Artifact Evaluation for Data-Driven Inference of Representation Invariants

ANONYMOUS AUTHOR(S)

1 INTRODUCTION

The Hanoi paper is available here. In this pdf, we:

- Describe the steps for installation of the artifact (§2).
- Describe the file structure of the artifact repository (§3).
- Describe how to run the program (§4).

2 INSTALLATION

Installation can be performed either with a VM, or manually.

2.1 Virtual Machine

The virtual machine is available here. The username is hanoi and the password is hanoi.

2.2 Manual Installation

The virtual machine installation is suggested for ease of use. The below table provides a description of the install step, the command we performed for that step (if applicable), and the relevant version (if applicable).

Description of Step	Command (if applicable)	Version (if applicable)
Download OS		Ubuntu 18.04.4
Download Opam	sudo apt install opam	1.2.2-6
Initialize Opam	opam init	
Switch OCaml Version	opam switch 4.07.0+flambda	4.07.0+flambda
Setup Switch in Current Shell	eval `opam config env`	
Install Dune	opam install dune	1.2.1
Install Core Dependencies	opam depext conf-m4.1	
Install Core	opam install core	0.11.3
Install Ppx_deriving	opam install ppx_deriving	4.2.1
Install Async	opam install async	0.11.0
Install CTypes Dependencies	opam depext conf-pkg-config.1.1	
Install Libffi-dev	sudo apt install libffi-dev	3.2.1-8
Install CTypes	opam install ctypes	0.14.0
Install CTypes-Foreign	opam install ctypes-foreign	0.4.0
Install Menhir	opam install menhir	20180905
Install Pip	sudo apt install python-pip	9.0.1-2.3~ubuntu1.18.04.1
Install EasyProcess	pip install easyprocess	0.2.10
Install MatPlotLib	pip install matplotlib	2.2.5

3 FILE STRUCTURE

In the following descriptions, `$` will refer to the repository root. In other words, on the VM, `$` means `~/HanoiArtifactEvaluation`.

`$/AE.pdf`	This document.
`$/code`	Folder containing the source code for Hanoi.
`$/code/benchmarks`	Folder containing the benchmarks used for data collection.
`$/code/benchmarks-no-helper`	Folder containing the benchmarks without helper functions added.
`$/code/Makefile`	File containing a number of commands for building and testing Hanoi.
`$/code/paper_run`	Folder containing data and graphs from the run used in the submitted version of the paper.
`$/code/HanoiCmdLine.exe`	(generated) Executable for running Hanoi.
`$/code/generated_data/generated_data.csv`	(generated) File containing the run data.
`$/code/generated_graphs/times.eps`	(generated) File containing the graph of time vs benchmarks completed.

4 RUNNING

We suggest evaluators check our program by generating the data, and ensuring the graphs are consistent with our graphs and our claims, and similarly for the tables.

The simplest means of getting data is using the `make generate-all` command from within `$/code`. This will create all of `HanoiCmdLine.exe`, `generated_data.csv`, and `times.eps`. This data generation is restartable – once a benchmark has been completely run, it will be saved in the csv. To rerun a benchmark, merely delete that benchmark row from the csv file. This can be used to validate the graphs and tables in the evaluation section of the paper. This command will take about a day to complete.

By default, we set up the scripts with a timeout 30 minutes, and to run each test only once (to take less time for the evaluators). The evaluators can change the timeout to t seconds if they update `TIMEOUT_TIME` in `generate_data.py` to $t + 5$. The evaluators can change each test to run n times if they update `REPETITION_COUNT` in `generate_data.py` to n. Increasing timeout time or repetition count increases the time it takes to run the benchmarks.

Evaluators can more easily read the csv file on the virtual machine by running
`csvtool readable generated_data.csv | view -`.

One can run individual benchmarks with `HanoiCmdLine.exe [filename]`.

Running `HanoiCmdLine.exe -help` will provide a variety of command-line options to run Hanoi with different options.

Running `python generate-data.py [folder]` will run all the benchmarks within a folder, and save them to `generated_data.csv`.

Running `python transform-data.py [file]` will make a time vs benchmarks completed graph for the benchmarks in a csv file.