计算机科学与技术学院

毕业设计（论文）

文献综述

设计（论文）题目： 基于动态符号执行的源程序分析

专 业 班 级： 计算机科学与技术1002班

学 生 姓 名： 徐华福 学号： U201014264

指 导 教 师： 李国徽

2014年 3月 1日

# 一、前言

## 1.1 写作目的

本课题研究的是“基于动态符号执行的源程序分析”。撰写本文的目的是整理动态符号执行相关的文献资料，归纳国内外前人在动态符号执行和源程序分析与检测上的研究工作，以参考和借鉴。

## 1.2 相关概念

本课题最终的目的是对源程序分析，因此必然会涉及到程序分析技术。程序分析是指对计算机程序行为进行自动分析的过程，主要可以分为静态分析技术和动态分析技术两类。

程序静态分析（Program Static Analysis）是指在不运行代码的方式下，通过词法分析、语法分析、控制流分析等技术对程序代码进行扫描，验证代码是否满足规范性、安全性、可靠性、可维护性等指标的一种代码分析技术。常用的静态分析技术的实现方法有常见的实现技术有模型检验、数据流分析、抽象解释、谓词转换、定理证明、类型推导、符号执行等。

动态分析技术是通过运行程序，在输入不同的情况下跟踪程序执行，以检测分析程序的程序分析技术。与静态分析技术相比，动态分析技术侧重于对程序功能的测试。静态分析技术则是对程序代码的进行分析，检测出程序中存在的缺陷和不合理之处。

符号执行是静态分析技术的一种。符号执行是指在不执行程序的前提下，用符号值表示程序变量的值，然后模拟程序执行来进行相关分析的技术。符号执行技术使用符号值表示变量，来模拟程序执行，得到的变量的值是由输入变量的符号值和常量组成的表达式。

动态符号执行是以具体数值作为输入来模拟执行程序代码，与传统静态符号执行相比，其输入值的表示形式不同。动态符号执行使用具体值作为输入，同时启动模拟执行器。然后，从当前路径的分支语句的谓词中搜集所有符号约束。然后修改该符号约束内容构造出一条新的可行的路径约束，并用约束求解器求解出一个可行的新的具体输入，接着符号执行引擎对新输入值进行一轮新的分析。通过使用这种输入迭代产生变种输入的方法，理论上所有可行的路径都可以被计算并分析一遍。

## 1.3 综述范围

综述主要涉及程序分析、软件测试、多线程程序等方面。

引用的文献时间范围为：1976-2010年。

# 二、主体部分

程序中一个漏洞可能带来无法估量的损失，特别是在广泛使用的协议或者软件中。比如，OpenSSL“Heartbleed”安全漏洞危及到成千上万的服务器，很多知名的网站的安全都受到威胁，影响到广大网民的信息和财产安全。因此，软件的安全性越来越受到人们的重视，如何检测软件中存在的安全问题也逐渐成为软件工程领域的研究热点。

一般说来，程序验证要求通过推理或者穷举的手段来判定程序的行为是否符合规约。由于要涵盖所有可能情况，而程序设计语言的复杂性使得程序的复杂性随着程序尺寸的增大呈指数级增长，同时证明任一程序正确与否本身是一个不可判定问题，因此程序验证目前只用于证明一些关键的核心模块的正确性而没有得到更广泛的应用。就目前而言，程序验证方法虽然可以保证软件质量，但是往往需要有一定经验的用户花费相当多的时间，因而并不一定能提高软件的生产率[1]。

另一方面，在现实的软件开发中大量的时间被用于发现和消除软件中的错误，也就是软件测试。除了一些大公司在大型软件系统开发中使用了一些自动测试手段[2]，在很多情况下，软件测试仍然停留在手工测试阶段。手工测试不仅效率很低，而且容易出错。测试任务往往很繁重，在资源有限、时间紧迫的情况下测试任务常常不能充分完成。测试和调试手段的匾乏己经成为制约软件生产率和软件质量的一个瓶颈。

程序检测按是否需要执行程序可以分为：静态分析技术和动态分析技术两类。动态分析就是执行程序，再观察其行为是否满足要求。既可以由用户直接观察，也可以使用一定的辅助工具。例如，PurifyPlus[3]等通过在程序中加入代码来动态地监视程序的运行状态。

# 三、总结

# 四、参考文献

[1] Brooks Jr F P．The Mythical Man-Month，Anniversary Edition：Essays on Software Engineering[M]．Pearson Education，1995．

[2] Fewster M，Graham D．Software test automation：effective use of test execution tools[M]．ACM Press/Addison-Wesley Publishing Co，1999．

[3] Begic G． An introduction to runtime analysis with Rational PurifyPlus[J]．Nov， 2003，19：11．