# 基于“生成-检验”框架的软件代码错误自动修复技术研究

（申请清华大学工学博士学位论文）

培 养 单 位：软 件 学 院
学　　　科：软 件 工 程
研　究　生：郭　心　睿
指 导 教 师：孙　家　广　教　授
副指导教师：顾　明　教　授
联合导师：宋　晓　宇　教　授

二〇一七年六月

# Automated Debugging Based on “Generate-and-Validate" Systems

Dissertation Submitted to

**Tsinghua University**

in partial fulfillment of the requirement

for the degree of

**Doctor of Philosophy**

in

**Software Engineering**

by

**Guo Xinrui**

Dissertation Supervisor : Professor Sun Jiaguang

Associate Supervisor : Professor Gu Ming

**February, 2017**

# 关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；（3）根据《中华人民共和国学位条例暂行实施办法》，向国家图书馆报送可以公开的学位论文。

本人保证遵守上述规定。

（保密的论文在解密后应遵守此规定）

作者签名：______________ 导师签名：______________

日　　期：______________ 日　　期：______________

# 摘　要


论文的摘要是对论文研究内容和成果的高度概括。摘要应对论文所研究的问题及其研究目的进行描述，对研究方法和过程进行简单介绍，对研究成果和所得结论进行概括。摘要应具有独立性和自明性，其内容应包含与论文全文同等量的主要信息。使读者即使不阅读全文，通过摘要就能了解论文的总体内容和主要成果。

论文摘要的书写应力求精确、简明。切忌写成对论文书写内容进行提要的形式，尤其要避免“第 1 章……；第 2 章……；……”这种或类似的陈述方式。

本文介绍清华大学论文模板 ThuThesis 的使用方法。本模板符合学校的本科、硕士、博士论文格式要求。

本文的创新点主要有：

- 用例子来解释模板的使用方法；
- 用废话来填充无关紧要的部分；
- 一边学习摸索一边编写新代码。

关键词是为了文献标引工作、用以表示全文主要内容信息的单词或术语。关键词不超过 5 个，每个关键词中间用分号分隔。（模板作者注：关键词分隔符不用考虑，模板会自动处理。英文关键词同理。）


**关键词：** TEX；LATEX；CJK；模板；论文

# Abstract

An abstract of a dissertation is a summary and extraction of research work and contributions. Included in an abstract should be description of research topic and research objective, brief introduction to methodology and research process, and summarization of conclusion and contributions of the research. An abstract should be characterized by independence and clarity and carry identical information with the dissertation. It should be such that the general idea and major contributions of the dissertation are conveyed without reading the dissertation.

An abstract should be concise and to the point. It is a misunderstanding to make an abstract an outline of the dissertation and words “the first chapter”, “the second chapter” and the like should be avoided in the abstract.

Key words are terms used in a dissertation for indexing, reflecting core information of the dissertation. An abstract may contain a maximum of 5 key words, with semi-colons used in between to separate one another.

**Key words:** TEX; LATEX; CJK; template; thesis

# 目 录

# 主要符号对照表

| | |
|---|---|
| HPC | 高性能计算 (High Performance Computing) |
| cluster | 集群 |
| Itanium | 安腾 |
| SMP | 对称多处理 |
| API | 应用程序编程接口 |
| PI | 聚酰亚胺 |
| MPI | 聚酰亚胺模型化合物，N-苯基邻苯酰亚胺 |
| PBI | 聚苯并咪唑 |
| MPBI | 聚苯并咪唑模型化合物，N-苯基苯并咪唑 |
| PY | 聚吡咙 |
| PMDA-BDA | 均苯四酸二酐与联苯四胺合成的聚吡咙薄膜 |
| $\Delta G$ | 活化自由能 (Activation Free Energy) |
| $\chi$ | 传输系数 (Transmission Coefficient) |
| $E$ | 能量 |
| $m$ | 质量 |
| $c$ | 光速 |
| $P$ | 概率 |
| $T$ | 时间 |
| $v$ | 速度 |
| 劝学 | 君子曰：学不可以已。青，取之于蓝，而青于蓝；冰，水为之，而寒于水。木直中绳。輮以为轮，其曲中规。虽有槁暴，不复挺者，輮使之然也。故木受绳则直，金就砺则利，君子博学而日参省乎己，则知明而行无过矣。吾尝终日而思矣，不如须臾之所学也；吾尝跂而望矣，不如登高之博见也。登高而招，臂非加长也，而见者远；顺风而呼，声非加疾也，而闻者彰。假舆马者，非利足也，而致千里；假舟楫者，非能水也，而绝江河，君子生非异也，善假于物也。积土成山，风雨兴焉；积水成渊，蛟龙生焉；积善成德，而神明自得，圣心备焉。故不积跬步，无以至千里；不积小流，无以成江海。骐骥一跃，不能十步；驽马十驾，功在不舍。锲而舍之，朽木不折；锲而不舍，金石可镂。蚓无爪牙之利，筋骨之强，上食埃土，下饮黄泉，用心一也。蟹 |

