TrueMinutes for the SV-COMP / Test-Comp 2025 Community Meetings

https://competition.zulipchat.com/join/xlatiyku2wpwj6lpe5fpjdnx/ (doesn't work any more)
https://competition.zulipchat.com/

Meeting room (can be entered any time):
https://lmu-munich.zoom-x.de/j/91030644678?pwd=aGFjTUV0YVo3Y1JvSTBaOUVCS2tSZz09
Passcode: 961308

Meeting Minutes for last year:
https://docs.google.com/document/d/1pyh9Ln_a8AZGr1CSmzPee6MkiHK0fBhb7Kd4ZzOyOEk/edit

Full Timeline for SV-COMP (AoE): https://sv-comp.sosy-lab.org/2025/dates.php
(See below, was agreed in the meeting 2024-09-24)

TODO Items Status

- Dirk (2024-10-08.9 - updating Contributing.md)
- Jan (2024-10-29.6.c - writing rules for validation track)
- Dirk (2024-11-05.8 - add second test-suite using gcc and clang)
- Dirk (2024-11-05.9 - rules page out of date)
- Dirk (2024-11-05.10.b - disqualification requests)
- Dirk (2024-11-05.10.c - disqualification requests)
- Dirk (2024-11-05.14.a - update the scripts to validate violation witnesses for no-data-race)
- Dirk (2024-11-12.6 - Setup PCs Test-Comp and on the webpage)
- Dirk (2024-11-12.7 - mark freezed benchmarks as "svcomp25-freeze")
- Jan (2024-11-12.8 - Add the tables present in 2024-11-05.12 into the rules in an appropriate place)
- Dirk (2024-11-12.9 - merge the MR with a more precise formulation of valid-memtrack)
- Dirk (2024-11-12.10 - add link to Zulip to the web)
- Soha (2024-11-26.11 - check the format of GraphML witnesses for Java)

2025-05-08 16:00 - 18:00 GMT-5 (Test-Comp Community Meeting at ETAPS)

Participants: Dirk Beyer, Jan Strejcek, Matin Jonáš, Marek Chalupa, Ravindra Metta, Peter Schrammel, Edoardo Manino, Márk Somorjai, Nikolai Kosmatov
Note Takers: Jan
Agenda (please add your name to your topic):

- We would like to present total coverage achieved by all generated tests for each benchmark. It would need more executions of TestCov.

- We would like to have better/more test coverage measurement tools (one is on the way).
- Why don't we use some existing industrial tools measuring test suite quality, for example based on mutation testing. Mutation testing can be done efficiently (all mutants are encoded into one binary). It is used in practice. It can be executed as another category/ranking.
- Should we use verification tools for test generation (via some generic transformations/wrapper)? Short and inconclusive discussion.
- Time limits: should they be changed? Keep it as it is.
- Change graphs illustrating the results of Test-Comp? Jan suggests to make one axis the benchmark number (in the order of achieved coverage) and the other the achieved coverage. Martin suggests to show achieved coverage based on time limit (similarly as we do in SV-COMP). This would need to store the time when each test was emitted.

2025-05-05 14:00 - 18:00 GMT-5 (Community Meeting at ETAPS)

Participants: Dirk Beyer, Marian Lingsch-Rosenfeld, Jan Strejcek, Marek Jankola, Zsófia Ádám, Levente Bajczi, Gidon Ernst, Po-Chun Chien, Marie-Christine Jakobs, Nils Lommen (In total 16 participants, please add your name)
Note Takers: Marian Lingsch-Rosenfled
Agenda (please add your name to your topic):

