



编译原理

第四章 语法分析——自上而下分析

丁志军

dingzj@tongji.edu.cn

内容线索

1. 语法分析器的功能

2. 自上而下分析方法概述

3. LL (1) 分析方法

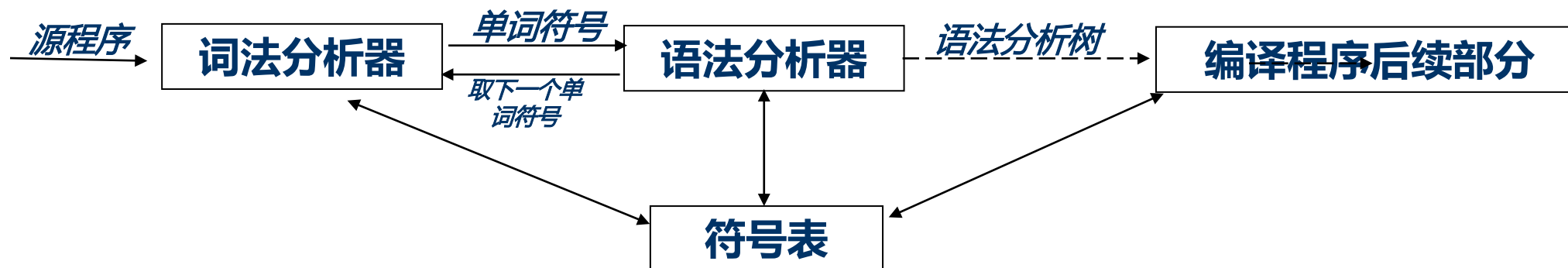
4. 递归下降分析程序

5. 预测分析程序

■ 语法分析的任务：

对任一给定 $w \in V_T^*$ ，判断 $w \in L(G)$?

