# SV-COMP 2025
## 14th Competition on Software Verification

**Dirk Beyer and Jan Strejček**
(Competition Chairs)

2025-05-05, Hamilton

Report in Proc. TACAS 2025, doi:10.1007/978-3-031-90660-2_9

Supported by:



# Motivation - Goals

1. Community suffers from unreproducible results
   → Establish set of benchmarks
2. Publicity for tools that are available
   → Provide state-of-the-art overview
3. Support the development of verification tools
   → Give credits and visibility to developers
4. Establish and develop standards
   → Specification language, Property definitions,
   Benchmark definitions, Witness formats, Validation process

# Schedule of Sessions at ETAPS

**Session 1:**

- Competition report by organizers
- System presentations
- Short discussion

**Session 2:**

- Open jury meeting, community discussion, moderated by organizers

# Procedure – Time Line

Three Steps – Three Mile Stones:

- Benchmark submission deadline
- System submission
- Notification of results (approved by teams)

The mile stones are further supported by several deadlines, such as the benchmark freezing, tool submission for training, ...

# Verification Problem

Input:

- C program (GNU/ANSI C standard) and property
  → Reachability safety
  → No overflow
  → Memory safety (valid-deref, valid-free, valid-memtack)
  → Memory cleanup
  → Termination
  → No Data race
- or Java program and property
  → Assertion validity
  → No runtime exception

Output:

- TRUE + correctness witness (property holds)
- FALSE + violation witness (property does not hold)
- UNKNOWN (failed to compute result)

# Validation Problem

Input:

- C program (GNU/ANSI C standard)
- property
- correctness or violation witness

Output:

- TRUE = correctness witness confirmed / violation witness refuted
- FALSE = correctness witness refuted / violation witness confirmed
- UNKNOWN = failed to decide

# Environment

Machines (1000 $ consumer machines):

- CPU: 3.4 GHz 64-bit Quad-Core CPU
- RAM: 33 GB
- OS: GNU/Linux (Ubuntu 24.04)

Resource limits for **verification**:

- 15 GB memory
- 15 min CPU time on 4 processing units

Resource limits for **validation**:

- 7 GB memory
- 15 min CPU time on 2 processing units (correctness)
- 1.5 min CPU time on 2 processing units (violation)

# Scoring Schema

Common principles: Ranking measure should be

- easy to understand
- reproducible
- computable in isolation for one tool for verification track

SV-COMP:

- Ranking measure reflects the quality of verification work
- Expressed by a community-agreed score
- Tie-breaker is CPU time

For the validation track, the verdicts of the witnesses are based on voting, because we cannot afford the manual effort necessary to establish the ground truth for thousands of generated witnesses.

# Scoring Schema for Verification Track (2025, unchanged)

| Reported result | Points | Description |
|---|---|---|
| FALSE correct | $+1$ | Error found and confirmed |
| FALSE incorrect | $-16$ | False alarm (imprecise analysis) |
| TRUE correct | $+2$ | Proof found and confirmed |
| TRUE incorrect | $-32$ | Missed bug (unsound analysis) |
| UNKNOWN | $0$ | Failure, out of resources, ... |

# Scoring Schema for Validation Track (2025, unchanged)

| Reported result | Points | Description |
|---|---|---|
| **on correctness witnesses** | | |
| FALSE correct | $+1$ | Witness was correctly refuted |
| FALSE incorrect | $-16$ | Witness was refuted but it is correct |
| TRUE correct | $+2$ | Witness was correctly confirmed |
| TRUE incorrect | $-32$ | Witness was confirmed but it is incorrect |
| **on violation witnesses** | | |
| FALSE correct | $+1$ | Witness was correctly confirmed |
| FALSE incorrect | $-16$ | Witness was confirmed but it is incorrect |
| TRUE correct | $+2$ | Witness was correctly refuted |
| TRUE incorrect | $-32$ | Witness was refuted but it is correct |

