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

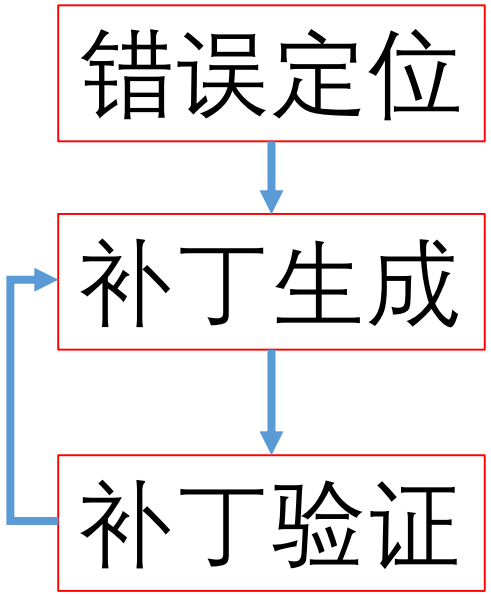
- 2000~2004，电子科技大学本科
- 2004~2006，北京大学研究生
  - 导师：梅宏、杨芙清
- 2006~2009，日本东京大学博士
  - 导师：胡振江、武市正人
- 2009~2011，加拿大滑铁卢大学博士后
  - 导师：Krzysztof Czarnecki
- 2012~，北京大学“百人计划”研究员（Tenure-Track）
- 研究方向：软件分析、编程语言设计

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：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

“生成-验证”缺陷修复

输入：一个程序和一组测试，至少有一个测试没有通过
输出：一个补丁，可以使程序通过所有测试



代表性工作

- GenProg
  - [Westley Weimer: ICSE'09, GECCO'09, CACM'10, ICSE'12]
  - 方法：
    - 复制其他语句来替换/插入到之前/删除错误语句
    - 采用遗传算法从基本操作合成补丁
  - 实证研究：55/105，8$/bug
- 引发一系列相关工作
  - AutoFiix, Nopol, RSRepair, MintHint, AutoRepair, SemFix, DirectFix, SPR…
- 程序员的前景一片光明，“躺着也能把钱挣了”的时代眼看就要到来

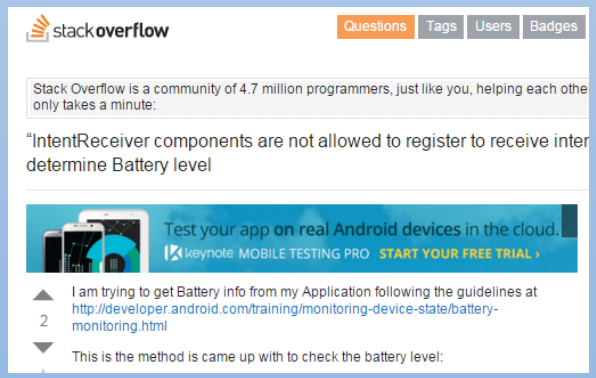
转折

- [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, Angelix
  - 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

从QA网站学习 [ASE15]



精准条件修复 [ICSE17]

过滤错误的修复[ICSE18]

[ASE15] Qing Gao, Hansheng Zhang, Jie Wang, Yingfei Xiong, Lu Zhang, Hong Mei. Fixing Recurring Crash Bugs via Analyzing Q&A Sites. ASE'15

[ICSE17] Yingfei Xiong, Jie Wang, Runfa Yan, Jiachen Zhang, Shi Han, Gang Huang, Lu Zhang. Precise Condition Synthesis for Program Repair. ICSE'17

[ICSE18] Xinyuan Liu, Muhan Zeng, Yingfei Xiong, Lu Zhang, Gang Huang. Identifying Patch Correctness in Test-Based Automatic Program Repair. ICSE 2018

