

Off-the-Shelf Cooperative Software Verification by Splitting Programs Dynamically

Cedric Richter ¹, Marek Chalupa ², Marie-Christine Jakobs ³, and Heike Wehrheim ⁴

Abstract: Cooperative software verification aims at improving the effectiveness and efficiency of software verification by enabling multiple verification tools to work together. Existing cooperative verification schemes, however, typically require specialized interfaces or tight integration, limiting their practical applicability. In our International Conference on Software Engineering (ICSE) 2025 publication, we introduce *dynamic program splitting* – a novel approach that enables cooperation between *off-the-shelf* verification tools without requiring any modifications. The key idea is to probe verification tools on a given verification task under a time limit and split tasks dynamically that cannot be verified under the given time constraints. The process continues recursively, incrementally identifying program parts that are particularly hard for the given verifier. The remaining unverified parts are provided to a second verifier, allowing to leverage the complementary strengths of different tools. Our evaluation on SV-COMP benchmarks demonstrates that cooperation through dynamic program splitting enables the verification of tasks that none of the individual verifiers can solve on their own.

Keywords: Software Verification, Cooperation, Program Decomposition

Overview. Software verifiers have become increasingly effective at checking the correctness of software with respect to a given specification. However, no single software verifier consistently outperforms all others across all verification tasks. Existing verifiers exhibit different – and often complementary [BS25] – strengths and weaknesses. This observation motivates the idea of cooperation, where multiple verification tools collaborate to solve a common verification problem. Existing approaches to cooperation, however, often require specialized interfaces for communication or are tightly integrated into a single framework. Such constraints limit cooperation to a small subset of tools that support this kind of interaction. We thus present *dynamic program splitting* (DPS) [Ri25] as a way to enable cooperation between arbitrary *off-the-shelf* software verification tools.

The key idea of DPS is to dynamically apply program splitting on demand whenever the verifier considers the current verification task difficult to verify. Difficulty is decided by a (user defined) time limit on the verification process. Splitting is performed on the programs' execution tree by splitting the program at branching locations. This results into subprograms (subtasks) that each encode a subset of all feasible execution traces. DPS is applied recursively on subtasks until (a) a specification violation is found, (b) all subtasks

¹ University of Luxembourg, Luxembourg, cedric.richter@uni.lu,  <https://orcid.org/0000-0003-2906-6508>

² Institute of Science and Technology Austria, Klosterneuburg, Austria, mchalupa@ist.ac.at,  <https://orcid.org/0000-0003-1132-5516>

³ LMU Munich, Munich, Germany, m.jakobs@lmu.de,  <https://orcid.org/0000-0002-5890-4673>

⁴ Carl von Ossietzky Universität Oldenburg, Oldenburg, Germany, heike.wehrheim@uol.de,  <https://orcid.org/0000-0002-2385-7512>

are verified to fulfill the specification, or (c) some user-defined splitting limit is reached. In the case of (c) that the verifier cannot verify the task completely, the remaining subtasks are *merged* together into a single task. The resulting *residual* task, summarizing the progress of the first verifier, can be seamlessly provided to a second verifier, enabling effective cooperation between the two. As all inputs in the process are programs, cooperation between off-the-shelf verification tools is directly achieved. Our evaluation on SV-COMP benchmarks shows that DPS can enable effective cooperation between off-the-shelf verification tools, sometimes outperforming more integrated cooperation techniques.

Relevance and Impact. Software verification is an important but also very challenging software engineering problem. It is important because we rely on the correctness of software systems and it is challenging as the problem is in general undecidable. To address the challenges of verification, many software verification tools with different strengths and weaknesses have been developed. Cooperative verification aims to combine these strengths, but so far no cooperative approach has been proposed that can fully utilize all publicly available off-the-shelf verification tools. By enabling arbitrary tools to cooperate on single verification task via DPS, we therefore see our contribution as a significant step towards unleashing the full potential of cooperative verification. Because of the generality of our approach, we also see potential for adoption in other software engineering domains, e.g. testing.

Data Availability. All our implementations and data are archived and available at Zenodo⁵ and are evaluated as an artifact in the ICSE 2025 artifact evaluation. Furthermore, our approach is integrated into the software verifier BUBAAK [CR25] which we submitted as part of our contribution to SV-COMP 2025.

References

- [BS25] Beyer, D.; Strejček, J.: Improvements in software verification and witness validation: SV-COMP 2025. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, pp. 151–186, 2025.
- [CR25] Chalupa, M.; Richter, C.: BUBAAK: Dynamic Cooperative Verification: (Competition Contribution). In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, pp. 212–216, 2025.
- [Ri25] Richter, C. et al.: Cooperative Software Verification via Dynamic Program Splitting. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE). IEEE Computer Society, pp. 629–629, 2025.

⁵ <https://doi.org/10.5281/zenodo.14859021>