编译原理

第一章 引论

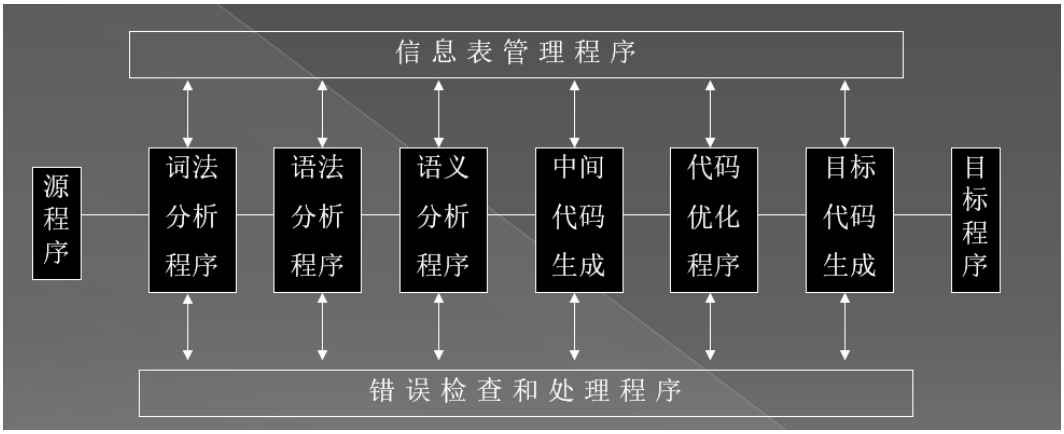
1. 翻译程序的分类：

   汇编程序：汇编语言源程序→机器语言目标程序

   解释程序：边解释边执行，保存中间结果

   编译程序：高级语言源程序→目标语言（机器语言/汇编语言）程序

2. 编译程序组成词法分析程序、语法分析程序、语义分析程序、中间代码生成、代码优化程序、目标代码生成、错误检查和处理程序、信息表管理程序



3. 编译程序的分遍（趟程）

   概念：从头到尾扫描源程序并做有关加工处理。

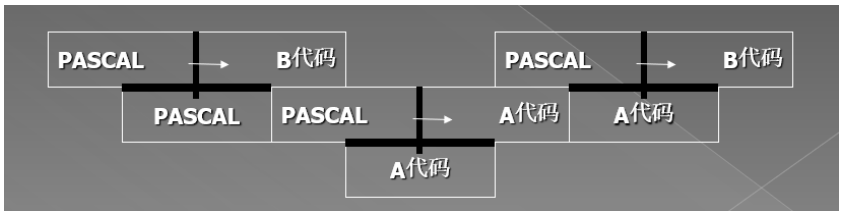
   决定因素：计算机存储容量大小，编译程序功能强弱，源语言繁简，目标程序优化程度，设计和实现编译程序时使用工具的先进程度，参加人员多少和素质

