基于 XML Schema 技术的编译符号表生成方法*)

聂 南[1,2] 谢晓东[1] 甘 勇[2]

(华中科技大学计算机科学与技术学院 武汉 430074)[1] (郑州轻工业学院计算机与通信学院 郑州 450002)[2]

摘 要 传统的编译中间代码通常不能在移动、嵌入式和分布式等环境之间转换,而符号表的构造与管理贯穿整个中间代码的生成过程。本文提出一种基于 XML Schema 及其相关技术生成编译中间代码的符号表的方法。首先给出整体方案,然后阐述了如何运用 XML Schema 等技术表示编译器的前端,以及后端的目标机体系结构。生成的编译器符号表能通过 XML 工具统一管理和验证,从而使生成的编译中间代码有较高的可移植性,能被不同环境中的编译器采用。

关键词 编译, XML Schema, DOM, XQuery

A Compiler Symbol Table Generation Approach Based on XML Schema

NIE Nan[1,2] XIE Xiao-Dong[1] GAN Yong[2]

(School of Computer Science and Technology, Hua Zhong University of Science and Technology, Wuhan 430071)[1]

(School of Computer and Communication, Zhengzhou University of Light Industry, Zhengzhou 450002)[2]

Abstract Traditional compiling intermediate code is normally hard to be transformed between heterogeneous environments, such as mobile, embedded and distributed sstems. The building and management of symbol table is related with all stages of intermediate code. So a compiling symbol table building method based on XML Schema and its related techniques is proposed. Firstly, the design architecture of compiling process is given, then it expatiated how to represent front end and object machine architecture of backend of compiler. As a result, the symbol table can be managed and validated by XML tools, so the compiling intermediate code is with high portability, which can be applied and translated into compilers of different environments.

Keywords Compile, XML schema, DOM, XQuery

1 引言

目前,对编译器的研究已有非常成熟的理论基础和大量实用的工具[1]。可是随着嵌入式、移动和分布式等异构系统的不断发展,现在迫切需求能够适用于多种语言环境的编译技术,开发出支持多目标机的新型编译器。例如,嵌入式系统对编译器技术提出了新的要求:嵌入式系统具有较短的生命周期,因此需要能够快速生成编译器。为此,最好的解决方法就是编译器采用多目标设计方法,即对于不同的处理器,只需要修改编译器与机器相关的后端代码即可生成相应的编译器[2]。

作为自描述的标记语言,XML 能够根据具体应用灵活地表现异构数据源中的各种信息,包括应用程序之间的数据交换、结构化和半结构化文档以及数据库中数据的输出。这使得其在各个行业的应用具有开创性。文[3]提出将 XML 作为编译中间代码,只是把 XML 语法与编译语法分析中的文法做了比较并进行了转换;文[9,10] 提出采用 XML 重建抽象语法树(AST)的方法。从这些已有的研究来分析,XML 技术在编译过程中的应用研究还比较片面;另一方面,随着 XML 技术的不断更新,XML Schema, XQuery 等技术在不断发展成熟。基于此,本文提出采用 XML Schema 及其相关技术构造编译器中间代码符号表的方法,规划了整体设计思想,阐述了如何应用 XML Schema 描述编译过程词法表,语法产生式,语法树和目标机器结构,并且给出符号表的文档实例,从而达到编译符号表通用且可视化,生成的中间代码更容易被不同环境下的目标代码采用和转化的目的。

2 编译中间代码的总体框架设计

编译器一般分为两个部分:前端(front end)和后端(backend)。前端与源语言相关,它对源程序进行词法分析、语法分析和语义分析,将源程序转化为一种中间表示形式(IR)。IR 作为一种公共交换格式存放,并形成后端的输入。基于这种常规形式,XML 能够作为编译器开发的中间形式有以下原因。首先, XML 的 Schema 等技术具有与编译技术类似的功能,例如 XML Schema 具有丰富的类型定义,严格的语法结构,并且能够扩展,还可以用来表示正则表达式[4,5]。传统的编译过程,例如词法,语法,语义分析过程,符号表都能使用 XML 技术重新构造。其次, XML 是一种基于文法规范的文本,其树型的编码使人容易读。这种特征允许实践优化技术,也能够让编译过程所有阶段生成的结果树变得易读。再者,XML 被设计成编码树,XML 的相关工具能够解析它们。此外,一些组织逐步推出支持其解析和验证的工具包,最著名的是 Microsoft 的 MSXML 和 Apache 的 Xerces 系列。一些研究开发人员也对此做了补充,例如文[6] 设计实现了基于 XML Schema 的 XML 局部验证接口,包括类型检查和 ID 约束检查等。总之,采用 XML Schema 等技术构造编译器的符号表,语法树框架,在此基础上设计 XML 文档,使用 DOM,XQuery 等技术解析遍历它们,能够实现一种新型的编译过程,而且该过程是规范严谨的。总体设计思想如图 1 所示。

