缺陷修复技术

熊英飞

北京大学软件工程研究所

报告人介绍 – 熊英飞

- 2000~2004，电子科技大学本科
- 2004~2006，北京大学研究生
  - 导师：梅宏、杨芙清
- 2006~2009，日本东京大学博士
  - 导师：胡振江、武市正人
- 2009~2011，加拿大滑铁卢大学博士后
  - 导师：Krzysztof Czarnecki
- 2012~，北京大学“百人计划”研究员（Tenure-Track）
- 研究方向：软件分析、编程语言设计

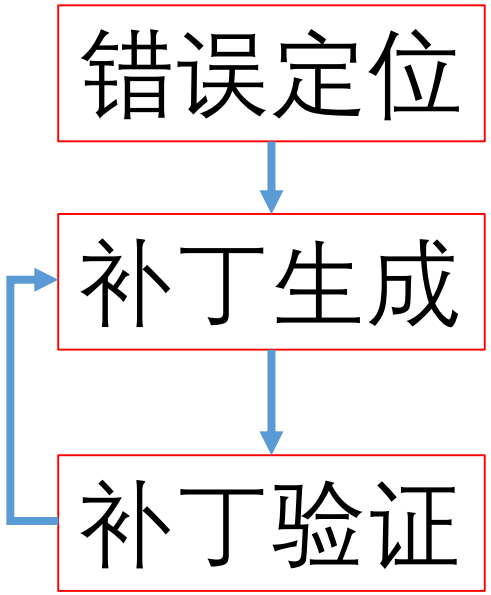
缘起

- 人和Bug的斗争从来没有停止过
- 缺陷检测：到底有没有Bug
  - 从上世纪60年代开始
  - 代表技术：软件测试、软件验证
- 缺陷定位：Bug在哪里
  - 从上世纪90年代开始
  - 代表技术：基于频谱的缺陷定位、统计性调试
- 缺陷修复：自动消除Bug
  - 约从2000年之后开始
  - 代表技术：生成-验证缺陷修复技术
- 未来愿景：交互式自动程序开发
  - Issue=Bug Report+Feature Request

缺陷自动修复

输入：一个程序和一个正确性条件，并且程序不满足该条件
输出：一个补丁，可以使程序满足正确性条件

典型框架：
“生成-验证”框架



正确性条件的主要类型

- 特定类型缺陷的正确性条件
- 形式化规约
- 测试

特定类型缺陷的修复

- 根据缺陷的特点，针对固定缺陷可以给出明确的修复条件
- 并行缺陷
  - 死锁、竞争等
  - Shan Lu（芝加哥大学）、蔡彦（中科院软件所）
- HTML兼容性缺陷修复
  - 通过调整CSS让两个浏览器显示完全相同
  - William G.J. Halfond（南加州大学）
- 内存泄露
  - 我们团队的工作
  - 稍后介绍

基于形式化规约的缺陷修复

- 大多数通用缺陷无法预定义正确性条件
- 假设程序员提供了形式化规约定义了正确性条件
- Qlose, Deductive Repair
  - 采用程序综合技术来生成满足规约的程序
  - Loris D'Antoni（威斯康星）、Viktor Kuncak（马普研究所）
- AutoFix
  - 基于形式化规约生成测试然后修复，并不能保证满足规约
  - 裴玉（香港理工大学）、Bertrand Meyer (ETH Zurich)

基于测试的缺陷修复

- 实践中的程序通常没有完整的形式化规约
- 以通过测试为修复成功的标准

代表性工作

- GenProg
  - Claire Le Goues（卡内基梅隆）、Westley Weimer（弗吉尼亚）
  - 错误定位：采用统计性调试
  - 补丁生成：
    - 基本操作：复制其他语句/删除语句
    - 采用遗传算法从基本操作合成补丁
  - 补丁验证：运行程序中的测试验证补丁
  - 实证研究：55/105，8$/bug
- 其他代表性工作
  - DirectFix/SemFix：Abhik Roychoudhury（新加坡国立）
  - Nopol：玄跻峰（武汉大学）
  - PAR：Sung Kim（香港科技大学）
  - RSRepair：毛晓光（国防科技大学）

