一个面向JSD方法的规格说明语言转换系统*

周晓彤 宗丽苹 丁茂顺

(北京联想计算机集团公司,北京 100080) (中国科学院软件研究所,北京 100080)

摘要 JSLC系统是一个规格说明语言JSL的转换系统.它的实现反映了一种规格说明语言的转换方法.本文详细介绍了JSLC系统的组成结构、实现基础和技术要点.

关键词 规格说明语言,事件驱动,进程,数据流,状态向量.

规格说明语言(Specification Language)的功能一般是按照一定的语言规范来描述系统的规格、需求和功能,以指导系统的开发工作,其特点是容易理解,但可执行性较差.规格说明语言的应用范围很广,例如,第四代语言描述的系统可以看成是第三代语言系统的规格说明语言,它的执行可以通过转换成第三代语言来实现.

语言的转换技术贯穿着许多领域,例如,计算机辅助软件工程CASE(Computer Aided Software Engineering)环境中规格说明语言的编译执行,面向第四代语言的NOMAD等系统中所采用的SQL语言的执行,机器翻译等等,都用到了语言转换的技术和思想.规格说明语言的转换技术,涉及到语法制导、中间语言生成、自动机和可计算性等方面的内容.

JSLC系统的设计思想是:在一个面向JSD方法的规格说明语言JSL到C语言的转换过程中,以表作为中间媒介,经过一系列表的转换,把最终的表的结构和目标语言相对应.而根据经典算法,一些最基本的表是必不可少的,就本系统来讲,这些基本表是一些中心数据库,其它类型的表是这些表的应用.

JSLC系统在完成JSL语言向C语言的转换过程中所用到的基本技术主要是语法监测技术和表格变换及其管理技术.

① 语法监测技术

语法监测技术一般通过语法监测器来实现.与传统的语法制导技术相比,该项技术有两大特点:

· 不需要进行严格的检错处理.这是因为JSL语言可以看成为一种第四代语言,它接

* 本文1991-08-02收到,1992-05-21定稿

作者**周晓彤**,28岁,1991年硕士毕业于中国科学院软件研究所,主要研究领域为软件工程方法学、第四代语言及语言翻译.**宗丽苹**,女,29岁,1990年硕士毕业于中国科学院软件研究所,主要研究领域为软件构造学,第四代语言及用户接口.**丁茂顺**,52岁,研究员,主要研究领域为软件工程,软件工具环境及软件接口.

本文通讯联系人:周晓彤,北京 100080,埃迪尔电子信息技术开发公司

近于自然语言,语法要求简单,容许有一定的不确定性,这些不确定性的因素可在向第三代语言的转换过程中予以克服.

·不需按语法进行严格的归约.这是因为这两种高级语言之间的转换只需通过对各自部分的对应就能完成.

② 表格变换及其管理技术

JSLC 系统是以表格作为中间语言来实现 JSL 语言与 C 语言的对应的.表格由中心环境表和高层应用表所组成.其中,对中心环境表的管理采用数据库的管理方法,并对每个表按其关键字建立各自的 B 树索引,这样就可以提高对中心环境表的操作效率.这点在用 JSL 语言所描述的系统很庞大时,优点尤其明显.高层应用表是对中心环境表施以联结、部分联结等操作而成.

JSLC 系统是一个规格说明语言 JSL 的转换系统.它的实现反映了一种规格说明语言的转换方法.本文详细介绍了 JSLC 系统的组成结构、实现基础和技术要点.

