

✧ 自顶向下 (Top-Down) 语法分析

- ✧ 基本思想
- ✧ 带回溯的自顶向下分析
- ✧ 自顶向下预测分析
- ✧ LL(1) 分析
- ✧ 文法变换：消除左递归、提取左公因子
- ✧ LL(1) 分析中的出错处理
- ✧ LL(K) 文法的有关结论（选讲）

◇ 语法分析

- 核心问题：识别 (*recognition*) 与解析 (*parsing*)

对任意上下文无关文法 $G = (V, T, P, S)$ 和任意 $W \in T^*$ ，是否有 $W \in L(G)$ ？若成立，则给出分析树或（最左）推导步骤；否则，进行报错处理。

- 两种实现途径

自顶向下 (*top-down*) 分析

自底向上 (*bottom-up*) 分析

☆ 自顶向下分析思想

- 从文法开始符号出发进行推导；每一步推导都获得文法的一个句型；直到产生出一个句子，恰好是所期望的终结字符串
- 每一步推导是对当前句型中剩余的某个非终结符进行扩展，即用该非终结符的一个产生式的右部替换该非终结符
- 如果不存在任何一个可以产生出所期望的终结字符串的推导，则表明存在语法错误

◇ 自顶向下分析举例

– 单词序列 aaab 的一个自顶向下分析过程

文法 $G(S)$:	S	($S \rightarrow AB$)
$S \rightarrow AB$	$\Rightarrow AB$	($A \rightarrow aA$)
$A \rightarrow aA \mid \varepsilon$	$\Rightarrow aAB$	($B \rightarrow b$)
$B \rightarrow b \mid bB$	$\Rightarrow aAb$	($A \rightarrow aA$)
	$\Rightarrow aaAb$	($A \rightarrow aA$)
	$\Rightarrow aaaAb$	($A \rightarrow \varepsilon$)
	$\Rightarrow aaab$	

☆ 一般方法

— 两类非确定性

在每一步推导中，选择对哪一个非终结符、
哪一个产生式都可能非确定的

分析成功的结果：得到一个推导

◇ 举例

— 单词序列 aaab 的一个自顶向下分析过程

文法 $G(S)$:

(1) $S \rightarrow AB$

(2) $A \rightarrow aA$

(3) $A \rightarrow \varepsilon$

(4) $B \rightarrow b$

(5) $B \rightarrow bB$

S (1)
 $\Rightarrow AB$ (2)
 $\Rightarrow aAB$ (5)
 $\Rightarrow aAbB$ (2)
 $\Rightarrow aaAbB$ (2)
 $\Rightarrow aaaAbB$ (3)
 $\Rightarrow aaabB$ (回溯)
.....

复杂度很高
失败条件较复杂

◇ 改进的方法

- 仅有产生式选择是非确定的

在每一步推导中，总是对最左边的非终结符进行替换，但选择哪一个产生式是非确定的
分析成功的结果：得到一个最左推导

原理：每个合法的句子都存在至少一个起始于开始符号的最左推导；一个终结符串，只要存在一个起始于开始符号的最左推导，它就是一个合法的句子

从左向右扫描输入单词，失败条件较简单

带回溯的自顶向下分析

◇ 改进的方法举例

– 单词序列 aaab 的一个自顶向下分析过程

文法 G (S) :	S	(1)
	\Rightarrow AB	(2)
(1) $S \rightarrow AB$	\Rightarrow aAB	(3)
(2) $A \rightarrow aA$	\Rightarrow aB	(回溯)
(3) $A \rightarrow \varepsilon$	\Leftarrow aAB	(2)
(4) $B \rightarrow b$	\Rightarrow aaAB	(2)
(5) $B \rightarrow bB$	\Rightarrow aaaAB	(3)
	\Rightarrow aaaB	(5)
复杂度降低	\Rightarrow aaabB	(回溯)
失败条件简化	\Leftarrow aaaB	(4)
	\Rightarrow aaab	(成功)

◇ 确定的自顶向下分析

- 非终结符选择和产生式选择都是确定的

在每一步推导中，总是对最左边的非终结符进行展开，且选择哪一个产生式是确定的，因此是一种无回溯的方法

从左向右扫描，可能向前查看 (lookahead)
确定数目的单词

分析成功的结果：得到唯一的最左推导

分析条件：对文法需要有一定的限制

☆ 举例（向前查看 2 个单词）

— 单词序列 $a^n b^m$ ($n \geq 0, m > 0$) 的预测分析过程

文法 $G(S)$:

(1) $S \rightarrow AB$

(2) $A \rightarrow aA$

(3) $A \rightarrow \varepsilon$

(4) $B \rightarrow b$

(5) $B \rightarrow bB$

只要向前查看 2 个
单词，就可预测分
析 $L(G)$ 中所有句子

S (1)

$\Rightarrow AB$ (2)

$\Rightarrow aAB$ (2)

.....

$\Rightarrow a^n AB$ (3)

$\Rightarrow a^n B$ (5)

$\Rightarrow a^n bB$ (5)

.....

$\Rightarrow a^n b^{m-1} B$ (4)

$\Rightarrow a^n b^m$ (成功)

◇ 左递归带来的问题

— 考虑下列文法识别 ba^n 的分析过程

文法 $G(S)$:

(1) $S \rightarrow Sa$

(2) $S \rightarrow b$

S (1)

$\Rightarrow Sa$ (1)

$\Rightarrow Saa$ (1)

$\Rightarrow Saaa$ (1)

.....

$\Rightarrow Sa^n$ (2)

$\Rightarrow ba^n$

需要向前查看 $n+2$ 个单词，
才能确定这样的推导序列

但是：无论向前查看的单词数确定为多少，
都无法满足预测分析 $L(G)$ 中所有句子的需求

◇ 要求文法不含左递归

— 例：直接左递归

$$P \rightarrow Pa$$
$$P \rightarrow b$$

.....

— 例：间接左递归

$$P \rightarrow Aa$$
$$A \rightarrow Pb$$

.....

— 可以通过文法变换消除左递归

专门讨论

◇ 左公因子带来的问题

- 如下文法需要向前查看3个单词来预测分析 $L(G)$ 中的句子

文法 $G(S) : S \rightarrow abA \mid abB$
 $A \rightarrow a$
 $B \rightarrow b$

- 对于如下文法无法确定需要向前查看多少个单词来预测分析 $L(G)$ 中的句子

文法 $G(S) : S \rightarrow aAb \mid aAc$
 $A \rightarrow a \mid aA$

◇ 通常要求文法不含左公因子

- 可以通过文法变换消除左公因子
专门讨论

☆ 应用较普遍的自顶向下预测分析是
LL (1) 分析

- 要求文法一定是 LL (1) 文法
专门讨论

☆ LL (1) 的含义

- 第一个“L”，代表从左 (Left) 向右扫描单词
- 第二个“L”，代表产生的是最左 (Leftmost) 推导
- “1”代表向前查看 (lookahead) 一个单词

☆ 对文法的限制

- 要求文法是 LL (1) 的
- 什么是 LL (1) 文法?

