软件测试实训项目测试报告

撰写人：赵洋

最后更新时间：2018/08/23

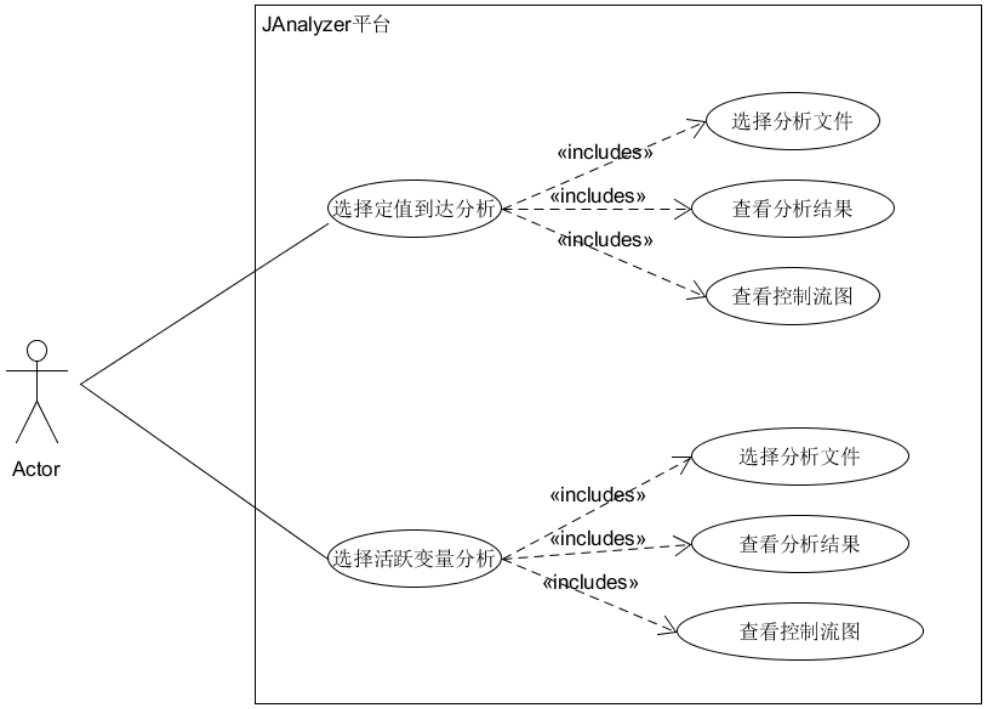
目录

一、引言

a) 目的

对我们设计的结果进行测试分析

1.设计多种测试用例来对我们以下几个结果进行分析



2.同时对生成的 GUI 界面进行分析测试

对界面美观与按键测试分析

b) 背景

JAnalyzer 有完成抽象语法树生成、名字表生成和软件结构提取这三个基础构件。这三个功能是对源代码做静态分析的基础，任何对软件源代码的理解和分析都需要在生成抽象语法树和名字表的基础上，基于软件的结构完成。

所以需要对我们所做的软件，在 AST,Variable Table 以及 Software Structure 所提取结果检测。对定值和活跃变量结果对比分析，同时对可

视化界面的结果进行测试性能进行分析以及外观用户使用进行测试。

c) 缩略语

简单来说，通过设计多个 JAVA 测试代码用例进行分析同时对 GUI 的界面进行分析。

d) 参考文献

[1]陆卫东,金成植.基于活跃变量分析的流图语言的部分求值器[J].软件学报,1997(01):30-36.