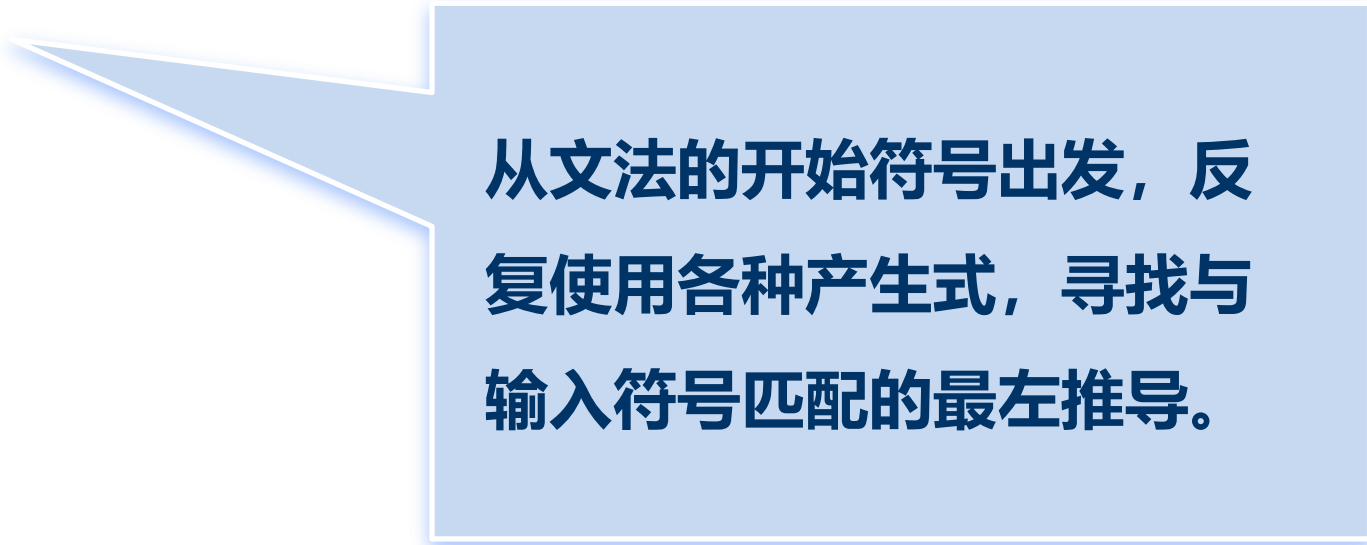
■ 语法分析器：按照产生式规则，做识别w的工作



语法分析器在编译程序中的地位

■ 自上而下分析

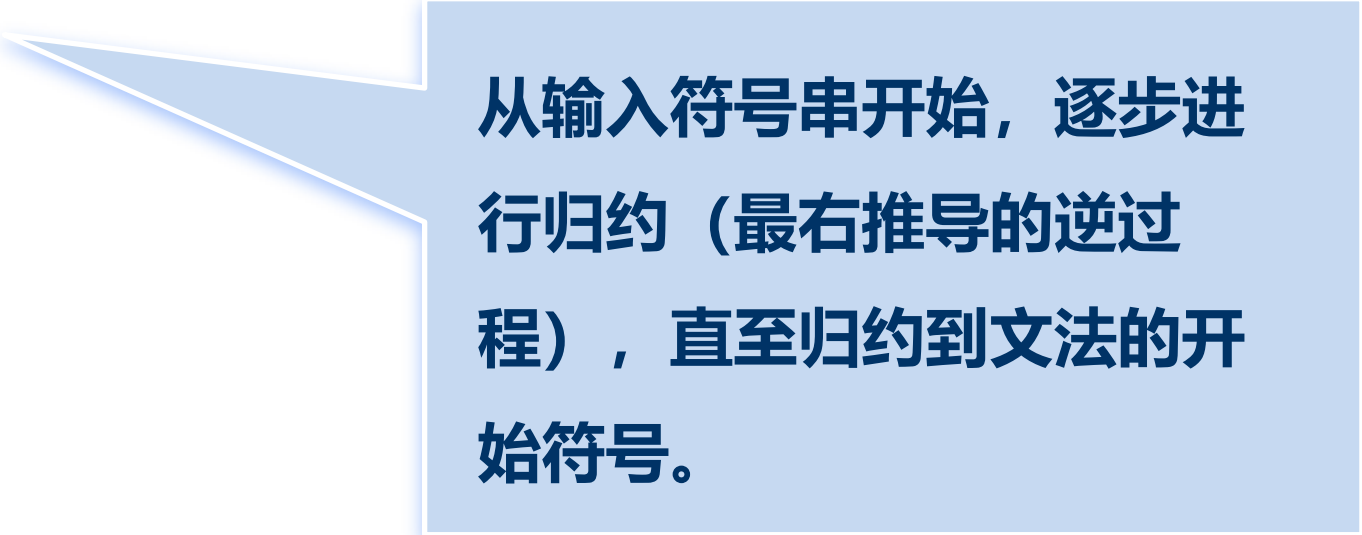
- 构建语法树
- 下推自动机
- LL (K) 分析法



从文法的开始符号出发，反复使用各种产生式，寻找与输入符号匹配的最左推导。

■ 自下而上分析

- 移进-规约方法
- CYK算法
- early算法
- 算符优先分析法
- LR分析法



从输入符号串开始，逐步进行归约（最右推导的逆过程），直至归约到文法的开始符号。

内容线索

√. 语法分析器的功能

2. 自上而下分析方法概述

3. LL (1) 分析方法

4. 递归下降分析程序

5. 预测分析程序

自上而下分析

- 从文法的开始符号出发，向下推导，推出句子
- 对任何的输入串(单词符号)，试图用一切可能的办法, 从文法的开始符号出发，自上而下地为输入串建立一棵语法树，即为输入串寻找一个最左推导。

语法分析示例

$G[S]: S \rightarrow id:=E \quad E \rightarrow E+E \quad E \rightarrow E * E$
 $E \rightarrow -E \quad E \rightarrow (E) \quad E \rightarrow id$

$id:=-id$ 是否为 $G[S]$ 的句子?

$S \Rightarrow id:=E \Rightarrow id:= -E \Rightarrow id:= -id$

基本思想

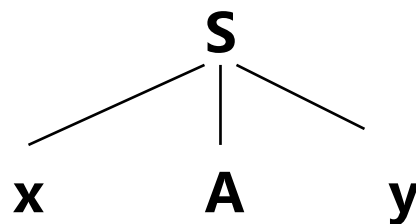
最左推导过程中，为当前字符选择一个产生式，根据产生式右部首字符进行匹配

示例

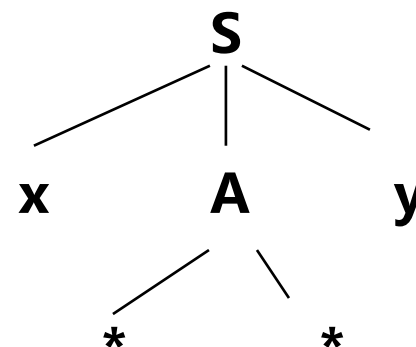
设文法 $G[S]$: $S \rightarrow xAy$, $A \rightarrow ** \mid *$, 判定输入串 $x * y$ 是否为它的句子?

$x * y$
↑

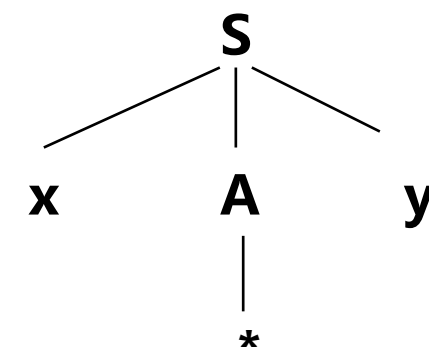
用 $S \rightarrow xAy$



用 $A \rightarrow **$
(回溯)



用 $A \rightarrow *$
(成功)



推导过程:

$S \Rightarrow xAy$

$\Rightarrow x**y$ (回溯)

$\Rightarrow x*y$ (成功)

语法分析示例

$G[S]: S \rightarrow id := E \quad E \rightarrow E + E \quad E \rightarrow E * E$

$E \rightarrow -E \quad E \rightarrow (E) \quad E \rightarrow id$

$id := -id + id + id$ 是否为 $G[S]$ 的句子?

