软件安全入门

第五课 符号执行

目录
CONTENTS

01 符号执行基本模型

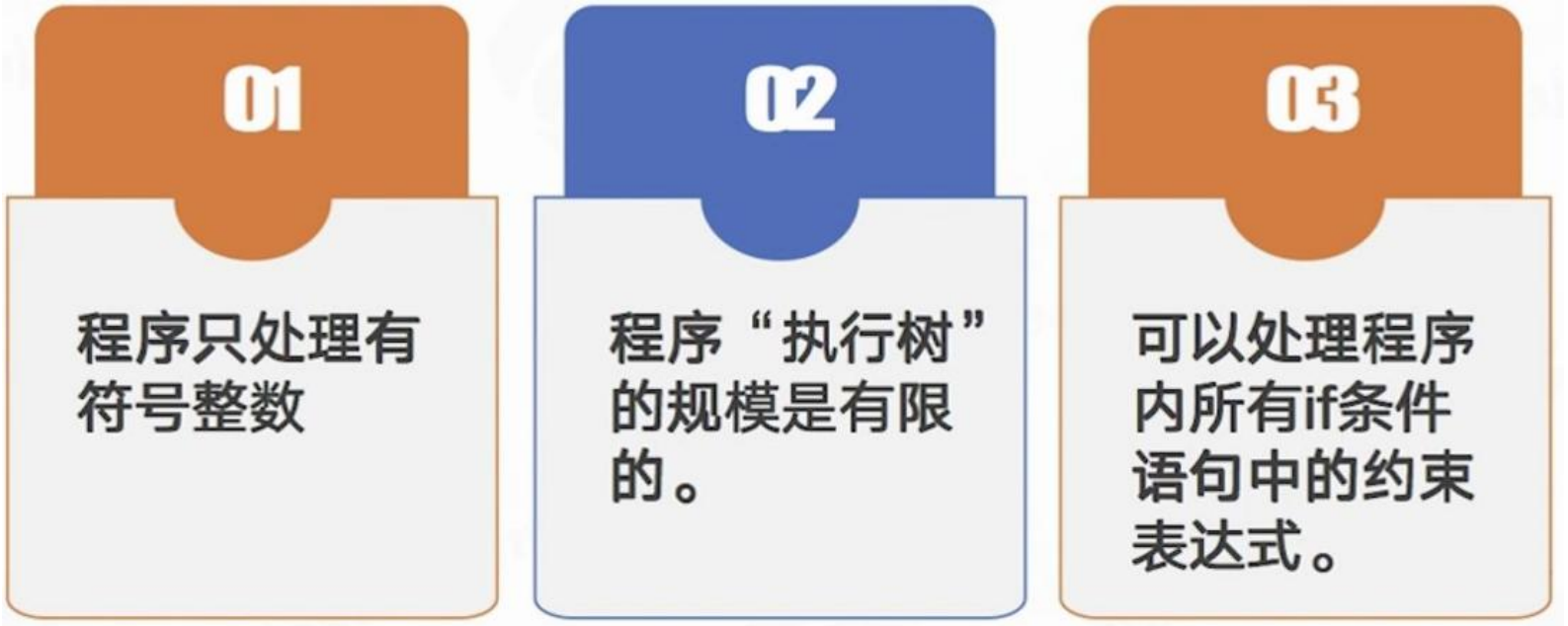
02 动态符号执行技术

03 并行符号执行技术

04 选择符号执行技术

主要内容包括符号执行的基本思想、符号执行树、约束求解等。

符号执行的基本思想是：通过使用符号变量代替具体值作为程序或函数的参数，并用符号表达式来表示与符号值相关的程序变量的值。符号执行的理想使用场景：



执行树是用来描述程序执行路径的树形结构。执行树中的一个节点对应程序中的一条语句，程序语句之间的执行顺序或跳转关系对应执行树中节点间的边。

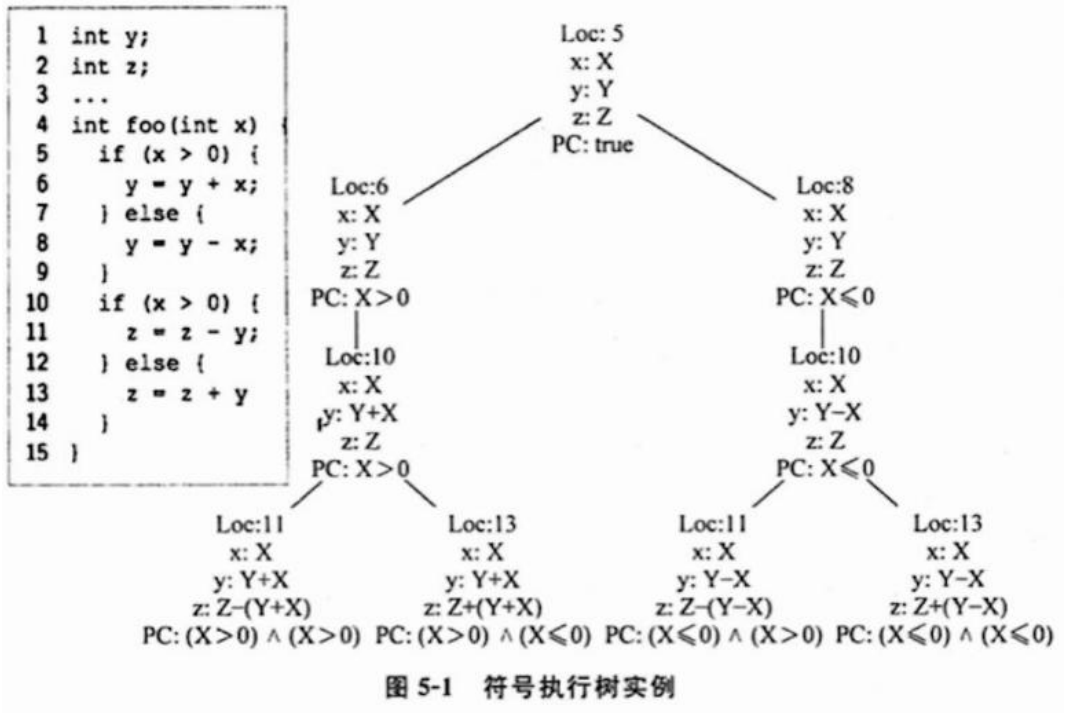


图 5-1 符号执行树实例

在遇到程序分支指令时，程序的执行也相应地搜索每个分支，分支条件被加入到符号执行保存的程序状态π中，π表示当前路径的约束条件。在收集了路径约束条件之后，使用约束求解器来验证约束的可解性，以确定该路径是否可达。

```
1. foo(int x,int y,int z){
2.    a=0,b=0,c=0
3.    if(x>0){a=-2;}
4.    if(y<5){
5.       if(y+z>0){b=1;}
6.       c=2;
7.    }
8.    if(a+b+c==3)
9.          //some error
10.   exit()
11.}
```

图 1　符号执行的示例代码

算法 1　符号执行算法

Symbolic execution():

1. Create initial task
2. pc=0,π=∅,σ=∅
3. add task (pc,π,σ) onto worklist
4. while(worklist is not empty):
5. pull some task (pc,π,σ) from worklist
6. if it potentially forks at(pc,π,σ)
7. Execute
8. if $\pi_0 \cap p$ feasible:
9. add task (pc_1,($\pi_0 \cap p$),σ_0)
10. if $\pi_0 \cap \neg p$ feasible:
11. add task (pc_1,($\pi_0 \cap \neg p$),σ_0)
12. . end while

