编译原理：LL1、LR0、SLR1、LR1、LALR1

1 FIRST&FOLLOW

1.1 FIRST集合

First(α) 是可从 α 推导得到的句型的首终结符号 的集合

Definition ($\textsc{First}(\alpha)$ 集合)
对于任意的 (产生式的右部) $\alpha \in (N \cup T)^*$:

$$\textsc{First}(\alpha) = \left\{ t \in T \cup \{\epsilon\} \mid \alpha \overset{*}{\Rightarrow} t\beta \vee \alpha \overset{*}{\Rightarrow} \epsilon \right\}.$$

计算FIRST(X)集合算法：

- 计算每个符号X的First(X)集合

```
1: procedure FIRST(X)
2:     if X ∈ T then                                  ▷ 规则 1: X 是终结符
3:         FIRST(X) = X
4:     for X → Y1Y2...Yk do                           ▷ 规则 2: X 是非终结符
5:         FIRST(X) ← FIRST(X) ∪ {FIRST(Y1) \ {ε}}
6:         for i ← 2 to k do
7:             if ε ∈ L(Y1...Yi−1) then
8:                 FIRST(X) ← FIRST(X) ∪ {FIRST(Yi) \ {ε}}
9:         if ε ∈ L(Y1...Yk) then                     ▷ 规则 3: X 可推导出空串
10:            FIRST(X) ← FIRST(X) ∪ {ε}
```

不断应用上面的规则, 直到每个 $\textsc{First}(X)$ 都不再变化 (**闭包!!!**)

- 计算每个符号串 α 的 First(α) 集合

$$\alpha = X\beta$$

$$\textsc{First}(\alpha) = \begin{cases} \textsc{First}(X) & \epsilon \notin L(X) \\ (\textsc{First}(X) \setminus \{\epsilon\}) \cup \textsc{First}(\beta) & \epsilon \in L(X) \end{cases}$$

最后, 如果 $\epsilon \in L(\alpha)$, 则将 ϵ 加入 $\textsc{First}(\alpha)$。

1.2 FOLLOW集合

Follow(A) 是可能在某些句型中紧跟在 A 右边的终结符的集合

Definition ($\textsc{Follow}(A)$ 集合)
对于任意的 (产生式的左部) 非终结符 $A \in N$:

$$\textsc{Follow}(A) = \left\{ t \in T \cup \{\$\} \mid \exists s.\ S \overset{*}{\Rightarrow} s \triangleq \beta A t \gamma \right\}.$$

计算FOLLOW(X)集合算法：

- 为每个非终结符 X 计算 Follow(X) 集合

```
1: procedure FOLLOW(X)
2:     for X 是开始符号 do                            ▷ 规则 1: X 是开始符号
3:         FOLLOW(X) ← FOLLOW(X) ∪ {$}
4:     for A → αXβ do                                 ▷ 规则 2: X 是某产生式右部中间的一个符号
5:         FOLLOW(X) ← FOLLOW(X) ∪ (FIRST(β) \ {ε})
6:         if ε ∈ FIRST(β) then
7:             FOLLOW(X) ← FOLLOW(X) ∪ FOLLOW(A)
8:     for A → αX do                                  ▷ 规则 3: X 是某产生式右部的最后一个符号
9:         FOLLOW(X) ← FOLLOW(X) ∪ FOLLOW(A)
```

不断应用上面的规则, 直到每个 $\textsc{Follow}(X)$ 都不再变化 (**闭包!!!**)

2 LL(1)

如果文法 G 的预测分析表是无冲突的, 则 G 是 LL(1) 文法

对于当前选择的非终结符, 仅根据输入中当前的词法单元即可确定需要使用哪条产生式

> 预测分析表指明了每个非终结符在面对不同的词法单元或文件结束符时, 该选择哪个产生式 (按编号进行索引) 或者报错

- L : 从左向右 (left-to-right) 扫描输入
- L : 构建最左 (leftmost) 推导
- 1 : 只需向前看一个输入符号便可确定使用哪条产生式