   适用情况：当源语言较繁，编译程序功能很强，目标程序优化程度较高且计算机存储容量较小时，采用多遍扫描方式。

4. 系统程序设计语言：能够编写编译程序或其他系统软件的高级语言。

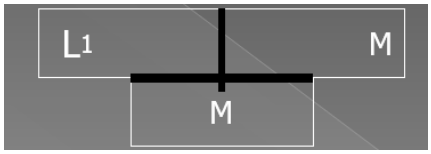
5. 自编译：一种高级语言与之对应的编译程序也能直接用该语言本身写出来。

6. 交叉编译：一个 A 机器上编译程序能产生 B 机器的目标代码。

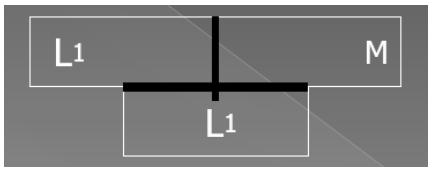


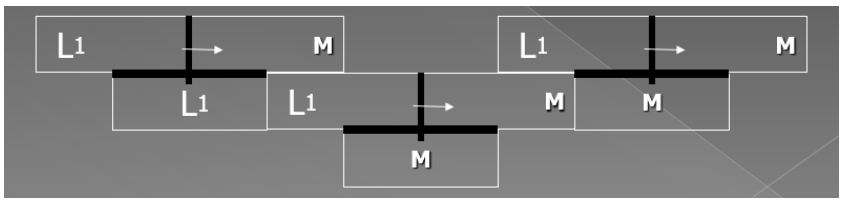
7. 自展技术：由一个功能较小的编译程序，一级一级扩充而变成一个功能较强的编译程序。