[2] JAnalyzer 基础构建开发文档

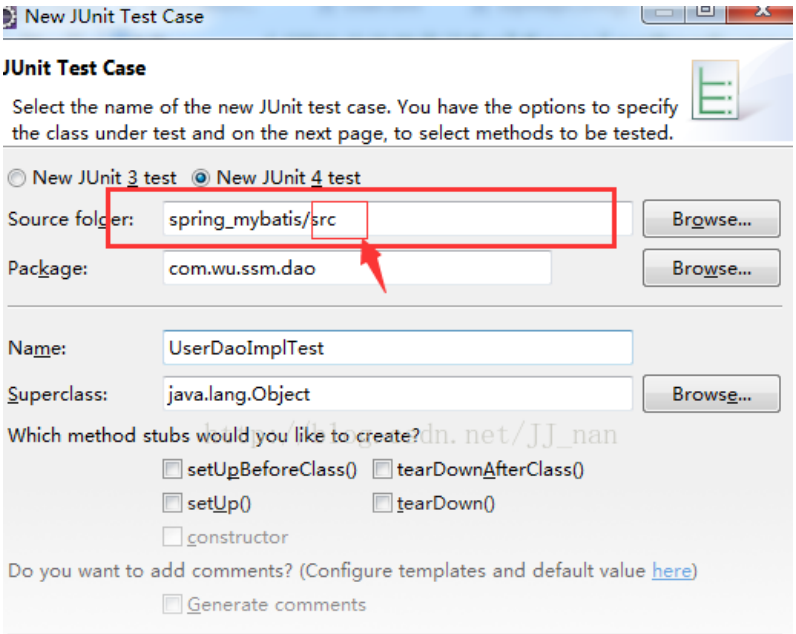
[3]网上相关博客和论坛

[4]维基百科

二、测试概要

a) 测试方法

使用 Junit 进行分析



同时 使用 guidancer 进行 GUI 界面的分析

b) 范围

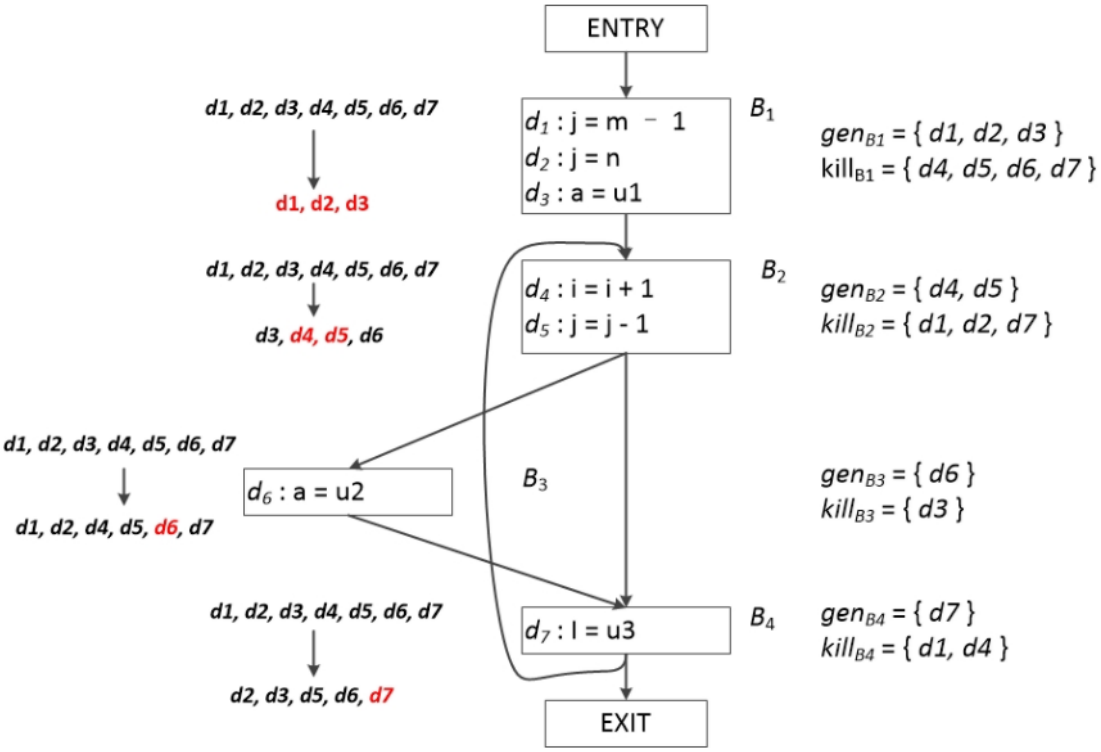
测试范围

我们要对数据流基本的测试以及分析

到达定值:

我们生成了多个达到定值的测试用例

分别测试 Java 代码在不同代码环境下到达定值的分析如图是我们测试的一个举例



我们通过常量变量多种范围进行测试

活跃变量分析:

对于活跃变量，我们在代码中间分别添加不同的字段

从而在不同的代码段进行比较，去查看活跃变量分析结果是否和网上和我们标准测试相同

c) 测试环境

Eclipse jave ee

Windows 10

Gve dev 环境下

d) 工具

配置好的 Junit 和 Guidancer

三、测试结果与缺陷分析

a) 功能

分别运行到达定值和活跃变量的功能分析

到达定值的基本分析是到位的

效果图与基本验证是正确的。

所以基本的到达定值的功能是完成了的。

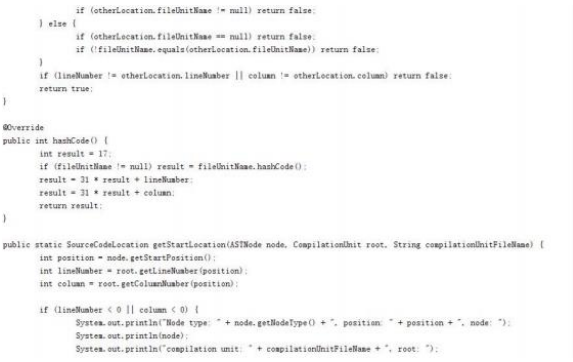
而活跃变量分析同样如下

对后面数据流会不会使用到活跃变量我们进行了相关代码的测试

```
for each node n in CFG
in[n] = ∅; out[n] = ∅;          // 初始化
repeat
    for each node n in CFG
        in' [n] = in[n]          // 保存当前结果
        out' [n] = out[n]
        in[n] = use[n] U (out[n] - def[n]) // 数据流方程
        out[n] = U s in succ[n]in[s]
    // 判断是否收敛
untill in' [n] = in[n] and out' n[n] = out[n] for all n
```