*)基金项目:中国湖北省自然科学基金(2005ABA266),河南省自然科学基金(0611054800)。聂 南 博士生,讲师,研究方向为软件工程与理论。谢晓东 博士,讲师,研究方向为软件工程与理论。甘 勇 教授,博士,CCF 高级会员,研究方向为软件工程,系统集成。

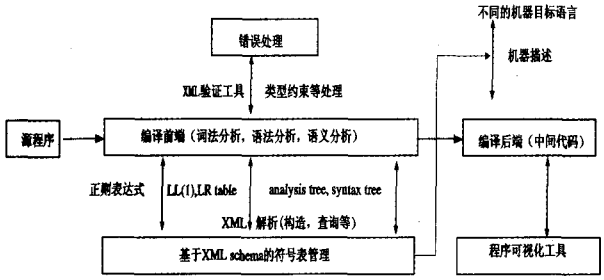

图1 编译过程框架

通过XML及其相关技术，传统的编译过程能转换源程序为XML格式的中间代码。它不仅能被用来在不同的目标语言之间转换，而且能被各种程序可视化工具跟踪和测试。此外，编译的各部分是相辅相成的，例如，错误处理部分不仅要使用XML验证工具，而且还要通过程序可视化软件报告出错地点，并给出简明准确的提示信息。

3 基于XML Schema的符号表构造与管理

这一节介绍了如何运用XML Schema描述编译器前端的词法分析的正则表达式，语法分析的产生式，语义分析的语法树，后端的目标机体系结构，最终实现符号表的构造和管理。

3.1 编译前端的XML Schema描述

设计编译器的第一个阶段包括词法分析。首先，要设计一个与词法表相匹配的XML Schema文件，对于词法的构成使用XML Schema的正则表达式实现。XML Schema具有简单和复杂数据类型，可以用来构造绝大部分程序语言的数据类型。然后对照建造XML格式的词法表。此外，词法表分析器所对应的文档要求标记化它们的标签、属性和数据。例如，语言的复杂数据类型可以使用XML Schema模型组(sequence,choice,all)来表示，可以用复杂类型的子元素构建更丰富的多层结构化的数据类型。

编译第二阶段要求生成一个语法分析器。它通过使用第一阶段的词法分析器，决定是否一个文档与XML最初描述的文法相一致。如所知，编译器的文法是由一系列产生式组成的，这种特征与XML Schema的描述语言Model Schema Language (MSL)[7]是非常类似的。一个根元素由上而下产生若干个子元素，根元素对应产生式的左端，子元素对应产生式的右端。语法树的文法产生式定义如下：

R→A| B|c|D|…

B→xy |M |N …

N→J|k…

其中大写字母表示元素，小写字母表述元素的属性，R代表根元素。其描绘文法的每个非终结符为一个XML元素。对于非终结符的每个产生式右边是一个可变的内容模型(想象成模型组)。不变的终结符既然不增加信息通常作为属性处理。不同的非终结符能被包装成终结符产生式来解决这个问题。

由于产生式的结构是符合XML Schema规范的，这使得其推导过程都是可验证的。因此XML Schema相关技术可以用来表示程序语言广泛使用的上下文无关文法，例如BNF范式和扩展巴克斯范式(EBNF)[8]。其文法推导过程中的文法表能够通过XML Schema定义，推导过程通过DOM解析XML实现。采用此方法的优点包括：1)语法描述完整，描述语义所用的方法都对应有属性格式定义；2)有较好的扩展能力；3)有利于实现文法驱动的编译功能一致性验证。

作为源程序的中间表示，抽象语法树不仅被许多编译器选作程序中间表示形式，用于编译的语义分析阶段，使翻译从分析中分离出来，而且在程序分析等领域也具有广泛的应用。GNU编译器GCC前端结果建立抽象语法树(AST)，产生的AST文件经过一定的格式转换可以变为XML文档[9,10]。然后创建其对应的XML Schema文档，这使得整个编译器前端都能够基于XML Schema来定义与描述，同时也便与编译器后端统一起来，支持目标机器描述。

3.2 描述目标系统体系结构

从中间代码到目标代码的转换与目标处理器相关。显然，编译器能够快速地被移植到不同的目标机器，采用的最直接的方法就是修改一些与目标机相关的后端文件。这些文件描述目标系统的体系结构，包括指令集、寻址方式、存储器以及其他硬件资源等。常规多目标设计方法通常有如下三种类型：开发者多目标、参数多目标和用户多目标[11]。现有的许多编译器多采用第一，第三种方法。参数多目标是在设计编译器的时候，专门设计若干用户可以改变的参数。通过修改这些参数，实现编译器到新目标机器的移植。但是这种方法通常限于某一类具有相似体系结构的目标机器[2]。

