EPiC Series in Computing

Volume 48, 2017, Pages 175–180

ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems

ARCH-COMP17 Repeatability Evaluation Report

Taylor T. Johnson[1]

Vanderbilt University,
Department of Electrical Engineering and Computer Science,
Institute for Software Integrated Systems,
Nashville, TN, United States
taylor.johnson@vanderbilt.edu
http://.www.TaylorTJohnson.com

Abstract

This report presents the results of the repeatability evaluation for a friendly competition for formal verification of continuous and hybrid systems. The friendly competition took place as part of the workshop Applied Verification for Continuous and Hybrid Systems (ARCH) in 2017. In its first edition, thirteen tools have been applied to solve benchmark problems for the six competition categories, of which, ten tools were evaluated and passed the repeatability evaluation. The repeatability results represent a snapshot of the current landscape of tools and the types of benchmarks for which they are particularly suited and for which others may repeat their analyses. Due to the diversity of problems in verification of continuous and hybrid systems, as well as basing on standard practice in repeatability evaluations, we evaluate the tools with pass and/or failing being repeatable. These results probably provide the most complete assessment of tools for the safety verification of continuous and hybrid systems up to this date.

1 Introduction

The presented *repeatability evaluation for verification of continuous and hybrid systems* summary for the ARCH friendly competition aims at providing an overview of the usability and reproducibility of results for the current capabilities of verification tools. The verification community has a rich history of publishing strong papers emphasizing computational contributions, but subsequent re-creation of these computational elements is often challenging because details of the implementation are unavoidably absent in the paper due to space restrictions. To address this challenge, some authors post code and data to their websites, but there is often only marginal formal incentive to do so, and typically there is no easy way to determine whether others can actually use or extend the results. Owing to such factors, computational results often become non-reproducible, sometimes even by the research group which originally produced them. The goal of this repeatability evaluation process is to improve the reproducibility of computational results for the tools competing on the selected benchmarks evaluated in the competition. More broadly, a key goal of the competition itself is to improve repeatability and interoperability of these software tools, to help develop more standard benchmarks for evaluating tools and easing comparisons of these tools and their analyses.

This report summarizes the repeatability evaluation (RE) results obtained in the 2017 friendly competition of the ARCH workshop[1]. The obtained results in the competition have been verified by an independent repeatability evaluation conducted by the author of this report. To establish further trustworthiness of the results, the artifacts, code, documentation, benchmarks, etc. with which the repeatability results have been obtained are publicly available on the ARCH website.

The competition featured six categories and thirteen software tools, ten of which were evaluated in this repeatability evaluation, the remaining three of which were not evaluated due to time constraints and are not included in the discussion below. The six categories of problems with the respective lead for each category are:

- AFF: affine and piecewise affine dynamics (lead: Matthias Althoff),
- NLN: nonlinear dynamics (lead: Xin Chen),
- HPWC: piecewise constant dynamics (lead: Goran Frehse),
- HBMC: bounded model checking (lead: Lei Bu),
- FALS: falsification (lead: Georgios Fainekos), and
- PARA: parameter-centric problems (lead: Georgios Fainekos).

The ten tools evaluated, broken into their competition categories are:

- AFF
  - CORA: Matthias Althoff [2],
  - HyLAA: Stanley Bak [5],
  - Flow*: Xin Chen [8],
  - SpaceEx: Goran Frehse [9],
  - Isabelle/HOL: Fabian Immler [10, 11],
  - HyDRA: Stefan Schupp [13], and
  - XSpeed: Rajarshi Ray [12].
- NLN
  - CORAMatthias Althoff [2],
  - Flow*Xin Chen [8], and
  - Isabelle/HOLFabian Immler [10, 11].
- HPWC
  - Bach: Lei Bu [6],
  - SpaceEx(and the PHAVerscenario): Goran Frehse [9],
  - HyDRA: Stefan Schupp [13], and
  - XSpeed: Rajarshi Ray [12].
- HBMC

[1]Workshop on Applied Verification for Continuous and Hybrid Systems (ARCH), cps-vo.org/group/ARCH

  - Bach: Lei Bu [6],
  - HyDRA: Stefan Schupp [13], and
  - XSpeed: Rajarshi Ray [12].