$S \Rightarrow id := E \Rightarrow id := -E \Rightarrow id := -id$ (错误)

$S \Rightarrow id := E \Rightarrow id := -E \Rightarrow id := -E + E$

$\Rightarrow id := -E + E + E \Rightarrow id := -E + E + E + E \Rightarrow \dots$

带回溯自上而下分析面临的问题

- **虚假匹配问题**

- **回溯**

- 回溯会引起时间和空间的大量消耗

- **文法的左递归问题**

- 一个文法是含有左递归的，如果存在非终结符P

$$P \xRightarrow{+} P\alpha$$

- 含有左递归的文法将使自上而下的分析过程陷入无限循环

- **报告分析不成功时，难于知道输入串中出错的确切位置。**

- **实际上采用了一种穷尽一切可能的试探法，因此效率很低，代价很高**

内容线索

√. 语法分析器的功能

√. 自上而下分析方法概述

3. LL (1) 分析方法

4. 递归下降分析程序

5. 预测分析程序

LL(1)分析法

- 从左(**L**eft)到右扫描输入串；构造最左(**L**eftmost)推导；分析时每步向前看一个(**1**)字符。
- 目的：构造不带回溯的自上而下分析算法
 - 左递归的消除
 - 消除回溯，提左因子
 - FIRST集合，FOLLOW集合
 - LL(1)分析条件
 - LL(1)分析方法

左递归文法

- 一个文法含有下列形式的产生式时,

a) **直接递归**

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

b) **间接递归**

$$A \rightarrow B\beta$$

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

称为**左递归文法**。

- 如果一个文法是左递归时, 则不能采用自顶向下分析法。

示例1

文法 $S \rightarrow Sa$

$S \rightarrow b$

是直接左递归

语言是: $L = \{ ba^n \mid n \geq 0 \}$

示例2

文法 $P_1 \rightarrow aP_2$

$P_1 \rightarrow P_2b$

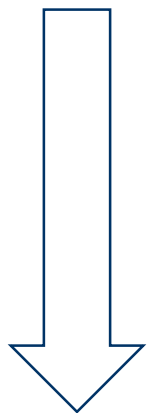
$P_2 \rightarrow P_1c$

$P_2 \rightarrow d$

是间接左递归

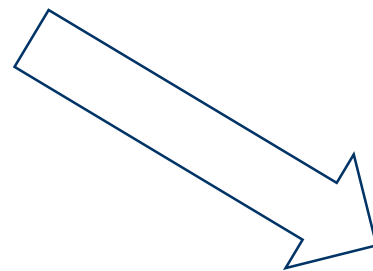
消除直接左递归

$P \rightarrow P\alpha \mid \beta$ ($\alpha \neq \epsilon$, β 不以 P 开头)



$P \rightarrow \beta P'$

$P' \rightarrow \alpha P' \mid \epsilon$



β

$\beta\alpha$

$\beta\alpha\alpha$

$\beta\alpha\alpha\alpha$

... ..

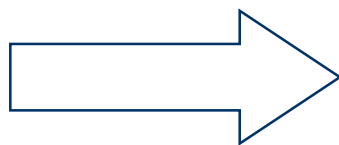


示例

文法 $E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid i$



$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \varepsilon$

$F \rightarrow (E) \mid i$

- 一般地, 假定关于P的产生式是

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

其中: $\alpha_i \neq \varepsilon$, β_i 不以P开头,

则改写为: $P \rightarrow \beta_1 P' \mid \beta_2 P' \mid \dots \mid \beta_n P'$

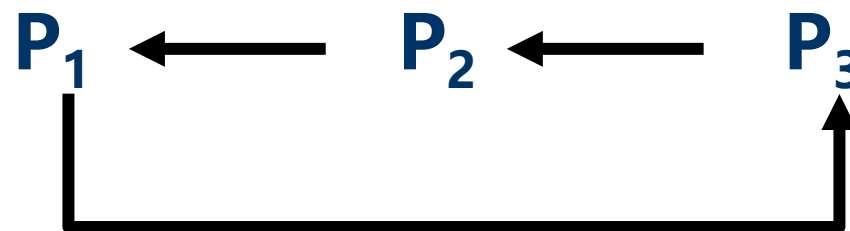
$$P' \rightarrow \alpha_1 P' \mid \alpha_2 P' \mid \dots \mid \alpha_m P' \mid \varepsilon$$