先引入两个重要概念

☆ 两个重要概念

- First 集合
- Follow 集合

☆ First 集合

— 定义

设 $G = (V_T, V_N, P, S)$ 是上下文无关文法
对 $\alpha \in (V_T \cup V_N)^*$,

$$\text{First}(\alpha) = \{ a \mid \alpha \Rightarrow^* a\beta, a \in V_T, \beta \in (V_T \cup V_N)^*, \\ \text{或者 } \alpha \Rightarrow^* \epsilon \text{ 时 } a = \epsilon \}$$

或者

$$\text{First}(\alpha) = \{ a \mid \alpha \Rightarrow_{lm}^* a\beta, a \in V_T, \beta \in (V_T \cup V_N)^*, \\ \text{或者 } \alpha \Rightarrow_{lm}^* \epsilon \text{ 时 } a = \epsilon \}$$

◇ 计算 First 集合

对所有 $x \in V_N \cup V_T \cup \{\varepsilon\} \cup \{v \mid A \rightarrow u \in P, \text{ 且 } v \text{ 是 } u \text{ 的后缀}\}$, 则

- 对 $x \in V_T \cup \{\varepsilon\}$, 置 $\text{First}(x) = \{x\}$; 对其它 x , 置 $\text{First}(x) = \Phi$
- 重复如下过程, 直到所有 First 集合没有变化为止:

(1) 对于 $A \rightarrow \varepsilon \in P$, 置 $\text{First}(A) = \text{First}(A) \cup \{\varepsilon\}$.

(2) 对于 $Y_1 Y_2 \dots Y_K \in \{v \mid A \rightarrow u \in P, \text{ 且 } v \text{ 是 } u \text{ 的后缀}\}$, 其中 $k \geq 1$, $Y_j \in V_N \cup V_T$ ($1 \leq j \leq k$), 若 $\forall j: 1 \leq j \leq i-1 (\varepsilon \in \text{First}(Y_j)) \wedge \varepsilon \notin \text{First}(Y_i)$, 其中 $1 \leq i \leq k$, 则令

$$\text{First}(Y_1 Y_2 \dots Y_K) = \bigcup_{j=1}^i \text{First}(Y_j) - \{\varepsilon\}$$

否则, 若 $\forall j: 1 \leq j \leq k (\varepsilon \in \text{First}(Y_j))$, 则令

$$\text{First}(Y_1 Y_2 \dots Y_K) = \bigcup_{j=1}^k \text{First}(Y_j).$$

(3) 若有 $A \rightarrow Y_1 Y_2 \dots Y_K \in P$, 则置 $\text{First}(A) = \text{First}(A) \cup \text{First}(Y_1 Y_2 \dots Y_K)$.

◇ 例：计算 First 集合

文法 $G(S)$:

(1) $S \rightarrow AB$

(2) $A \rightarrow Da \mid \varepsilon$

(3) $B \rightarrow cC$

(4) $C \rightarrow aADC \mid \varepsilon$

(5) $D \rightarrow b \mid \varepsilon$

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

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

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

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

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

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

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

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

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

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

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

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

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

◇ 例：计算 First 集合(续)

文法 $G(S)$:

(1) $S \rightarrow AB$

(2) $A \rightarrow Da \mid \varepsilon$

(3) $B \rightarrow cC$

(4) $C \rightarrow aADC \mid \varepsilon$

(5) $D \rightarrow b \mid \varepsilon$

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

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

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

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

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

$\text{First}(A) = \{\varepsilon\}$

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

$\text{First}(C) = \{\varepsilon\}$

$\text{First}(D) = \{\varepsilon, b\}$

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

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

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

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

✧ 例：计算 First 集合(续)

文法 $G(S)$:

(1) $S \rightarrow AB$

(2) $A \rightarrow Da \mid \varepsilon$

(3) $B \rightarrow cC$

(4) $C \rightarrow aADC \mid \varepsilon$

(5) $D \rightarrow b \mid \varepsilon$

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

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

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

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

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

$\text{First}(A) = \{\varepsilon\}$

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

$\text{First}(C) = \{\varepsilon\}$

$\text{First}(D) = \{\varepsilon, b\}$

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

$\text{First}(Da) = \{a, b\}$

$\text{First}(cC) = \{c\}$

$\text{First}(aADC) = \{a\}$

◇ 例：计算 First 集合(续)

文法 $G(S)$:

(1) $S \rightarrow AB$

(2) $A \rightarrow Da \mid \varepsilon$

(3) $B \rightarrow cC$

(4) $C \rightarrow aADC \mid \varepsilon$

(5) $D \rightarrow b \mid \varepsilon$

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

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

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

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

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

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

$\text{First}(B) = \{c\}$

$\text{First}(C) = \{a, \varepsilon\}$

$\text{First}(D) = \{\varepsilon, b\}$

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

$\text{First}(Da) = \{a, b\}$

$\text{First}(cC) = \{c\}$

$\text{First}(aADC) = \{a\}$

☆ 例：计算 First 集合(续)

文法 $G(S)$:

(1) $S \rightarrow AB$

(2) $A \rightarrow Da \mid \varepsilon$

(3) $B \rightarrow cC$

(4) $C \rightarrow aADC \mid \varepsilon$

(5) $D \rightarrow b \mid \varepsilon$

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

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

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

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

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

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

$\text{First}(B) = \{c\}$

$\text{First}(C) = \{a, \varepsilon\}$

$\text{First}(D) = \{\varepsilon, b\}$

$\text{First}(AB) = \{a, b, c\}$

$\text{First}(Da) = \{a, b\}$

$\text{First}(cC) = \{c\}$

$\text{First}(aADC) = \{a\}$

◇ 例：计算 First 集合(续)

文法 $G(S)$:

(1) $S \rightarrow AB$

(2) $A \rightarrow Da \mid \varepsilon$

(3) $B \rightarrow cC$

(4) $C \rightarrow aADC \mid \varepsilon$

(5) $D \rightarrow b \mid \varepsilon$

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

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

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

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

$\text{First}(S) = \{a, b, c\}$

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

$\text{First}(B) = \{c\}$

$\text{First}(C) = \{a, \varepsilon\}$

$\text{First}(D) = \{\varepsilon, b\}$

$\text{First}(AB) = \{a, b, c\}$

$\text{First}(Da) = \{a, b\}$

$\text{First}(cC) = \{c\}$

$\text{First}(aADC) = \{a\}$

☆ Follow 集合

— 定义

设 $G = (V_T, V_N, P, S)$ 是上下文无关文法, 对每个 $A \in V_N$,

$$\text{Follow}(A) = \{ a \mid S\# \Rightarrow^* \alpha A \beta\# \text{ 且 } a \in \text{First}(\beta\#), \\ \alpha, \beta \in (V_T \cup V_N)^* \}$$

(# 代表输入单词序列右边的结束符)

☆ 计算 Follow 集合

- 置 $\text{Follow}(S) = \{\#\}$, 置所有其它的 Follow 集合为 \emptyset ;
- 重复如下步骤, 直至所有 Follow 集不再变化为止:

对于 $A \rightarrow \alpha B \beta \in P$, 把 $\text{First}(\beta) - \{\epsilon\}$ 加至 $\text{Follow}(B)$;

若有 $\epsilon \in \text{First}(\beta)$, 则把 $\text{Follow}(A)$ 加至 $\text{Follow}(B)$ 中.