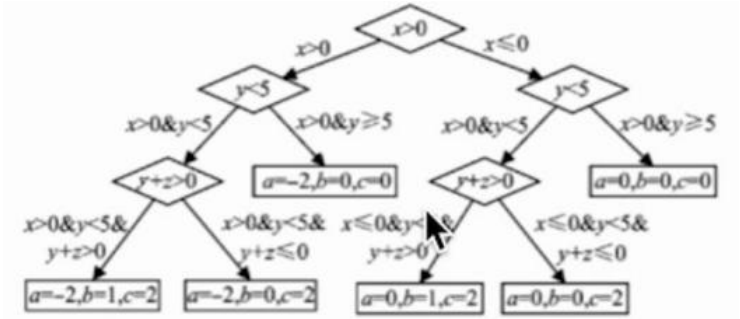
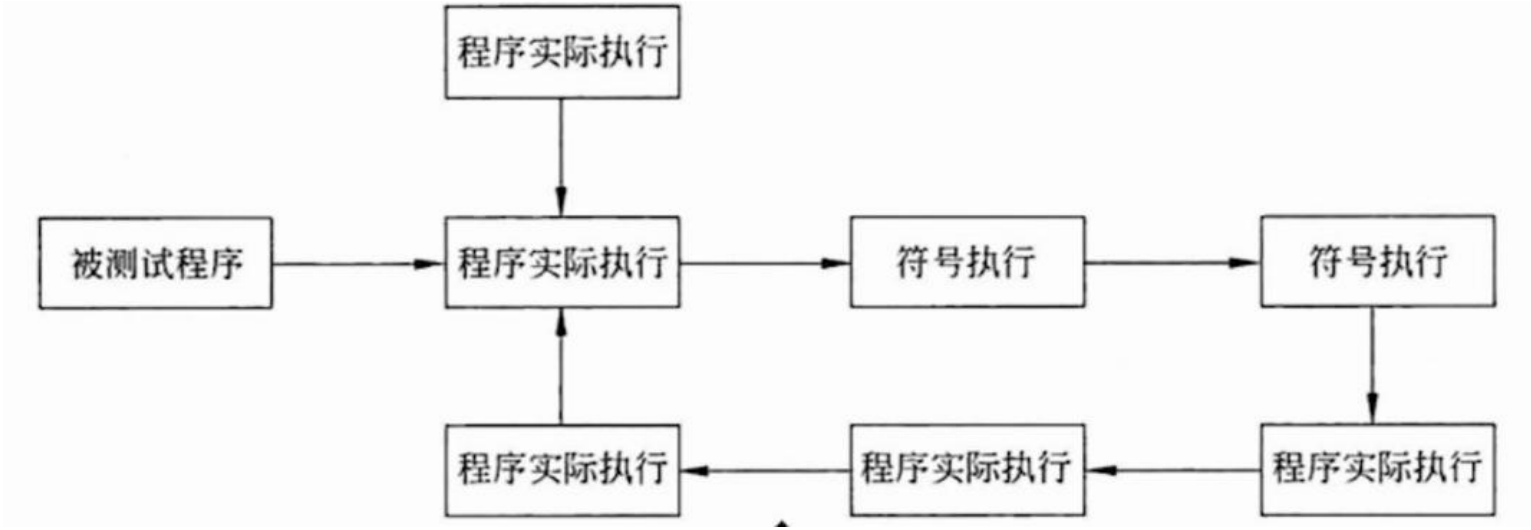


图 2　示例代码的程序执行树

主要内容包括动态符号执行技术的基本原理、基本流程、关键技术等。

动态符号执行的基本思想是：以具体的数值作为输入执行程序代码，在程序实际执行路径的基础上，用符号执行技术对路径进行分析，提取路径的约束表达式，根据路径搜索策略(深度、广度)对约束表达式进行变形，求解变形后的表达式并生成新的测试用例，不断迭代上面的过程，直到完全遍历程序的所有执行路径。