2.1 构建预测分析表

先计算FIRST、FOLLOW集合

按照以下规则, 在表格 $[A, t]$ 中填入生成式 $A \to \alpha$ (编号):

$$t \in \textsc{First}(\alpha) \tag{1}$$

$$\alpha \overset{*}{\Rightarrow} \epsilon \wedge t \in \textsc{Follow}(A) \tag{2}$$

因其 “唯一”, 必要变充分

注：或者关系，两者都要检查！

例子：

$$\begin{array}{l} (1)\ X \to Y \\ (2)\ X \to a \\ (3)\ Y \to \epsilon \\ (4)\ Y \to c \\ (5)\ Z \to d \\ (6)\ Z \to XYZ \end{array}$$

$$\begin{array}{l} \textsc{First}(X) = \{a, c, \epsilon\} \\ \textsc{First}(Y) = \{c, \epsilon\} \\ \textsc{First}(Z) = \{a, c, d\} \\ \textsc{First}(XYZ) = \{a, c, d\} \\ \textsc{Follow}(X) = \{a, c, d, \$\} \\ \textsc{Follow}(Y) = \{a, c, d, \$\} \\ \textsc{Follow}(Z) = \emptyset \end{array}$$

	a	c	d	$\$$
X	1, 2	1	1	1
Y	3	3, 4	3	3
Z	6	6	5, 6	

2.2 改造为LL(1)

2.2.1 提取左公因子

$S \rightarrow i\ E\ t\ S \mid i\ E\ t\ S\ e\ S \mid a$

$E \rightarrow b$

提取左公因子

$S \rightarrow i\ E\ t\ S\ S' \mid a$

$S' \rightarrow e\ S \mid \epsilon$

$E \rightarrow b$

2.2.2 消除左递归

- 直接左递归的消除方法：

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \dots A\alpha_m \mid \beta_1 \mid \beta_2 \mid \dots \beta_n$$

其中，β_i 都不以 A 开头

$$A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \cdots \mid \beta_n A'$$

$$A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \cdots \mid \alpha_m A' \mid \epsilon$$

例如：

E → E + T | T

消除后为：

E → T E'

E ' → + T E' | ϵ

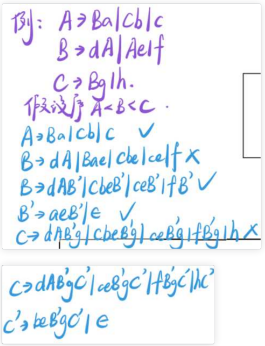
- 非直接左递归

思想：无环图

$$A_k \rightarrow A_l\alpha \implies l > k$$

1) 按照某个顺序将非终结符号排序为 $A_1, A_2, \dots, A_n$.
2) for（从1到 n 的每个 i）{
3) for（从1到 $i-1$ 的每个 j）{
4) 将每个形如 $A_i \rightarrow A_j\gamma$ 的产生式整换为产生式组 $A_i \rightarrow \delta_1\gamma \mid \delta_2\gamma \mid \cdots \mid \delta_k\gamma$，其中 $A_j \rightarrow \delta_1 \mid \delta_2 \mid \cdots \mid \delta_k$ 是所有的 A_j 产生式
5) }
6) 消除 A_i 产生式之间的立即左递归
7) }

例子：



2.3 LL(1)语法分析器伪代码

$$S \rightarrow F$$

$$S \rightarrow (S + F)$$

$$F \rightarrow a$$

	(	)	a	+	$
S	2		1		
F			3		

```
1: procedure S()
2:     if token = '(' then
3:         MATCH('(')
4:         S()
5:         MATCH('+')
6:         F()
7:         MATCH(')')
8:     else if token = 'a' then
9:         F()
10:    else
11:        ERROR(token, {'(', 'a'})
```

```
1: procedure MATCH(t)
2:     if token = t then
3:         token ← NEXT-TOKEN()
4:     else
```

```
1: procedure F()
2:     if token = 'a' then
```

