一个自动编译系统ACS

程虎 李爱武

(中国科学院软件研究所)

A AUTOMATIC COMPILER GENERATION SYSTEM ACS

Cheng Hu and Li Aiwu

(*Institute of Software, Academia Sinica*)

ABSTRACT

In this paper, we introduce a compiler generation system —ACS. ACS can automaticly generate a compiler of a programming language by input the programming language's lexical description, syntax description, semantic description and target machine description.

摘 要

本文介绍一个自动编译系统ACS。对某上下文无关文法语言，写出其词法描述、语法描述、语义描述和目标机描述。ACS 系统可以自动生成此语言在目标机上的编译程序。ACS 系统由六个子模块组成：词法自动生成器、语法自动生成器、语义自动生成器、中间语言转换程序、优化程序和代码自动生成器。本文将分别讨论这六个子模块的原理和功能，还将给出用ACS 系统生成的一个编译程序实例—Mini_Ada 编译程序。

§1. 系统概述和系统结构

随着计算机科学和技术的发展，计算机的种类和计算机程序设计语言的种类越来越多，因此有大量的编译程序需要完成。而手工编写编译程序的工作相当繁琐，多年来人们就在研究编译程序自动化的方法，近几年自动编译方面的研究有了很大进展。

自动编译系统是从源语言和目标机的形式描述来自动生成相应的编译程序。由于程序设计语言的繁多和机器种类的不同，很难有一种描述方法能描述众多程序语言和目标机。因此我们将问题分解，对词法分析、语法分析、语义分析、代码生成各部分分别实现自动生成。

1989 年 9 月 25 日收到，1990 年 1 月 12 日定稿，中国科学院院内基金资助项目。

在自动编译系统中，中间语言的选取也是一个关键问题。中间语言选取得高，代码生成生成器不容易完成；中间语言选取得低，语法/语义生成器不容易实现。所以，ACS系统采用了二级中间语言：高级中间语言DIANA 和低级中间语言AIM。

DIANA[2] 是一种与源语言和目标机无关的属性树型中间语言。DIANA 有170 种树结点和100 种属性，每一种树结点上有其固定数目的属性种类。DIANA 的170 种树结点根据文法组成不同的结构，能完全刻划出源语言的特性。

AIM[3] 是指令型中间语言，每条指令的操作数从0 到3 不等，每条指令和每个操作数都有各自的属性值。AIM 程序为一串指令序列，AIM 程序的结构是基本块状，基本块以一条标号定义指令开始，以一条转移指令结束。

ACS 系统由6 个子模块组成：词法分析生成器、语法分析生成器、语义分析生成器、中间语言转换程序、优化程序和代码生成生成器。ACS 系统的逻辑结构如图1。

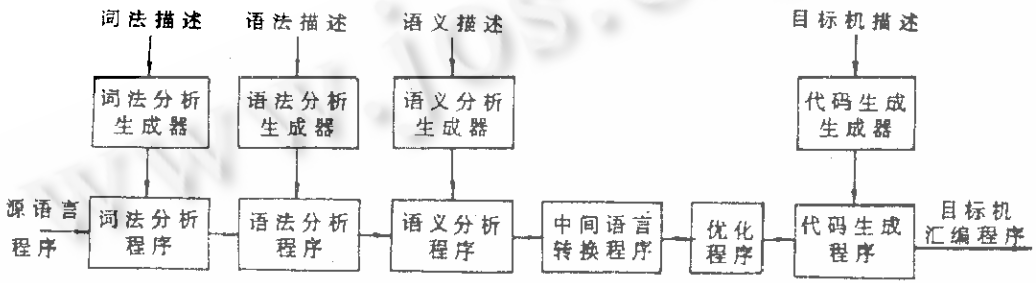

图1 ACS 系统逻辑结构图

§2. ACS 系统中各子模块的功能

2.1 词法分析生成器

词法分析生成器是基于有限自动机原理。可以将词法分析程序看成一个有限自动机，将要分析的源程序看成是有限自动机的输入，该有限自动机的输出就是要分析语言的基本单位— 单词。词法分析生成器就是这样的程序，它根据要分析语言的词法规则，根据有限自动机的理论，构造出一个分析该语言的有限自动机。