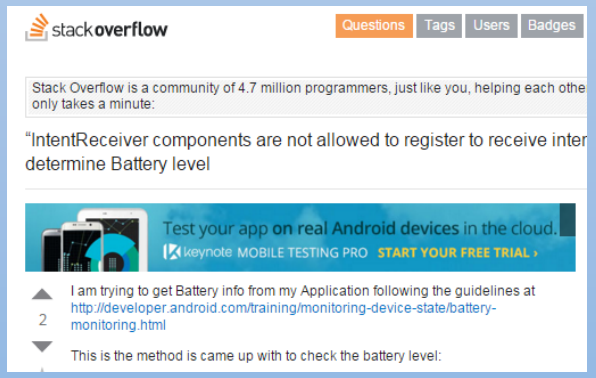
转折

- [Qi-ISSTA'15]
  - GenProg被认为修复的55个缺陷中，只有2个是正确的
  - 根本原因：通过测试并不代表是正确的修复
- [Le Goues-FSE'15]
  - 详细实验了GenProg，AE等多个主流修复方法，采用了更大的数据集，更多的测试集
  - 结果基本一致
- 其他后续工作
  - Prophet：龙凡（多伦多大学）、Martin Rinard（麻省理工大学）
  - Angelix：Abhik Roychoudhury（新加坡国立）
  - 补丁的正确率最好也不到40%

我们的工作：高正确率的缺陷修复

内存泄露自动修复

从QA网站学习



精准条件修复

[1] Qing Gao, Yingfei Xiong, Yaqing Mi, Lu Zhang, Weikun Yang, Zhaoping Zhou, Bing Xie, Hong Mei. Safe Memory-Leak Fixing for C Programs. ICSE'15
[2] Qing Gao, Hansheng Zhang, Jie Wang, Yingfei Xiong, Lu Zhang, Hong Mei. Fixing Recurring Crash Bugs via Analyzing Q&A Sites. ASE'15
[3] Yingfei Xiong, Jie Wang, Runfa Yan, Jiachen Zhang, Shi Han, Gang Huang, Lu Zhang. Precise Condition Synthesis for Program Repair. ICSE'17

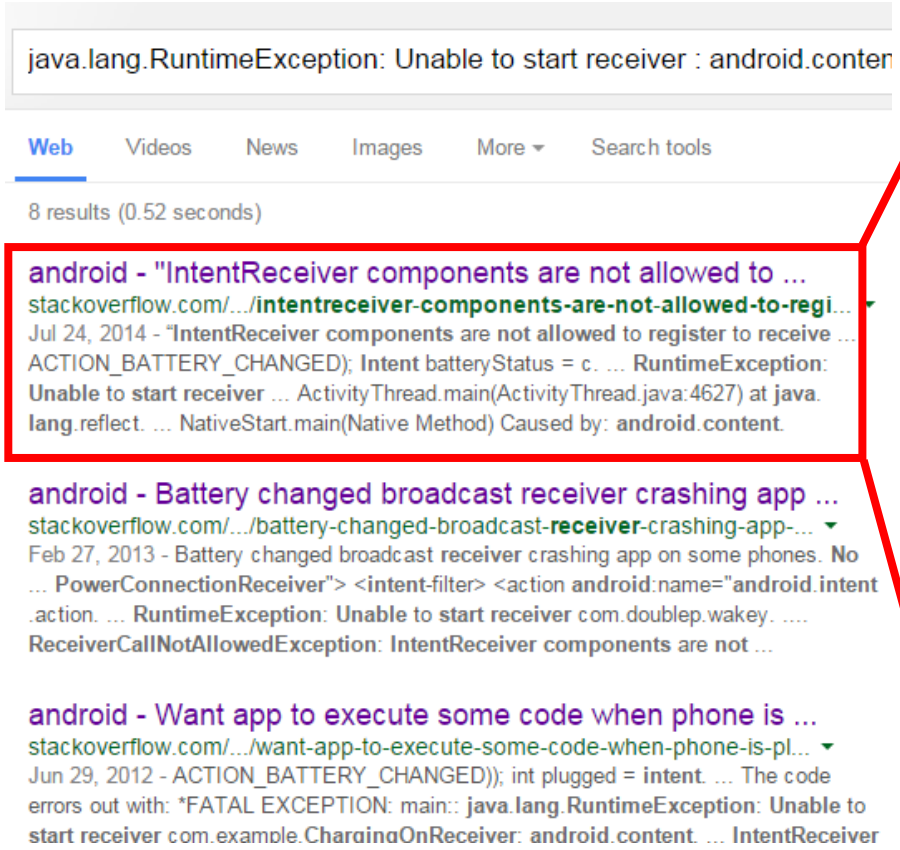
从QA网站学习

- 开发人员遇到未知错误的时候会怎么办？

```
29    public void onReceive (final Context context, final Intent intent) {
30       final int action = intent.getExtras().getInt(KEY_ACTION, -1);
31       final float bl = BatteryHelper.level(context);
32       LOG.i("AlarmReceiver invoked: action=%s bl=%s.", action, bl);
33       switch (action) {
...          ...
51       }
52    }
```

java.lang.RuntimeException: Unable to start receiver com.vaguehope.onosendai.update.AlarmReceiver:

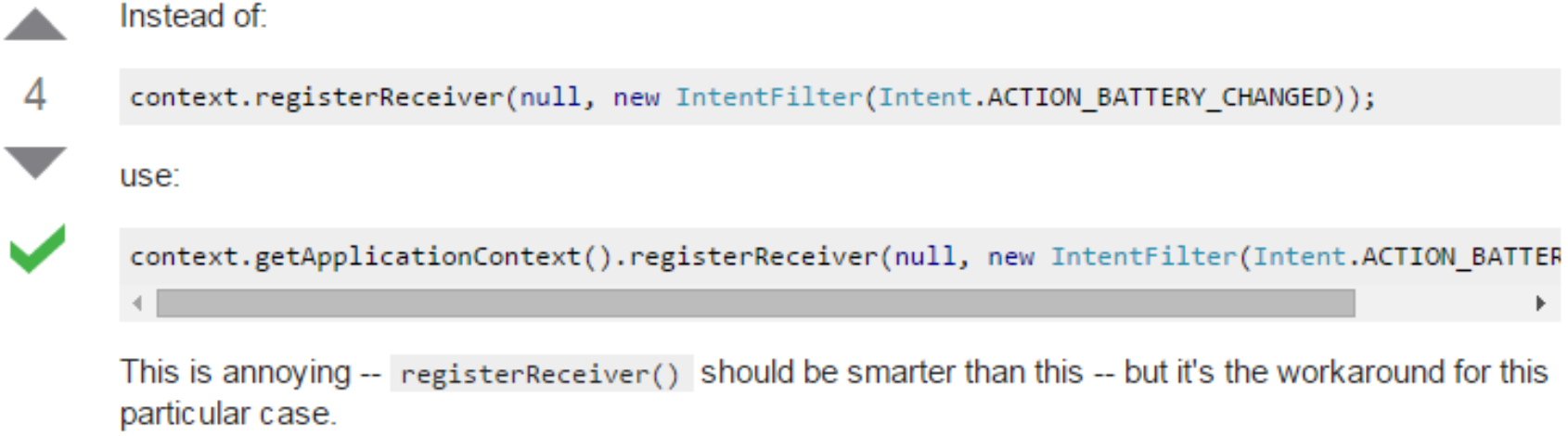
从QA网站学习


java.lang.RuntimeException: Unable to start receiver : android.conten
Web
Videos
News
Images
More
Search tools
8 results (0.52 seconds)
android - "IntentReceiver components are not allowed to ...
stackoverflow.com/.../intentreceiver-components-are-not-allowed-to-regi...
Jul 24, 2014 - "IntentReceiver components are not allowed to register to receive ... ACTION_BATTERY_CHANGED); Intent batteryStatus = c. ... RuntimeException: Unable to start receiver ... ActivityThread.main(ActivityThread.java:4627) at java.lang.reflect. ... NativeStart.main(Native Method) Caused by: android.content.
android - Battery changed broadcast receiver crashing app ...
stackoverflow.com/.../battery-changed-broadcast-receiver-crashing-app-...
Feb 27, 2013 - Battery changed broadcast receiver crashing app on some phones. No ... PowerConnectionReceiver"> <intent-filter> <action android:name="android.intent.action. ... RuntimeException: Unable to start receiver com.doublep.wakey. .... ReceiverCallNotAllowedException: IntentReceiver components are not ...
android - Want app to execute some code when phone is ...
stackoverflow.com/.../want-app-to-execute-some-code-when-phone-is-pl...
Jun 29, 2012 - ACTION_BATTERY_CHANGED)); int plugged = intent. ... The code errors out with: *FATAL EXCEPTION: main:: java.lang.RuntimeException: Unable to start receiver com.example.ChargingOnReceiver: android.content. ... IntentReceiver


stackoverflow
Questions
Tags
Users
Badges
Stack Overflow is a community of 4.7 million programmers, just like you, helping each othe
only takes a minute:
"IntentReceiver components are not allowed to register to receive inter
determine Battery level
Test your app on real Android devices in the cloud.
keynote MOBILE TESTING PRO START YOUR FREE TRIAL ›
2
2
I am trying to get Battery info from my Application following the guidelines at http://developer.android.com/training/monitoring-device-state/battery-monitoring.html
This is the method is came up with to check the battery level:
public void sendBatteryInfoMessage(){
IntentFilter iFilter = new IntentFilter(Intent.ACTION_BATTERY_
Intent batteryStatus = c.registerReceiver(null, iFilter);

从QA网站学习的困难

- 自然语言理解是很困难的



Instead of:

```
context.registerReceiver(null, new IntentFilter(Intent.ACTION_BATTERY_CHANGED));
```

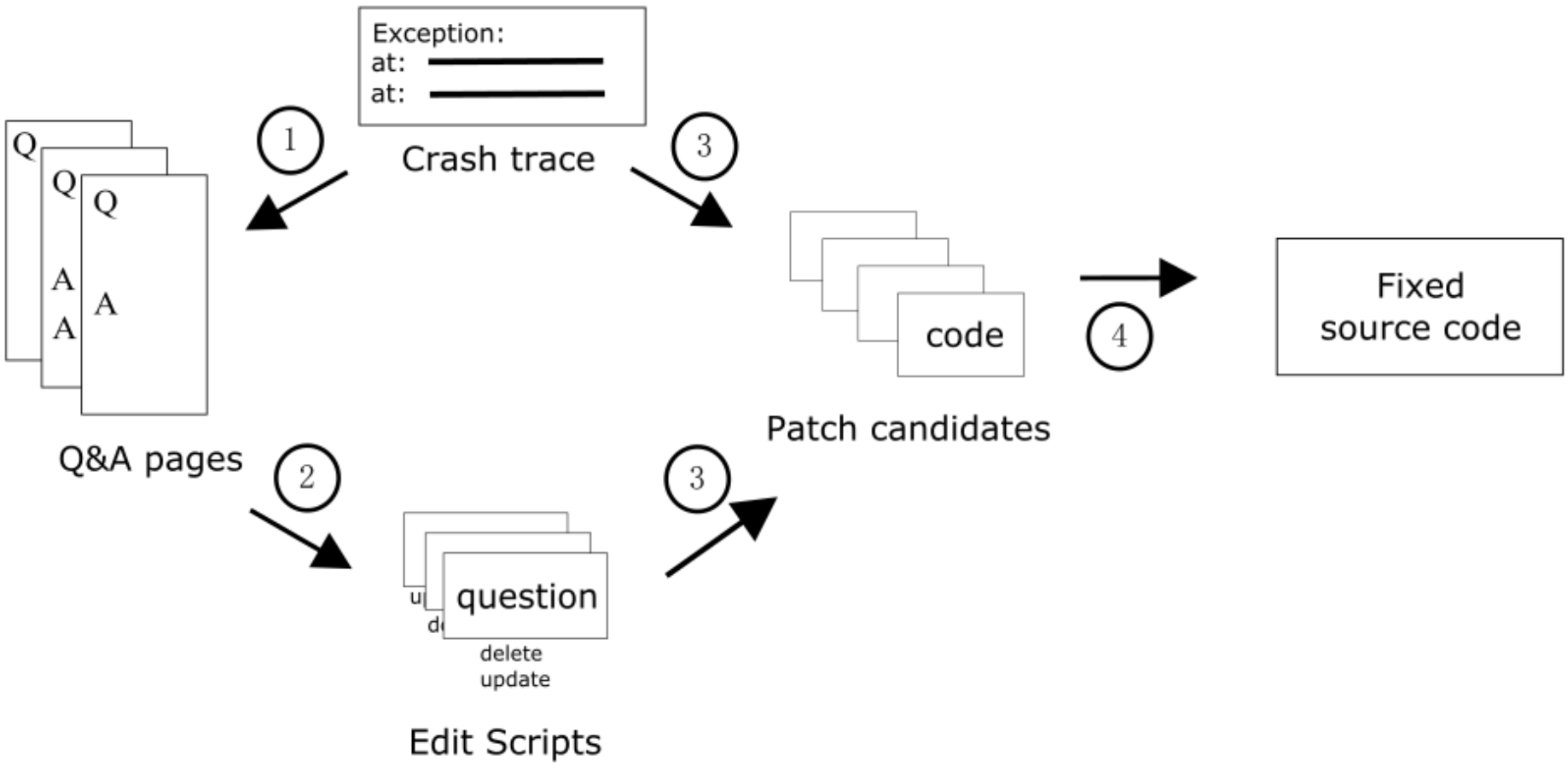
use:

```
context.getApplicationContext().registerReceiver(null, new IntentFilter(Intent.ACTION_BATTER
```