(1) 用机器语言 M 建立 L1（语言核心部分）

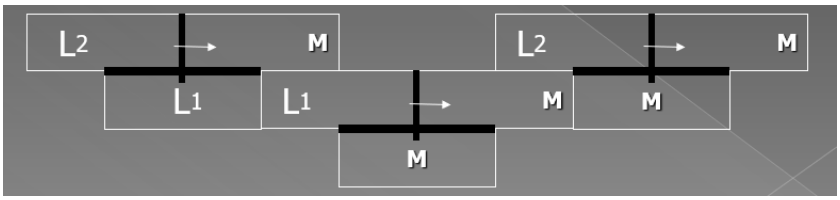


(2) 用 L1 重写 L1，通过原来机器语言编译程序验证

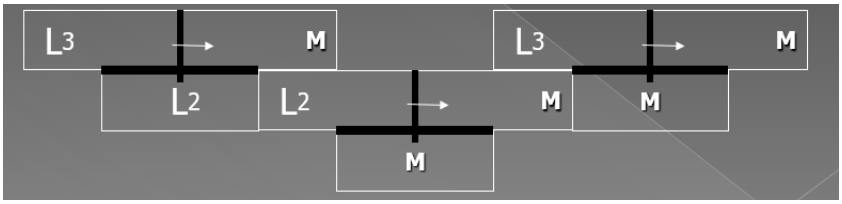




(3) 把语言功能从 L1 扩展到 L2



(4) 把语言功能从 L2 扩展到 L3



(5) 把语言功能扩展到 L

8. 移植：

(1) 综合几种型号计算机抽象出一个通用的汇编语言

(2) 利用交叉编译方法

第二章 语言的基本知识

1. 巴科斯范式 BNF：为了在 ALGOL60 报告中来描述 ALGOL 语言
2. 终结符/非终结符

> 例：S::=0S1, S::=01
>
> 非终结符 V_N={S}，终结符 V_T={0,1}，字汇表 V={S, 0, 1}

3. 文法是规则的有穷集合，用四元组 (V_N, V_T, P, Z) 形式定义

   P：产生式集，$Z \in V_N$ 为 G 开始符号（识别符）

4. 直接推导 $\Rightarrow$ ：仅使用一次规则，长度为 1

   推导 $\Rightarrow+$ ：至少使用一次规则，长度≥1

   广义推导 $\Rightarrow*$ ：经过 0 步或若干步的推导，长度≥0

5. 句型、句子、语言

   如果符号串 x 由识别符 Z 推导（长度≥1）而得，称 x 为 G 的一个句型。

   如果 x 仅由终结符组成，则称句型 x 为该文法的一个句子。

   由某文法所产生的一切句子的集合称为由该文法所确定的语言。

6. 递归规则：在规则左部和右部具有相同的非终结符规则（P78 改写法消除左递归）

   递归文法：至少含有一个递归非终结符的文法

   判断：如果一个语言是无穷的，则描述该语言的文法必定是递归的。

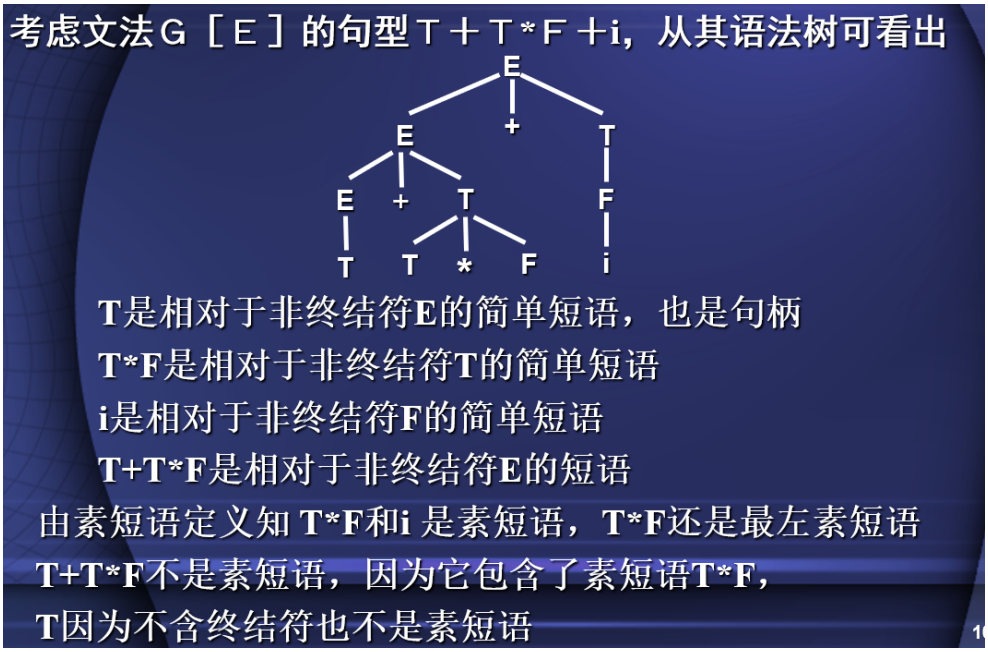
7. 如果一个语言是无穷的，则描述该语言的文法必定是递归的。
8. 短语：$Z \Rightarrow *xUy$, $U \in V_N$ 且 $U \Rightarrow +u$, $u \in V^+$

   简单短语：$Z \Rightarrow *xUy$, $U \in V_N$ 且 $U \Rightarrow u$

   句柄：句型最左边的简单短语（语法树最下层最左侧）

素短语：至少包含一个**终结**符号，且除自身外，不再包含其他的素短语

最左素短语：句型中最左边的素短语



9. 最右推导：规范推导，最左规约：规范规约；两者互为逆过程

10. 语法树

11. 文法的二义性：一个文法中的某句型对应两棵不同的语法树（一个文法中的某句子对应两个不同的最左推导或最右推导）

12. 文法和语言的分类

    0 型文法（短语结构文法），对应**图灵机**(TM)

    a::=b

    其中 $a \in V^+$, 且至少含有一个非终结符, $b \in V^*$

    1 型文法（上下文有关/敏感文法），对应**线性界限自动机**(LBA)

    aAb::=awb

    其中 $a,b \in V^*$, $A \in V_N$, $w \in V^+$

    2 型文法（上下文无关文法），对应**下推自动机**(PDA)

    A::=w

    $A \in V_N$, $w \in V^*$

    3 型文法（正规文法），对应**有穷状态自动机**(FA)

