**DARWIN: An Approach for Debugging** **Evolving Programs**

MF1233055 周昊一

程序员并不会凭空一下子写出整个程序.不如说,程序是慢慢演化的.在工业界的软件开发过程中,软件演化的复杂过程是通过程序版本的签入来控制的.对演化过程中的程序的验证就成了一个很大的问题.据说维护一个软件与管理它的演化的开销要占去整个项目的90%.

为了解决这样的问题,前人已经在软件测试方法学方面进行了长时间的研究.回归测试就是一个众所周知的概念.在最简单的情况下,就是在程序改变后用原来的测试用例再次进行测试.但是即使测试中出现了错误,也无法定位错误产生的真正原因.

假设一个修改过的程序P’在没有通过一个测试t,而原程序P通过了,则我们的目标就是生成一个bug报告Rep(t,P,P’)来解释为什么P’不能通过测试t而P能.实际上,我们并不要求P’和P是同一个程序,只要求P和P’对于他们通常的输入来说是等价的.

现有的方法有以下几种:

1. 变化检测,检查P’中改变了的地方,这些地方很可能就与bug有关.但是这种方法无法处理P和P’是两种完全不同的实现的情况.
2. 踪迹比较,将成功执行的测试的踪迹和失败时的踪迹进行比较.

在本文中,作者提出了一种自动化,并且可伸缩的解决方案,用于定位回归测试中发现的bug.

基本的思路如下:

1. 对于一个在P’中失败而在P中通过的测试t,比较t在P和P’中的踪迹
2. 自动生成另一个测试用例t’满足:1)t’和t在P中的路径一致;2)t’和t在P’中的路径不一致.
3. 通过比较t’和t在P’中的不同路径得出bug报告.

对于三种常见的错误类型:

1. 分支错误,由于这种方法就是基于控制流的,因此天然地适合定位分支错误.
2. 赋值错误,对于不影响控制流的错误,这种方法无法直接定位到根源,但是可以用一种策略将这种错误转化为分支错误.
3. 代码缺失,通常代码缺失会导致P比P’中有更多的路径,通过比较t在P和P’中的踪迹可以生成t’,进而定位bug.