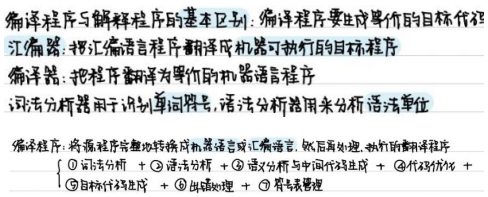
第一章 绪论



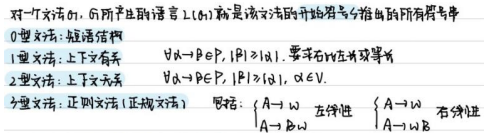
1.前端:词法分析、语法、语义、中间代码生成、代码优化器；后端:代码优化器、目标代码生成器
2.词法分析:扫描源程序，转换为 TOKEN，找语法错误，管理符号表。
3.语法分析:组词成句，找错，分析树
4.语义分析:获取信息,找错,静态绑定
5.用 T 形图表示语言翻译：从源语言翻译为目标语言，而翻译器本身是表示语言
6.高级语言程序：编写、预处理、编译、汇编、链接、运行

第二章 文法(E->T+F 这种就是文法)

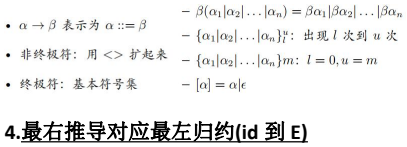
1.一些定义
①字母表:非空又穷集合(如{a,b,c})
②字母表上字符串: ε 空串；若 x 是 Σ上的字符串，而 a∈Σ，则 xa 是Σ上的符号串。y 是Σ上的符号串，当且仅当它由 (1) 和 (2) 导出。
③乘积Σ1Σ2 = {ab|a∈Σ1, b∈Σ2}
④幂Σ0 = {ε}， Σn = Σn−1Σ
⑤正闭包+，科林闭包*(多个 ε)
⑥∀L⊆Σ*，L 称为Σ的一个语言 ∀x ∈ L，x 叫做 L 的一个句子。
2.文法 G 四元组。V 非终结符、T 终结符、P 产生式集合、S 开始符号

$$\text{语法范畴 } A\ L(A)=\left\{w|w\in T^*\text{且}A\overset{*}{\Rightarrow}w\right\}$$
$$\text{语言 (Language) } L(G)=\left\{w|w\in T^*\text{且}S\overset{*}{\Rightarrow}w\right\}$$
$$\text{句子 (Sentence) } \forall w\in L(G)$$

句型∀α∈(V ∪ T)*如果 S*⇒ α，则称 α 是 G 产生的一个句型。



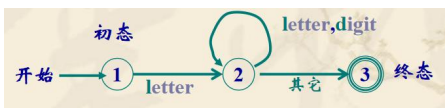
3.BNF 范式：



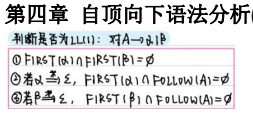
4.最右推导对应最左归约(id 到 E)
5.•短语：子树的结果是相对于子树根的短语•直接短语：只有父子两代的子树的结果•句柄：一个句型的分析树中最左的、只有父子两代的子树的结果•一个句型的短语数量等于非独子的中间结点数量

第三章 词法分析

1.**正则表达式(RE)（正规式、正规表达式）**：正则语言另一种表达,例如标识符：l(l|d)*
2.FA 的语法图



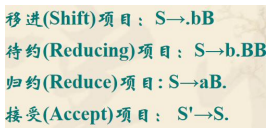
第四章 自顶向下语法分析(LL(1))



第五章 自底向上语法分析

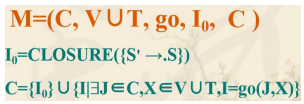
1.素短语：能一步推出，含终结符
2.算符文法(OG)G 中不存在具有相邻非终结符的产生式(A → α BC β)
3.算符优先文法(OPG)要求 a≡b,a>b,a<b 仅一个成立。
4.LR(K)分析法：从左到右输入(L),最左归约(R),超前读入 k 个符号。**有归约-归约 or 移近-归约冲突，该文法就不成立，SLR 同理，LR(1)无冲突**

– LR（0）项目：产生式的右部的某个位置有「.」

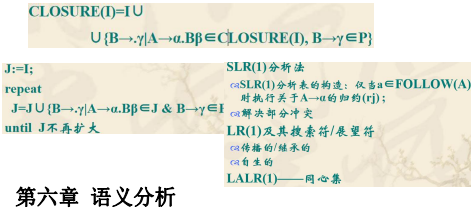


缺点：必须是 OPG，效率低，有查不到的语法错误，希望通过文法产生式进行派生或归约

5.**DFA 就是画的大图，公式如下**

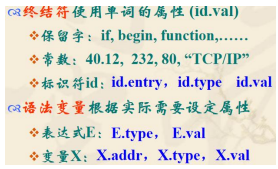


6.活前缀就是.之前的东西
7.CLOSEURE({DFA 每个方块里第一个式子}) = {这个方块里所有的式子}



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）



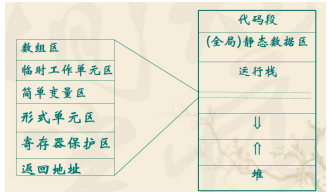
2.属性分类：①综合属性:如 **E.val/type**
(附属)固有属性 T=int②继承属性 L1.in:=**L.in**③S 属性:只含①(语法制导定义)④L 属性:包含①②
3.**翻译模式：语义动作插入到产生式某个位置**，{...}来表示，最左派生。（具象）

