《Practical Memory Leak Detection using Guarded Value-Flow Analysis》Review

论文信息

Practical Memory Leak Detection using Guarded Value-Flow Analysis Sigmund Cherem, Lonnie Princehouse, Radu Rugina

工作概述

这篇论文提出了一个检测内存泄漏和空间重复释放的方法，并开发出对应的工具。论文以内存释放这一问题为例，讨论了如何处理资源申请和释放的匹配检测问题，类似的问题还包括文件的打开和关闭、锁的获取和释放等等。 本文工作的主要核心点为根据源程序构建出VFG，并考虑每条语句边上的约束，计算出从资源申请点到释放点整条路径上的约束公式，最后通过约束求解器得到检测的结果。工具的结构图如下所示。其中核心部分为与Guard条件相关的三个部分：Unguarded Graph Reachability、Guard Analysis和Guarded Graph Reachability，通过讨论不同的语句类型给出了guard条件的不同结果。 本文以内存泄漏和双重释放两类错误为例，对其框架做了系统的测试。总的来说，本文的工作核心点在于构建VFG并计算每条边上的条件公式，通过调用SAT求解器检测目标缺陷。为了提高效率和工具的实用性，文中提出了一系列的优化策略，比如在计算约束公式时实时进行化简，防止出现公式指数爆炸等问题。这些都是本文工作的亮点。

不足与局限

1. 文中提出的约束条件都是命题逻辑层面的，可以通过SAT约束求解器求解。但是在很多情况下，程序中的guard条件很复杂，超过了命题逻辑的表达范围，可能需要高阶求解器求解，比如SMT，求解起来具有一定的时间复杂度，这样会使得整个系统检查的时间代价骤增，影响最终工具集成到开发环境中。 一个可能的解决方案是采用SMT求解替代SAT求解，根据实际程序的规模选择适当的SMT约束求解器：如果源码规模不大，可以尝试采用微软开发的Z3 SMT Solver；如果规模较大或旨在降低时间开销，可以尝试针对程序语言开发轻量级的SMT Solver，这样可以在既满足实际需求的情况下减少时间开销。
2. 文中对指针运算的讨论不够充分，更有效地做法是构建不同指针变量之间的关系图，建立起指针变量指向的内存空间之间的关系。这样能支持更多的指针操作。比如p和q为指向heap的两个指针，q相对于p的偏移量为m，则构造一个有向图存储这样相对的偏移关系。当存在指针运算时，通过值分析得到相对偏移量，在上面构造的有向图上查询得到准确的指针变量。比如： 构造出的指针关系图为: x ---(+4)---- > y 在最终free时，对参数做值分析，在指针关系图中找到释放的空间为x。这是论文中提到的简单的例子，对于复杂的空间申请和指针运算，可以采用类似的方法。通过引入指针关系图记录偏置关系能够很好地处理与指针运算相关的空间释放问题。
3. 文中对循环的近似处理可能会导致结果的精确度低。只假定循环一次这样的操作过于粗糙。为了避免多次循环导致guard条件无限增长，可以考虑在分析前计算每个循环中各个location处的循环不变量。当进行实际分析时，结合各个location处的不变量信息计算guard条件。常用循环不变量工具有apron工具，提取出仿射型的循环不变式；其他类型的循环不变量还有k-induction invariant，适合提取固定增量形式的不变式。
4. 工具的某些模块可能存在效率问题，除了上面提到的约束求解模块，其他的相关模块比如Def-Use关系分析可以尝试采用并行的思想进行子模块设计。比如根据函数的调用关系，将源码中的函数簇拆分成多个连通分支，每个连通分支交付给一个线程进行处理。这样的改进能够加快Def-Use关系的生成。