消除左递归的算法思想

文法 $P_2 \rightarrow P_1 b$

$P_3 \rightarrow P_2 c$

$P_1 \rightarrow P_3 d$



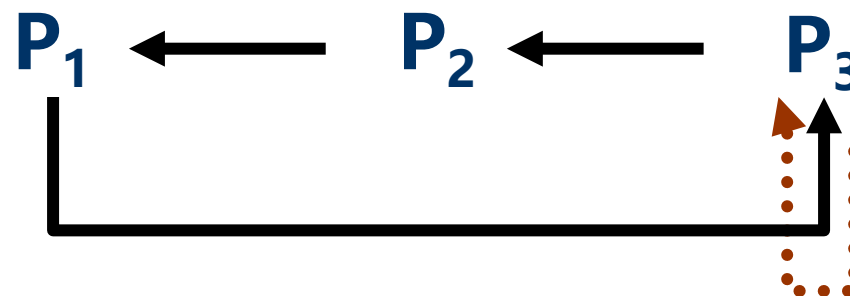
文法 $P_1 \rightarrow P_3 d$

$P_2 \rightarrow P_1 b$

$P_3 \rightarrow P_2 c$

$\Rightarrow P_1 bc$

$\Rightarrow P_3 dbc$



消除左递归算法

(1) 排序: P_1, P_2, \dots, P_n

(2) FOR $i := 1$ TO n DO

BEGIN

FOR $j := 1$ TO $i - 1$ DO

把形如 $P_i \rightarrow P_j \gamma$ 的规则改写为:

$P_i \rightarrow \delta_1 \gamma \mid \delta_2 \gamma \mid \dots \mid \delta_k \gamma$

其中: $P_j \rightarrow \delta_1 \mid \delta_2 \mid \dots \mid \delta_k$ 是关于 P_j
的所有规则;

消除关于 P_i 规则的直接左递归。

END

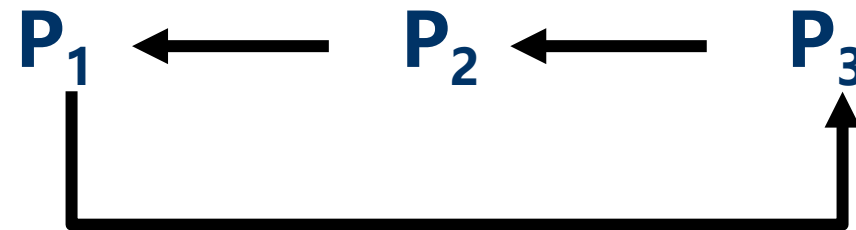
(3) 化简: 删除永不使用的产生式

算法示意

文法 $P_2 \rightarrow P_1 b$

$P_3 \rightarrow P_2 c$

$P_1 \rightarrow P_3 d$



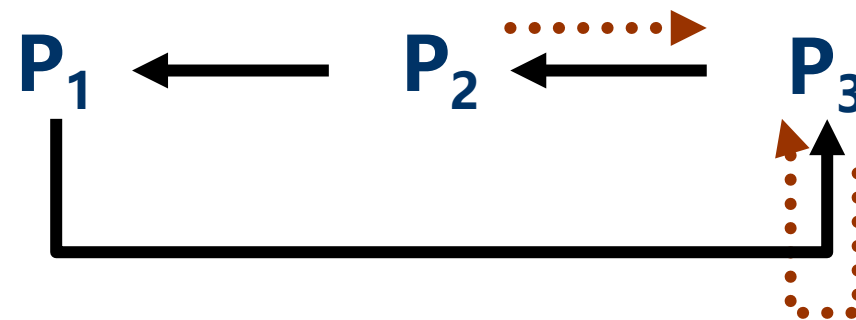
文法 $P_1 \rightarrow P_3 d$

$P_2 \rightarrow P_1 b$

$\Rightarrow P_3 db$

$P_3 \rightarrow P_2 c$

$\Rightarrow P_3 dbc$



示例

文法 $G[P_3]$: $P_3 \rightarrow P_2c|c$ $P_2 \rightarrow P_1b|b$ $P_1 \rightarrow P_3a|a$

有推导: $P_3 \Rightarrow P_2c \Rightarrow P_1bc \Rightarrow P_3abc$, 存在左递归。

按 P_1 (1)、 P_2 (2)、 P_3 (3)排序, 执行算法得:

$i=1$, j 从1至0, P_1 的产生式 $P_1 \rightarrow P_3a|a$ 无直接左递归, 无需消除直接左递归。

$i=2$, j 从1至1, P_2 的候选式含 P_1 , P_1 的产生式代入 P_2 的产生式得: $P_2 \rightarrow P_3ab|ab|b$, 无直接左递归。

$i=3$, j 从1至2:

$j=1$, P_3 的候选式不含 P_1 , 所以无需替换;

