编译原理

第四章 语法分析——自上而下分析

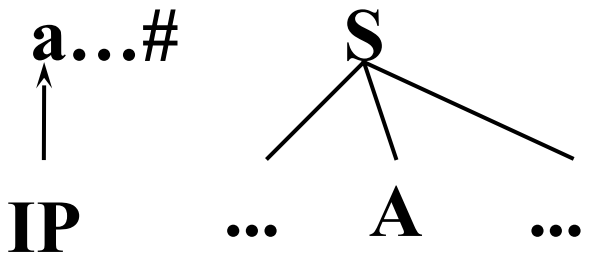
第四章 语法分析—自上而下分析

- 语法分析器的功能
- 自上而下分析面临的问题
- LL(1) 分析法
- 递归下降分析程序构造
- 预测分析程序

第四章 语法分析—自上而下分析

- 语法分析器的功能
- 自上而下分析面临的问题
- LL(1) 分析法
  - 消除文法的左递归
  - 克服回溯
- 递归下降分析程序构造
- 预测分析程序

4.5 预测分析程序



- LL(1) 分析法 ——假设要用非终结符 A 进行匹配，面临的输入符号为 a ， A 的所有产生式为

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

1. 若 a∈FIRST(α_i) ，则指派α_i 执行匹配任务；

