程序断言动态检测工具的设计与实现 *

刘树锟[1], 阳小华[2], 陈继锋[1], 彭 浩[1]
(1. 湖南涉外经济学院 计算机科学与技术学部, 长沙 410205; 2. 南华大学 计算机学院, 湖南 衡阳 421001)

摘 要: 重点研究了断言的动态检测方法与检测过程,并在关系数据库理论的基础上实现了交互式的断言动态检测工具,即 TDDPA。该工具具有动态检测程序断言等功能,并通过将运行轨迹收集到数据库中来实现检测到的各种断言形式分析,说明了 TDDPA 总体设计结构及实现过程。结果证明 TDDPA 能更方便有效地发现程序中所蕴涵的断言。

关键词: 程序断言; 动态检测; 软件可靠性; 条件查询
中图分类号: TP311 **文献标志码**: A **文章编号**: 1001-3695(2009)11-4140-03
doi:10.3969/j.issn.1001-3695.2009.11.040

Design and implementation of tool for dynamical detecting program assertion

LIU Shu-kun[1], YANG Xiao-hua[2], CHEN Ji-feng[1], PENG Hao[1]
(1. *Dept. of Computer Science & Technology, Hunan International Economics University, Changsha* 410205, *China*; 2. *Dept. of Computer, University of South China, Hengyang Hunan* 421001, *China*)

Abstract: This paper mainly discussed the process and methods of detecting program assertion. Presented a new interacted assertion detecting tool based on RDBMS, which was named TDDPA. It supported such functions as dynamical detecting program assertion and giving query conditions by user, providing user a more convenient interface. TDDPA analyzes and reports assertion by collecting running trace to database, which is different from traditional one. The base structure and implementation of the tool are illustrated in details. Experiments show that TDDPA is accurate and convenient in detecting program assertion.
Key words: program assertion; dynamically detect; reliability of software; condition query

从出现迄今虽然只有短短的几十年,但是软件在当今社会中已经成为最重要的系统之一。在现实世界的各个领域,从航天器、核电站到石油勘探开采、天气预报再到生产控制、计划调度再到网络游戏、信息获取等,软件系统都发挥着至关重要的作用。因此,软件程序的质量也成为人们广泛关注、高度重视的热点问题,特别是那些具有高可靠性要求的应用,如航天器、核电站控制等[1]。

借助于断言以提高程序的可靠性并不是一件全新的事物。许多程序设计语言,如 C 和 C++ 均有名为 assert 的宏,用于书写断言,它们经过预处理后被扩展为对应的语句嵌入在程序中,从而实现简单的运行监测;而 Java 也内置了断言语句。Eiffel 语言的成功[1],带来了一系列针对其他语言的类似研究和工具,如 C++[2]、.NET[3]、Java[4] 等。但是,由于它们通常采用程序设计语言书写断言,存在一些共同缺陷,如表达能力不足、缺乏量词等词汇;抽象程度不够,依赖具体的实现语言等。为了克服已有程序员手写断言等方式带来的缺点,比如,只能检测一些直观的断言形式并且断言发现技术十分呆板,很难根据实际需要定义新的断言表达式[5],并为进一步开展基于合约的程序断言研究提供支具(tool of dynmical detecting program assertion,TDDPA)。在 TDDPA 中,采用断言自动检测的方式,为用户提供灵活的断言检测机制,不仅可以自动生成一系列直观的断言形式,还可以根据用户实际需要定义新的断言表达式。

1 程序断言动态检测过程

程序断言检测的基本流程如图 1 所示。理论上可以在程序的任意地方检测、发现程序断言(程序不变量),只要在相应的位置上插入跟踪代码即可。但是实际上,断言检测主要在程序方法的入口和出口进行。其中入口断言称为过程的前置条件,出口断言称为过程的后置条件[8]。类不变量是被附加到了类暴露例程的前置和后置条件上。但从根本上讲,类不变量是一种刻画类和类实例的基本的一致性和完整性的方法。类不变量是应用到整个类的,它描述一种属性,类的每个实例在该属性对外部可见的任何时候都必须确保它是一致的。由图 1 可知,断言动态检测过程一般分为四个主要步骤:a)源程序编配;b)测试用例集生成;c)在测试用例集上运行程序,并实时地收集运行轨迹;d)分析收集的程序轨迹进行断言提取。一般而言,程序编配的任务是对程序进行适当的处理,为程序运行轨迹收集奠定基础,源程序编配过程的实质问题就是在程序代码中插入大量的描述轨迹的代码,从而通过编配程序的运行,记录它的运行轨迹。其主要的工作是确定观测点和观测变量。在断言检测过程中程序运行是必不可少的。那么,如果要很精确地检测程序断言信息,必须多次运行程序,即不同的方法、函数要通过多次执行,以遍历更多的路径。但是,如果做到