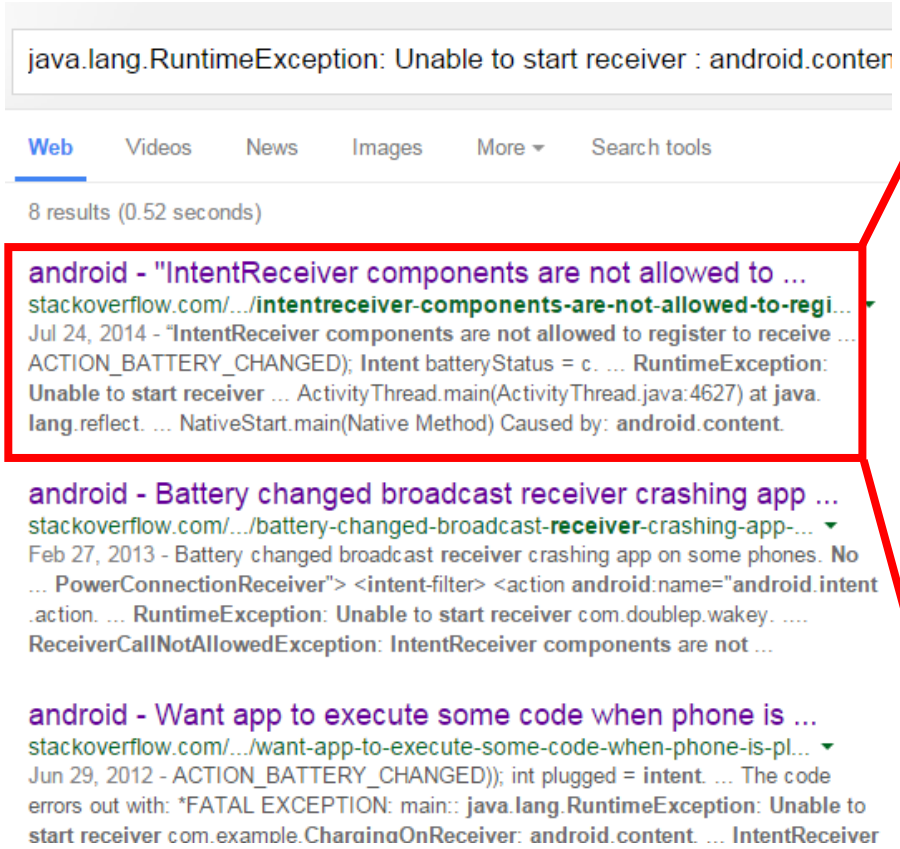
从QA网站学习

- 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



挑战：自然语言理解是很困难的

从QA网站学习

- 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

My broadcast receiver is

```
    @Override
    public void onReceive(Context context, Intent intent) {

        Bundle extras = intent.getExtras();
        String message = extras != null ? extras.getString("com.parse.Data")
                : "";

        Log.e("message ", " " + message);
        JSONObject jObject;
        try {
            jObject = new JSONObject(message);
            //objectId = jObject.getString("id");
            time = jObject.getString("time");
            msg = jObject.getString("title");
            title = jObject.getString("msg");
                GCMMessage gcmMessage = new GCMMessage();

            //gcmMessage.setMsg_id(1);
            gcmMessage.setMsg_body(msg);
            gcmMessage.setMsg_title(title);
            gcmMessage.setType(0);
            gcmMessage.setDateTime(time);

                DatabaseUtil.insertMessage(context, gcmMessage);

        }
        catch (JSONException e) {
            e.printStackTrace();
        }
}
```

When I reboot my phone then also it showing same error..., otherwise it is working fine.

问题

I'll have a guess that `message` has the value of `""` or `NULL`

1

```
    JSONObject jObject;
    try {
        if (message != null && !message.equals("") {
          jObject = new JSONObject(message);
          //objectId = jObject.getString("id");
          time = jObject.getString("time");
          msg = jObject.getString("title");
          title = jObject.getString("msg");
          GCMMessage gcmMessage = new GCMMessage();

          //gcmMessage.setMsg_id(1);
          gcmMessage.setMsg_body(msg);
          gcmMessage.setMsg_title(title);
          gcmMessage.setType(0);
          gcmMessage.setDateTime(time);

          DatabaseUtil.insertMessage(context, gcmMessage);
        }

      }
      catch (JSONException e) {
        e.printStackTrace();
      }
```