六跪而二螯，非蛇蟮之穴无可寄托者，用心躁也。——荀况

# 第 1 章　框架扩展

## 1.1　引言

基于“生成-检验”框架的程序自动修复系统以源代码和其对应的测试用例为输入，输出一组能够使测试集中所有测试用例通过的程序变体供开发人员参考。从设计的目标来看，“生成-检验”系统希望能够接手开发人员的调试工作，提高软件开发效率。然而从实验数据来看，现有系统在修复规模稍大的程序时速度仍然比较慢。例如，SPR 在实验对象程序 php 上常常需要几个小时才能完成修复，我们实现的系统 PFDebug 在修复 Closure Compiler 上的错误时也要花费几个小时。另一方面，由于系统中搜索引擎能够在有限时间内搜索完的搜索空间有限，现有系统一般只能在单一位置生成如表达式替换、方法替换等比较简单的修改方案，导致现有系统的修复正确率也不太高，例如 SPR 在 GenProg 的测试集上修复了 39/69 个错误，PFDebug 修复了 24/357 个错误。由于速度慢、正确率低，基于“生成-检验”框架的自动修复系统与实际应用仍有一定的距离。

从使用者的角度，遵循“生成-检验”框架开发的系统在计算过程中间不需要开发人员的参与，也不需要了解程序错误相关的更多信息。这一特点使得系统使用非常方便——只要启动程序，等待结果就可以了。但是，考虑到现有自动调试技术的发展水平，这种模式未必是“生成-检验”系统的最佳利用方式。事实上，如果能够让开发人员与系统有一定的互动，充分利用开发人员的调试经验、对调试任务的理解、错误类型的初步判断等信息，系统的修复速度可能得到进一步的提高，搜索空间也可以更大，正确率也随之提高。

基于上述想法，本章提出扩展“生成-检验”框架，使系统能够与开发人员充分共享信息，优化修复效果。扩展方式有两种，第一种是“交互式调试”。基本思想是，利用开发人员对程序的理解，向开发人员提供接口描述他对程序运行状态的判断，使系统能够将错误限定于较窄的范围内，提高错误定位的准确度。此外，系统允许开发人员将调试任务分为几个小任务逐个解决，这使得系统可以处理需要在多个位置修改才能完成的调试任务。第二种是针对单类别错误修复的可扩展框架，即规范“生成-检验”框架的基本结构，使得错误定位方法易于替换，搜索空间易于剪裁，方便在此结构上构造针对特定错误类型的修复系统。

为验证框架扩展的有效性，本章首先在已有框架上实现了“交互式调试”使用模式，形成了新的系统 SmartDebug，并以多个实际程序为调试对象比较使用 SmartDebug 和人工调试的调试效率。实验表明，使用 SmartDebug 确实加速了调试

任务完成过程。此外，本章整理了已有系统的框架结构，设计了方便扩展的开放接口，并在此框架上实现了针对空指针异常（NullPointerException）的修复系统。我们在 CWE_NullPointerDereference 测试集上进行试验，成功修复了其中？？/？？个程序错误。

本章的主要贡献如下：

- 提出“交互调试”扩展方式并实现，形成新的系统 SmartDebug。在一组真实程序上进行实验，结果表明 SmartDebug 能够加速开发人员完成调试任务的过程
- 提出将现有系统扩展为针对特定类别错误修复系统的扩展方式，并给出开放接口定义，将现有系统重构为可扩展框架 xDebug
- 在 xDebug 框架内实现针对 Java 空指针异常的修复系统 NPEDebug，并在 CWE_NullPointerDereference 测试集上完成实验，成功修复其中??/?? 个错误，证实 xDebug 的可用性

## 1.2　交互式调试

### 1.2.1　概述

介绍“交互式调试”的基本思想,阐述将其引入后“生成-检验”框架应做出的相应调整及其对系统错误定位模块和搜索模块的正面作用。The interactive debugging usage model aims to take advantage of programmers’ understanding of the program under test to break down complicated bugs into simple fractions and narrow down possible bug locations to improve the debug efficiency.

From the programmers’ perspective, SmartDebug works as a personal consultant. During the common debugging process using JDT debug frontend, the programmer is able to describe his judgment of the program running state at a certain point of execution to SmartDebug through *Checkpoints*. If the program executes correctly, the programmer can mark down this execution point as “correct". Otherwise, he or she can input a boolean expression that should evaluate to `true` if the program runs correctly as a description of his or her expectations.

SmartDebug tracks the current debugging process. Figure 2 shows a snapshot of the debug process control panel. The panel lists every checkpoint and their current satisfaction status. For each failing test case, SmartDebug will find the first failing checkpoint it encounters during execution. The programmer may choose any failing checkpoints as the debug target in the next step.

When searching for fixes, the fault localizer utilizes the information of checkpoint to achieve more accurate fault localization. We adopt the Ochiai[56] fault localization metric, however the execution trace of the test case containing the target checkpoint is divided into two segments by the last correct checkpoint. The first segment is counted as a successful execution trace, while the second failing. Only code lines covered by the second segment will be treated as possible fix positions. Therefore SmartDebug is able to focus on a possibly small fraction of programs.

In the validation process, we do not rigorously require the candidate fixes to pass all test cases, instead we report every fix that can fix the target checkpoint while we rank the fixes according to the number of test cases the program passes if they are applied.