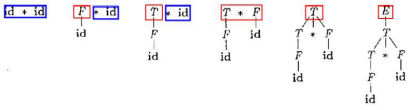
3 LR

3.1 什么是LR

L：从左向右 (Left-to-right) 扫描输入

R：构建反向 (Reverse) 最右推导

在任意时刻, 语法分析树的上边缘与剩余的输入构成当前句型



$$E \Longleftarrow T \Longleftarrow T*F \Longleftarrow T*\mathbf{id} \Longleftarrow F*\mathbf{id} \Longleftarrow \mathbf{id}*\mathbf{id}$$

LR 语法分析器使用栈存储语法分析树的上边缘

它包含了语法分析器目前所知的所有信息

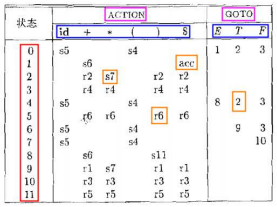
两大操作: 移入输入符号 与 按产生式归约

主要问题：

- 何时规约？
- 按哪条产生式规约？

3.2 LR分析表

LR 分析表指导 LR 语法分析器



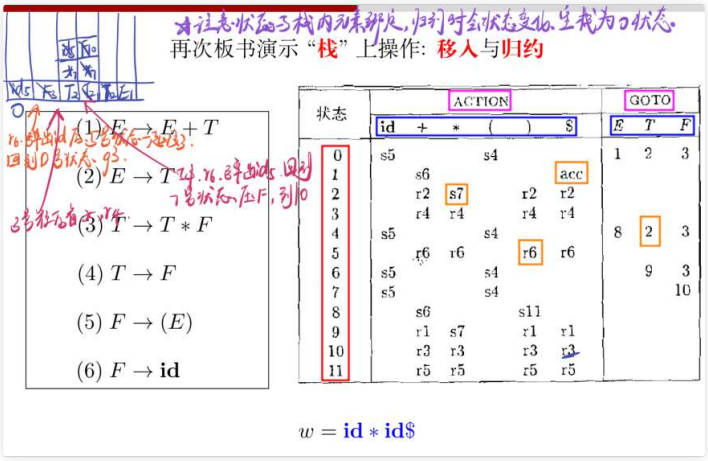
状态	ACTION						GOTO		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

在当前状态 (编号)下, 面对当前文法符号时, 该采取什么动作

ACTION 表指明动作, GOTO 表仅用于归约时的状态转换

sn	移入输入符号, 并进入状态 n
rk	使用 k 号产生式进行归约
gn	转换到状态 n
acc	成功接受, 结束
空白	错误

- 例子：栈上的移入与规约



- 问题：在当前状态 (编号)下, 面对当前文法符号时, 该采取什么动作

  思路：可以用自动机跟踪状态变化 (自动机中的路径 ⇔ 栈中符号/状态编号)

- 何时规约？

  必要条件: 当前状态中, 已观察到某个产生式的完整右部 ，也就是 句柄

> Definition (句柄 (Handle))
>
> 在输入串的 (唯一) 反向最右推导中, 如果下一步是逆用产生式 A → α 将 α 归约为 A, 则称 α 是当前句型的句柄。
>
> | 最右句型 | 句柄 | 归约用的产生式 |
> |---|---|---|
> | $\mathbf{id}_1 * \mathbf{id}_2$ | $\mathbf{id}_1$ | $F \to \mathbf{id}$ |
> | $F * \mathbf{id}_2$ | F | $T \to F$ |
> | $T * \mathbf{id}_2$ | $\mathbf{id}_2$ | $F \to \mathbf{id}$ |
> | $T * F$ | $T * F$ | $T \to T * F$ |
> | T | T | $E \to T$ |

- Theorem 存在一种 LR 语法分析方法, 保证句柄总是出现在栈顶

4 LR(0)

4.1 LR(0)文法

如果文法G的LR(0)分析表是无冲突的，则G是LR(0)文法

无冲突：ACTION表中每个单元格最多只有一种操作

- L : 从左向右 (Left-to-right) 扫描输入
- R : 构建反向 (Reverse) 最右推导
- 0 : 归约时无需向前看

4.2 LR(0)自动机

Definition (LR(0) 项 (Item))