$j=2$, P_3 的候选式含 P_2 , 将 $P_2 \rightarrow P_3ab|ab|b$ 代入 S 的候选式得:

$$P_3 \rightarrow P_3abc|abc|bc|c$$

再消除直接左递归得:

$$P_3 \rightarrow abcP_3'|bcP_3'|cP_3'$$

$$P_3' \rightarrow abcP_3'|\epsilon$$

消除无用产生式: $P_2 \rightarrow P_3ab|ab|b$, $P_1 \rightarrow P_3a|a$,

得文法: $P_3 \rightarrow abcP_3'|bcP_3'|cP_3'$

$$P_3' \rightarrow abcP_3'|\epsilon$$

文法对应的正规式: $V1 = (abc|bc|c)(abc)^*$ 。

示例

文法G[S]: $S \rightarrow Qc|c$ $Q \rightarrow Rb|b$ $R \rightarrow Sa|a$

解: 1) 排序: S(1)、Q(2)、R(3)

2) 代入得: $S \rightarrow Qc|c$

$Q \rightarrow Rb|b$

$R \rightarrow Rbca|bca|ca|a$

消除直接左递归:

$S \rightarrow Qc|c$

$Q \rightarrow Rb|b$

$R \rightarrow bcaR' | caR' | aR'$

$R' \rightarrow bcaR' | \epsilon$

消除隐含的左递归算法与非终极符排序方法无关

- 例如，有产生式：

语句 → **if** 条件 then 语句 else 语句

| **while** 条件 do 语句

| **begin** 语句表 end

若要寻找一个语句，那么关键字 if, while, begin 就提示某个替换式是唯一的替换式。

无回溯！

- 例如，有产生式：

语句 → **if** 条件 then 语句 else 语句

| **if** 条件 then 语句

| **while** 条件 do 语句

| **begin** 语句表 end

产生回溯！

回溯原因

若当前符号 = a ，下一步要展开 A ，

而 $A \rightarrow \alpha_1 | \alpha_2 | \dots | \alpha_n$ ，

怎样选择 α_i ？

(1) 以 a 为头的 α_i 如果只有一个，则替换唯一；

(2) 若以 a 为头有多个 α_i 的，则替换不唯一，可能需要回溯，这是文法的问题，应该变换文法。

文法的要求

- (1) 不含左递归
- (2) 对每个非终结符的候选式，其任何推导的头符号（终结符）集合两两不相交。

- 符号串 α 的**终结首符集** $\text{FIRST}(\alpha)$ 定义为：

$$\text{FIRST}(\alpha) = \{ a \mid \alpha \xRightarrow{*} a... , a \in V_T \}$$

特别地，若 $\alpha \xRightarrow{*} \varepsilon$ ，则规定 $\varepsilon \in \text{FIRST}(\alpha)$ 。

- 以上条件（2）可表示为：对文法的任一非终结符号 A ，若

$$A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$$

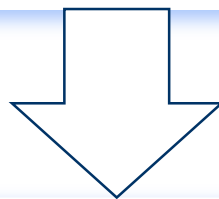
则应有 $\text{FIRST}(\alpha_i) \cap \text{FIRST}(\alpha_j) = \Phi, i \neq j$

回溯解决方法

- 提取公共左因子，将文法改造成任何非终结符的所有候选首符集两两不相交。

$$A \rightarrow \delta\beta_1 \mid \delta\beta_2 \mid \dots \mid \delta\beta_n \mid \gamma_1 \mid \gamma_2 \mid \dots \mid \gamma_m$$

(其中 γ_1 、 γ_2 、 \dots 、 γ_m 不以 δ 开头)



$$A \rightarrow \delta A' \mid \gamma_1 \mid \gamma_2 \mid \dots \mid \gamma_m$$

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

示例

文法G: $S \rightarrow aSb | aS | \epsilon$

解: 提取: $S \rightarrow aS(b | \epsilon)$

$S \rightarrow \epsilon$

引入新符: $S \rightarrow aSA | \epsilon$

$A \rightarrow b | \epsilon$

一般地, 经过反复提取左因子可把每一个非终结符的所有候选首符集变成两两不相交。

LL (1) 分析条件

- 若文法已经消除了左递归，且对每个非终结符满足

$$\text{FIRST}(\alpha_i) \cap \text{FIRST}(\alpha_j) = \Phi$$

➤ 否就解决了**虚假匹配**和**回溯**的问题？

- 对某个输入符号 a ，及待匹配的非终结符

$$A \quad (A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n)$$

➤ a 属于某个候选式的FIRST集合

➤ a 不属于任何候选式的FIRST集合，即对任意 α_i ， $a \notin \text{FIRST}(\alpha_i)$

示例

$G(S): S \rightarrow aA|d$

$A \rightarrow bAS|d|\epsilon$

$\text{FIRST}(S) = \{a, d\}$

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

输入符号串abd是否为句子?