Instead of:

4

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

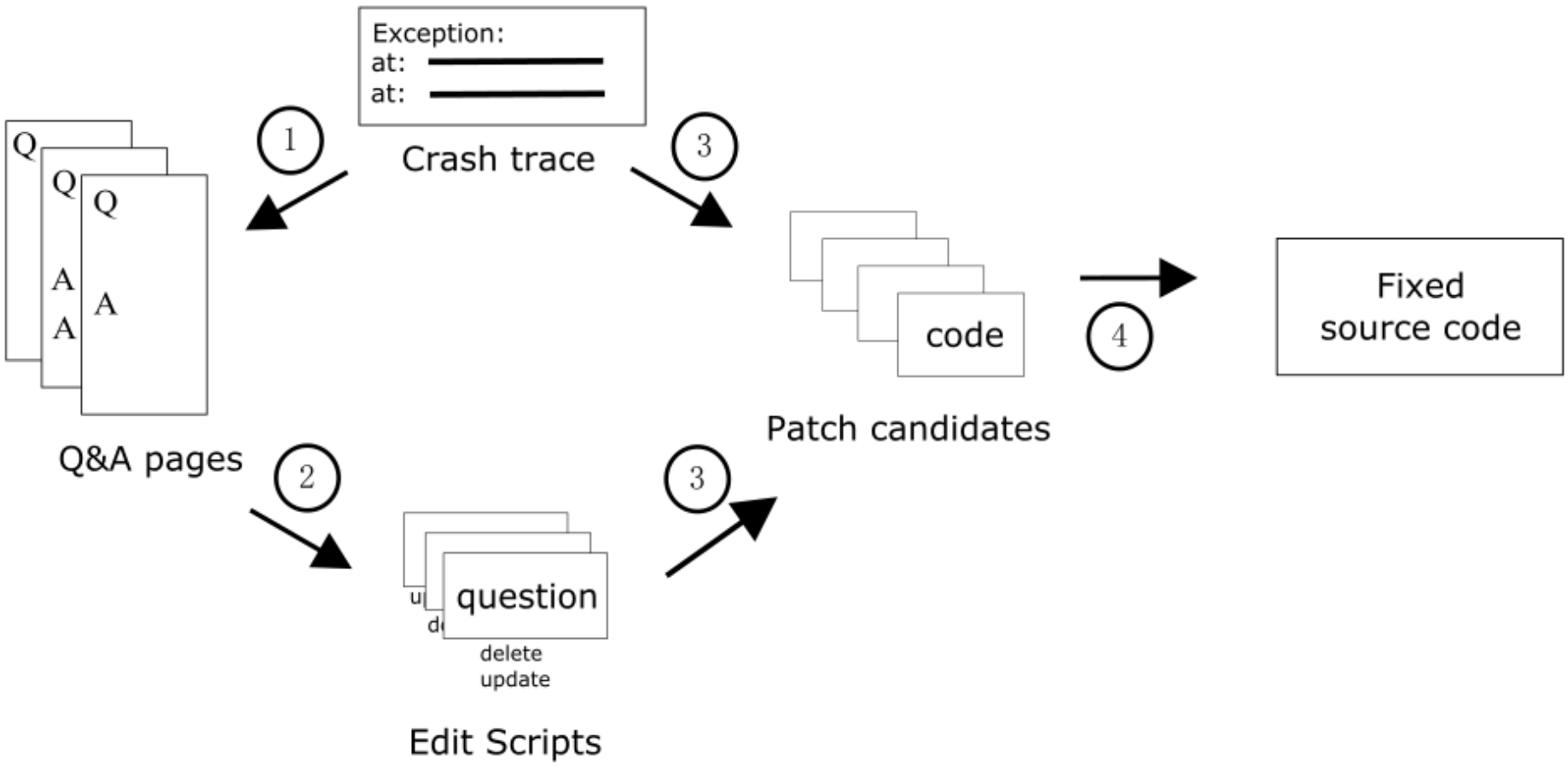
use:

```
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.

答案

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%

是否能修复更多缺陷？

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;
```

