Artifact - Historia: Refuting Callback Reachability with Message-History Logics

ACM Reference Format:
. 2023. Artifact - Historia: Refuting Callback Reachability with Message-History Logics. 1, 1 (July 2023), 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

This document explains the artifact for the Historia paper [1]. The goal of this document is to first give a set of instructions for reproducing the experimental results and then give a technical explanation of how the implementation connects to the technical contributions. For inputs, this artifact takes a compiled Android application in the form of an APK file, a location in the application for the assertion, and a CBCFTL specification of realizable message histories. Outputs are either "safe" or "alarm" in which case an abstract message history witnessing the alarm will be available.

2 PREREQUISITES - RUNNING THE HISTORIA DOCKER CONTAINER

We have configured the experiments to be run within a Docker container provided with this artifact. This Docker file may be found in the root directory of this archive and is labeled `historia.docker`. Please follow the instructions to install docker from https://docs.docker.com/engine/install/.

System Requirements. [**TODO:** *min ram, reasonably modern cpu*]
Importing the docker container can be done with the following command.

```
docker import historia.docker
```

The docker container may be run with the following command. [**TODO:** *Is this enough res?*] [**TODO:** *expose web port and swap with jupyter command*]

```
docker run --memory="16G" --memory-swap="32G" --rm -it historia bash
```

[**TODO:** *keeps getting killed even with 64G???*]
All subsequent steps may be done through the web interface at a URL printed by the terminal window the docker command was run. This URL should start with http://localhost:8080 and contain an encoded security token. Opening this URL should show a Jupyter notebook. [**TODO:** *screenshot of jupyter notebook*]

Author's address:

3 REPRODUCING THE HISTORIA RESULTS

In this section, we explain how to generate the tables shown in the evaluation section. Since the full experiments require substantial system resources, we strongly suggest reproducing a subset of the experiments. However, all data and scripts are available to run the full experiment if desired.

Due to the required system resource requirements, the experiments were run on one of three available machines:

(1) AMD Ryzen 7 5800X3D 8-Core Processor with 128GB of ram.
(2) AMD EPYC 7763 64-Core Processor with 256GB of ram.
(3) Xeon Gold 6240R CPU 48-Core Processor with 192 GB of ram.

The number of timeouts and runtime required will depend on the available resources. [**TODO:** *we have curated a subset of the experiments that may be run on 8GB of ram*]

3.1 Running RQ1 (minimal resources)

[**TODO:** *just disable one that OOMs and have some instructions to re-enable*]

To run the experiments for the first table, open the terminal in the Jupyter notebook listed earlier and run the following commands:

```
cd home/bounder
bash runExperiments.sh
```

The output will be located in the file `home/bounder/log/logging.log`. Each row of Table 1 may be found labeled by "Row" followed by the row number and version (i.e. "bug" or "fix"). For example, the buggy version of `getAct`[??] is "Row 1 bug" and the fixed version is "Row 1 fix". There will be several rows of text displaying the data from the table. The output is labeled with "actual:" and may say "Witnessed", "Timeout", or [**TODO:** *safe*].

[**TODO:** *explain message counts etc*]

[**TODO:** *SM: explain how to compare flowdroid and infer*]

3.2 Running RQ1 (full)

[**TODO:**]

3.3 Running RQ2 (minimal resources)

3.4 Running RQ2 (full)

[**TODO:**]

[**TODO:** *third priority is comparison with infer/flow*]

4 RUNNING AND INTERPRETING HISTORIA ON CUSTOM INPUTS

[**TODO:** *second priority*]

Historia may be run on arbitrary Android applications as long as an APK can be compiled in debug mode. This is usually accomplished using the command `./gradlew assembleDebug` but will vary from application to application. A location and safety property must be chosen ahead of time. However, we recommend only writing CBCFTL specifications as needed.

Running Historia Through Jupyter. The recommended way to run Historia is using a Jupyter notebook. This allows the inputs to the tool to be defined using the Scala data structures for input rather than JSON.

[**TODO:** *give sample app to walk through this process*]

[**TODO:** *explain this process completely*]

[**TODO:** *explain counter examples*]

5 DEVELOPING FOR HISTORIA

[**TODO:** *fourth priority*]

In this section, we explain the implementation of each technical contribution in Historia.

5.1 Running Unit Tests

[**TODO:** *this may need to go in later connecting formalism section*]

5.2 Application-Only Control-Flow Graph

[**TODO:**]

5.3 Message-History Program Logic (MHPL)

[**TODO:**]

5.4 Callback Control-Flow Temporal Logic (CBCFTL)

[**TODO:**]

5.5 Combining Abstract Message Histories with Callback Control Flow

[**TODO:**]

5.6 Optimizations

[**TODO:**]

REFERENCES

[1] Shawn Meier, Sergio Mover, Gowtham Kaki, and Bor-Yuh Evan Chang. 2023. Historia: Refuting Callback Reachability with Message-History Logics.. In *Object-Oriented Programming Systems, Languages, and Applications (OOPSLA).*