词法分析生成器的输入为词法描述。词法描述有三种形式：描述语言字符集；描述表示语言词法规则的正规表达式；描述语言的关键字。例如：

L → ABCDEFGHIJKLMNOPQRSTUVWXYZ ##

LE → e E ##

DI → 0123456789##

词法分析生成器的逻辑结构如图2。它接受源语言的词法描述，扩展正规表达式，产生不确定的有限状态图，产生确定的有限状态图，简化状态图，产生状态表和输出表。这二个表和一个预先设计好的驱动程序结合起来就产生出词法分析程序。

2.2 语法分析生成器

ACS 系统的语法分析生成器生成满足LALR (1) 文法语言的语法分析程序。LR 类语法分析程序可以看作一个带先进后出栈的确定有限状态自动机，把已经移入和归约的整个符号表串存放在栈里，再根据当前的输入串和使用的产生式，进行移进和归约，从而完成语法分析。这种语法分析程序是一张语法分析表加上表驱动程序。对不同的源语

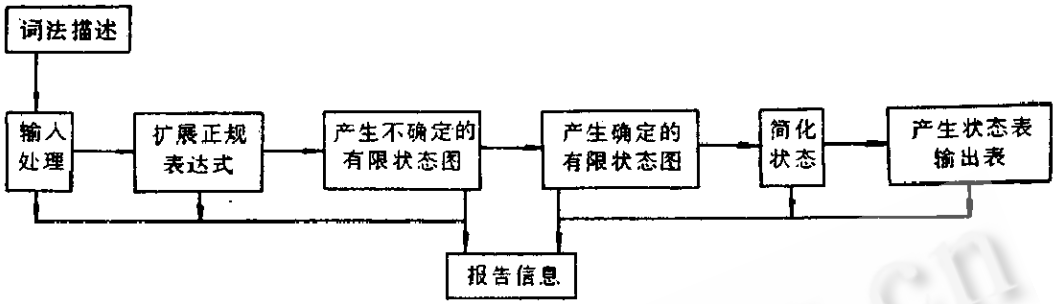

图2 词法分析生成器框图

言，表驱动程序都是一样的，只有语法分析表不同。语法分析生成器的任务就是根据语法规则，对输入的字符串进行语法分析，再根据给定的上下文无关文法构造LR (0) 分析表、LALR (1) 分析表。

. 语法公式处理模块主要处理用BNF 表示的语法描述输入。

. 树结点描述处理模块处理输入的树结点描述，产生树结点，并将它转变为内部表示形式。程序产生模块利用这种内部形式，产生相应的树产生程序。

语法分析生成器逻辑结构如图3。

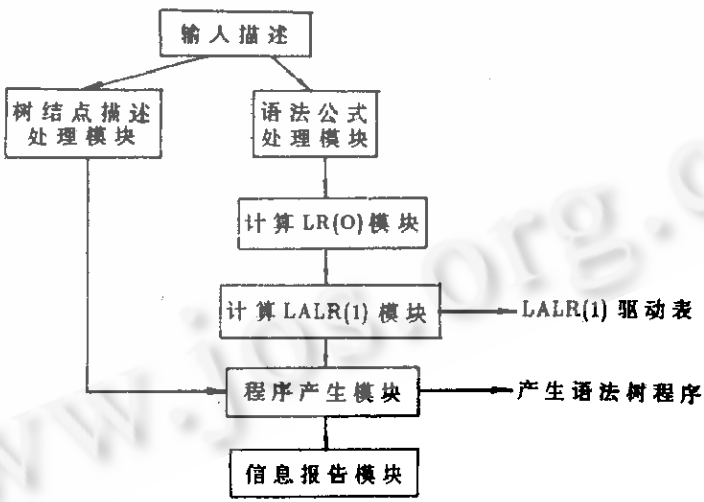

图3 语法分析生成器框图

2.3 语义分析生成器

语义分析生成器根据属性文法的理论，可以为很大一类上下文无关文法构造语义分析程序。

基于属性文法的理论，语义分析生成器从功能上来说是要把以产生式为单位的语义描述，一步一步地变为相应的语义计值程序。为了实现这一目的，语义分析生成器必须

至少包括下面三个部分：1.) 输入描述，2.) 属性依赖性分析，3.) 程序生成模块。这三大部分分别由一些模块来实现。按照各模块执行的先后次序，这些模块是符号描述处理模块、语义规则描述处理模块、产生式直接依赖关系模块、符号直接依赖关系模块、导出属性依赖关系模块、访问序列模块、语义分析程序生成模块。