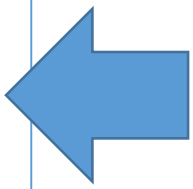
测试 1:
Input: a = 1, b = 50
Oracle: lcm = 50

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:

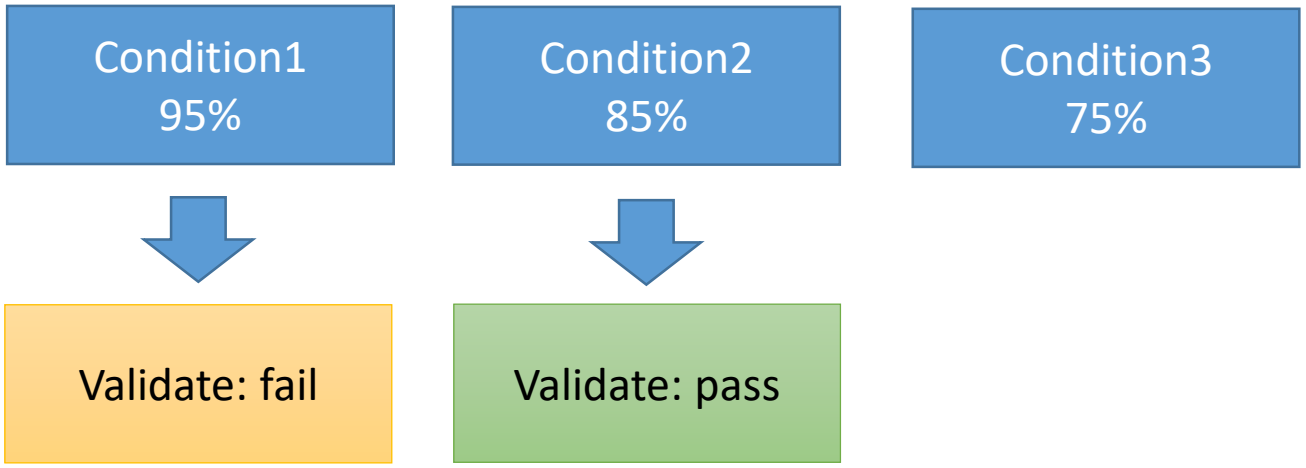
- a > 1
- b == 1
- lcm != 50
