LR文法

移入-归约分析

例：设有以下文法G：

```
E → E + T | T
T → T * F | F
F → (E) | id
```

设输入为“id+id*id”，采用如下的方法分析：

栈	剩余输入	动作
$	id+id*id$	移入
$id	+id*id$	归约：F→id
$F	+id*id$	归约：T→F
$T	+id*id$	归约：E→T
$E	+id*id$	移入
$E+	id*id$	移入
$E+id	*id$	归约：F→id
$E+F	*id$	归约：T→F
$E+T	*id$	移入
$E+T*	id$	移入
$E+T*id	$	归约：F→id
$E+T*F	$	归约：T→T*F
$E+T	$	归约：E→E+T
$E	$	接受

栈的初始为空，表示为“$”；剩余输入的初始为输入字符串加上“$”

检查栈中的元素，如果栈顶存在能按照G的文法归约的部分就将其规约，否则将剩余输入入栈，直到栈中为开始符号，且输入为空，接受该文法。

每次归约的符号串称为“句柄”。栈内符号串和剩余输入构成“规范句型”。

LR分析表结构

例：对于下面的文法G

```
S → bBB
B → aB
B → b
```

假设输入的符号串为bab。

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表示将符号a、状态n压栈，rn表示将第n个产生式归约。表格的内容是已知的，而不是要求填写的。根据表格进行移入-归约分析。

LR分析器的工作过程

初始化：

$$\begin{array}{ll} s_0 & \\ \$ & a_1a_2\ldots a_n\ \$ \end{array}$$

一般情况下：

$$\begin{array}{ll} s_0s_1\ldots s_m & \\ \$X_1\ldots X_m & a_ia_{i+1}\ldots a_n\ \$ \end{array}$$

①如果$ACTION[s_m, a_i] = sx$

$$\begin{array}{ll} s_0s_1\ldots s_mx & \\ \$X_1\ldots X_ma_i & a_{i+1}\ldots a_n\ \$ \end{array}$$

②如果$ACTION[s_m, a_i] = rx$表示用第x个产生式$A \to X_{m-(k-1)}\cdots X_m$进行归约，那么格局变为：

$$\begin{array}{ll} s_0\,s_1\ldots s_{m\text{-}k} & \\ \$\,X_1\ldots X_{m\text{-}k}A & a_i\,a_{i+1}\ldots a_n\ \$ \end{array}$$

如果$GOTO[s_{m-k,A}] = y$，那么格局变为：

$$\begin{array}{ll} s_0\,s_1\ldots s_{m\text{-}k}\ y & \\ \$\,X_1\ldots X_{m\text{-}k}A & a_i\,a_{i+1}\ldots a_n\ \$ \end{array}$$

③如果$ACTION[s_m, a_i] = acc$，那么分析成功

④如果$ACTION[s_m, a_i] = err$，那么出现语法错误

LR分析算法：

```
令a为w$的第一个符号;
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为下一个输入符号;
    } else if(ACTION[s,a]=归约A→β){
        从栈中弹出|β|个符号;
        将GOTO[t,A]压入栈中;
        输出产生式A→β;
    } else if(ACTION[s,a]=acc) break; /*语法分析完成*/
    else 调用错误恢复例程;
}
```

LR(0)

增广文法：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'→S而得到的文法G'

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状态。

例：

给定文法G，由5个产生式组成，一共15个项目如下表所示：

①S'→S	②S→vI:T	③I→I,i	④I→i	⑤T→r
	(2)S→·vI:T			
	(3)S→v·I:T	(7)I→·I,i		
	(4)S→vI·:T	(8)I→I·,i		
(0)S'→·S	(5)S→vI:·T	(9)I→I,·i	(11)I→·i	(13)T→·r
(1)S'→S·	(6)S→vI:T·	(10)I→I,i·	(12)I→i·	(14)T→r·

