EPiC Series in Computing

Volume 110, 2026, Pages 166–170

Proceedings of 13th Int. Workshop on Applied Verification for Continuous and Hybrid Systems

ARCH-COMP'26 Repeatability Evaluation Report

Nico Holzinger and Tobias Ladner

Technical University of Munich, Germany
{nico.holzinger, tobias.ladner}@tum.de

Abstract

The repeatability evaluation for the 13th International Competition on Verifying Continuous and Hybrid Systems (ARCH-COMP'26) is summarized in this report. The competition was held as part of the Applied Verification for Continuous and Hybrid Systems (ARCH) workshop in 2026. In its 13th edition, participants submitted their tools via an automated evaluation system developed over recent years. Each submission includes a Dockerfile and the necessary scripts for running the tool, enabling consistent execution in a containerized environment with all dependencies preinstalled. This setup improves comparability by running all tools on the same hardware. Submissions and results are automatically synchronized with a Git repository for repeatability evaluation and long-term archiving. We plan to further extend the evaluation system by refining the submission pipeline, aiming to enable automated evaluation across all competition categories.

1 Introduction

This report summarizes the status of this year's ARCH[1] friendly competition and provides an overview of the participating tools in each category. ARCH-COMP aims to improve the repeatability of code produced by researchers and industry partners by archiving the self-contained submissions, which are tested on an independent platform. This is in strong contrast to the usual practice, where code is often either not available at all or can barely reproduce the results: While there are many platforms to submit code along with a paper, there is, unfortunately, only little incentive to do so after a paper is accepted. Thus, the code might stay within the research group and if not developed further, certain aspects break over time such as dependencies of used libraries, making it impossible to reproduce the results – even for the original authors of the paper. Our automated evaluation system[2] prevents this issue as the code is run on independent hardware, where all dependencies have to be specified for the code to be runable. As all tools are run on the same hardware, the comparability of the results, such as computation time, is improved as well. The code and the obtained results are archived in a Git repository[3], categorized by year, category, and participated tool. Each tool folder (`/<year>/<category>/<tool>`) contains a script that runs the entire evaluation with a single click. The results obtained via the evaluation system are archived within `./results_ref` as reference.

[1] Applied Verification for Continuous and Hybrid Systems (ARCH): https://cps-vo.org/group/ARCH
[2] Evaluation System of ARCH-COMP: https://arch.repeatability.cps.cit.tum.de
[3] Archive: https://gitlab.com/goranf/ARCH-COMP

The remainder of this report is structured as follows: In Sec. 2, we provide a high-level overview of the repeatability evaluation, broken down by category and listing the participating tools. In Sec. 3, we present details about this year's evaluation, including improvements over the previous year. Finally, in Sec. 4, we conclude the report and outline our goals for next year's ARCH-COMP.

2 Repeatability Evaluation Overview

The repeatability evaluation of the competition featured seven categories and eleven software tools, where several tools participated in multiple categories, but have been counted distinctly for their participation in each category. While the introduction of the automatic evaluation system has led to an overall improvement in the repeatability evaluation, not all categories have participated in the automatic evaluation yet. The categories that tools participated in the repeatability evaluation are:

- AFF: affine and piecewise affine dynamics (2 tools),
- AINNCS: artificial intelligence and neural network control systems (4 tools),
- FALS: falsification (no tools participating in the repeatability evaluation),
- HSTP: hybrid systems theorem proving (no tools participating in the repeatability evaluation),
- NLN: nonlinear dynamics (5 tools),
- PCDB: piecewise constant dynamics and bounded model checking (no tools participating in the repeatability evaluation), and
- SM: stochastic models (no tools participating in the repeatability evaluation).

For the categories that have tools that participated in the repeatability evaluation, the tools evaluated, broken into their competition categories and alphabetically sorted, are:

- AFF:
  - CORA [1], and
  - JuliaReach [4];
- AINNCS:
  - CORA [1, 2],
  - CROWN-Reach [10],
  - immrax [6], and
  - JuliaReach [4];
- NLN:
  - Ariadne [3],
  - CORA [1],
  - DynIbex [5],

– JuliaReach [4], and
– PRoTECT [9];

All of the tools listed above were deemed repeatable based on the evaluation, as detailed in the next section. The repeatability package for each tool in its respective category is available in the ARCH competition archive[4].

