
编译原理

- 第一章 编译程序概述
 - 第二章 PL/0编译程序的实现
 - 第三章 文法和语言
 - 第四章 词法分析
 - 第五章 自顶向下语法分析方法
 - 第六章 自底向上优先分析方法
 - 第七章 LR分析方法
 - 第八章 语法制导翻译和中间代码生成
 - 第九章 符号表
 - 第一〇章 代码优化
 - 第一一章 代码生成
-

内容回顾

- 自上而下语法分析要解决的关键问题是什么？
 - ✓ 假定要被代换的最左非终结符号是 B ，且有 n 条规则： $B \rightarrow A_1 | A_2 | \dots | A_n$ ，那么如何确定用哪个右部去替代 B ？
 - 确定的自顶向下分析思想是什么？
 - ✓ 从开始符号出发，不断替换非终结符，根据当前输入的单词符号就可以唯一选定要替换的产生式。
-

内容回顾

- 怎样根据当前输入的单词符号选定要替换的产生式？
 - 根据当前输入符号落在哪个产生式规则的选择集合中，来唯一确定产生式进行推导。
- 确定的自顶向下分析方法中，产生式的选择集合是什么？
 - 对于产生式 $A \rightarrow \alpha$ ， $A \in V_N$ ， $\alpha \in V^*$ ，且 α 不能推导出 ϵ ，则 $\text{select}(A \rightarrow \alpha) = \text{First}(\alpha)$ ；
 - 对于产生式 $A \rightarrow \alpha$ ， $A \in V_N$ ， $\alpha \in V^*$ ，且 α 能推导出 ϵ ，则 $\text{select}(A \rightarrow \alpha) = (\text{First}(\alpha) - \{\epsilon\}) \cup \text{Follow}(A)$ ；

内容回顾

- 哪类文法产生的句子才能用确定的自顶向下方法进行分析？
 - LL(1)文法
- LL(1)的含义是什么？
 - 从左L向右扫描输入串，分析过程中采用最左L推导，只需向右看1个符号就可确定如何推导（选择哪个产生式进行推导）。
- LL(1)文法的定义是什么？
 - 对每个非终结符A的两个不同产生式：
 - $A \rightarrow \alpha, A \rightarrow \beta$ 满足 $\text{Select}(A \rightarrow \alpha) \cap \text{Select}(A \rightarrow \beta) = \emptyset$ ，其中 α, β 不同时推出 ϵ
- 怎样对LL(1)文法产生的句子用确定的自顶向下方法进行分析？
 - 给定输入串，根据当前输入符号落在哪个产生式规则的选择集合中，来唯一确定产生式进行推导。

二、LL(1)文法的判别

例：若文法G[S]为： $S \rightarrow AB \mid bC$

$A \rightarrow \varepsilon \mid b$

$B \rightarrow \varepsilon \mid aD$

$C \rightarrow AD \mid b$

$D \rightarrow aS \mid c$

判别文法是否是LL(1)文法。

解： (1) 求出能推出 ε 的非终结符

(2) 计算first集：

方法一： 计算first集合的算法：

对于每一个文法符号 $x \in V$ ， 计算first(x)的方法如下：

a) 若 $x \in V_T$, 则 $\text{first}(x) = \{x\}$

b) 若 $x \in V_N$, 且有 $x \rightarrow a\dots$, $a \in V_T$, 则 $a \in \text{first}(x)$

c) 若 $x \in V_N$, $x \rightarrow \varepsilon$, 则 $\varepsilon \in \text{first}(x)$

d) 若 $x \in V_N$, y_1, y_2, \dots, y_i 都 $\in V_N$, 产生式
 $x \rightarrow y_1 y_2 \dots y_n$, 当 y_1, y_2, \dots, y_{i-1} 都 $\xRightarrow{*} \varepsilon$ 时 ($1 \leq i \leq n$),

则 $\text{first}(y_1) - \{\varepsilon\}$, $\text{first}(y_2) - \{\varepsilon\}$, \dots , $\text{first}(y_{i-1}) - \{\varepsilon\}$,
 $\text{first}(y_i)$ 都包含在 $\text{first}(x)$ 中。

e) 当上式中所有 $y_i \xRightarrow{*} \varepsilon$ ($1 \leq i \leq n$),

则 $\text{first}(x) = \text{first}(y_1) \cup \text{first}(y_2) \cup \dots \cup \text{first}(y_n) \cup \{\varepsilon\}$

按上面的规则可得上例文法中每个文法符号的
first集合如下：

first(S)=

first(A)=

first(B)=

first(C)=

first(D)=

$S \rightarrow AB \mid bC$

$A \rightarrow \epsilon \mid b$

$B \rightarrow \epsilon \mid aD$

$C \rightarrow AD \mid b$

$D \rightarrow aS \mid c$

按上面的规则可得上例文法中每个文法符号的 first 集合如下：

$$\begin{aligned} \text{first}(S) &= \{ \text{first}(A) - \{ \varepsilon \} \} \cup \{ \text{first}(B) - \{ \varepsilon \} \} \\ &\cup \{ \varepsilon \} \cup \{ b \} = \{ a, b, \varepsilon \} \end{aligned}$$

$$\text{first}(A) = \{ b \} \cup \{ \varepsilon \} = \{ b, \varepsilon \}$$

$$\text{first}(B) = \{ \varepsilon \} \cup \{ a \} = \{ a, \varepsilon \}$$

$$\text{first}(C) = \{ \text{first}(A) - \{ \varepsilon \} \} \cup \text{first}(D) \cup \text{first}(b) = \{ a, b, c \}$$

$$\text{first}(D) = \{ a \} \cup \{ c \} = \{ a, c \}$$

$$S \rightarrow AB \mid bC$$

$$A \rightarrow \varepsilon \mid b$$

$$B \rightarrow \varepsilon \mid aD$$

$$C \rightarrow AD \mid b$$

$$D \rightarrow aS \mid c$$

一个文法符号串的first集合计算方法:

如果文法符号串 $\alpha \in V^*$, $\alpha = X_1 X_2 \dots X_n$,

1、当 $X_1 \xRightarrow{*} \varepsilon$, 则 $\text{first}(\alpha) = \text{first}(X_1)$

2、当对任何 j ($1 \leq j \leq i-1$, $2 \leq i \leq n$), $\varepsilon \in \text{first}(X_j)$

则 $\text{first}(\alpha) = (\text{first}(X_1) - \{\varepsilon\}) \cup (\text{first}(X_2) - \{\varepsilon\})$
 $\cup \dots \cup (\text{first}(X_{i-1}) - \{\varepsilon\}) \cup \text{first}(X_i)$

3、当 $\text{first}(X_j)$ 都含有 ε 时 ($1 \leq j \leq n$), 则

$\text{first}(\alpha) = \text{first}(X_1) \cup \text{first}(X_2) \cup \dots \cup \text{first}(X_j) \cup \{\varepsilon\}$

根据上面规则，每个产生式的右部符号串开始符号集合为：

$\text{first}(AB)=$

$\text{first}(bC)=$

$\text{first}(\varepsilon)=$

$\text{first}(aD)=$

$\text{first}(AD)=$

$\text{first}(b)=$

$\text{first}(aS)=$

$\text{first}(c)=$

$S \rightarrow AB \mid bC$

$A \rightarrow \varepsilon \mid b$

$B \rightarrow \varepsilon \mid aD$

$C \rightarrow AD \mid b$

$D \rightarrow aS \mid c$

根据上面规则，每个产生式的右部符号串开始符号集合为：

$$\text{first}(AB) = \text{first}(A) \cup \text{first}(B) \cup \{\varepsilon\} = \{a, b, \varepsilon\}$$

$$\text{first}(bC) = \{b\}$$

$$\text{first}(\varepsilon) = \{\varepsilon\}$$

$$\text{first}(aD) = \{a\}$$

$$\text{first}(AD) = (\text{first}(A) - \{\varepsilon\}) \cup \text{first}(D) = \{a, b, c\}$$

$$\text{first}(b) = \{b\}$$

$$\text{first}(aS) = \{a\}$$

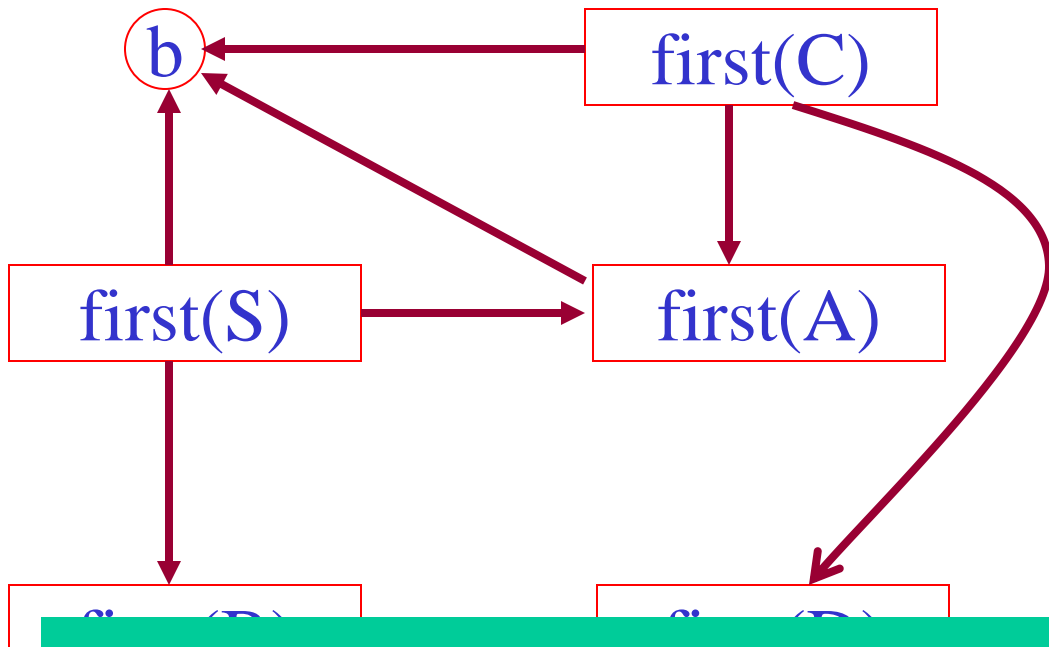
$$\text{first}(c) = \{c\}$$

根据规则3
因为 $A \Rightarrow \varepsilon$
 $B \Rightarrow \varepsilon$

根据规则2
因为 $A \Rightarrow \varepsilon$
 ~~$D \Rightarrow \varepsilon$~~

方法二：由关系图法求文法符号的first集合

- 1、每个文法符号对应图中的一个结点，终结符结点用符号本身标记，非终结符结点用 $\text{first}(A)$ 标记。
- 2、若文法中有 $A \rightarrow \alpha X \beta$ ， $\alpha \xrightarrow{*} \varepsilon$ ，则从对应A的结点到对应X结点连一条箭弧。
- 3、凡是从 $\text{first}(A)$ 结点有路径可到达的终结符结点所标记的终结符都为 $\text{first}(A)$ 的成员。
- 4、根据判别步骤1确定 ε 是否为某非终结符 first 的成员，若是则将 ε 加入该非终结符的 first 集中。



文法为:

$S \rightarrow AB \mid bC$

$A \rightarrow \varepsilon \mid b$

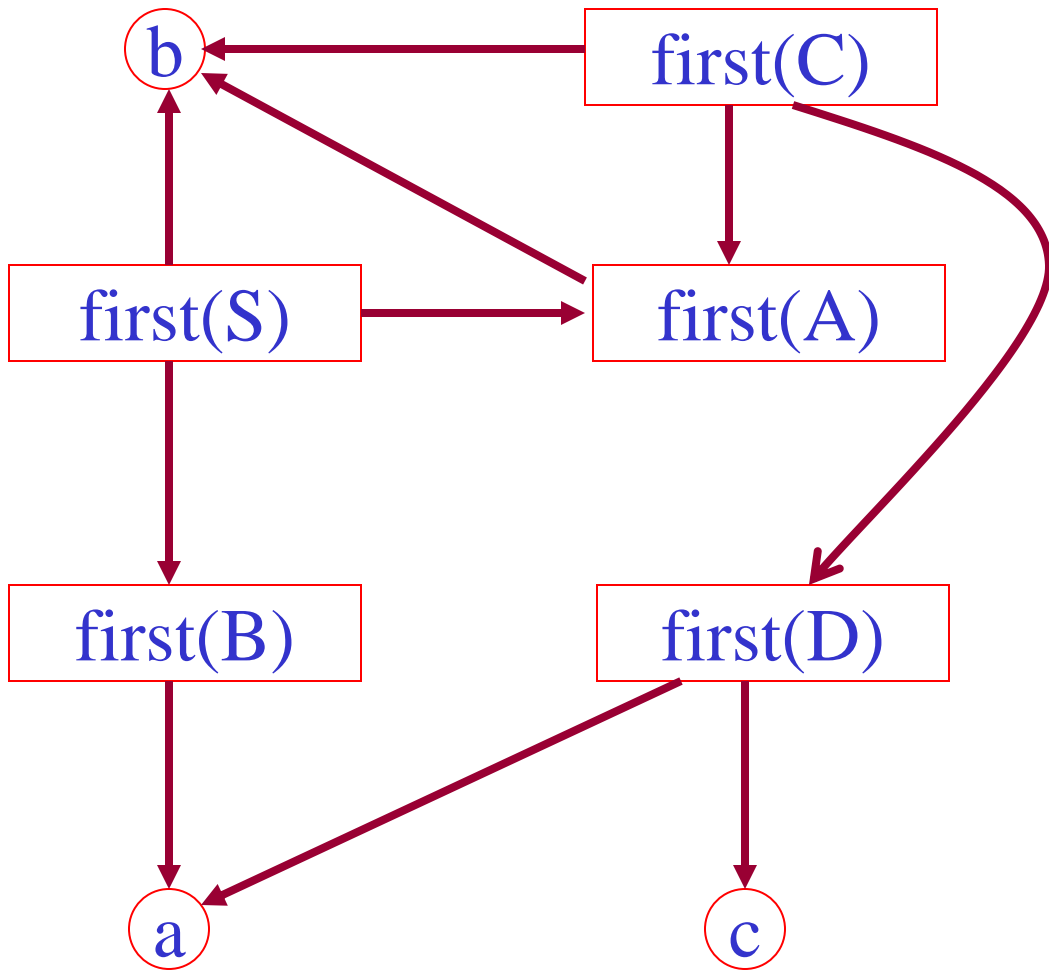
$B \rightarrow \varepsilon \mid aD$

$C \rightarrow AD \mid b$

$D \rightarrow aS \mid c$

S 同理: 从A到b, 从B到a, 从C到A、D、b, 从D到a、c画一条弧。

(2) 规则1: 非终 (3) 规则2: $A \rightarrow \alpha X \beta$, $\alpha \Rightarrow \varepsilon$, 标记。本文法中则A到X画一条箭弧。



文法为:

$S \rightarrow AB \mid bC$

$A \rightarrow \varepsilon \mid b$

$B \rightarrow \varepsilon \mid aD$

$C \rightarrow AD \mid b$

$D \rightarrow aS \mid c$

(4)规则3: $\text{first}(A)$ 结点有路径可到达的终结符结点所标记的终结符都为 $\text{first}(A)$ 的成员。

(5)规则4: 若 ϵ 是某非终结符 first 的成员, 则将 ϵ 加入该非终结符的 first 集中。

所以, $\text{first}(S) = \{a, b\}$

$\text{first}(A) = \{b\}$

$\text{first}(B) = \{a\}$

$\text{first}(C) = \{a, b, c\}$

$\text{first}(D) = \{a, c\}$

所以, $\text{first}(S) = \{a, b, \epsilon\}$

$\text{first}(A) = \{b, \epsilon\}$

$\text{first}(B) = \{a, \epsilon\}$

$\text{first}(C) = \{a, b, c\}$

$\text{first}(D) = \{a, c\}$

(3) 计算Follow集:

方法一: 根据Follow定义计算, 算法如下:

求follow集的算法:

(1) 设S为开始符号, 则 $\# \in \text{follow}(S)$;

(2) $\text{Follow}(A)$ 是所有句型中出现在紧接A之后的终结符或“#”。

(II) 构造集合FOLLOW的算法

设 $S, A, B \in V_n$,

算法：连续使用以下规则，直至FOLLOW集合不再扩大

- (1) 若 S 为开始符号,则把“#”加入FOLLOW(S)中
- (2) 若 $A \rightarrow \alpha B \beta$ ($\beta \neq \epsilon$),则把FIRST(β)- $\{\epsilon\}$ 加入FOLLOW(B)
- (3) 若 $A \rightarrow \alpha B$ 或 $A \rightarrow \alpha B \beta$, 且 $\beta \stackrel{*}{\Rightarrow} \epsilon$ 则把FOLLOW(A)加入FOLLOW(B)

Follow(S)=

Follow(A)=

Follow(B)=

Follow(C)=

Follow(D)=

文法为:

$S \rightarrow AB \mid bC$

$A \rightarrow \varepsilon \mid b$

$B \rightarrow \varepsilon \mid aD$

$C \rightarrow AD \mid b$

$D \rightarrow aS \mid c$

Follow(S)={#}

Follow(A)={a, #, c}

Follow(B)={#}

Follow(C)={#}

Follow(D)={#}

S为开始符号，
#加入
follow(S)中。

文法为：

$S \rightarrow AB \mid bC$

$A \rightarrow \varepsilon \mid b$

$B \rightarrow \varepsilon \mid aD$

$C \rightarrow AD \mid b$

$D \rightarrow aS \mid c$

S

$\Rightarrow AB$

$\Rightarrow B$

$\Rightarrow Bb$

$\Rightarrow A$

$\Rightarrow AaD$

$S \Rightarrow AB \Rightarrow A\varepsilon \Rightarrow A$

$\Rightarrow AB \Rightarrow AaD$

$\Rightarrow bC \Rightarrow bAD \Rightarrow bAc$

方法二：根据关系图法求非终结符Follow集。 P82

- ① 文法 G 中的每个符号和“#”对应图中的一个结点,对应终结符和“#”的结点用符号本身标记。对应非终结符的结点(如 $A \in V_N$)则用 FOLLOW(A)或 FIRST(A)标记。
- ② 从开始符号 S 的 FOLLOW(S)结点到“#”号的结点连一条箭弧。
- ③ 如果文法中有产生式 $A \rightarrow \alpha B \beta X$, 且 $\beta \xrightarrow{*} \epsilon$, 则从 FOLLOW(B)结点到 FIRST(X)结点连一条弧, 当 $X \in V_T$ 时, 则与 X 相连。
- ④ 如果文法中有产生式 $A \rightarrow \alpha B \beta$ 且 $\beta \xrightarrow{*} \epsilon$ 则从 FOLLOW(B)结点到 FOLLOW(A)结点连一条箭弧。
- ⑤ 对每一 FIRST(A)结点如果有产生式 $A \rightarrow \alpha X \beta$, 且 $\alpha \xrightarrow{*} \epsilon$, 则从 FIRST(A)到 FIRST(X)连一条箭弧。
- ⑥ 凡是从 FOLLOW(A)结点有路径可以到达的终结符或“#”号的结点,其所标记的终结符或“#”号即为 FOLLOW(A)的成员。

方法二：根据关系图法求非终结符Follow集。P82

文法为：

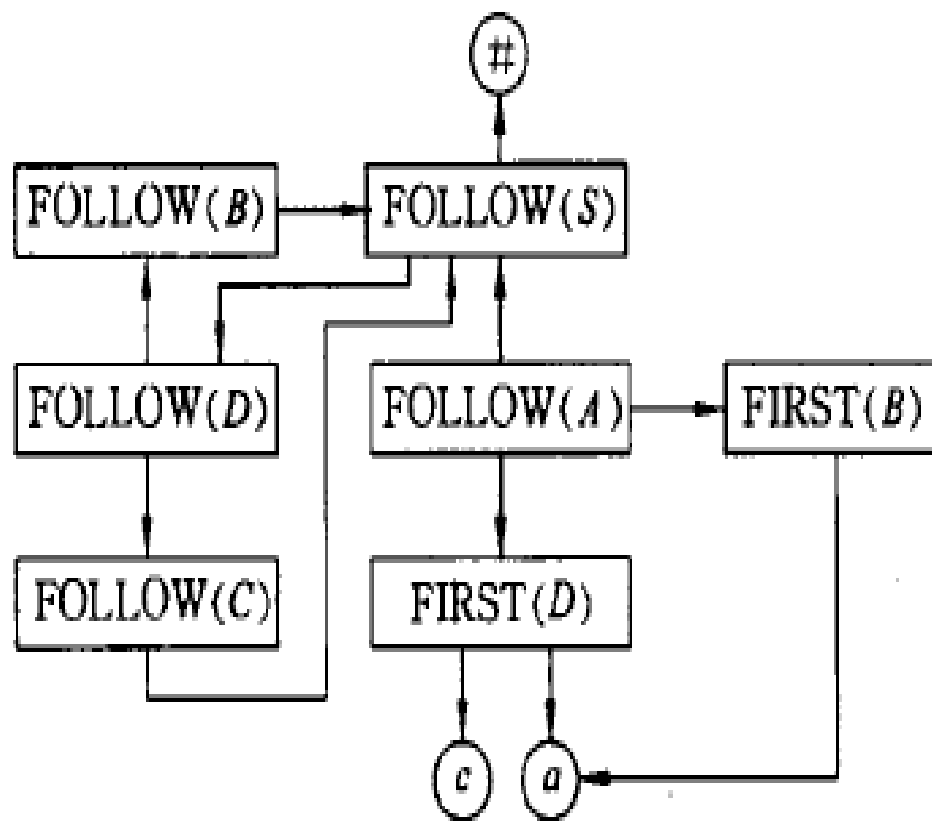
$S \rightarrow AB \mid bC$

$A \rightarrow \varepsilon \mid b$

$B \rightarrow \varepsilon \mid aD$

$C \rightarrow AD \mid b$

$D \rightarrow aS \mid c$



文法为:

$S \rightarrow AB \mid bC$

$A \rightarrow \varepsilon \mid b$

$B \rightarrow \varepsilon \mid aD$

$C \rightarrow AD \mid b$

$D \rightarrow aS \mid c$

(3) 计算Select集:

选择集合的定义

给定上下文无关文法的产生式 $A \rightarrow \alpha$, $A \in V_N$,

$\alpha \in V^*$, 若 $\alpha \Rightarrow \varepsilon$, 则 $\text{Select}(A \rightarrow \alpha) = \text{First}(\alpha)$,