### 1.2.2 相关工作

### 1.2.3 应用示例

### 1.2.4 系统结构

Figure 1 shows the system architecture of SmartDebug. SmartDebug consists two major components, the *Interactive Frontend* and the *Fix Generation Backend*. The Interactive Frontend provides facilities for programmers to describe their judgments and expectations of the program running state through “Checkpoints" (*Checkpoint Manager*). It also tracks the debugging process of the program, i.e. the satisfaction status of each Checkpoint on each test case, guiding the programmer through the whole debugging task (*Debug Process Controller*). The *Fix Generation Backend* first localizes the bug by collecting and analyzing the execution trace of each test case (the *Fault Localizer)*, then generates program mutations within a predefined *Search Space* according to the generated ranking list of suspicious fix sites (*Search Engine*), after which some of the mutations go through a *Filter* that removes mutations impossible to fix the bug. Finally the survived mutations are validated by applying them back to the program and rerun the test suite (*Final Validator*).

Search space specification is vital in fix correctness rate in generate-and-validate systems. SmartDebug adopts the mutation patterns listed in Table I, whose first column lists the name of patterns and the second column provides brief explanations.

表 1.1　SmartDebug v.s. Human

| No. | Bug Summary | SD(s) | H(s) |
|---|---|---|---|
| 1 | wrong usage of loop variables | 282 | 300 |
| 2 | wrong operator | 95 | 691 |
| 3 | wrong usage of a local variable | 1055 | 694 |
| 4 | wrong usage of loop variables | 260 | 423 |
| 5 | wrong usage of loop variables | 309 | 410 |
| 6 | wrong usage of loop variables | 198 | 341 |
| 7 | wrong usage of a numeric variable | 215 | 829 |
| 8 | wrong usage of a local variable | 228 | 600 |

### 1.2.5　扩展的错误定位

### 1.2.6　调试进度控制

### 1.2.7　实验结果

We collected 25 versions of buggy programs from a coding exam for first year graduate students and asked students in the same year to debug these programs with or without help of SmartDebug. We recorded the time cost on both situations and compared them in Table III.

### 1.2.8　总结

## 1.3　针对单类别错误的可扩展框架

### 1.3.1　概述

举例说明“生成-检验”系统在修复特定类别错误上的局限性，阐述将针对特定类别错误修复算法整合进“生成-检验”系统中的架构设计，以空指针（NPE）为例具体说明该架构的可扩展性。

### 1.3.2　相关工作

### 1.3.3　框架设计

### 1.3.4　扩展示例

单类别错误修复实例：在 CWE-Null-Dereference 测试集上的评测结果。

## 1.4　本章小结

# 第 2 章　SmartDebug 工具设计与实现

## 2.1　引言

介绍 SmartDebug 工具的集成环境及应用对象，其所含基本功能模块。

## 2.2　功能模块

检查点管理模块；修复策略配置模块；修复建议提示与应用模块。

## 2.3　应用实例

在 defects4J 中较大程序上的应用；在实际编程过程中收集的真实错误上的应用。

## 2.4　本章小结

# 第 3 章　总结与展望

## 3.1　工作总结

总结本文针对现有“生成-检验”自动错误修复框架提出的多项优化技术，分析优点和局限性。

## 3.2　研究展望

分析能够进行的进一步研究，如“预过滤”技术的扩展应用、单类别错误的支持类型的扩展。

# 插图索引

# 表格索引

# 公式索引

# 参考文献

[1] Peters D, Parnas D L. Generating a test oracle from program documentation: Work in progress. Proceedings of the 1994 ACM SIGSOFT International Symposium on Software Testing and Analysis, New York, NY, USA: ACM, 1994. 58–65.

[2] Harman M, McMinn P, Shahbaz M, et al. A comprehensive survey of trends in oracles for software testing. Technical report, Technical Report Research Memoranda CS-13-01, Department of Computer Science, University of Sheffield, 2013.

[3] Manolache L, Kourie D G. Software testing using model programs. Software: Practice and Experience, 2001, 31(13):1211–1236.

[4] McMinn P, Stevenson M, Harman M. Reducing qualitative human oracle costs associated with automatically generated test data. Proceedings of the First International Workshop on Software Test Output Validation. ACM, 2010. 1–4.

[5] Wong W E, Horgan J R, London S, et al. Effect of test set minimization on fault detection effectiveness. Software Engineering, 1995. ICSE 1995. 17th International Conference on. IEEE, 1995. 41–41.

[6] Wong W E, Horgan J R, Mathur A P, et al. Test set size minimization and fault detection effectiveness: A case study in a space application. Journal of Systems and Software, 1999, 48(2):79–89.

[7] Do H, Elbaum S G, Rothermel G. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. Empirical Software Engineering: An International Journal, 2005, 10(4):405–435.

[8] Jones J A, Harrold M J. Empirical evaluation of the tarantula automatic fault-localization technique. Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, New York, NY, USA: ACM, 2005. 273–282.

[9] Abreu R, Zoeteweij P, Gemund A. An evaluation of similarity coefficients for software fault localization. Dependable Computing, 2006. PRDC ’06. 12th Pacific Rim International Symposium on, 2006. 39–46.