目标机后端文件若通过XML文档来描述，则很适合采用参数多目标方法。以GCC的ARM目标机体系结构描述文件为例，其中主要由三个文件组成一个特定机器的描述，嵌入式系统ARM系列的目标机器都可以用以下这三个文件来描述：(1)arm.h；(2)arm.c (3)arm.md。通过观察这三个文件的结构，我们可以依次实现其XML方式的转化表示。以arm.h为例，主要包括#define，typedef，extern，struct，if，do while这六种定义格式，在XML Schema的语法中能够找出这前四种定义格式的表达方式，它们分别是：元素的默认值和固定值；类型的枚举值；全局元素声名；复杂类型；后两种可以通过类似的XQuery查询语言实现。通过该方法产生的体系结构描述文件，在描述同类型的其它机器时，通常只需要修改类似的参数就可以达到相同的目的。

根据体系结构描述，查询模块能够在数据库中建立起寄存器描述表、操作域描述表等数据符号表。目标体系结构这些信息采用XML文挡描述和存储，借助一些XML开发工具，其很容易被不同的目标代码生成语言处理。

3.3 构建编译符号表

在编译过程中常用到的表格包括常量表、变量表(标识符表)、保留字表、数组表和过程信息表等。统称为符号表，又称为单词属性字表。在词法分析，语法分析，语义分析甚至目标机器描述阶段，所用到的词法表，LL(1)，LR等语法，语义分析表等各种表格都可以归结为符号表。编译的整个过程中，符号表被频繁地用来建立表项、查找表项、填充和引用表项的属性。因此符号表的设计和实现关系着整个编译系统的效率[12]。传统的符号表通常采用高级语言定义数据结构，如线性表、各种搜索树等。并且实现相应的插入、查找和删除处理，效率较高，缺点是不够透明和缺乏通用性。

这里可以使用XML Schema技术对符号表进行定义。符号表是包含有关子程序、变量等信息的数据库，每个符号表能够由若干个子表组成，每个子表由若干个表项组成，包括名字域和属性域两个部分。符号表的名字域描述了某个单词的名字，属性域是表项的值部分，描述了某单词的有关属性。因此可以使用XML Schema中的element及其attribute来定义符号表项，而且定义的符号表逻辑性强，自顶向下层次清晰，表项的名字与属性信息分开定义，便于处理。通过下节提到的DOM，XQuery等技术能够很容易地完成建立符号表、插入元素、查找操作等各种符号表相关的操作，同时对于使用者

来说，不必知道实现细节就能够进行符号表操作。通过XMLSPY5.0创建的XML Schema文档及对应的symbol.xml如图2所示。

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSPY v2004 rel. 2 U
(http://www.xmlspy.com)-->
<symbol xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Documents and
Settings\Administrator\桌面\新建文件夹\symbol2.xsd">
<operatoritem name="+" id="1"/>  <!--操作符项，id值需要约束检
查-->
    <operatoritem name="-" id="2"/>
    <symbolitem>add2supply</symbolitem> <!--标识符项-->
    <symbolitem>Add1one</symbolitem>
    <symbolitem>xy</symbolitem>
<!--其它符号表项省略-->
</symbol>
```

图2　XML符号表

创建的符号表对应的Schema文档symbol.xsd，它建立了标识符的正则表达式，验证对应XML实例的数据正确性，完整性以及是否满足模式规定的种种约束。最后加入操作符属性ID的唯一性，实现约束检查。内容如图3所示。

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="symbol">
   <xsd:complexType>
     <xsd:sequence>
     <xsd:element ref="operatoritem"
maxOccurs="unbounded">
       <!-- 对实际文档格式进行了简化 -->
     <xsd:complexType>
       <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="id" type="xsd:integer"
use="required"/>
     </xsd:complexType>
     </xsd:element>
        <xsd:element ref="symbolitem"
maxOccurs="unbounded">
        <!-- 标识符数目不限 -->
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
      <xsd:pattern
value="[(A-Z)|(a-z)]([A-Z]|[0-9]|[a-z])*"/>
          <!--标识符以A或a开头，字母数字组成 -->
            </xsd:restriction>
            </xsd:simpleType>
            </xsd:element>
        </xsd:sequence>
     </xsd:complexType>
     <xsd:unique name="EOneKey"> <!-- 操作符ID唯一性约
束 -->
        <xsd:selector xpath=".//operatoritem"/>
        <xsd:field xpath="@id"/>
     </xsd:unique>
   </xsd:element>
</xsd:schema>
```