通过上述代码

我们分析多个测试样例



得到结果和我们需求一致

接着我们使用 GUIdancer 对操作界面进行测试

只有对应的两个按键我们通过

JAVAee 使用 GUIdancer 测试

都有响应而且人工检测测试结果正确是可以使用的

b) 性能

相关性能速度分成两个算法考虑

到达定值算法：
输入：一个流图，其中每个基本块 *B* 的 *kill(B)* 集和 *gen(B)* 集都已经计算出来了。
输出：到达流图中各个基本块 *B* 的入口点和出口点的定值的集合，即 *IN[B]* 和 *OUT[B]* 。
方法：我们使用迭代的方法来求解。一开始，我们"估计"对于所有基本块 *B* 都有 *OUT[B]* = Ø，并逐步逼近想要的 *IN* 和 *OUT* 值。因为我们必须不停地迭代直到各个 *IN* 值（因此各个 *OUT* 值也）收敛，所以我们使用一个 *bool* 变量 *change* 来记录每次扫描各基本块时是否有 *OUT* 值发生改变。

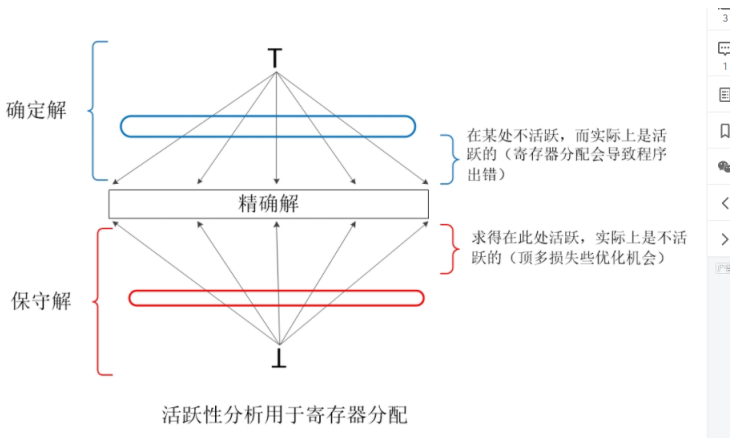
```
1)  OUT[ENTRY] = Ø;
2)  for(除ENTRY之外的每个基本块B) OUT[B] = Ø;
3)  while(某个OUT值发生改变)
4)    for(除ENTRY之外的每个基本块B) {
5)      IN[B] = U P是B的一个前驱 OUT[P]; // 应用控制流方程
6)      OUT[B] = genB U (IN[B] - killB); //应用数据流方程
      }
```