若 $\alpha \Rightarrow \varepsilon$, 则 $\text{Select}(A \rightarrow \alpha) = (\text{First}(\alpha) - \{\varepsilon\}) \cup \text{Follow}(A)$

(3) 计算Select集:

因为 $A \Rightarrow \varepsilon$

$B \Rightarrow \varepsilon$

每个产生式的Select集合计算为

$$\text{Select}(S \rightarrow AB) = (\text{first}(AB) - \{\varepsilon\}) \cup \text{Follow}(S) = \{b, a, \#\}$$

$$\text{Select}(S \rightarrow bC) = \text{first}(bC) = \{b\}$$

$$\text{Select}(A \rightarrow \varepsilon) = (\text{first}(\varepsilon) - \{\varepsilon\}) \cup \text{Follow}(A) = \{c, a, \#\}$$

$$\text{Select}(A \rightarrow b) = \text{first}(b) = \{b\}$$

$$\text{Select}(B \rightarrow \varepsilon) = (\text{first}(\varepsilon) - \{\varepsilon\}) \cup \text{Follow}(B) = \{\#\}$$

$$\text{Select}(B \rightarrow aD) = \text{first}(aD) = \{a\}$$

$$\text{Select}(C \rightarrow AD) = \text{first}(AD) = \{b, a, c\}$$

$$\text{Select}(C \rightarrow b) = \text{first}(C) = \{b\}$$

$$\text{Select}(D \rightarrow aS) = \text{first}(aS) = \{a\}$$

$$\text{Select}(D \rightarrow c) = \text{first}(c) = \{c\}$$

所以select的交集为:

$$\text{Select}(S \rightarrow AB) \cap \text{Select}(S \rightarrow bC) = \{b\} \neq \phi$$

$$\text{Select}(A \rightarrow \varepsilon) \cap \text{Select}(A \rightarrow b) = \phi$$

$$\text{Select}(B \rightarrow \varepsilon) \cap \text{Select}(B \rightarrow aD) = \phi$$

$$\text{Select}(C \rightarrow AD) \cap \text{Select}(C \rightarrow b) = \{b\} \neq \phi$$

$$\text{Select}(D \rightarrow aS) \cap \text{Select}(D \rightarrow c) = \phi$$

因此该文法不是LL(1)文法。

编程序实现LL(1)文法的自动判别

- 对于每一个文法符号 $x \in V$ ，计算 $\text{first}(x)$
- 对每个产生式规则的右部的文法符号串计算 first 集合
- 对每个非终结符号 A ，计算 $\text{Follow}(A)$
- 对每个产生式规则 $A \rightarrow \alpha$ ，计算 $\text{Select}(A \rightarrow \alpha)$
- 如果对每个非终结符 A 的两个不同产生式：

$A \rightarrow \alpha, A \rightarrow \beta$

满足 $\text{Select}(A \rightarrow \alpha) \cap \text{Select}(A \rightarrow \beta) = \emptyset$ ，其中 α 、 β 不同时推出 ϵ ，则当前文法为LL(1)文法

三、某些非LL(1)文法到LL(1)文法的等价变换

若文法中含有直接或间接左递归，含有左公共因子，该文法肯定不是LL(1)文法，要进行变换。

1、提取左公共因子

若文法中含有形如 $A \rightarrow \alpha\beta | \alpha\gamma$ 的产生式，这导致了对相同左部的产生式其右部的First集相交，也就是 $\text{Select}(A \rightarrow \alpha\beta) \cap \text{Select}(A \rightarrow \alpha\gamma) \neq \phi$ ，不满足LL(1)文法的充分必要条件。

则须进行提取左公共因子的等价变换： $A \rightarrow \alpha(\beta | \gamma)$

写成一般形式： $A \rightarrow \alpha A'$

$A' \rightarrow \beta | \gamma$

例1：若文法G1的产生式为：

$S \rightarrow aSb$

$S \rightarrow aS$

$S \rightarrow \varepsilon$

提取左公因子后得：

例1：若文法G1的产生式为：

$S \rightarrow aSb$

$S \rightarrow aS$

$S \rightarrow \varepsilon$

提取左公因子后得： $S \rightarrow aS(b | \varepsilon)$

$S \rightarrow \varepsilon$

进一步变换： $S \rightarrow aSA$

$A \rightarrow b$

$A \rightarrow \varepsilon$

$S \rightarrow \varepsilon$

是否LL(1)文法？

例2: 若文法G2的产生式:

$$A \rightarrow ad$$
$$A \rightarrow Bc$$
$$B \rightarrow aA$$
$$B \rightarrow bB$$

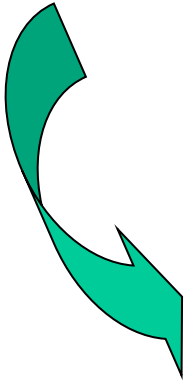
是否存在左公因子?

怎样消除左公因子?

例2: 若文法G2的产生式:

$$A \rightarrow ad$$
$$A \rightarrow Bc$$
$$B \rightarrow aA$$
$$B \rightarrow bB$$

解:

$$\left. \begin{array}{l} A \rightarrow ad \\ A \rightarrow aAc \\ A \rightarrow bBc \\ B \rightarrow aA \\ B \rightarrow bB \end{array} \right\} A \rightarrow a(d|Ac)$$

$$\left\{ \begin{array}{l} A \rightarrow aA' \\ A' \rightarrow d \\ A' \rightarrow Ac \end{array} \right.$$

最后变换为:

$$A \rightarrow aA'$$

$$A \rightarrow bBc$$

$$A' \rightarrow d$$

$$A' \rightarrow Ac$$

$$B \rightarrow aA$$

$$B \rightarrow bB$$

是否LL(1)文法？

经过文法提取左公共因子后的文法**不一定**是LL(1)文法。

经过文法提取左公共因子后的文法，若有多余的产生式，则必须进行化简。

例3：若有文法G3的产生式： $S \rightarrow aSd$

$$S \rightarrow Ac$$
$$A \rightarrow aS$$
$$A \rightarrow b$$

解： 文法替换： $S \rightarrow aSd$

$S \rightarrow aSc$

$S \rightarrow bc$

$A \rightarrow aS$

$A \rightarrow b$

$S \rightarrow aS(d|c)$

$S \rightarrow aSA'$

$A' \rightarrow d|c$

$S \rightarrow bc$

$A \rightarrow aS$

$A \rightarrow b$

文法中非终结符A变成不可到达的符号,产生式也就变为多余的,所以应删除.