文法 G 的一个 LR(0) 项是 G 的某个产生式加上一个位于体部的点

项指明了语法分析器已经观察到了某个产生式的某个前缀

$$A \to XYZ$$

$$[A \to \cdot XYZ]$$

$$[A \to X \cdot YZ]$$

$$[A \to XY \cdot Z]$$

$$[A \to XYZ \cdot]$$

(产生式 $A \to \epsilon$ 只有一个项 $[A \to \cdot]$)

Definition (项集)

项集就是若干项构成的集合

句柄识别自动机的一个状态可以表示为一个项集

Definition (项集族)

项集族就是若干项集构成的集合

句柄识别自动机的状态集可以表示为一个项集族

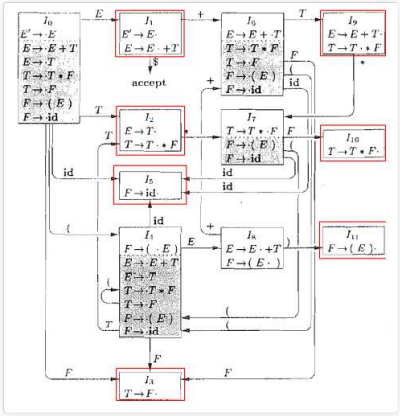
Definition (增广文法 (Augmented Grammar))

文法 G 的增广文法 G′ 是在 G 中加入产生式 S ′ → S 得到的文法

目的: 告诉语法分析器何时停止分析并接受输入符号串

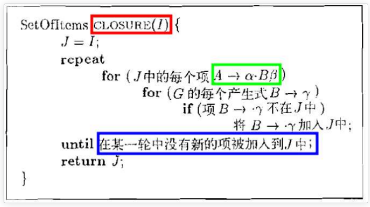
当语法分析器面对 $且要使用 S ′ → S 进行归约时, 输入符号串被接受

Example：LR(0) 句柄识别自动机（红圈表示接受状态）



4.3 构造LR(0)自动机

需要知道的：闭包的计算



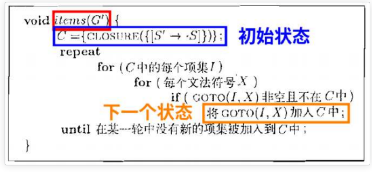
- init

$$\text{CLOSURE}(\{[E' \to \cdot E]\})$$

- 演化

$$J = \text{GOTO}(I, X) = \text{CLOSURE}\left(\left\{[A \to \alpha X \cdot \beta] \middle| [A \to \alpha \cdot X\beta] \in I\right\}\right)$$

$$(X \in N \cup T)$$



- 接受状态

$$F = \{I \in C \mid \exists k.\ [k : A \to \alpha \cdot] \in I\}$$

- accep状态

$$[S' \to S \cdot]$$

下，遇见$的转移

千万不要漏掉accept！

4.4 构造LR(0)分析表

1. 先构造出LR(0)自动机，每个自动机的状态对应LR(0)分析表中的一个状态
2. 根据以下规则，构造LR(0)分析表

$$(1)\ \text{GOTO}(I_i, a) = I_j \wedge a \in T \implies \text{ACTION}[i, a] \leftarrow sj$$

$$(2)\ \text{GOTO}(I_i, A) = I_j \wedge A \in N \implies \text{GOTO}[i, A] \leftarrow gj$$

$$(3)\ [k : A \to \alpha \cdot] \in I_i \wedge A \neq S' \implies \forall t \in T \cup \{\$\}.\ \text{ACTION}[i, t] = rk$$

$$(4)\ [S' \to S \cdot] \in I_i \implies \text{ACTION}[i, \$] \leftarrow acc$$

3. 如果文法 G 的LR(0) 分析表是无冲突的, 则 G 是 LR(0) 文法

5 SLR(1)

Simple LR(1)

5.1 LR(0)存在的问题

LR(0) 分析表每一行 (状态) 所选用的归约产生式是相同的

5.2 改进

对LR(0)的规约规则进行改进

归约:

(3) $[k: A \to \alpha \cdot] \in I_i \wedge A \neq S' \implies \forall t \in \text{FOLLOW}(A).\ \text{ACTION}[i,t] = rk$