$S \Rightarrow aA$

$\Rightarrow abAS$

$\Rightarrow abdS$

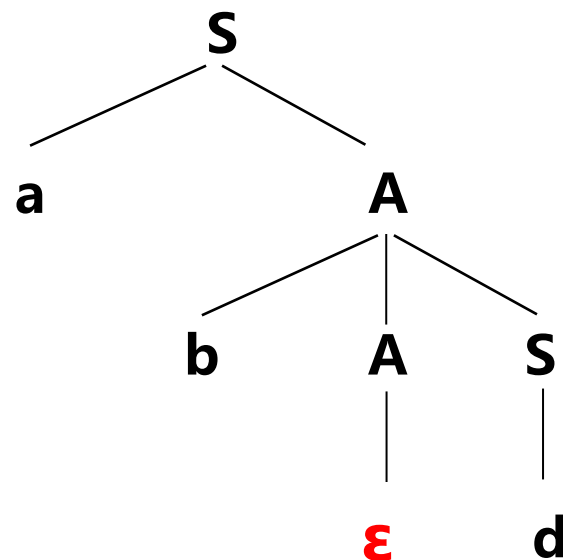
。 。 。

$S \Rightarrow aA$

$\Rightarrow abAS$

$\Rightarrow abS$

$\Rightarrow abd$



这是因为A有产生式 $A \rightarrow \epsilon$,而从开始符号S可以得出 $S \xRightarrow{*} \dots Ad \dots$

- 设S是文法G的开始符号，对G的任何非终结符A，定义**A的后继终结符号集**为：

$$\text{FOLLOW}(A) = \{ a \mid S \Rightarrow \dots Aa\dots, a \in V_T \}$$

- 特别地,若 $S \Rightarrow \dots A$ ，则规定 $\# \in \text{FOLLOW}(A)$ 。

FOLLOW(A)是所有句型中出现在紧接A之后的终结符或“#”。

结论

- 当非终结符A面临输入符号a, 且 $a \notin \text{FIRST}(\alpha_i)$ (对任意i)时, 如果A的某个候选首符集包含 ϵ (即 $\epsilon \in \text{FIRST}(A)$), 那么, 当 $a \in \text{FOLLOW}(A)$ 时, 就允许A自动匹配 (即选用 $A \rightarrow \epsilon$ 工作), 否则, 认为a的出现是一种语法错误。
- 要正确进行不带回溯的语法分析, 文法应满足的第三个条件可表示为: 若A的候选首符集中包含 ϵ , 则

$$\text{FIRST}(A) \cap \text{FOLLOW}(A) = \Phi$$

LL (1) 文法

■ 如果文法G满足以下条件:

(1) 文法消除了左递归;

(2) 文法中每个非终结符A的各个产生式的候选首符集两两不相交, 即

若 $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$,

则 $\text{FIRST}(\alpha_i) \cap \text{FIRST}(\alpha_j) = \Phi$, $(i \neq j)$;

(3) 对文法中的每个非终结符A, 若它存在某个候选首符集中包含 ϵ , 则 $\text{FIRST}(A) \cap \text{FOLLOW}(A) = \Phi$,

则称该文法G为**LL(1)文法**。

LL (1) 分析方法

- 对一个LL (1) 文法, 可以对某个输入串进行有效的无回溯的自上而下分析。
- 设面临的输入符号为 a , 要用非终结符 A 进行匹配, 且 $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$, 则可如下分析:
 - (1) 若 $a \in \text{FIRST}(\alpha_i)$, 则指派 α_i 执行匹配任务;
 - (2) 否则
 - ① 若 $\epsilon \in \text{FIRST}(A)$, 且 $a \in \text{FOLLOW}(A)$, 则让 A 与 ϵ 自动匹配;
 - ② 否则, a 的出现是一种语法错误。

示例

$G(S): S \rightarrow aA|d, A \rightarrow bAS|d|\epsilon$ 是否为LL(1)文法?

解: (1) 不含左递归;

(2) 对于S, $\text{First}(aA)=\{a\}$ $\text{First}(d)=\{d\}$

对于A, $\text{First}(bAS)=\{b\}$ $\text{First}(d)=\{d\}$

$\text{First}(\epsilon)=\{\epsilon\}$

任一非终结符的候选首符集两两不相交

满足条件 (2)

(3) $\text{FIRST}(S)=\{a, d\}$ $\text{FIRST}(A) = \{b, d, \epsilon\}$

$\text{Follow}(S)=\{\#, a, d\}$ $\text{Follow}(A)=\{a, d, \#\}$

所以 $\text{FIRST}(A) \cap \text{Follow}(A)=\{d\} \neq \Phi$

不满足条件 (3)

所以, 该文法不是LL (1) 文法

示例

如果文法为 $G(S): S \rightarrow aA|d, A \rightarrow bAS|\epsilon$ 是否为LL(1)文法?

