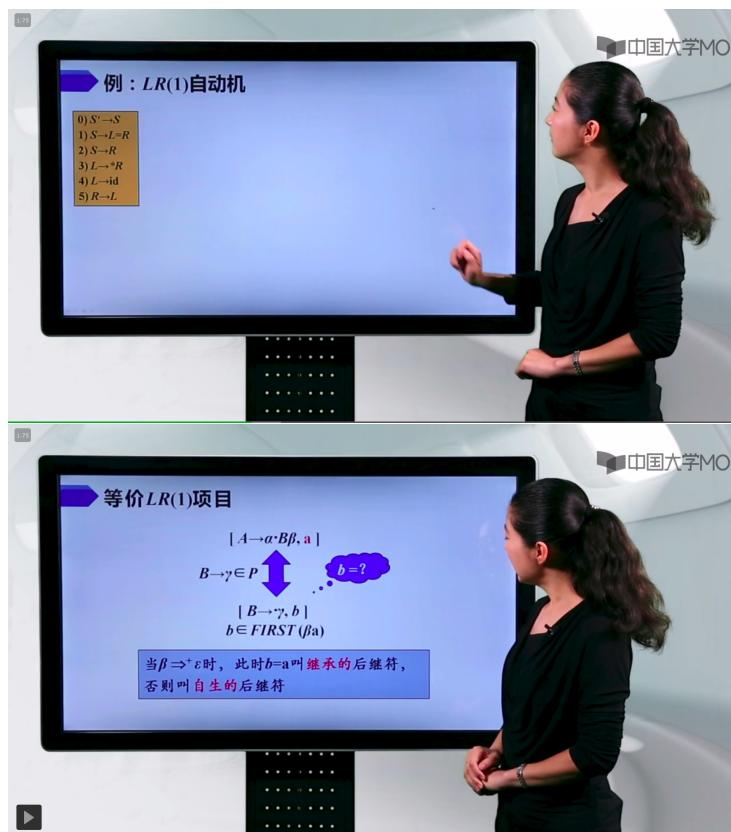
如果LR(0)分析表中没有语法分析动作冲突，那么给定的文法就称为LR(0)文法

不是所有CFG都能用LR(0)方法进行分析。也就是说，CFG不总是LR(0)文法

## 3.2 SLR



## 3.3 LR(1)



**LR(1)项目集闭包的计算**

$CLOSURE(I) = I \cup \{ [B \rightarrow \gamma, b] \mid [A \rightarrow a\beta, a] \in CLOSURE(I), B \rightarrow \gamma \in P, b \in FIRST(\beta a)\}$

```

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

```

**LR(1)后继项目集闭包的计算**

$GOTO(I, X) = CLOSURE(\{[A \rightarrow aX\beta, a] \mid [A \rightarrow aX\beta, a] \in I\})$

```

SetOfItems GOTO ( I, X ) {
    将 J 初始化为空集;
    for ( I 中的每个项 [A → a·Xβ, a] )
        将项 [A → aXβ, a] 加入到集合 J 中;
    return CLOSURE ( J );
}

```

**为文法  $G'$  构造规范 LR(1) 项集族**

```

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;
}

```