International Journal on Software Tools for Technology Transfer
https://doi.org/10.1007/s10009-026-00865-x

GENERAL

Special Issue CSV 2025

Challenges of Software Verification (CSV'25)

Luca Olivieri[1] · Vincenzo Arceri[2] · Luca Negrini[1] · Gianluca Caiazza[1]

Accepted: 8 May 2026

Abstract
Given the increasing complexity and expanding frontiers of software systems during the last century, verification has become essential to ensuring reliability, security, and trustworthiness. The goal of the *Challenges of Software Verification Symposium* (CSV) is to monitor the state-of-the-art in this field, exploring challenges to this scientific discipline. This special issue of Software Tools for Technology Transfer presents novel theoretical directions and practical applications of these techniques. The papers in this special issue are extended versions of selected symposium contributions from the proceedings of the 4th Challenges of Software Verification Symposium, which took place at the Ca' Foscari University of Venice, Venice, Italy, from June 5th and 6th, 2025.

Keywords Sofware verification · System verification · Formal methods · Software engineering

1 The history of the CSV symposium

The *Challenges of Software Verification Symposium* has become a yearly meeting point for researchers in formal methods, static analysis, and software verification. The series was inaugurated in 2022 on the occasion of the ceremony awarding Prof. Patrick Cousot a PhD in Computer Science Honoris Causa by Ca' Foscari University of Venice. The first edition[1] featured 15 invited short talks, and extended versions of selected contributions were subsequently published by *Springer Nature* [1].

The following editions further consolidated the format and scientific scope of the symposium. The second[2] and third[3] editions, held in 2023 and 2024, featured 18 and 21 invited speakers, respectively, and resulted in Special Sections of the journal *Software Tools for Technology Transfer* collecting extended and revised versions of selected contributions [2, 5]. Over time, CSV has established a balanced program spanning foundational advances and practical applications in software verification.

The fourth edition[4] of the Symposium was held on June 5th and 6th, 2025, following the format and topics of the previous editions. 23 invited researchers from international institutions presented their work at the Symposium, engaging in full talks (25 minutes each), and this special issue of the journal *Software Tools for Technology Transfer* collects revised and extended versions of 6 such talks.

2 This special issue

The guest editors invited a selection of the speakers of CSV 2025 to submit extended versions of their presentations, which were all peer-reviewed in a single-blind process.

The papers selected for final publication not only present advances in the field of software verification [3, 4, 6–9], but also address fundamental theoretical questions, such as the undecidability of clause reachability in legal contracts [4]; techniques to improve pointer nullness analysis [8]; general abstraction frameworks for sequences, functions, and operators [9]; empirical evaluation of large language models in

[1] https://unive-ssv.github.io/events/2022/05/20/csv.html.
[2] https://unive-ssv.github.io/events/2023/05/25/csv.html.
[3] https://unive-ssv.github.io/events/2024/06/06/csv.html.
[4] https://unive-ssv.github.io/events/2025/06/05/csv.html.

✉ L. Olivieri
luca.olivieri@unive.it
V. Arceri
vincenzo.arceri@unipr.it
L. Negrini
luca.negrini@unive.it
G. Caiazza
pietro.ferrara@unive.it

[1] Ca' Foscari University, Venice, Italy
[2] University of Parma, Parma, Italy

Published online: 18 May 2026

understanding code semantics [6]; novel combinations of large language models and fuzzing for generating weakest preconditions [7]; and challenges arising in the analysis of quantum programs [3].

In the following sections, we provide a summary of each paper featured in this special issue.

2.1 Clause-reachability is undecidable in legal contracts [4]

The authors introduce *Micro-Stipula* a stateful calculus that can be applied to implement legal contracts and where clauses are activated by interactions with the environment or by evaluating time expressions. Then, they show that determining whether a clause is never executed is undecidable, even in restricted fragments such as time-ahead, instantaneous, and determinate. However, they identify a decidable subfragment at the intersection of the instantaneous and determinate fragments, where reachability can be determined.

2.2 Inferring contracts by abstract interpretation with application to pointer nullness analysis [8]

The authors propose a semantic static analysis to infer nullable and non-null pointer annotations in low-level programs. Defined on a minimal imperative language and formulated as a least fixpoint computation over pointer annotations with a sound type-checker, the method can extend beyond nullability. They prove that it guarantees absence of runtime errors with a precise type-checker, and any detected errors are real with a sound checker. Implemented in Codex, a static analyzer for C and binary code, the evaluation shows inferred annotations outperform manual ones on challenging programs.

2.3 Abstractions of sequences, functions and operators [9]

The authors describe a *domain abstraction* mechanism, which formalises the abstraction of a symbolic semantic equation by a numerical functional equation, also paves the way towards applications to dimensionality reduction. Then, they introduce $\mathfrak{B}$-*bound domains* – constraint-based abstract domains using predefined boundary functions (e.g. polynomials, exponentials) – enabling the inference of *non-linear numerical invariants*. In particular, they provide theoretical constructions and illustrative examples, laying the groundwork for future implementations.