3 Repeatability Evaluation Details

The procedure for the repeatability evaluation builds on previous iterations of ARCH-COMP first held in 2017 [7], and on the automatic evaluation system, which was introduced in 2023 [8]. An overview of the submission and evaluation workflow is shown in Fig. 1.

Users submit their tools as a zip file via the website of the evaluation system[5]. The submission is stored and scheduled by the evaluation system before being moved to a worker server (see Sec. A for specifications), where it is executed in a containerized environment using Docker. To make the submission reproducible, the dependencies and setup of the tool are specified by the tool authors in the respective Dockerfile. The generated verification results and plots are saved in a results folder, which is extracted after the run and archived together with the submission itself[6]. The verification results should be stored in a standardized CSV file so that they can be displayed on the website and reused easily in the report.

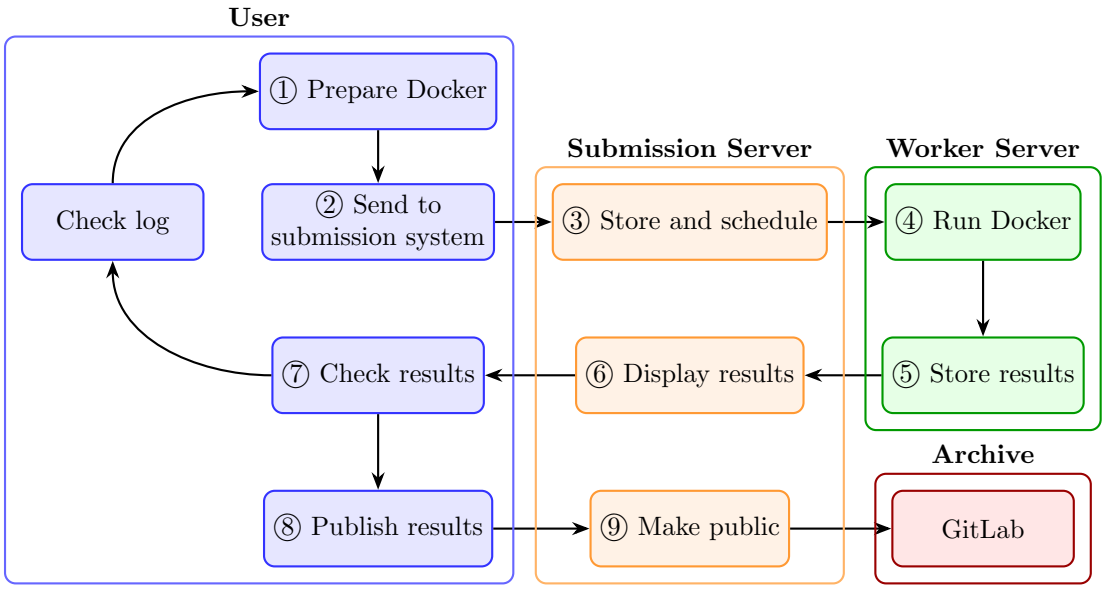

Figure 1: Overview of the ARCH-COMP repeatability evaluation workflow, from Docker-based tool preparation and submission to execution, result inspection, and public archiving.

[4] Archive: https://gitlab.com/goranf/ARCH-COMP
[5] Evaluation System of ARCH-COMP: https://arch.repeatability.cps.cit.tum.de
[6] Archive: https://gitlab.com/goranf/ARCH-COMP

After execution, the evaluation system displays the produced results and provides a log file for the complete submission run. This allows tool authors to double-check the results before publication and to identify bugs that may occur when running the code within the evaluation system. If the repeatability check fails, or if the reported results differ from the expected behavior, authors can revise their submission and repeat the process. Once the results are confirmed, they can be published and added to the public archive. The automatic evaluation system therefore gives immediate feedback on whether the tools are repeatable, as well as on the benchmark results and runtimes of the submission. Thus, the effort that authors invest in preparing their submissions is also reflected in the repeatability evaluation.