☆ 例：计算 First 和 Follow 集合

文法 $G(S)$:

(1) $S \rightarrow AB$

(2) $A \rightarrow Da \mid \varepsilon$

(3) $B \rightarrow cC$

(4) $C \rightarrow aADC \mid \varepsilon$

(5) $D \rightarrow b \mid \varepsilon$

$\text{First}(B) = \{c\}$

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

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

$\text{First}(DC) = \{b, a, \varepsilon\}$

$\text{First}(C) = \{a, \varepsilon\}$

$\text{Follow}(S) = \{\#\}$

$\text{Follow}(A) = \{\}$

$\text{Follow}(B) = \{\}$

$\text{Follow}(C) = \{\}$

$\text{Follow}(D) = \{\}$

☆ 例：计算 First 和 Follow 集合

文法 $G(S)$:

(1) $S \rightarrow AB$ \checkmark

(2) $A \rightarrow Da \mid \varepsilon$

(3) $B \rightarrow cC$

(4) $C \rightarrow aADC \mid \varepsilon$

(5) $D \rightarrow b \mid \varepsilon$

$\text{First}(B) = \{c\}$

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

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

$\text{First}(DC) = \{b, a, \varepsilon\}$

$\text{First}(C) = \{a, \varepsilon\}$

$\text{Follow}(S) = \{\#\}$

$\text{Follow}(A) = \{\}$

$\text{Follow}(B) = \{\}$

$\text{Follow}(C) = \{\}$

$\text{Follow}(D) = \{\}$

$\text{Follow}(A) = \{c\}$

$\text{Follow}(B) = \{\#\}$

☆ 例：计算 First 和 Follow 集合

文法 $G(S)$:

(1) $S \rightarrow AB$

(2) $A \rightarrow Da \mid \varepsilon$ \checkmark

(3) $B \rightarrow cC$

(4) $C \rightarrow aADC \mid \varepsilon$

(5) $D \rightarrow b \mid \varepsilon$

$\text{First}(B) = \{c\}$

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

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

$\text{First}(DC) = \{b, a, \varepsilon\}$

$\text{First}(C) = \{a, \varepsilon\}$

$\text{Follow}(S) = \{\#\}$

$\text{Follow}(D) = \{a\}$

$\text{Follow}(A) = \{c\}$

$\text{Follow}(B) = \{\#\}$

$\text{Follow}(C) = \{\}$

$\text{Follow}(D) = \{\}$

☆ 例：计算 First 和 Follow 集合

文法 $G(S)$:

(1) $S \rightarrow AB$

(2) $A \rightarrow Da \mid \varepsilon$

(3) $B \rightarrow cC$ \checkmark

(4) $C \rightarrow aADC \mid \varepsilon$

(5) $D \rightarrow b \mid \varepsilon$

$\text{First}(B) = \{c\}$

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

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

$\text{First}(DC) = \{b, a, \varepsilon\}$

$\text{First}(C) = \{a, \varepsilon\}$

$\text{Follow}(S) = \{\#\}$

$\text{Follow}(C) = \{\#\}$

$\text{Follow}(A) = \{c\}$

$\text{Follow}(B) = \{\#\}$

$\text{Follow}(C) = \{\}$

$\text{Follow}(D) = \{a\}$

☆ 例：计算 First 和 Follow 集合

文法 $G(S)$:

(1) $S \rightarrow AB$

(2) $A \rightarrow Da \mid \varepsilon$

(3) $B \rightarrow cC$

(4) $C \rightarrow aADC \mid \varepsilon \checkmark$

(5) $D \rightarrow b \mid \varepsilon$

$\text{First}(B) = \{c\}$

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

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

$\text{First}(DC) = \{b, a, \varepsilon\}$

$\text{First}(C) = \{a, \varepsilon\}$

$\text{Follow}(S) = \{\#\}$

$\text{Follow}(A) = \{c\}$

$\text{Follow}(B) = \{\#\}$

$\text{Follow}(C) = \{\#\}$

$\text{Follow}(D) = \{a\}$

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

$\text{Follow}(D) = \{a, \#\}$

☆ 例：计算 First 和 Follow 集合

文法 $G(S)$:

(1) $S \rightarrow AB$ \checkmark

(2) $A \rightarrow Da \mid \varepsilon$ \checkmark

(3) $B \rightarrow cC$ \checkmark

(4) $C \rightarrow aADC \mid \varepsilon$ \checkmark

(5) $D \rightarrow b \mid \varepsilon$

$\text{First}(B) = \{c\}$

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

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

$\text{First}(DC) = \{b, a, \varepsilon\}$

$\text{First}(C) = \{a, \varepsilon\}$

$\text{Follow}(S) = \{\#\}$

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

$\text{Follow}(B) = \{\#\}$

$\text{Follow}(C) = \{\#\}$

$\text{Follow}(D) = \{a, \#\}$

\checkmark

☆ 定义： 预测集合 (*Predictive Set*)

设 $G = (V_T, V_N, P, S)$ 是上下文无关文法。

对任何产生式 $A \rightarrow \alpha \in P$ ，其预测集合 $PS(A \rightarrow \alpha)$ 定义为：

— 如果 $\varepsilon \notin first(\alpha)$ ，那么

$$PS(A \rightarrow \alpha) = first(\alpha);$$

— 如果 $\varepsilon \in first(\alpha)$ ，那么

$$PS(A \rightarrow \alpha) = (first(\alpha) - \{\varepsilon\}) \cup follow(A)$$

☆ 定义：LL (1) 文法

文法 G 是 LL (1) 文法，当且仅当对于 G 的每个非终结符 A 的任何两个不同产生式 $A \rightarrow \alpha \mid \beta$ ，下面的条件成立：

$$PS(A \rightarrow \alpha) \cap PS(A \rightarrow \beta) = \emptyset$$

◇ 举例：验证如下文法 G (S) 不是 LL (1) 文法

文法 G (S) :

