

符号测试

介绍

本报告旨在通过阅读《A Survey of Symbolic Execution Techniques》、《Symbolic Execution and Program Testing》和《Symbolic Execution for Software Testing-Three Decades Later》等相关资料，了解符号测试的基本概念、主要技术和重要挑战。符号测试是一种基于符号执行的软件测试方法，通过在程序的符号状态下进行路径探索和约束求解，发现潜在的错误和异常情况。

基本概念

符号测试是一种自动化测试方法，通过替换程序中的输入变量为符号值，以符号的方式执行程序，生成路径约束条件，并使用约束求解器求解这些约束条件，以探索程序的不同路径和执行状态。符号测试的基本概念包括符号执行、路径探索和约束求解等。

主要技术

1. 符号执行引擎：符号执行引擎是符号测试的核心组件，能够对程序进行符号执行，生成路径约束条件并探索不同的路径。常见的符号执行引擎有 KLEE、SAGE 和 EXE 等。
2. 路径探索策略：路径探索策略决定了符号执行引擎如何选择路径进行探索，包括深度优先搜索、广度优先搜索和随机搜索等。
3. 约束求解器：约束求解器用于求解生成的路径约束条件，找到满足条件的具体输入值，以探索不同的执行路径。常用的约束求解器有 Z3、STP 和 SMT-LIB 等。
4. 符号化数据生成：符号化数据生成技术用于生成符号输入数据，覆盖不同的程序路径和执行状态，以发现潜在的错误和异常情况。

重要挑战

1. 路径爆炸：由于程序的复杂性和路径组合的数量庞大，符号执行往往面临路径爆炸问题，导致执行时间和内存消耗过大。
2. 符号执行路径不完整：符号执行无法处理涉及输入数据的循环、递归和动态内存分配等问题，导致路径覆盖不完整。
3. 约束求解效率：约束求解是符号测试的关键步骤，但约束求解器的效率和可扩展性对于处理复杂程序和大规模测试用例集合至关重要。
4. 精确性和误报：符号测试可能产生误报，即错误地报告正常路径为错误路径，或者漏报一些真实的错误。

结论

符号测试作为一种基于符号执行的软件测试方法，在发现软件缺陷和错误方面具有一定的优势。然而，路径爆炸、符号执行路径不完整、约束求解效率和误报等挑战限制了符号测试的实际应用。进一步的研究和创新将有助于克服这些挑战，并提高符号测试的效果和可行性。