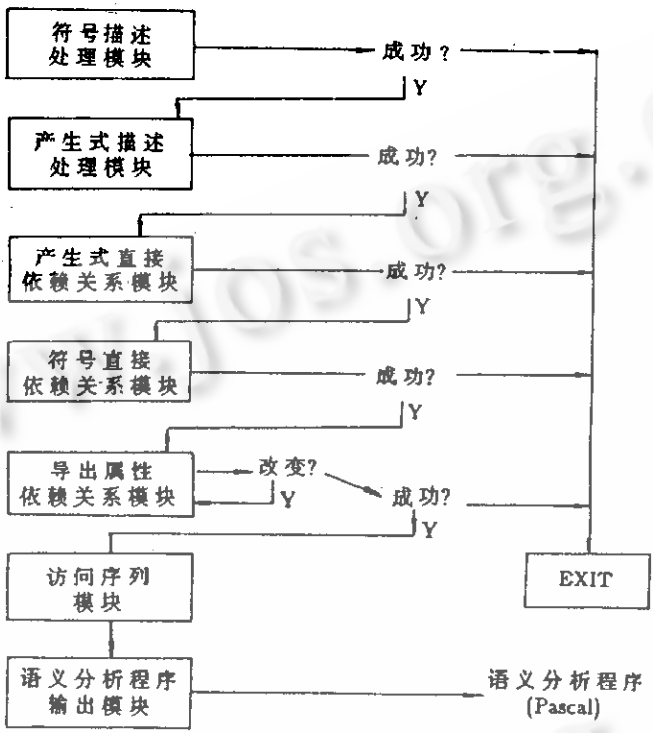

图4 语义分析生成器框图

描述处理模块主要有两个任务：一是从语法级上检查所输入的符号描述和语义规则描述是否正确；二是将符号描述和语义规则描述转换为内部形式，为后面模块做准备。

属性依赖性分析的目的在于根据输入描述，求出属性间的依赖关系，以便最后确定语义计值顺序。

生成模块主要有两个任务：一是按照定义对每个产生式P 中的顺序进行排序，产生相应的属性序列AS(P)；二是将属性序列转换成指令序列IS(P)；最后形成语义分析程序。

以上各模块之间通过内部数据结构相互联系，这些模块间的关系如图4。

2.4 中间语言转换程序

中间语言转换程序完成高级中间语言DIANA 到低级中间语言AIM 的转换。由于DIANA 属性树型结构比较复杂，AIM 指令型结构相对简单，所以DIANA 到AIM 的转换利用属性文法原理[4]，采用语法制导的递归下降方法完成的。使用这种方法，首先要写出DIANA 语言的LL(1) 文法；其次定出DIANA 的LL(1) 文法中所有终结符和非终结符的属性计值次序，并使其满足L- 属性计值规则；然后再写出DIANA 到AIM 的属性翻译文法。

根据DIANA 到AIM 的属性翻译文法。给出每个非终结符所对应的递归过程和每个{动作} 符号所对应的生成AIM 程序的过程。文法的非终结符所对应的过程完成相应的属性计值和导出由此非终结符导出的符号序列，文法的{ 动作} 符号所对应的过程完成相应的属性计值并产生AIM 程序。

中间语言转换程序的逻辑结构如图5。

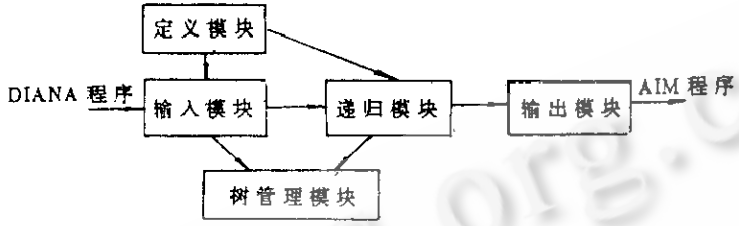

图5 中间语言转换程序框图

. 定义模块定义DIANA 的属性树结构、AIM 指令的数据结构和一些全局变量。

. 输入模块完成对DIANA 程序的词法/ 语法分析，并形成DIANA 的内部结构。

. 树管理模块由一些DIANA 树操作组成，完成构造树和访问树结点。

. 递归模块是中间语言转换程序的核心，它由文法中非终结符对应的递归过程和文法中{ 动作} 符号对应的过程组成，完成DIANA 的内部结构到AIM 的内部结构转换。