- FALS
  - s-Taliro: Georgios Fainekos [3].

- PARA
  - s-Taliro: Georgios Fainekos [3]..

Due to timing constraints, three tools that participated in the competition were not included in the repeatability evaluation. They are: Axelerator [7], Lyse, and VeriSiMPL [1].

2 Repeatability Evaluation Plan, Execution, and Results

The repeatability evaluation was conducted following the presentations of the competition results at the ARCH'17 workshop. The basic mechanism followed in the repeatability evaluation was similar to that done in related conferences, such as the Hybrid Systems: Computation Control conference series, which has featured a repeatability evaluation in the past several iterations, including this year (http://hscc2017.ece.illinois.edu/re.html). Three basic criteria are generally evaluated: coverage, instructions, and quality, each of which may be rated on a scale of one through five, where one indicates a missing component or significantly below acceptability, and five indicates the criteria significantly exceeds expectations. Coverage measures the repeatability packages' ability to regenerate the images, tables, and log files presented in the competition. Instructions measures the packages' ability to describe to another researcher how to reproduce the results, including installation of the tool and how to execute it. Quality measures the packages' level of documentation and trustworthiness of results with respect to the quality of the software tool and the results it produces. This report does not describe the ratings of these review criteria for each tool evaluated, only the aggregate result of whether the submission was repeatable or not.

The competitors were sent instructions to provide their tool setup instructions and tool execution commands for the benchmarks evaluated in their respective categories. The repeatability evaluation was performed on the competition benchmarks, the selection of which has been conducted within the forum of the ARCH website (cps-vo.org/group/ARCH), which is visible for registered users and registration is open for anyone. The tool authors responded to the request within two weeks, and the author of this report conducted an independent repeatability evaluation in the week following receipt of the tools and benchmarks.

All tools—except Axelerator, Lyse, and VeriSiMPLwhich did not participate for time constraints—participating in all categories of the competition participated in the repeatability evaluation and were evaluated to have passed the repeatability evaluation with their benchmark analysis results deemed repeatable. The repeatability evaluation was conducted by the author, and took approximately one week to complete. Most tools were able to be installed and executed by the author with their provided instructions, but the author interacted with three tool developers for additional instruction for installing, executing, and/or plotting their results. Overall, the tool developers provided sufficient information to install, execute, and repeat the results they obtained in the competition. The majority of the tool authors provided a script to execute their tool with appropriate parameters for all the benchmarks.

The host machine ($M_{Repeatability\_Host}$) used for executing the tools was a Microsoft Surface Pro 4 with a quad-core Intel Core i7 6650U processor at 2.20GHz and 16GB RAM. A VMWare virtual machine ($M_{Repeatability\_VM}$) with 64-bit Ubuntu 16.04 was used for all tools execpt for Flow*, which was executed in its own Oracle VirtualBox virtual machine ($M_{Repeatability\_VM\_Flow*}$) with Ubuntu 16.04. The VMWare virtual machine $M_{Repeatability_VM}$was limited to 8GB of available RAM and access to all cores. The VirtualBox virtual machine $M_{Repeatability_VM_Flow*}$was limited to 4GB of available RAM and access to a single core.

3 Conclusion and Outlook

This report presents a summary of the repeatability evaluation for the first friendly competition for the formal verification of continuous and hybrid systems, conducted as part of the ARCH'17 workshop. The detailed reports for the categories can be found in the proceedings and on the ARCH website: cps-vo.org/group/ARCH. All documentation, benchmarks, and execution scripts for the repeatability evaluation are also archived on the ARCH website.

For future competitions and repeatability evaluations, several factors may be improved by the community in future competitions. First, while several participants used the somewhat common input format of SpaceExin part via HyST [4], there is greater need for utilizing a common language for specifying models and specifications. A related challenge is making amenable the ability to specify comparable parameters for different tools, as well as the particular problem domain/category (verification vs. falsification, etc.). Second, a greater challenge compared to standardizing on an input format, is determining more quantitative means to compare the output results of the tools. While figures and yes/no/maybe verified results for a given specification are one method, developing a common output format may yield greater gains and improve the ability to make quantitative comparisons. Third, this evaluation did not consider any performance aspects, which once the competition has taken greater hold, will be a significant challenge for the repeatability evaluation to also repeat the performance results. Thankfully for this challenge, several other communities have developed means for making fair comparisons, such as in the software verification competition (SV-COMP). There are many other aspects that can improve, but it is the author's hope that this competition and evaluation serve to improve both the reproducibility of computational results in the hybrid and continuous systems verification community, as well as improve the scientific rigor of the field.

