Unrealizable Cores for Reactive Systems Specifications: Artifact

Shahar Maoz
Tel Aviv University
Tel Aviv, Israel

Rafi Shalom
Tel Aviv University
Tel Aviv, Israel

***Abstract*—This document describes the artifact that accompanies the ICSE'21 paper "Unrealizable Cores for Reactive Systems Specifications". The artifact includes the specifications that were used in the experiments that are described in the paper. It further includes an executable that allows interested readers to reproduce these experiments and inspect their results. Additionally, the executable is applicable to any specification in Spectra format, which allows conducting similar experiments over any Spectra specification. We hope the artifact will be useful for researchers who are interested in reactive synthesis, specifically in different means to deal with unrealizable specifications.**

I. Introduction

One of the main challenges of reactive synthesis, an automated procedure to obtain a correct-by-construction reactive system, is to deal with unrealizable specifications. One means to deal with unrealizability, in the context of GR(1), an expressive assume-guarantee fragment of LTL that enables efficient synthesis, is the computation of an unrealizable core, which can be viewed as a fault-localization approach.

In [9], we presented QuickCore, a novel algorithm that accelerates unrealizable core computations by relying on the monotonicity of unrealizability, on an incremental computation, and on additional properties of GR(1) specifications. We further presented Punch, a novel algorithm to efficiently compute all unrealizable cores of a specification. Finally, we presented means to correctly handle specifications that include higher-level constructs beyond pure GR(1) elements.

This document describes the reusable artifact that accompanies the paper. The artifact includes the specifications that were used in the experiments that are described in the paper. It further includes an executable that allow interested readers to reproduce these experiments and inspect their results, and the raw data that resulted from the experiments described in the paper.

II. An Overview of the Artifact

A. Corpus of Specifications

The artifact corpus of specifications consists of the benchmarks `SYNTECH15` and `SYNTECH17` [5], [7], [10], which include a total of 227 specifications of 10 autonomous Lego robots, written by 3rd year undergraduate computer science students in a project class taught by the authors of [7]. We use all the unrealizable GR(1) specifications from these benchmarks, i.e., 14 unrealizable specifications from `SYNTECH15` (which we label `SYN15U` in the paper) and 26 unrealizable specifications from `SYNTECH17` (which we label `SYN17U` in the paper).

In addition, the artifact corpus of specifications includes 5 different sizes of `AMBA` [2] and of `GENBUF` [3] (1 to 5 masters, 5 to 40 senders resp.), from each of the 3 variants of unrealizability described in [4]. We label these 30 specifications by `AM+GN` in the paper.

Finally, the artifact includes a folder named `Small`, which consists of several small specifications, which are not a part of the corpus. These include the running example specification of the paper (in file `LiftInt3.spectra`), its pure GR(1) form found in [1] (in file `AlurLift3.spectra`), and a few other specifications, mainly taken from recent papers about GR(1) unrealizability [6], [8]. Whereas most of the specification in our corpus are rather complicated, and some take very long time to process, these small specifications in the artifact allow instantaneous single core and all cores computations with small, easily observable correct results.

B. Implementation of Algorithms

The artifact includes implementations of several algorithms. First, implementations of three single core computation algorithms: delta debugging (`DDMin`), `QuickXplain`, and `QuickCore`. Second, implementations of three all cores computation algorithms: `TD`, which is a naive top down search algorithm, and two variants of the `Punch` algorithm, namely `PQC` and `PUD`.

The implementation is integrated into the Spectra synthesis environment [7], [10]. However, in the artifact, it is packaged such that experiments can independently run from the command line, outside the Spectra environment. In particular, executing the artifact does not require the installation of Spectra.

C. Using the Artifact

All experiments are run over specification sets by providing their folder name.

There are two different tests that run single core algorithms. One of the tests of single core algorithms provides the running times of `DDMin` and `QuickCore` over a specification set. It validates the cores found by `QuickCore`, and includes various statistics such as core sizes, number of realizability checks, etc. The other test of single core algorithms provides the running times of `DDMin` and `QuickXplain` over a specification set. It validates the cores found by `QuickXplain`, and similarly includes various statistics. It is also possible to determine the number of runs performed per specification. Performing multiple runs is required because JVM garbage collection and BDD dynamic-reordering add variance to running times.

There are three different experiments related to all cores computations, one for each of the three algorithms `TD`, `PQC`, and `PUD`. These experiments provide similar information and validation as above, plus information that is unique to all cores computations, such as the size of the minimal core, and the size of the intersection of all the cores. As a byproduct, the experiments create log files with additional information that is not reported in the paper, e.g., the time at which each core was computed. It is possible to set a timeout in minutes for runs of all cores algorithms. We used a 10 minute timeout in the experiments reported in the paper.

For convenience, we added scripts that run complete experiments over the whole corpus and include scripts that perform the exact experiments we conducted in order to obtain the data for the paper. We also included the raw data which is the output of our experiments on our specific setup.

The artifact includes detailed technical documentation on how to install it and how to execute these experiments (file names, folder names, options, etc.).

III. Potential Future Uses

We hope that the artifact will be useful for researchers who are interested in reactive synthesis. It may be specifically useful for researchers who are interested in different means to deal with unrealizable specifications. We consider the following potential future uses.

- Use the corpus of specifications as a benchmark when examining future algorithms for unrealizable single and all cores computations.
- Use the corpus of specifications and the executable in a future study that aims at examining the usefulness of cores to engineers.
- Use the executable on additional Spectra specifications, beyond those in our corpus, in a study of the algorithms presented in the paper.

IV. Acknowledgements

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 638049, SYNTECH).

References

[1] R. Alur, S. Moarref, and U. Topcu. Counter-strategy guided refinement of GR(1) temporal logic specifications. In *FMCAD*, pages 26–33. IEEE, 2013.

[2] R. Bloem, S. J. Galler, B. Jobstmann, N. Piterman, A. Pnueli, and M. Weiglhofer. Interactive presentation: Automatic hardware synthesis from specifications: a case study. In *2007 Design, Automation and Test in Europe Conference and Exposition, DATE 2007, Nice, France, April 16-20, 2007*, pages 1188–1193, 2007.

[3] R. Bloem, S. J. Galler, B. Jobstmann, N. Piterman, A. Pnueli, and M. Weiglhofer. Specify, compile, run: Hardware from PSL. *Electr. Notes Theor. Comput. Sci.*, 190(4):3–16, 2007.

[4] A. Cimatti, M. Roveri, V. Schuppan, and A. Tchaltsev. Diagnostic information for realizability. In *VMCAI*, pages 52–67, 2008.

[5] E. Firman, S. Maoz, and J. O. Ringert. Performance heuristics for GR(1) synthesis and related algorithms. *Acta Inf.*, 57(1-2):37–79, 2020.

[6] A. Kuvent, S. Maoz, and J. O. Ringert. A symbolic justice violations transition system for unrealizable GR(1) specifications. In *ESEC/FSE*, pages 362–372, 2017.

[7] S. Maoz and J. O. Ringert. Spectra: A specification language for reactive systems. *Software and Systems Modeling*, 2021. To appear.

[8] S. Maoz, J. O. Ringert, and R. Shalom. Symbolic repairs for GR(1) specifications. In *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, pages 1016–1026, 2019.

[9] S. Maoz and R. Shalom. Unrealizable cores for reactive systems specifications. In *ICSE*, 2021. To appear.

[10] Spectra Website. http://smlab.cs.tau.ac.il/syntech/spectra/.