## 第一章 绪论

编译程序与解释程序的基本区别:编译程序要比对等价的目标代码 江偏器:书汇编语言程序翻译成机器可執行的目标程序

痛泽器: 把程序翻译为等价的机器语言程序

词法分析器用于识别单词将号,诸法分析器用来分析语法单位

痛译程序:将振程序字壁地转换成机器语言或汇编语言, 妖后再处理, 執行的對解译程序 【①词话的析 +③语法的精 +③语义的精与中间代码生成 + ④代码优化 + 【⑤目标代码生成 + ⑥出磷处理 + ①有气表管理

1.前端:词法分析、语法、语义、中间代码生成、代 码优化器;后端:代码优化器、目标代码生成器 2.词法分析:扫描源程序,转换为 TOKEN,找语法错 误,管理符号表。

3.语法分析:组词成句, 找错, 分析树

4.语义分析:获取信息,找错,静态绑定

5.用 T 形图表示语言翻译: 从源语言翻译为目标语 言,而翻译器本身是表示语言

6.高级语言程序:编写、预处理、编译、汇编、链接、 运行

## 第二章 文法(E->T+F 这种就是文法)

1.一些定义

①字母表:非空又穷集合(如{a,b,c})

②字母表上字符串: ε 空串; 若 x 是 Σ上的字符串, 而  $a \in \Sigma$ ,则 xa 是  $\Sigma$  上的符号串。y 是  $\Sigma$  上的符号 串,当且仅当它由(1)和(2)导出。

③乘积 $\Sigma$ 1 $\Sigma$ 2 = {ab|a $\in$   $\Sigma$ 1, b $\in$   $\Sigma$ 2}

④幂 $\Sigma$ 0 = { $\epsilon$ },  $\Sigma$ n =  $\Sigma$ n-1 $\Sigma$ 

⑤正闭包+,科林闭包\*(多个 ε)

⑥ $\forall$ L $\subseteq$   $\Sigma$ \*, L 称为 $\Sigma$ 的一个语言  $\forall$ x ∈ L, x 叫做 L的一个句子。

2.文法 G 四元组。V 非终结符、T 终结符、P 产生式 集合、S开始符号

语法范畴  $A L(A) = \{ w | w \in T^* \mathbb{1} A \stackrel{*}{\Rightarrow} w \}$ 语言 (Language)  $L(G) = \{w | w \in T^* \perp S \stackrel{*}{\Rightarrow} w \}$ 句子 (Sentence)  $\forall w \in L(G)$ 

句型 $\forall \alpha$ ∈(V ∪ T)\*如果 S\*⇒α,则称 α 是 G 产生 的一个句型。

### 对一个文法的, 而所产生的语言上(的)就是该文法的开始符号分推出的所有符号串 0型文法:短语结构 Hd→BeP, IPI≥Id1.要并有W左的攻等的 1型文法:上下文有关

Yd→PEP, IPINIAI, dEV. 2型文法:上下文元系 ク型文法:正则文法(正规文法) 图括: {A→ W 左徐性 A→ Bω A→W 右分性 A→WB

## 3.BNF 范式:

- $-\beta(\alpha_1|\alpha_2|...|\alpha_n) = \beta\alpha_1|\beta\alpha_2|...|\beta\alpha_n$
- {α<sub>1</sub>|α<sub>2</sub>|...|α<sub>n</sub>}": 出现 l 次到 u 次
- 终极符:基本符号集  $- [\alpha] = \alpha |\epsilon$

## 4.最右推导对应最左归约(id 到 E)

5.•短语: 子树的结果是相对于子树根的短语•直接短 语: 只有父子两代的子树的结果•句柄: 一个句型的 分析树中最左的、只有父子两代的子树的结果•一个 句型的短语数量等于非独子的中间结点数量