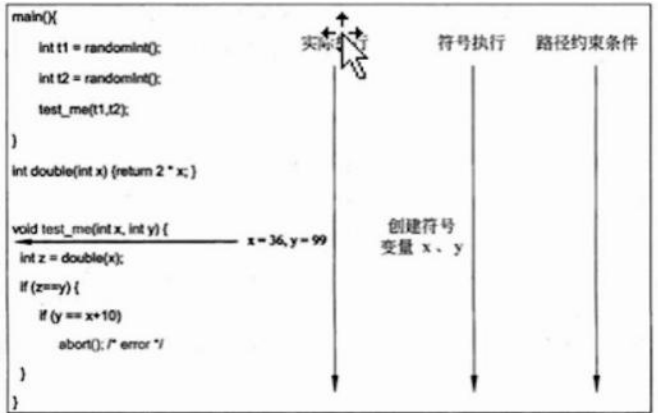
2.2 动态符号执行实例

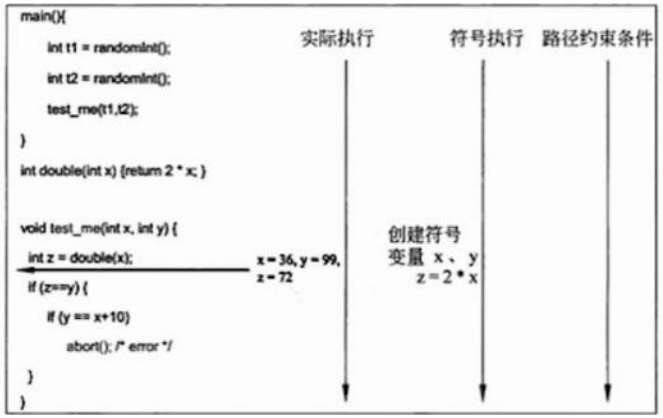
(1)动态执行程序提取搜索约束pCo。

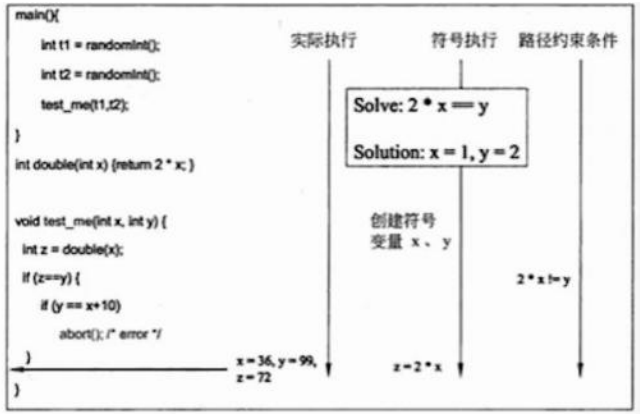
(2)使用路径搜索算法，按搜索策略对路径中的约束条件取反，生成新的路径约束条件pCb。

(3)使用约束求解器对pCp进行求解并生成新的测试用例testb。

(4)使用testo对程序进行测试，引导程序执行其他分支，进入新的子树区域搜索。







SAGE是一款基于二进制文件的符号执行引擎，SAGE技术的核心是它的路径搜索算法分代搜索，通过重复4个任务(测试、执行路径记录、覆盖率收集、符号执行)来实现分代搜索算法。