第七章 中间代码生成

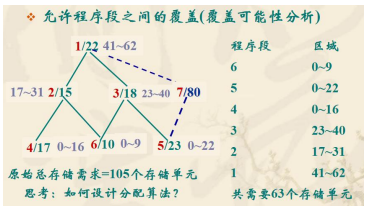
1.变量说明的翻译：
在符号表中记录种别、类型、相对地址、作用于等，计算相对地址
2.数组翻译

第八章 运行管理

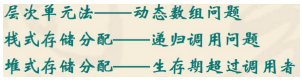
1.符号表管理：关键字(保留字)表、层次表、标识符表(过程表、变量表、标号表)，常数表
作用:记录源程序中符号的属性，为编译提供信息：地址、类型。建立表项(元组),以标识符为关键字。
实现:二叉树、线性表、散列。优先考虑查找性能。结构成员、函数参数、分程序结构为特殊问题。
2.绑定:符号名和目标数据的地址绑定。运行时是动态绑定，编译时是静态绑定
3.存储组织



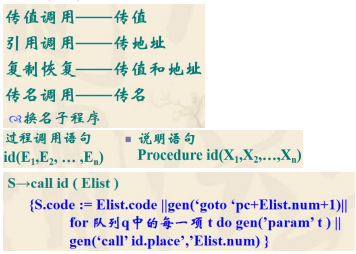
数据静态分配（顺序分配、分层分配），分层：



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

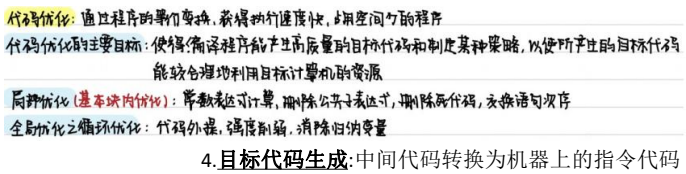


4.参数传递



第九章 优化

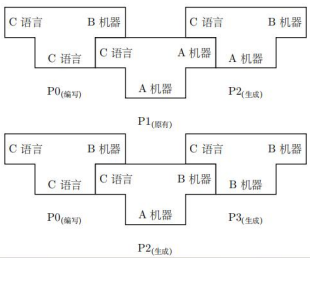
1.提高运行速度，节省存储空间
2.与机器无关的优化:常量合并、公共子表达式提取
3.有关:循环展开、向量化、访存优化、寄存器排布



4.**目标代码生成**:中间代码转换为机器上的指令代码或汇编代码。**面向具体机器，有效利用寄存器**
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。

题

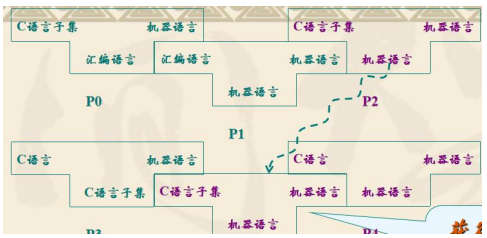
1.A 机上有一个 C 语言编译器，是否可以利用此编译器实现 B 机器上的 C 语言编译器？



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

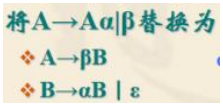
利用本机原有，构建新的（本机编译器利用）

自展（只用一个机器）

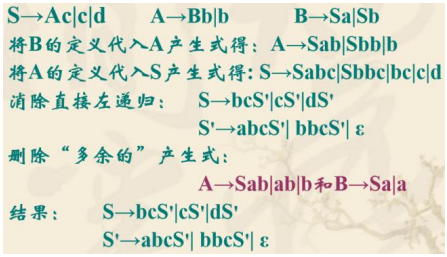


1. 用汇编语言实现一个C子集的编译程序(P0—A)
2. 用汇编程序(P1)处理该程序,得到P2(P2:可直接运行)
3. 用C子集编制C语言的编译程序(P3—A)
4. 用P2编译P3，得到P4

2.消除直接左递归



3.消除间接左递归



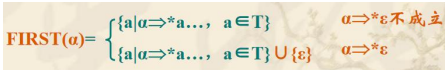
简单来说，就是先把左部排序，B 能推出 S，放在第一位，A 能推出 B，放在第二位……然后有如下算法：

```
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

5.FIRST 集算法



6.FOLLOW 算法

Algorithm 3 求 (FOLLOW(A))

```
1: procedure FOLLOW
2:     FOLLOW(S) ← #
3:     if A → αBβ then
4:         FOLLOW(A) ← FOLLOW(A) ∪ (FIRST(B) − {ε})
5:     end if
6:     if A → αB or αBβ, β ⇒* ε then
7:         FOLLOW(B) ← FOLLOW(B) ∪ FOLLOW(A)
8:     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