## 第三章 词法分析

1. <u>正则表达式(RE)(正规式、正规表达式)</u>:正则语 言另一种表达,例如标识符: I(IId)\*

2.FA 的语法图



## 第四章 自顶向下语法分析(LL(1))

判断是否为LL(1): 对A-OolB @ FIRSTININEIRSTIBI = 0

②着d 為 E, FIRST(d) ∩ FOLLOW(A)=Ø ③若β▲ E, FIRST(B) N FOLLOW(A)=Ø

## 第五章 自底向上语法分析

1.素短语: 能一步推出, 含终结符

2.算符文法(OG)G中不存在具有相邻非终结符的产生 式(A  $\rightarrow \alpha$  BC  $\beta$ )

3.算符优先文法(OPG)要求a≡b,a>b,a<b仅一个成立。 4.LR(K)分析法: 从左到右输入(L),最左归约(R),超前读 入 k 个符号。有归约-归约 or 移近-归约冲突,该文 法就不成立,SLR 同理,LR(1)无冲突

- LR (0) 项目:产生式的右部的某个位置有「.」

移进(Shift)项目; S→.bB 待约(Reducing)项目; S→b.BB 归约(Reduce)项目: S→aB. 接受(Accept)项目: S'→S.

缺点:必须是 OPG,效率低,有查不到的语法错 误,希望通过文法产生式进行派生或归约

## 5.DFA 就是画的大图,公式如下

 $M=(C, V \cup T, go, I_0, C)$ 

 $I_0$ =CLOSURE( $\{S' \rightarrow .S\}$ )

 $C=\{I_0\} \cup \{I|\exists J \in C, X \in V \cup T, I=g_0(J,X)\}$ 

# 6.活前缀就是.之前的东西 <u>7.CLOSEURE({DFA 每个方块里第一个式子}) = {这</u> 个方块里所有的式子}

CLOSURE(I)=IU  $\cup \{B \rightarrow .\gamma | A \rightarrow \alpha.B\beta \in C LOSURE(I), B \rightarrow \gamma \in P\}$ 

J:=I; repeat caSLR(I)会析表的构造,保含a∈FOLLOW(A) J=JU{B→,γ|A→α.Bβ∈J&B→γ∈I。 同解共都分评实 until J不再扩大 LR(1)及其搜索符/展望符 LALR(1)——同 企集

## 第六章 语义分析

1.属性文法: A=(G,C,F)G:上下文无关文法; C:属 性有穷集合; F:关于属性的计算规则。是语法制 导的文法。语法指导是实验三的表。(抽象)

□ 終结符使用单词的属性 (id.val) ❖保留字: if, begin, function,..... ◆ 常 教: 40.12, 232, 80, "TCP/IP" ◆标识符id: id.entry, id.type id.val ∞<mark>语法变量根据实际需要设定属性</mark> ◆表达式E: E.type, E.val ◆变量X: X.addr, X.type, X.val

2.属性分类: ①综合属性:如 E.val/type

代码优化: 通过程序的等价变换, 获得纳行速度快, 计用空间分的程序

代码优化的主要目标:使得《编译程序的产生高质量的目标代码和制定某种联略,以使所产生的目标代码 能较合理地利用且标计量机的资源

1.提高运行速度, 节省存储空间

❖ 允许程序段之间的覆盖(覆盖可能性分析)

优点①编译时决定存储位置②访问效率高

用途①子程序代码段②全局变量

缺点如下动态分配的功能所示

层次单元法——动态数组问题

栈式存储分配——递归调用问题

传值

传名

以技調用语句 电说明语句  $id(E_1,E_2,\ldots,E_n) \qquad \text{Procedure } id(X_1,X_2,\ldots,X_n)$ 

{S.code := Elist.code ||gen('goto 'pc+Elist.num+1)|| for 队列q中的每一项 t do gen('param' t ) ||

gen('call' id.place','Elist.num) }

堆式存储分配——生存期超过调用者

1/22 41~62

4/17 0~16 6/10 0~9

4.参数传递

传值调用——

传名调用-

CR操名子程序

S→call id (Elist)

第九章 优化

引用调用——传地址

复制恢复——传值和地址

原始总存储需求=105个存储单元

思考: 如何设计分配算法?