其中等价的项目有：(0)(2)、(3)(7)(11)、(5)(13)，因为当项目圆点后面是一个非终结符，存在等价项目。

把所有等价的项目组成一个项目集合(I)，称为项目集闭包，每个项目集闭包对应自动机的一个状态。

例：设文法G：(1)S'→S；(2)S→BB；(3)B→aB；(4)B→b，构造其LR(0)自动机。

初始状态I_0：S'→·S；S→·BB；B→·aB；B→·b

$I_0 \xrightarrow{S} I_1$：S'→S·

$I_0 \xrightarrow{B} I_2$：S→B·B；B→·aB；B→·b

$I_0 \xrightarrow{a} I_3$：B→a·B；B→·aB；B→·b

$I_0 \xrightarrow{b} I_4$：B→b·

$I_2 \xrightarrow{B} I_5$：S→BB·

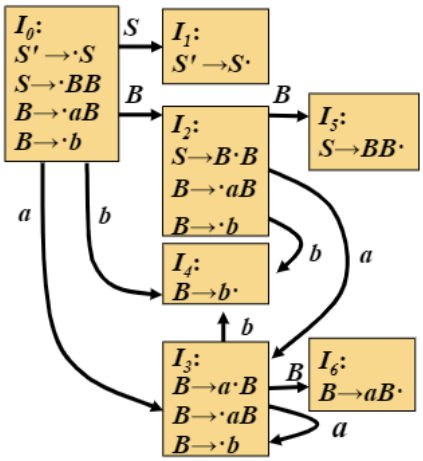
$I_2 \xrightarrow{a} I_3$：B→a·B；B→·aB；B→·b

$I_2 \xrightarrow{b} I_4$：B→b·

$I_3 \xrightarrow{B} I_6$：B→aB·

$I_3 \xrightarrow{a} I_3$：B→a·B；B→·aB；B→·b

$I_3 \xrightarrow{b} I_4$：B→b·



于是可以构造出前文所示的LR(0)分析表。

CLOSURE()函数：

```
SetOfItems CLOSURE(I){
    J=I;
    repeat
        for(J中的每一个项A→α·Bβ)
            for(G的每个产生式B→γ)
                if(项B→·γ不在J中)
                    将B→·γ加入J中;
    until 在某一轮中没有新的项被加入到J中;
    return J;
}
```

GOTO()函数：

```
SetOfItms GOTO(I,X){
    将J初始化为空集;
    for(I中的每个项A→α·Xβ)
        将项A→αX·β加入到集合J中;
    return CLOSURE(J);
}
```