- (1) $S \rightarrow AB$
- (2) $A \rightarrow Da \mid \varepsilon$
- (3) $B \rightarrow cC$
- (4) $C \rightarrow aADC \mid \varepsilon$
- (5) $D \rightarrow b \mid \varepsilon$

$$PS(A \rightarrow Da) = \{b, a\}$$

$$PS(A \rightarrow \varepsilon) = \{c, b, a, \#\}$$

$$PS(C \rightarrow aADC) = \{a\}$$

$$PS(C \rightarrow \varepsilon) = \{\#\}$$

$$PS(D \rightarrow b) = \{b\}$$

$$PS(D \rightarrow \varepsilon) = \{a, \#\}$$

$$\text{First}(Da) = \{b, a\}$$

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

$$\text{First}(aADC) = \{a\}$$

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

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

$$\text{Follow}(C) = \{\#\}$$

$$\text{Follow}(D) = \{a, \#\}$$

$$PS(A \rightarrow Da) \cap PS(A \rightarrow \varepsilon) \neq \emptyset$$

$$PS(C \rightarrow aADC) \cap PS(C \rightarrow \varepsilon) = \emptyset$$

$$PS(D \rightarrow b) \cap PS(D \rightarrow \varepsilon) = \emptyset$$

☆ LL (1) 分析的实现

– 递归下降 LL (1) 分析程序

每个非终结符对应一个分析子程序

– 表驱动 LL (1) 分析程序

借助于预测分析表和一个下推栈

◇ 递归下降LL(1)分析程序

— 工作原理

每个非终结符都对应一个子程序。该子程序的行为根据语法描述来明确：根据下一个输入符号来确定按照哪一个产生式进行处理，再根据该产生式的右端，

- 每遇到一个终结符，则判断当前读入的单词是否与该终结符相匹配，若匹配，再读取下一个单词继续分析；不匹配，则进行出错处理
- 每遇到一个非终结符，则调用相应的子程序

◇ 非终结符对应的递归下降子程序

- 例 对于下列关于 function 的唯一产生式

$\langle \text{function} \rangle \rightarrow \text{FUNC ID (} \langle \text{parameter_list} \rangle \text{) } \langle \text{statement} \rangle$

($\langle \text{function} \rangle$, $\langle \text{parameter_list} \rangle$, 和 $\langle \text{statement} \rangle$ 是非终结符)

```
void ParseFunction()
{
    MatchToken(T_FUNC);    //匹配FUNC
    MatchToken(T_ID);      //匹配ID
    MatchToken(T_LPAREN);  // 匹配 (
    ParseParameterList();
    MatchToken(T_RPAREN);  // 匹配 )
    ParseStatement();
}
```

◇ 非终结符对应的递归下降子程序

— 例 续上页

```
void MatchToken(int expected)
{
    if (lookahead != expected) //判别当前单词是否与
    {                           //期望的终结符匹配
        printf("syntax error \n");
        exit(0);
    }
    else // 若匹配,消费掉当前单词并读入下一个
        lookahead = getToken(); //调用词法分析程序
}
```

递归下降 LL (1) 分析程序

◇ 递归下降LL(1)分析程序举例

— 例 对于下列文法 $G(S)$:

$$S \rightarrow AaS \mid BbS \mid d$$
$$A \rightarrow a$$
$$B \rightarrow \varepsilon \mid c$$
$$PS(S \rightarrow AaS) = \{a\}$$
$$PS(S \rightarrow BbS) = \{c, b\}$$
$$PS(S \rightarrow d) = \{d\}$$
$$PS(A \rightarrow a) = \{a\}$$
$$PS(B \rightarrow \varepsilon) = \{b\}$$
$$PS(B \rightarrow c) = \{c\}$$
$$\text{First}(AaS) = \{a\}$$
$$\text{First}(BbS) = \{c, b\}$$
$$\text{First}(d) = \{d\}$$
$$\text{First}(a) = \{a\}$$
$$\text{First}(\varepsilon) = \{\varepsilon\}$$
$$\text{First}(c) = \{c\}$$
$$\text{Follow}(S) = \{\#\}$$
$$\text{Follow}(A) = \{a\}$$
$$\text{Follow}(B) = \{b\}$$

$PS(S \rightarrow AaS)$, $PS(S \rightarrow BbS)$ 以及 $PS(S \rightarrow d)$ 互不相交, $PS(B \rightarrow \varepsilon)$ 和 $PS(S \rightarrow d)$ 互不相交, 所以, $G(S)$ 是 LL(1) 文法

递归下降 LL (1) 分析程序

◇ 非终结符对应的 递归下降子程序

— 一般结构

设 A 的产生式:

$$A \rightarrow u_1 \mid u_2 \mid \dots \mid u_n,$$

相对于非终结符 A
的递归下降子程序
ParseA 的一般结
构如右图所示

```
void ParseA()
{
    switch (lookahead) {
        case  $PS(A \rightarrow u_1)$ :
            /* code to recognize  $u_1$  */
            break;
        case  $PS(A \rightarrow u_2)$ :
            /* code to recognize  $u_2$  */
            break;
        ...
        case  $PS(A \rightarrow u_n)$ :
            /* code to recognize  $u_n$  */
            break;
        default:
            printf("syntax error \n");
            exit(0);
    }
}
```

递归下降 LL (1) 分析程序

– 接上例 针对文法G(S)构造的递归下降分析程序

G(S): $S \rightarrow AaS \mid BbS \mid d$

$A \rightarrow a$

$B \rightarrow \varepsilon \mid c$

$PS(S \rightarrow AaS) = \{a\}$

$PS(S \rightarrow BbS) = \{c, b\}$

$PS(S \rightarrow d) = \{d\}$

```
void ParseS( )
{
    switch (lookahead) {
        case a:
            ParseA( );
            MatchToken(a);
            ParseS( );
            break;
```

```
        case b,c:
            ParseB( );
            MatchToken(b);
            ParseS( );
            break;
        case d:
            MatchToken(d);
            break;
        default:
            printf("syntax error \n");
            exit(0);
    }
}
```

递归下降 LL (1) 分析程序

– 接上例 针对文法G(S) 构造的递归下降分析程序

G(S): $S \rightarrow AaS \mid BbS \mid d$
 $A \rightarrow a$
 $B \rightarrow \varepsilon \mid c$

```
void ParseB( )
{
    if (lookahead==c) {
        MatchToken(c);
    }
    else if (lookahead==b) {
    }
    else {
        printf("syntax error \n");
        exit(0);
    }
}
```

$PS(A \rightarrow a) = \{a\}$
 $PS(B \rightarrow \varepsilon) = \{b\}$
 $PS(B \rightarrow c) = \{c\}$

```
void ParseA( )
{
    if (lookahead==a) {
        MatchToken(a);
    }
    else {
        printf("syntax error \n");
        exit(0);
    }
}
```

◇ 实际应用中的推广

上面只讨论了根据普通文法构造递归下降分析程序。实际上，也可以将产生式右端扩展为更复杂的描述表达式，即除了文法符号之间的连接运算之外，还可以有选择、重复、任选以及优先括号等运算（如 EBNF 范式中的运算），以使语法描述更加简洁，分析程序更加高效（比较：若将其展开为普通文法，则需要引入多个非终结符，增加多个对应的子程序）。

- $X_1 | X_1 | \dots | X_m$ 多个成分之间的选择
- $\{X\}$ 成分 X 的重复（0 到多次）
- $[X]$ 成分 X 的任选（0 或 1 次）
- (X) 成分 X 优先