区城

0~9

0~22 0~16

17~31

41~62

共需要63个存储单元

程序段

局部抗化(基本缺肉价化): 常数表达了计算, 删除公共子表达寸, 删除两代码, 交换语句次序

全局忧化之循环忧化:代码外提,强度削弱,消除归纳变量

或汇编代码。**面向具体机器,有效利用寄存器** (附属)固有属性 T=int②继承属性 L1.in:=L.in③S 5.表格管理:管理编译过程中的各种符号表,辅助完

成语法、语义检查,完成静态绑定,管理编译过程。 6.前端与机器无关,后端反之。好处:实现新语言 or 新机器,只需要前后端中的一个即可。

4. 目标代码生成:中间代码转换为机器上的指令代码

2.与机器无关的优化:常量合并、公共子表达式提取

3.有关:循环展开、向量化、访存优化、寄存器排布

# 第七章 中间代码生成

{...}来表示,最左派生。(具象)

1 变量说明的翻译:

在符号表中记录种别、类型、相对地址、作用于 等, 计算相对地址

属性:只含①(语法制导定义)④L属性:包含①②

3.翻译模式: 语义动作插入到产生式某个位置,

2.数组翻译

静态:编译时直接完成相应的分配工作 动态:生成代码,以便在运行时调用分配子 程序完成分配工作

## 第八章 运行管理

1.符号表管理: 关键字(保留字)表、层次表、标 识符表(过程表、变量表、标号表),常数表 作用:记录源程序中符号的属性,为编译提供信 息:地址、类型。建立表项(元组),以标识符为关

实现:二叉树、线性表、散列。优先考虑查找性 能。结构成员、函数参数、分程序结构为特殊问 题。

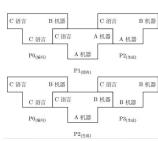
2.绑定:符号名和目标数据的地址绑定。运行时是 动态绑定,编译时是静态绑定

3.存储组织



数据静态分配(顺序分配、分层分配),分层:

1.A 机上有一个 C 语言编译器,是否可以利用此编译 器实现 B 机器上的 C 语言编译器?



解:1. 用 C 语言编 制 B 机器的编译 程序 PO(C → B)、 用 P1 编译 P0, 得 到 P2、用 P2 编译 PO, 得到 P3



利用本机原有,构建新的(本机编译器利用)

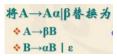


自展 (只用一个机器)

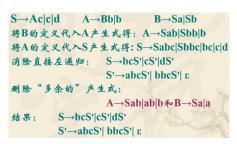


- 1. 用汇编语言实现一个 C子集的编译程序(P0-人)
- 2. 用汇编程序(P1)处理该程序,得到P2(P2:可直接运行) 3. 用C子集编制 C语言的编译程序(P3—人)
- 4. 用P2编译P3, 得到P4

2.消除直接左递归



3.消除间接左递归



简单来说,就是先把左部排序,B能推出S,放在第 一位, A 能推出 B, 放在第二位……然后有如下算法: for i=1 to n{

for j = 1 to i-1

Aj 代入每一个 Ai->Aj β

消除 Ai 的直接左递归

}简单来说,就是遍历 A1 到 An,对于 Ai,每次寻找 A1 到 Ai-1 中能代入到右部的,并代入。结束寻找之 后消除直接左递归。最后记得删除冗余。

4.提取公因子

$$\begin{split} & \approx A {\longrightarrow} \alpha \beta_1 |\alpha \beta_2| ... |\alpha \beta_n| |\gamma_1| |\gamma_2| ... |\gamma_m \\ & \stackrel{\bullet}{\leadsto} A {\longrightarrow} \alpha A' |\gamma_1| |\gamma_2| ... |\gamma_m \not\stackrel{\bullet}{\leadsto} A' {\longrightarrow} \beta_1 |\beta_2| ... |\beta_n \end{split}$$