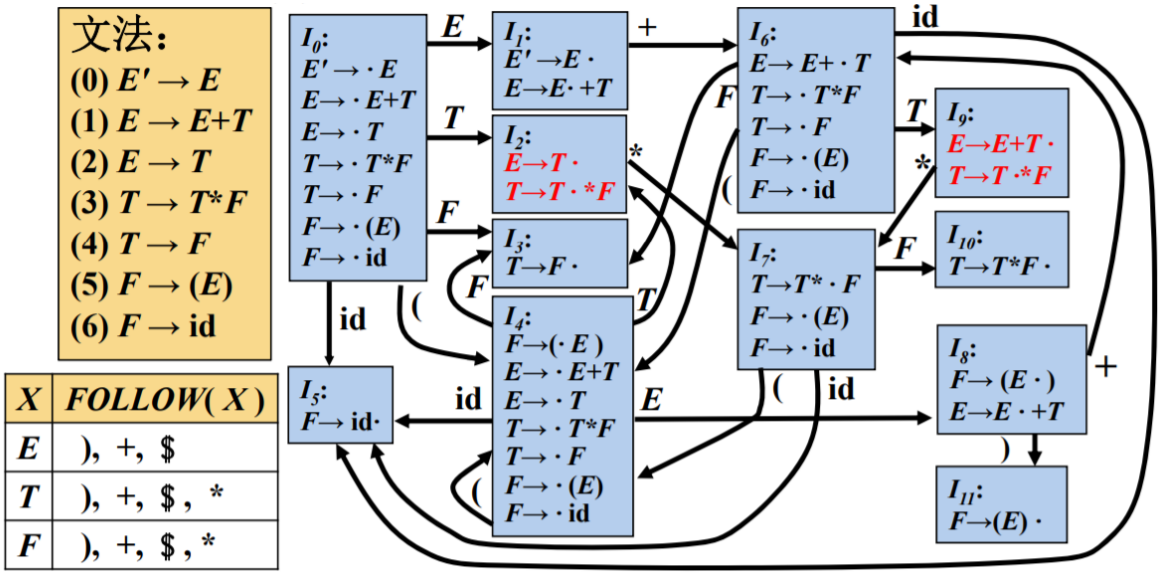
LR(0)状态集构造：

```
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)中存在移进/归约冲突和归约/归约冲突。因此有如下的SLR分析。

SLR

例：



通过计算存在冲突元素的FOLLOW集，如果下一个输入符号在元素的FOLLOW集中，那么就归约；否则移入。上例中，当输入为"*"时，"*"不属于E的FOLLOW集，将其移入。

SLR分析表：

状态	ACTION						GOTO		
	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
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

SLR分析思想：

已知项目集I：

①m个移进项目：

$A_1 \to \alpha_1 \cdot a_1\beta_1$
$A_2 \to \alpha_2 \cdot a_2\beta_2$
$\cdots$
$A_m \to \alpha_m \cdot a_m\beta_m$

②n个归约项目：

$B_1 \to \gamma_1 \cdot$
$B_2 \to \gamma_2 \cdot$
$\cdots$
$B_n \to \gamma_n \cdot$

如果集合$\{a_1, a_2, \ldots, a_m\}$和$FOLLOW(B_1)$，$FOLLOW(B_2)$，...，$FOLLOW(B_n)$两两不相交，则项目集I中的冲突可以按以下原则解决：

设a是下一个输入符号

- 若$a \in \{a_1, a_2, \ldots, a_m\}$，则移进a
- 若$a \in FOLLOW(B_i)$，则用产生式$B_i \to \gamma_i$归约
- 此外，报错

由于SLR分析也会存在冲突，所以有如下的LR(1)分析。

LR(1)

将一般形式为$[A \to \alpha \cdot \beta, a]$的项称为 LR(1) 项，其中$A \to \alpha\beta$是 一个产生式，a是一个终结符(这里将$视为一个特殊的终结符)，它表示在当前状态下，A后面必须紧跟的终结符，称为该项的展望符。

例：设文法G：(0) S' → S；(1) S →L=R；(2) S → R；(3) L →*R；(4) L →id；(5) R → L，构造其LR(1)自动机

初始状态I_0：S' → ·S，$；S → ·L=R；S → ·R，$；L →·*R，=；L →·id，=；R → ·L，$；L →·*R，$；L →·id，$

（`L →·*R，=；L →·id，=`，这两个式子是由于`S → ·L=R`中圆点后面是非终结符，所以写L的两个产生式，而由于跟在L后的是“=”，所以后面用“=”；而在`L →·*R，$；L →·id，$`中则是“$”）