我们发现逆后序性能最高

而且会达到保守解

这样复杂度也是 O（V*logV）的节点

而对于活跃变量



活跃性分析用于寄存器分配

也是具有最小活跃变量向精确解靠近的算法分析过程。

性能是比较好的。

四、测试结论与建议

a) 项目概况

整个项目都是对两个算法的优化与完成工程实现

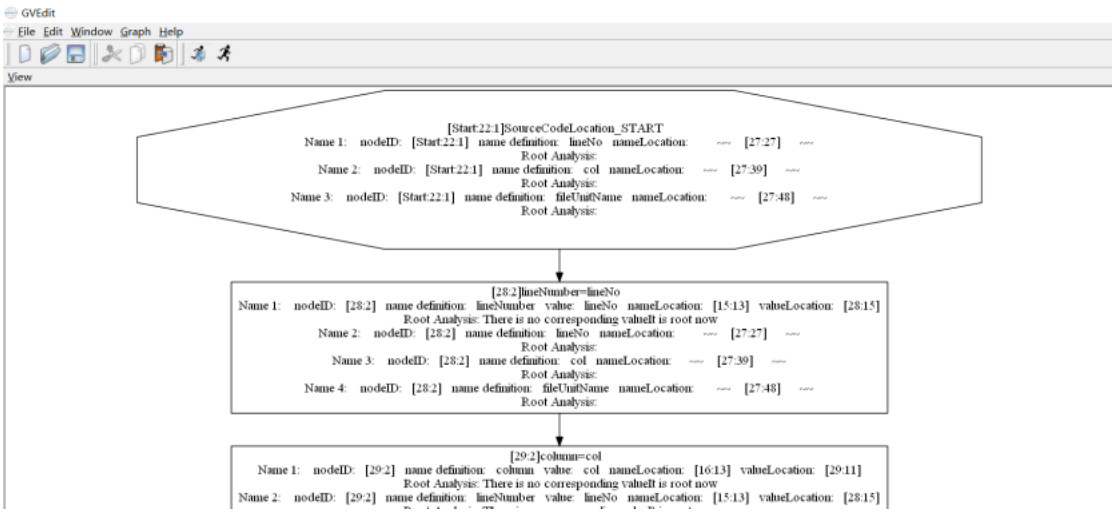
同时生成可视化界面

测试效果是非常好的

总共 15 组代码测试样例

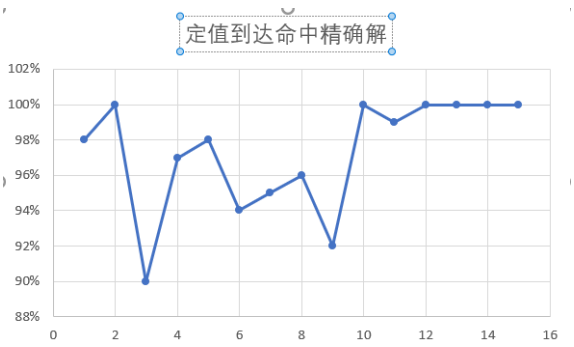
分析到达定值和活跃变量基本上都是全部 Hit

其中一组生成的样例如下



这是定值到达分析的控制流

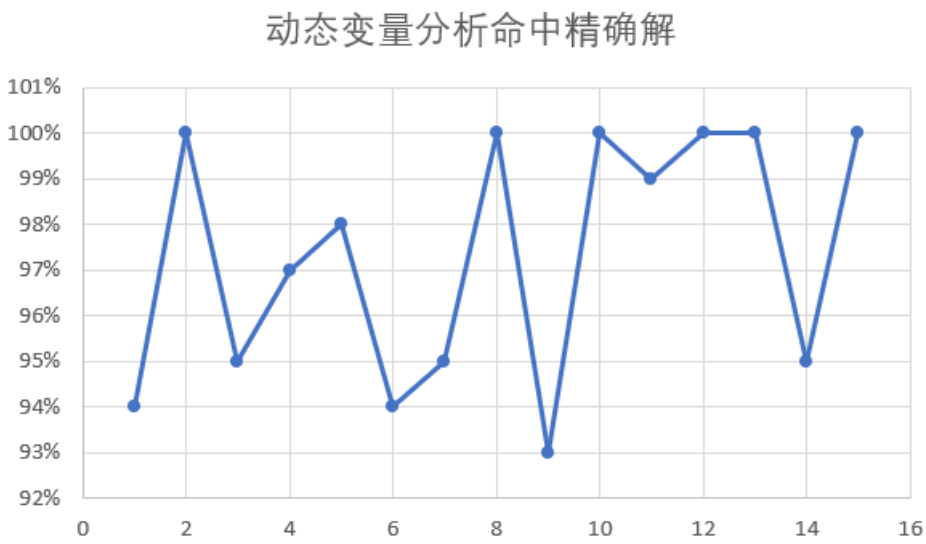
其中基本的到达精确解全部命中



基本都在 90-100 的区间中

所以正确率非常高

而活跃变量分析结果如下



也是基本在 94-100

命中率相对于到达定值更高

可能与测试样例有关更加准确性能也更好

而 GUI 制作也非常好

所以按键基本要求都能到达

.dot 文件也能生成能够满足用户对该工具的使用

b) 项目时间

2018 年 7 月-8 月

c) 项目情况

整体到达定值分析和活跃变量分析

该得到的具有最小活跃变量（亦即尽量向精确解靠近）的集合。

而到达定值会更加超越保守解，从测试结果分析来看

主要原因如下

算法不断从空向到达定值结果越来越多的方向靠近，最终会跨越精确解到达保守解的部分，主要因为两个原因导致一定会越过精确解：

（1）不考虑路径条件，假设所有路径都可达；这样某些定值最终会到达他们本来到达不了的地方

（2）存在别名时，给无法确认的“别名”赋值时，给所有变量添加一个定值

d) 项目性能总结

测试整体性能都非常好

算法在分析测试代码基本的时间复杂度都能够达到

能够基本达到要求

GUI 和图像生成都与实际结果相符，并且能够很清晰看到。

总的来说，测试结果与实际活跃变量和到达定值相符，GUI 界面简洁容易使用同时生成图像清晰易懂。