$S \rightarrow aSA'$

$A' \rightarrow d|c$

$S \rightarrow bc$

此外也存在某些文法不能在有限步骤内提取左公共因子。

例4: 若有文法G4的产生式:

$$S \rightarrow Ap|Bq$$

$$A \rightarrow aAp|d$$

$$B \rightarrow aBq|e$$

$$S \rightarrow Ap|Bq$$

$$A \rightarrow aAp|d$$

$$B \rightarrow aBq|e$$

$$S \rightarrow aApp|aBqq$$

$$S \rightarrow dp|eq$$

$$A \rightarrow aAp|d$$

$$B \rightarrow aBq|e$$

$$S \rightarrow a(App|Bqq)$$

$S \rightarrow aS'$

$S' \rightarrow App|Bqq$

$S \rightarrow dp|eq$

$A \rightarrow aAp|d$

$B \rightarrow aBq|e$

$S \rightarrow aS'$

$S \rightarrow dp|eq$

$S' \rightarrow aApppp|aBqqq$

$S' \rightarrow dpp|eqq$

$A \rightarrow aAp|d$

$B \rightarrow aBq|e$

$$S \rightarrow aS'$$
$$S \rightarrow dp|eq$$
$$S' \rightarrow aS''$$
$$S' \rightarrow dpp|eqq$$
$$S'' \rightarrow Appp|Bqqq$$
$$A \rightarrow aAp|d$$
$$B \rightarrow aBq|e$$

可以看出产生式A.B继续替换，只能使文法的产生式愈来愈多无限增加下去，变成循环递归，不能得到提取左公共因子的预期结果。

上面例子说明：

- 不一定每个文法的左公共因子都能在有限的步骤内替换成无左公共因子的文法。
- 一个文法提取了左公共因子后，只解决了相同左部产生式右部的FIRST集不相交问题，当改写后的文法不含空产生式，且无左递归时，则改写后的文法是LL(1)文法。否则还需用LL(1)文法的判别方式进行判断才能确定是否为LL(1)文法。

2、消除左递归

1. 一个文法含有下列形式的产生式时，

$$\text{a) } A \rightarrow A\beta \quad A \in V_N, \beta \in V^*$$

$$\text{b) } A \rightarrow B\beta$$

$$B \rightarrow A\alpha \quad A, B \in V_N, \alpha, \beta \in V^*$$

称为左递归文法。其中a)是直接递归，b)是间接递归。

如果一个文法是左递归时，则不能采用自顶向下分析法。

例1：文法 $S \rightarrow Sa$

输入串：baaa#

$S \rightarrow b$ 是直接左递归

所产生的语言是： $L = \{ ba^n \mid n \geq 0 \}$

例2: 文法为: $A \rightarrow aB$

$A \rightarrow Bb$

$B \rightarrow Ac$

$B \rightarrow d$

输入串: adbcbcbc#

是间接左递归

※ 消除左递归的变换方法:

(1) 消除直接左递归:

方法: 改为右递归。如

$\left\{ \begin{array}{l} S \rightarrow Sa \\ S \rightarrow b \end{array} \right.$ 改写为: $\left\{ \begin{array}{l} S \rightarrow b S' \\ S' \rightarrow a S' | \varepsilon \end{array} \right.$

一般情况下,假定关于 A 的全部产生式是:

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

其中, $\alpha_i (1 \leq i \leq m)$ 不等于 ϵ , $\beta_j (1 \leq j \leq n)$ 不以 A 开头

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

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



(2) 消除间接左递归:

方法: 间接左递归 \Rightarrow 直接左递归 \Rightarrow 右递归

例3: 文法G6为:

$A \rightarrow aB$

$A \rightarrow Bb$

$B \rightarrow Ac$

$B \rightarrow d$



$A \rightarrow aB$

$A \rightarrow Bb$

$B \rightarrow aBc$

$B \rightarrow Bbc$

$B \rightarrow d$

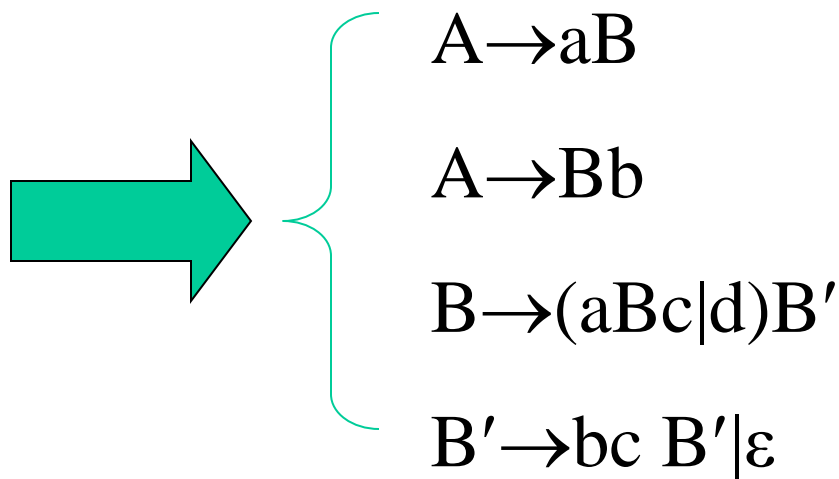
$B \rightarrow aBc|d$

$B \rightarrow Bbc$



$B \rightarrow (aBc|d)B'$

$B' \rightarrow bcB'|\epsilon$



对语法中一切左递归的消除要求语法中不含回路，即无 $A \xrightarrow{+} A$ 的推导。

满足该文法的充分条件是：语法中不包含形如 $A \rightarrow A$ 的有害规则和 $A \rightarrow \epsilon$ 的产生式。

第5章 自顶向下语法分析方法

- 一、确定的自顶向下分析思想
 - 二、LL(1)文法的判别
 - 三、某些非LL(1)文法到LL(1)文法等价变换
 - 四、不确定的自顶向下分析思想
 - 五、确定的自顶向下分析方法
-

