*论文题目：Isomorphic Regression Testing: Executing Uncovered Branches without Test Augmentation. (中文题目：同构回归测试)

*作者：张洁、娄一翎、张令明、郝丹、张路、梅宏

*单位：北京大学博士研究生张洁、博士研究生娄一翎，美国德克萨斯大学达拉斯分校张令明教授，北京大学郝丹教授、张路教授、梅宏教授。

联系方式：email 地址 zhangjie_marina@pku.edu.cn

*文章发表信息：(论文集，请提供论文集名称，出版社，出版时间；杂志，请提供杂志名称，年，卷，期，页码)

论文集：FSE 2016

出版时间：2016 年 11 月

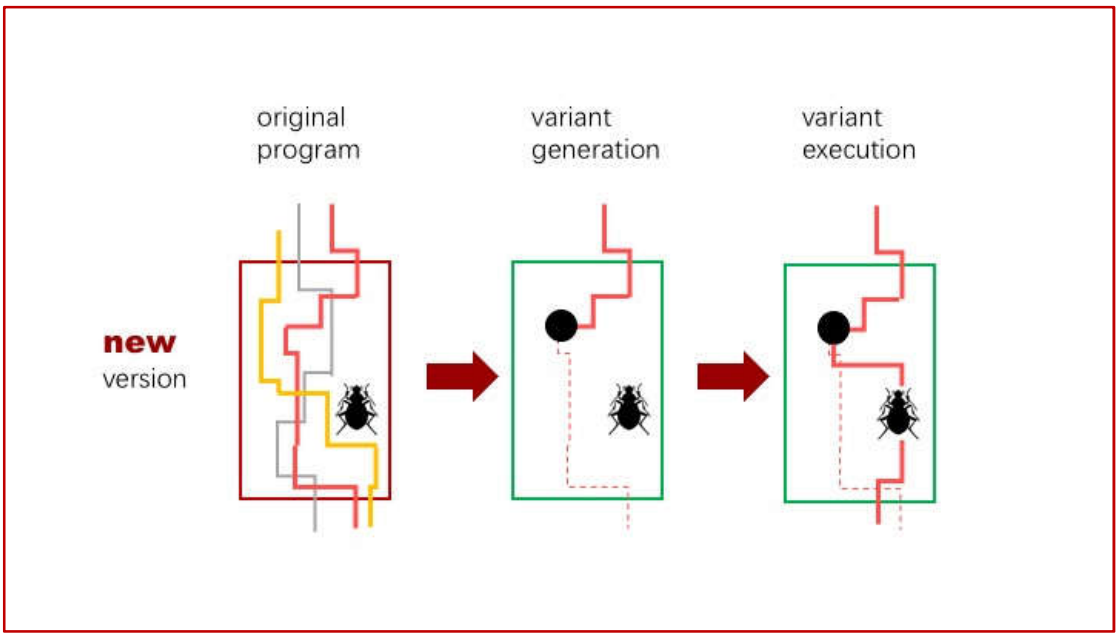
*原文链接地址：http://dl.acm.org/citation.cfm?doid=2950290.2950313

*正文：

同构回归测试

在软件开发过程中，软件更新时常发生，怎样保证软件更新后的软件质量呢？这就是回归测试的任务。传统上，回归测试通过检测软件异常行为来保证软件质量。然而，在实践中所有的软件行为不可能全部被检测到，尤其针对于大型的复杂软件系统。为帮助开发人员更好的进行回归测试，传统工作集中在增加测试用例上，即通过人工或自动生成测试输入的方法，观测测试输出以捕获程序行为。这种方法虽一定程度上有效，但也存在很大缺陷：人工编写测试用例费时费力，不能覆盖的代码较多，而且容易受人主观判断的误导（例如忽略某些特别容易存在缺陷的类、方法等）；自动生成测试用例技术存在很多问题，例如代价大和无法很好的处理数组、字符串等。基于目前软件测试技术存在的缺陷，很多软件不得不在面临诸多安全威胁的状态下发布，有时甚至造成重大财产损失甚至人员伤亡。因此，我们迫切需要新技术来辅助现有技术以更好的进行回归测试、保障软件质量。

