DOI:10.16208/j.issn1000-7024.2005.04.046

软件测试自动化静态分析研究

文昌辞， 王昭顺
(北京科技大学，北京 100083)

摘 要：为了加快软件测试的速度，减少人力投入，可以采用测试自动机对软件进行自动测试。测试自动化静态分析工具是一种软件，可以利用它对程序的源代码进行分析，自动测试应用系统的很多方面。在软件测试中，静态分析工具并不执行所测试的程序，只是扫描所测试程序的正文，对程序的源代码进行分析，它类似于编译程序中的词法分析和语法分析，但工作量远不止于此。

关键词：软件测试自动化；扫描；静态分析工具
中图法分类号：TP311.56 **文献标识码**：A **文章编号**：1000-7024 (2005) 04-0987-03

Static analysis research of software automatic test

WEN Chang-ci, WANG Zhao-shun
(Beijing University of Science and Technology, Beijing 100083, China)

Abstract: Aiming to accelerate software test and reduce manpower expense, automatic test machine may be used to work automatically. Automatic test analysis tool is a kind of software, which can analyze the source code and automatically test the system in many aspects. In the test procedure, it will not execute the program, but only analyze the source code statically like lexical analysis tool and syntax analysis tool, and its workload is more.

Key words: software automatic test; scan; static analysis tool

1 引 言

静态分析工具一般由 4 部分组成：语言程序的预处理器、数据库、错误分析器和报告生成器。预处理器把词法分析和语法分析结合在一起以识别各种类型的语句。它把源程序划分为若干程序模块单元(如主程序和子程序)，同时生成包含变量使用、变量类型、标号与控制流等信息的许多表格。有些表格是全局表，它们反映整个程序的全局变量信息，如模块名、函数及过程调用关系、全局变量等。有些表格是局部表，它们对应到各个模块，记录模块中的各种结构信息，如标号引用表、分支索引表、变量属性表、语句变量引用、数组或记录特性表等。这些信息表格都被放入数据库中，以方便被测代码的信息查询和信息更新。不少测试工具有专门设计用来存放各种信息的数据库，通常以命令语言的形式来作为查询语言。错误分析器在用户指导下利用命令语言或查询语言与系统通信进行查错，并把检查结果通过表格的形式输出。

2 设计静态分析工具

静态分析工具通常要对源代码中的变量进行检查，此目的是检查该变量是否在使用前已定义，是否在定义后未被使用过。简单的实现方法是对变量的定义进行记录，对使用次数进行计数，这需要在词法分析和语法分析的基础上再建立

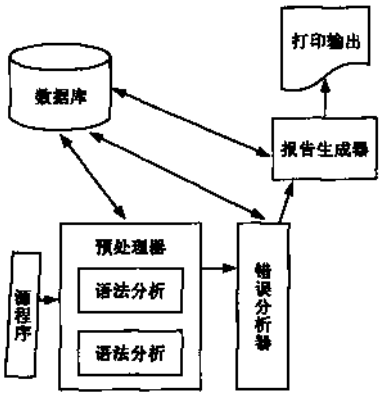

图 1 静态分析工具的运作流程

3 张表。第 1 张表对变量的定义进行记录，它包括该变量的变量名、在第几行第几句代码中对变量进行了定义、用什么类型定义了变量。第 2 张表对变量的使用次数进行计数，它包括该变量在第 1 张表中对应的入口、目前为止它的使用次数、最近一次在第几行第几句代码中对变量所代表的存储单元进行了操作。第 3 张表记录源代码中的变量冗余或者未定义的信息。

静态分析工具还检查模块接口的一致性，主要是检查子程序调用形式参数与实际参数的个数、类型是否一致，输入输出参数的定义和使用是否匹配。这通过把所有的模块名存在

收稿日期：2004-05-21。
作者简介：文昌辞 (1980-)，男，苗族，湖北来凤人，硕士，研究方向为计算机软件；王昭顺，男，副教授，博士，研究方向为计算机软件。

一张表中并且为每个模块设置一个接口调用表来实现。该表中记录了模块接口的形式参数的类型、排列顺序以及形式参数的缺省值，还记录了输出参数的类型。检查时只需挨个匹配即可，若实参未定义则将此错误信息存入数据库中用于存放出错信息的表中。

静态分析工具还检查在逻辑上可能有错误的结构以及多余的不可达的程序段，如交叉转入转出的循环语句、为控制循环变量赋值、存取其它模块的局部数据等。还要找出变量错误可能会影响到哪些语句，影响到哪些其它变量等。这通过建立“变量、语句交叉引用表”、“子程序调用顺序表”、“公用区、子程序交叉引用表”等表格，在进行分析的时候不断更新表格的信息来解决。

2.1 词法分析

文本文件中记录了所有的保留字，保留字间用空格分隔开。在此引入一组全局变量和过程，作为实现词法分析的基本成分。

(1) character 字符变量，存放最近读进的源程序字符。

(2) token 字符数组，存放构成单词符号的字符串。

(3) getchar(),子程序过程，把下一个输入字符读到character中，把源程序中的读入指针向前移动一个字节位置。

(4) getbc(),子程序过程，检查 character 中的字符是否为空白、换行标记或者结尾标记。若是，则调用 getchar()直到 character 中进入一个非空白也非标记字符。

(5) concatenation(),子程序过程，把 character 中的字符连接到 token 中的字符的后面。

(6) letter(),布尔函数，判断character中的字符是否为字母。

(7) digit(),布尔函数，判断 character 中的字符是否为数字。

(8) reserve(),对 token 中的字符串查找保留字表，回送相关信息。

(9) retract(),把读入的源程序字符的向前指针回调一个位置，character 变为空白符。

(10) buildlist(),为一般标识符和常数建立符号表。

(11) return_info(),收集并返回必要的信息供语法分析。

2.2 输入自定义文法产生式

将文法大致上分成 3 个部分，但是 3 部分之间有交叉，在构造语法分析表时将所有的产生式集中在一起，构造一个大的语法分析表。这些产生式由人工输入存放在文本文件中。运行程序时可通过设置选项根据这些产生式自动生成 LL 语法分析表，并且在内存中和文本文件中都保存一份该语法分析表；也可从文本文件中读入以前运行程序时构造的LL 语法分析表到内存。以下是自定义文法产生式的一个示例。

2.2.1 表达式文法

```
expr->term A
A->+ term A | - term A | = term A|>= term A|<= term A|< term A|> term A|! term A|null
term->factor B
B->* factor B|/ factor B|null
Factor->id E1|num|( expr )
```

2.2.2 简单句文法

```
stmt->type Z
Z->;|,Z|= E S1|E1 S1
stmt->F = E;
E->expr|* F|& id E1
E1->[ E ] E1|null
type1->int|char|float|double|
type->const type1|auto type1|register type1|static type1|type1
F->id E1|* F
Z1->Z|S1|stmt
```

2.2.3 复杂句文法

```
stmta->F = E bb|null
bb->,F=E bb|null
com->while ( expr ) com|do com while ( expr );
com->for ( Z1 expr;stmta) com
com->if ( expr ) com C
C->null|else com
com->{ D }|stmt fu|null|; com
D->null|com D
fu->null|com
start-> M S
M ->#M1|null
S->main ( ) {combine }
combine->com combine|null
M1->define id expr M|include " Y " M
Y->id Q|num Q|. Q
Q->null|Y
```

这里采用的是自顶向下的 LL 语法分析法。以上的自定义文法已消除左第归、提取公因子，并且在定义中保证了表达式运算中优先级高的先运算，优先级低的后运算，从而不用再定义算符优先文法。但是有一个问题，就是二义性的问题，不过由于在 C 语言语法中，要求 else 和最近的 if 匹配，所以设置一下语法分析表中的优先级就可以解决这个问题，即在语法分析过程中，栈顶为 com，输入出现 if 时，将“if (expr) com C”压入栈中，并且在语法分析树中当前节点添加子节点 if、(、expr、)、com、C。

2.3 建立语法分析表

计算每个非终结符的 FIRST 集合和 FOLLOW 集合，根据它们建立语法分析表。

在建立 LL 语法分析表的过程中，会发现冲突，但是可以根据实际中 C 语言的语法规定设置优先级，消除冲突。此文法可以识别#include 、#define，可以识别 if...else、for、while、do、简单语句以及空语句“;”的交叉嵌套。

2.4 语法分析错误处理

在构造语法分析表的过程中，正常的产生式编号大于、等于 1。那些没有填入产生式编号的入口应填入错误处理代号。这里，规定可以填入 0 或者 -1。如果某个终结符能推导出 null，那么与它的 follow 集合元素相对应的入口在前面的过程中就已经填入了产生式编号，但是还可能有一些入口没有填写，处理办法是把该非终结符对应的那些入口填入 0，在分析中若读到产生式编号为 0，则标志源代码中少了语法元素，此时应该出栈，继续分析。把剩下的没有填入编号的入口都填入 -1，它

也标志着源代码中少了语法元素，只不过该错误比产生式为0的错误要严重。在记录分析到的错误时，规定了一类最小区域，在该区域内若发现错误，只记录一次。这可避免由于一个错误而记录多次的情况发生。具体设计如下：

(1) 产生式编号大于0时，标志当前不处于寻找同步集合元素阶段，标志本区域为未记录过出错信息，即是说明语法分析没有发现错误或者分析恢复正常了，此后若发现错误，则要记录。

(2) 产生式编号等于0时，若本区域记录过出错信息，则不再记录该错误信息，否则要记录。还要标志当前不处于寻找同步集合元素阶段。

(3) 产生式编号大于等于1时，若记录过出错信息，则不记录，否则要记录出错信息：①若当前处于寻找同步集合元素的阶段，则保持输入不变，出栈直到栈顶与输入相匹配成同步，匹配成同步后，标志不处于寻找同步集合阶段；②若当前不处于寻找同步集合元素的阶段，则看输入是什么，若输入是“{”、“}”、“;”、“(”、“)”则出栈，读取下一个输入，进入寻找同步集合元素阶段。若为其它的输入，则栈不变，继续输入。

以上的处理可以使得在一个由分号分隔的语句内、一个表达式内、在一段比较小的由大扩号扩起来的代码段中最多只记录一次出错信息。

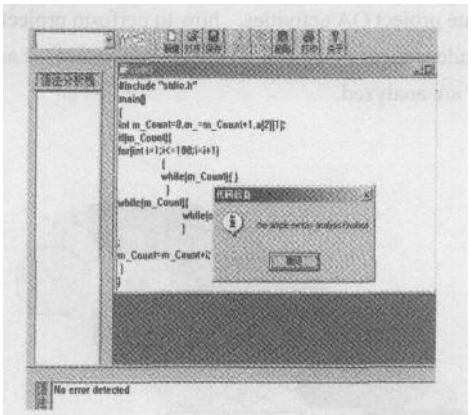

图2 未发现语法错误的分析实例

2.5 用VC++实现

在MFC Application Wizard自动生成多文档结构之后，还定义以下一些类。

ClexicalRecordset：通过它往数据库中写入词法分析的信息。

CSyntaxlRecordset：通过它往数据库中写入语法分析的信息。

Coutput：负责输出语法分析的错误信息。

CtreeNode：在内存中建立语法分析树，以便进行下一步的语义分析。

CWorkSpace：输出在分析过程中语法分析栈的信息。

Cstack：定义栈的结构，它与CtreeNode联系比较紧密。

CwenToolBar：自定义的多功能工具栏，包含了所有的自定义功能。

CstackMember：自定义的栈元素结构。

CBootWindow：启动窗口用来延时，在延时期间若发现数据库则建立与数据库的连接。

CDeleteLeftRecursion：在转换产生式时消除左递归。

CAbstractCommonFactor：在转换产生式时提取公因子。

CLlTableCreate：自动生成LL语法分析表，内含计算FIRST集和FOLLOW集的成员函数。

CLlAnalyzeBoot:LL文法分析的驱动模块，它从分析表中读入语法分析入口的相关信息，进行相应的处理。

CLrTableCreate：自动生成LR语法分析表。

CLrAnalyzeBoot:LR文法分析的驱动模块，它从分析表中读入语法分析入口的相关信息，进行相应的处理。

CLogicalJudge：语义分析，检查变量是否在使用前已定义，是否在定义后未被使用过，检查在逻辑上可能有错误的结构以及多余的不可达的程序段。

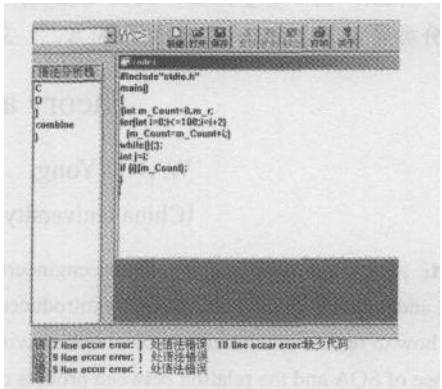

图3 发现语法错误的分析实例

3 结 论

静态分析工具在软件测试中的投入运用，可大量节省人力的投入。结合词法分析和语法分析给出的信息，静态分析工具可以检查所测程序违反编程标准的错误，如模块大小、模块结构、注释的约定、各种类型源语句的出现次数等。结合语义分析，它还能完成对一些特性的统计：函数过程引用情况、标识符使用的交叉索引、标识符在每个语句中使用的情况、任何输入输出数据都执行不到的孤立代码段、公共变量与局部变量的各种统计。

参考文献：

[1] 郑人杰, 殷人昆, 陶永雷. 实用软件工程 [M]. 第2版.北京:清华大学出版社, 1998. 255-257.

[2] Kenneth C Louden. 编译原理及实践[M]. 北京:机械工业出版社, 2000. 105-142.

[3] 杜淑敏, 王永宁. 编译程序设计原理[M]. 北京:北京大学出版社, 1988. 42-50.

[4] Jeff Prosise. MFC Windows程序设计[M]. 北京:清华大学出版社, 2001.

[5] Stanley B Lippman, Josee Lajoie. C++ Primer中文版[M]. 北京: 中国电力出版社, 2002.

[6] 萨师煊, 王珊. 数据库系统概论 [M]. 北京: 高等教育出版社, 1983.

[7] Rafe Colburn. SQL实用全书[M]. 北京: 电子工业出版社, 2001.