. 输出模块完成AIM 到格式输出。

当读入DIANA 程序时，按照DIANA 的LL(1) 文法，执行非终结符对应的递归过程，这些过程识别DIANA 程序并调用相应的{ 动作} 符号对应的过程，递归过程递归执行结束则就完成了DIANA 到AIM 的翻译转换。

2.5 优化程序

优化程序对AIM 中间语言进行如：表达式优化、强度削弱、循环优化等一系列优化。优化程序采用归纳指令码方法(Inductive Instruction Code) 优化AIM 程序。归纳指令码IIC 算法如下：

AIM 是指令型语言。据此，初始化时为每条指令按字典顺序进行编码，称为指令码。AIM 共有101 条指令(编码值为0—100)。另外，某些指令可能有编码修正值，修正以后的指令码大于100，并且不重复。以指令码为基础，按指令之间调用关系确定每条指令在程序段中的相对IIC 码，这里相对是一条指令在不同的位置可能有不同的IIC 码。

AIM 程序中，任一表达式指令序列均以变量或常量结束，而其它指令可看作是由这些变量或常量归纳所得，若二个表达式同形，则这些量相应相等或具有相等编码值。此时，为了确定表达式是否冗余，还必须确保第二个表达式中的变量在两个表达式之间不能被改变。IIC 码可以肯定或否定这一点，具有相同的IIC 码序列的两个表达式必有冗余。

如果表达式EXP1 和EXP2 冗余，则两者之间所有相应的操作数冗余。因此确定IIC 序列的基准在于最基础的操作数，即变量或常量。

算法的基本思想是：一旦对中间代码的归纳搜索遇到变量或常量，则回溯可能的变量或常量，判断该变量或常量是否冗余，若是则置两者IIC 码相同，否则为其赋予新的

IIC 值。然后利用归纳过程求其操作指令的IIC 值。在求得的IIC 序列中，若有两段代码的IIC 码相同，则冗余。

优化程序的逻辑结构如图6。控制块协调各块运行；信息收集块读入AIM 程序，建立5种表；基本块压缩查找无限循环，消除无用块，合并顺序执行块；IIC 块建立IIC 值，进行常量合并，常量传播，变量传播；EXP 块消除无用赋值，冗余语句，冗余表达式；LOOP 块查找循环，不变代码外提，强度削弱。

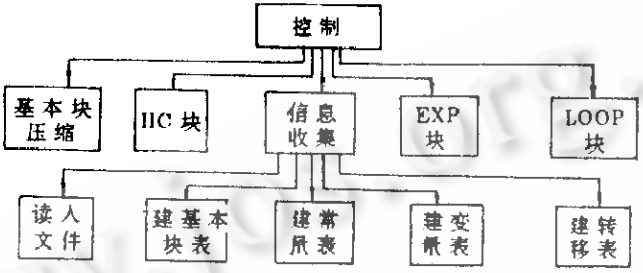

图6 优化程序框图

2.6 代码生成生成器

ACS 系统中的代码生成生成器采用Graham Glanville 的表驱动方法。这种方法的基本思想是：认为代码生成应做的工作和语法分析应做的工作是类似的。语法分析阶段是接受源语言程序，产生出中间语言程序；而代码生成则是接受中间语言程序，产生出目标机代码。因此代码生成可以采用语法分析采用的方法，并借用语法分析的理论。

代码生成生成器主要由三部分组成：机器描述语言TMDL(Target Machine Description Language)、代码生成表构造程序、表驱动代码生成程序。机器描述语言TMDL 将机器性质描述出来，为代码生成生成器提供目标机信息。TMDL 又分为四部分，第一部分是终结符描述，描述原子、运算符、各种语法符号、分隔符等；第二部分是寄存器描述；第三部分是非终结符描述；第四部分是机器指令描述，它采用中间语言产生式与汇编信息对应的形式，使与机器有关的信息以表的形式从与机器无关的算法中分离出来。当想改变目标机和中间语言时，只需要改变TMDL 描述，而不需要修改其它程序。代码生成表构造程序以目标机的TMDL 描述为输入，用语法分析的方法构造出目标机的代码生成表。表驱动代码生成程序接受中间语言程序，根据代码生成表，对中间语言程序进行分析，产生目标机的汇编程序。