1 JSLC 系统概述

1.1 JSLC 系统的运行环境

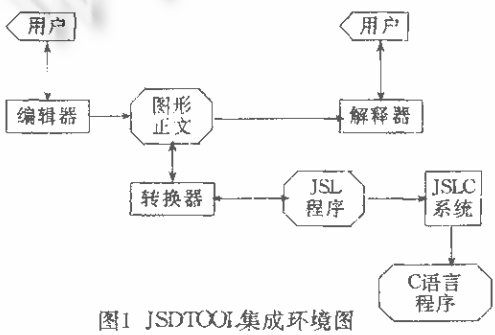

图1 JSDTOOL集成环境图

JSLC 系统是对一个面向 JSD 方法的软件开发环境 JSDTOOL 中规格说明语言 JSL 进行语言转换的系统.它主要完成 JSL 语言描述向 C 语言程序的转换,通过对所转换而成的 C 语言程序加以编译执行来达到相应软件系统的设计目标.和解释器相比,JSLC 系统大大地提高了 JSL 语言的执行效率.在系统设计时,我们从系统的可适应性和可推广性的角度考虑,JSLC 系统应既可在 DOS 环境下运行,又可在 JSDTOOL 集成环境下执行.JSLC 系统在 JSDTOOL 环境中的位置可在图 1 中反映出来.

1.2 JSLC 系统的源语言结构

JSL 语言是 JSLC 系统的源语言,它以结构化正文形式来描述软件系统,有严格的语法限制.JSL 语言由变量说明、事件说明、数据流说明、状态向量说明和进程说明等 5 部分所组成,还设有一个系统标志符.

① 变量说明部分:用来对 JSL 语言描述中所用到的变量加以说明.

② 事件说明部分:说明进程中的所有事件,而事件又定义了进程的行为.说明一个事件也就是定义一个事件类型,包括事件的属性及其所属的进程.

③ 数据流说明部分:对应于一个数据流类,给出其元素的消息类型,并且对以数据流的形式所连接的两类进程进行说明.在实现上数据流有一个无限的缓冲区,用于存放事件说明部分中所定义的消息(进程说明部分中的读、写语句对这些消息进行存取).

④ 状态向量说明部分:定义各进程之间状态向量的连接关系,包括状态向量的名字、要引用的内部变量和存取方法等.状态向量的名字是一个消息通道,用以传送对一个进程中内

部状态进行读取的请求消息或回答消息. 存取方法指出了选取进程的条件和访问进程的顺序.

⑤ 进程说明部分：由两部分组成. 第一部分描述了进程调度的情况，包括进程的同步和异步通信；第二部分描述了每个进程的内部结构和具体功能，包括对数据流和状态向量等进行存取和使用的方法(有读、写和取状态向量等). 其中，读语句的功能是从数据流中得到消息，并把该消息存放在给定的变量中. 写语句的功能是把消息写进数据流. 取状态向量语句的功能是读取另一个进程的状态向量；系统执行这条语句时，将发送一个消息给被读取的进程，要求得到该进程的内部状态，所返回的结果就是其状态向量.

下面是一个用 JSL 语言所写的"简单银行管理系统"描述程序中进程说明的部分内容.

```
PROCESS
  ...
    customer—1 SEQ
      ...
        withdraw SEQ
          balance = balance — .amount;
          if (balance < 0)
            swrite (exceptionreport "overdrawn"
                  .customer—name balance);
          sread(c);
        withdraw END
      ...
    customer—1 END
  ...
PROCESS END
```

1.3　JSLC 系统的目标语言结构

对 JSL 语言实施转换而生成的一种 C 语言的子集是 JSLC 系统的目标语言. 与 JSL 语言描述相对应，这种 C 语言程序模仿了 JSL 语言描述中进程调度的思想，具有描述事件的数据类型，其主控结构也是一种事件驱动的结构，即有一个等待事件的函数 Wait—ForEve( )被主程序所调用；事件被用户选定后，系统就调用相应的进程处理函数；进程则执行相应的动作函数. 这种 C 语言程序的主控结构(即高层函数调用关系)描述如下.

```
main()
{
    Wait—ForEve ( Event. Pro—Id, Event.
    Action);
    switch(Event.Pro—Id)
    {
        case 0: Process0(Event.Action);
                break;
        case 1: Process1(Event.Action);
                break;
        case 2: Process2(Event.Action);
                break;
        ...
        default: break;
    }
}
Process0(Action)
int Action;
{
    switch(Action)
    {
        case 0: Action0();
                break;
        case 1:Action1();
                break;
        ...
        default: break;
    }
}
...
Action0()
{
    赋值语句;
    条件语句;
    数据库查询语句;
    系统的输入输出语句;
}
```

1.4 JSLC系统的源语言和目标语言之间的对应关系

高级语言之间的转换应该明确源语言和目标语言各部分的对应关系.JSL语言描述与C语言程序之间的对应关系见表.