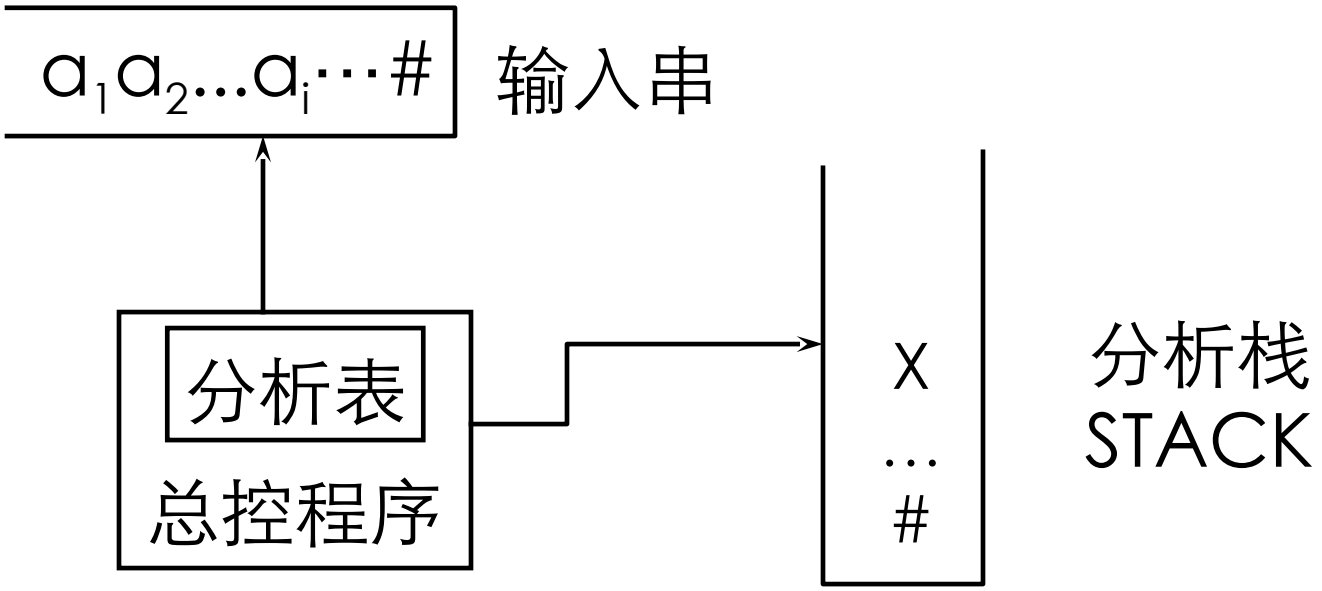
2. 若 a 不属于任何一个候选首符集，则：

   (1) 若ε属于某个 FIRST(α_i) 且 a∈FOLLOW(A) ，则让 A 与ε自动匹配。

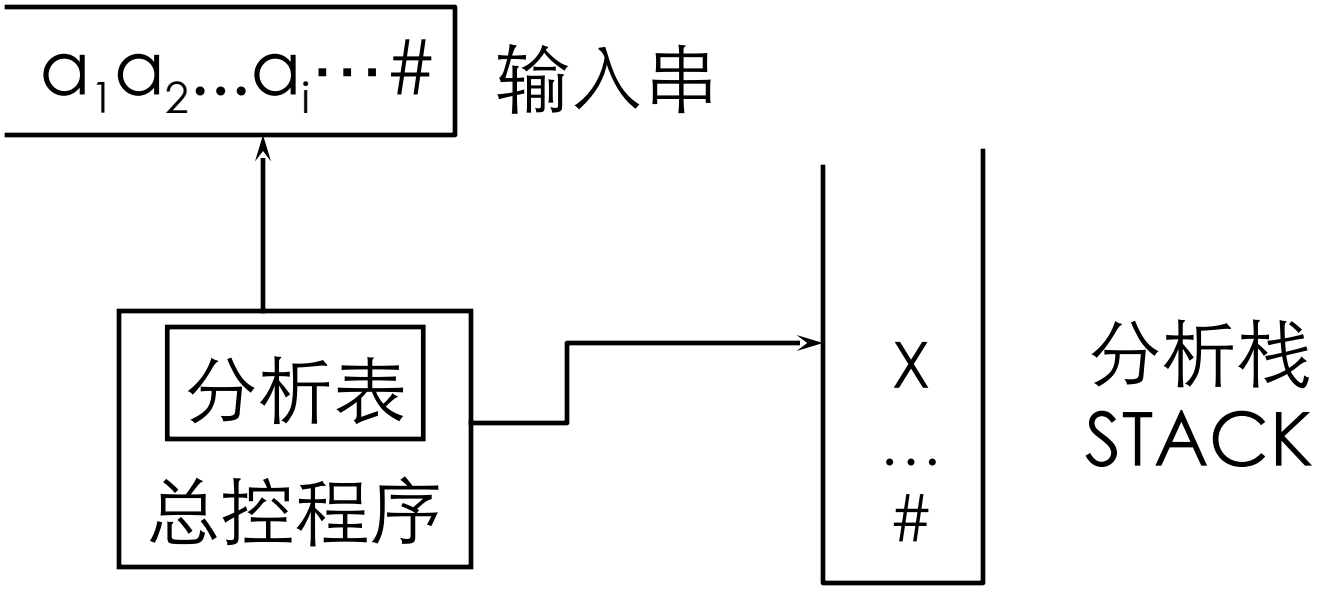
   (2) 否则， a 的出现是一种语法错误。

- 计算思维的典型方法
  - 知识与控制的分离
  - 自动化

预测分析程序构成

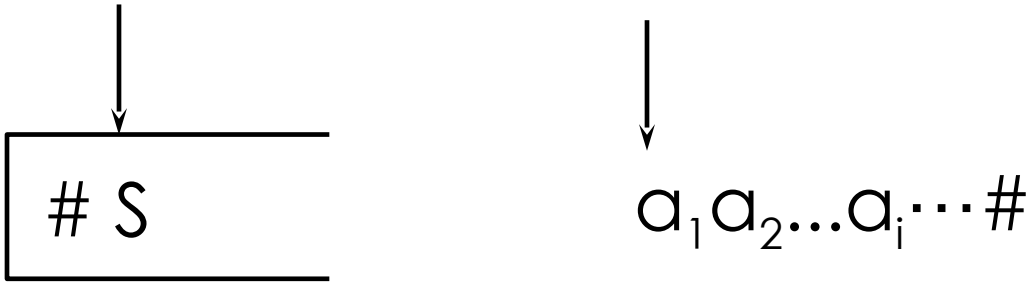


- 总控程序，根据现行栈顶符号和当前输入符号，执行动作
- 分析表 M[A，a] 矩阵，$A \in V_N$，$a \in V_T$ 是终结符或‘#’
- 分析栈 STACK 用于存放文法符号



