
软件分析

# 缺陷修复技术

熊英飞
北京大学
2016

# 缺陷修复技术


- 定位缺陷之后，能否自动生成补丁？

- 输入：
  - 一个程序
  - 一组测试，至少有一个挂了
- 输出
  - 一个补丁
  - 应用补丁后，程序能跑通所有测试


# GenProg

# GenProg


- 2009年由弗吉尼亚大学Westley Weimers和Claire Le Goues提出
- 标志缺陷修复技术兴起的代表性工作
- 全自动修复程序中的缺陷，通过所有测试

# GenProg工作流程

- 基本流程
  - 定位缺陷位置
  - 循环：是否通过所有测试
    - 变异程序
    - 运行测试
  - 最小化补丁
- 定位缺陷位置：基于频谱的错误定位
- 最小化补丁：利用Delta Debugging最小化补丁

# GenProg工作流程

- 变异操作
  - 复制别的语句替换当前语句
  - 在当前语句之后插入别的语句
  - 删除当前位置语句
- 遗传操作
  - 选择两个适应度高的程序
  - 交换其中的变异操作
- 适应度计算
  - 通过测试越多，程序适应度越高

# 实验设置


- 2012年GenProg大型实验
- 实验对象
  - 选择行数>50000行，测试>10个，修改历史>300的程序
  - 用最新版本的测试用例检查缺陷，如果旧版本不能通过新版本的某个测试用例，则最新的一个不能通过的旧版本作为有缺陷的程序
- 每个程序最多有45个缺陷，总共105个缺陷
- 该测试集日后发展为ManyBugs标准测试集

# 实验效果


- 实验结论
  - 105个缺陷修复了55个
  - 总共花费403美元，平均7.32一个，每个耗时半天左右
  - 作者手动检查了2个缺陷，发现GenProg的修复都和人工修复是等价的

# GenProg的改进-AE


- 2013年由Westley Weimer等人提出
- 在修改的时候避免产生等价变换
- 设置了一系列规则快速检查特定类型的等价变换
- 在GenProg的测试集上做了验证，开销约为原来的三分之一

# GenProg工作影响


- 第一篇ICSE09论文被评为Distinguished Paper
- 7年总引用过1000次
- 论文主要博士生（三作）被CMU聘为Assistant Professor


# GenProg质疑

# 2011年Lionel Briand教学论文


- Andrea Arcuri, Lionel C. Briand: A practical guide for using statistical tests to assess randomized algorithms in software engineering. ICSE 2011: 1-10
- 指出现有很多论文统计方法应用不当
- 特别是很多方法连随机方法都不对比
  - 特别点名GenProg论文

# PAR

- 香港科技大学Sung Kim等人提出
- 替换GenProg中的变异模板为人工模板，如：
  - 在问题语句前面插入null检查
  - 更改方法调用中的参数变量
  - 重新调用一个签名相同但名字不同的方法
- 在Java上做了验证
  - 选择6个项目，从Issue-tracking系统中搜索缺陷，共119个，每个Bug产生10代或最多运行8小时
  - Par修复了27个，GenProg修复了16个
    - 作者分析GenProg表现效果差是因为GenProg的策略在Java上不适用
  - 学生和开发人员人工给补丁打分
    - PAR得分明显高于GenProg，在一些项目上甚至高于人工修复
- 法国Martin Monperrus后来撰文对PAR进行批评

# RSRepair


- 起源于国防科技大学毛晓光团队的系列研究
- ICSM 2012: 将程序分块编译好，这样之后只需要重新编译变化的部分，加快编译速度
- ICSM 2013: 将测试排序技术和修复验证相结合，以期望更快的发现修复不成功
  - 需要放弃遗传算法，因为无法计算适应度
- ICSE 2014: GenProg中的遗传算法不如随机搜索
  - 主要原因：计算适应度函数代价太大
- RSRepair：将GenProg中的遗传算法换成随机搜索，同时对测试排序，发现效果显著优于GenProg

# Kali

- 源于2015年麻省理工大学Martin Rinard的论文
- 验证了GenProg、AE、RSRepair
- 以GenProg为例说明结果
  - 414个补丁中只有110个通过测试，修复18个缺陷，而不是55个（总共105个）
  - 110个通过测试的补丁中经过人工检验只有5个正确，该5个补丁修复了2个缺陷
  - 大多数补丁都是简单的删除出错的功能
- 专门设计了只删除功能的Kali，发现效果和GenProg相当