1. Who will be taking notes?
2. Discussion about the proposed changes for SV-COMP26
    - There was no opposition to any of the changes
    - Small discussion about if the benchmark definition xml's need to be in the Git repository
        - The conclusion was to move this change to 2027
    - Small discussion about the use of AI in verifiers
        - Some clarification for the result of the discussion in Frauenchiemsee was required, but the same decision was rectified
    - There was a renewed question if Falsification Overall should be kept
        - There were 6 people in favor of it and 0 against it
    - There was a small discussion about if True Overall should be incorporated
        - If there is Falsification Overall is included then True Overall is included
        - There are tools which do not produce counterexamples, which would benefit from it
        - **Conclusion**: It will be implemented as decided at the Frauenchiemsee meeting
3. Correctness witness format 1.0 for Java verifiers
    - It would be good, if there was a format for this, but it is unclear if the Java verification community will do it
4. Report from the SV-COMP Meeting at Frauenchiemsee (Dirk, Marian)
    - Done during point 2
5. Reproduction Report/Parallel Execution (Levi, w/ Zsofia, Zoltan)
    - Re-run of presentation at Frauenchiemsee, with some minor advancements
    - Presentation was done, there was only a few questions about it
6. Virtual best verifier (homework from Frauenchiemsee) (Levi)

    - https://home.mit.bme.hu/~bajczi/virtual-best.table.html
      https://home.mit.bme.hu/~bajczi/virtual-best.diff.html
      (some ideas, will be updated)
    - **Conclusion**: The discussion is delegated to the next time

2025-04-01 9:00 - 18:00 CET (Community Meeting at Frauenchiemsee)

Participants: Dirk Beyer, Marian Lingsch-Rosenfeld, Jan Strejcek, Marek Jankola, Simmo Saan, Michael Schwarz, Daniel Dietsch, Frank Schüssele, Matthias Heizmann, Paulína Ayaziová, Henrik Wachowitz, Karoliine Holter, Zsófia Ádám, Levente Bajczi, Raphaël Monat, Charles Moloney, Jonathan Craig, Malte Mues, Gidon Ernst, Dominik Klumpp, Christian Schulze, Max Barth, Bruno Farias, Christoph Hof, Matthias Kranz
Note Takers: Raphaël Monat, Gidon Ernst (11:00-12:00), Thomas Lemberger, Zsófia Ádám (13:00 - 18:00)
Agenda (please add your name to your topic):

category structures, scores, ranking, rules

1. Presentation by Thomas: Intel-TDX new benchmarks, added in SoftwareSystems Category
    - Verification tasks are created from multiple C files and a slicing process. A harness is created with some stubs/shadow data, e.g., nondet hooks are resolved at that time
    - Challenges for instrumentation/harness creation: assembly, external variables, multiple files, dead code
    - Challenges for verifiers: bit-precise, huge structs leading to lots of code to initialize them
    - Question for discussion: custom annotations for initializing complex types in SV-Comp?
        - Problem: how to initialize pointers in structs? _nondet_pointer was removed from the rules due to unclear semantics.
        - What about arrays with symbolic sizes? It would not be relevant here (array of predefined size)
        - Pragmatic solution: initialize pointers to NULL, let other initialization harness complete this later.
        - Header files could provide implementation to make it runnable
        - Action item: create a merge request to the rules [Thomas Lemberger & Nian-Ze Lee]
        - Daniel from Ultimate team is also interested
    - Question: what to do with tasks that are too difficult for verifiers?
        - [no discussion]
2. Smoke test with FM-Weck (Henrik, Dirk)
    - History: automated CI checks for Coveriteam 2 years ago.
    - Goal: make sure tools are running correctly in a specified container
    - Proposal: add `smoke_test` shell script at root of tool archive, to run different tests ensuring that the tool seems to be working correctly in the various settings required for the SV-Comp. Clarify that it should *not* be Python etc. Allow empty script (ie, `exit 0`)
    - Action item: merge request to rules to require this [Henrik Wachowitz]

- fm-weck “package manager” for fm-tools repository. Then the smoke tests could be run in container image provided through fm-weck.
- Benefits:
    + Documented tool usage inside the archive (even if the documentation on the web disappears)
    + Reproducible way to test artifact
- Talk receives positive feedback :)
- Note: docker image specification is already part of fm-tools entries, will be used in the future to run tools in SV-COMP
- Question: Can we use fm-weck inside containers? Need containers that allows nested containers.