JSL语言描述	C语言程序
①变量说明部分	变量说明
②事件说明部分	进程调度函数(即主函数)以决定系统的进程调度 类型说明和变量说明 动作调度函数(即进程处理函数)
③数据流说明部分	与信息的输入、输出等功能有关的函数
④状态向量说明部分	与数据库查询功能有关的函数
⑤进程说明部分	
动作描述	动作处理函数
其中,赋值语句	赋值语句
if语句	if语句
sread语句	用户交互功能的函数
swrite语句	屏幕显式函数或数据库修改函数
getsv语句	数据库查询函数

通过这种对应关系所得到的C语言程序,既有输入和输出功能,又有数据库管理功能,以及各种动作处理函数的具体描述;在此之上还有动作和进程调度函数,是比较完整的C语言程序.

2 JSLC系统的组成结构

2.1 JSLC系统的组成

JSLC系统的组成图如图2所示.

2.2 JSLC系统的功能模块简介

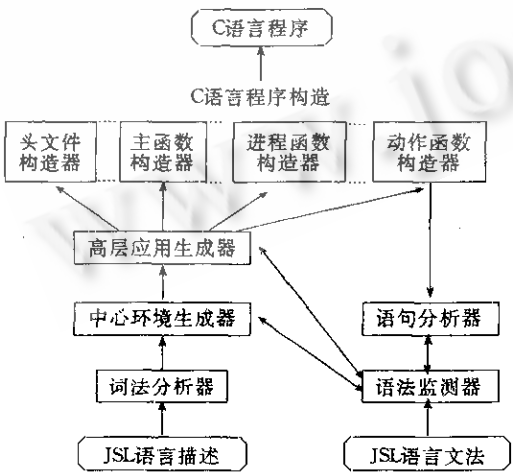

图2 JSLC系统的组成图

① 词法分析器:该部件从JSL语言描述中读入词法单位,并且返回该词法单位的类型和内部值.

② 中心环境生成器:该部件调用词法分析器来构成JSLC系统的中心环境表,即变量表、动作表、数据流表、状态向量表等.

③ 高层应用生成器:该部件在中心环境表的基础上,形成面向目标语言的数据表或数据结构,即进程表、类型表、函数表等.

④ 语法监测器:该部件分别在中心环境表的构造、高层应用的生成和C语言程序的构造时,对词法错误和语法错误分

别进行监测和处理.

⑤ 头文件构造器:该部件生成C语言程序的头文件,包括include文件、系统自定义类型和全程变量.

⑥ 主函数构造器:该部件生成C语言程序的main( )函数,即进程调度函数.

⑦ 进程函数构造器:该部件构造出C语言程序的各类进程函数,即进程的动作调度函数.

⑧ 动作函数构造器:该部件构造出C语言程序的动作函数体,使描述的动作得以实现.

⑨ 语句分析器:该部件被函数构造器所调用,分析JSL语言描述中动作描述的各种语句,并转换成相应于C语言程序的语句.

3 JSLC系统的主要实现技术

JSL语言描述转换成C语言程序的关键两点是:① JSL语言描述中的进程说明部分转换成C语言的进程函数选取机制;② JSL语言描述中的动作描述转换成C语言程序的动作函数体.转换过程中以各种数据结构作为中间媒介.