# Fair and Transparent

Jury:

- Team: one member of each participating candidate
- Term: one year (until next participants are determined)

Systems:

- All systems are openly available at Zenodo
- Essential information available in FM-Tools repository
- Configurations and Setup in GitLab repository
  $\rightarrow$ Integrity and reproducibility guaranteed

# 80 Competition Candidates in 2025

Qualification:

- 62 in verification track
- 18 in validation track
- One person can participate with different tools
- One tool can participate with several configurations (frameworks, no tool-name inflation)

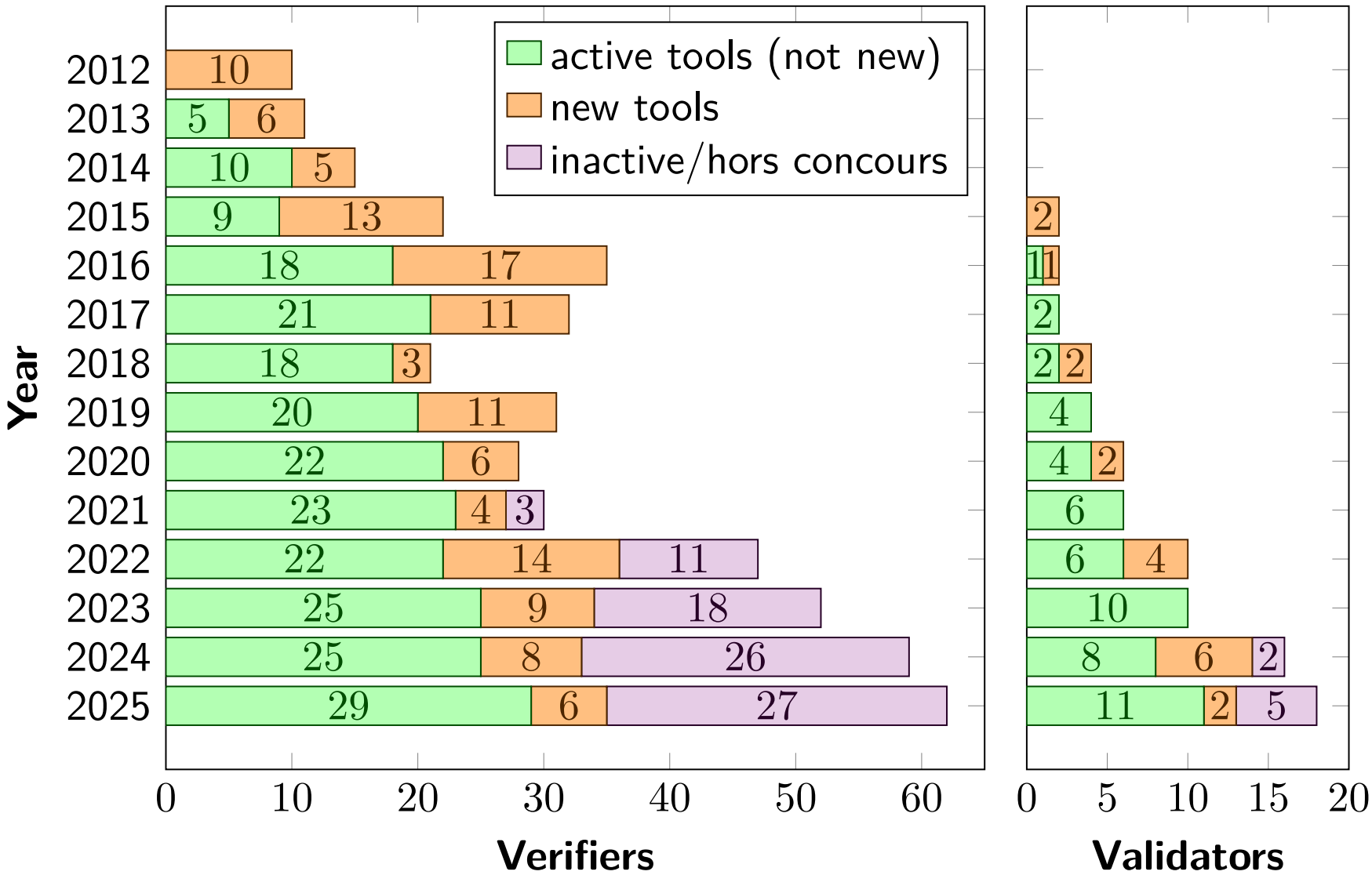
Benchmark quality:

- Community effort, documented on GitLab

Role of organizer:

- Just service: Advice, Technical Help, Executing Runs, Evaluation

# Number of Participants



# Benchmark Sets

- Everybody can submit benchmarks (conditions apply)
- Eight meta categories when closed (scores normalized):
  - ReachSafety: 11 268 tasks
  - MemSafety: 4 042 tasks
  - ConcurrencySafety: 3 175 tasks
  - NoOverflows: 8 211 tasks
  - Termination: 2 328 tasks
  - SoftwareSystems: 4 329 tasks
  - Overall: 33 353 tasks
  - JavaOverall: 673 tasks

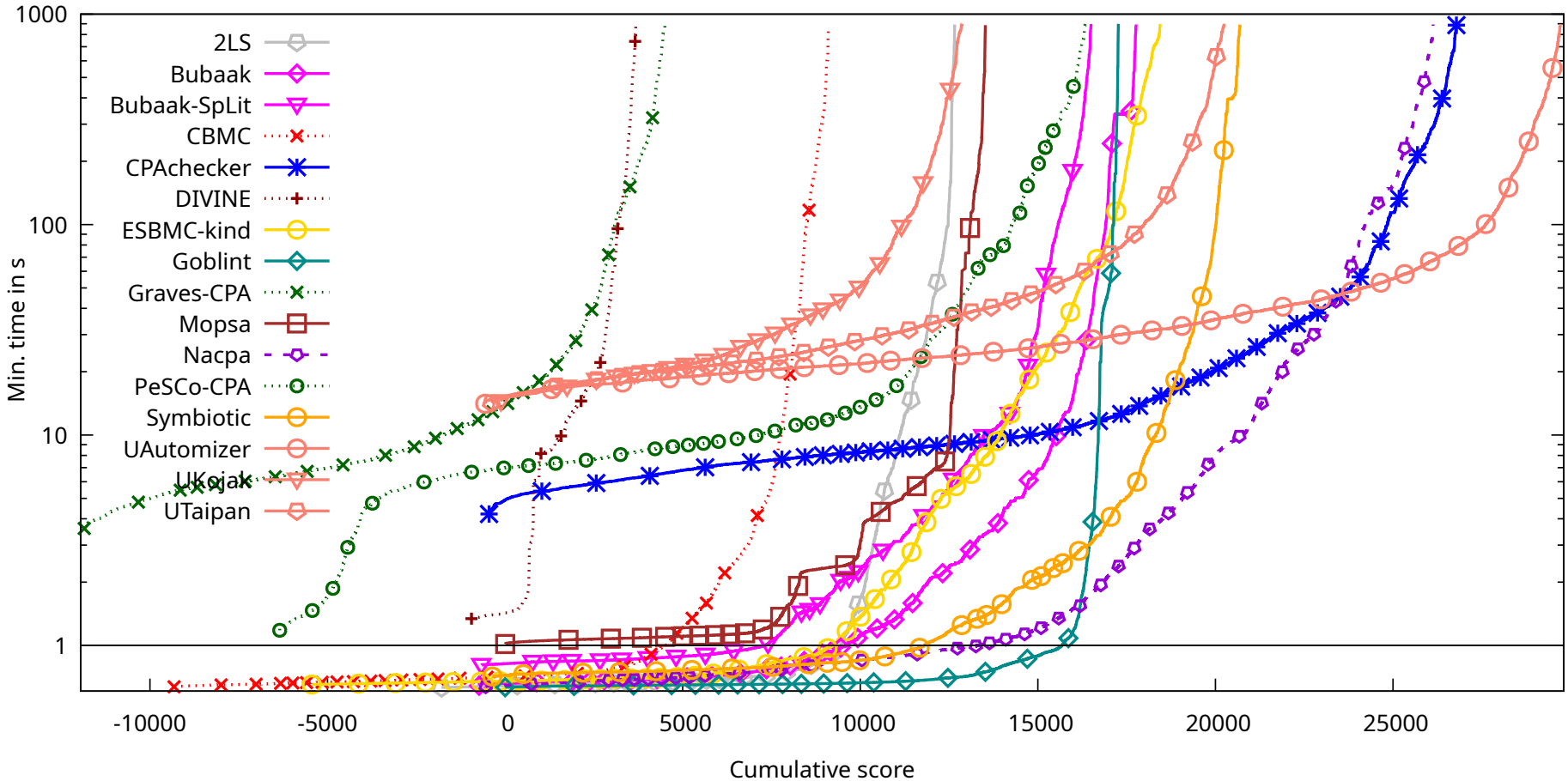
# Reproducibility

- SV-Benchmarks:
  https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks
- SV-COMP Setup:
  https://gitlab.com/sosy-lab/sv-comp/bench-defs
- Resource Measurement and Process Control:
  https://github.com/sosy-lab/benchexec
- Archives:
  https://gitlab.com/sosy-lab/benchmarking/fm-tools
- Witnesses:
  https://doi.org/10.5281/zenodo.15012077

Computation Effort:

- 942 284 verification runs (2 312 days of CPU time),
  pre-runs: 3.3 million verification runs (17 years of CPU time)
- 21.8 million validation runs (2 573 days of CPU time),
  pre-runs: 88 million validation runs

# Results – Example: Overall



# Impact / Achievements

- Large benchmark set of verification tasks
  → established and used in many papers
  for experimental evaluation
- Good overview over state-of-the art
  → covers model checking and program analysis