[10] Abreu R, Zoeteweij P, Golsteijn R, et al. A practical evaluation of spectrum-based fault localization. Journal of Systems and Software, 2009, 82(11):1780 – 1792. SI: {TAIC} {PART} 2007 and {MUTATION} 2007.

[11] Naish L, Lee H J, Ramamohanarao K. A model for spectra-based software diagnosis. ACM Trans. Softw. Eng. Methodol., 2011, 20(3):11:1–11:32.

[12] XIE X, CHEN T Y, KUO F C, et al. A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization. ACM Transactions on Software Engineering and Methodology (TOSEM), 2013..

[13] Le T D, Thung F, Lo D. Theory and practice, do they match? a case with spectrum-based fault localization. Software Maintenance (ICSM), 2013 29th IEEE International Conference on, 2013. 380–383.

[14] Yu Y, Jones J A, Harrold M J. An empirical study of the effects of test-suite reduction on fault localization. Proceedings of the 30th International Conference on Software Engineering, New York, NY, USA: ACM, 2008. 201–210.

[15] Masri W, Abou-Assi R, El-Ghali M, et al. An empirical study of the factors that reduce the effectiveness of coverage-based fault localization. Proceedings of the 2Nd International Workshop on Defects in Large Software Systems: Held in Conjunction with the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2009), New York, NY, USA: ACM, 2009. 1–5.

[16] Masri W, Assi R. Cleansing test suites from coincidental correctness to enhance fault-localization. Software Testing, Verification and Validation (ICST), 2010 Third International Conference on, 2010. 165–174.

[17] Masri W, Assi R A. Prevalence of coincidental correctness and mitigation of its impact on fault localization. ACM Trans. Softw. Eng. Methodol., 2014, 23(1):8:1–8:28.

[18] Wang X, Cheung S C, Chan W K, et al. Taming coincidental correctness: Coverage refinement with context patterns to improve fault localization. Proceedings of the 31st International Conference on Software Engineering, Washington, DC, USA: IEEE Computer Society, 2009. 45–55.

[19] Kochhar P S, Le T D B, Lo D. It’s not a bug, it’s a feature: Does misclassification affect bug localization? Proceedings of the 11th Working Conference on Mining Software Repositories, New York, NY, USA: ACM, 2014. 296–299.

[20] Herzig K, Just S, Zeller A. It's not a bug, it's a feature: How misclassification impacts bug prediction. Proceedings of the 2013 International Conference on Software Engineering, Piscataway, NJ, USA: IEEE Press, 2013. 392–401.

[21] Memon A M, Pollack M E, Soffa M L. Automated test oracles for guis. SIGSOFT Softw. Eng. Notes, 2000, 25(6):30–39.

[22] Aggarwal K K, Singh Y, Kaur A, et al. A neural net based approach to test oracle. SIGSOFT Softw. Eng. Notes, 2004, 29(3):1–6.

[23] Peters D, Parnas D. Using test oracles generated from program documentation. Software Engineering, IEEE Transactions on, 1998, 24(3):161–173.

[24] Davis M D, Weyuker E J. Pseudo-oracles for non-testable programs. Proceedings of the ACM ’81 Conference, New York, NY, USA: ACM, 1981. 254–257.

[25] Brown D, Roggio R, Cross I, et al. An automated oracle for software testing. Reliability, IEEE Transactions on, 1992, 41(2):272–280.

[26] Chen T, Tse T H, Zhou Z. Fault-based testing in the absence of an oracle. Computer Software and Applications Conference, 2001. COMPSAC 2001. 25th Annual International, 2001. 172–178.

[27] Chen T, Tse T, Zhou Z Q. Fault-based testing without the need of oracles. Information and Software Technology, 2003, 45(1):1 – 9.

[28] Murphy C, Kaiser G E. Automatic detection of defects in applications without test oracles. 2010..

[29] Murphy C, Shen K, Kaiser G. Automatic system testing of programs without test oracles. Proceedings of the Eighteenth International Symposium on Software Testing and Analysis, New York, NY, USA: ACM, 2009. 189–200.

[30] Cheon Y, Kim M Y, Perumandla A. A complete automation of unit testing for java programs. 2005..

[31] Jones J A, Harrold M J, Stasko J. Visualization of test information to assist fault localization. Proceedings of the 24th international conference on Software engineering. ACM, 2002. 467–477.

[32] Steimann F, Frenkel M, Abreu R. Threats to the validity and value of empirical assessments of the accuracy of coverage-based fault locators. Proceedings of the 2013 International Symposium on Software Testing and Analysis, New York, NY, USA: ACM, 2013. 314–324.

[33] Wikipedia. Tf-idf — wikipedia, the free encyclopedia, 2014. http://en.wikipedia.org/w/index.php?title=Tf%E2%80%93idf&oldid=589593815. [Online; accessed 21-January-2014].

[34] Pan K, Kim S, Whitehead Jr E J. Toward an understanding of bug fix patterns. Empirical Software Engineering, 2009, 14(3):286–315.

[35] Martinez M, Monperrus M. Mining software repair models for reasoning on the search space of automated program fixing. Empirical Software Engineering, 2015, 20(1):176–205.

[36] Fluri B, Wuersch M, PInzger M, et al. Change distilling:tree differencing for fine-grained source code change extraction. IEEE Transactions on Software Engineering, 2007, 33(11):725–743.