☆ 实际应用中的推广

将产生式右端扩展后，同样要求它的 First 集合，以适应递归下降分析程序的构造方法。

- $\text{First} (X_1 \mid X_2 \mid \dots \mid X_m) = \text{First} (X_1) \cup \dots \cup \text{First} (X_m)$
- $\text{First} (\{ X \}) = \text{First} (X) \cup \{\epsilon\}$
- $\text{First} ([X]) = \text{First} (X) \cup \{\epsilon\}$
- $\text{First} ((X)) = \text{First} (X)$

◇ 实际应用中的推广

将产生式右端扩展后，子程序的处理过程中需要针对不同运算选择不同的语句形式（普通文法只有连接运算，所以只对应顺序语句）。

- $X_1 \mid X_2 \mid \dots \mid X_m$ 对应选择语句
- $\{ X \}$ 对应循环语句
- $[X]$ 对应 If-Then 语句
- (X) 对应 Block 语句
- 可参考 PL/0 编译器的语法分析程序

◇ 表驱动LL(1)分析程序

— 工作原理 利用预测分析表和一个下推栈实现

初始时，下推栈只包含#；首先将文法开始符号入栈；之后依如下步骤：（1）若栈顶为终结符，则判断当前读入的单词是否与该终结符相匹配，若匹配，再读取下一单词继续分析；不匹配，则进行出错处理；（2）若栈顶为非终结符，则根据该非终结符和当前输入单词查预测分析表，若相应表项中是产生式（唯一的），则将此非终结符出栈，并把产生式右部符号从右至左入栈；若表项为空，则进行出错处理；（3）重复（1）和（2），直到栈顶为#同时输入也遇到结束符#时，分析结束

◇ 预测分析表

- 表驱动分析程序需要的二维表 M
- 表的每一行 A 对应一个非终结符
- 表的每一列 a 对应某个终结符或输入结束符 $\#$
- 表中的项 $M(A,a)$ 表示栈顶为 A ，下一个输入符号为 a 时，可选的产生式集合
- 对于 LL (1) 文法，可以构造出一个 $M(A,a)$ 最多只包含一个产生式的预测分析表，可称之为 LL (1) 分析表
- $M(A,a)$ 不含产生式时，对应一个出错位置

◇ 预测分析表的构造算法

- 对文法 G 的每个产生式 $A \rightarrow \alpha$ 执行如下步骤:

对每个 $a \in PS(A \rightarrow \alpha)$, 将 $A \rightarrow \alpha$ 加入 $M[A, a]$

- 把所有无定义的 $M[A, a]$ 标上“出错标志”

可以证明: 一个文法 G 的预测分析表不含多重入口, 当且仅当该文法是 LL(1) 的

表驱动 LL (1) 分析程序



清华大学

《编译原理》

◇ 预测分析表的构造举例

— 对于下列文法 $G(S)$:

$S \rightarrow AaS \mid BbS \mid d$

$A \rightarrow a$

$B \rightarrow \varepsilon \mid c$

$PS(S \rightarrow AaS) = \{a\}$

$PS(S \rightarrow BbS) = \{c, b\}$

$PS(S \rightarrow d) = \{d\}$

$PS(A \rightarrow a) = \{a\}$

$PS(B \rightarrow \varepsilon) = \{b\}$

$PS(B \rightarrow c) = \{c\}$

可构造如下预测分析表:

	a	b	c	d	#
S	$S \rightarrow AaS$	$S \rightarrow BbS$	$S \rightarrow BbS$	$S \rightarrow d$	
A	$A \rightarrow a$				
B		$B \rightarrow \varepsilon$	$B \rightarrow c$		

表驱动 LL (1) 分析程序

◇ 表驱动预测分析程序分析算法

```
初始时 ‘#’入栈，然后文法开始符号入栈；首个输入符号读进 a；  
flag = TRUE；  
while (flag) do {  
    栈顶符号出栈并放在X中；  
    if ( $X \in V_T$ ) {  
        if ( $X == a$ )  
            把下一个输入符号读进a；  
        else ERROR；  
    }  
    else if ( $X == \#$ ) {  
        if ( $a == \#$ ) flag = FALSE；  
        else ERROR；  
    }  
    else if ( $M[X, a] == \{X \rightarrow X_1 X_2 \dots X_k\}$ )  $X_k, X_{k-1}, \dots, X_1$ 依次进栈；  
    else ERROR；  
}  
/*分析成功，过程完毕*/
```

表驱动 LL (1) 分析程序

✧ 表驱动预测分析过程举例

— 对于下列文法 $G(S)$:

$S \rightarrow AaS \mid BbS \mid d$

$A \rightarrow a$

$B \rightarrow \varepsilon \mid c$

剩余的输入串

aabd#

S
#

分析输入串 **aabd** 的过程:

	a	b	c	d	#
S	$S \rightarrow AaS$	$S \rightarrow BbS$	$S \rightarrow BbS$	$S \rightarrow d$	
A	$A \rightarrow a$				
B		$B \rightarrow \varepsilon$	$B \rightarrow c$		

表驱动 LL (1) 分析程序

✧ 表驱动预测分析过程举例

— 对于下列文法 $G(S)$:

$S \rightarrow AaS \mid BbS \mid d$

$A \rightarrow a$

$B \rightarrow \varepsilon \mid c$

剩余的输入串

aabd#

A
a
S
#

分析输入串 *aabd* 的过程:

	a	b	c	d	#
S	$S \rightarrow AaS$	$S \rightarrow BbS$	$S \rightarrow BbS$	$S \rightarrow d$	
A	$A \rightarrow a$				
B		$B \rightarrow \varepsilon$	$B \rightarrow c$		

表驱动 LL (1) 分析程序

✧ 表驱动预测分析过程举例

— 对于下列文法 $G(S)$:

$S \rightarrow AaS \mid BbS \mid d$

$A \rightarrow a$

$B \rightarrow \varepsilon \mid c$

剩余的输入串

aabd#

a
a
S
#

分析输入串 **aabd** 的过程:

	a	b	c	d	#
S	$S \rightarrow AaS$	$S \rightarrow BbS$	$S \rightarrow BbS$	$S \rightarrow d$	
A	$A \rightarrow a$				
B		$B \rightarrow \varepsilon$	$B \rightarrow c$		

表驱动 LL (1) 分析程序



清华大学

《编译原理》

✧ 表驱动预测分析过程举例

— 对于下列文法 $G(S)$:

$S \rightarrow AaS \mid BbS \mid d$

$A \rightarrow a$

$B \rightarrow \varepsilon \mid c$

剩余的输入串

$a b d \#$

**a
 S
 $\#$**

分析输入串 **$a a b d$** 的过程:

	a	b	c	d	#
S	$S \rightarrow AaS$	$S \rightarrow BbS$	$S \rightarrow BbS$	$S \rightarrow d$	
A	$A \rightarrow a$				
B		$B \rightarrow \varepsilon$	$B \rightarrow c$		