解: (1) 不含左递归;

(2) 对于S, $\text{First}(aA) = \{a\}$ $\text{First}(d) = \{d\}$

对于A, $\text{First}(bAS) = \{b\}$ $\text{First}(\epsilon) = \{\epsilon\}$

任一非终结符的候选首符集两两不相交

满足条件 (2)

(3) $\text{FIRST}(S) = \{a, d\}$ $\text{FIRST}(A) = \{b, \epsilon\}$

$\text{Follow}(S) = \{\#, a, d\}$ $\text{Follow}(A) = \{\#, a, d\}$

满足条件 (3)

所以, 该文法是LL (1) 文法

示例

修改文法为 $G(S): S \rightarrow aA|d, A \rightarrow bAS|\epsilon$

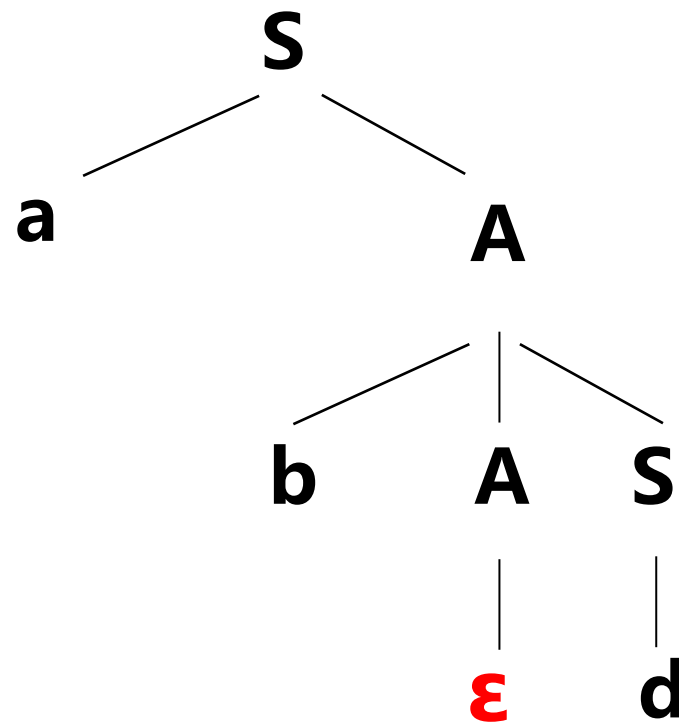
输入符号串abd是否为句子?

$S \Rightarrow aA$

$\Rightarrow abAS$

$\Rightarrow abS$

$\Rightarrow abd$



FIRST (α) 构造

对于符号串 $\alpha = X_1X_2\cdots X_n$, 构造 FIRST (α)

- (1) 置 $\text{FIRST}(\alpha) = \text{FIRST}(X_1) - \{\epsilon\}$;
- (2) 若对 $1 \leq j \leq i-1$, $\epsilon \in \text{FIRST}(X_j)$, 则把 $\text{FIRST}(X_i) - \{\epsilon\}$ 加到 $\text{FIRST}(\alpha)$ 中;
- (3) 若对 $1 \leq j \leq n$, $\epsilon \in \text{FIRST}(X_j)$, 则把 ϵ 加到 $\text{FIRST}(\alpha)$ 中。

FIRST (X) 构造

对于文法G的每个文法符号 $X \in V_T \cup V_N$, 构造FIRST (X) 的方法是:

- (1) 若 $X \in V_T$, 则 $\text{FIRST} (X) = \{X\}$;
- (2) 若 $X \in V_N$, 且有产生式 $X \rightarrow a...$, $a \in V_T$, 则把a加入到FIRST (X) 中, 若有 $X \rightarrow \epsilon$, 则把 ϵ 加入FIRST (X) ;
- (3) 若 $X \in V_N$, 且 $X \rightarrow Y \dots$, $Y \in V_N$, 则把
FIRST (Y) - $\{\epsilon\}$ 加到FIRST (X)中,
若 $X \rightarrow Y_1 Y_2 \dots Y_k$, $Y_1, Y_2, \dots, Y_{i-1} \in V_N$, $\epsilon \in \text{FIRST} (Y_j)$, 则把
($1 \leq j \leq i - 1$)
FIRST (Y_i) - $\{\epsilon\}$ 加到FIRST (X)中。
特别地, 若 $\epsilon \in \text{FIRST} (Y_j)$ ($1 \leq j \leq k$), 则 $\epsilon \in \text{FIRST} (X)$

示例

G: $E \rightarrow TE'$
 $E' \rightarrow +TE' \mid \varepsilon$
 $T \rightarrow FT'$
 $T' \rightarrow *FT' \mid \varepsilon$
 $F \rightarrow (E) \mid i$