预测分析程序的工作图

分析开始时：

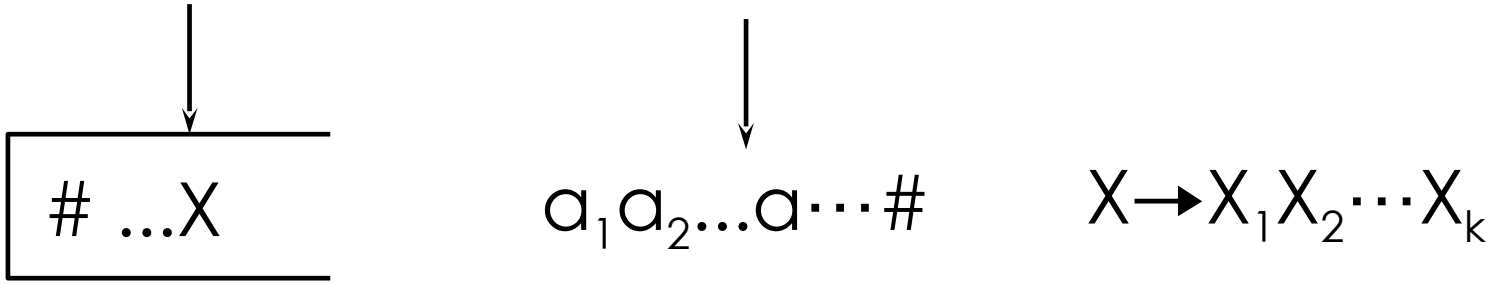


- 总控程序根据现行栈顶符号 X 和当前输入符号 a，执行下列三种动作之一：

1. 若 X = a = ‘#’，则宣布分析成功，停止分析。

2. 若 X = a ≠ ‘#’，则把 X 从 STACK 栈顶逐出，让 a 指向下一个输入符号。

匹配成功



3. 若 X 是一个非终结符，则查看分析表 M 。

推导

- 若 M[X ， a] 中存放着关于 X 的一个产生式，把 X 逐出 STACK 栈顶，把产生式的右部符号串按反序一一推进 STACK 栈 (若右部符号为ε，则意味不推什么东西进栈) 。在把产生式的右部符号推进栈的同时应做这个产生式相应的语义动作。
- 若 M[X ， a] 中存放着“出错标志”，则调用出错诊察程序 ERROR 。

预测分析程序的总控程序

BEGIN

首先把‘#’然后把文法开始符号推进 STACK 栈;

把第一个输入符号读进 a ;

FLAG:=TRUE;

WHILE FLAG DO

BEGIN

把 STACK 栈顶符号上托出去并放在 X 中;

IF $X \in V_T$ THEN

IF X= a THEN 把下一输入符号读进 a

ELSE ERROR

匹配成功

分析成功

ELSE IF X=‘#’ THEN

IF X=a THEN FLAG:=FALSE

ELSE ERROR

ELSE IF M[X,a]={X→$X_1X_2\cdots X_k$}THEN

推导

把 $X_k,X_{k-1},\cdots,X_1$ ——推进 STACK 栈

/* 若 $X_1X_2\cdots X_k=\varepsilon$，不推什么进栈 */

ELSE ERROR

END OF WHILE;

STOP /* 分析成功，过程完毕 */

END

- 例 4.6　对于文法 G(E)

  E→TE′

  E′→+TE′ | ε

  T→FT′

  T′→*FT′ | ε

  F→(E) | i

输入串为 $i_1*i_2+i_3$，利用分析表进行预测分析

	i	+	*	(	)	#
E	E→TE′			E→TE′		
E′		E′→+TE′			E′→ε	E′→ε
T	T→FT′			T→FT′		
T′		T′→ε	T′→*FT′		T′→ε	T′→ε
F	F→i			F→ (E)		

步骤	符号栈	输入串	所用产生式
0	#E	$i_1*i_2+i_3$#	
1	#E′T	$i_1*i_2+i_3$#	E→TE′
2	#E′T′F	$i_1*i_2+i_3$#	T→FT′
3	#E′T′i	$i_1*i_2+i_3$#	F→i

	i	+	*	(	)	#
E	E→TE′			E→TE′		
E′		E′→+TE′			E′→ε	E′→ε
T	T→FT′			T→FT′		
T′		T′→ε	T′→*FT′		T′→ε	T′→ε
F	F→i			F→ (E)		

步骤	符号栈	输入串	所用产生式
3	#E′T′i	$i_1*i_2+i_3$#	F→i
4	#E′T′	$*i_2+i_3$#	
5	#E′T′F*	$*i_2+i_3$#	T′→*FT′
6	#E′T′F	i_2+i_3#	
7	#E′T′i	i_2+i_3#	F→i

	i	+	*	(	)	#
E	E→TE′			E→TE′		
E′		E′→+TE′			E′→ε	E′→ε
T	T→FT′			T→FT′		
T′		T′→ε	T′→*FT′		T′→ε	T′→ε
F	F→i			F→ (E)		

步骤	符号栈	输入串	所用产生
7	#E′T′i	i_2+i_3#	F→i
8	#E′T′+i_3#		
9	#E′	+i_3#	T′→ε
10	#E′T+	+i_3#	E′→+TE′
11	#E′T	i_3#	

	i	+	*	(	)	#
E	E→TE′			E→TE′		
E′		E′→+TE′			E′→ε	E′→ε
T	T→FT′			T→FT′		
T′		T′→ε	T′→*FT′		T′→ε	T′→ε
F	F→i			F→ (E)		