表驱动 LL (1) 分析程序



清华大学

《编译原理》

◇ 表驱动预测分析过程举例

— 对于下列文法 $G(S)$:

$S \rightarrow AaS \mid BbS \mid d$

$A \rightarrow a$

$B \rightarrow \varepsilon \mid c$

剩余的输入串

$bd\#$

S
 $\#$

分析输入串 **$aabd$** 的过程:

	a	b	c	d	#
S	$S \rightarrow AaS$	$S \rightarrow BbS$	$S \rightarrow BbS$	$S \rightarrow d$	
A	$A \rightarrow a$				
B		$B \rightarrow \varepsilon$	$B \rightarrow c$		

表驱动 LL (1) 分析程序

◇ 表驱动预测分析过程举例

— 对于下列文法 $G(S)$:

$S \rightarrow AaS \mid BbS \mid d$

$A \rightarrow a$

$B \rightarrow \varepsilon \mid c$

剩余的输入串

$bd\#$

**B
 b
 S
 $\#$**

分析输入串 **$aabd$** 的过程:

	a	b	c	d	#
S	$S \rightarrow AaS$	$S \rightarrow BbS$	$S \rightarrow BbS$	$S \rightarrow d$	
A	$A \rightarrow a$				
B		$B \rightarrow \varepsilon$	$B \rightarrow c$		

表驱动 LL (1) 分析程序

◇ 表驱动预测分析过程举例

— 对于下列文法 $G(S)$:

$S \rightarrow AaS \mid BbS \mid d$ 剩余的输入串
 $A \rightarrow a$ **$bd\#$**
 $B \rightarrow \varepsilon \mid c$

b
 S
 $\#$

分析输入串 **$aabd$** 的过程:

	a	b	c	d	#
S	$S \rightarrow AaS$	$S \rightarrow BbS$	$S \rightarrow BbS$	$S \rightarrow d$	
A	$A \rightarrow a$				
B		$B \rightarrow \varepsilon$	$B \rightarrow c$		

表驱动 LL (1) 分析程序



清华大学

《编译原理》

☆ 表驱动预测分析过程举例

— 对于下列文法 $G(S)$:

$S \rightarrow AaS \mid BbS \mid d$

$A \rightarrow a$

$B \rightarrow \varepsilon \mid c$

剩余的输入串

$d\#$

S
 $\#$

分析输入串 $aabd$ 的过程:

	a	b	c	d	#
S	$S \rightarrow AaS$	$S \rightarrow BbS$	$S \rightarrow BbS$	$S \rightarrow d$	
A	$A \rightarrow a$				
B		$B \rightarrow \varepsilon$	$B \rightarrow c$		

表驱动 LL (1) 分析程序

◇ 表驱动预测分析过程举例

— 对于下列文法 $G(S)$:

$S \rightarrow AaS \mid BbS \mid d$ 剩余的输入串
 $A \rightarrow a$ $d\#$
 $B \rightarrow \varepsilon \mid c$

$d\#$

分析输入串 $aabd$ 的过程:

	a	b	c	d	#
S	$S \rightarrow AaS$	$S \rightarrow BbS$	$S \rightarrow BbS$	$S \rightarrow d$	
A	$A \rightarrow a$				
B		$B \rightarrow \varepsilon$	$B \rightarrow c$		

表驱动 LL (1) 分析程序

✧ 表驱动预测分析过程举例

— 对于下列文法 $G(S)$:

$S \rightarrow AaS \mid BbS \mid d$ 剩余的输入串

$A \rightarrow a$ #

$B \rightarrow \varepsilon \mid c$



#

分析输入串 **aabd** 的过程:

	a	b	c	d	#
S	$S \rightarrow AaS$	$S \rightarrow BbS$	$S \rightarrow BbS$	$S \rightarrow d$	
A	$A \rightarrow a$				
B		$B \rightarrow \varepsilon$	$B \rightarrow c$		

◇ 文法变换：消除左递归、提取左公因子

- LL(1) 文法通常不含左递归和左公因子
- 许多文法在消除左递归和提取左公因子后可以变换为LL(1)文法
- 但不含左递归和左公因子的文法不一定是LL(1)文法

◇ 左递归消除规则

– 消除直接左递归

对形如

$$P \rightarrow P\alpha \mid \beta$$

的产生式，其中 α 非 ε ， β 不以 P 打头，
可改写为：

$$P \rightarrow \beta Q$$

$$Q \rightarrow \alpha Q \mid \varepsilon$$

其中 Q 为新增加的非终结符

◇ 左递归消除规则

– 消除直接左递归

对更一般的形如

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

的一组产生式，其中 α_i ($1 \leq i \leq m$) 不为 ε ， β_j ($1 \leq j \leq n$) 不以 P 打头，

可改写为：

$$P \rightarrow \beta_1 Q \mid \beta_2 Q \mid \dots \mid \beta_n Q$$

$$Q \rightarrow \alpha_1 Q \mid \alpha_2 Q \mid \dots \mid \alpha_m Q \mid \varepsilon$$

其中 Q 为新增加的非终结符

✧ 左递归消除举例

原文法 $G[E]$: $E \rightarrow E + T \mid T$
 $T \rightarrow T * F \mid F$
 $F \rightarrow (E) \mid a$

消除左递归后的文法 $G'[E]$:

- | | |
|----------------------------------|----------------------------------|
| (1) $E \rightarrow TE'$ | (2) $E' \rightarrow + TE'$ |
| (3) $E' \rightarrow \varepsilon$ | (4) $T \rightarrow FT'$ |
| (5) $T' \rightarrow * FT'$ | (6) $T' \rightarrow \varepsilon$ |
| (7) $F \rightarrow (E)$ | (8) $F \rightarrow a$ |

◇ 左递归消除规则

— 消除一般左递归

对无回路($A \Rightarrow^+ A$)、无 ε -产生式的文法，通过下列步骤可消除一般左递归（包括直接和间接左递归）：

(1) 以某种顺序将文法非终结符排列 $A_1, A_2 \dots A_n$

(2) for $i = 1, n$ do {

 for $j = 1, i-1$ do {

 用 $A_i \rightarrow \alpha_1 r \mid \alpha_2 r \dots \mid \alpha_k r$ 反复替代形如 $A_i \rightarrow A_j r$ 的规则，
 其中 $A_j \rightarrow \alpha_1 \mid \alpha_2 \dots \mid \alpha_k$ 是关于 A_j 的全部产生式；

 }

 消除 A_i 规则的直接左递归；

}

(3) 化简由 (2) 得到的文法

✧ 左递归消除举例

原文法 $G[S]$: $S \rightarrow PQ \mid a$
 $P \rightarrow QS \mid b$
 $Q \rightarrow SP \mid c$

非终结符排序为 S 、 P 、 Q ，按造消除一般左递归的方法，进行如下变换：

$Q \rightarrow SP \mid c$

$\Rightarrow Q \rightarrow PQP \mid aP \mid c$

$\Rightarrow Q \rightarrow QSQP \mid bQP \mid aP \mid c$

$\Rightarrow Q \rightarrow bQPR \mid aPR \mid cR$
 $R \rightarrow SQPR \mid \varepsilon$

