Foundational Response-Time Analysis as Explainable Evidence of Timeliness (Artifact)

Marco Maida
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Sergey Bozhko
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany
Saarbrücken Graduate School of Computer Science, Universität des Saarlandes, Germany

Björn B. Brandenburg
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Abstract

This artifact provides the means to validate and reproduce the results of the associated paper "Foundational Response-Time Analysis as Explainable Evidence of Timeliness". The artifact demonstrates how to *(i)* generate task sets needed to run the experiments, *(ii)* prepare and run POET on the generated input, *(iii)* plot the figures presented in the paper, and *(iv)* visually inspect the generated certificates.

2012 ACM Subject Classification Computer systems organization → Real-time systems; Software and its engineering → Formal software verification

Keywords and phrases hard real-time systems, response-time analysis, uniprocessor, Coq, Prosa, fixed priority, EDF, preemptive, non-preemptive, verification

Digital Object Identifier 10.4230/DARTS.8.1.7

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 803111), and from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 391919384.

Related Article Marco Maida, Sergey Bozhko, and Björn B. Brandenburg, "Foundational Response-Time Analysis as Explainable Evidence of Timeliness", in 34th Euromicro Conference on Real-Time Systems (ECRTS 2022), LIPIcs, Vol. 231, pp. 19:1–19:25, 2022.
`https://doi.org/10.4230/LIPIcs.ECRTS.2022.19`

Related Conference 34th Euromicro Conference on Real-Time Systems (ECRTS 2022), July 5–8, 2022, Modena, Italy

1 Scope

The associated paper introduces foundational response-time analysis (RTA) as a means to produce strong and independently checkable evidence of temporal correctness. In a foundational RTA, each response-time bound calculated comes with an auto-generated certificate of correctness — a short and human-inspectable sequence of machine-checked proofs that formally show the claimed bound to hold. As a proof of concept, the associated paper presents POET, the first foundational RTA tool. The artifact contains the source code of the Coq development as well as the Python-based tool and provides the means to validate and reproduce the results of the associated paper.

2 Content

The artifact package includes:

- The `README.md` (as well as `README.html`) file contains detailed instructions on how to run and validate the results of the experiments.
- The `prosa` directory contains the Coq part of the development. It is a modified version of the Prosa library. It has several subfolders:

  - The `proofgen` folder contains the main Coq part of POET (e.g., all refinements).
  - The folders `util`, `behavior`, `model`, `analysis`, and `results` contain the parts of the Prosa library required to support the proof generation described in the paper.
  - `scripts` is an auxiliary folder of the Prosa project that may be safely ignored.
- The `poet` directory contains the Python part of the development. It has several files and subfolders. The most interesting files/folders are the following:
  - The `templates/certificates` folder contains templates that are used by POET to generate `.v` files.
  - The `src/pipeline/coq_generator.py` file contains code that fills such templates with Coq-statements specific to the given task set.
  - The `src/rta` folder contains the part of the program responsible for the (untrusted) computation of the response times, the search space, etc.
- The `tsgenerator` directory is an auxiliary folder containing Python scripts needed to generate the kind of task sets used in the evaluation.
- The `plotter` directory is an auxiliary folder containing Python scripts used to plot the figures shown in the paper.
- The file `experiments_done.zip` contains the task sets and the corresponding analysis results that underlie the figures in the paper.
- The file `requirements.txt` lists the required Python libraries.
- The `Docker` directory contains additional files used to build the Docker images provided for ease of use.
- The Python scripts starting with a number correspond to the steps discussed in `README.md`.

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition, the artifact is also available at: `https://people.mpi-sws.org/~sbozhko/ECRTS22/POET.html`.

4 Tested platforms

The artifact was tested on a desktop computer using `64-bit macOS 11.6` and `64-bit Debian GNU/Linux 9 (stretch)`; it does not assume or require any particular hardware configuration. The artifact should work on any system that supports:

- Coq 8.14.0, coq-mathcomp-ssreflect 1.14.0, and CoqEAL 1.1.0.
- Python 3.7 or a later version

Note that the artifact also provides a docker image; hence, it is expected that the artifact is runnable on any system supporting docker.

5 License

The artifact is freely available under the BSD 2-Clause License.

6 MD5 sum of the artifact

441f79e266ae76a6001175de36c9852c

7 Size of the artifact

24.8 MB