2.4 LLMs and fuzzing in tandem: a new approach to automatically generating weakest preconditions [6]

The authors combine Large Language Models (LLMs) with fuzz testing to generate weakest preconditions, introducing *Fuzzing Guidance* to steer the language models using execution feedback. The weakest precondition of a program defines the largest set of initial states from which all terminating executions satisfy a given postcondition. Generating weakest preconditions is crucial for applications such as program verification and run-time error checking. *Fuzzing Guidance* uses fuzz testing to approximate the validity of candidate weakest preconditions and refines the language model output accordingly. Experiments on deterministic Java array programs show that language models can produce viable weakest preconditions, and *Fuzzing Guidance* significantly improves their accuracy.

2.5 Understanding code semantics: a benchmark study of LLMs [7]

The authors study LLMs ability to detect semantically equivalent and inequivalent programs, that is, whether they produce the same output for the same input. They apply semantics-preserving transformations such as copy propagation and constant folding to several Python functions and evaluate seven state-of-the-art LLMs, including ChatGPT, Claude, Gemini, and DeepSeek, under zero-shot prompting with and without minimal context. Despite strong code-generation skills, models misclassify 41% of equivalent cases without context and 29% with context. The authors argue that improving LLMs themselves, through targeted fine-tuning, contrastive learning on equivalent and nonequivalent implementations, or training on transformation-invariant code, will be necessary for robust semantic understanding. Meanwhile, practitioners can achieve better results by selecting stronger models, carefully engineering prompts, or writing code with tools that normalize low-level differences before inference.

2.6 Challenges in quantum programs analysis [3]

The authors provide a survey on static analysis approaches for quantum programs, which have been proposed in the literature, distinguishing between dataflow-oriented approaches, which are based on a graph representation of the program information flow, and domain-oriented approaches, which consist of the definition of abstract domains representing the program property to be analysed. To illustrate these two perspectives concretely, they also present in detail two specific analyses: a dataflow analysis for managing quantum variables and uncomputation, and a static analysis based on abstract interpretation for detecting state entanglement.

Acknowledgements We thank all the authors for their contributions, the organizers of CSV 2025 for their work in making the event possible, and the referees who reviewed the extended versions of the papers that appear in this special issue.

Funding information Work partially supported by SERICS (PE00000014 – CUP H73C22000890001) under the NRRP MUR program funded by the EU – NGEU, and by iNEST – Interconnected NordEst Innovation Ecosystem funded by PNRR (Mission 4.2, Investment 49 1.5) NextGeneration EU (ECS_00000043 – CUP H43C22000540006).

References

1. Arceri, V., Cortesi, A., Ferrara, P., Olliaro, M., et al.: Challenges of Software Verification. Springer, Berlin (2023). https://doi.org/10.1007/978-981-19-9601-6
2. Arceri, V., Negrini, L., Olivieri, L., Ferrara, P.: Challenges of software verification. Int. J. Softw. Tools Technol. Transf. **26**(6), 669–672 (2024). https://doi.org/10.1007/s10009-024-00778-7
3. Assolini, N., Di Pierro, A., Mastroeni, I.: Challenges in quantum programs analysis. Int. J. Softw. Tools Technol. Transf. (2026, in this issue). https://doi.org/10.1007/s10009-026-00845-1
4. Delzanno, G., Laneve, C., Sangnier, A., Zavattaro, G.: Clause-reachability is undecidable in legal contracts. Int. J. Softw. Tools Technol. Transf. (2026, in this issue). https://doi.org/10.1007/s10009-026-00841-5
5. Ferrara, P., Arceri, V., Cortesi, A.: Challenges of software verification: the past, the present, the future. Int. J. Softw. Tools Technol. Transf. **26**(4), 421–430 (2024). https://doi.org/10.1007/s10009-024-00765-y
6. King, D., Koutavas, V., Kovacs, L.: Llms and fuzzing in tandem: a new approach to automatically generating weakest preconditions. Int. J. Softw. Tools Technol. Transf. (2026, in this issue). https://doi.org/10.1007/s10009-026-00844-2
7. Laneve, C., Spanò, A., Ressi, D., Rossi, S., Michele, B.: Understanding code semantics: a benchmark study of llms. Int. J. Softw. Tools Technol. Transf. (2026, in this issue). https://doi.org/10.1007/s10009-026-00842-4
8. Robert, P., Lemerre, M., Sighireanu, M.: Inferring contracts by abstract interpretation with application to pointer nullness analysis. Int. J. Softw. Tools Technol. Transf. (2026, in this issue). https://doi.org/10.1007/s10009-026-00846-0
9. Rustenholz, L., Lopez-Garcia, P., Hermenegildo, M.V.: Abstractions of sequences, functions and operators. Int. J. Softw. Tools Technol. Transf. (2026, in this issue). https://doi.org/10.1007/s10009-026-00843-3