EPiC Series in Computing
Volume 61, 2019, Pages 162–169
ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems

ARCH-COMP19 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 3rd International Competition on Verifying Continuous and Hybrid Systems (ARCH-COMP'19). The competition took place as part of the workshop Applied Verification for Continuous and Hybrid Systems (ARCH) in 2019, affiliated with the Cyber-Physical Systems and Internet of Things (CPS-IoT Week'19). In its third edition, twenty-five tools submitted artifacts through a Git repository for the repeatability evaluation, applied to solve benchmark problems for eight competition categories. The majority of participants adhered to new requirements for this year's repeatability evaluation, namely to submit scripts to automatically install and execute tools in containerized virtual environments (specifically Dockerfiles to execute within Docker). 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 2019 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 eight categories and 21 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 (4 tools),
- AINNCS: artificial intelligence and neural network control systems (2 tools),
- FALS: falsification (1 tool),
- HBMC: bounded model checking (1 tools),
- HPWC: piecewise constant dynamics (3 tools),
- HSTP: hybrid systems theorem proving (2 tools),
- NLN: nonlinear dynamics (4 tools), and
- SM: stochastic models (4 tools).

The tools evaluated, broken into their competition categories are:

- AFF
  - CORA: Nikolas Kochdumper and Matthias Althoff [1],
  - HyLAA: and continuous-time HyLAA (HyLAA^C): Stanley Bak [3],
  - SpaceEx: Goran Frehse [11], and
  - JuliaReach: Marcelo Forets [8].
- AINNCS
  - nnv: Hoang-Dung Tran, Diego Manzanas-Lopez, Patrick Musau, Weiming Xiang, and Taylor T. Johnson [21, 20], and
  - Verisig: Taylor Carpenter and Radoslav Ivanov [14].
- FALS
  - FalStar: Gidon Ernst, Zhenya Zhang, Paolo Arcaini, and Yoriyuki Yamagata [22].

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

- HBMC
  - Bach: Lei Bu [9].
- HPWC
  - Bach: Lei Bu [9]
  - PHAVer-lite: Enea Zaffanella, and
  - SpaceEx (and the PHAVer scenario): Goran Frehse [11].
- HSTP
  - KeYmaera 3: Stefan Mitsch and Andre Platzer [17], and
  - KeYmaera X: Stefan Mitsch, Andre Platzer, Andrew Sogokon, and Yong Kiam Tan [12].
- NLN
  - Ariadne: Luca Geretti and Pieter Collins [4, 5],
  - CORA: Niklas Kochdumper and Matthias Althoff [1],
  - Isabelle/HOL: Fabian Immler [13, 15], and
  - JuliaReach: Marcelo Forets and Christian Schilling [7].
- SM (repeatability provided by Nathalie Cauchi)
  - HYPEG: Carina Pilch and Anne Remke [16],
  - LyapMMC: Mahmoud Salamati and Sadegh Soudjani [18],
  - SCDPN: Henk Blum and Hao Ma [6], and
  - StocHy: Nathalie Cauchi and Alessandro Abate [10].

Several tools that participated in the competition did not participate in the repeatability evaluation, so only those that participated are listed. In future iterations, we encourage all participants of the competition to complete the repeatability evaluation to make it easier for others in the research community to build on these results, and are considering requiring repeatability participation in the future.

2 Repeatability Evaluation Plan, Execution, and Results

The repeatability evaluation was conducted following the presentations of the competition results at the ARCH'19 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://hscc2019.eecs.umich.edu/html/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

Category	Tool	Dockerfile?	Execution Scripts?
AFF	Hylaa	Yes	Yes
	SpaceEx	Yes	Yes
	CORA	No	Yes
	JuliaReach	Yes	Yes
AINNCS	nnv	Yes	Yes
	Verisig	Yes	Yes
FALS	FalStar	No	Yes
HBMC	BACH	Yes	Yes
HPWC	BACH	Yes	Yes
	PHAVer	No	Yes
	PHAVerLite	Yes	Yes
HSTP	KeYmaera X 443	Yes	Yes
	KeYmaera X 463	Yes	Yes
NLN	Ariadne	Yes	Yes
	CORA	No	Yes
	Isabelle	Yes	Yes
	JuliaReach	Yes	Yes
SM	HYPEG	No	Yes
	LyapMMC	Yes	Yes
	SCDPN	Yes	Yes
	StocHy	Yes	Yes

Table 1: Summary of repeatability artifacts for each category and tool that participated in the evaluation.

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 participants 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. Relative to past iterations of the repeatability evaluations, where the evaluation was conducted primarily on a VMWare virtual machine by installing and executing all the tools, the usage of Docker significantly simplified the repeatability and we strongly encourage using this type of mechanism for repeatability evaluations. All tools were able to be installed by setting up the Docker containers, then executed by the author with their provided Dockerfiles and 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 used Docker by providing Dockerfiles, and also provided a script to execute their tool with appropriate parameters for all the benchmarks.

The host machine ($\mathrm{M}_{\mathrm{Repeatability\_Host}}$) 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. Docker Desktop for Windows version 2.0.0.3 (31259, build: 8858db3) was used and containers ($\mathrm{M}_{\mathrm{Repeatability\_VM}}$) were configured with 2 available cores and 8 GB available memory.

3 Conclusion and Outlook

This report presents a summary of the repeatability evaluation for the third competition for the formal verification of continuous and hybrid systems (ARCH-COMP'19), conducted as part of the ARCH'19 workshop at CPS-IoT Week'19. The detailed reports for the categories can be found in the proceedings (https://cps-vo.org/group/ARCH/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 still be improved by the community in future competitions. First, while the somewhat common input format of SpaceEx in 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. Particularly, for the stochastic models category, there are currently no standardized formats, so effort is highly recommended to address this standardization, although this area is even more challenging than non-stochastic hybrid systems, as there are many ways to model sources of uncertainty (such as through stochastic transitions a la Markov chain transitions, continuous uncertainty with stochastic differential equations, etc.). Similarly, for the AINNCS category, standardization of formats for representing both plants (e.g. as SpaceEx) and machine learning components (e.g., neural networks) should be standardized, and for the neural networks, recent efforts such as the Open Neural Network Exchange (ONNX) format or the more recent formalization of neural network semantics and specifications such as VNN-LIB should be leveraged. 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 still 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 [19]. 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 number SHF 1736323, the Air Force Office of Scientific Research (AFOSR) under contract number 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 $\mathbf{M_{Repeatability\_Host}}$

- 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 $\mathbf{M_{Repeatability\_VM}}$

- Processor: Intel Core i7-8650U @ 1.90GHz (4 cores available)
- Memory: 8GB
- Average CPU Mark on www.cpubenchmark.net: 8923 (full), 2269 (single thread)
- Docker Desktop for Windows Version 2.0.0.3 (31259, Build: 8858db3), various container operating systems used per individual participant Dockerfiles and dependencies

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] Andrea Balluchi, Alberto Casagrande, Pieter Collins, Alberto Ferrari, Tiziano Villa, and Alberto L. Sangiovanni-Vincentelli. Ariadne: a framework for reachability analysis of hybrid automata. In *PROCEEDINGS OF THE INTERNATIONAL SYPOSIUM ON MATHEMATICAL THEORY OF NETWORKS AND SYSTEMS*, 2006.

[5] Luca Benvenuti, Davide Bresolin, Pieter Collins, Alberto Ferrari, Luca Geretti, and Tiziano Villa. Assume-guarantee verification of nonlinear hybrid systems with ariadne. *International Journal of Robust and Nonlinear Control*, 24(4):699–724, 2014.

[6] Henk A.P. Blom, Hao Ma, and G.J. (Bert) Bakker. Interacting particle system-based estimation of reach probability for a generalized stochastic hybrid system. *IFAC-PapersOnLine*, 51(16):79 – 84, 2018. 6th IFAC Conference on Analysis and Design of Hybrid Systems ADHS 2018.

[7] Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Kostiantyn Potomkin, and Christian Schilling. Juliareach: A toolbox for set-based reachability. In *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, pages 39–44, New York, NY, USA, 2019. ACM.

[8] 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.

[9] 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.

[10] Nathalie Cauchi and Alessandro Abate. StocHy: Automated verification and synthesis of stochastic processes. In Tomáš Vojnar and Lijun Zhang, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 247–264, Cham, 2019. Springer International Publishing.

[11] 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.

[12] 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.

[13] Fabian Immler. Verified reachability analysis of continuous systems. In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems: 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings*, pages 37–51. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[14] Radoslav Ivanov, James Weimer, Rajeev Alur, George J. Pappas, and Insup Lee. Verisig: Verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, pages 169–178, New York, NY, USA, 2019. ACM.

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

[16] Carina Pilch, Fabian Edenfeld, and Anne Remke. Hypeg: Statistical model checking for hybrid petri nets: Tool paper. In *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS 2017, pages 186–191, New York, NY, USA, 2017. ACM.

[17] 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.

[18] Mahmoud Salamati, Sadegh Soudjani, and Rupak Majumdar. Approximate time bounded reachability for CTMCs and CTMDPs: A lyapunov approach. In Annabelle McIver and Andras Horvath, editors, *Quantitative Evaluation of Systems*, pages 389–406, Cham, 2018. Springer International Publishing.

[19] Stefan Schupp, Erika Ábrahám, Ibtissem Ben Makhlouf, and Stefan Kowalewski. HyPro: A c++: A library of state set representations for hybrid systems reachability analysis. In Clark Barrett, Misty Davies, and Temesghen Kahsai, editors, *NASA Formal Methods: 9th International Symposium, NFM 2017, Moffett Field, CA, USA, May 16-18, 2017, Proceedings*, pages 288–294. Springer International Publishing, 2017.

[20] Weiming Xiang and Taylor T Johnson. Reachability analysis and safety verification for neural network control systems. *arXiv preprint arXiv:1805.09944*, 2018.

[21] Weiming Xiang, Hoang-Dung Tran, and Taylor T. Johnson. Output reachable set estimation and verification for multi-layer neural networks. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, March 2018.

[22] 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.