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）

- 初值
  - if $A \to a \cdots \in P$ or $A \to Ba \cdots \in P$
    - $a \in \text{FIRSTOP}(A)$
  - if $A \to \ldots a \in P$ or $A \to \ldots aB \in P$
    - $a \in \text{LASTOP}(A)$
- 迭代
  - if $A \to B \cdots \in P$, $\text{FIRSTOP}(A) = \text{FIRSTOP}(A) \cup \text{FIRSTOP}(B)$
  - if $A \to \ldots B \in P$, $\text{LASTOP}(A) = \text{LASTOP}(A) \cup \text{LASTOP}(B)$

9.构造算符优先分析表

$- A \to X_1 X_2 \ldots X_n$

$* \; X_i X_{i+1} \in TT \Rightarrow X_i \equiv X_{i+1}$

$* \; X_i X_{i+1} X_{i+2} \in TVT \Rightarrow X_i \equiv X_{i+2}$

$* \; X_i X_{i+1} \in TV \Rightarrow \forall a \in \text{FIRSTOP}(X_{i+1}), X_i \lessdot a$

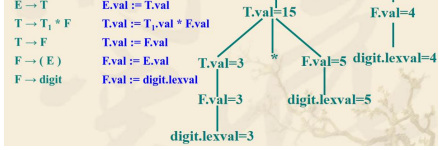
$* \; X_i X_{i+1} \in VT \Rightarrow \forall a \in \text{LASTOP}(X_{i+1}), X_i \gtrdot a$

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

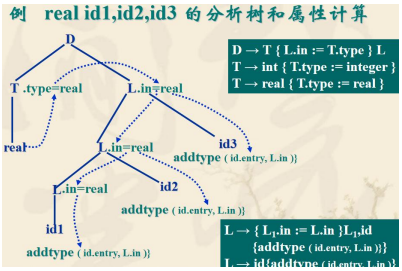
分析的时候，#在 id 前边，那么就横着先找到#，竖着找到 id，得到小于号；id 在+的前边，横着先找到 id，在找到+，找到大鱼号

10.LR(0)分析表

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）



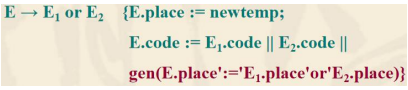
11.属性计算



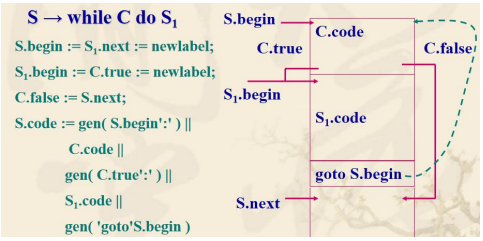
12.赋值语句翻译

产生式	语义规则
S→id:=E	S.code:=E.code||gen(id.place':='E.place)
$E \to E_1+E_2$	E.place:=newtemp;E.code := E_1.code|| E_2.code || gen(E.place':='E_1.place'+' E_2.place)
$E \to E_1*E_2$	E.place:=newtemp;E.code := E_1.code || E_2.code || gen(E.place':='E_1.place'*' E_2.place)
$E \to -E_1$	E.place:=newtemp; E.code := E_1.code || gen(E.place':=''uminus' E_1.place) gen(E.place':=0-'E_1.place)
$E \to (E_1)$	E.place:= E_1.place; E.code := E_1.code
E→ id	E.place:= id.place; E.code := ''
E→ num	E.place:= num.val; E.code := ''

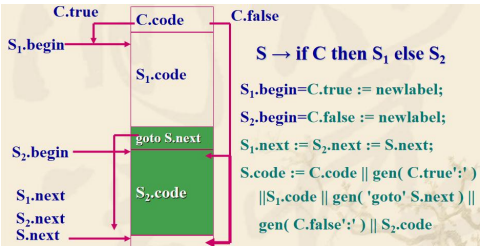
13.布尔表达式



14.while



15.if



16.

0型文法 (PSG)	1型文法 (CSG)	2型文法 (CFG)	3型文法 (RG)	3型文法 (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	E→id	T→1|2	H→a|b
bB→bb	bB→bb	E→E-E	T→aT|bT	H→Ha|Hb
bC→b	bC→bc	E→E/E	T→1T|2T	H→H1|H2
cC→ cc	cC→ cc			

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递归。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：

模块化：将编译过程分为不同的阶段可以使得编译器的实现更加模块化和可维护。词法分析和语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法和数据结构。

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。

错误检测：将词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。词法分析器可以检测和报告词法错误，如拼写错误或非法字符，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。

8、 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。

5、 什么是语法制导翻译？

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、 请简要阐述高级程序设计语言参数传递的常用方式

1、传值：计算实参并将其右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、传值结果：将传值和传地址两种方式结合 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

1、 简要解释编译程序中的遍（趟）的含义。

就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，每遍的工作由外存上获得的前一遍的中间结果开始，完成它所含的有关工作之后，再把结果记录于外存。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。

3、 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？

这是由于在自顶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。