# Martin Rinard


- 斯坦福Monica Lam弟子，ACM Fellow，大量Best Paper Award，多篇20年最有影响论文
- 最早提出自动修复软件缺陷的概念
  - 2003年就开始发表相关论文
  - 以前主要关注动态数据的修复
  - 2008年开始关注程序本身的修复，在ACM Communication发表Position论文一篇
  - 2009年在SOSP上发表14名作者、8家单位论文一篇，提出全自动修复二进制文件中缺陷的ClearView方法


# GenProg并行工作

# ClearView


- Martin Rinard团队在2009年的工作
- 缺陷定位：通过Monitor定位，Monitor报告出错的语句即为缺陷的语句
  - 大致等于程序崩溃时的语句
- 变异程序
  - 使用Daikon从程序中分析出不变式
    - a==1
  - 程序崩溃后，检测出最相关的不变式
    - 在当前执行中被违反并且在其他执行中通过次数尽量多
  - 修复生成
    - 利用模板从不变式生成
    - if (!(a==1)) a = 1;
    - if (!(a==1)) return;

# AutoFix-E


- 香港城市大学裴玉和ETH Zurich的Bertrand Meyer团队工作
- 和ClearView类似，但以方法为单位而不是以崩溃位置为单位
  - 学习每个方法被调用前的不变式
  - 学习每个方法对系统状态的改变情况
    - 如bind()方法会导致变量bound变成true
  - 在失败的运行中，如果发现有不变式在调用前被违反，则生成以下两种修复
    - 删除该调用
    - 调用相应的方法对系统状态进行改变
- 原则上应该比ClearView要强，但二者没有直接比较
  - 允许在多个地方修改，允许调用方法

# Nopol


- 武汉大学玄跻峰和法国Martin Monperrus团队2012年工作
- 第一篇专门修复if条件的论文
- 通过Predicate Switching定位缺陷
- 收集所有通过测试和失败测试的约束
- 用SMT求解约束

# SemFix


- 新加坡国立大学Ahibk Roychoudhury团队在2013年工作
- 同Nopol的思想类似，但是扩展到任意表达式
- 首先用基于频谱的方法定位到出错表达式
- 收集所有通过测试和失败测试的约束
- 用SMT求解约束


# 后GenProg时代工作

# 后GenProg时代


- Martin Rinard的论文暴露出现有修复技术的主要问题是不能以通过测试为目标
- 修复技术的目标调整为生成和原程序员补丁相同的补丁
- 基本手段：对能通过测试的补丁进行排序

# DirectFix和Angelix


- 新加坡国立大学Ahibk Roychoudhury团队在SemFix上的后续工作
- 生成语法上差别最小的修复
- 用语法树上被改动的元素个数定义差别
- 把原问题转成MaxSMT问题求解
- Angelix在DirectFix的基础上提升了性能，使得可用于大型程序
- ManyBugs数据集上的缺陷修复数量
  - 成功通过测试28个
  - 正确补丁10个

# Qlose


- 微软Rishabh Singh等人的工作
- 把语法上的差别最小改成了语义差别最小
- 语义差别最小定义为执行的控制流差别最小
- 但只在作业程序上做了验证，没有在大型程序上验证

# SPR和Prophet


- Martin Rinard团队龙凡在2015年的工作
- SPR：集成Nopol和部分PAR模板的修复工具
- Prophet：用机器学习方法对SPR可能生成的补丁进行排序，按顺序验证
- ManyBugs数据集上的缺陷修复数量
  - 成功通过测试42个
  - 正确补丁18个
- 目前C语言上最好的修复工具

# QACrashFix


- 北京大学高庆等人的工作
- 面向崩溃缺陷
- 从QA网站上寻找关于相同崩溃的讨论
  - 抛出的Crash Message相同
- 通过比较讨论代码分析出补丁
- 验证结果
  - 能修复的缺陷较少
  - 但修复的正确率很高（80%的补丁正确）

# ACS


- 北京大学王杰等人的工作
- 精确生成if条件，提高修复正确率
- 两阶段生成
  - 首先选择if语句中的变量
    - 利用变量使用的局部性原理
  - 其次选择在变量上采用的操作
    - 通过在代码库中统计P(oper|context)，其中oper是操作，context包含变量类型、变量名和方法名
- 在Java数据集Defects4J上的效果
  - 修复18个缺陷
  - 生成补丁78.3%是正确的
- 目前Java上最好的修复工具

# 缺陷修复展望

- 虽然困难，但仍然充满希望的新领域
- 学术界最活跃的研究领域之一
  - 2013年、2016年均有Dagstuhl召开
- 工业界大量关注和投入
  - 华为、360、富士通
- 最终通往自动编程的可行途径

- 欢迎同学们加入我们！