This is annoying -- `registerReceiver()` should be smarter than this -- but it's the workaround for this particular case.

- 观察：程序员常常只用编程语言语言交流的
- 解决方案：直接比较代码片段

方法概览


Exception:
at:
at:
Crash trace
1
3
Q
Q
Q
A
A
A
Q&A pages
code
Patch candidates
4
Fixed
source code
2
3
question
delete
update
Edit Scripts

实验效果

- 24个Android崩溃缺陷
  - 预先人工验证过在StackOverflow上能找到答案

- 正确修复：8
- 错误修复：2
- 正确率：80%
- 召回率：33%

精确条件修复

条件错误是很常见的

```
   lcm = Math.abs(a+b);
+ if (lcm == Integer.MIN_Value)
+   throw new ArithmeticException();
```

缺少边界检查

```
-  if (hours <= 24)
+ if (hours < 24)
      withinOneDay=true;
```

条件过强

```
-  if (a > 0)
+ if (a >= 0)
      nat++;
```

条件过弱

ACS修复系统

- ACS = Accurate Condition Synthesis
- 两组修复模板

条件修改

- 首先定位到有问题的条件，然后试图修改条件
  - 扩展：if ($D) => if ($D || $C)
  - 收缩：if ($D) => if ($D && $C)

返回预期值

- 在出错语句前插入如下语句
  - if ($C) throw $E;
  - if ($C) return $O;

挑战和解决方案

```
  int lcm=Math.abs(
     mulAndCheck(a/gdc(a,b),b));
+if (lcm == Integer.MIN_VALUE) {
+   throw new ArithmeticException();
+}
  return lcm;
```

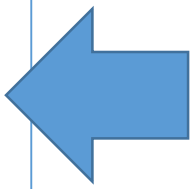
测试 1:
Input: a = 1, b = 50
Oracle: lcm = 50

测试 2:
Input: a = Integer.MIN_VALUE, b = 1
Oracle: Expected(ArithmeticException)

正确条件:
lcm == Integer.MIN_VALUE

可以通过测试的条件:

- a > 1
- b == 1
- lcm != 50
- …



排序

排序方法1:
按数据依赖对变量排序

- 变量使用局部性：最近被赋值的变量更有可能被使用。

- 根据数据依赖对变量排序
  - lcm = Math.abs(mulAndCheck(a/gdc(a, b), b))
  - lcm > a, lcm > b

排序方法2:
根据Java文档过滤变量

```
/** ...
 * @throws IllegalArgumentException if initial is not between
 * min and max (even if it <em>is</em> a root)
**/
```

抛出IllegalArgumentException时，只考虑将“initial”变量用在条件里

排序方法3:
根据现有代码对操作排序

- 在变量上使用的操作跟该条件的上下文紧密相关

变量类型

```
Vector v = …;
if (v == null) return 0;
```

变量名字

```
int hours = …;
if (hours < 24)
    withinOneDay=true;
```

方法名字

```
int factorial() {
  …
  if (n < 21) {
    …
```

- 根据已有的代码库统计条件概率

Defects4J上的验证

Approach	Correct	Incorrect	Precision	Recall
ACS	18	5	78.3%	8.0%
jGenProg	5	22	18.5%	2.2%
Nopol	5	30	14.3%	2.2%
xPAR	3	–[4]	–[4]	1.3%[2]
HistoricalFix[1]	10(16)[3]	–[4]	–[4]	4.5%(7.1%)[2,3]

内存管理

- 安全攸关软件的开发必然涉及内存管理问题
- 软件工程经典问题，数千篇论文
- 垃圾回收
  - 广泛用于Java, Go等大量新语言
  - 通过动态扫描内存发现需要回收的内存

垃圾回收 vs 安全攸关软件

- 大量系统资源无法通过垃圾回收管理
  - 文件句柄、线程锁等
- 在安全攸关软件中，内存对象常常也无法通过垃圾回收管理
  - 实时嵌入式系统，运行资源有限
  - 大数据处理系统，垃圾收集耗时过长
    - 处理数据量达到10G时，垃圾收集运行时间占程序运行总时间一半以上

内存泄露的例子


```c
#include <stdlib.h>
#include <stdio.h>

void f(int *p, int **q){
  *q = p;
}
void g(int *p){
  free(p);
}
int h(int size, int num, int sum){
  int *p = (int*)malloc(sizeof(int)*size);
  int **q = (int**)malloc(sizeof(int*));
  if (size == 0)
    g(p);
  else
    for (int i = 0; i < size; ++i)
      if (p[i] != num){
        f(p, q);
        sum += (*q)[i];
      }
      else
        return i;
  printf("%d", sum);
  return sum;
}
```

内存泄露的例子

```
1  #include <stdlib.h>
2  #include <stdio.h>
3
4  void f(int *p, int **q){
5    *q = p;
6  }
7  void g(int *p){
8    free(p);
9  }
10 int h(int size, int num, int sum){
11   int *p = (int*)malloc(sizeof(int)*size);
12   int **q = (int**)malloc(sizeof(int*));
13   if (size == 0)
14     g(p);
15   else
16     for (int i = 0; i < size; ++i)
17       if (p[i] != num){
18         f(p, q);
19         sum += (*q)[i];
20       }
21       else
22         return i;
23   printf("%d", sum);
24   return sum;
25 }
```

free(q);

free(p);

free(q);

free(p);

free(q);

修复内存泄露仍是难题

- 需要考虑多个条件
  - 内存释放前必须已分配
  - 内存释放时必须有能从栈上访问的路径
  - 在任意路径上，内存不能被释放两次
  - 在任意路径上，内存使用前不能被释放
- 漏掉任何一条都将导致致命错误
- 实践中，常常出现发现内存泄露但不敢修改的情况

研究成果：自动内存修复技术

- 通过对C程序代码的分析，自动查找内存泄露并修复
- 保证修复的正确性
  - 对于任意插入的free语句和任意执行路径
    - 释放前分配：在执行到free之前所指的内存已经分配
    - 无双重释放：在该路径上没有任何其他free语句释放同一块内存
    - 无释放后使用：在该free之后所释放内存不能再被使用
- 能在较短时间内完成对大型程序的分析工作

技术路线和创新

- 反复使用数据流分析

- 处理各种复杂情况
  - 循环、全局变量、多重分配、空指针判断等问题

修复结果

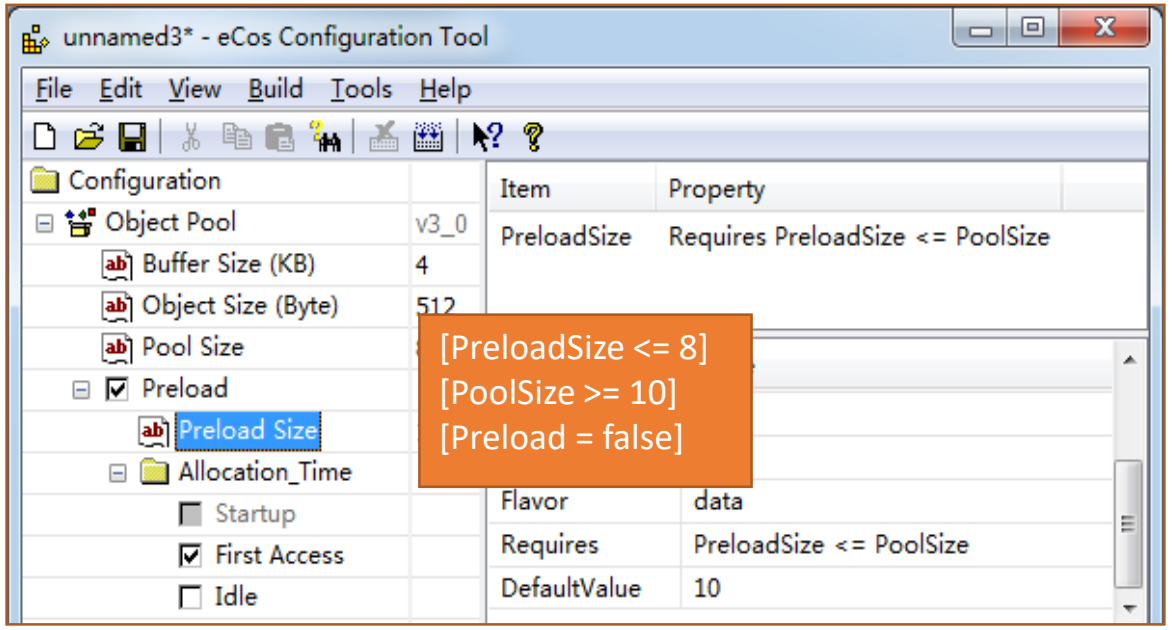
Program	#Fixed	#Maximum Detected	Percentage(%)	#Fixes	#Useless Fixes
art	0	1	0	0	0
equake	0	0	N/A	0	0
mcf	0	0	N/A	0	0
bzip2	1	1	100	1	0
gzip	1	1	100	1	0
parser	0	0	N/A	0	0
ammp	20	20	100	36	0
vpr	0	0	N/A	0	0
crafty	0	0	N/A	0	0
twolf	0	5	0	0	0
mesa	0	9	0	0	0
vortex	0	0	N/A	0	0
perlbmk	1	8	13	1	0
gap	0	0	N/A	0	0
gcc	2	44	5	2	0
total	25	89	28	41	0

时间开销

Program	Compiling and Linking (sec)	LeakFix (sec)		Total (sec)	Percentage(%)
		Pointer Analysis	Fix Analysis		
art	0.20	0.02	0.01	0.23	13.0
equake	0.21	0.01	0.02	0.24	12.5
mcf	1.19	0.02	0.01	1.22	2.5
bzip2	0.36	0.03	0.02	0.41	12.2
gzip	1.31	0.04	0.04	1.39	5.8
parser	1.68	0.18	0.07	1.93	13.0
ammp	2.98	0.12	0.37	3.47	14.1
vpr	2.51	0.20	0.31	3.02	16.9
crafty	3.53	0.16	0.23	3.92	9.9
twolf	6.22	0.27	0.20	6.69	7.0
mesa	9.36	5.36	5.97	20.69	54.8
vortex	9.00	0.94	0.83	10.77	16.4
perlbmk	9.50	18.20	39.20	66.90	85.8
gap	6.03	7.36	22.36	35.75	83.1
gcc	10.99	31.76	95.81	142.99	89.2

其他成果: 软件配置交互式修复

- Linux内核包含6000余条配置项，1000余条约束
- 嵌入式操作系统eCos包含1000余配置项，600余条约束
- 调研表明：用户往往不知如何修复配置中的错误。

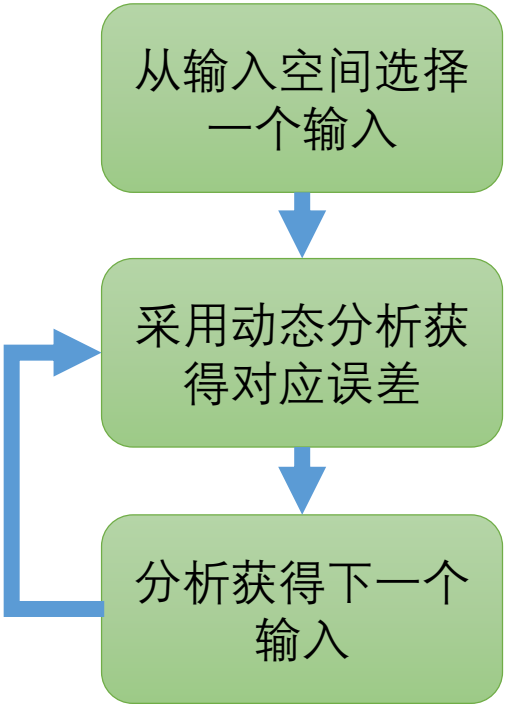


全自动生成修复列表，首次同时保证修复的正确性、最优性和完整性。

成果发表于IEEE Transactions on Software Engineering，并被选为网站首页论文

其他成果：浮点误差测试技术

误差常常导致灾难性后果



全自动查找程序中的浮点误差
发现GSL科学计算库中的多处潜在误差问题
相关研究成果发表于ICSE'15，并入围ACM SIGSOFT Distinguished Paper Award候选