## 5.FIRST 集算法



## 6.FOLLOW 算法

Algorithm 3 求 (FOLLOW(A))

- 1: procedure FOLLOW
- $FOLLOW(S) \leftarrow \#$
- if  $A \rightarrow \alpha B\beta$  then
- $FOLLOW(A) \leftarrow FOLLOW(A) \cup (FIRST(B) \{\epsilon\})$
- if  $A \to \alpha Bor \alpha B \beta$ ,  $\beta \stackrel{*}{\Rightarrow} \epsilon$  then
- $FOLLOW(B) \leftarrow FOLLOW(B) \cup FOLLOW(A)$
- end if
- 9: end procedure

## 第一条不需解释

第二条说白了就是,紧挨着 B 后面的是 A,而且 A 还能再推导出东西,那么B的FOLLOW添加A的FIRST 第三条说白了就是, A能推出 B, 那么 A的 FOLLOW 一定在 B FOLLOW 里

## 7. 画预测表

l→Tl	E' E' →+	TE'  ε T	→FT' T	'' →*FT'	ε F→(	E) id
	表达	达式文	法的	预测分	析表	
语法			终极	符号		
变量	id	+	*	(	)	#
E	→TE'			→TE'		
E'		→+TE*			→ε	→ε
T	→FT'	1		→FT'		
T'		→ε	→*FT'	MILES	→ε	-

语法变量	终极符号								
	id	+	*	(	)	#			
E	→TE'			→TE'	synch	synch			
E'		→+TE'			→ε	→ε			
T	→FT'	synch		→FT'	synch	synch			
T'		→ε	→*FT'		→ε	→ε			
F	→id	synch	synch	→(E)	synch	synch			

①求出每一个右部(无终结符)的 FIRST②填伊普西龙 横看,对照着能一步推出伊普西龙的符号的 FOLLOW 写。填产生式右部横看,对照着右部的 FIRST。synch 横着看,对照着不能一步推出伊普西龙的符号的

### FOLLOW 写

## 8.求 FIRSTOP LASTOP(注意顺序)

- \* if  $A \rightarrow a \cdots \in P$  or  $A \rightarrow Ba \cdots \in P$  $a \in FIRSTOP(A)$ \* if  $A \to \dots a \in P$  or  $A \to \dots aB \in P$
- $a \in LASTOP(A)$
- \* if  $A \to \dots B \in P$ , LASTOP $(A) = \text{LASTOP}(A) \cup \text{LASTOP}(B)$ 9.构造算符优先分析表
- $-A \rightarrow X_1 X_2 \dots X_n$ 
  - $* X_i X_{i+1} \in TT \Rightarrow X_i \equiv X_{i+1}$
  - \*  $X_i X_{i+1} X_{i+2} \in TVT \Rightarrow X_i \equiv X_{i+2}$

\* if  $A \to B \cdots \in P$ . FIRSTOP $(A) = \text{FIRSTOP}(A) \cup \text{FIRSTOP}(B)$ 

- \*  $X_i X_{i+1} \in TV \Rightarrow \forall a \in FIRSTOP(X_{i+1}), X_i \lessdot a$
- \*  $X_i X_{i+1} \in VT \Rightarrow \forall a \in LASTOP(X_{i+1}), X_i > a$

	+		*	1	(		id	#
+	3	>	1	K	1	7	₹.	3
	*	>	4	*	4	>	<	*
*	>	3	3	3	4	>	4	*
1	*	7	3	7	4.	3	4	2
(	*	*	4	*	4	=	4	
)	2	3	3	>		*		*
id	3	>	7	3		2		3
#	4	4	1	1	4	100	1	ace