- Participants have an archived track record
  of their achievements
- Infrastructure and technology for
  controlling the benchmark runs (cf. StarExec)

[Competition Report and System Descriptions
are archived in Proceedings of TACAS 2025]

# New Development in 2025

- Organization committee
- More verification tasks (in each meta category)
- New Java property: no runtime exceptions (demo)
- Handcrafted witnesses in validation track
- New base categories (most prominently Intel-TDX-Module)
- Witnesses in format 2.0 also for violation witnesses
- Split hors concours into *inactive* and *meta verifiers*
- Void tasks and empty categories excluded from score computation
- Medals only for positive scores
- Sponsorship program with Huawei

# Better Support of Witness Format 2.0 by Validators

| | Witness Format 1.0 | | Witness Format 2.0 | |
|---|---|---|---|---|
| **Validator** | Correctness | Violation | Correctness | Violation |
| ConcurrentW2T | | ✓ | | |
| CPAchecker | ✓ | ✓ | ✓ | ✓ |
| CPA-w2t[⌀] | | ✓ | | |
| CProver-w2t[⌀] | | ✓ | | |
| Dartagnan | | ✓ | | |
| Goblint | | | ✓ | |
| GWIT[⌀] | | ✓ | | |
| JCWIT[⌀] | ✓ | | | |
| LIV | ✓ | | ✓ | |
| MetaVa | ✓ | ✓ | ✓ | ✓ |
| MetaVal++[new] | | | ✓ | |
| Mopsa | | | ✓ | |
| NITWIT[⌀] | | ✓ | | |
| Symbiotic-Witch | | ✓ | | |
| UAutomizer | ✓ | ✓ | ✓ | ✓ |
| UReferee | ✓ | | ✓ | |
| Wit4Java | | ✓ | | |
| Witch | | | | ✓ |