收稿日期: 2009-03-04; **修回日期**: 2009-04-09 **基金项目**: 湖南省教育厅基金资助项目(08C516);湖南省自然科学基金资助项目(05JJ30117);湖南省教育厅重点基金资助项目(07A034)

作者简介:刘树锟(1979-),男,河北沧州人,讲师,硕士,主要研究方向为软件工程、数据库技术(llsskkllsskk@163.com);阳小华(1963-),男,湖南衡阳人,教授,博导,博士,主要研究方向为软件工程、数据库技术、信息检索技术等;陈继锋(1966-),男,湖南长沙人,教授,博士,主要研究向为软件工程;彭浩(1978-),男,湖南长沙人,讲师,硕士,主要研究方向为软件工程.

全面地描述程序运行过程,那么必须要精心设计测试用例集合。目前测试用例的选择有两类典型的方法,即白盒法和黑盒法。白盒法根据路径、分枝等覆盖准则选择测试用例;黑盒法使用等价类、边界条件等技术决定测试用例。运行轨迹收集就是用适当的测试用例集运行程序,收集并保存位于观测点上的观测变量的运行值。在每一次测试中,当程序运行经过预定的观测点时就将产生一条检测记录,一次运行产生的全部检测记录序列,组成程序的一个轨迹。运行轨迹收集需要考虑的主要问题是轨迹记录的格式与存储方法,它们直接影响系统的效率[5,6]。

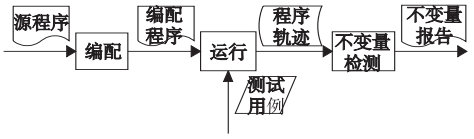

图 1　断言检测流程图

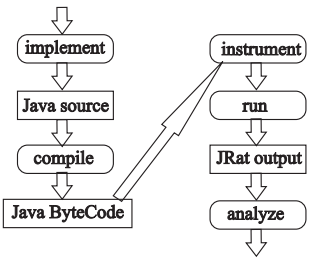

图 2　编配过程图

目前的断言提取方法主要是基于模式提取的,即首先定义好比较常见的程序断言类型,然后根据收集的程序轨迹文件中的各个变量在不同时刻的不同状态值,依此把各个变量进行比较。如果发现变量关系与模板中断言匹配,则报告出此断言;否则继续比较,直到把所有的变量值比较完为止。

2　TDDPA 需求描述

为了使 TDDPA 方便实用(该工具主要针对 Java 程序进行断言检测),更好地发挥断言自动检测方法的优势, TDDPA 的需求如下:

a)能够按照用户的具体实时要求构造不同条件的断言检查条件,能够和用户进行交互访问。

b)支持的断言的种类,包括前置条件、后置条件、类不变量、循环不变量。

c)断言应具有较强的描述能力,断言中支持全称量词、存在量词,并且能够表达蕴涵关系[7]。

d)用户可以选择是否进行断言检查以及断言检查的级别。

e)断言的检查时机比较固定,一般情况下,类不变量在方法的出口、入口和抛出异常的地方进行检查,构造方法只需在出口处检查类不变量;前置条件在声明该前置条件的方法的入口处进行检查;后置条件在声明该后置条件的方法的出口处进行检查[7]。

f)不违反合理断言时,断言对程序不产生任何可见的影响;断言被违反时,抛出异常,中止程序,所抛出的异常要能够准确地定位异常发生的位置。

g)断言以注释的形式出现在 Java 源文件中, Java 程序在执行过程中不受断言的影响。

3　TDDPA 的设计

TDDPA 原型系统在 Windows XP 平台下基于 C#语言,结合 SQL Server 2000 数据库完成。TDDPA 系统主要分为四个主体类来实现,分别为轨迹文件收集存储类、查询条件定义类,轨迹文件分析类、程序断言提取类。下面从轨迹文件存储、程序断言表达、程序断言提取等几个方面来分析原型。

