EPiC Series in Computing
Volume 54, 2018, Pages 128–134
ARCH18. 5th International Workshop on Applied Verification of Continuous and Hybrid Systems

ARCH-COMP18 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 the 2nd International Competition on Verifying Continuous and Hybrid Systems (ARCH-COMP'18). The competition took place as part of the workshop Applied Verification for Continuous and Hybrid Systems (ARCH) in 2018. In its second edition, twenty-five tools submitted artifacts for the repeatability evaluation and applied to solve benchmark problems for seven competition categories. 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.

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 2018 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 (`https://cps-vo.org/group/ARCH`) and a Git version control repository (`https://gitlab.com/goranf/ARCH-COMP`).

The repeatability evaluation of the competition featured seven categories and 25 software tools, where several tools participated in multiple categories but have been counted distinctly for their participation in each category. The categories of problems in which tools participated in the repeatability evaluation are:

- AFF: affine and piecewise affine dynamics (8 tools),
- FALS: falsification (1 tool),
- HBMC: bounded model checking (2 tools),
- HPWC: piecewise constant dynamics (3 tools),
- HSTP: hybrid systems theorem proving (3 tools),
- NLN: nonlinear dynamics (6 tools), and
- SM: stochastic models (3 tools).

The tools evaluated, broken into their competition categories are:

- AFF
  - C2E2: Sayan Mitra [8, 7],
  - CORA: Matthias Althoff [1],
  - HyLAA: and continuous-time HyLAA (HyLAA^C): Stanley Bak [3],
  - Flow*: Xin Chen [6],
  - SpaceEx: Goran Frehse [9],
  - HyDRA: Stefan Schupp [15], and
  - JuliaReach: Marcelo Forets [4].
- FALS
  - FalStar: Zhenya Zhang and Gidon Ernst [19].
- HBMC
  - Bach: Lei Bu [5] and
  - HyDRA: Stefan Schupp [15].
- HPWC
  - Bach: Lei Bu [5]

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

  - PHAVer-lite: Enea Zaffanella, and
  - SpaceEx (and the PHAVer scenario): Goran Frehse [9].
- HSTP
  - HHL Prover: Hengjun Zhao [18],
  - KeYmaera 3: Stefan Mitsch and Andre Platzer [14], and
  - KeYmaera X: Stefan Mitsch and Andre Platzer [10].
- NLN
  - CORA: Matthias Althoff [1],
  - Flow*: Xin Chen [6], and
  - Isabelle/HOL: Fabian Immler [12, 13].
- SM (repeatability provided by Nathalie Cauchi)
  - SReachTools: Abraham Vinod [17],
  - FAUST^2: Soudjani Sadegh [16], and
  - C2E2: Arnd Hartmanns [11].

Several tools that participated in the competition did not participate in the repeatability evaluation.

2 Repeatability Evaluation Plan, Execution, and Results

The repeatability evaluation was conducted following the presentations of the competition results at the ARCH'18 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 (https://www.hscc2018.deib.polimi.it/repeatability-evaluation). 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, which were collected on a Git repository (`https://gitlab.com/goranf/ARCH-COMP`) by the competitors issuing commits and subsequent pull/merge requests that were reviewed and approved by the author. 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.

For all the tools listed above, which are those participating in the repeatability evaluation, all 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 two weeks to complete. Most tools were able to be installed and executed by the author with their provided instructions, but the author interacted with some tool developers for additional instruction for installing, executing, and/or plotting their results, in some cases interacting through the version control repository. Overall, the tool developers provided sufficient information to install, execute, and repeat the results they obtained in the competition, although there were some issues with installation, such as missing dependencies or incompatible library versions. The majority of the tool authors provided a script to execute their tool with appropriate parameters for all the benchmarks.

The host machine ($M_{\text{Repeatability\_Host}}$) used for executing the tools was a Microsoft Surface Book 2 with a quad-core (8 logical cores) Intel Core i7 8650U processor at 1.90GHz and 16GB RAM. A VMWare virtual machine ($M_{\text{Repeatability\_VM}}$) with 64-bit Ubuntu 18.04 LTS was used for all tools except for Flow*, which was executed in its own Oracle VirtualBox virtual machine ($M_{\text{Repeatability\_VM\_Flow*}}$) with Ubuntu 16.04, which was converted to a VMWare virtual machine image. The VMWare virtual machine $M_{\text{Repeatability_VM}}$was limited to 4GB of available RAM and access to four cores.

3 Conclusion and Outlook

This report presents a summary of the repeatability evaluation for the second competition for the formal verification of continuous and hybrid systems, conducted as part of the ARCH'18 workshop. The detailed reports for the categories can be found in the proceedings and on the ARCH website: http://cps-vo.org/group/ARCH. All documentation, benchmarks, and execution scripts for the repeatability evaluation are also archived on the ARCH website, and authors contributed their repeatability evaluations to the Git repository: https://gitlab.com/goranf/ARCH-COMP.

For future competitions and repeatability evaluations, several factors may be improved by the community in future competitions. First, while the somewhat common input format of SpaceExin part via HyST [2] provides some means for standardizing problem specifications, there is still a greater need for utilizing a common language for specifying models and specifications. Providing the ability to specify comparable parameters across different tools, as well as the particular problem domain/category (verification vs. falsification, etc.), remains a major challenge. Second, a greater challenge remains compared to standardizing inputs, is determining more quantitative means to compare the output results of the tools, although some libraries for common representations of reachable sets are starting to become available that may aid this process in the future, such as HyPro [15]. Figures of reachable sets and yes/no/maybe verified results for a given specification are means to make comparisons currently, but developing and standardizing a common output format may provide increased benefits and improve the ability to make quantitative comparisons between methods and tools. Third, the evaluation and competition so far did not consider any performance comparisons, but as the competition evolves, this remains 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 with respect to performance criteria, such as in the software verification competition (SV-COMP). Beyond these suggested improvements, there are still numerous aspects to improve, but in part through this competition and evaluation, our efforts may serve to enhance the reproducibility of computational results and increase the scientific rigor in verifying these systems.

4 Acknowledgments

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311, CNS 1713253, SHF 1527398, and SHF 1736323, the Air Force Office of Scientific Research (AFOSR) under contract numbers FA9550-15-1-0258 and FA9550-18-1-0122, and the Defense Advanced Research Projects Agency (DARPA) under contract number FA8750-18-C-0089. 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 AFOSR, DARPA, or NSF.

A Specification of Used Machines

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

- Processor: Intel Core i7-8650U @ 1.90GHz
- Memory: 16GB
- Average CPU Mark on `www.cpubenchmark.net`: 8923 (full), 2269 (single thread)
- Host Operating System: Windows 10

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

- Processor: Intel Core i7-8650U @ 1.90GHz (4 cores available)
- Memory: 4GB
- Average CPU Mark on `www.cpubenchmark.net`: 8923 (full), 2269 (single thread)
- VMWare Virtual Machine Operating System: Ubuntu 18.04 LTS

References

[1] 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.

[2] 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.

[3] 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.

[4] Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Frédéric Viry, Andreas Podelski, and Christian Schilling. Reach set approximation through decomposition with low-dimensional sets and high-dimensional matrices. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (Part of CPS Week)*, HSCC '18, pages 41–50, New York, NY, USA, 2018. ACM.

[5] 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.

[6] 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.

[7] Parasara Sridhar Duggirala, Chuchu Fan, Matthew Potok, Bolun Qi, Sayan Mitra, Mahesh Viswanathan, Stanley Bak, Sergiy Bogomolov, Taylor T. Johnson, Luan Viet Nguyen, Christian Schilling, Andrew Sogokon, Hoang-Dung Tran, and Weiming Xiang. Tutorial: Software tools for hybrid systems verification, transformation, and synthesis: C2e2, hyst, and tulip. In *Proceedings of the IEEE Multi-Conference on Systems and Control (¡a href="http://www.msc2016.org/"¿MSC 2016¡/a¿)*, Las Vegas, NV, USA, September 2016.

[8] ParasaraSridhar Duggirala, Sayan Mitra, Mahesh Viswanathan, and Matthew Potok. C2e2: A verification tool for stateflow models. In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 9035 of *Lecture Notes in Computer Science*, pages 68–82. Springer Berlin Heidelberg, 2015.

[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] Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25*, pages 527–538, Cham, 2015. Springer International Publishing.

[11] Arnd Hartmanns and Holger Hermanns. The modest toolset: An integrated environment for quantitative modelling and verification. In Erika Abraham and Klaus Havelund, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 8413 of *Lecture Notes in Computer Science*, pages 593–598. Springer Berlin Heidelberg, 2014.

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

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

[14] André Platzer and Jan-David Quesel. Keymaera: A hybrid theorem prover for hybrid systems (system description). In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning*, pages 171–178, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

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

[16] Sadegh Esmaeil Zadeh Soudjani, Caspar Gevaerts, and Alessandro Abate. Faust2: Formal abstractions of uncountable-state stochastic processes. In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 272–286, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[17] Abraham P. Vinod and Meeko M. K. Oishi. Scalable underapproximative verification of stochastic lti systems using convexity and compactness. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (Part of CPS Week)*, HSCC '18, pages 1–10, New York, NY, USA, 2018. ACM.

[18] Shuling Wang, Naijun Zhan, and Liang Zou. An improved hhl prover: An interactive theorem prover for hybrid systems. In Michael Butler, Sylvain Conchon, and Fatiha Zaïdi, editors, *Formal Methods and Software Engineering*, pages 382–399, Cham, 2015. Springer International Publishing.

[19] Z. Zhang, G. Ernst, S. Sedwards, P. Arcaini, and I. Hasuo. Two-layered falsification of hybrid systems guided by monte carlo tree search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2018.