符号执行技术的现状与展望

符号执行是一种程序分析技术,它使用符号变量代替实际的数据,通过运算来推导这些符号变量的关系和约束条件。它可以自动探索程序的路径并推导约束。符号执行技术最早可以追溯到1976年,近20年来随着约束求解器和SAT/SMT求解器的发展,符号执行技术得到长足进步。现在已成为程序验证、测试生成、程序分析等领域非常重要的一种技术手段。目前符号执行技术主要应用于自动软件测试生成、程序验证、程序分析等。如用于测试生成,可以自动生成程序的测试用例;用于程序验证,可以形式化验证程序的正确性;用于程序分析,可以自动分析程序可能达到的状态空间。现在有很多开源的符号执行工具,如KLEE、angr、SymEx、jSYM等,还有商业工具如IBM Security AppScan等。这些工具和平台各有不同的功能和应用重点。随着约束求解器和SAT/SMT求解器的不断进步,符号执行技术的应用范围还会进一步拓展,如可用于系统级软件的验证与分析;与机器学习相结合,可以实现“符号执行的选择性探索”;也许最终可以在一定程度上实现“全自动的程序分析”。

首先是符号执行的定义和概念符号执行是一种程序分析技术,它使用符号变量代替实际的数据,通过运算来推导这些符号变量的关系和约束条件。它可以自动探索程序的路径并推导约束。符号执行的基本思想是:给定一个程序P和一个输入空间I,对于每个输入i∈I,都有一个对应的输出o∈O和一个执行路径p∈P.符号执行就是将输入i抽象为一个或多个符号变量x,然后沿着每个可能的路径p执行P(x),并记录下路径上遇到的条件语句和赋值语句所形成的约束条件c(x).最后,利用约束求解器或SAT/SMT求解器来求解c(x)是否可满足,如果可满足,则给出一个具体的输入值i'使得c(i')为真;如果不可满足,则说明该路径不可达。

符号执行可以看作是一种静态分析技术,因为它不需要实际运行程序;也可以看作是一种动态分析技术,因为它需要模拟执行程序。符号执行与传统的静态分析技术相比,具有更高的精度和效率;与传统的动态分析技术相比,具有更广的覆盖范围和更强的自动化能力。符号执行的概念最早可追溯到1976年,King提出了一种基于符号变量和约束求解器的软件测试方法。此后,随着计算机科学和工程领域的发展,符号执行技术也不断发展和完善。在1980年代至1990年代初期,符号执行技术主要面临两个挑战:一是如何处理复杂的数据结构和指针操作;二是如何处理大规模和实际的程序。为了解决这些问题,出现了一些改进的符号执行技术,如基于抽象解释的符号执行、基于混合执行的符号执行、基于动态符号执行的符号执行等。这些技术通过引入抽象层次、混合具体和符号值、动态生成符号值等方式,来提高符号执行的可扩展性和实用性。在1990年代中期至2000年代初期,符号执行技术的发展受到了约束求解器和SAT/SMT求解器的影响。约束求解器是一种可以求解特定领域的约束问题的软件工具,如线性规划、整数规划、布尔代数等。SAT/SMT求解器是一种可以求解命题逻辑或一阶逻辑公式是否可满足的软件工具。这些工具的出现和发展,极大地提高了符号执行技术处理复杂约束条件的能力和效率。在2000年代中期至今,符号执行技术已经成为程序验证、测试生成、程序分析等领域很重要的一种技术手段。它已经被应用于各种类型和规模的程序,如嵌入式系统、操作系统、网络协议、数据库系统、安全软件等。它也已经被集成到了各种工具和平台中,如KLEE、angr、SymEx、jSYM等。它还与其他技术相结合,如模型检测、静态分析、动态分析、机器学习等,形成了新的研究方向和应用场景。

目前符号执行技术主要应用于自动软件测试生成、程序验证、程序分析等。下面分别介绍这些应用领域。自动软件测试生成是指利用符号执行技术自动生成程序的测试用例,以覆盖程序的尽可能多的路径或状态。这样可以提高软件测试的效率和质量,并发现潜在的错误或漏洞。自动软件测试生成的基本步骤是:首先,将程序输入抽象为一个或多个符号变量;然后,沿着每个可能的路径执行程序,并记录下路径上遇到的条件语句和赋值语句所形成的约束条件;最后,利用约束求解器或SAT/SMT求解器来求解每个路径上的约束条件,如果可满足,则给出一个具体的输入值作为测试用例;如果不可满足,则说明该路径不可达。自动软件测试生成有很多优点:一是可以自动化地生成测试用例,无需人工干预;二是可以覆盖程序的多种情况和边界条件,无需人工预测;三是可以发现程序中难以触发或难以检测的错误或漏洞;四是可以与其他测试技术相结合,如随机测试、模糊测试、回归测试等。自动软件测试生成也有一些挑战:一是如何处理复杂的数据结构和指针操作;二是如何处理大规模和实际的程序;三是如何处理不确定性和非确定性的因素,如环境变量、用户输入、时间、随机数等;四是如何处理程序中的库函数、系统调用、外部接口等;五是如何评估测试用例的质量和效果。为了解决这些挑战,出现了一些改进的自动软件测试生成技术,如基于混合执行的测试生成、基于动态符号执行的测试生成、基于搜索的测试生成等。这些技术通过引入混合具体和符号值、动态生成符号值、搜索算法等方式,来提高测试用例的生成速度和覆盖率。接着是程序验证，程序验证是指利用符号执行技术形式化验证程序的正确性,即证明程序是否满足给定的规范或属性。这样可以提高软件的可靠性和安全性,并避免潜在的错误或漏洞。程序验证的基本步骤是:首先,将程序输入抽象为一个或多个符号变量;然后,沿着每个可能的路径执行程序,并记录下路径上遇到的条件语句和赋值语句所形成的约束条件;最后,利用约束求解器或SAT/SMT求解器来求解每个路径上的约束条件是否满足给定的规范或属性,如果都满足,则说明程序是正确的;如果有不满足的,则说明程序是错误的,并给出反例。程序验证有很多优点:一是可以形式化地证明程序的正确性,无需人工检查;二是可以验证程序的多种规范或属性,如功能正确性、安全性、终止性、死锁性等;三是可以验证各种类型和规模的程序,如嵌入式系统、操作系统、网络协议、数据库系统、安全软件等;四是可以与其他验证技术相结合,如模型检测、抽象解释、类型系统等。符号执行是一种程序分析技术,它使用符号变量代替实际的数据,通过运算来推导这些符号变量的关系和约束条件。它可以自动探索程序的路径并推导约束。符号执行技术最早可追溯到1976年,近20年来随着约束求解器和SAT/SMT求解器的发展,符号执行技术得到长足进步。现在已成为程序验证、测试生成、程序分析等领域很重要的一种技术手段。目前符号执行技术主要应用于自动软件测试生成、程序验证、程序分析等。如用于测试生成,可以自动生成程序的测试用例;用于程序验证,可以形式化验证程序的正确性;用于程序分析,可以自动分析程序可能达到的状态空间。现在有很多开源的符号执行工具,如KLEE、angr、SymEx、jSYM等,还有商业工具如IBM Security AppScan等。这些工具和平台的功能和应用领域各有侧重。随着约束求解器和SAT/SMT求解器的不断进步,符号执行技术的应用范围还会进一步扩大,如可用于系统级软件的验证与分析;与机器学习相结合,可以实现“符号执行的选择性探索”;也许最终可以在一定程度上实现“全自动的程序分析”。