A::=aB 或 A::=a （右线性）　　A::=Ba 或 A::=a （左线性）

A, B∈V_N, a∈V_T

文法的判断：

3 型：左边有且仅有一个非终结符；右边格式确定

2 型：左边有且仅有一个非终结符；右边字符个数有限

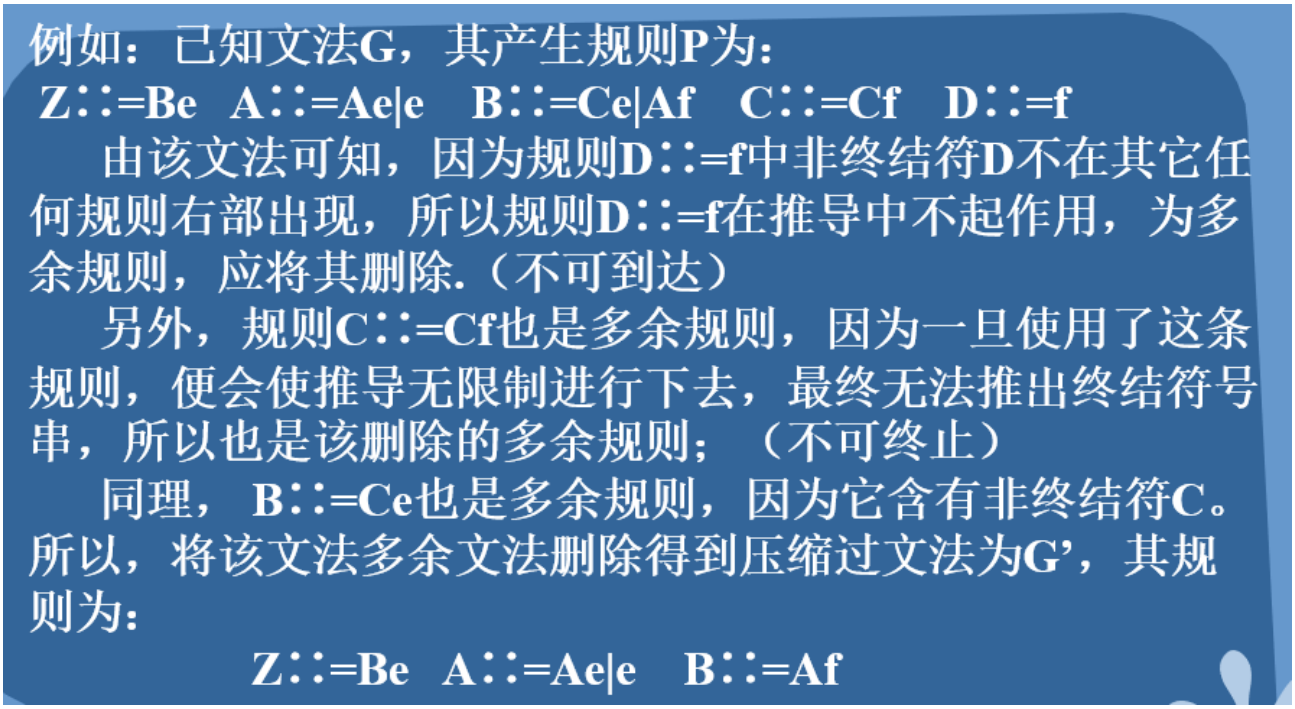
1 型：左边至少有一个非终结符；右边字符个数有限

13. 压缩过文法

文法中不含有形如 A::=A 的规则

每一个非终结符必须在某句型中出现（不可到达）

非终结符 A 必须推出终结符串 t（不可终止）



第三章 词法分析

1. 输入：符号串
2. 词法分析的主要任务是识别单词
3. 单词：具有独立含义的最小语法单位

4. 扫描缓冲区：当预处理子程序在处理出一串确定长度的输入字符时，要将其字符串装入另一个输入缓冲区中，这个缓冲区称为扫描缓冲区。
5. 超前搜索：仅向前读取字符，判别该字符是什么，不做别的处理工作。当判明情况后，再回过头来处理已读过的字符。
6. **由正规文法构造状态转换图**

(1) 由左线性文法构造状态转换图

U::=Ba　　$[B] \xrightarrow{a} [U]$

U::=a　　$[S] \xrightarrow{a} [U]$　　（箭头指向左边，S 初态）

(2) 由右线性文法构造状态转换图

U::=aB　　$[U] \xrightarrow{a} [B]$

U::=a　　$[U] \xrightarrow{a} [Q]$　　（箭头指向右边，Q 终态）

7. **词法分析程序的构造（设计题）**

画出状态转换图之后进行遍历。

https://blog.csdn.net/wangyang1354/article/details/17290033

8. 确定有穷自动机(DFA)的五元组

M = (K, V_T, M, S, Z)