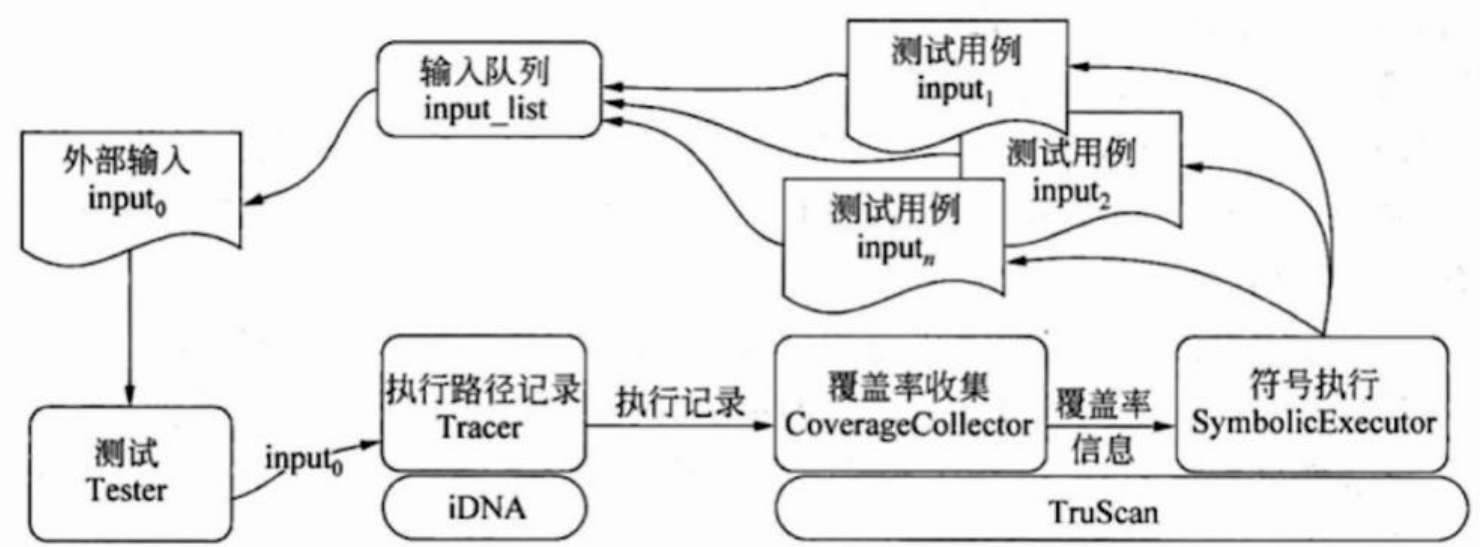


图 5-17　SAGE 的系统架构

主要内容包括并行符号执行技术的基本思想、和并行系统介绍。

3.1 基本思想

传统符号执行面临3个主要的问题：路径爆炸、路径约束求解过载、内存使用过载。使得无法有效测试规模庞大或逻辑复杂的程序。并行符号执行方案的提出就是希望通过计算集群可无穷扩展的内存空间和CPU资源来缓解符号执行中的路径爆炸问题。在并行技术与符号执行技术融合的过程中，需要解决两个关键问题：

分布式环境下的路径搜索策略

并行系统需要解决的首要问题是如何有效避免各节点对程序执行路径的重复搜索。

对于并行系统来说执行树未知的影响很大，因为同一时刻各工作节点都在对路径进行探索，各节点很难实时共享已搜索路径信息，造成不同节点对同一路径的冗余探索，影响系统效率。

