EPiC Series in Computing
Volume 90, 2022, Pages 222–230
Proceedings of 9th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH22)

ARCH-COMP22 Repeatability Evaluation Report

Taylor T. Johnson[1]

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

Abstract

The repeatability evaluation for the 6th International Competition on Verifying Continuous and Hybrid Systems (ARCH-COMP'22) is summarized in this report. The competition took place as part of the workshop Applied Verification for Continuous and Hybrid Systems (ARCH) in 2022, affiliated with the 41st International Conference on Computer Safety, Reliability and Security (SAFECOMP'22). In its sixth edition, 25 tools had submitted artifacts through a Git repository for the repeatability evaluation, which were applied to solve benchmark instances through 7 competition categories. The majority of participants adhered to the specifications of this year's repeatability evaluation, which was to submit scripts to automatically install and execute tools in containerized virtual environments (specifically Dockerfiles to execute within Docker containers). Some categories used performance evaluation information from a common execution platform. The repeatability results represent a snapshot of current tools and the types of benchmarks on which they are well suited, and so that 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 of being repeatable.

1 Introduction

This report summarizes the *repeatability evaluation* of the 2022 friendly competition of the ARCH workshop[1], namely the ARCH-COMP friendly competition, and aims to provide an overview of the usability and reproducibility of results for the participating verification tools. The verification community publishes papers that emphasize 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,

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

sometimes even by the researchers who originally produced them. Over about the past decade and increasingly in the past few years, the community has instituted artifact evaluations and repeatability 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 confidence in the results.

This remainder of this report presents a summary of the repeatability evaluation (RE) results. The results obtained in the competition have been verified by an independent repeatability evaluation conducted by the author of this report. To establish further confidence in 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 17 software tools, where several tools participated in multiple categories, but have been counted distinctly for their participation in each category. The categories of problems that tools participated in the repeatability evaluation are:

- AFF: affine and piecewise affine dynamics (5 tools),
- AINNCS: artificial intelligence and neural network control systems (4 tools),
- FALS: falsification (2 tools),
- HSTP: hybrid systems theorem proving (3 tools),
- NLN: nonlinear dynamics (6 tools),
- PCDB: piecewise constant dynamics and bounded model checking (4 tools), and
- SM: stochastic models (1 tool).

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

- AFF
  - CORA [1],
  - SpaceEx [11],
  - HyDRA [27],
  - JuliaReach [8, 26], and
  - XSpeed [25].
- AINNCS
  - NNV [32, 31, 28, 29],
  - JuliaReach [8],
  - CORA [21], and
  - POLAR [14].
- FALS
  - FalStar [33] and

  - ARIsTEO [23].
- HSTP
  - HHL Prover [30],
  - IsaVODEs, and
  - KeYmaera X [24, 12].
- NLN
  - Ariadne [4, 6],
  - CORA [1],
  - Dynibex [10],
  - Kaa [20],
  - JuliaReach [7], and
  - KeYmaera X [24, 12].
- PCDB
  - Bach [9]
  - PHAVer-lite [5],
  - SAT-Reach [22], and
  - XSpeed [25].
- SM
  - SySCoRE [13].

All of the tools listed above were deemed repeatable based on the evaluation, as summarized next and detailed further in the next section that describes in more depth the process and results. In advance of the competition, the author merged pull/merge requests submitted by the tool authors in the Git repository, and executed the tools on the common AWS platform, detailed in the appendix. In some instances, interaction and revisions to the scripts, Dockerfiles, or other artifacts were necessary, and the author generally provided feedback in the comments of the pull requests to provide a centralized location of the feedback. In a couple instances, the authors provided execution logs and results in addition or in contrast to these steps, but the author also reviewed these results, albeit the requested execution setup is ideal. For those tools and categories that reproduced performance measurements, the author posted the results typically as figures or archives to the comments of the pull requests, where the authors checked and confirmed them.

A few tools may have participated in the competition, but did not participate in the repeatability evaluation, so only those tools that participated in the repeatability evaluation by providing information through the Git repository 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 given the enhancement in confidence it provides.