- …



排序

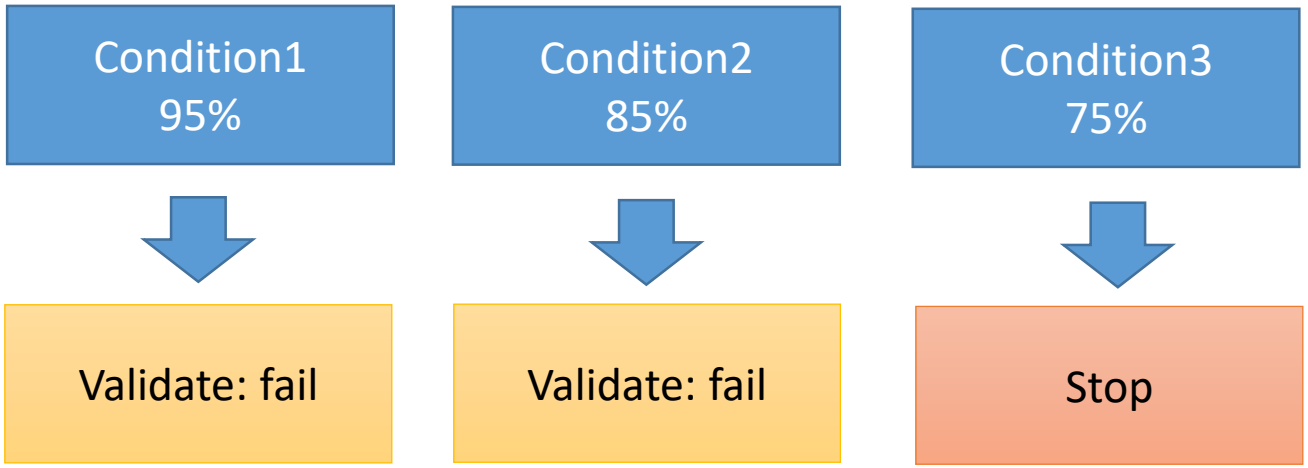
基本思路：对条件进行排序

- 按条件的正确可能性进行排序
- 用测试逐个验证
- 当可能性太低的时候放弃验证



基本思路：对条件进行排序

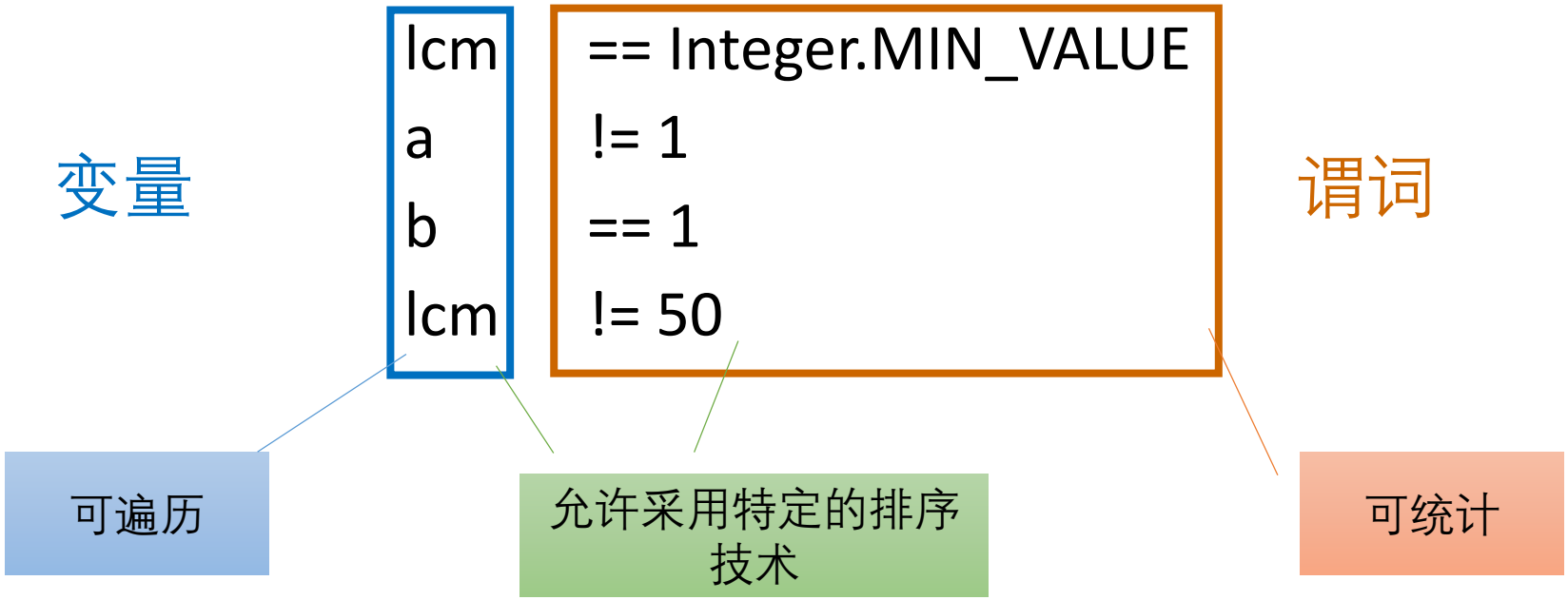
- 按条件的正确可能性进行排序
- 用测试逐个验证
- 当可能性太低的时候放弃验证



按正确可能性排序很困难

- 正确条件的空间较大
  - 无法直接对空间中的条件排序
  - 无法通过统计得到概率

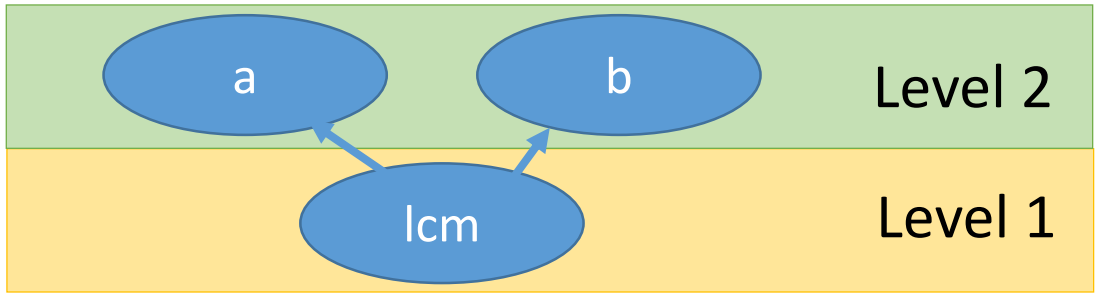
解决方案：分治



首先排序变量
然后根据变量排序谓词

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))



- 只考虑前两层的变量

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

- 数据集： Defects4J上的四个项目
  - Time, Lang, Math, Chart
  - 总共224个缺陷

Approach	Correct	Incorrect	Precision	Recall
ACS	**17**	**6**	**73.9%**	**7.5%**
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]