步骤	符号栈	输入串	所用产生
11	#E′T	i_3#	
12	#E′T′F	i_3#	T→FT′
13	#E′T′i	i_3#	F→i
14	#E′T′#		
15	#E′	#	T′→ε
16	#	#	E′→ε

	i	+	*	(	)	#
E	E→TE′			E→TE′		
E′		E′→+TE′			E′→ε	E′→ε
T	T→FT′			T→FT′		
T′		T′→ε	T′→*FT′		T′→ε	T′→ε
F	F→i			F→ (E)		

分析表 M[A ， α] 的构造

- 构造 FIRST(α) 和 FOLLOW(A)
- 构造分析表 M[A ， α]

- 计算思维的典型方法
  - 知识与控制的分离
  - **自动化**

如何构造 FIRST 和 FOLLOW 集合

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

$$FOLLOW(A) = \{a \mid S \overset{*}{\Rightarrow} ...Aa..., a \in V_T\}$$

构造 FIRST(α)

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

- $\alpha = X$ ， $X \in V_T \cup V_N$
- $\alpha = X_1X_2 \cdots X_n$ ， $X_i \in V_T \cup V_N$

构造 FIRST(α)

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

- $\alpha = X$ ， $X \in V_T \cup V_N$
- $\alpha = X_1X_2\cdots X_n$ ， $X_i \in V_T \cup V_N$

构造每个文法符号的 FIRST 集合

- 对每一文法符号 $X \in V_T \cup V_N$ 构造 FIRST(X)

  连续使用下面的规则，直至每个集合 FIRST 不再增大为止：

  1. 若 $X \in V_T$，则 FIRST(X) = {X}。
  2. 若 $X \in V_N$，且有产生式 $X \rightarrow a\cdots$，则把 a 加入到 FIRST(X) 中；若 $X \rightarrow \varepsilon$ 也是一条产生式，则把ε也加到 FIRST(X) 中。

构造每个文法符号的 FIRST 集合

3.

- 若 $X \rightarrow Y\cdots$ 是一个产生式且 $Y \in V_N$，则把 FIRST(Y) 中的所有非ε - 元素都加到 FIRST(X) 中；
- 若 $X \rightarrow Y_1Y_2\cdots Y_k$ 是一个产生式，Y_1，…，Y_{i-1} 都是非终结符，
  - 对于任何 j，$1 \le j \le i-1$，$FIRST(Y_j)$ 都含有ε (即 $Y_1\cdots Y_{i-1}\varepsilon$)，则把 $FIRST(Y_i)$ 中的所有非ε - 元素都加到 FIRST(X) 中
  - 若所有的 $FIRST(Y_j)$ 均含有ε，$j = 1, 2, \cdots, k$，则把ε加到 FIRST(X) 中。

构造 FIRST(α)

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

- $\alpha = X$ ， $X \in V_T \cup V_N$
- $\alpha = X_1X_2\cdots X_n$ ， $X_i \in V_T \cup V_N$

构造任何符号串的 FIRST 集合

- 对文法 G 的任何符号串$\alpha = X_1X_2\cdots X_n$ 构造集合 FIRST(α)

  1. 置 FIRST(α) = FIRST(X_1)\{ε} ;

  2. 若对任何 $1\le j\le i-1$ ，$\varepsilon\in$ FIRST(X_j) ，则把 FIRST(X_i)\{ε} 加至 FIRST(α) 中；特别是，若所有的 FIRST(X_j) 均含有ε， $1\le j\le n$ ，则把ε也加至 FIRST(α) 中。显然，若$\alpha=\varepsilon$则 FIRST(α) = {ε} 。

构造 FOLLOW(A)

$$FOLLOW(A) = \{a \mid S \overset{*}{\Rightarrow} \ldots Aa \ldots, a \in V_T\}$$

构造每个非终结符的 FOLLOW 集合

- 对于文法 G 的每个非终结符 A 构造 FOLLOW(A) 的办法是，连续使用下面的规则，直至每个 FOLLOW 不再增大为止：

  1. 对于文法的开始符号 S，置 # 于 FOLLOW(S) 中；
  2. 若 $A \rightarrow \alpha B \beta$ 是一个产生式，则把 $FIRST(\beta) \setminus \{\varepsilon\}$ 加至 FOLLOW(B) 中；
  3. 若 $A \rightarrow \alpha B$ 是一个产生式，或 $A \rightarrow \alpha B \beta$ 是一个产生式而 $\beta \overset{*}{\Rightarrow} \varepsilon$（即 $\varepsilon \in FIRST(\beta)$），则把 FOLLOW(A) 加至 FOLLOW(B) 中。