6 LR(1)

6.1 LR(0)与SLR(1)存在的问题

- LR0

在 $LR(0)$ 自动机中, 某个项集 I_j 中包含 $[A \to \alpha \cdot]$

则在之前的某个项集 I_i 中包含 $[B \to \beta \cdot A\gamma]$ 与 $[A \to \cdot\alpha]$

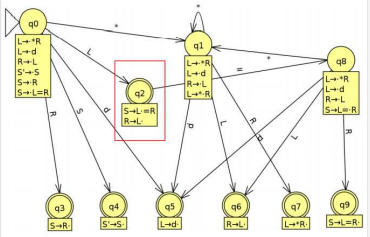
这表明只有 $a \in \text{FIRST}(\gamma)$ 时, 才可以进行 $A \to \alpha$ 归约

但是, 对 I_i 求闭包时, 仅得到 $[A \to \cdot\alpha]$, 丢失了 $\text{FIRST}(\gamma)$ 信息

- SLR1

即使考虑了 $= \in \text{FOLLOW}(A)$, 对该文法来说仍然不够

因为, 这仅仅说明在某个句型中, a 可以跟在 A 后面



该文法没有以 $R = \cdots$ 开头的最右句型

6.2 LR(1)项

Definition (LR(1) 项 (Item))

[A → α · β, a] (a ∈ T ∪ {$}) 此处, a 是向前看符号, 数量为 1

思想: α 在栈顶, 且剩余输入中开头的是可以从 βa 推导出的符号串

也就是说 , [A → α·, a]只有下一个输入符号为 a 时, 才可以按照 A → α 进行归约

6.3 LR(1)自动机

- LR1闭包计算

```
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;
}
```

$\forall b \in \text{FIRST}(\beta a).\ [B \to \cdot\gamma, b] \in I$

- LR1初始化

初始状态: $\text{CLOSURE}([S' \to \cdot S, \$])$

- LR1的GOTO计算

```
SetOfItems 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);
}
```

$J = \text{GOTO}(I, X) = \text{CLOSURE}\left(\left\{[A \to \alpha X \cdot \beta, a] \middle| [A \to \alpha \cdot X\beta, a] \in I\right\}\right)$

$(X \subset N \cup T)$

- LR1自动机构造

```
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;
}
```

初始状态: $\text{CLOSURE}([S' \to \cdot S, \$])$

6.4 LR(1)分析表

(1) $\text{GOTO}(I_i, a) = I_j \wedge a \in T \implies \text{ACTION}[i, a] \leftarrow sj$

(2) $\text{GOTO}(I_i, A) = I_j \wedge A \in T \implies \text{GOTO}[i, A] \leftarrow gj$

(3) $[k: A \to \alpha \cdot, a] \in I_i \wedge A \neq S' \implies \text{ACTION}[i, a] = rk$

(4) $[S' \to S \cdot, \$] \in I_i \implies \text{ACTION}[i, \$] \leftarrow acc$

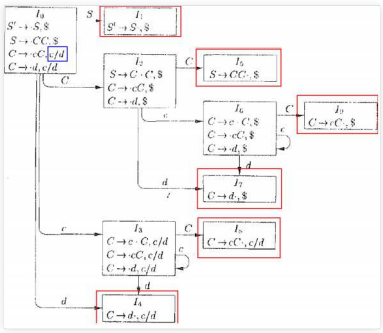
7 LALR(1)

7.1 LR(1)的问题

LR(1) 虽然强大, 但是生成的 LR(1) 分析表可能过大, 状态过多

LALR(1) : 合并具有相同核心 LR(0)项的状态 (忽略不同的向前看符号)

7.2 合并核心项



例如，合并图中的(4,7),(3,6),(8,9)

7.3 引入冲突？

对于 LR(1) 文法, 合并得到的 LALR(1) 分析表是否会引入冲突？

- 不会 引入 移入/归约冲突

  假设合并后出现冲突，[A → α·, a] 与 [B → β · aγ, b]

  则在 LR(1) 自动机中, 存在某状态同时包含 [A → α·, a] 与 [B → β · aγ, c] (c随便是什么)