基于此，我们提出了“同构回归测试”技术（Isomorphic Regression Testing）,可以在不增加测试用例的情况下帮助开发人员发现更多缺陷。如第一幅图所示，在软件新版本中，假设开发人员引入了一个新的 bug，但现有测试用例并不能找到这个 bug，因为相应的代码都没有被覆盖到。同构回归测试修改了该版本中的一部分代码，生成了程序变体（program variant）。在执行程序变体时，一部分测试用例会改变原执行路径，从而在不增加测试用例的情况下轻而易举的让现有测试用例覆盖原本未覆盖的代码。



修改代码的规则可以有很多种，最简单的修改代码的规则其实就是将条件分支节点逆转，0 变为 1，1 变为 0，这样一来只要原来有一条分支被覆盖，那么逆转条件后另外一条分支就肯定能被覆盖。这样通过修改代码来增加覆盖率的方式比通过增加测试用例来增加覆盖率简单多了。

然而，但是，怎么判断程序中有 bug 呢？代码被修改之后，就不是原来的程序了呀，原来的测试预言根本不能再继续被使用。然而，在回归测试中，完全可以利用上一个版本的同样的程序变体的行为来充当测试预言。当我们对两个版本做完全相同的改变时，他们改变后的行为也应该是相同的，除非新版本中存在 change 或者 bug。

同构回归测试技术可以被分解为三个基本步骤。首先，为保证能够在两个版本上做相同的代码改变，该技术可以使用控制流图匹配的方法，找到完全相同的代码片段。然后，根据现有测试用例的分支覆盖情况，该技术生成、执行程序变体并记录执行结

果，再将两个版本的程序变体的执行结果与原程序执行结果进行对比。如果两个版本的原有程序上执行结果相同但是程序变体的执行结果不同，就可以说明新的潜在的 bug 存在了，而且该 bug 是通过同构回归测试技术里新执行的分支暴露出来的。

为验证同构回归测试技术的有效性，我们随机从 GitHub 的前 1000 个 most popular Java projects 上面找了 10 个 Java 项目，并使用变异测试工具 Pit 植入错误来对检错效果进行验证。结果表明，同构回归测试技术能够多覆盖 5.3%～80%的分支，并能够检测出 5.3%～70%的原来测试用例未检测出来的错误。

projects	B_all	B_uncover	B_ISON	B_prop
cors-filter	146	8	4	50.0%
digester	1,432	547	318	58.1%
evo-inflector	26	7	1	14.3%
gelfj	216	109	55	50.5%
gson-fire	194	80	20	25.0%
hashids	74	29	17	60.7%
jackson	154	131	7	5.3%
java-jwt	104	11	4	36.4%
jopt-simple	378	10	8	80.0%
scrypt	116	68	43	63.2%

说明：以上表格中，第二列为每个项目原有的分支总数，第三列为现有测试用例未覆盖的分支数，第四列和第五列为在现有测试用例不能覆盖的分支当中，同构回归测试技术多覆盖的分支数以及所占比例。

projects	M_all	M_unkill	M_ISON	B_prop
cors-filter	195	73	15	20.5%
digester	200	116	22	19.0%
evo-inflector	105	51	8	15.7%
gelfj	200	149	9	6.0%
gson-fire	200	108	12	11.1%
hashids	200	65	25	38.5%
jackson	200	152	8	5.3%
java-jwt	158	69	33	47.8%
jopt-simple	200	56	3	5.4%
scrypt	200	92	64	69.6%

说明：以上表格中，第二列为每个项目原有的错误总数，第三列为现有测试用例未检测出的错误数，第四列和第五列为在现有测试用例不能检测出的错误当中，同构回归测试技术额外检测出的错误数以及所占比例。

同构回归测试技术在以下四个方面具有创新性。首先，程序变体也可以被看作一种变异体（mutant），因此除了模拟程序中的错误这个初始应用以外，该技术开拓了 mutant 的应用领域，首次利用探索 mutant 的行为来间接探索原有软件异常行为；第二，该测试技术首次在不增加测试用例的情况下发现更多的缺陷；第三，该技术利用了一种新型的蜕变测试技术（Metamorphic Testing），利用相同测试输入在不同程序

之间的测试输出之间的关系来检测缺陷；最后，该技术可以被看作一种低代价的符号执行技术，虽然目的也是执行更多的路径，但是不需要解约束，因而具有更广的适用性。

作者简介：

本文作者包括北京大学博士研究生张洁、博士研究生娄一翎，美国德克萨斯大学达拉斯分校张令明教授，北京大学郝丹教授、张路教授、梅宏教授。

说明：*为必填项。