四、不确定的自顶向下分析思想

假定要被代换的最左非终结符号是 B ，且有 n 条规则： $B \rightarrow A_1 | A_2 | \dots | A_n$ ，那么如何确定用哪个右部去替代 B ？

- LL(1)文法：根据规则的选择集合来确定——确定的自顶向下分析法：。
 - 非LL(1)文法：带回溯的自顶向下分析法——不确定的自顶向下分析法
 - “不确定”的意思：当某个非终结符的产生式有多个候选，而面临当前的输入符号无法确定选用哪个产生式，从而引起回溯。
-

下面讲三个例子说明回溯：

1、由于相同左部的产生式的右部First集交集不为空而引起回溯。

例：文法 $S \rightarrow xAy$

$A \rightarrow ab|a$

求输入串为 xay 语法树。

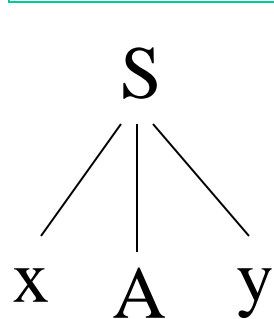


例：文法 $S \rightarrow xAy$

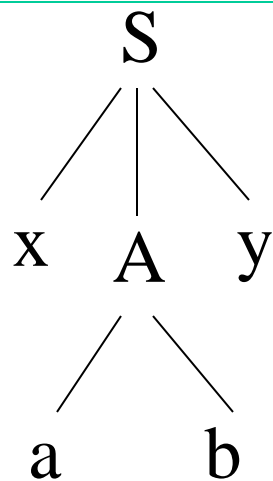
$A \rightarrow ab|a$

求输入串为 xay 语法树。

$\text{first}(ab)$ 与 $\text{first}(a)$
的交集不为空

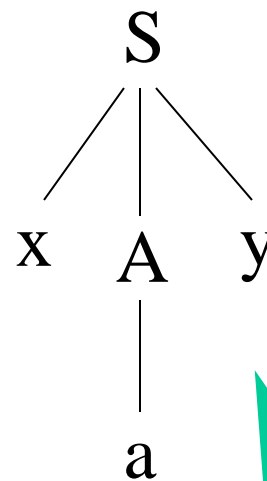
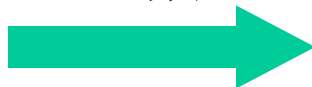