$I_0 \xrightarrow{S} I_1$：S'→S·，$；

$I_0 \xrightarrow{L} I_2$：S→L·=R，$；R→L·，$

$I_0 \xrightarrow{R} I_3$：S→R·，$

$I_0 \xrightarrow{*} I_4$：L → *·R，=；L → *·R，$；R → ·L，=；R → ·L，$；L → ·*R，=；L → ·*R ，$；L → ·id，=；L → ·id，$

$I_0 \xrightarrow{id} I_5$：L → id·；=；L → id·，$

$I_2 \xrightarrow{=} I_6$：S → L = ·R，$；R → ·L，$；L → ·*R，$；L → ·id，$

$I_4 \xrightarrow{R} I_7$：L → *R·，=；L → *R·；$

$I_4 \xrightarrow{L} I_8$：R → L·，=；R → L·，$

$I_4 \xrightarrow{id} I_5$：L → id·；=；L → id·，$

$I_6 \xrightarrow{R} I_9$：S → L=R·，$

$I_6 \xrightarrow{L} I_{10}$：R → L·，$

$I_6 \xrightarrow{*} I_{11}$：L → *·R，$；R → ·L，$；L → ·*R，$；L → ·id，$

$I_6 \xrightarrow{*} I_{12}$：L → id·，$

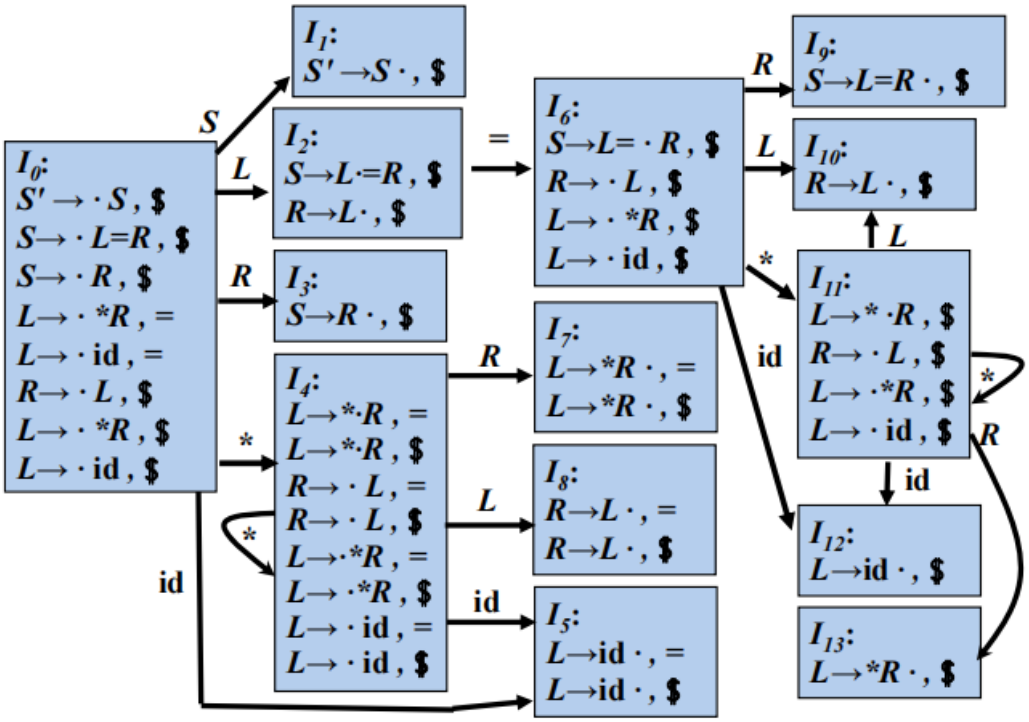
$I_6 \xrightarrow{id} I_{13}$：L → *R·；$

$I_{11} \xrightarrow{R} I_{13}$：L → *R·；$

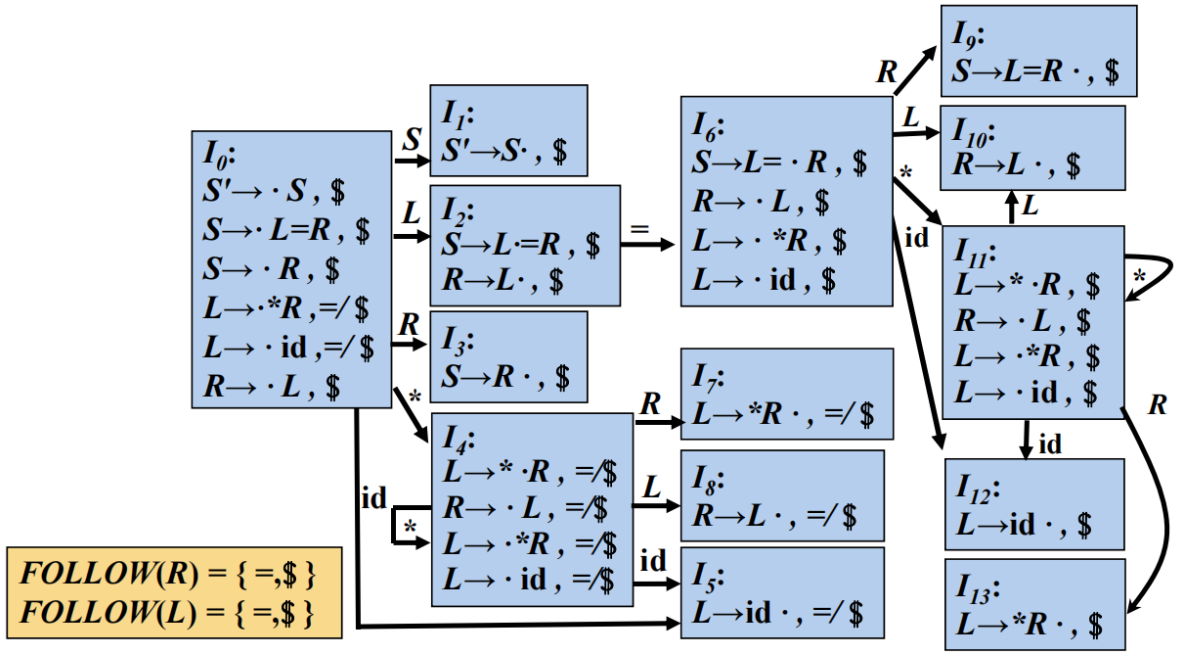
$I_{11} \xrightarrow{L} I_{10}$：L → id·，$

$I_{11} \xrightarrow{*} I_{11}$：L → *·R，$；R → ·L，$；L → ·*R，$；L → ·id，$

$I_{11} \xrightarrow{id} I_{12}$：L → id·，$



适当化简，



LR(1)分析表：

状态	ACTION				GOTO		
	*	id	=	$	S	L	R
0	s4	s5			1	2	3
1				acc			
2			s6	r5			
3				r2			
4	s4	s5				8	7
5			r4	r4			
6	s11	s12				10	9
7			r3	r3			
8			r5	r5			
9				r1			
10				r5			
11	s11	s12				10	13
12				r4			
13				r3			

CLOSURE()函数：

