Project Milestone I
CS639A - Fall 2020

Baldip Singh Bijlani, Yugesh Ajit Kothari

Progress Brief

The goal of this project was to identify if we can leverage Angelic verification to suppress false positive bug reports in C/C++ programs that use EmscriptenSDK APIs. As initial part of the project, we have set up the entire toolchain required to compile and verify such programs. After experimenting with some examples, we have also decided to restrict the scope of this project by concerning ourselves with memory-safety bugs only. Our next step was to find benchmarks, which is where the project currently is - however (as we explain ahead) since finding benchmarks was turning out to be rather difficult, we plan to seed the files we have with memory safety bugs manually.

Tasks Completed

We have been able to complete the following tasks :

1. Setup Docker to build and install necessary dependencies needed to run experiments.
2. Basic bash scripts to compile and run smack.
3. Tested setup by running a few examples (tests from emscripten repository) with different corral flags.
4. Converged scope of project to checking for memory-safety properties (e.g., null-check, etc) because smack supports instrumentation for memory-safety checks out-of-the-box.

Tasks in Progress[1]

1. Finding benchmarks that can leverage Angelic Verification technique, i.e. code on which corral gives false positives due to lack of modelling of emscripten APIs.
    - Such open source programs are virtually non-existent outside the Emscripten test-suite. Therefore, next point takes precedence over this.
2. Seeding available Emscripten tests with memory safety bugs in an unbiased fashion.

[1] Part of Milestone1, yet to be completed.

Challenges

1. LLVM versioning:

- Emscripten SDK always uses tip of the tree (ToT) LLVM build, which means using tools which depend on different versions of llvm posed a bit of inconvenience. There is no clear documentation from the Emscripten community (nor was their mailing list much help) to compile emsdk for a particular version of llvm, and therefore setting up tools took longer than expected.

2. System libraries:

- The emscripten sdk re-defines a LOT of the system library functions (for e.g., in libc/libcxx) and also changes typedef for a lot of types defined in system libraries. This caused some compilation issues and we had to figure out manually changing paths, creating symlinks and setting appropriate path variables.

3. Finding Benchmarks:

- This is perhaps the biggest challenge we are facing in this project. We have tried running smack+corral on multiple tests in the Emscripten compiler suite. Most report no errors, and some report false alarms. However, often these false alarms have nothing to do with external library calls in the emscripten sdk and are generated due to unrelated trivial technicalities (like standard file format of a .wav file). **Based on feedback received, our current plan is to create such benchmarks by manually seeding available code with memory safety bugs.**

Adjusted milestones

Put together, this is the remaining work for completion of the project :

13th November (Milestone 1.x, Internal milestone):
- Prepare benchmarks by seeding bugs.

16th November (Milestone 1.x, Internal milestone):
- Run baseline verification without corral models

20th November (Milestone 2):
- Experiment and prepare Corral models.
- Run verification
    - With Corral with manually written models/stubs for library calls

27th November (Milestone 2.x, Internal milestone):
- Run verification
    - With Corral using AV
- Write documentation for setup and benchmarks.