$S \rightarrow xAy$



若选择 $A \rightarrow ab$,
就于 xay 不匹配

回溯



回溯，用 $A \rightarrow a$
就匹配

2、由于相同左部非终结符的右部能 $\xRightarrow{*}\epsilon$ ，且该非终结符Follow集中含有其右部First集的元素而引起回溯。

例： $S \rightarrow aAS \mid b$

$A \rightarrow bAS \mid \epsilon$

求对输入串ab#的推导树。

2、由于相同左部非终结符的右部能 $\Rightarrow^* \epsilon$ ，且该非终结符Follow集中含有其右部First集的元素而引起回溯。

例： $S \rightarrow aAS \mid b$

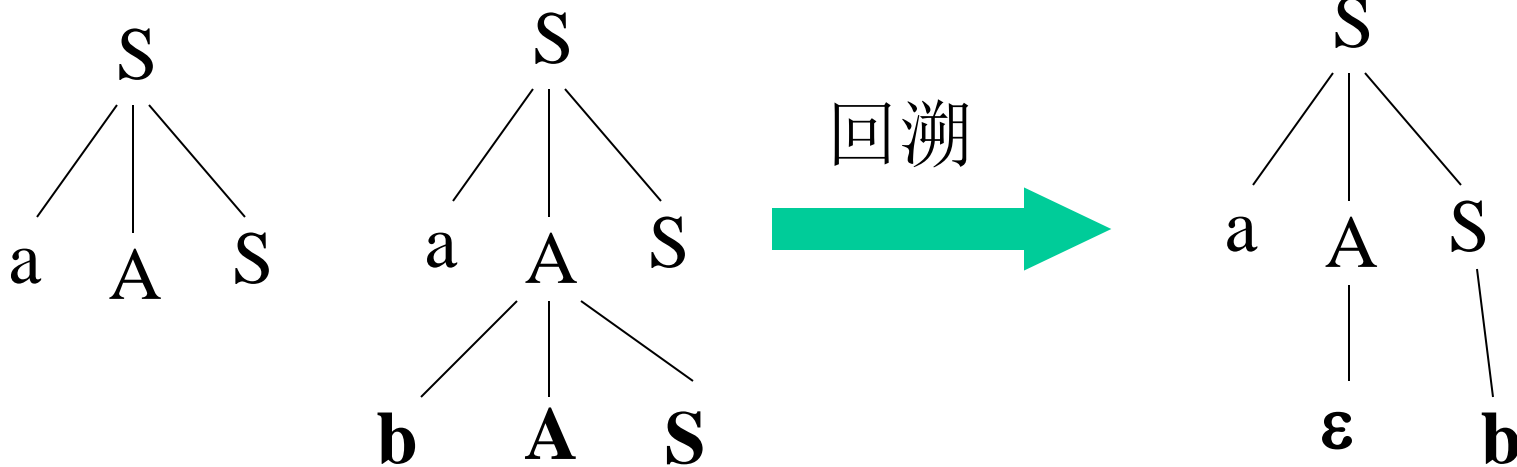
$A \rightarrow bAS \mid \epsilon$

求对输入串ab#的推导树。

$A \Rightarrow \epsilon$

$\text{Follow}(A) = \{a, b\}$

$\text{first}(bAs) = \{b\}$



3、由于语法含有左递归而引起回溯。

例： $S \rightarrow Sa$

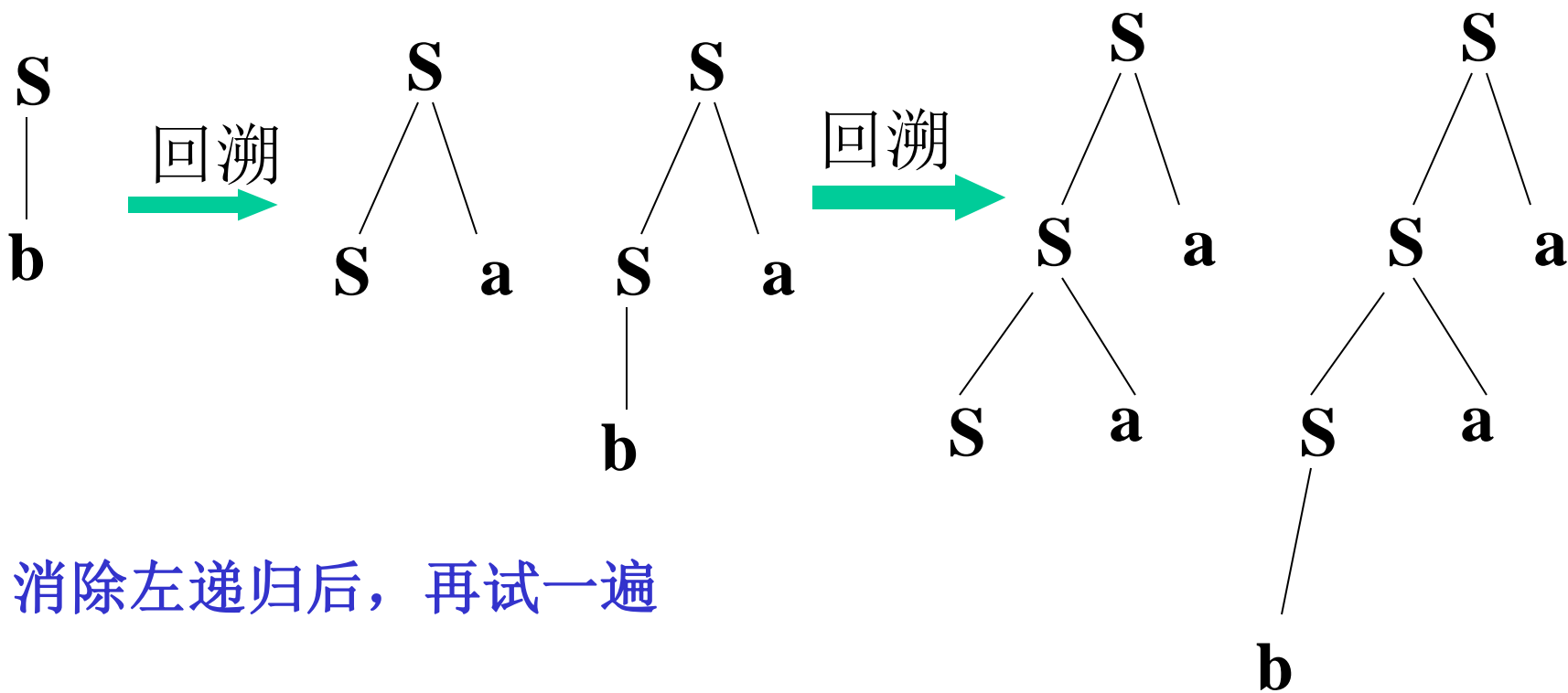
$S \rightarrow b$ 若推导 $baa\#$



3、由于语法含有左递归而引起回溯。

例： $S \rightarrow Sa$

$S \rightarrow b$ 若推导 $baa\#$



3、由于文法含有左递归而引起回溯。

例： $S \rightarrow bS'$

$S' \rightarrow aS'$

$S' \rightarrow \varepsilon$ 若推导 $baa\#$



由以上例子可以看出：带回溯的自顶向下分析是一个试探过程，当分析不成功时则推翻分析，退回到适当位置再重新试探其余可能的推导。

因此，需要记录所选过的产生式，直到把所有可能的推导序列都试完仍不成功，才能确认输入串不是该文法的句子。

（为证明输入串不合法，需要穷举或遍例所有的推导）

编译程序真正实现时往往边分析边插入语义动作，因而带回溯分析代价很高，效率很低，在实用编译程序中几乎不用。

典型例题

例题 2 文法 $G[S]: S \rightarrow Aa | b$

$$A \rightarrow SB$$
$$B \rightarrow ab$$

1. 试对 $G[S]$ 进行改写, 并判断改写后的文法是否为 LL(1) 文法?
2. 对于一个文法若消除了左递归, 提取了左公共因子后是否一定为 LL(1) 文法?

例题 3 判断文法 $G[S]: S \rightarrow Ab | Ba$

$$A \rightarrow aA | a$$
$$B \rightarrow a$$

是 LL(1) 的吗? 若不是, 请改写为等价的 $G'[S]$, 证明改写后的文法是否为 LL(1) 的。

课后作业

- 上次作业：P100 第2题 (1) 、 (2)
- P101 第7题 (1) (2) (3)