- 例 4.6　对于文法 G(E)

  E→TE′
  E′→+TE′ | ε
  T→FT′
  T′→*FT′ | ε
  F→(E) | i

构造每个非终结符的 FIRST 和 FOLLOW 集合

FIRST(E) ={(,i}	FOLLOW(E) ={),#}
FIRST(E′)={+, ε}	FOLLOW(E′)={),#}
FIRST(T) ={(,i}	FOLLOW(T) ={+,),#}
FIRST(T′)={*, ε}	FOLLOW(T′)={+,),#}
FIRST(F) ={(,i}	FOLLOW(F) ={*,+,),#}

分析表 M[A ， α] 的构造

- 构造 FIRST(α) 和 FOLLOW(A)
- 构造分析表 M[A ， α]

分析表 M[A， a] 的构造

- 在每个非终结符 A 及其任意候选α都构造出 FIRST(α) 和 FOLLOW(A) 的基础上
- 构造 G 的分析表 M[A， a]， 确定每个产生式 A→α 在表中的位置

1. 对文法 G 的每个产生式 A→α 执行第 2 步和第 3 步;
2. 对每个终结符 a ∈FIRST(α)，把 A→α 加至 M[A， a] 中;
3. 若ε∈ FIRST(α)，则对任何 b∈FOLLOW(A) 把 A→α 加至 M[A， b] 中。
4. 把所有无定义的 M[A， a] 标上“出错标志”。

- 例 4.6　对于文法 G(E

E→TE′
E′→+TE′ | ε
T→FT′
T′→*FT′ | ε
F→(E) | i

2. 对每个终结符 $a \in FIRST(\alpha)$，把 $A \to \alpha$ 加至 $M[A, a]$ 中；

3. 若 $\varepsilon \in FIRST(\alpha)$，则对任何 $b \in FOLLOW(A)$ 把 $A \to \alpha$ 加至 $M[A, b]$ 中

构造每个非终结符的 FIRST 和 FOLLOW 集合：

FIRST(E) ={(,i}
FIRST(E′)={+, ε}
FIRST(T) ={(,i}
FIRST(T′)={*, ε}
FIRST(F) ={(,i}

FOLLOW(E) ={),#}
FOLLOW(E′)={),#}
FOLLOW(T) ={+,),#}
FOLLOW(T′)={+,),#}
FOLLOW(F) ={*,+,),#}

	i	+	*	(	)	#
E	E→TE′			E→TE′		
E′		E′→+TE′			E′→ε	E′→ε
T	T→FT′			T→FT′		
T′		T′→ε	T′→*FT′		T′→ε	T′→ε
F	F→i			F→ (E)		

LL(1) 文法与二义性

- 如果 G 是左递归或二义的，那么， M 至少含有一个多重定义入口。因此，消除左递归和提取左因子将有助于获得无多重定义的分析表 M 。
- 可以证明，一个文法 G 的预测分析表 M 不含多重定义入口，当且仅当该文法为 LL(1) 的。
- LL(1) 文法不是二义的。

G(S):

$S \rightarrow iCtS \mid iCtSeS \mid a$

$C \rightarrow b$

提取左因子之后，改写成：

G(S):

$S \rightarrow iCtSS' \mid a$

$S' \rightarrow eS \mid \varepsilon$

$C \rightarrow b$

最近匹配原则

	a	b	e	i	t	#
S	S→a			S→iCtSS′		
S′			S′→ε S′→eS			S′→ε
C		C→b		F→ (E)		

小结

- 预测分析程序的结构
- 预测分析方法
- 预测分析表的构造
  - 消除左递归，消除回溯
  - 计算 FIRST 、 FOLLOW 集合
  - 构造预测分析表

第四章 语法分析——自上而下分析

- 自上而下分析面临的问题
  - 文法的左递归性
  - 回溯
- 构造不带回溯的自上而下分析算法
  - 消除文法的左递归的方法
  - 提取左公共因子，克服回溯
- LL(1) 文法的条件
  - FIRST 、 FOLLOW 集合
- LL(1) 分析法
  - 递归下降分析程序
  - 预测分析程序

作业

- P81—1 ， 2
- P82—3(选作 2 个)