[37] Goues C L, Nguyen T, Forrest S, et al. Genprog: A generic method for automatic software repair. IEEE Transactions on Software Engineering, 2012, 38(1):54–72.

[38] Goues C L, Dewey-Vogt M, Forrest S, et al. A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8 each. Software Engineering (ICSE), 2012 34th International Conference on, 2012. 3–13.

[39] Qi Y, Mao X, Lei Y, et al. The strength of random search on automated program repair. Proceedings of the 36th International Conference on Software Engineering, New York, NY, USA: ACM, 2014. 254–265.

[40] Weimer W, Fry Z P, Forrest S. Leveraging program equivalence for adaptive program repair: Models and first results. Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on, 2013. 356–366.

[41] Qi Z, Long F, Achour S, et al. An analysis of patch plausibility and correctness for generate-and-validate patch generation systems (supplementary material). 2015..

[42] Kim D, Nam J, Song J, et al. Automatic patch generation learned from human-written patches. Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013. 802–811.

[43] Chandra S, Torlak E, Barman S, et al. Angelic debugging. Software Engineering (ICSE), 2011 33rd International Conference on. IEEE, 2011. 121–130.

[44] DeMarco F, Xuan J, Le Berre D, et al. Automatic repair of buggy if conditions and missing preconditions with smt. Proceedings of the 6th International Workshop on Constraints in Software Testing, Verification, and Analysis. ACM, 2014. 30–39.

[45] Nguyen H D T, Qi D, Roychoudhury A, et al. Semfix: program repair via semantic analysis. Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013. 772–781.

[46] Mechtaev S, Yi J, Roychoudhury A. Directfix: Looking for simple program repairs. Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on, volume 1. IEEE, 2015. 448–458.

[47] Long F, Rinard M. Staged program repair with condition synthesis. Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, New York, NY, USA: ACM, 2015. 166–178.

[48] Long F, Rinard M. Automatic patch generation by learning correct code. Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. ACM, 2016. 298–312.

[49] Long F, Rinard M. An analysis of the search spaces for generate and validate patch generation systems. Proceedings of the 38th International Conference on Software Engineering, New York, NY, USA: ACM, 2016. 702–713.

[50] Just R, Jalali D, Ernst M D. Defects4j: A database of existing faults to enable controlled testing studies for java programs. Proceedings of the 2014 International Symposium on Software Testing and Analysis, New York, NY, USA: ACM, 2014. 437–440.

[51] JDT E. Eclipse java development tools(jdt).

[52] Galenson J, Reames P, Bodik R, et al. Codehint: Dynamic and interactive synthesis of code snippets. Proceedings of the 36th International Conference on Software Engineering, New York, NY, USA: ACM, 2014. 653–663.

[53] T. j. watson libraries for analysis (wala), 2017. wala.sourceforge.net. [Online; accessed 12-March-2016].

[54] Durieux T, Martinez M, Monperrus M, et al. Automatic repair of real bugs: An experience report on the defects4j dataset. CoRR, 2015, abs/1505.07002.

[55] Guo X. Smartdebug: An interactive debug assistant for java. Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, New York, NY, USA: ACM, 2016. 1127–1129.

[56] Abreu R, Zoeteweij P, Gemund A J C v. An evaluation of similarity coefficients for software fault localization. Proceedings of the 12th Pacific Rim International Symposium on Dependable Computing, Washington, DC, USA: IEEE Computer Society, 2006. 39–46.

# 致 谢

衷心感谢导师孙家广教授和顾明老师、宋晓宇老师对本人的精心指导。他们的言传身教将使我终生受益。

感谢清华大学学生艺术团舞蹈队的同学们，与你们在一起的时光总是非常美妙，台上台下我们共享世间美好。

感谢软件学院羽毛球队的所有同学，特别感谢嘉祥、朱晗、崇哥对我的耐心指导，你们为我打开了新世界的大门，让我感受到竞技体育特殊的魅力。

感谢父母多年来的支持。

感谢好友们一路相伴，特别感谢熊曦、包子、丢丢，是你们在我最困难的时候陪伴我纾解痛苦，让我仍能坚持到最后。

感谢实验室的兄弟姐妹，多年互相支持鼓励甚为不易。相聚是缘，后会有期。

感谢 ThuThesis，感谢 LATEX，论文因你们而优雅。

# 声 明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签 名：__________ 日 期：__________

# 附录 A　公式 2-2 和 2-3 的证明

Let $\mathbf{P}(*)$ denote the probability of $*$. Let $O_c$ be the correct test oracle. There are two cases of false negatives and two cases of false positives. For expression simplicity, we encode them as follows:

1. $P_O F_{O_c} P_{O'}$:$O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F} \wedge O'(t_i) = \mathcal{P}$
2. $F_O P_{O_c} F_{O'}$:$O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{P} \wedge O'(t_i) = \mathcal{F}$
3. $P_O P_{O_c} F_{O'}$:$O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{P} \wedge O'(t_i) = \mathcal{F}$
4. $F_O F_{O_c} P_{O'}$:$O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{F} \wedge O'(t_i) = \mathcal{P}$

Let $r$ be the error rate of the test oracle, i.e. the ratio of faulty oracle judgements to all oracle judgements. Then the probability that $t_i$ is false negative is

$$\mathbf{P}(fn(t_i)) = \mathbf{P}(P_O F_{O_c} P_{O'}) + \mathbf{P}(F_O P_{O_c} F_{O'}) \tag{A-1}$$

And the probability that $t_i$ is false positive is

$$\mathbf{P}(fp(t_i)) = \mathbf{P}(P_O F_{O_c} P_{O'}) + \mathbf{P}(F_O P_{O_c} F_{O'}) \tag{A-2}$$

And now we calculate $\mathbf{P}(P_O F_{O_c} P_{O'})$, $\mathbf{P}(F_O P_{O_c} F_{O'})$, $\mathbf{P}(P_O F_{O_c} P_{O'})$ and $\mathbf{P}(F_O P_{O_c} F_{O'})$.

$$\begin{aligned}
&\mathbf{P}(P_O F_{O_c} P_{O'}) \\
=&\mathbf{P}(O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F} \wedge O'(t_i) = \mathcal{P}) \\
=&\mathbf{P}(O'(t_i) = \mathcal{P} | O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F}) \cdot \mathbf{P}(O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F})
\end{aligned} \tag{A-3}$$

$$\begin{aligned}
&\mathbf{P}(O'(t_i) = \mathcal{P} | O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F}) \\
=&\mathbf{P}(Suspicion(t_i) \leq thres) \\
=&\mathbf{P}(\frac{Vote_{if}}{Vote_{ip} + Vote_{if}} \leq thres) \\
=&\mathbf{P}(\frac{\sum_{t_{iq} \in T_{i,f}} Sim(t_i, t_{iq})}{\sum_{t_{iq} \in T_i} Sim(t_i, t_{iq})} \leq thres)
\end{aligned} \tag{A-4}$$

We assume that test cases in $T_i$ are similar enough to $t_i$ and the similarity are almost the same. Then approximately $\forall t_{iq} \in T_i$, $Sim(t_i, t_{iq}) \approx sim_i$, where $sim_i$ represents the average similarity of all $Sim(t_i, t_{iq})$. Then we have

$$\begin{aligned}
&\mathbf{P}(\frac{\sum_{t_{iq} \in T_{i,f}} Sim(t_i, t_{iq})}{\sum_{t_{iq} \in T_i} Sim(t_i, t_{iq})} \le thres) \\
\approx&\mathbf{P}(\frac{|T_{i,f}| \times sim_i}{|T_i| \times sim_i} \le thres) \\
=&\mathbf{P}(\frac{|T_{i,f}|}{|T_i|} \le thres) \\
=&\mathbf{P}(|T_{i,f}| \le |T_i| \times thres) \\
=&\sum_{w=0}^{\hat{n}} \mathbf{P}(|T_{i,f}| = w)
\end{aligned} \tag{A-5}$$

where $\hat{n} = \lfloor n \times thres \rfloor$.

Precise calculation of $\mathbf{P}(|T_{i,f}| = w)$ (given $w$) is obviously very complicated. Fortunately in our case an accurate estimation is good enough. To simplify the problem, we make the following two assumptions:

*Assump. 1* : $\forall t_{iq} \in T_i \cup \{t_i\}, O_c(t_{iq}) \in \{\mathcal{P}, \mathcal{F}\}$ *i.i.d.*.

*Assump. 2* : $\forall t_{iq_1}, t_{iq_2} \in T_i \cup \{t_i\}, iq_1 \ne iq_2, \mathbf{P}(O_c(t_{iq_1}) = O_c(t_{iq_2})) = sim_i\ (constant)$

The reasons for making such assumptions are explained in Section 3.3 thus not repeated here. Based on these two assumptions,

$$\begin{aligned}
&\mathbf{P}(|T_{i,f}| = w) \\
=&\mathrm{C}_n^w(\mathbf{P}(O(t_{iq}) = \mathcal{F}))^w(\mathbf{P}(O(t_{iq}) = \mathcal{P}))^{(n-w)} \\
=&\mathrm{C}_n^w(\mathbf{P}(O_c(t_{iq}) = \mathcal{F})\mathbf{P}(O(t_{iq}) = O_c(t_{iq})) + \mathbf{P}(O_c(t_{iq}) = \mathcal{P})\mathbf{P}(O(t_{iq}) \ne O_c(t_{iq})))^w \\
&\cdot(\mathbf{P}(O_c(t_{iq}) = \mathcal{P})\mathbf{P}(O(t_{iq}) = O_c(t_{iq})) + \mathbf{P}(O_c(t_{iq}) = \mathcal{F})\mathbf{P}(O(t_{iq}) \ne O_c(t_{iq})))^{(n-w)}
\end{aligned}$$

where $t_{iq}$ is an arbitrary test case in $T_i$.