K：状态有穷的非空集合，K 中每一个元素是一个状态

V_T：有穷输入字母表

M：K×V_T 到 K 的单值映射

S：开始状态

Z：终止状态集合

9. 非确定的有穷自动机(NFA)

与 DFA 的区别：

NFA 开始状态有多个

NFA 是多集映射

10. 构造正规表达式的正规集 P74Q19

L((a|b)*(aa|bb)(a|b)*) = {a,b}*{aa,bb}{a,b}*

11. 由正规文法构造相应的正规式 P74Q18 (I)

右线性文法 X=aX+b 通解：X=a*b

左线性文法 X=Xa+b 通解：X=ba*

12. 由正规式构造确定有穷自动机 DFA（子集法）

(1) 由正规式构造转换系统

(2) 写出状态转换集、状态转移矩阵

(3) 划分终态集、非终态集

(4) 构造 DFA

第四章 语法分析

1. 输入：单词
2. LL(1)分析法、递归下降分析法——自顶向下分析法

   算符优先分析法、LR 分析法——自底向上分析法（判断）
3. 改写法消除文法左递归（简答）

   $A ::= Aa_1|Aa_2|\ldots|Aa_n|b_1|b_2|\ldots|b_n$

   改写为： $A ::= b_1A'|b_2A'|\ldots|b_nA'$

   $A' ::= a_1A'|a_2A'|\ldots|a_nA'|\varepsilon$
4. LL(1)分析法

   (1) 求 FIRST 集

   (2) 求 FOLLOW 集

   a) 对于文法的开始符号 S，令#∈FOLLOW(S)

   b) 若文法中有形如 A::=αBβ 的规则，且 β≠ε，则将 FIRST(β)中一切非 ε 符号加

进 FOLLOW(B)中

c) 若文法中有形如 A::=αB 或 A::=αBβ, β⇒∗ε 的规则，则 FOLLOW(A)中全部终结符均属于 FOLLOW(B)

求 FOLLOW(X)时，在文法中找 X，(A::=X)如果 X 后面**没有**字符，则加入 **FOLLOW**(A)，如果 X **有**字符，加入字符的 **FISRT** 集（如果无法推出空串）或者 A 的 FOLLOW 集（如果可以推出空串）

(3) 判断是否为 LL(1)文法

G 的任意两个规则 A::=α|β 满足下面条件：

a) FIRST(α)∩FIRST(β)=Φ

b) α 和 β 中最多只有一个可能推出空串

c) 如果 β⇒∗ε，那么 α 推出任何串不会以 FOLLOW(A)中的终结符开始，即 **FIRST(α)∩FOLLOW(A)=Φ**

(4) 构造 LL(1)分析表

规则 A→X

a) 对 FIRST(α)中每一终结符 a，置 M[A, a] = A→X

b) 若 ε∈FIRST(α)，则对属于 FOLLOW(A)中的每一个符号 b（b 为终结符或#），置 M[A, b] = A→ε

c) 把 M 中所有不能按规则 a, b 定义的均置为出错

(5) 分析符号串

TIP: 倒序进栈

5. 简单优先分析法

(1) 关系 L：L＝{(U, S) | U→S…, S∈(V_T ∪ V_N)}

关系 L 的意思是：由文法规则左部非终结符与右部首符号组成的有序偶集合。

设文法G［A］
A::=Af | B, B::=DdC | De, C::=e, D::=Bf
由L定义求得
L＝{(A，A)，(A，B)，(B，D)，(C，e)，(D，B)}

(2) 关系 R：R＝{(U,S) | U → …S, S∈(V_T ∪ V_N)}

关系 R 的意思是：由文法规则左部符号与右部尾符号组成的有序偶集合。

(3) Warshall 算法（已知 L 求 L^+）

```
for i in range(n):
    for j in range(n):
        if A[j, i]=1:
            for k in range(n):
                A[j, k] = A[j, k] + A[i, k]
```