In this year's iteration, we redesigned the submission website to improve its maintainability, usability, and overall user experience. The updated website provides a more modern and intuitive interface for tool authors, making it easier to navigate the submission process, inspect evaluation results, and access relevant information. These changes aim to lower the effort required to prepare and revise submissions, while keeping the repeatability evaluation process transparent and accessible.

In ARCH-COMP, users usually know all the specifications of each benchmark in advance, and they can fine-tune their tools to obtain optimal results. However, this expert knowledge is not available when the tool is used externally, and default settings are usually chosen. As in last year's iteration of ARCH-COMP, we also experimented in the AFF category by using secret benchmarks that were not known to the participants of this category, in order to avoid fine-tuning their tools. The specifications of the secret benchmarks are only added prior to the evaluation of the submission in a standardized format. Details of this format are given in the AFF category report published alongside this report.

4 Conclusion and Outlook

This report summarizes the repeatability evaluation for the 13th edition of the competition on formal verification of continuous and hybrid systems (ARCH-COMP'26), held as part of the ARCH'26 workshop. Detailed category reports are available in the proceedings[7] and on the ARCH website[8]. All documentation, benchmarks, and execution scripts related to the repeatability evaluation are archived online. Looking forward, we aim to include every ARCH-COMP category in the automated evaluation system to ensure that all results are fully reproducible. We are also refining the system by addressing issues identified during this year's run. Overall, the automated evaluation system has streamlined the process and contributes significantly to the community by enabling one-click reproducibility.

Acknowledgments

The author gratefully acknowledges financial support from the research training group ConVeY (grant GRK 2428) and the project AL 1185/33-1, both funded by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG). Special thanks go to Volkan Şahin for his significant contributions to the development and maintenance of the evaluation system.

[7] ARCH proceedings: https://cps-vo.org/group/ARCH/proceedings
[8] ARCH website: https://cps-vo.org/group/ARCH

A Hardware Specification

This year, we run all tools on the same hardware using tool-specific docker images. The specification of the server used for the evaluation is given below:

- Processor: AMD EPYC 7742 64-Core
- Memory: 995 GB
- OS: Ubuntu 22.04
- Docker: 20.10.21

References

[1] M. Althoff. An introduction to CORA 2015. In *Proceedings of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151, 2015.

[2] Matthias Althoff, Niklas Kochdumper, Tobias Ladner, Maximilian Perschl, and Mark Wetzlinger. CORA manual. *Technical University of Munich*, 2025.

[3] Andrea Balluchi, Alberto Casagrande, Pieter Collins, Alberto Ferrari, Tiziano Villa, Alberto L. Sangiovanni Vincentelli, et al. Ariadne: A framework for reachability analysis of hybrid automata. In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*, pages 1259–1267, 2006.

[4] Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Kostiantyn Potomkin, and Christian Schilling. JuliaReach: A toolbox for set-based reachability. In *Proceedings of the ACM International Conference on Hybrid Systems: Computation and Control*, pages 39–44, 2019.

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

[6] Akash Harapanahalli, Saber Jafarpour, and Samuel Coogan. immrax: A parallelizable and differentiable toolbox for interval analysis and mixed monotone reachability in JAX. *IFAC-PapersOnLine*, 58(11):75–80, 2024.

[7] Taylor T. Johnson. ARCH-COMP17 repeatability evaluation report. In *ARCH@ CPSWeek*, pages 175–180, 2017.

[8] Taylor T. Johnson. ARCH-COMP23 repeatability evaluation report. In *Proceedings of 10th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 96, pages 189–195, 2023.

[9] Ben Wooding, Viacheslav Horbanov, and Abolfazl Lavaei. Protect: Parallelized construction of safety barrier certificates for nonlinear polynomial systems. In *Theoretical Aspects of Computing – ICTAC*, pages 448 – 458, 2025.

[10] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems*, 31, 2018.