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'22 workshop. The basic mechanism followed in the repeatability evaluation was similar to that done in related conferences, and builds on the evaluation conducted in prior iterations of ARCH-COMP [15, 16, 17, 18, 19].The primary difference in the ARCH-COMP repeatability relative to those done at conferences is this evaluation was done solely by the author of this report, and not an evaluation committee. In many repeatability evaluations, three basic criteria are generally evaluated: coverage, instructions, and quality, each of which may be rated on a scale, typically of one through five, where one indicates a missing component or significantly below acceptability, and five indicates the criteria significantly exceeds expectations. Coverage evaluates the repeatability packages' ability to regenerate the images, tables, and log files presented in the competition. Instructions evaluates the packages' ability to describe to another researcher how to reproduce the results, including installation of the tool and how to execute it. Quality evaluates 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 as deemed by the submitted package and corresponding artifacts.

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 to enable sharing of these models and benchmarks.

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 required approximately four weeks to complete. As in the last three iterations of the repeatability evaluation at ARCH-COMP19 [17], ARCH-COMP20 [18], and ARCH-COMP21 [19], the usage of Docker significantly simplified the repeatability process, 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, which required significantly more time to conduct and was more error-prone. The majority of tool authors that followed these specifications 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 ($\mathrm{M}_{\mathrm{Repeatability\_Host}}$) used for executing the tools and benchmarks was an Amazon EC2 g4dn.4xlarge instance. As aforementioned, a couple participants submitted logs and execution artifacts instead of an execution environment due to the specific requirements of these frameworks, but were also evaluated as being sufficiently representative and repeatable based on the instructions provided. Finally, some participants submitted CodeOcean capsules, which is another and perhaps superior method, as the actual built container is available in perpetuity and also has an underlying Dockerfile, although it makes more complex the reproduction of performance evaluation measures.

As begun at ARCH-COMP20, 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) presented performance evaluation results generated for the repeatability evaluation in their competition results and category reports. 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 brief report summarizes the repeatability evaluation for the fifth competition for the formal verification of continuous and hybrid systems (ARCH-COMP'22), conducted as part of the ARCH'22 workshop at the 41st International Conference on Computer Safety, Reliability and Security (SAFECOMP'22). 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.

As in previous iterations of the competition and corresponding repeatability evaluation, several aspects to improve the process were identified. In particular, across the hybrid systems verification community, there are still needs for (1) greater standardization of input formats, (2) standardization of output formats and results, and (3) improved execution in a common computational platform so that results, particularly performance metrics and relative comparisons, are more meaningful. Of these challenges, this iteration of the repeatability evaluation continued some improvement upon the standardization of output formats and results, and execution on a common computational platform, with both the AFF and NLN categories including partial performance evaluation results produced through this repeatability process on standardized execution hardware. This however could be improved further in future iterations with increased standardization of the aggregated results, such as by using spreadsheets automatically generated by the batch scripts.

For future competitions and repeatability evaluations, several factors may still be improved by the community, participants, and organizers. 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. Future participants may make increased use of the HyST design studio on the CPS-VO to address this issue, if desired, as it provides a common input and execution platform (`https://cps-vo.org/group/hyst`). In some categories however, there still remain more fundamental issues with input formats and specifications. Particularly, for the stochastic models and falsification categories, there are currently no standardized formats, so effort is highly recommended to address such 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.) and that in falsification, often model agnostic ("black box") approaches are used. For the AINNCS category, standardization of formats for representing both plants (e.g. as SpaceEx models) and machine learning components (e.g., neural networks) should continue to be pursued, and for the neural networks, recent community 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/vnn2022) [3]. As has been the case in past iterations of ARCH-COMP, 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.

A second challenge still remains to determine 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 [27]. 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. This however remains a challenge for the community as a whole, beyond ARCH-COMP itself.

Thirdly, while this iteration continued for the third time partial performance comparisons in several categories, this remains a significant challenge for the repeatability evaluation to also repeat the performance results fairly, as this is an important criterion for potential industrial application of these methods. In the future, performance evaluation should continue to be pursued, both for the potential end-user comparisons and for understanding which methods are most effective for which problems.