(4) 文法优先关系

S1>S2　　S1 是句柄的一部分（恰好是尾），但 S2 不是

S1=S2　　S1 和 S2 两者都在句柄之中

S1<S2　　S2 在句柄中（恰好是头），而 S1 不在句柄中

(5) 简单优先文法

在字汇表中，任意两个符号之间至多只有一种优先关系成立；

任意两条规则都没有相同的右部。

(6) 简单文法优先关系矩阵

$<$ = (=)(L^+)
$>$ = $(R^+)^T$ = L^*

(7) 简单优先文法的分析算法

G[Z]:　Z::=bMb　M::=(L|a　L::=Ma) 优先关系矩阵：

	Z	b	M	L	a	(	)
Z							
b			≐		⋖	⋖	
M		≐			≐		
L		⋗			⋗		
a		⋗			⋗		≐
(			⋖	≐	⋖	⋖	
)		⋗			⋗		

分析符号串 b(aa)b：

步骤	符号栈	关系	输入串	规则
1	#	⋖	b(aa)b#	
2	#b	⋖	(aa)b#	
3	#b(	⋖	aa)b#	
4	#b(a	⋗	a)b#	
5	#b(M	≐	a)b#	M::=a
6	#b(Ma	≐	)b#	
7	#b(Ma)	⋗	b#	
8	#b(L	⋗	b#	L::=Ma)
9	#bM	≐	b#	M::=(L
10	#bMb	⋗	#	
11	#Z	⋗	#	Z::=bMb

6. 算符优先文法

(1) 算符优先关系

A ·< B　　A→aB, B⇒+b 或 B⇒+Cb

A ≐ B　　A→ab, 或 A→aBb

A >· B　　A→Bb, B⇒+a 或 B⇒+aC

(2) 算符文法（OG）

不含有形如 U::=…AB…的规则（A, B∈V_N）的文法

(3) 算法优先文法（OPG）

一个算符文法的任意两个终结符号对(a, b)之间**至多**只满足≐，>·，·<三个关系之一

(4) 自底向上分析符号串

文法 G[E]:　　E::=E+T|T　　T::=T*F|F　　F::=(E)|i

	+	*	(	)	i
+	>·	·<	·<	>·	·<
*	>·	>·	·<	>·	·<
(	·<	·<	·<	=	·<
)	>·	>·		>·	
i	>·	>·		>·	

自底向上分析符号串i*(i+i)

符号栈S	关系	输入串	最左素短语
#	·<	i*(i+i)#	
#i	>·	*(i+i)#	i
#V	·<	*(i+i)#	
#V *	·<	(i+i)#	
# V *(	·<	i+i)#	
#V *(i	>·	+i)#	i
# V *(V	·<	+i)#	
# V *(V+	·<	i)#	
# V *(V+i	>·	)#	i
# V *(V+V	>·	)#	V+V
# V *(V	=	)#	
# V *(V)	>·	#	(V)
# V *V	>·	#	V*V
#V		#	

V：代表非终结符，

关系大小比较时将 i, V 剔除

7. 对于简单优先文法，若字汇表中包含 n 个符号，构造优先函数后，所需存储空间大小由 n×n 减少到 2n

8. LR(K)分析

(1) LR(K)分析器是这样一种分析程序，它总是从左到右扫描输入串，并按自底向上进行规范规约，在这种分析过程中，它至多只向前查看 K 个输入符号就能确定当前的动作是移进还是规约。

(2) 从逻辑上来说，一个 LR 分析器包括两部分：一个总控程序和一张分析表。一般来说，所有 LR 分析器总控程序是一样的，只是分析表各不相同。

9. LR(0), SLR(1), LR(0), LALR 文法类型的判断

LR(0)文法中不存在移进-规约冲突（S::=E •，E::E • +）

SLR(1)：存在冲突的 I_i 中，{移进项目 V_T}∩FOLLOW(归约项目左部 V_N)=Φ

LALR：同心集合并后没有“归约-归约”冲突

不满足以上：LR(1)

