CAUT 软件用户使用手册

二零一四年十一月二十日

目录

1 引言

CAUT(C Analysis and Unit Testing)是一款基于动态符号执行(dynamic symbolic execution)技术，针对 C 程序自动生成测试用例的工具。CAUT 支持单元测试和全局测试。CAUT 由苏亭(tsuletgo@gmail.com)与 CAUT 小组共同开发维护。

本使用手册，向用户介绍了 CAUT 的运行环境、软件功能、软件安装以及使用方法等。

手册说明

```
$ cd ./CAUT/
```

上例中的美元符($) 代表命令行提示符，是用于提示用户需要在命令行中输入除美元符($) 外的其他内容(cd ./CAUT/)。

英文缩写

CIL – C Intermediate Language – C 中间语言

CPU – Central Processing Unit – 中央处理器

MC/DC – Modified Condition/Decision Coverage – 修订的条件/判定覆盖

参考资料

CAUT 项目主页：http://lab205.org/caut/

CAUT GitHub 项目地址：https://github.com/tingsu/caut-lib

CAUT 的 License:

https://github.com/tingsu/caut-lib/blob/master/LICENSE

MC/DC 的维基百科：http://zh.wikipedia.org/wiki/修改條件判斷覆蓋 （中文）

http://en.wikipedia.org/wiki/Modified_condition/decision_coverage

2 简介

CAUT 是一款基于动态符号执行技术，针对 C 程序自动生成测试用例的工具。CAUT 支持单元测试和全局测试。CAUT 的全名是 C Analysis and Unit Test，即 C 语言分析和单元测试工具。

目前 CAUT 支持覆盖驱动测试满足分支和 MC/DC 标准。同时该工具也可以扩展满足其他的逻辑覆盖标准，例如条件覆盖，条件/判定覆盖，或多种标准综合的标准。目前 CAUT 上实现了多种搜索策略。

CAUT 执行时会先通过 CIL 预处理用户的被测单元，接着与 CAUT 内核编译后执行测试并生成测试报告。

3 运行环境

3.1 硬件要求

CAUT 对内存和 CPU 的要求不高，绝大多数计算机都可以运行 CAUT。

3.2 支持的操作系统

CAUT 目前仅支持 Linux 操作系统，不支持 Windows 平台。为 Linux 提供源代码下载。

4 安装

我们以 Linux 的 Ubuntu 发行版本为例，介绍如何安装 CAUT 软件。

4.1 下载源码

CAUT 在 GitHub 上提供软件下载。可以使用 git 工具:

```
$ git clone https://github.com/tingsu/caut-lib
```

4.2 下载 Ocaml

运行 CAUT 前需要安装 Ocaml 3.11

```
$ sudo apt-get install ocaml
```

4.3 编译 c-ocaml-sqliet-lib 库

在运行 CAUT 前,需要编译 CAUT 源程序内的 c-ocaml-sqlite-lib,用来支持 Ocaml 中的 sqlite 接口。具体步骤如下:

```
$ cd ./tools/c-ocaml-sqlite
$ make clean
$ make
$ sudo make install
```

默认安装目录为: /user/lib/ocaml

4.4 系统中添加相关路径

在运行 CAUT 前需要添加 lpsolve 的路径，同时还要添加 CAUT 的工作目录。

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/work/caut-lib/lpsolve
$ export C_INCLUDE_PATH=$C_INCLUDE_PATH:/work/caut-lib
```

4.5 CAUT 生成文件说明

如果 xx.c 是被测文件，则会产生以下代码：

生成文件名	说明
xx.cil.c	CAUT 前端执行后的 c 文件
xx.orig.cil.c	原始的 c 文件
xx.i	预处理后的 c 文件
xx.c.db	储存 cfg 相关信息的数据库文件

4.6 CAUT 源码和目录说明

目录/文件	子目录/文件	说明
README		说明文件
tutorial		一个简单教程
caut.h		caut 头文件
benchmarks/		一些 GNU Coreutils 中的测试代码
	./caut.sh	调用 cilly.byte.exe
	./reg.sh	caut 库编译执行文件
	./Makefile	makefile
	./caut_br_testing.sh	执行分支测试
lib/		caut 库
cil/		CIL
tools/		一些工具
lpsolve		lpsolve 库

5 常用操作

以下我们介绍命令行运行 CAUT 的方法。

5.1 第一个例子

假设我们有如下一段简单的代码,命名其为 bubble.c,添加在 benchmarks 目录下:

```
#define MAXL 6
void bubble( int v[MAXL], int n ) {
        int i, j, k;
        if(n >= MAXL)
                return;
        for ( i = n; i > 1; --i )
                for ( j = 1; j < i; ++j )
                        if ( v[j] > v[j + 1] ) { /* compare */
                                k = v[j];        /* exchange */
                                v[j] = v[j + 1];
                                v[j + 1] = k;
                        }
}
```

首先在文件头添加包含 caut.h 的语句:

```
#include "caut.h"
```

其次在文件底部，加上 testme 函数，并声明被测单元的输入变量，以及使用 CAUT_INPUT 来申明被测单元的形参和全局变量，例如仍旧以 bubble.c 为例子，则测试 bubble 单元的 testme 函数如下:

```
void testme() {           //add test driver
        int v[MAXL];
        int n;
        CAUT_INPUT(v);    //CAUT_INPUT function parameters, global vars and etc.
        CAUT_INPUT(n);
        bubble(v,n);
}
```

5.2 前端预处理被测文件

5.2.1 前端预处理被测文件

我们通过脚本来预处理被测文件，仍旧以 bubble.c 为例子，则脚本如下：

```
$ source caut.sh ./bubble.c
```

5.2.2 编译 CAUT 内核

使用脚本将预处理后的文件与 CAUT 内核一起编译，以 bubble.c 为例子，则脚本如下：

```
$ source reg.sh ./bubble.c
```

5.2.3 执行测试

接下来，我们使用脚本执行测试

```
$ source caut_br_testing.sh ./bubble.c $max_iter_cnt $path_search_strategy $strategy_param
```

这里，$path_search_strategy 可以是：random_path/klee/crest/pps

当使用"pps"，你可以指定正整数例如 1,2,...，可以用来指定覆盖获得的预测等级

仍旧以 bubble.c 为例子，则脚本如下：

```
$ source caut_br_testing.sh ./bubble.c 100 pps 2
```

具体测试结果会生成在 ./obj-df-log/。

5.2.2 切换 MC/DC 测试

如果需要切换成 MCDC 测试，需要在 Makefile 中的$(CC)部分修改成如下语句。

```
$ $(EXE):$(SRC)
          $(CC) -g -o $@ -I .. -L ../lpsolve $? ../lib/caut-mcdc.a -llpsolve55 -ldl -lpthread
```