	69	分析过程	/400	
		动作	栈	输入缓冲区
	1)	初始化	#	$id_1+id_2*id_3\#$
例 分析id+id*id =	2)	移进	#id <sub>1</sub>	+id2*id3#
, , , , , , , , , , , , , , , , , , ,	3)	炒约 E→id	#E	+id2*id3i
♦ # ≮ id ≯ + id* id #	4)	移进	#E+	id,*id,#
*# ≮ F + ≮ id ≯ * id # *	5)	移进	#E+ic	l <sub>2</sub> *id <sub>3</sub> #
*#≮ F+≮ F*≮ id#	6)	炒约 E→id	#E+I	E *id <sub>3</sub> #
*# < F + < F * < id > #	7)	移进	#E+1	E* id <sub>3</sub> #
*# < F + < F * F #		移进	#E+I	E*id <sub>3</sub> #
	9)	炒约 E→id	#E+I	E*E #
<pre>*# &lt; F + &lt; F * F &gt; #</pre>		归约 E→E*E		
<pre> # ≮ F + T ≯ #</pre>		归约 E→E+E		#
♦ # E #	_		#IL	#
♥ # L #	12)	接受		

填表的时候, 形如 E+的竖着看, 对应 LASTOP(E) 写大于; 形如+E 横着看, 对应 FIRSTOP(E)写小于。 等于就形如(E),或者两个终结符挨着 分析的时候, #在 id 前边, 那么就横着先找到#, 竖着找到 id,得到小于号; id 在+的前边,横着 先找到 id, 在找到+, 找到大鱼号

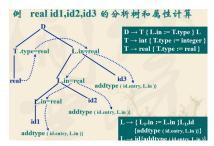
## 10.LR(0)分析表

约定:	状	动作表action			转移表goto	
sn:将符号a、状	态	a	b	#	S	В
态n压入栈	0	s3	s4		1	2
rn:用第n个产生	1			acc		
式进行归约	2	s3	s4			5
LR(0), SLR(1)	3	s3	s4			6
LR(0), SLR(1) , LR(1), LALR	4	r3	r3	r3		Jan Jan
将以不同的原则	5	r1	r1	r1		
构造这张分析表	6	r2	r2	r2		74

S的标号是I的下标,r的标号是产生式的标号(自 定义或题目给)



## 11.属性计算



12.赋值语句翻译

产生式	语义规则						
S→id:=E	S.code:=E.code  gen(id.place':='E.place)						
$E \rightarrow E_1 + E_2$	E.place:=newtemp;E.code := E <sub>1</sub> .code   E <sub>2</sub> .code    gen(E.place':='E <sub>1</sub> .place'+' E <sub>2</sub> .place)						
$E \rightarrow E_1 * E_2$	E.place:=newtemp;E.code := E <sub>1</sub> .code    E <sub>2</sub> .code    gen(E.place':='E <sub>1</sub> .place'*' E <sub>2</sub> .place)						
$E \rightarrow -E_1$	$\begin{split} & E.place:=newtemp;  E.code:=E_1.code \mid   \\ & gen(E.place':=''uminus' E_1.place) \cdot \frac{gen(E.place':=0.'E_1.place)}{gen(E.place':=0.'E_1.place)} \end{split}$						
$E \rightarrow (E_1)$	E.place:= $E_1$ .place; E.code := $E_1$ .code						
$E \rightarrow id$	E.place:= id.place; E.code := "						
E→ num	E.place:= num.val; E.code := "						

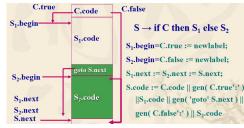
### 13.布尔表达式

$$\begin{split} E \rightarrow E_1 \text{ or } E_2 \quad & \{E.place := newtemp; \\ \quad & E.code := E_1.code \parallel E_2.code \parallel \\ \quad & gen(E.place':='E_1.place'or'E_2.place)\} \end{split}$$