```
SetOfItems CLOSURE(I){
    repeat
        for(I中的每一个项[A→α·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;
    return I;
}
```

GOTO()函数：

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

LR(1)项集族构造：

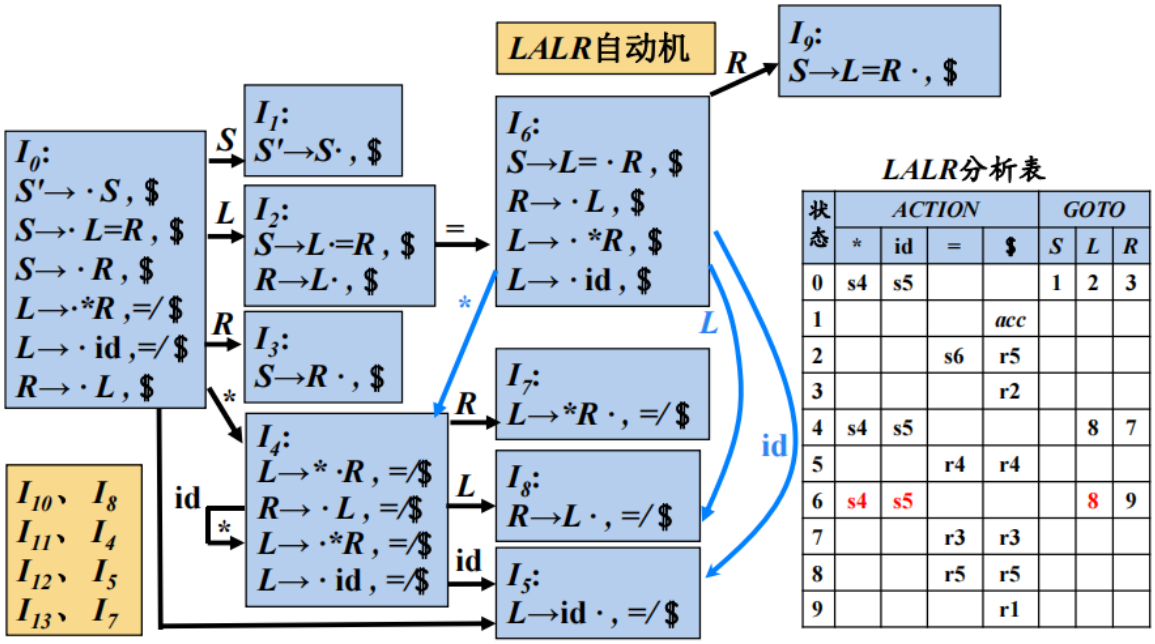
```
void items(G'){
    将C初始化为{CLOSURE({[S'→·S,$]})};
    repeat
        for(C中的每个项集I)
            for(每个文法符号X)
                if(GOTO(I,X)非空且不在C中)
                    将GOTO(I,X)加入C中;
    until 不再有新的项集加入到C中;
}
```

LALR

基本思想：

- 寻找具有相同核心的LR(1)项集，并将这些项集合并为一个项集。所谓项集的核心就是其第一分量的集合；
- 然后根据合并后得到的项集族构造语法分析表
- 如果分析表中没有语法分析动作冲突，给定的文法就称为LALR(1)文法，就可以根据该分析表进行语法分析。

接上例，构造LALR自动机，需要合并同心项集。



同心项集合并可能会产生归约/归约冲突。

特点：

- 形式上与LR(1)相同
- 大小上与LR(0)/SLR相当
- 分析能力介于SLR和LR(1)二者之间：SLR < LALR(1) < LR(1)，合并后的展望符集合仍为FOLLOW集的子集。