10. LR(0)分析表的构造

(1) 对于项目集 I_i 中有形如 $A::=\beta_1 \cdot X\beta_2$ 项目，且 $GO(I_i, X)=I_j$，若 $X=a \in V_T$,则置 $ACTION[i, a]=S_j$,若 $X \in V_N$,则置 $GOTO[i, X]=j$

(2) 若归约项目 A::＝β·属于 I_i，设 A::=β 是文法第 j 个规则，则对任意终结符 a 和句子右界符#，均置 ACTION[i, a 或#]＝r_j，表示按文法第 j 条规则将符号栈顶符号串 β 归约为 A。

(3) 若接受项目 S’::=S·属于 I_i，则置 ACTION[i, #]=acc

(4) 分析表中，凡不能用前 3 个规则填入信息的空白格位置上，均表示出错。

11. LR(1)分析表的构造

(1) LR(1)项目集规范族

i. I 的任何项目都属于 CLOSURE(I)

ii. 若项目[A::=α·Bβ]属于 CLOSURE(I), B::=η 是一个规则，那么对 FIRST(βa)中

的每个终结符 b, 则有形如[B::= • η, b]的所有项目也属于 CLOSURE(I)

(2) LR(1)分析表的构造 【GO(I, X)=CLOSURE(J)】

i. 若项目[A::= α·aβ, b]属于 I_i, 且 $GO(I_i, a)=I_j$, $a \in V_T$, 则置 ACTION[i, a]=S_j

ii. 若项目[A::=α·, a]属于 I_i, 设 A::=α 是文法的第 j 个规则,则置 ACTION[i, a]=r_j

iii. 若项目[S'::=S·]属于 I_i, 则置 ACTION[i, #]为 acc

iv. 若 $GO(I_i, A)=I_j$, $A \in V_N$, 则置 GOTO[i, A]=j

12. LALR 分析表的构造

(1) 构造文法 G 的 LR(1)的项目集族 $C=\{I_0, I_1, \ldots, I_n\}$

(2) 把全部同心集合并，记为 $C'=\{J_0, J_1, \ldots, J_n\}$, 其中含有项目[S'::=·S, #]的 J_i 为分析表的初态

(3) 从 C'构造 ACTION 表：

i. 若[A::=α·aβ,b]∈Ji, 且 GO(Ji, a)=Ji, $a \in V_T$, 则置 ACTION[i, a]=S_j

ii. 若[A::=α·]∈Ji, 则置 ACTION[i, a]=r_i

iii. 若[S'::=S·, #]∈J_i，则置 ACTION[i, #]＝acc

(4) 构造 GOTO 表

若 $GO(J_i, A)=J_j$, $A \in V_N$，则置 GOTO[i, A]＝j

第五章 语法制导翻译技术

1. 语法制导翻译方法就是以语法分析为主导的语义处理，在对源程序进行语法分析的过程中嵌入语义动作。

2. 中间语言的表示

(1) 逆波兰表示（后缀表示）

x:=5
　　x5:=
x:=a*b-c/d
　　xab*cd/-:=

(2) 用后缀式来表示条件语句

if E then S_1 else S_2

$E'p_1$ JEZ $S'_1 p_2$ JUMP S'_2

E', S'_1, S'_2 分别为 E, S_1, S_2 的后缀式

p_1 表示 S'_2 在数组 POST 的起始位置

p_2 表示 S'_2 之后那个符号位置

(3) 三元式(OP, ARG1, ARG2)

x:=-b*(c+d)

(1) (-, b,)
(2) (+, c, d)
(3) (*, (1), (2))
(4) (:=, x, (3))

(4) 四元式(OP, ARG1, ARG2, RESULT)

x:=-b*(c+d)

(1) (-, b, , T_1)
(2) (+, c, d, T_2)
(3) (*, T_1, T_2, T_3)
(4) (:=, T_3, ,x)