3.1 中心环境表

中心环境表是整个JSLC系统转换工作的基础,由中心环境生成器扫描JSL语言描述而得到.这些表全面地反映了JSL语言程序的特征,而且各表之间无冗余,每个表内部各项无函数依赖关系.这些表是:

变量表——主要内容:所属类型名、变量名称、变量关键字、变量个数.

动作表——主要内容:所属进程标志符、动作名称、所属数据流名称、动作属性指针.

数据流表——主要内容:数据流名称、数据流的发送者、数据流的接收者.

状态向量表——主要内容:状态向量名、状态向量的发送者、状态向量的接收者、状态向量的信息指针.

这些中心环境表要被高层应用生成器、主函数构造器、进程函数构造器、动作函数构造器、语句分析器等所引用,完成各自的功能.

3.2 高层应用表

高层应用表建立在中心环境表的基础上,由高层应用生成器所生成.这些表是:

进程表——主要内容:进程标志符、进程变量指针、进程所拥有动作的头指针.

类型表——主要内容:类型名称、类型所拥有分量的头指针.

函数表——主要内容:函数所属进程、函数名称、指向函数中参数的指针、指向函数中变量的指针.

3.3 进程函数选取机制的实现

进程函数选取机制由主函数构造器和进程函数构造器组成.

主函数构造器以进程表为基础,构造出C语言程序的主控函数main( )——C语言程序顶层的事件驱动控制机制和多路开关结构.

进程构造器由关键字生成模块和动作调度函数生成模块所组成,共同完成进程函数体的构造:

① 关键字生成模块作用于中心环境表中的变量表和高层应用表中的类型表,它的实现

思想模仿了多任务操作系统中用数据结构来对应进程的原理，用C语言程序中的数据类型结构变量的数组来对应JSL语言描述中的进程.JSL语言描述中的进程一定有某些特点能映射成C语言程序中数据类型的关键字，而该模块的主要功能就是确定C语言程序中数据类型的关键字，并根据关键字来选中某个数据类型的结构变量，使之与JSL语言描述中的某个进程相对应.这样，C语言程序中对结构变量的选取机制就完全反映了JSL语言描述中进程的选取机制.

② 动作调度函数生成模块作用于函数表和变量表，把JSL语言描述中的进程对动作的选取机制转换成C语言程序中的动作调度函数.

3.4 动作函数的实现

函数构造器用来实现C语言程序中动作函数的生成.它首先读入JSL语言描述中的每个进程的各条语句，用语句分析器加以分析，形成该动作函数的语句类型链；然后将函数表中对应的两个指针指向语句分析器所形成的参数链和变量链，最后根据所有这些链表向输出文件中输出该动作函数.

语句分析器的功能是：根据所读入的语句，确定其类型(JSL的赋值语句、条件语句、循环语句、写数据语句、读数据语句、取状态向量语句等)，并填写函数构造器所得到的语句类型链表中的结点.同时，对所处理语句中的每一变量进行两遍处理：首先判断它是否为动作

