EPiC Series in Computing

Volume 74, 2020, Pages 175–183

ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)

ARCH-COMP20 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 4th International Competition on Verifying Continuous and Hybrid Systems (ARCH-COMP'20). The competition took place as part of the workshop Applied Verification for Continuous and Hybrid Systems (ARCH) in 2020, affiliated with the IFAC World Congress. In its fourth edition, twenty-eight tools submitted artifacts through a Git repository for the repeatability evaluation, applied to solve benchmark problems for seven competition categories. The majority of participants adhered to the 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), and several categories used performance evaluation information from a common execution platform. 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 participating verification tools. The verification community publishes 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 papers. To address this challenge, some authors post code and data to their websites, but there is often limited 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 researchers who originally produced them. Recently, the community has instituted artifact evaluations and re-

peatability evaluations in various phases of review processes to address these issues. The goal of the repeatability evaluation for ARCH-COMP is to improve the reproducibility of computational results for the tools competing on the selected benchmarks evaluated in the competition and to provide further trustworthiness of the results.

This report summarizes the repeatability evaluation (RE) results obtained in the 2020 friendly competition of the ARCH workshop[1]. The results obtained 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 28 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),
- AINNCS: artificial intelligence and neural network control systems (3 tools),
- FALS: falsification (0 tools),
- HSTP: hybrid systems theorem proving (3 tools),
- NLN: nonlinear dynamics (6 tools),
- PCDB: piecewise constant dynamics and bounded model checking (3 tools; merged in this iteration from the earlier categories HBMC: bounded model checking and HPWC: piecewise constant dynamics), and
- SM: stochastic models (5 tools).

The tools evaluated, broken into their competition categories are:

- AFF
  - C2E2 [15, 14],
  - CORA [2],
  - HyLAA and continuous-time HyLAA (HyLAA^C) [4],
  - SpaceEx [16],
  - HyDRA [25],
  - JuliaReach [9], and
  - XSpeed [24].
- AINNCS
  - NNV [32, 31, 27, 28],
  - ReachNN* [19], and
  - VenMAS [1].

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

- FALS
  - No tools submitted repeatability evaluation artifacts.
- HSTP
  - HHL Prover [30],
  - Isabelle/HOL [20], and
  - KeYmaera X [23, 17].
- NLN
  - Ariadne [5, 7],
  - CORA [2],
  - Dynibex [13],
  - Flow* [12],
  - Isabelle/HOL [20], and
  - JuliaReach [8].
- PCDB
  - Bach [10]
  - PHAVer-lite [6], and
  - XSpeed [24].
- SM
  - AMYTISS [21],
  - FAUST^2 [26],
  - Mascot-SDS [18, 22],
  - SReachTools [29], and
  - StocHy [11].

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 primarily before and partially following the presentations of the competition results at the ARCH'20 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://berkeleylearnverify.github.io/HSCC_2020/re.html). Three basic criteria are generally evaluated: coverage, instructions, and

Category	Tool	Dockerfile?	Execution Scripts?	Performance Evaluation?
AFF	C2E2	Yes	Yes	Yes
	CORA	Yes	Yes	Yes
	Hydra	Yes	Yes	Yes
	Hylaa	Yes	Yes	Yes
	JuliaReach	Yes	Yes	Yes
	SpaceEx	Yes	Yes	Yes
	XSpeed	Yes	Yes	Yes
AINNCS	NNV	Yes	Yes	No
	ReachNN*	No	Yes	No
	VenMAS	No	Yes	No
FALS	-	-	-	
HSTP	HHL	Yes	Yes	No
	Isabelle	Yes	Yes	No
	KeYmaera X	Yes	Yes	No
NLN	Ariadne	Yes	Yes	Yes
	CORA	No	Yes	Yes
	Dynibex	Yes	Yes	Yes
	Flow*	Yes	Yes	Yes
	Isabelle	Yes	Yes	Partially
	JuliaReach	Yes	Yes	Yes
PCDB	BACH	Yes	Yes	Yes
	PHAVerLite	Yes	Yes	Yes
	XSpeed	Yes	Yes	Yes
SM	AMYTISS	Yes	Yes	No
	FAUST^2	No	Yes	No
	Mascot-SDS	No	Yes	No
	SReachTools	No	Yes	No
	StocHy	No	Yes	No

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

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 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 of this report. 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. As in the last iteration repeatability evaluation at ARCH-COMP19, the usage of Docker significantly simplified the repeatability and we strongly encourage using this type of mechanism for repeatability evaluations, relative to earlier efforts where the evaluation was conducted primarily on a VMWare virtual machine by installing and executing all the tools. 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. All tools that provided Dockerfiles were able to be installed by setting up the Docker containers, then 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. The host machine ($M_{\text{Repeatability\_Host}}$) used for executing the tools and benchmarks was an Amazon EC2 g4dn.4xlarge instance.

New this year, several categories provided batch execution scripts that would execute all tools on all benchmarks in a given category, with a standardization process conducted on the CPS-VO forums for the output format to generate performance comparison tables in the individual category reports. This process in particular had a few difficulties as it only had been tested in most cases when attempting the repeatability evaluation, but most issues were resolved, and several categories (AFF, NLN, PCDB) presented performance evaluation results generated for the repeatability evaluation. 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.

