基于形式化方法的测试用例自动生成技术

2022 年 5 月 16 日

目录

1 简介

在软件开发的生命周期中，软件测试是确保软件高质量的一个重要手段。但是软件测试只能说明软件在当前测试环境下的软件行为是否符合用户的预期，却无法保证软件的正确性和安全性，软件的安全隐患很大程度上是由编程语言自身的设计缺陷引起的，而这也是软件测试无法顾及的方面。因此，形式化方法应用而生。形式化方法是指通过严格的数学推导即逻辑推理，对程序应该持有的性质进行正确性的证明。学术界针对不同的应用场景进行程序逻辑的创新，这其中就包括霍尔逻辑、分离逻辑、并发逻辑等等；工业界也使用形式化的方法辅助编程语言的设计和开发，例如 Rust 编程语言。上述的工作都希望形式化方法尤其是验证方法学能够广泛应用于软件开发过程中，而不仅仅只局限在学术和安全领域相关的研究上。但是，由于形式化方法自身耗费的人力物力财力巨大，较低的投入产出效益难以大面积推广。所以，本项目试图介绍一种将形式化方法引入到程序开发流程中，以辅助程序员进行软件测试，提升软件测试的效率和可信程度。

2 技术背景

2.1 软件测试

通常来说，软件测试的流程为分析测试需求——制定测试计划——设计测试用例——执行测试——编写测试报告。测试需求的分析源主要包含两部分：需求规格说明和软件源码。规格说明了用户所需求软件的主要功能，该说明对于测试用例的设计有重要参考意义，但是无论在何种开发模型下（瀑布、螺旋或者迭代）软件源码对测试计划的制定都起决定性作用，因为软件测试更多地依赖后者的详细设计，所以说在软件测试过程中的用例设计只是针对程序本身，而一定程度上忽略了用户的需求。所以，一方面程序员在测试用例设计和执行过程中很大程度上是目的清晰而又盲目，有目的指程序员的确是对程序本身的功能进行测试，盲目是指程序员的测试可能偏离了用户的需求。另一方面，程序员是通过测试覆盖率来评定测试是否完全，测试覆盖率是依据测试用例所能涵盖的控制流，这种方法也只能说明被测控制流功能上的正确性，无法证明更多的事情。结合上面两点，总结出软件测试的缺点和亟待完善的问题：

1. 软件需求与测试设计分割，导致的测试的盲目性。

2. 测试技术自身的局限性，只说明软件某些路径下功能的正确，无法证明软件的正确。

2.2 形式化方法

通俗地讲，形式化方法主要对软件所要求具有的性质进行描述，该种描述是基于特定逻辑所构建的形式化的公式语言。针对性质描述的处理办法主要有两种：手动证明和交给自动定理证明器。手动证明要求程序员有较高的学术素养，需要其熟悉程序逻辑的演算规则和高阶公式的处理办法，并不能用以大范围推广和应用于平常的软件开发中。性质描述交给自动定理证明器相对来说简单许多，只需要程序员编写相应的类编程语言的性质规范即可。性质规范是对程序性质的描述，针对这种描述进行基于逻辑的数学方法推导可以规避过于追求覆盖率而导致测试任务繁复的问题。更重要的是，形式化方法允许程序员从需求侧定义软件的性质描述，并且随着软件的开发进度，前一阶段的性质描述可以在下一阶段复用并进一步详细化。所以，形式化方法能够用以弥补软件测试过程中存在的诸多问题。

3 技术方案