对一种TMDL 描述，代码生成表只需要运行代码生成表构造程序产生一次，以后对每个中间语言程序，只运行表驱动代码生成程序就可产生出它在目标机的汇编代码。代码生成表和表驱动代码生成程序一起构成代码生成程序。代码生成生成器的逻辑结构如图7。

§3. Mini_Ada 编译程序

将Mini_Ada 语言的词法描述、语法描述和语义描述分别输入到词法分析生成器、语法分析生成器和语义分析生成器中，生成Mini_Ada 语言的词法分析程序、语法分析程序

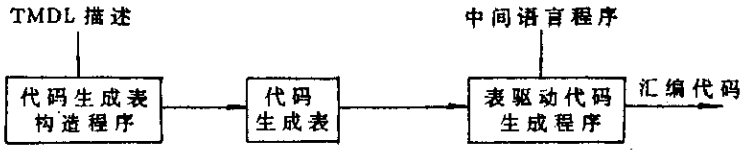

图7 代码生成生成器框图

和语义分析程序；将目标机IBM PC/XT 的描述输入到代码生成生成器中，生成相应的代码生成程序[1]。上述所生成的程序加上中间语言转换程序和优化程序就构成Mini_Ada 语言的编译程序。生成的编译程序参见图1。对给出的Mini_Ada 源程序，运行Mini_Ada 编译程序，生成IBM PC/XT 机上的汇编代码。

例如:

Mini_Ada 源程序如下:

```
with uda_io;
procedure example is
use uda_io;
i, j, k: constant integer:=0;
l, m. n: integer: =1;
begin
l:=l+i;
m:=m+j;
n:=n+k
end example;
```

生成的IBM PC/XT 机器上的汇编程序如下:

```
STACK SEQMENT PARA STACK 'STACK'
DB 256 DUP (0)
STACK ENDS
DATA SEQMENT PARA PUBLIC 'DATA'
T14_1 DW 0
ELB0014 EQU T14_1
T14_2 DW 0
DBY0014 EQU T14_2
T14_3 DW 0
T14_4 DW 0
T14_5 DW 0
T14_6 DW 0
T14_7 DW 0
T14_8 DW 0
DATA ENDS
CODE SEQMENT PARA PUBLIC 'CODE'
START PROC FAR
ASSUME CS: CODE
PUSH DS
MOV AX, 0
PUSH AX
```

```
MOV AX, DATA
MOV DS, AX
ASSUME DS: DATA
CALL WW
LABEL 3:
MOV AX, 0000000H
MOV T14_5, AX
MOV AX, 0000000H
MOV T14_4, AX
MOV AX, 0000000H
MOV T14_3, AX
MOV AX, 1H
MOV T14_6, AX
MOV AX, 1H
MOV T14_7, AX
MOV AX, 1H
MOV T14_8, AX
RET
START ENDP
WW PROC NEAR
LABEL 1: RET
WW ENDP
CODE ENDS
END START
```

§4 结束语

ACS 系统共有20200 行源程序。在VAX / VMS 操作系统支持下，运行在VAX 11 系列计算机上。ACS 系统适用于类ADA 过程型语言，可以为任何目标机生成汇编代码。ACS 系统采用二级中间语言(DIANA 和AIM)，并有优化程序对AIM 程序进行优化，生成的汇编代码质量较好。但目前的ACS 系统仅完成对Mini_Ada 语言生成IBM PC/XT 上汇编代码，ACS 系统还有待于进一步工程化。

参加本项目工作的还有王用宁、王京辉、田玉萍、吕智、张绮、高伟、唐常青同志。

参考文献

[1] 中国科学院软件研究所，自动编译系统鉴定会资料，1989.4。

[2] G. Goos, W. A. Wulf, "Diana An Intermediate Language (Lecture Note in Computer Science 161)", Springer-Verlag, 1983.

[3] H. Jansohm, R. Landwehr, "A Code Generator for Ada", Technical Report Nr-82/33.

[4] M. Waite, G. Goos, "Compiler Construction", Springer-Verlag, 1984. (中译本，陈嘉生、程虎等译，上海科技文献出版社出版，1989。)