3 Conclusion and Outlook

This report summarizes the repeatability evaluation for the fourth competition for the formal verification of continuous and hybrid systems (ARCH-COMP'20), conducted as part of the ARCH'20 workshop at the 2020 IFAC World Congress. 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.

In previous iterations of the competition and corresponding repeatability evaluation, several aspects to improve the process were identified. In particular, there are still needs for (1) greater standardization of input formats, (2) standardization of output formats and results, and (3) increased execution in a common computational platform so that results, particularly performance metrics, are more meaningful. Of these challenges, this iteration of the repeatability evaluation improved upon the standardization of output formats and results, and execution on a common computational platform.

For future competitions and repeatability evaluations, several factors may still be improved by the community in future competitions. While the relatively common input format of SpaceEx in part via HyST [3] provides some means for standardizing problem specifications, there is still a greater need for utilizing a common language for specifying models and specifications. Future participants may make further use of the HyST design studio on the CPS-VO to address this issue (https://cps-vo.org/group/hyst). For some categories though, there are more fundamental issues with input formats. 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 pursued, 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 (http://www.vnnlib.org/) should be leveraged, and taking advantage of lessons learned in the Verification of Neural Networks Competition (VNN-COMP, https://sites.google.com/view/vnn20/vnncomp). As has been the case in past iterations, 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 [25]. 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, while this year for the first time performance comparisons were considered in several categories, this remains a significant challenge for the repeatability evaluation to also repeat the performance results. 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 the community.

4 Acknowledgments

The material presented in this report is based upon work supported by the National Science Foundation (NSF) under grant number FMitF 1918450, the Air Force Office of Scientific Research (AFOSR) through contract number FA9550-18-1-0122, and the Defense Advanced Research Projects Agency (DARPA) Assured Autonomy program through contract number FA8750-18-C-0089. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR, DARPA, or NSF.

A Specifications of Used Machines

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

- Amazon EC2 Instance Type: g4dn.4xlarge
- Processor: Intel Xeon Scalable (2nd Generation Cascade Lake), 16 vCPUs (AWS/EC2 Custom), 2.5 GHz base, roughly Xeon Gold 5200 Series with 24 physical cores
- Memory: 64GB
- Average CPU Mark on www.cpubenchmark.net: 25740 (full), 2396 (single thread) (for comparable Xeon Gold 5200 series)
- Host Operating System: Ubuntu

References

[1] Michael E. Akintunde, Elena Botoeva, Panagiotis Kouvaros, and Alessio Lomuscio. Formal verification of neural agents in non-deterministic environments. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, page 25–33, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems.

[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] 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.

[4] 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.

[5] 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.

[6] Anna Becchi and Enea Zaffanella. Revisiting polyhedral analysis for hybrid systems. In Bor-Yuh Evan Chang, editor, *Static Analysis*, pages 183–202, Cham, 2019. Springer International Publishing.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Julien Alexandre dit Sandretto and Alexandre Chapoutot. Validated explicit and implicit Runge–Kutta methods. *Reliable Computing*, 22(1):79–103, Jul 2016.

[14] 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 (MSC 2016)*, Las Vegas, NV, USA, September 2016.

[15] 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.

[16] 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.

[17] 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.

[18] Kyle Hsu, Rupak Majumdar, Kaushik Mallik, and Anne-Kathrin Schmuck. Multi-layered abstraction-based controller synthesis for continuous-time systems. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (Part of CPS Week)*, HSCC '18, page 120–129, New York, NY, USA, 2018. Association for Computing Machinery.

[19] Chao Huang, Jiameng Fan, Wenchao Li, Xin Chen, and Qi Zhu. Reachnn: Reachability analysis of neural-network controlled systems. *ACM Trans. Embed. Comput. Syst.*, 18(5s), October 2019.

[20] 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.

[21] Abolfazl Lavaei, Mahmoud Khaled, Sadegh Soudjani, and Majid Zamani. Amytiss: Parallelized automated controller synthesis for large-scale stochastic systems. In Shuvendu K. Lahiri and Chao Wang, editors, *Computer Aided Verification*, pages 461–474, Cham, 2020. Springer International Publishing.

[22] Rupak Majumdar, Kaushik Mallik, and Sadegh Soudjani. Symbolic controller synthesis for büchi specifications on stochastic systems. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, HSCC '20, New York, NY, USA, 2020. Association for Computing Machinery.

[23] 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.

[24] Rajarshi Ray, Amit Gurung, Binayak Das, Ezio Bartocci, Sergiy Bogomolov, and Radu Grosu. Xspeed: Accelerating reachability analysis on multi-core processors. In Nir Piterman, editor, *Hardware and Software: Verification and Testing: 11th International Haifa Verification Conference, HVC 2015, Haifa, Israel, November 17-19, 2015, Proceedings*, pages 3–18. Springer International Publishing, 2015.

[25] 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.

[26] 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.

[27] Hoang-Dung Tran, Feiyang Cai, Manzanas Lopez Diego, Patrick Musau, Taylor T. Johnson, and Xenofon Koutsoukos. Safety verification of cyber-physical systems with reinforcement learning control. *ACM Trans. Embed. Comput. Syst.*, 18(5s), October 2019.

[28] Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor T. Johnson. Nnv: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In Shuvendu K. Lahiri and Chao Wang, editors, *Computer Aided Verification*, pages 3–17, Cham, 2020. Springer International Publishing.

[29] 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.

[30] 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.

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

[32] 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.