编译原理

qhy

2017 年 9 月 9 日

目录

1 介绍

预期收货:

- 通过学习编译原理,写出更高效的代码
- 针对目标,自编写编译器

  对某个模型机的编译器进行设计

编译器和解释器的区别?
编译器是转换器,解释器是执行系统。

程序编译的过程主要分为两大阶段:

1. 查错
   - 词法分析
   - 语法分析
   - 语义分析
2. 综合(翻译)
   1. 产生中间代码进一步优化
   2. 目标代码生成

高级程序处理过程:初始源程序→预处理→源程序→编译→目标汇编→机器代码
注:生成机器代码的时候,并不是直接生成,而是先生成对应的汇编代码,再生成机器代码。
编译过程:每个阶段的输出作为下一个阶段的输入(即数据从一种形式转换成另一种形式)。
编译过程的每个阶段都包括两个相同的处理:表格管理和出错处理。
表格管理是保存编译过程每个阶段的数据和结果,出错处理则是对编译过程遇到的语法,词法等错误进行处理。

1.1 词法分析

主要分为两大步骤,扫描和分解
扫描为从左到右扫描
分解为以介符对语句进行分割(注:双引号内的介符不可划分)
最后结果使用一个二元组保存,格式为(种类,值)

1.2 语义分析

语义分析:不断构造语法树
如:类型不对,数组越界等
过程:单词符号串→语法分析→语法短语
识别规则:描述程序结果的规则,通常由递归规则表示。
输出:合法的语法树

1.3 语义分析

语义分析:包括静态语义和动态语义
常见的错误包括类型不匹配,数组越界等
输出:生成中间代码
结果使用四元式表示,格式为:(运算符,运算对象1,运算对象2,结果)

1.4 代码优化

代码优化:对代码进行优化化简

1.5 目标代码生成

目标代码生成:与目标机器紧紧相关,先生成对应的汇编代码,再生成机器代码

1.6 表格管理与出错处理

表格管理与出错处理:每个阶段都执行这一操作

1.7 编译程序划分

编译程序划分:分析与综合(翻译)两个阶段
按是否与目标机器相关:前端(无关)和后端(相关)

2 文法和语言

掌握自下而上和自上而下的分析方法。

程序设计的定义:语言是一个记号系统。
分析程序设计语言的两个阶段:
- 每个程序的构成规律
- 每个程序的含义

2.1 语法

语法:是一组规定,用它可以形成和产生一个合适的程序。
描述工具:文法
语法只可以判断结构是否合法。

2.2 语义

语义:
- 静态语义
- 动态语义

静态语义:一系列限定的规则,确定哪些合乎语法的程序是合适的。
动态语义:运行或动态链接时进行的判断。
描述工具:指称语义,操作语义
作用:检查类型匹配,变量作用域等。

2.3 文法

如何描述语句?

1. 生成方式(文法)
   语言中每个句子可以用严格定义的规则进行构造
2. 识别方式(自动机)
   用一个过程,经有限次计算后会停止回答"是" ,若属于句子,要么回答"否",要么永远持续下去

语言:可以是有穷的也可以是无穷的,我们要做的是,找出语言的有穷表示。
文法的作用:

1. 使用有穷的规则描述无穷的语言
2. 严格定义句子的结构,是判断句子结构是否合法的依据

方法:
::=表示定义一条规则
⇒表示应用这条规则,完成句子的变换(即由...推导出...的意思)

2.3.1 符号和符号串

字符表(符号集):由字母、数字和若干专用字符组成的非空有限集合。
符号串:字母表中的符号组成的任何有穷序列。
比如:a, aca是$A:\{a,b,c\}$的符号串。
符号串的长度:符号串含有符号的个数
符号串的运用:

- 连接
  定义:$x=$"ST",$y=$"aby"
  则$xy=$"$STaby$"
- 方幂
  $a^n=\underbrace{aa\cdots aa}_{\text{n个a}}$
- 集合的乘积
  定义:$A=\{a,b\}, B=\{0,1\}$
  则$AB=\{a0,a1,b0,b1\}$
  注:A,B是符号串的集合,并且AB的结果中A的符号串在前面
- 集合的方幂
  $A^2=AA$