### 14.while



### 15.if



	0型文法	1型文法	2型文法	3型文法	3型文法
	(PSG)	(CSG)	(CFG)	(RG)	(RG)
	S→aBC	S→aBC	E→E+E	I→a b	
	S→aSBC	S→aSBC	E→E*E	I→aT bT	
	CB→BC	CB→BC	E→(E)	T→a b	
	aB→d	aB→ab		TALL OF	
	bB→bb	bB→bb	E→id	T→1 2	H→a b
	bC→b	bC→bc	E→E-E	T→aT bT	H→Ha Hb
16.	cC→ cc	cC→ cc	E→E/E	T→1T 2T	H→H1 H2

# 9、自顶向下的语法分析方法中需要解决的主要问题?如何表示?

主要需要解决回溯与左绝归。回溯:匹配多个候选式无法快速匹配;左递归:推导过程 无休止。解决:提取公共左因子、消除直接及间接左递归。

## 10、将语法分析和词法分析分开的好处:

10、祝海体介列和闽本方列77市以交1 核块化、将编序过程分为不同的除良可以使得编译器的实现更加模块化和可维护。词法 分析和部法分析分别处理不同的任务,通过将它们分开,可以更容易地设计和实现每个 阶段的算注和数据结构。 简化处理:将源代码逐个字符进行分析会导致处理的复杂性增加,因为字符序列可以有

无限多的组合方式。将源代码分解为有限的词法单元集合后, 语法分析阶段可以更专注 于语法规则的分析和生成语法树。

丁语法规则的分析和互放语法例。 错误检测:将河法分析和语法分析分开可以使编译器更容易识别和报告源代码中的错误。河法分析器可以检测和报告语法错误,如拼写错误成率法字符,而语法分析器可以 检测和报告语法错误,如缺失的分号或不匹配的括号。

7、 什**么是自展? 什么是交叉编译?** 自展过程就是用低级语言先实现一个简单的编译器。然后用这个编译器的语言再去编写 一个更高级的编译器——这个新编译器是旧编译器的扩展——的过程。编译器的运行环 境与产生程序的运行环境不同的编译过程叫做交叉编译。

计算机执行用高级语言编写的程序有哪些途径? 其主要区别是什么? 解释和编译。解释不生成目标代码。

## 什么是语法制导翻译?

5、11 (本定程体例才翻译: 是指在语法规则的制导下,通过计算语义规则,完成对输入符号的翻译。由于使用属性 文法时把语法规则和语义规则分开,但在使用语法规则进行推导或归约的同时又使用这 些语义规则来指导翻译与最终产生目标代码,所以称为语法制导翻译。

6、 请彻**赛厢达高级程序设计语言参聚传递的常用方式** 1. 传值,计算实参并将其石值传递给被调用过程2. 传地址,调用过程将实参地址传递 给被调用过程3. 传信线果。将传值和传递此两种方式结合 4、传名,只有在被调用过 程中用到形参时才动态的建立起它与实参的联系

## 简要解释编译程序中的遗(趟)的含义。

及是对源程序或者源程序的中间结果从头到尾扫描一次,并作有关的加工处理,生成新 的中间结果和目标程序。通常。每通的工作由外存上获得的前一通的中间结果开始,完 成它所含的有关工作之后,再继续果记录于外存。既可以将几个不同阶段合为一遍,也 可以把一个阶段的工作分为若干遍。

2、何为"标识符"?何为"名字"?两者的区别是什么? 在程序设计语言中,标识符是一个最基本的概念,其定义为. 凡以字母开头的字母数字 在程序设计语言中, 标识符是一个最基本的概念, 其定义为, 凡以字母开头的字母数字 序列(有限个字符) 都是标识符。当给予某标识符以确切的含义时, 这个标识符就叫做 名字。程序语言中各种名字都是用标识符表示的, 只是标识符是一个没有意义的字符序 列, 而名字却有着确切的意义和属性。

# 简述为什么自顶向下的语法分析技术不能处理具有左递归的文法?

这是由于在自顷向下的语法分析技术中,要解决的问题是根据当前输入符号判断将栈顶 (最左)的非终结符号替换成哪条规则的右部,若文法具有左递归,则在分析过程中, 无法判断出替换的规则,造成无穷递归求解的过程。