A Formal Link Between Response Time Analysis and Network Calculus (Artifact)

Pierre Roux
ONERA, Toulouse, France
DTIS – Université de Toulouse, F-31055 Toulouse, France

Sophie Quinton
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France

Marc Boyer
ONERA, Toulouse, France
DTIS – Université de Toulouse, F-31055 Toulouse, France

Abstract

Classical Response Time Analysis (RTA) and Network Calculus (NC) are two major formalisms used for the verification of real-time properties. The related paper offer mathematical links between these two different theories. Based on these links, it then proves the equivalence of various key notions in both frameworks. This enables specialists of both formalisms to get increase confidence on their models, or even, like the authors, to discover errors in theorems by investigating apparent discrepancies between some notions expected to be equivalent.

The presented mathematical results are all mechanically checked with the interactive theorem prover Coq, building on existing formalizations of RTA and NC. Establishing such a link between NC and RTA paves the way for improved real-time analyses obtained by combining both theories to enjoy their respective strengths (e.g., multicore analyses for RTA or clock drifts for NC).

This artifact enables to reproduce these proofs.

2012 ACM Subject Classification Computer systems organization → Real-time system specification; Networks → Formal specifications; Software and its engineering → Formal methods; General and reference → Verification

Keywords and phrases Response Time Analysis, Network Calculus, dense time, discrete time, response time, formal proof, Coq

Digital Object Identifier 10.4230/DARTS.8.1.3

Related Article Pierre Roux, Sophie Quinton, and Marc Boyer, "A Formal Link Between Response Time Analysis and Network Calculus", in 34th Euromicro Conference on Real-Time Systems (ECRTS 2022), LIPIcs, Vol. 231, pp. 5:1–5:22, 2022. `https://doi.org/10.4230/LIPIcs.ECRTS.2022.5`

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

1 Scope

This artifact enables to reproduce the formal proofs of the related paper.

2 Content

The artifact package includes:

- a `README.md`;
- files with the `*.v` extension containing the source code of the Coq formal proof;
- a `Makefile` relying on a `_CoqProject` file to help build the proof.

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://zenodo.org/record/6472818`.

4 Tested platforms

The artifact requires a system able to run the Coq software with 10Gb of free disk space and 4Gb of memory.

5 License

6 MD5 sum of the artifact

d862829cd021560d6c30410459e1645a

7 Size of the artifact

16 kiB

A Virtual Machine

A virtual machine with all required softwares already installed is provided at `https://zenodo.org/record/6473253` with instructions in its file /home/ecrts2022/link/README.md. If need be, the login/password of the virtual machine are ecrts2022/ecrts2022. The remaining alternatively details installation and compilation on a fresh system.

B Installation and compilation on a fresh system

B.1 Prerequisites

This needs

- Coq (tested with version 8.13.2)
- Mathcomp (tested with version 1.13.0)
- Mathcomp Analysis (tested with version 0.3.13)
- Hierarchy Builder (tested with version 1.0.0)
- Mathcomp Dioid (tested with version 0.2)
- Mathcomp Zify (tested with version 1.2.0+1.12+8.13)

To install all those dependencies, the easisest solution is to use the package manager OPAM `https://opam.ocaml.org/`:

```
% opam repo add coq-released https://coq.inria.fr/opam/released
% opam update
% opam install coq.8.13.2 coq-mathcomp-algebra.1.13.0 coq-mathcomp-analysis.0.3.13 \
    coq-hierarchy-builder.1.0.0 coq-mathcomp-dioid.0.2 coq-mathcomp-zify.1.2.0+1.12+8.13
```

This should take a few dozen minutes. Note that you'll need OPAM 2 (old OPAM 1 won't work). This may require a few system dependencies, for instance the developement files of the GMP library which can be installed by `apt-get install libgmp-dev` on Debian based systems, before re-running `opam install ...`.

B.2 Installing Prosa

```
% git clone https://gitlab.mpi-sws.org/RT-PROOFS/rt-proofs.git
% cd rt-proofs
% git checkout e3567e9524184ad59c61cd6162114c217c046689
% ./create_makefile.sh --without-classic
% make  # or make -j3
% make install
```

B.3 Installing NCCoq

```
% git clone https://gitlab.mpi-sws.org/proux/nc-coq.git
% cd nc-coq
% git checkout dfbe838790b107f644b7939bc58e9daee9804760
% make  # or make -j3
% make install
```

B.4 Compilation

Once above prerequisites are installed, just type make:

```
% make
```

The proof succeeds when there is no errors and a `file.vo` file is produced for each `file.v`. One can also check for the absence of additional axioms in the code (presence of the keywords `Axiom`, `Parameter` or `Admitted`)

B.5 Documentation

To generate the documentation:

```
% make html
```

You can then open `html/toc.html` with your favorite browser.