- 可能会 引入 归约/归约 冲突

8 例题

8.1 LR0、SLR1

给定下述文法 G,

$$L \to LP \quad (1)$$

$$L \to P \quad (2)$$

$$P \to (P) \quad (3)$$

$$P \to () \quad (4)$$

(1) 简述 G 所对应的语言;

(2) 为 G 构造 $LR(0)$ 自动机;

注意: 为了尽量统一状态编号, 便于批改, 当计算 CLOSURE 时, 请按照文法编号大小顺序加入新项。当计算 GOTO(I, X) 时, 请按照 I 中项的出现顺序依次考虑可能的转移符号 X。

要求: 给出初始状态 I_0 的计算方法以及 GOTO$(I_0, ()$ 的计算方法。

(3) 为该文法设计 $LR(0)$ 分析表; 该文法是 $LR(0)$ 文法吗? 请说明理由。

(4) 为该文法设计 $SLR(1)$ 分析表; 该文法是 $SLR(1)$ 文法吗? 请说明理由。

要求: 请说明归约的设置条件。

(5) 如果该文法是 $SLR(1)$ 文法, 请给出识别输入串 (())() 时自动机所经历的状态 (编号)。

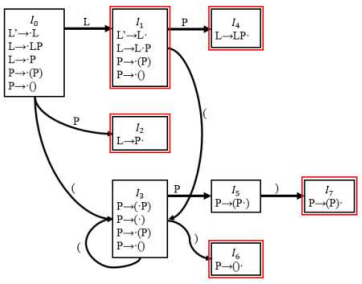
解答:

(1) 一组或多组成对括号，一组成对括号的形式为 (),(()),((...))，但不包括一对括号内包含多个成对括号的情况，如 (()())

(2) $I_0 =$ CLOSURE$(\{L' \to \cdot\, L\})$
$=$ CLOSURE$(\{L' \to \cdot\, L, L \to \cdot\, LP, L \to \cdot\, P\})$
$= \{L' \to \cdot\, L, L \to \cdot\, LP, L \to \cdot\, P, P \to \cdot\, (P), P \to \cdot\, ()\}$

GOTO$(I_0, () =$ CLOSURE$(\{P \to (\cdot\, P), P \to (\cdot\,)\})$
$= \{P \to (\cdot\, P), P \to (\cdot\,), P \to \cdot\, (P), P \to \cdot\, ()\}$

如图所示:



(3) 该文法是 $LR(0)$ 文法，因为 $LR(0)$ 分析表没有冲突

状态	ACTION			GOTO	
	(	)	$	L	P
0	s_3			g_1	g_2
1	s_3		acc		g_4
2	r_2	r_2	r_2		
3	s_3	s_6			g_5
4	r_1	r_1	r_1		
5		s_7			
6	r_4	r_4	r_4		
7	r_3	r_3	r_3		

(4) $FOLLOW(L) = \{(, \$\}$
$FOLLOW(P) = \{(,), \$\}$

状态	ACTION			GOTO	
	(	)	$	L	P
0	s_3			g_1	g_2
1	s_3		acc		g_4
2	r_2		r_2		
3	s_3	s_6			g_5
4	r_1		r_1		
5		s_7			
6	r_4	r_4	r_4		
7	r_3	r_3	r_3		

(5) 0 3 3 6 5 7 2 1 3 6 4 1
I_0: next-token (s_3 to I_3 stack:$(_3$
I_3: next-token (s_3 to I_3 stack:$(_3(_3$
I_3: next-token) s_6 to I_6 stack:$(_3(_3)_6$
I_6: next-token) r_4、g_5 to I_5 stack:$(_3P_5$
I_5: next-token) s_7 to I_7 stack:$(_3P_5)_7$
I_7: next-token (r_3、g_2 to I_2 stack:P_2
I_2: next-token (r_2、g_1 to I_1 stack:L_1
I_1: next-token (s_3 to I_3 stack:$L_1(_3$
I_3: next-token) s_6 to I_6 stack:$L_1(_3)_6$
I_6: next-token \$ r_4、g_4 to I_4 stack:L_1P_4
I_4: next-token \$ r_1、g_1 to I_1 stack:L_1
I_1: next-token \$ acc