4 Acknowledgments

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311, EPCN 1509804, and SHF 1527398, the Air Force Research Laboratory (AFRL) through contract numbers FA8750-15-1-0105 and FA8650-12-3-7255 via subcontract number WBSC 7255 SOI VU 0001, and the Air Force Office of Scientific Research (AFOSR) under contract numbers FA9550-15-1-0258 and FA9550-16-1-0246. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

A Specification of Used Machines

A.1 $M_{\text{Repeatability\_Host}}$

- Processor: Intel Core i7-6650U @ 2.20GHz
- Memory: 16GB
- Average CPU Mark on www.cpubenchmark.net: 4918 (full), 1812 (single thread)
- Host Operating System: Windows 10

A.2 $M_{\text{Repeatability\_VM}}$

- Processor: Intel Core i7-6650U @ 2.20GHz (4 cores available to VM)
- Memory: 8GB
- Average CPU Mark on www.cpubenchmark.net: 4918 (full), 1812 (single thread)
- VMWare Virtual Machine Operating System: Ubuntu 16.04

A.3 $M_{\text{Repeatability\_VM\_Flow*}}$

- Processor: Intel Core i7-6650U @ 2.20GHz (1 core available to VM)
- Memory: 4GB
- Average CPU Mark on www.cpubenchmark.net: 4918 (full), 1812 (single thread)
- Oracle VirtualBox Virtual Machine Operating System: Ubuntu 16.04

References

[1] Dieky Adzkiya, Yining Zhang, and Alessandro Abate. Verisimpl 2: An open-source software for the verification of max-plus-linear systems. *Discrete Event Dynamic Systems*, 26(1):109–145, March 2016.

[2] M. Althoff and D. Grebenyuk. Implementation of interval arithmetic in CORA 2016. In *Proc. of the 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 91–105, 2016.

[3] Yashwanth Annpureddy, Che Liu, Georgios Fainekos, and Sriram Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2011.

[4] Stanley Bak, Sergiy Bogomolov, and Taylor T. Johnson. HyST: A source transformation and translation tool for hybrid automaton models. In *Proc. of the 18th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC)*. ACM, 2015.

[5] Stanley Bak and Parasara Sridhar Duggirala. HyLAA: A tool for computing simulation-equivalent reachability for linear systems. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, HSCC '17, pages 173–178, New York, NY, USA, 2017. ACM.

[6] Lei Bu, You Li, Linzhang Wang, and Xuandong Li. Bach: Bounded reachability checker for linear hybrid automata. In *Proceedings of the 2008 International Conference on Formal Methods in Computer-Aided Design*, FMCAD '08, pages 9:1–9:4, Piscataway, NJ, USA, 2008. IEEE Press.

[7] Dario Cattaruzza, Alessandro Abate, Peter Schrammel, and Daniel Kroening. *Unbounded-Time Analysis of Guarded LTI Systems with Inputs by Abstract Acceleration*, pages 312–331. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[8] X. Chen, Erika Abraham, , and Sriam Sankaranarayanan. Talyor model flowpipe construction for non-linear hybrid systems. In *IEEE Real-Time Systems Symposium*, pages 183–192, 2012.

[9] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In *Computer Aided Verification (CAV)*, LNCS. Springer, 2011.

[10] Fabian Immler. Verified reachability analysis of continuous systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 37–51. Springer, 2015.

[11] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[12] Rajarshi Ray, Amit Gurung, Binayak Das, Ezio Bartocci, Sergiy Bogomolov, and Radu Grosu. *XSpeed: Accelerating Reachability Analysis on Multi-core Processors*, pages 3–18. Springer International Publishing, 2015.

[13] Stefan Schupp, Erika Ábrahám, Ibtissem Ben Makhlouf, and Stefan Kowalewski. *HyPro: A C++: ALibrary of State Set Representations for Hybrid Systems Reachability Analysis*, pages 288–294. Springer International Publishing, 2017.