Finally, for future iterations, rather than a single person (the author) reproducing the results of the categories, we suggest forming a committee with a representative from each category to repeat the results of that category. In earlier iterations of ARCH-COMP this was unnecessary, but the numbers of categories, tools, and complexity of artifacts therein has risen, and it is time to distribute the workload, and also take advantage of the localized expertise of participants within each category. While arguments could be made regarding independence of reproducing the results and this would now be done by one of the participants in a category, as the competition is a friendly one, and as similar conflicts exist in conferences, it is a minor concern relative to the enhanced benefits of having the localized expertise. Mechanisms could be deployed for fairness, such as a voting scheme where two or more participants in a given category reproduce the results of the others in that category. An alternative mechanism under discussion is to use an automated evaluation and submission process, although this requires in-advance infrastructure and greater planning for the participants, relative to just needing to provide Dockerfiles and execution scripts, so there are advantages and disadvantages. In either case, this will significantly ease the workload of the evaluation, and also likely improve the usability of the tools, as given the number of packages to repeat by one person would be decreased and the category participant would be more familiar with the specific intricacies of the given category.

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 1910017 and 2220426, the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-22-1-0019, 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 DARPA or NSF.

A Specifications of Used Machines

A.1 $\mathrm{M}_{\textbf{Repeatability\_Host}}$

- 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] 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, Changliu Liu, and Taylor Johnson. The second international verification of neural networks competition (vnn-comp 2021): Summary and results, 2021.

[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] 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.

[6] 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.

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

[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] Sofie Haesaert, Petter Nilsson, and Sadegh Soudjani. Formal multi-objective synthesis of continuous-state mdps. In *2021 American Control Conference (ACC)*, pages 3428–3433, 2021.

[14] Chao Huang, Jiameng Fan, Xin Chen, Wenchao Li, and Qi Zhu. POLAR: A polynomial arithmetic framework for verifying neural-network controlled systems. In *International Symposium on Automated Technology for Verification and Analysis*, pages 414–430. Springer, 2022.

[15] Taylor T. Johnson. ARCH-COMP17 repeatability evaluation report. In Goran Frehse and Matthias Althoff, editors, *ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 48 of *EPiC Series in Computing*, pages 175–180. EasyChair, 2017.

[16] Taylor T. Johnson. ARCH-COMP18 repeatability evaluation report. In Goran Frehse, editor, *ARCH18. 5th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 54 of *EPiC Series in Computing*, pages 128–134. EasyChair, 2018.

[17] Taylor T. Johnson. ARCH-COMP19 repeatability evaluation report. In Goran Frehse and Matthias Althoff, editors, *ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 61 of *EPiC Series in Computing*, pages 162–169. EasyChair, 2019.

[18] Taylor T Johnson. ARCH-COMP20 repeatability evaluation report. In Goran Frehse and Matthias Althoff, editors, *ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)*, volume 74 of *EPiC Series in Computing*, pages 175–183. EasyChair, 2020.

[19] Taylor T. Johnson. Arch-comp21 repeatability evaluation report. In Goran Frehse and Matthias Althoff, editors, *8th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21)*, volume 80 of *EPiC Series in Computing*, pages 153–160. EasyChair, 2021.

[20] Edward Kim and Parasara Sridhar Duggirala. Kaa: A python implementation of reachable set computation using bernstein polynomials. In Goran Frehse and Matthias Althoff, editors, *ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)*, volume 74 of *EPiC Series in Computing*, pages 184–196. EasyChair, 2020.

[21] Niklas Kochdumper, Christian Schilling, Matthias Althoff, and Stanley Bak. Open-and closed-loop neural network verification using polynomial zonotopes. *arXiv preprint arXiv:2207.02715*, 2022.

[22] Atanu Kundu, Sarthak Das, and Rajarshi Ray. SAT-Reach: A bounded model checker for affine hybrid systems. *ACM Trans. Embed. Comput. Syst.*, oct 2022.

[23] Claudio Menghi, Shiva Nejati, Lionel Briand, and Yago Isasi Parache. Approximation-refinement testing of compute-intensive cyber-physical models: An approach based on system identification. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 372–384, 2020.

[24] 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.

[25] 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.

[26] Christian Schilling, Marcelo Forets, and Sebastian Guadalupe. Verification of neural-network control systems by integrating taylor models and zonotopes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7):8169–8177, Jun. 2022.

[27] 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.

[28] 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.

[29] 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.

[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.

[33] 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.