Remember $r$ is the error rate of the test oracle, then $\mathbf{P}(O(t_{iq}) = O_c(t_{iq})) = 1 - r$, $\mathbf{P}(O(t_{iq}) \ne O_c(t_{iq})) = r$. Use $\beta_i$ to represent $\mathbf{P}(O_c(t_{iq}) = \mathcal{F}), t_{iq} \in T_i$, then $\mathbf{P}(O_c(t_{iq}) =$

$\mathcal{P}) = 1 - \beta_i$, and

$$\begin{aligned} &\mathbf{P}(|T_{i,f}| = w) \\ =&\mathrm{C}_n^w(\beta_i(1-r) + (1-\beta_i)r)^w((1-\beta_i)(1-r) + \beta_i r)^{(n-w)} \\ =&\mathrm{C}_n^w(\beta_i + r - 2\beta_i r)^w(1 - (\beta_i + r - 2\beta_i r))^{(n-w)} \end{aligned} \tag{A-6}$$

$r$ is completely dependent on the oracle error itself, so we cannot compute its value from elsewhere. However, $\beta_i$ can be deducted through $sim_i$. $\forall t_{iq_1}, t_{iq_2} \in T_i \cup \{t_i\}$, $iq_1 \neq iq_2$, according to *Assump. 1*,

$$\begin{aligned} &\mathbf{P}(O_c(t_{iq_1}) = O_c(t_{iq_2})) \\ =&\mathbf{P}(O_c(t_{iq_1}) = \mathcal{P})\mathbf{P}(O_c(t_{iq_2}) = \mathcal{P}) + \mathbf{P}(O_c(t_{iq_1}) = \mathcal{F})\mathbf{P}(O_c(t_{iq_2}) = \mathcal{F}) \\ =&(1-\beta_i)^2 + \beta_i^2 \end{aligned} \tag{A-7}$$

while according to *Assump. 2*,

$$\mathbf{P}(O_c(t_{iq_1}) = O_c(t_{iq_2})) = sim_i$$

Therefore,

$$(1-\beta_i)^2 + \beta_i^2 = sim_i$$

Solving this equation, we get $\beta_{i1} = \frac{1+\sqrt{2sim_i - 1}}{2} or \beta_{i2} = \frac{1-\sqrt{2sim_i - 1}}{2}$. Since we have assumed that all test cases in $T_i \cup \{t_i\}$ are similar enough, $sim_i$ should be close to 1, and $2sim_i - 1 > 0$. As we know that $O_c(t_i) = \mathcal{F}$, therefore

$$\mathbf{P}(O_c(t_{iq}) = \mathcal{F}) = \beta_{i1} = \frac{1+\sqrt{2sim_i - 1}}{2} \tag{A-8}$$

Synthesizing equation A-4, A-5 and A-6,

$$\begin{aligned} &\mathbf{P}(O'(t_i) = \mathcal{P} | O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F}) \\ =&\sum_{w=0}^{\hat{n}} \mathrm{C}_n^w(\beta_{i1} + r - 2\beta_{i1} r)^w(1 - (\beta_{i1} + r - 2\beta_{i1} r))^{(n-w)} \end{aligned} \tag{A-9}$$

Similarly, we have

$$\begin{aligned}
&\mathbf{P}(F_O P_{O_c} F_{O'}) \\
=&\mathbf{P}(O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{P} \wedge O'(t_i) = \mathcal{F}) \\
=&\mathbf{P}(O'(t_i) = \mathcal{F} | O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{P}) \cdot \mathbf{P}(O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{P})
\end{aligned} \tag{A-10}$$

and

$$\begin{aligned}
&\mathbf{P}(O'(t_i) = \mathcal{F} | O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{P}) \\
=&\sum_{w=0}^{\hat{n}} \mathbf{P}(|T_{i,p}| = w) \\
=&\sum_{w=0}^{\hat{n}} \mathrm{C}_n^w \cdot (\mathbf{P}(O(t_{iq}) = \mathcal{P}))^w \cdot (\mathbf{P}(O(t_{iq}) = \mathcal{F}))^{(n-w)} \\
=&\sum_{w=0}^{\hat{n}} \mathrm{C}_n^w \cdot ((1-\beta_i)(1-r) + \beta_i r)^w \cdot (\beta_i(1-r) + (1-\beta_i)r)^{(n-w)}
\end{aligned} \tag{A-11}$$

where $\beta$ represents $\mathbf{P}(O_c(t_{iq}) = \mathcal{F})$. In this case, $O_c(t_i) = \mathcal{P}$, therefore

$$\mathbf{P}(O_c(t_{iq}) = \mathcal{F}) = \beta_{i2} = \frac{1 - \sqrt{2sim_i - 1}}{2}$$

Since $\beta_{i1} + \beta_{i2} = 1$, we have

$$\begin{aligned}
&\mathbf{P}(O'(t_i) = \mathcal{F} | O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{P}) \\
=&\sum_{w=0}^{\hat{n}} \mathrm{C}_n^w ((1-\beta_{i2})(1-r) + \beta_{i2} r)^w (\beta_{i2}(1-r) + (1-\beta_{i2})r)^{(n-w)} \\
=&\sum_{w=0}^{\hat{n}} \mathrm{C}_n^w (\beta_{i1}(1-r) + (1-\beta_{i1})r)^w ((1-\beta_{i1})(1-r) + \beta_{i1} r)^{(n-w)} \\
=&\mathbf{P}(O'(t_i) = \mathcal{P} | O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F})
\end{aligned} \tag{A-12}$$