分布式环境下的负载均衡策略

并行系统需要解决的核心问题是如何划分任务集合、使得各工作节点避免出现负载不均衡的情况。

并行系统能够将程序执行树动态划分成多个互不重合的子树，但很难保证子树的规模均衡，因为程序执行树本身就不是平衡二叉树。

SCORE是基于分代搜索算法的并行执行系统，其包含两种类型的节点：服务端和客户端。系统中有7种类型的协议包在服务器和客户端节点间传输来实现节点的协同工作。

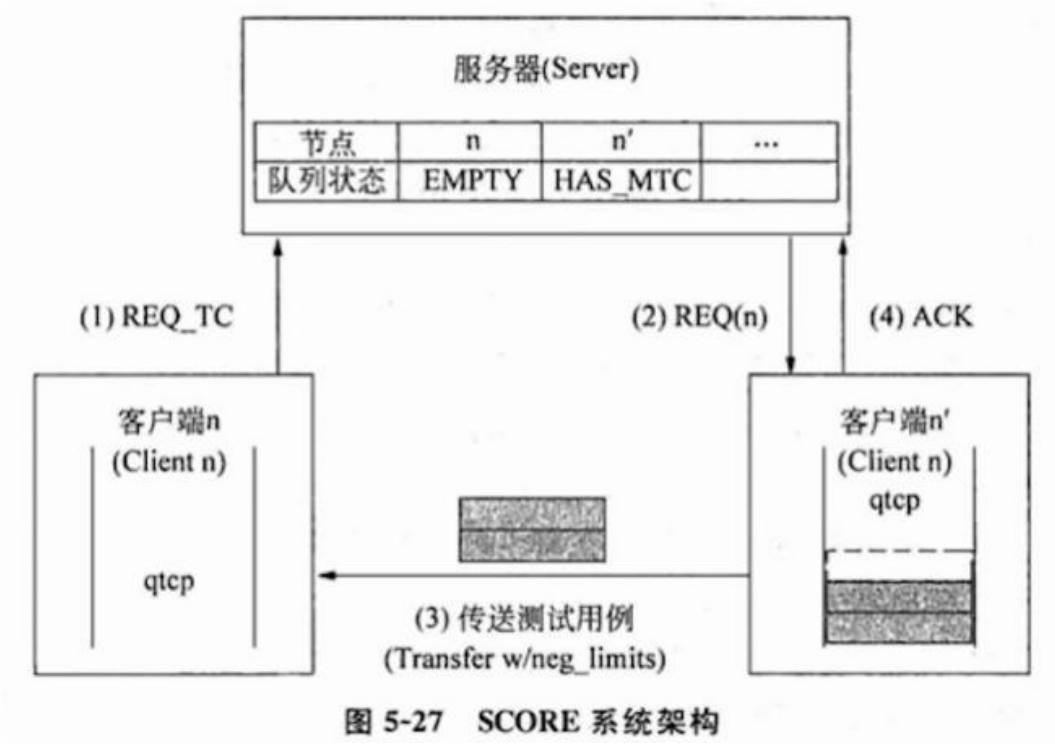


图 5-27　SCORE 系统架构

Cloud9是第一个并行符号执行系统，其包含负载均衡节点(load balancer)和工作节点(worker)两种类型节点。每个worker节点独立的探索程序执行树中的一个子树。每个worker节点上都部署了一个基于KIEE实现的符号执行引擎，包括符号执行环境、约束求解模块、路径搜索模块3部分。

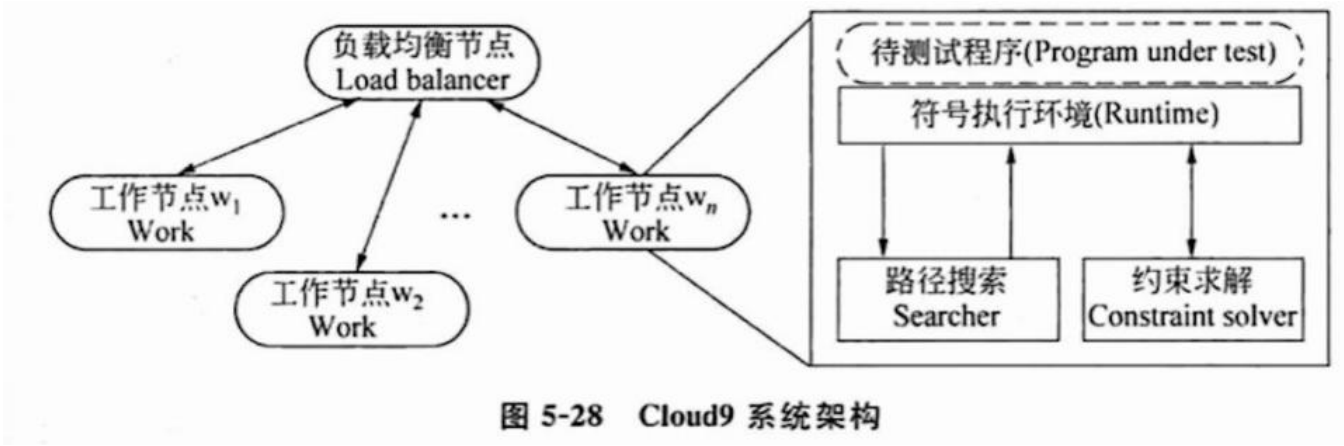


图 5-28　Cloud9 系统架构

04 选择符号执行技术

主要内容包括选择符号执行技术的基本思想、选择符号执行实例等。

选择符号执行技术的基本思想是：当测试程序执行到测试人员感兴趣的程序段时，对代码进行符号执行分析，当程序执行到不感兴趣的外部调用函数段时，对代码进行单路径具体执行。基于此思想实现了工具S2E。

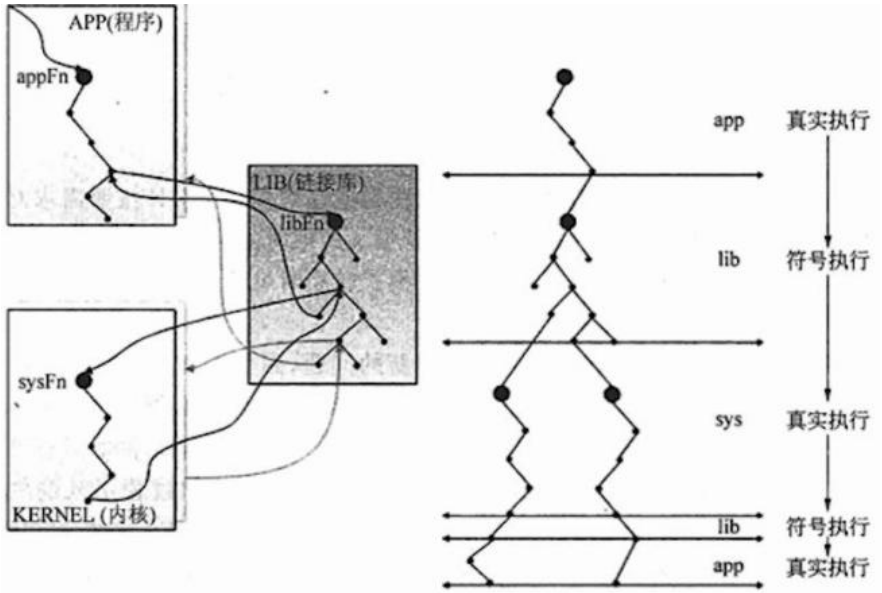


图 5-36　选择符号执行实例

4.2 主流的符号执行工具

源代码分析工具

1. EXE & KLEE

EXE使用的是动态符号执行中的执行生成测试，优势是可以强制执行程序的任何可达路径。KLEE基于EXE改进，运行在LLVM上，能生成高覆盖率的测试集。

2.DART, CUTE和 jCUTE

DART的优势是可以对任何编译型程序进行完全自动化的测试。CUTE是基于DART开发的，引入了对多线程的支持， jCUTE是针对java语言的。

二进制分析工具

1.angr

angr是用python开发的高度模块化的开源二进制分析框架，功能包括二进制的符号执行、智能状态合并、各类程序流图恢复等，主要创新点是支持跨平台多架构，继承了程序切片和状态拟合等方法。

2.SAGE

SAGE结合了模糊测试和动态符号执行。SAGE基于DART开发，拓展了Fuzz功能。

课程小结

·1.符号执行基本模型

·主要内容包括：符号执行的基本思想、符号执行树、约束求解等。

·2.动态符号执行技术

·主要内容暴露：动态符号执行技术的基本原理、基本流程、关键技术等。

·3.并行符号执行技术

·主要内容包括：并行符号执行技术的基本思想、和并行系统介绍。

·4.选择符号执行技术

·主要内容包括：选择符号执行技术的基本思想、符号执行实例等。

谢谢观赏