3. Reproduction Report/Parallel Execution (Levi, w/ Zsofia, Zoltan)
    - Goals: Evaluate reproducibility, “numerical” stability on non-LMU machines, third-party understanding of SV-Comp process/scripts.
    - Reproduced on Hungarian HP (184 nodes, 2xAMD EPYC 7763 (64-core), 256GB RAM). Ran under three weeks to reproduce results of active participants. 24% less walltime, 32% less cputime, 11% more memory.
    - 495828 tasks. 7.74% of mismatching verdicts: half known reproducibility problems, half resource limits (up to marginal flipped results for mlb/swat)
    - Data is available on home.mit.bme.hu/~bajczi/sv-comp-repro-25
    - Good news: all medals confirmed for tools with no known problems
    -
    - Action item: investigate flipped results [?]
    - Action item: Merge Request for rules page to specify Ubuntu LTS version,
    - Action item: enforce package declaration test via CI. Possibly enforce check against base image, not full image. Related to smoke test [Henrik]
    - Action item?: document assumptions on infrastructure to reproduce (e.g. whether overlay file systems are needed)
    - Action item: document assumption that working dir must be tool dir (check in CI/smoke tests?), also which dirs are readwrite and which are writeonly, aim to avoid overlay-fs requirement. /tmp is needed. Clarify that the tool directory is fresh for each execution. Fix something related to Ultimate, too [Henrik, Daniel]
    - Action item: add more tool metadata (cgroups access, containerization, …)
    - Levi is happy to prepare the merge requests
    -
    - Known problems:
        + Missing packages (15 tools, against rules). Maybe issue with stripped down vs default version for ubuntu? https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/534
        + Assumed read-write access. Cluster overlays did not support it. https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/535
        + Resultfiles pattern mismatch (against rules). But not discovered on benchcloud. https://gitlab.com/sosy-lab/software/benchcloud/-/issues/704
        + Working dir must be tool directory. discovered due to some too flexible interpretation of patterns by benchcloud https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/536

        + Missing tool metadata
          https://gitlab.com/sosy-lab/benchmarking/fm-tools/-/issues/20
          (probably will need a rule change once it is finalized what to allow)

1. Cyberagentur introduces themselves
    - Advertise funding for projects in the area of verification
    - Invites informal chat or write an informal E-Mail kranz@cyberagentur.de hof@cyberagentur.de

4. remove FalsificationOverall? Or use all tasks, including termination and introduce VerificationOverall? [Dirk]
    - => Rule is not well known
    - Counts `false` answers from tools across all categories; same normalization by all tasks. Goal: count wrong alarm: false, event with label: true
    - Termination doesn't count in the category (for historic reasons).
    - Discussion: keep category? Include termination tasks? Drop category?
        - Problem: tools that always say `true` would theoretically win this category, even though false alarms
        - Normalization requires to include tasks with false alarms
    - Conclusion:
        - keep the category, but include termination
    - Do we want a category, where only true results count (mirroring falsification): True-Overall
        - Proposed earlier
        - Arguments against:
            - scoring already was good for tools giving true results;
            - adds complexity
        - Arguments pro:
            - inconsistent to only have FalsificationOverall but not TrueOverall
        - Also possible: alternative ranking (this was present at some point, but was dropped, why?)
            - We had: "most correct", "most energy-efficient", "best new tool"
        - **Decision 1: We add category True-Overall and rename FalsificationOverall to False-Overall. Both are full, non-demo categories.**
        - **Decision 2: We keep the way the category False-Overall is assembled (all tasks where tools produce verdict 'false')...**
        - **Decision 3: … but add previously missing termination tasks to False-Overall**
        - **Decision 4: We reintroduce categories with alternative rankings "most-correct", "most energy-efficient", "best new tool" on the results webpage**
    - Accessibility and usability of competition results on the webpage: Data is difficult to access at the moment (Zenodo artifact with strange file names)
        - Would be good to implement/have a way of easy and fast access with filtering and visualization
        - **Decision: The community is open to make data more easily accessible, but there's no volunteer.**

5. **[DONE]** Change names of Java properties

- **Announcement:** We rename Java property files to be consistent with C property files