求每个非终结符号的FIRST集合

解

$\text{FIRST}(E) = \text{FIRST}(T)$
 $= \text{FIRST}(F)$
 $= \{ (, i \}$

$\text{FIRST}(E') = \{ +, \varepsilon \}$

$\text{FIRST}(T') = \{ *, \varepsilon \}$

示例

G: $E \rightarrow TE'$

$E' \rightarrow +TE' \mid \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \varepsilon$

$F \rightarrow (E) \mid i$

求每个产生式右部符号串的
FIRST集合

解

$\text{FIRST}(TE') = \{ (, i \}$

$\text{FIRST}(+TE') = \{ + \}$

$\text{FIRST}(FT') = \{ (, i \}$

$\text{FIRST}(*FT') = \{ * \}$

$\text{FIRST}((E)) = \{ (\}$

$\text{FIRST}(i) = \{ i \}$

示例

$E \rightarrow TE'$; $E' \rightarrow +TE' \mid \varepsilon$; $T \rightarrow FT'$; $T' \rightarrow *FT' \mid \varepsilon$; $F \rightarrow (E) \mid i$

$\text{FIRST}(E) = \text{FIRST}(T) - \{\varepsilon\} \cup \text{FIRST}(E') - \{\varepsilon\}$

$\text{FIRST}(E') = \{ +, \varepsilon \}$

$\text{FIRST}(T) = \text{FIRST}(F) - \{\varepsilon\} \cup \text{FIRST}(T') - \{\varepsilon\}$

$\text{FIRST}(T') = \{ *, \varepsilon \}$

$\text{FIRST}(F) = \{ (, i \}$

$\text{FIRST}(TE') = \text{FIRST}(T) - \{\varepsilon\} \cup \text{FIRST}(E') - \{\varepsilon\}$

$\text{FIRST}(+TE') = \{ + \}$

$\text{FIRST}(FT') = \text{FIRST}(F) - \{\varepsilon\} \cup \text{FIRST}(T') - \{\varepsilon\}$

$\text{FIRST}(*FT') = \{ * \}$

$\text{FIRST}((E)) = \{ (\}$

FOLLOW(A)构造

对于文法G的每个非终结符，构造FOLLOW(A)的方法是：

(1) 若A为文法开始符号，置#于FOLLOW(A) 中；

(2) 若有产生式 $B \rightarrow \alpha A \beta$,

则将 $\text{FIRST}(\beta) - \{\epsilon\}$ 加到FOLLOW (A)中；

(3) 若有 $B \rightarrow \alpha A$ 或 $B \rightarrow \alpha A \beta$, 且 $\beta \xRightarrow{*} \epsilon$

则将FOLLOW(B)加到FOLLOW(A)中；

(4) 反复使用以上规则, 直至 FOLLOW(A)不再增大为止。

示例

G: $E \rightarrow TE'$
 $E' \rightarrow +TE' \mid \varepsilon$
 $T \rightarrow FT'$
 $T' \rightarrow *FT' \mid \varepsilon$
 $F \rightarrow (E) \mid i$

求每个非终结符号的
FOLLOW集合

解

FOLLOW (E) = { #,) }

FOLLOW (E ') = FOLLOW (E) = { #,) }

FOLLOW (T) = { +, #,) }

FOLLOW (T') = FOLLOW(T) = { +, #,) }

FOLLOW (F) = { *, +, #,) }

示例

$E \rightarrow TE'$; $E' \rightarrow +TE' \mid \varepsilon$; $T \rightarrow FT'$; $T' \rightarrow *FT' \mid \varepsilon$; $F \rightarrow (E) \mid i$

$\text{FIRST}(E) = \text{FIRST}(T) = \text{FIRST}(F) = \{ (, i \}$

$\text{FIRST}(E') = \{ +, \varepsilon \}$

$\text{FIRST}(T') = \{ *, \varepsilon \}$

$\text{FOLLOW}(E) = \{ \# \} \cup \{) \} = \{ \#,) \}$

$\text{FOLLOW}(E') = \text{FOLLOW}(E) - \{ \varepsilon \} \cup \text{FOLLOW}(E') - \{ \varepsilon \} = \{ \#,) \}$

$\text{FOLLOW}(T) = \text{FIRST}(E') - \{ \varepsilon \} \cup \text{FOLLOW}(E) - \{ \varepsilon \} \cup \text{FOLLOW}(E') - \{ \varepsilon \} = \{ +, \#,) \}$

$\text{FOLLOW}(T') = \text{FOLLOW}(T) - \{ \varepsilon \} \cup \text{FOLLOW}(T') - \{ \varepsilon \} = \{ +, \#,) \}$

$\text{FOLLOW}(F) = \text{FIRST}(T') - \{ \varepsilon \} \cup \text{FOLLOW}(T) - \{ \varepsilon \} \cup \text{FOLLOW}(T') - \{ \varepsilon \}$
 $= \{ *, +, \#,) \}$