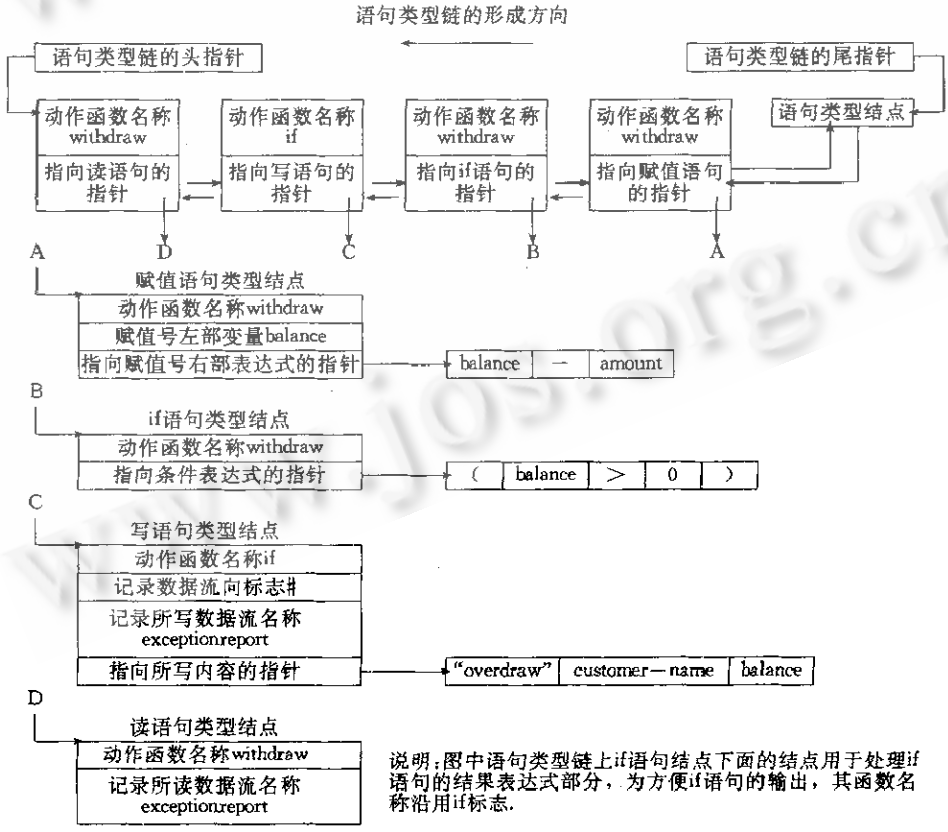

图3 动作函数withdraw中语句链的形成过程

函数的参数，若是，则把它挂到参数链上；再判断它是否为动作函数的变量，若是，则把它挂到变量链上.

语句类型链是一个双向链表，它的主要内容包括动作函数的名称，以及指向 JSL 语言描述中各种语句类型的指针.

下面以"简单银行管理系统"JSL 语言描述中的进程说明部分的动作 withdraw 转换成相应的 C 语言程序中动作函数 withdraw()为例，来讲明函数构造器的执行流程，见图3.

4 讨　论

如果对现有的 JSLC 系统再配上几种语法监测，就可以形成一个多种规格说明语言到多种目标语言的转换系统，即通用语言转换系统，而且这种系统还可以配备良好的用户界面. 这些，还需要我们作进一步深入的工作.

参考文献

1 Aho A V *et al*. Compilers——principles, techniques and tools. Addison—Wesley, 1986.

2 Gregory W, Wojtkowski W. Applications software programming with forth generation languages. Featuring PC NOMAD, 1990.

3 Jackson M A. System development. Printice Hall, 1982.

4 McClure C. CASE is software automation. Prentice Hall, 1989.

5 Yellin D M. Attribute grammar inversion and source—to—source translation. Springer—Verlag, 1988.

6 周伯生，董士海. 软件工程环境引论. 计算机研究与发展，1987(6).

7 N. 沃思著，曹德和等译. 算法＋数据＝程序. 北京：科学出版社，1987.

8 仲萃豪，丁茂顺. 应用软件的开发方法. 计算机科学，1991(1).

9 丁茂顺，易章雄. 支持 JSD 开发方法的工具 JSDTOOL. 多目标集成化可移植软件工程环境 MTIPS 技术报告，1990.

A TRANSLATING SYSTEM OF THE JSD—ORIENTED SPECIFICATION LANGUAGE

Zhou Xiaotong

(*Beijing Legend Computer Group Corporation, Beijing* 100080)

Zong Liping and Ding Maoshun

(*Institute of Software, The Chinese Academy of Sciences, Beijing* 100080)

Abstract　JSLC is a translating system of the specification language JSL. The implementation of JSLC implies a model of specification language translation. In this paper, the system organization, the fundamentals of implementation, and the key points of techniques for JSLC, are described in detail.

Key words　Specification language, event driven, process, data flow, state vector.