Synthesizing equations A-1, A-10, A-3, A-9 and A-12, we have

$$
\begin{aligned}
&\mathbf{P}(fn(t_i)) = \mathbf{P}(P_O F_{O_c} P_{O'}) + \mathbf{P}(F_O P_{O_c} F_{O'}) \\
=&\mathbf{P}(O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F} \wedge O'(t_i) = \mathcal{P}) + \mathbf{P}(O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{P} \wedge O'(t_i) = \mathcal{F}) \\
=&\mathbf{P}(O'(t_i) = \mathcal{P} | O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F}) \cdot \mathbf{P}(O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F}) \\
&+ \mathbf{P}(O'(t_i) = \mathcal{F} | O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{P}) \cdot \mathbf{P}(O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{P}) \\
=&\mathbf{P}(O'(t_i) = \mathcal{P} | O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F}) \\
&\cdot \mathbf{P}(O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F}) + \mathbf{P}(O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{P}) \\
=&\mathbf{P}(O'(t_i) = \mathcal{P} | O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F}) \cdot \mathbf{P}(O(t_i) \neq O_c(t_i)) \\
=&\mathbf{P}(O'(t_i) = \mathcal{P} | O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F}) \cdot r \\
\approx& r \sum_{w=0}^{\hat{n}} \mathrm{C}_n^w \phi_i^w (1 - \phi_i)^{(n-w)}
\end{aligned}
\tag{A-13}
$$

where $\phi_i = \beta_{i1} + r - 2\beta_{i1} r$.

Following the same routine,

$$
\begin{aligned}
&\mathbf{P}(P_O P_{O_c} F_{O'}) \\
=&\mathbf{P}(O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{P} \wedge O'(t_i) = \mathcal{F}) \\
=&\mathbf{P}(O'(t_i) = \mathcal{F} | O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{P}) \cdot \mathbf{P}(O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{P})
\end{aligned}
\tag{A-14}
$$

and

$$
\begin{aligned}
&\mathbf{P}(O'(t_i) = \mathcal{F} | O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{P}) \\
\approx& \sum_{w=\hat{n}+1}^{n} \mathbf{P}(|T_{i,f}| = w) \\
=& \sum_{w=\hat{n}+1}^{n} \mathrm{C}_n^w (1 - (\beta_{i1} + r - 2\beta_{i1} r))^w (\beta_{i1} + r - 2\beta_{i1} r)^{(n-w)} \\
=&\mathbf{P}(O'(t_i) = \mathcal{P} | O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{F})
\end{aligned}
\tag{A-15}
$$

then

$$\begin{aligned}
&\mathbf{P}(fp(t_i)) = \mathbf{P}(P_O P_{O_c} F_{O'}) + \mathbf{P}(F_O F_{O_c} P_{O'}) \\
=&(\mathbf{P}(O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{P} \wedge O'(t_i) = \mathcal{F})) + (\mathbf{P}(O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{F} \wedge O'(t_i) = \mathcal{P})) \\
=&\mathbf{P}(O'(t_i) = \mathcal{F} | O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{P}) \cdot \mathbf{P}(O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{P}) \\
&+ \mathbf{P}(O'(t_i) = \mathcal{P} | O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{F}) \cdot \mathbf{P}(O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{F}) \\
=&\mathbf{P}(O'(t_i) = \mathcal{F} | O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{P}) \\
&\cdot \mathbf{P}(O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{P}) + \mathbf{P}(O(t_i) = \mathcal{F} \wedge O_c(t_i) = \mathcal{F}) \\
=&\mathbf{P}(O'(t_i) = \mathcal{P} | O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F}) \cdot \mathbf{P}(O(t_i) = O_c(t_i)) \\
=&\mathbf{P}(O'(t_i) = \mathcal{P} | O(t_i) = \mathcal{P} \wedge O_c(t_i) = \mathcal{F}) \cdot (1 - r) \\
\approx&(1 - r) \sum_{w=\hat{n}+1}^{n} \mathrm{C}_n^w (1 - \phi_i)^w \phi_i^{(n-w)}
\end{aligned} \tag{A-16}$$

where $\phi_i = \beta_{i1} + r - 2\beta_{i1} r$.

Finally, we get

$$\mathbf{P}(fn(t_i)) \approx r \sum_{w=0}^{\hat{n}} \mathrm{C}_n^w \phi_i^w (1 - \phi_i)^{(n-w)} \tag{A-17}$$

$$\mathbf{P}(fp(t_i)) \approx (1 - r) \sum_{w=\hat{n}+1}^{n} \mathrm{C}_n^w (1 - \phi_i)^w \phi_i^{(n-w)} \tag{A-18}$$

where $\phi_i = \beta_{i1} + r - 2\beta_{i1} r$, $\beta_{i1} = \frac{1+\sqrt{2sim_i - 1}}{2}$, $sim_i$ is the average similarity of all test cases in the same $T_i$, $\hat{n} = \lfloor n \times thres \rfloor$, $r$ is the error rate of the test oracle.

# 个人简历、在学期间发表的学术论文与研究成果

## 个人简历

1990 年 4 月 19 日出生于吉林省长春市。

2008 年 8 月考入清华大学软件学院计算机软件专业，2012 年 7 月本科毕业并获得工学学士学位。

2012 年 9 月免试进入清华大学软件学院攻读工学博士学位至今。

## 发表的学术论文