链滴

程序表示方法

作者：Hanseltu

原文链接：https://ld246.com/article/1553265984631

来源网站：链滴

原文链接 程序表示方法

常用的5种程序表示方法

在我们对程序进行推理分析之前，我们必须要有分析的基础——程序表示模型。

比较复杂的模型有：

- 已编译的二进制代码
  - 难以将数据和代码分开
  - 通常应用于逆向工程或者安全分析任务
- 源代码
  - 代码量非常大，不同语言差别很大
  - 代码之间的关系难以提取
  - 通常和注释结合分析

一个好的程序表示模型应该可以明确地分析出代码之间的关系，主要相关程序图表示模型有以下五种下面将分别介绍：

抽象语法树 （Abstract Syntax Trees，AST）

定义：抽象语法树可以将程序源码转化为规范的树形结构。其中的叶子节点代表变量，数值和操作数中间节点代表操作符，语句等。如下图对源码的转换：

应用：使用抽象语法树可以进行语法分析和转化，主要的应用场景有：

- Simple bug pattern(简单bug 模式？？？)
- Style checking(类型检查)
- Refatoring(重构)
- Training prediction/completion models(训练预测和编译模型)

缺陷：即使是一样的程序，生成的抽象语法树可能不同。

控制流图（Control Flow Graphs，CFGs）

定义：程序控制流图可以表示程序执行时可能经过的路径。其中方框代表代码块，箭头表示可能的执路径。如下图：

缺陷：程序语言的特定功能通常被抽象化，不利于整个程序的分析

程序依赖图（Program Dependence Graphs，PDG）

定义：程序依赖图描述了代码之间是如何相互影响的。Y\rightarrowX表示Y影响了X，或者X受到了Y影响。

分类：两种主要的依赖关系

- 数据依赖
- 控制依赖（通过决策影响）

数据依赖：

控制依赖：

每个经过Y的节点都要经过X节点,X的作用类似于“看门狗”

应用：

- 调试：什么地方有可能产生bug?
- 安全：有没有敏感信息泄露?
- 测试：如何做到全覆盖测试?

程序调用图（Call Graphs）

定义：程序调用图描述了程序的组成。其中节点代表函数，箭头代表该函数可能调用的路径。如下图：

主要问题：如何处理函数指针？如何描述函数回调?

指向图（Points-to Graphs）

定义：通过变量的指向来分析程序。

主要存在的两个问题：

程序运行时表示方法

Trace技术

以上提到的5种程序表示方法都是基于静态的表示，即不执行代码分析出来的结果。若对程序进行动表示则需要动态分析技术。常用的技术有trace技术。

(上图左下角第二行和第三行不是很理解？？？)

动态分析缺陷：整个代码分析起来非常复杂，全部分析也不是很有必要。

总之，有了以上静态或者动态的模型，可以将程序进行转化并对将其应用在真实的代码中。

下小节将对程序中某些感兴趣的部分进行分析，即程序切片技术，而不是分析整个庞大的程序。

>> 回到课程主目录