8.2 LR1、LALR1

题目 1 ([10 = 1 + 4 + 2 + 3 分])
给定下述文法 G,

$$L \to LP \quad (1)$$

$$L \to P \quad (2)$$

$$P \to (P) \quad (3)$$

$$P \to () \quad (4)$$

(1) 为后面的小题计算必要的 First 集合与 Follow 集合 (可以直接转抄上次作业);

(2) 为 G 构造 $LR(1)$ 自动机;

注意: 为了尽量统一状态编号, 便于批改, 当计算 closure 时, 请按照文法编号大小顺序加入新项。当计算 $\text{goto}(I, X)$ 时, 请按照 I 中项的出现顺序依次考虑可能的转移符号 X。

要求: 给出初始状态 I_0 的计算方法以及 $\text{goto}(I_0, ()$ 的计算方法。

(3) 为该文法设计 $LR(1)$ 分析表; 该文法是 $LR(1)$ 文法吗? 请说明理由。

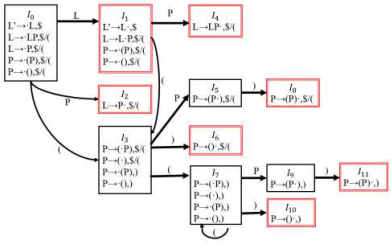
要求: 请说明归约的设置条件。

(4) 为该文法设计 $LALR(1)$ 分析表; 该文法是 $LALR(1)$ 文法吗? 请说明理由。

解答:
(1) $\text{First}(L) = \{(\}$
$\text{First}(P) = \{(\}$
$\text{Follow}(L) = \{(, \$\}$
$\text{Follow}(P) = \{(,), \$\}$

(2) 计算方法:
$I_0 = CLOSURE([L' \to \cdot\, L, \$])$
$= CLOSURE([L' \to \cdot\, L, \$], [L \to \cdot\, LP, \$], [L \to \cdot\, P, \$])$
$= CLOSURE([L' \to \cdot\, L, \$], [L \to \cdot\, LP, \$/(], [L \to \cdot\, P, \$/(])$
$= \{[L' \to \cdot\, L, \$], [L \to \cdot\, LP, \$/(], [L \to \cdot\, P, \$/(], [P \to \cdot\, (P), \$/(], [P \to \cdot\, (), \$/(]\}$

$GOTO(I_0, () = CLOSURE([P \to (\cdot\, P), \$/(], [P \to (\cdot\,), \$/(])$
$= \{[P \to (\cdot\, P), \$/(], [P \to (\cdot\,), \$/(], [P \to \cdot\, (P),)], [P \to \cdot\, (),)]\}$



(3) 如图所示, 是 $LR(1)$ 文法
规约条件: 在合并相同 $LR(0)$ 核心项后
$[k : A \to \alpha\, \cdot\, , a] \in I_i \wedge A \neq S' \Rightarrow ACTION[i, a] = rk$

状态	ACTION			GOTO	
	(	)	$	L	P
0	s_3			g_1	g_2
1	s_3		acc		g_4
2	r_2		r_2		
3	s_7	s_6			g_5
4	r_1		r_1		
5		s_8			
6	r_4		r_4		
7	s_7	s_{10}			g_9
8	r_3		r_3		
9		s_{11}			
10		r_4			
11		r_3			

(4) 合并 I_3 与 I_7、I_5 与 I_9、I_6 与 I_{10}、I_8 与 I_{11}
如图所示，是 $LALR(1)$ 文法

状态	ACTION			GOTO	
	(	)	$	L	P
0	s_{37}			g_1	g_2
1	s_{37}		acc		g_4
2	r_2		r_2		
37	s_{37}	s_{610}			g_{59}
4	r_1		r_1		
59		s_{811}			
610	r_4	r_4	r_4		
811	r_3	r_3	r_3		

分类: 基础知识

标签: 编译原理

1 0

« 上一篇： 编译原理：深入理解正则表达式与NFA、DFA状态机

» 下一篇： 编译原理：L属性、S属性语法制导

posted @ 2021-01-16 23:17 cpaulyz 阅读(5939) 评论(0) 编辑 收藏 举报