KLEE Installation

Symbolic Execution - KLEE

Brief Introduction

KLEE is a symbolic virtual machine built on top of the LLVM compiler infrastructure, and available under the UIUC open source license. It is capable of automatically generating tests that achieve high coverage on a diverse set of complex and environmentally-intensive programs.

Requirements: ubuntu 16.04 and above

1. Install dependencies

KLEE requires all the dependencies of LLVM (see here). In particular, you should install the programs and libraries listed below.

```
$ sudo apt install build-essential curl libcap-dev cmake libncurses5-dev unzip bison
flex libboost-all-dev perl zlib1g-dev
$ sudo apt install graphviz libtcmalloc-minimal4 libgoogle-perftools-dev
```

You should also install `lit` to enable testing, tabulate to support additional features in `klee-stats` and `wllvm` to make it easier to compile programs to LLVM bitcode:

```
$ sudo pip3 install lit wllvm
$ sudo apt-get install python3-tabulate
```

2. Build LLVM 3.4.2

Users can build llvm-3.4.2 from source code or use the pre-built package from:
`https://releases.llvm.org/download.html#3.4.2`

Build from sources:

```
http://releases.llvm.org/3.4.2/llvm-3.4.2.src.tar.gz -> llvm-3.4.2
http://releases.llvm.org/3.4.2/cfe-3.4.2.src.tar.gz -> llvm-3.4.2/tools/clang
$ cd llvm-3.4.2
llvm-3.4.2 $ ./configure --enable-optimized --disable-assertions --enable-targets=host
llvm-3.4.2 $ make -j `nproc`
```

3. Install constraint solver(s)

KLEE supports multiple different constraint solvers. You must install at least one to build KLEE.

1. STP Historically KLEE was built around STP so support for this solver is the most stable. For build instructions, see here.
2. Z3 is another solver supported by KLEE that is reasonably stable. You should use Z3 version 4.4. Z3 is packaged by many distributions. For build instructions, see here..

3. metaSMT supports various solvers, including Boolector, CVC4, STP, Z3 and Yices. We recommend branch v4.rc1 (git clone -b v4.rc1 ...). For build instructions, see here..

4. Get KLEE source

```
$ git clone https://github.com/klee/klee.git
```

5. Configure KLEE

KLEE must be built "out of source", so first create a build directory. You can create this wherever you like. Below, we assume you create this directory inside KLEEs repository.

```
$ mkdir build
```

Now cd into the build directory and run CMake to configure KLEE where <KLEE_SRC_DIRECTORY> is the path to the KLEE git repository you cloned in the previous step.

```
$ cd build
$ cmake <CMAKE_OPTIONS> <KLEE_SRC_DIRECTORY>
```

<CMAKE_OPTIONS> are the configuration options. These are documented in README-CMake.md.

6. Build KLEE

From the build directory created in the previous step run:

```
$ make
```

Important options are as follows:

KLEE can be run with different search strategies and command-line arguments. You can choose different search strategies with 2 different memory/time bound:

1. –search=dfs: Runs KLEE with the DFS search strategy.
2. –search=random-state: Runs KLEE with the maximum coverage search strategy.
3. Other search strategies can be seen by running "klee –help".
4. The command-line options "–max-memory=5000" and "–max-time=600" force KLEE to run on at most 5000 MB of Ram and for at most 10 minutes.

For more information, please check the KLEE Website.