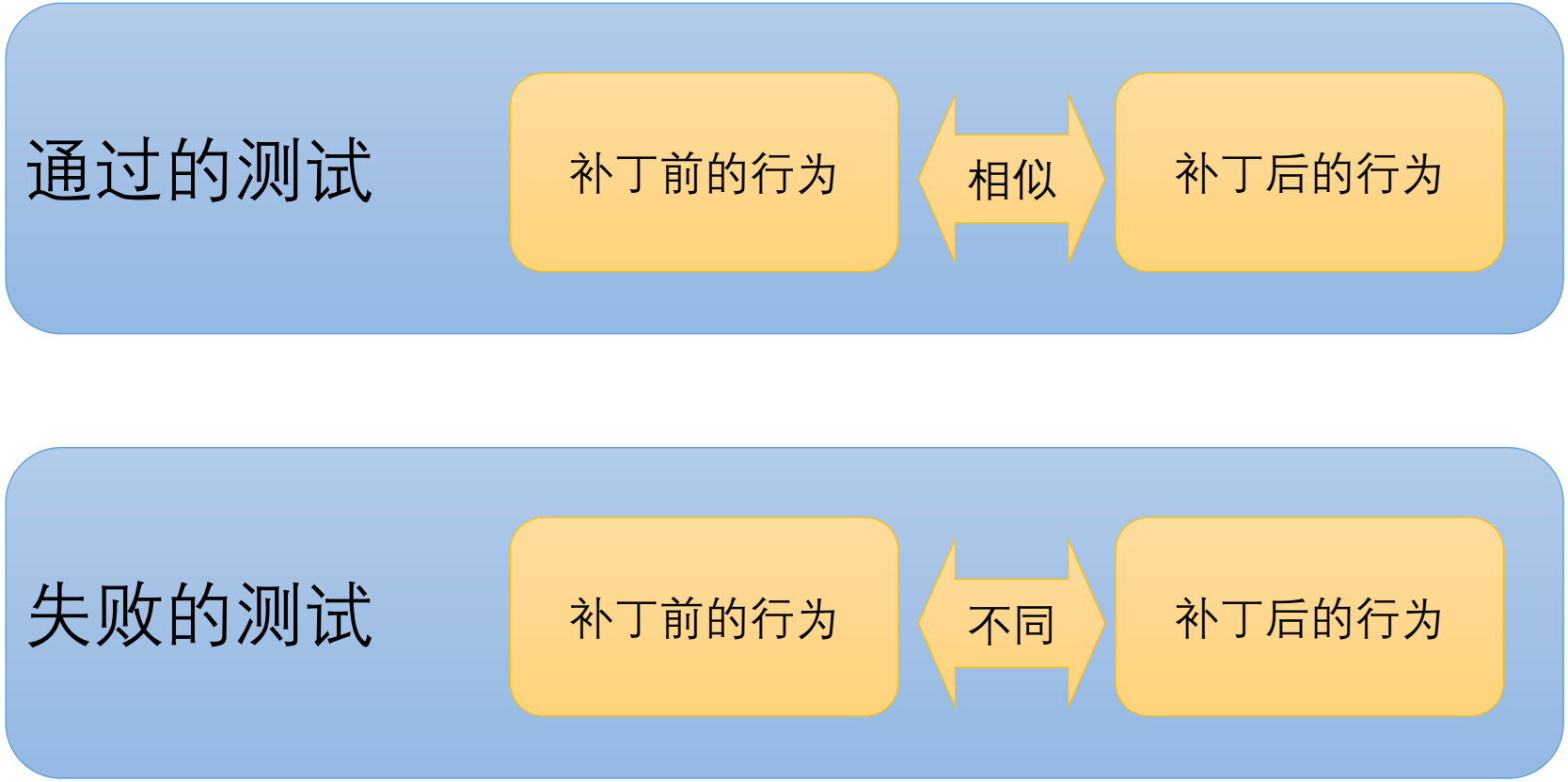
是否还能进一步提高准确率？

思路：修复正确率低主要是测试集太弱

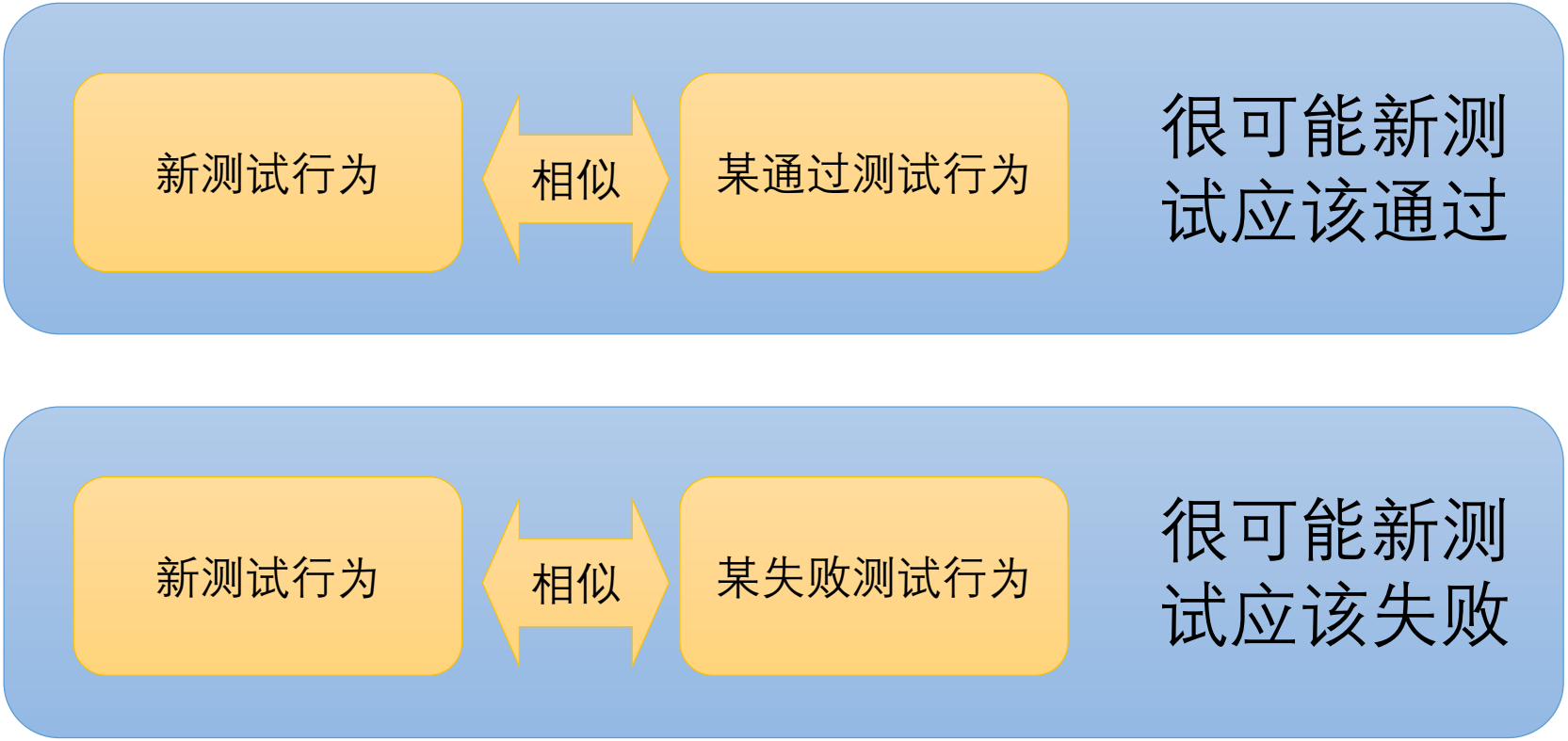
能否自动增强测试集？

增强测试集

针对预言的启发式规则PATCHSIM

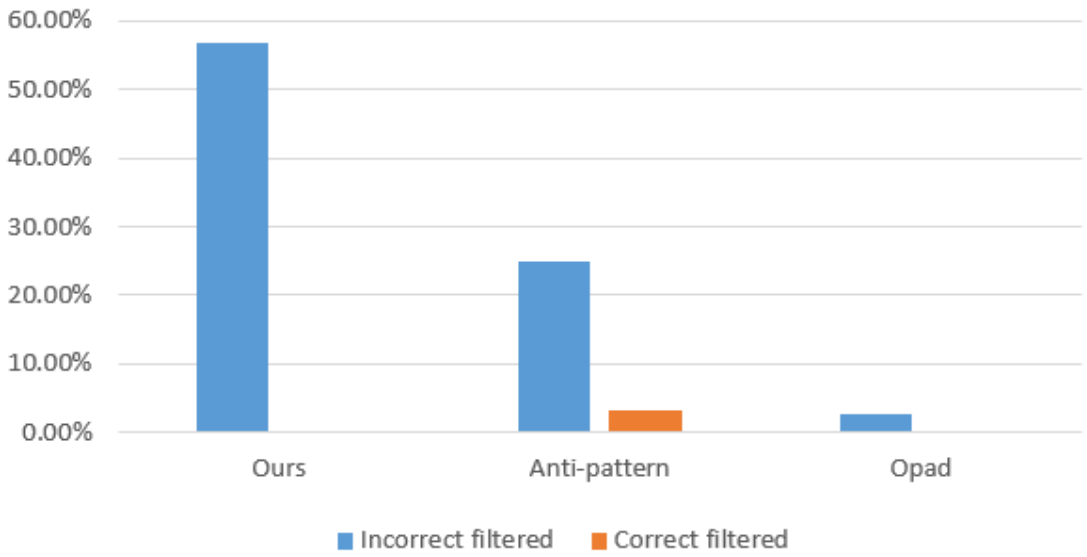


针对输入的启发式规则TESTSIM



验证结果

- 139个不同工具生成的补丁
  - 30个正确补丁，109个错误补丁



- 成功过滤56.9%的错误补丁，并且没有误伤正确补丁
- 将ACS的正确率提升到了85%

愿景

- 长远目标：自动程序开发
- 路线图：不断挑战修复更困难的缺陷
  - Issues = bug reports + feature requests

小组其他工作

特定缺陷的查找与修复	静态分析加速	交互式修复
• 内存泄露 • 浮点误差 • 编译器缺陷	• 变异分析加速 • 静态分析摘要	• 软件配置的范围修复

小组进行中工作

- 可解释的缺陷修复
- 基于统计的程序综合/定理证明综合
- 缺陷修复基础平台