Proceedings Paper:
Hall, M. and Walkinshaw, N. (2017) Data and analysis code for GP EFSM inference. In: 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME). 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME) , 02-07 Oct 2016, Raleigh, NC, USA. IEEE , p. 611. ISBN 978-1-5090-3806-0

https://doi.org/10.1109/ICSME.2016.22

Data and Analysis code for GP EFSM Inference

Neil Walkinshaw
School of Mathematics and Computer Science
University of Leicester, Leicester, UK
nw91@mcs.le.ac.uk

Mathew Hall
Department of Computer Science
University of Sheffield, Sheffield, UK
mathew.hall@shef.ac.uk

I. Description

This artifact captures the workflow that we adopted for our experimental evaluation in our ICSME paper on inferring state transition functions durfing EFSM inference [1]. To summarise, the paper uses Genetic Programming to infer data transformations, to enable the inference of fully 'computational' extended finite state machine models. This submission shows how we generated, transformed, analysed, and visualised our raw data. It includes everything needed to generate raw results and provides the relevant R-code in the form of a re-usable Jupyter Notebook (accompanied by a descriptive narrative).

The artifact consists of:

- Case studies used in our experiments
- A script to download and build the MINT tool
- A script to run the tool in the configurations we used
- A Jupyter Notebook that shows the analysis and reproduces the figures used in our paper
- A makefile that automates the process of launching the notebook for analysis
- Samples of the data generated by MINT

The artifact is contained in a Git repository available online [1]. To obtain a copy, clone it using `git clone`.

Section III describes how the artefact can be used to automatically download, build, and run our model inference tool. Section IV describes how to re-run the data analyses for our tool, how to explore new aspects of our data, and how to re-run our experiments with new data.

II. Requirements

The script for building and running MINT has been tested on OSX machines. In principle it should work on other platforms. It also requires a JDK (it has been built on 1.8, but it should work on 1.7) and Maven installation as well as Mercurial. Running the makefile requires Make to be installed.

III. Running MINT

The process of generating our results is encompassed in the `experiment` directory. Here we provide a Makefile and accompanying '`README`' file that describes how to use it. The Makefile automates the process of building MINT, including fetching its dependencies. It also includes pre-scripted goals (makefile targets) that will generate results for the case studies included in the paper. When run, the makefile will produce a JAR file for MINT (`EFSMTool.jar`) in the `experiment` directory.

To run the experiments, change to the `experiment` directory and run `make`. This will run MINT (after it has been built) 30 times for each case study generating results in the `results` directory. We ran our experiments on the Alice HPC service, specifying each configuration (case study and random seed) to parallelise the process.

IV. Running the Jupyter Notebook

The analysis is contained in a Jupyter Notebook[2] and uses R[3]. In order to use it, we provide two alternatives: running it natively (requiring installation of R, Jupyter, and the R kernel); or running it within a Docker container. In either use case, the Notebook is accessed via a web browser. We include instructions to run the notebook below (via the included Makefile targets).

The Jupyter notebook shows how the data generated by MINT is transformed and analysed. It includes a description of the process we followed to generate each of the figures in the paper. The notebook is interactive and can be used to explore and visualise the data in more depth than is possible within the paper.

1) Running natively: Jupyter can be installed by following the instructions on the Jupyter homepage. The R Kernel[4] for Jupyter must then be installed from within R. The notebook can then be launched using `make launch-jupyter-native` and will automatically launch a web browser for the notebook.

2) Running within a Docker container: The alternative route to running the notebook is to launch it in a Docker container. To do this, Docker needs to be installed from `http://docker.com`. Then the `make launch-jupyter-docker` command can be run to launch the server. Open `http://localhost:8888` to access the notebook. If using `docker-machine` use `docker-machine ip default`) to get the IP in place of `localhost`.

[1] `https://bitbucket.org/mathew_hall/icsme2016-data`

References

[1] N. Walkinshaw and M. Hall, Inferring Computational State Machine Models from Program Executions, Proceedings of the 32nd International Conference on Software Maintenance and Evolution (ICSME'16), 2016
[2] `http://jupyter.org/`
[3] `https://www.r-project.org/`
[4] `http://irkernel.github.io`