图3　符号表的XML Schema文档

symbol.xml在Stylus Studio5.0实现标识符的查询处理，经XQuery查询可以得到可视化的XML标识符项。查询语句如下所示：

```
<symbol> {for $symbolitem in document("symbol.xml")//symbolitem
where $symbolitem ! ="xy"
return if ( $symbolitem ="add2supply")then <test/> else $symbolitem }</symbol>
```

4　中间代码的解析可视化与验证

文档对象模型(Document Object Model，DOM)是基于树型结构的API，是由W3C制定的标准接口规范。DOM依据XML文档的结构将其转换为树型结构的文档对象模型，用户通过对该对象模型的访问，可以动态地创建文档，遍历文档结构，对XML文档中的数据进行修改，移动，删除和插入等操作。此外W3C于1998年推出了XQuery规范，到2004年底Full-Text正式规范完成制定，它可以对不同应用的XML数据进行数据处理、数据转换和检索。总之，这些功能的实现，使得抽象语法树等编译过程中的中间表示可视化。除了AST以外，其它的语义信息也能实现可视化[13]。商业XML编辑器如XMLSpy，或者开源图形化工具例如GraphViz，Vcg都能实现XML中间代码可视化。另一方面，由XML构造编译过程，其XML文档数据以及XPath表达式能通过模型检验技术进行验证，以便进行错误处理。例如文[14]已经实现了将XML Schema文档及其XPath查询语言转化为SPIN model checker的Promela输入语言，从而实现了XML软件系统的形式化验证。

结束语　本文主要讨论了通过XML Schema等技术实现编译器中间代码符号表生成的过程。符号表通过XML Schema技术重建后，可以利用一些XML工具对编译符号表统一管理，使得该编译符号表通用化而且可以验证。从而使得设计的编译中间代码容易维护，而且能够移植，同时该方法也有利于后端目标语言的生成。由于XML已经成为多种语言环境的数据描述格式，基于XML Schema的中间代码很容易被不同环境下的编译器采用。随着计算机运算速度和存储容量的飞速提高，使用XML工具而不用低级或中间级的编程语言在开发时间上也是一个优势。今后的工作是实现具体设计，验证该方法的实用性和可靠性。

参考文献

1　Aho A V, Sethi R, Ullman J D. Compilers: Principles, Techniques, and Tools. Addison-Wesly, 1986

2　任小西，李仁发，张克环，郭媛妮. 一种基于多目标设计方法的嵌入式编译器技术. 计算机应用，2004，24(2)：165～167

3　Germon R. Using XML as an Intermediate Form for Compiler Development. In: Proceedings of IDEAlliance XML Conference & Exposition 2001. http://www.idealliance.org/papers/XML2001/papers/html/03-05-04.html, December 2001

4　Biron P, Malhotra A. XML Schema part 2: Datatypes, W3C working draft. http://www.w3.org/TR/1999/xmlSchema-2, May 1999

5　Walmsley P著. XML模式权威教程M. 北京：清华大学出版社，2003

6　张昱，李凡. 用Xerces-J进行基于XML Schema的XML局部验证. 小型微型计算机系统，2005，26(8)：1369～1373

7　Brown A, Fuchs M, Robie J, et al. MSL a modelfor W3C XML Schema. In: Proc. of 10th World Wide Web Conf. (WWW), 2001. 191～200

8　郭德贵，刘磊，吴万春，等. 基于超文法的扩展文法转换技术. 吉林大学学报(理学版)，2006，44(1)：73～77

9　刘文伟，刘坚. 一个重建GCC抽象语法树的方法. 计算机工程与应用，2004，18：125～128

10　Antoniol G, Di Penta M, Masone G, et al. XOgastan: XML-oriented GCC AST analysis and transformation. In: Proceedings of the Third International Workshop on Source Code Analysis and Manipulation (SCAM'03). IEEE, 2003

11　Leupers R. Compiler Design Issues for Embedded Processors. IEEE Design&Test of Computers, 2002, 19(4)

12　邱坚，蒋维杜. 语言编译论域面向对象系统分析. 计算机工程，2004，30(2)：92～94

13　Yu Yijun, Beyls K, D'Hollander E H. Performance visualizations using XML representations. In: Proceedings of the International Conference on Information Visualization, v 8, Proceedings - Eighth International Conference on Information Visualisation, IV 2004. 795～800

14　Fu Xiang, Bultan T, Su Jianwen. Model Checking XML Manipulating Software. ISSTA'04, Boston, Massachusetts, USA, ACM, 2004