3.1　程序轨迹文件存储

程序动态运行的过程实质上可以认为是状态转换的过程。比如,在此以 Java 程序为例进行分析。Java 程序主要是一个层次结构,即程序包→类→方法→变量。所以,在分析程序的过程中就是在分析上述层次结构的具体信息。因此在记录动态状态信息时也要按照层次结构收集。如果按照程序的结构层次来记录程序的运行状态信息,对于将来程序内部各个层次关系的分析是很有利的。在数据库中定义类关系(关系名称为程序中每一个类的名称)、方法关系(关系名称为程序中每一个方法的名称)、变量关系(关系名称为程序中每一个变量的名称)。在 Java 程序中对于变量的分类有多种方式,在此记录状态信息时以全局变量(集合类型、非集合类型)、局部变量(集合类型、非集合类型)来分析。程序中包含多少类就会在数据库中动态生成多少张类关系表,同样的有多少个方法也会对应多少方法关系表,依此类推到变量。为了准确地进行程序断言查询,必须实时记录程序的运行轨迹,并要设计合理的数据库存储格式。运行轨迹收集以关系数据库为基础进行,所有的观测记录被保存在轨迹数据库中,观测点的记录对应于数据库中的表。对于复杂的数据类型,如数组、记录、对象等,采用主表关联子表的方式来表示。运行轨迹收集以 SQL 数据库为基础进行,所有的观测记录被保存在轨迹数据库中。

3.2　断言的表达形式

TDDPA 所支持的程序断言的种类包括:前置条件、后置条件、类不变量、循环不变量等。程序断言的主要成分是程序不变量表达式。原型中定义了六种程序断言表达式:

a)简单类型。这种类型是指变量或变量所构成的函数与常量之间通过关系运算符直接作用得到,其形式化表达为 pR_1c。其中 p 为一变量,c 为一常数,而 R_1 为关系运算符。

b)复杂类型。这种类型是指变量 p 所构成的函数与常量之间通过关系运算符作用得到,其形式化表达为 $f(p)R_1c$。其中 $f(p)$ 为变量 p 通过各种其他的运算得到的函数,c 为一常数,而 R_1 为关系运算符。

c)组合类型。由一个或多个简单或复杂类型的逻辑表达式程序断言通过逻辑关系组合而成。其中逻辑关系包括"&&"和"||"。例如:$p>1 \| p<-1$、$p<3 \| \cos(p)>0$ 等。其形式化表达式为 $pR_1c/f(p)R_1cR_2pR_1c/f(p)R_1c$。其中 R_2 为除"?:"之外的布尔逻辑运算符。

d)全称量词表达式(描述一个集合中的所有元素均满足某个条件)。

e)存在量词表达式(描述一个集合中存在满足某个条件的元素)[7]。

(f)蕴涵表达式(描述当某一条件满足时,另一个条件必须满足)[7]。

d)~f)三种断言表达式增强了程序断言的描述能力,使得用一般的布尔表达式很难表达的意思能够较为简洁地表达出来。

3.3　程序断言提取

TDDPA 工具对各种表达式的处理方法具有许多相同之处。因此设计一个基类 BaseExpressionclass,各种基本表达式都直接或间接地通过 BaseExpressionclass 类检测。各种基本表

达式在 Java 程序和新定义的程序断言中都有可能出现。而程序断言的定义中存在一些与全称量词、存在量词及蕴涵关系有关的表达式,这些是断言中所特有的。为了便于处理,从各种程序断言表达式中提取出一个共同基类 InavariantsExpressionclass。在六种程序断言表达式中,对于一般的布尔表达式,不必设计一个新类来处理这种情况;对于含有全称量词和存在量词的表达式,两者除了关键字“∀(x)”和“∃(x)”不同之外,表达式的结构完全相同,只需设计一个类就可以处理这种表达式;对于蕴涵关系表达式,新设计一个类对其进行处理。另外对于用户临时指定的断言查询条件,只需指定临时的一个类来处理,主要是处理一些相对比较复杂条件的 SQL 语句查询。

4　TDDPA 的实现

针对 3.2 节所表述的简单程序断言的各种体现形式,可以用存储在数据库中的轨迹数据作为断言提取的数据基础。程序断言的检测基本上是基于数据库技术的,即以关系数据库管理系统(DBMS)的 SQL 查询功能为基础加上高级程序语言编程技术来求解。本文中的 TDDPA 断言检测工具是在. NET 平台上用 C#语言开发完成的。整个工具采用窗体界面,如图 3 所示。