# Voting of Validators: Violation Witnesses 1.0

| | | | | | |
|---|---|---|---|---|---|
| | witnesses format | 1.0 | | correct | 34% |
| | validators | 11 | (2 for Java) | wrong | 6% |
| | witnesses | 125214 | | undecided | 59% |

| refutations \ witness confirmations | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 6 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 14 | 5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 164 | 72 | 19 | 63 | 2 | 0 | 0 | 0 | 0 |
| 3 | 2940 | 1933 | 3463 | 235 | 132 | 13 | 0 | 0 | 0 |
| 2 | 2839 | 7481 | 7394 | 1873 | 430 | 320 | 180 | 0 | 0 |
| 1 | 6922 | 15420 | 20894 | 5640 | 5406 | 4079 | 677 | 164 | 0 |
| 0 | 2198 | 7188 | 6866 | 5944 | 4131 | 7429 | 1496 | 847 | 339 |

# Voting of Validators: Violation Witnesses 2.0

| | | | |
|---|---|---|---|
| witnesses format | 2.0 | correct | 28% |
| validators | 4 | wrong | 1% |
| witnesses | 29819 | undecided | 71% |

| refutations \ witness confirmations | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 4 | 21 | 0 | 0 | 0 | 0 |
| 3 | 31 | 4 | 0 | 0 | 0 |
| 2 | 248 | 2006 | 81 | 0 | 0 |
| 1 | 2794 | 8740 | 1159 | 605 | 0 |
| 0 | 2789 | 3686 | 1864 | 4997 | 794 |

# Voting of Validators: Correctness Witnesses 1.0 and 2.0

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | witnesses format | | | 1.0 | | | correct | 52% | |
| | validators | | | 6 | (1 for Java) | | wrong | 0% | |
| | witnesses | | | 195918 | | | undecided | 48% | |

| refutations | witness confirmations | | | | | |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 104 | 14 | 16 | 0 | 0 | 0 |
| 1 | 864 | 1105 | 663 | 67 | 35 | 0 |
| 0 | 31340 | 60818 | 64778 | 22927 | 11039 | 2148 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | witnesses format | | | 2.0 | | | correct | 71% | |
| | validators | | | 8 | | | wrong | 0% | |
| | witnesses | | | 87147 | | | undecided | 29% | |

| refutations | witness confirmations | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 530 | 604 | 643 | 159 | 7 | 6 | 0 | 0 | 0 |
| 0 | 8497 | 15118 | 19540 | 18854 | 17546 | 4014 | 1261 | 284 | 84 |

# Planned Changes for 2026

- Simpler tool registration and qualification process
- Smoke tests via FM-Weck
- Benchmark-category renaming and property renaming
- *FalsificationOverall* will include also termination benchmarks
- New *True-Overall* category (counterpart of *FalsificationOverall*)
- Refinement of the termination property
  - no-cycle, bounded-recursion, no-blocking,...
- No assumption that memory allocation always succeeds
- Allow un-preprocessed C programs
- Reintroduce wall time track as a demo category
- New rules for AI-based tools
- Instant score results (during preruns)
- Instant (but incomplete) validation results (preruns)

# Planned Changes for 2026 (cont.)

- Witness format 2.1:
  - termination and non-termination witnesses
  - concurrency support
  - function contracts
- No support of correctness witnesses in format 1.0 (except for Java)
- Lower time limit for validation of correctness witnesses
- No weighting between wrong/correct validation tasks in validation track

# Sponsorship

New sponsorship agreement with Huawei!

- Travel support
- Demo category with awards
- Hardware support
- Student assistants

Huawei will contribute more industrial benchmark programs, will define a demo category on those, and assign prices.

# Thanks to:

- TACAS (PC Chairs + TACAS SC, thanks!)
- Organization committee
- Competition jury/program committee
- Participants from community (111 people)
- Sponsors: Huawei and LMU Munich
- Next we celebrate the winners

**Report**: