utopia 原理

Attempting to execute command: ['target\_analyzer', '--db', PosixPath('/root/UTopia/exp/snappy/output/build\_db/libsnappy.a.json'), '--extern', PosixPath('/root/UTopia/helper/external'), '--public', PosixPath('/root/UTopia/result/test/snappy/api.json'), '--out', PosixPath('/root/UTopia/result/test/snappy/target/libsnappy.a.json')]

自动生成driver的方法：（保证与源应用程序逻辑相关）

* 推断api的有效序列
* 直接从使用示例中提取有效的api序列
* 【注】单元测试 (UT)是软件开发中的一种测试方法，用于验证软件系统中最小可测试单元的功能是否能按预期进行。这些最小单元通常是软件**中的函数、方法或类等独立模块。**单元测试的目的是对这些单元进行独立测试，以确保它们在给定输入下产生正确的输出。

转换生成手段

* 合成有效的api调用序列
* 合成有效的api 调用参数

1. 有效的api调用序列

主要挑战：确定调用库的哪个api 以及调用顺序，因为api经常具有严格的顺序依赖关系。如FileStorage() → writeRaw() → release()。

如果随机构建api调用序列，在release()之后调用writeRaw()，由于其没有调用构造函数，这样导致的崩溃会被认为是虚假崩溃。

使用方法：从单元测试编写的显式api序列来避免合成合法序列的挑战。

1. 合成有效的api调用参数

合理参数之间的关系

* api之间的关系，函数的输出是另一个函数的输入
* api 内部的关系，两个参数之间的关系， length:参数作为另一个参数的长度，，index：作为另一个参数的索引

提出方式：保留测试单元的原始数据流，（变量的运行状态），使用静态分析找到模糊输入的位置以及变异方式。

跟定义：找到注入模糊输入的合适位置。（查找有抽象语法树的函数，clang ast matchers定位函数）

**根定义分析是一种反向数据流分析，其目的是获得右值**

**常数值的定义，这些常量值不能是从测试代码中其他变量派生出来的，而且可以是衍生出其他变量的常量**

通过向上跟踪所有输入参数，找到可能的定义及常量，进行模糊变异

研究挑战：

UT断言不仅检查临界状态，还检测结果是否与单元测试定义的特定测试值匹配

* 考虑是否越过断言检查

UT结构：

* 以GoogleTest（gtest，一种UT）为例，如下图，它向每个测试类公开了SetUp()、TestBody()和TearDown()接口（分别对应预测试、测试和后测试）。
* **识别测试函数**：UT OPIA利用Clang AST Matchers来识别UT框架中的测试函数（如SetUp(), TestBody(), TearDown()等）。
* **保持独立性**：通过明确调用这些函数，UT OPIA确保每个模糊测试周期内的测试用例都是独立的。

UT OPIA也没有将所有单元测试攒在一起形成一个新的测试驱动程序。相反，它为每个符合条件的单元测试生成一个独立的模糊驱动程序。这些驱动程序可以并行运行，以更高效地探索程序的不同部分。

/UTopia/result/test/snappy/target# ls

libsnappy.a.json

借助该json文件，通过对参数类型的不同定义，找到只作为输入的参数，进一步识别是否可以作为跟定义，去变异成模糊输入入口

* **256**表示BF\_crypt函数中第一个参数的方向设置为256（0x100），表示**In**方向，即用于输入。
* 类似地，**“BF\_crypt(1)”: 272**表明 BF\_crypt 函数中第二个参数的方向为 272（0x110），即同时具有**In**和**Out**方向，即既可用于输入也可用于输出

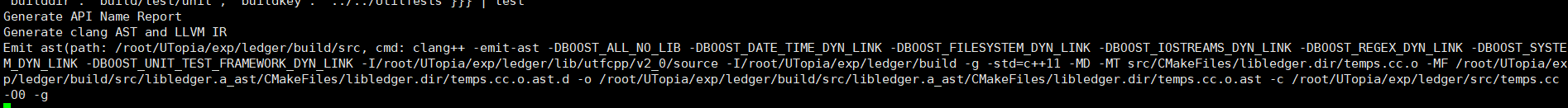
提取初始的种子

* 把提取替换的常数值作为初始种子

google test 原理

* 测试用例通过 TEST() 或 TEST\_F() 宏定义，分别对应独立测试用例和基于测试夹具（fixture）的测试用例。
* 测试夹具允许共享测试前的初始化（SetUp()）和清理（TearDown()）逻辑。

build 过程



问题： 工作流程：

helper.build

限制性一遍build ast bc

分析 cfg ,json

测试成功的项目库：snappy

已验证的项目库: cJSON

复现步骤：下载github ,执行reademe ,复现成功的项目库， 搭建container ,复现上面提到的两个项目库。

添加的配置：helper/make.yml 以及helper/build.yml

# 用法

## 1.以项目snappy为例

1.环境搭建

2.编译utopia

1. .运行

# cJSON

helper/make.yml

helper/build.yml