图 3　运行状态图

为了实现程序编配在 TDDPA 中采用了开源的 Java 程序分析工具——JRat[6]。JRat 是一个 Java Runtime 分析工具包。它可以让开发者更好地理解 Java 程序运行时的状态,同时收集 Java 程序动态执行数据,并且提供一个完整的框架来快速简便地分析这些收集到的数据。在此采用 JRat 程序分析工具来记录需要的程序运行过程中的类、方法、变量的值与状态。JRat 的简单流程如图 2 所示。程序编配是程序断言动态生成技术的关键组成部分,只有通过精确的编配验证,在以后的程序断言提取工作才会有意义。否则提取的程序断言是不精确的或者说是没有实际数据依据的。在检测程序断言的整体过程中 TDDPA 采用的是开放性质的语句池机制(这与进行非函数依赖程序断言检测的机制类似)。这种机制带来很多优点,比如对用户是开放的等。这是由 SQL 语句池本身开放属性所决定的。将事先定义好典型性质的断言检测 SQL 语句存储在语句池中,当进行轨迹文件分析时,可以从语句池中调用 SQL 语句。如果没有用户感兴趣的断言查询条件,那么用户可以根据自己的意愿,随时构造 SQL 语句然后存入语句池中(因为语句池是对用户开放的,用户可以随时把 SQL 语句存入语句池中)。这样用户就可以根据自己添加的查询条件进行断言提取。

5　结束语

目前程序断言检测工具有美国 MIT 程序分析组开发的程序不变量动态分析工具 Daikon 以及 Diduce。本文所描述的工具 TDDPA 从检测的灵活性分析,该断言检测方法检测方式灵活,不囿于具体的模式限制,可以随时指定查询条件。从断言检测形式来讲,检测的条件是根据查询条件来确定的,所以没有模式经验匹配的呆板性。TDDPA 较 Daikon 和 Diduce 在断言检测的易用性、交互性和使用灵活性上得到了改善,更有针对性。

参考文献:

[1] JEZEQUEL J M, MEYER B. Design by contract: the lessons of Ariane[J]. Computer, 1997, 30(1): 129-130.

[2] EDWARD S H, SITARAMAN M, WEIDE B W, *et al*. Contract-checking wrappers for C ++ components, TR RSRG-04-02[R]. South Carolina: Department of Computer Science, Clemson University, 2004.

[3] ARNOUT K, SIMON R. The . NET contract wizard: adding design by contract to language other than Eiffel[C]// Proc of TOOLs 39. [S.l.]: IEEE Computer Society, 2001:14-23.

[4] KRAMER R. iContract-the Java design by contract tool[C]//Proc of Technology of Object-oriented Language (TOOLS 26). 1998:295-307.

[5] SAFF D, ARTZI S, PERKINS J H, *et al*. Automatic test factoring for Java[C]//Proc of the 21st Annual International Conference on Automated Software Engineering. 2005:114-123.

[6] JRat the Java runtime analysis toolkit [EB/OL]. [2009-03-04]. http://jrat. sourceforge. net/flow. html.

[7] 单锦辉,姜瑛,刘江红,等. 基于合约的构件易测试性设计支撑工具的设计与实现[J]. 北京大学学报:自然科学版, 2005, 41(5): 815-819.

[8] ERNST M D. Dynamically discovering likely program invariants[D]. Washington DC: Department of Computer Science and Engineering, University of Washington, 2000.

(上接第 4139 页)

参考文献:

[1] DASILVA D D, PATON N W. User interface modeling in UMLi [J]. IEEE Software, 2003, 20(4): 62-69.

[2] MOLIN P J, MELIA S, PASTOR O. Just-UI: a user interface specification model [M]. London: Kluwer Academics Publisher, 2002: 63-73.

[3] 冯仕红, 鹿旭东, 万建成. 基于模型的多设备用户界面设计[J]. 通信学报, 2006, 27(11): 55-59.

[4] 秦严严, 田丰, 王晓春,等. 以交互为中心的 Post-WIMP 界面模型[J]. 软件学报, 2006, 17(4):691-702.

[5] GONZALEZ-RODRIGUEZ M, MANRUBIA J, VIDAU A, *et al*. Improving accessibility with user-tailored interfaces [J]. Applied Intelligence, 2009, 30(1): 65-71.

[6] 邓集波, 洪帆. 基于任务的访问控制模型[J]. 软件学报, 2003, 14(1):76-82.

[7] 张建, 孙吉贵, 李妮娅,等. 工作流系统中一个基于多权角色和规则的条件化 RBAC 安全访问控制模型[J]. 通讯学报, 2008, 29(2):8-16.

[8] VANDERDONCKT J. A MDA-compliant environment for developing user interfaces of information systems [C]//Proc of the 17th Conference on Advanced Information Systems Engineering. 2005: 16-31.

[9] 李凡, 李梦, 李京. 基于模型的 Web 页面自动生成系统 PAGS [J]. 计算机工程与应用, 2006, 42(27): 84-88.

[10] 马永杰, 杨志民. 面向用户的可复用系统菜单权限的设置[J]. 计算机应用研究, 2008, 25(4): 1108-1110.

[11] 姚婷,胡业发. 制造企业内容管理系统中基于角色和任务访问控制的研究[J]. 计算机应用研究, 2008, 25(12): 3575-3577.