结果：

$S \rightarrow PQ \mid a$
 $P \rightarrow QS \mid b$
 $Q \rightarrow bQPR \mid aPR \mid cR$
 $R \rightarrow SQPR \mid \varepsilon$

文法变换：消除左递归



清华大学

《编译原理》

✧ 左递归消除举例

原文法 $G[S]$: $S \rightarrow PQ \mid a$
 $P \rightarrow QS \mid b$
 $Q \rightarrow SP \mid c$

按造非终结符的另一种排序Q、P、S，依消除一般左递归的方法，进行如下变换：

$P \rightarrow QS \mid b$
 $\Rightarrow P \rightarrow SPS \mid cS \mid b$

$S \rightarrow PQ \mid a$
 $\Rightarrow S \rightarrow SPSQ \mid cSQ \mid bQ \mid a$

$\Rightarrow S \rightarrow cSQR \mid bQR \mid aR$
 $R \rightarrow PSQR \mid \varepsilon$

结果 $S \rightarrow cSQR \mid bQR \mid aR$
 $P \rightarrow SPS \mid cS \mid b$
 $Q \rightarrow SP \mid c$
 $R \rightarrow PSQR \mid \varepsilon$

◇ 提取左公因子规则

– 对形如

$$P \rightarrow \alpha\beta \mid \alpha\gamma$$

的一对产生式，可用如下三个产生式替换：

$$P \rightarrow \alpha Q$$

$$Q \rightarrow \beta \mid \gamma$$

其中Q为新增加的未出现过的非终结符

◇ 提取左公因子规则

- 一般含有左公因子的产生式形如

$$P \rightarrow \alpha\beta_1 \mid \alpha\beta_2 \mid \dots \mid \alpha\beta_m \\ \mid \gamma_1 \mid \gamma_2 \mid \dots \mid \gamma_n$$

其中，每个 γ 不以 α 开头.提取左公共因子，
产生式改写成：

$$P \rightarrow \alpha Q \mid \gamma_1 \mid \gamma_2 \mid \dots \mid \gamma_n \\ Q \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_m$$

✧ 提取左公因子举例

– 对文法 $G(S)$:

$$S \rightarrow \underline{\text{if}} \ C \ t \ S \mid \underline{\text{if}} \ C \ t \ S \ e \ S$$

$$C \rightarrow b$$

提取左公因子后，可改写为文法 $G'(S)$:

$$S \rightarrow \underline{\text{if}} \ C \ t \ S \ A$$

$$A \rightarrow e \ S \mid \varepsilon$$

$$C \rightarrow b$$

☆ 举例：许多文法在消除左递归和提取左公因子后可以变换为LL(1)文法

可验证如下文法 $G[E]$ 是LL(1)文法：

$$(1) \quad E \rightarrow TE' \qquad (2) \quad E' \rightarrow + TE'$$

$$(3) \quad E' \rightarrow \varepsilon \qquad (4) \quad T \rightarrow FT'$$

$$(5) \quad T' \rightarrow * FT' \qquad (6) \quad T' \rightarrow \varepsilon$$

$$(7) \quad F \rightarrow (E) \qquad (8) \quad F \rightarrow a$$

☆ 举例：不含左递归和左公因子的文法
不一定是LL(1)文法

$$\begin{aligned} S &\rightarrow \underline{\text{if}} \ C \ t \ S \\ &\quad | \ \underline{\text{if}} \ C \ t \ S \ e \ S \\ C &\rightarrow b \end{aligned}$$

提取左公因子后：

$$\begin{aligned} S &\rightarrow \underline{\text{if}} \ C \ t \ S \ A \\ A &\rightarrow e \ S \mid \varepsilon \\ C &\rightarrow b \end{aligned}$$

First集	Follow集
$\underline{\text{if}} \ C \ t \ S \ A : \{\underline{\text{if}}\}$	$S: \{\#, e\}$
$eS: \{e\}$	$A: \{\#, e\}$
$b \quad \{b\}$	$C: \{t\}$

$$M[A, e] = \{A \rightarrow e \ S, \ A \rightarrow \varepsilon\}$$

☆ 问题探讨

某些非LL(1)的文法也可采用LL(1)分析方法

$$\begin{aligned} S &\rightarrow \underline{\text{if}} \ C \ t \ S \\ &\quad | \ \underline{\text{if}} \ C \ t \ S \ e \ S \end{aligned}$$

$$C \rightarrow b$$

提取左公因子后:

$$\begin{aligned} S &\rightarrow \underline{\text{if}} \ C \ t \ S \ A \\ A &\rightarrow e \ S \quad | \quad \varepsilon \end{aligned}$$

$$M[A,e] = \{ \underline{A \rightarrow e \ S}, \\ A \rightarrow \varepsilon \}$$

优先使用

◇ 错误处理的原则

- 尽可能准确指出错误位置和错误属性
- 尽可能进行校正

◇ 预测分析中的出错处理

- 递归下降LL(1)分析中的出错处理
介绍一种短语层错误恢复技术
- 表驱动LL(1)分析中的出错处理
介绍一种简单的应急错误恢复技术

☆ 表驱动LL(1)分析中的错误处理

— 出错报告 (error reporting)

- 栈顶的终结符与当前输入符不匹配
- 非终结符 A 于栈顶, 面临的输入符为 a , 但分析表 M 的 $M[A, a]$ 为空

☆ 表驱动LL(1)分析中的错误处理

- 简单的应急恢复 (*panic-mode error recovery*)
跳过输入串中的一些符号直至遇到同步符号
(*synchronizing token*) 为止

同步符号的选择:

- 把 $\text{Follow}(A)$ 中的所有符号作为 A 的同步符号, 跳过输入串中的一些符号直至遇到这些“同步符号”, 把 A 从栈中弹出, 可使分析继续
- 把 $\text{First}(A)$ 中的符号加到 A 的同步符号集, 当 $\text{First}(A)$ 中的符号在输入中出现时, 可根据 A 恢复分析

◇ 递归下降分析程序的错误处理

— 短语层恢复可采取的流程

在进入某个语法单位时，检查当前符号是否属于进入该语法单位需要的符号集合 $BeginSym$ 。若不属于，则报错，并滤去补救的符号集合 S 外的所有符号

在语法单位分析结束时，检查当前符号是否属于离开该语法单位时需要的符号集合 **EndSym**。若不属于，则报错，并滤去补救的符号集合 **S** 外的所有符号



预测分析中的出错处理

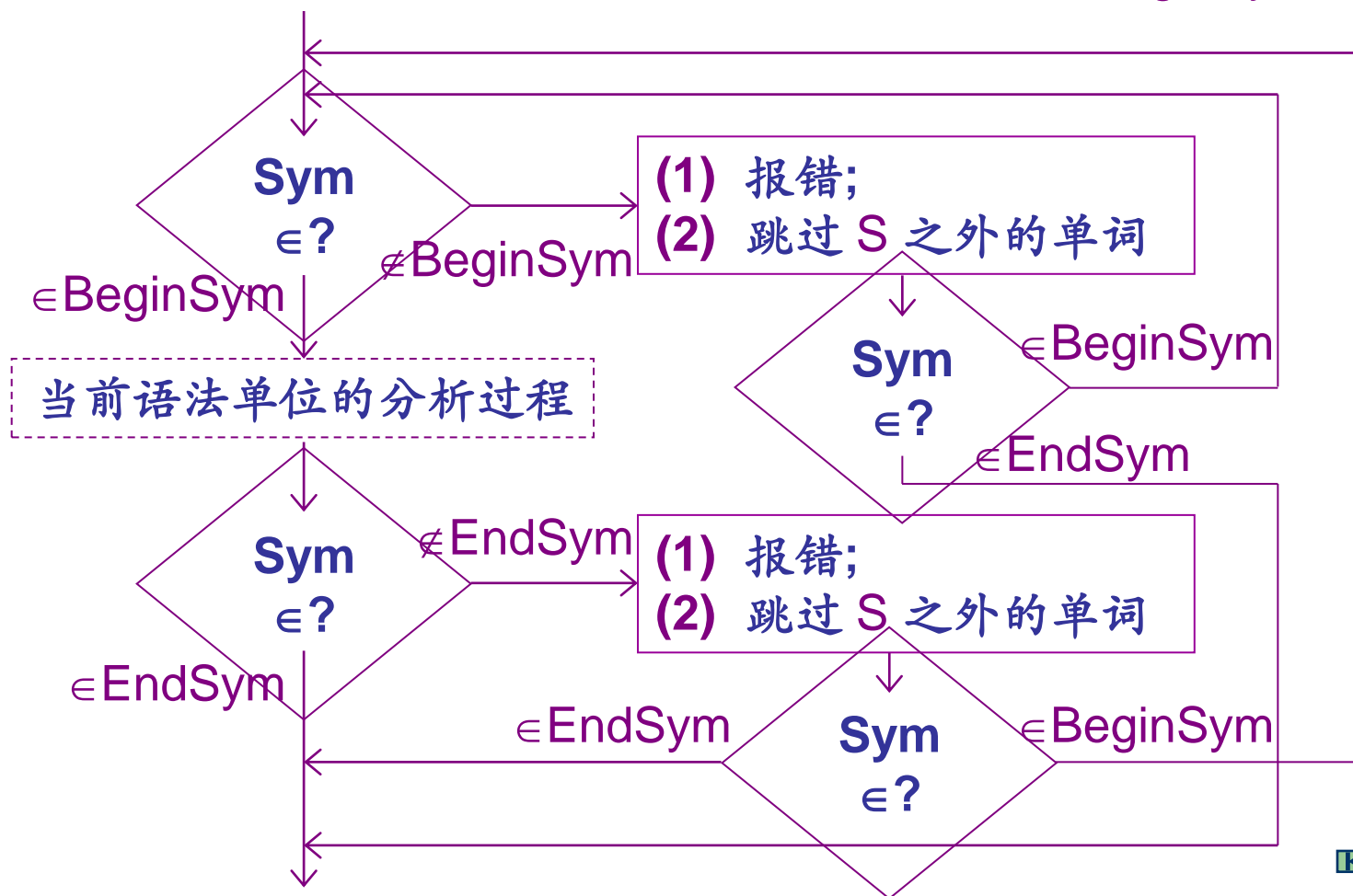
◇ 递归下降分析程序的错误处理

Sym 为正扫描的符号

S 为补救的符号集合

— 短语层恢复可采取的流程

$S = \text{BeginSym} \cup \text{EndSym}$



预测分析中的出错处理



清华大学

《编译原理》

◇ 递归下降分析程序的错误处理

— 短语层恢复举例

$$B \rightarrow [A] \mid (A)$$
$$A \rightarrow a$$

```
procedure ParseB ( EndSym )
{
  if ( lookahead  $\notin$  { '[', '(' } ) {
    报错; 跳过 S 之外的单词;    /* S = { '[', '(' }  $\cup$  EndSym */
  }
  while ( lookahead  $\in$  { '[', '(' } ) {
    if ( lookahead == '[' )
      { MatchToken('['); ParseA ( EndSym  $\cup$  { ']' } ); MatchToken(']'); }
    else { MatchToken('('); ParseA ( EndSym  $\cup$  { ')' } ); MatchToken(')'); }
    if ( lookahead  $\notin$  EndSym ) {
      报错; 跳过 S 之外的单词;    /* S = { '[', '(' }  $\cup$  EndSym */
    }
  }
}
```

◇ 递归下降分析程序的错误处理

— 短语层恢复举例

$$\begin{array}{l} B \rightarrow [A] \mid (A) \\ A \rightarrow a \end{array}$$

```
procedure ParseA ( EndSym )
{
    if ( lookahead  $\notin$  { 'a' } ) {
        报错; 跳过 S 之外的单词;    /* S = { 'a' }  $\cup$  EndSym */
    }
    while ( lookahead  $\in$  { 'a' } ) {
        MatchToken ( 'a' );
        if ( lookahead  $\notin$  EndSym ) {
            报错; 跳过 S 之外的单词;    /* S = { 'a' }  $\cup$  EndSym */
        }
    }
}
```

◇ 递归下降分析程序的错误处理

— 短语层恢复举例

```
void MatchToken(int expected)
{
    if (lookahead != expected) //判别当前单词是否与
    {                          //期望的终结符匹配
        printf("syntax error \n");
    }
    else // 若匹配,消费掉当前单词并读入下一个
        lookahead = getToken(); //调用词法分析程序
}
```

注：与前面相比，这里的MatchToken函数报错后不退出系统

◇ LL(K) 文法

— 推广LL(1)文法

可以通过向前查看 k 个符号来唯一确定产生式，以便在自顶向下预测分析中对相应的非终结符进行展开。

✧ LL(k) 文法

— 一些重要的结论

- 给定 $k > 0$, 一个CFG是否为LL(k) 文法是可判定的
- 对于一个CFG, 是否存在 $k > 0$, 使得该文法是 LL(k) 文法, 是不可判定的
- 对于一个CFG, 是否存在一个与之等价的 LL(k) 文法 ($k > 0$), 是不可判定的
- 两个LL(k) 文法的语言是否相等是可判定的
- LL(k) 文法是无二义文法
- LL(k) 文法中不存在左递归的非终结符
- 给定 $k > 0$, 不含 ε 产生式的 LL(k) 文法的语言类真包含于不含 ε 产生式的 LL($k+1$) 文法的语言类

课后作业

参见网络学堂公告：“第一次书面作业”

◇ 非书面作业

理解讲稿中递归下降分析程序的错误处理技术（含短语层恢复技术）。

思考：在使用表驱动技术的自顶向下分析程序中如何实现短语层恢复？

That's all for today.

Thank You