2.3.2 闭包

闭包:Σ的闭包为Σ上所有元素组合的集合Σ^*。即

$$\Sigma^* = \Sigma^0 \bigcup \Sigma^1 \bigcup \Sigma^2 \cdots \tag{1}$$

正闭包:闭包去掉空集元素。即

$$\Sigma^+ = \Sigma^* - \{\phi\} \tag{2}$$

2.3.3 文法

产生式(规则):一组有序对(α, β) 表示$\alpha \to \beta$ 或$\alpha ::= \beta$
文法的定义:四元组(V_N, V_T, P, S)
其中,V_N表示非终结符 , V_T表示终结符 , P表示产生式集合,S表示文档其实符号。
注:

- S为文档的其实符号,从具体情况上看,它就是句子本身,只是还没有经过解析。
- $V_N \bigcup V_T = Wh$

例子:
文法$G = (V_N, V_T, P, S)$
$V_N = \{S\}, P = \{S \to 0S1, S \to 01\}, V_T = \{0, 1\}$
开始符号为 S

文法$G = (V_N, V_T, P, S)$
$V_N = \{$标识符,字母,数字$\}$
$V_T = \{a, b, \cdots, z, 0, 1, \cdots, 9\}$

$$P = \left\{ \begin{array}{l} < \text{标识符} > \to < \text{字母} >, \\ < \text{标识符} > \to < \text{标识符} >< \text{字母} >, \\ < \text{标识符} > \to < \text{标识符} >< \text{数字} >, \\ < \text{字母} > \to a, \\ \cdots, \\ z, < \text{字母} > \\ < \text{数字} > \to 0, \\ \cdots, \\ 9, < \text{数字} > \end{array} \right\}$$

$S =<$ 标识符 $>$

2.3.4 简化表示

简化表示:只使用产生式来表示文法,四元组的其他三元在产生式中表示。

- 第一条产生式的左部表示S
- 用大写字母或者尖括号包围表示非终结符集合
- 用小写字母表示终结字符集合

- 左部相同的产生式的多个,可以用|(或) 来简化表示。
  比如:$A \to \alpha, A \to \beta$可以简记为$A \to \alpha|\beta$
  注意:$A \to \alpha|\beta$表示的是两条产生式而不是一条,其中α, β为候选式。

2.4 归纳与推导

直接归纳与直接推导:
若$v \Rightarrow w$则称v直接推导w 或 w 直接归纳为 v

归纳与推导:使用了多条产生式,记为

$$S \overset{+}{\Rightarrow} \alpha \tag{3}$$

或

$$S \overset{*}{\Rightarrow} \alpha \tag{4}$$

注:如果使用了一条产生式,进行了多次⇒也为直接推导, 推导是使用了多条产生式。

2.5 句型、句子、语言

句型与句子的定义:
对于文法$G[S]$,若$S \overset{*}{\Rightarrow} x$,则称x为文法G的句型。
若x仅由终结符组成,则称x为文法G的句子
注:

- 句型可以表示多个句子,句子是句型的一个情况之一。句子也是一个句型。
- 起始符也为句型,即
  隐含条件:$S \to S$, 所以S也是文法G的句型。

语言$L[G]$的定义:语言$L[G]$是文法$G[S]$的所有句子的集合。
注:

- $+, (, *$,等出现在句子中的成分,也是终结符。
- 可以通过产生式的具体表达,判断出某个操作的优先级。

文法的等价:
若$L[G_1] == L[G_2]$,则称文法G_1 和文法G_2 等价。

文法的种类:

- 0型文法
- 1型文法(上下文相关文法)
- 2型文法(上下文无关文法)
  大部分程序设计语言是2型文法
- 3型文法(正规文法)
  一般用来定义一个单词