6. Should we change the names of base categories (drop the prefixes?) Dirk proposes to call base categories <language>.<property>.<program_group>
   - What happens if the file is in more categories? - same as now, we are renaming categories, not the file
   - Example: concurrency-something is common (e.g. ConcurrencySafety-Main), but there is NoDataRace-Main
   - This would not affect 2nd level categories
   - . vs -
   - **Decision 1: We rename the categories to <language>.<property>.<program-group>. We use . instead of - to make new names unambiguous when program-group contains hyphens (e.g. 'C.unreach-call.SoftwareSystems-coreutils' vs. the ambiguous 'C-unreach-call-SoftwareSystems-coreutils')**
   - **Decision 2: Thomas Lemberger implements this [DONE]**
7. **[DONE]** Should Overall be called COverall or C-Overall (the same for FalsificationOverall)?
   - **Decision: C.Overall , C.True-Overall, C.False-Overall**
8. Move from reachSafety to assertionSafety?
   - Proposal:
     1. Rename category name to 'AssertionSafety'
     2. Actually use 'asserts' in programs
   - Needs investigation of existing asserts: https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/issues/1411
   - What kind of assert? assert.h, C assert, ACSL, etc. - answer: probably C assert
     i. actual implementation is not defined in the C standard
   - Standard assert is macro — related to (un-)preprocessing of tasks
   - Advantage of the proposal: We think that external people don't know 'reach_error', makes tasks more difficult to understand than if we just used 'assert(...)' or '__assert_fail(...)'
   - Issues with the proposal:
     i. In pre-processed files, asserts are not there anymore
     ii. The assert(...) of assert.h can have side effects. This is a more difficult property to check than the current 'reach_error()'.
     iii. There are lots of sv-benchmark programs with assert in them. These are currently not considered for unreach-call, but this would change.
   - Insight from industry: Industry people don't care about name of the function
   - **Decision: postpone this change**
9. Proposal: New subcategory: Termination concurrent (!1363) (Frank)
   - What property should be checked? What about deadlocks / livelocks?
     i. Simmo: general non-termination should be about *all* sources of non-termination.
     But we could *also* have a better-scoped no-deadlock property (Exists but unused in Java track: https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/java/properties/no-deadlock.prp?ref_type=heads).

Analogous to valid-memtrack being an approximation of valid-memcleanup.

ii. Hernan (will not attend the meeting): we should also consider livelocks, e.g. spinning conditions waiting for notification from other threads. I have many benchmarks related to this that I can add

- Should we consider any execution or only fair ones?
  i. You have to make some kind of fairness assumption (e.g., nothing would ever terminate if scheduler does nothing)
  ii. If deadlock is non-terminating, what is the counterexample? + even more corner cases with concurrency
  iii. Property has to be specified more precisely
- **Decision: Frank Schüssele creates proposal for property as MR to the rules repository (complementing this, see the existing MR for tasks (draft): https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/merge_requests/1363)**

10. Treat termination as a metaproperty (similarly to memsafety): if violated either infinite_run or unbounded_stack or unbounded_memory
    - Proposal: Add new sub-properties to property 'termination' so that verifiers have to tell which of the three reasons it is: infinite_run, unbounded_stack or unbounded_memory.
    - Idea about this proposal: In practice, there cannot be a non-terminating program, because it runs out of resources at some point. The verifier should recognize which of these resources get exhausted by the non-terminating run and tell the user.
    - Overarching question: Can we categorize sub-properties of termination that differentiate different kinds of non-termination?
    - Discussion about 'unbounded_memory' and whether this is actually possible.
      i. Who would consider non-terminating: loop, allocates some memory in each iteration
         1. You run out of pointers
         2. But non-terminating, although what will it do when running out of pointers is undefined behaviour? Or drop the assumption that malloc always succeeds?
         3. As a user something like "this program is allocating memory in an uncontrolled manner" is probably more useful than getting non-termination?
         4. Is it just 2 properties being violated? What is the 2nd one? "Unbounded allocations"
    - *In case we go for Proposal #9:* What about deadlock?
    - **Decision: Jan Strejcek is responsible. Discuss in a smaller group with people interested in termination [Jan, Gidon, Daniel,Levi,Zsófi, ?]**
      i. Group that works on non-termination witnesses may have to restrict witnesses to witnesses about integers, excluding non-termination that involves memory
11. Remove the assumption that memory allocation always succeeds? Or make it optional and explicit? (unsuccessful allocation can lead to execution of another code that contains a bug)

- Concrete proposal: Make the behavior of malloc explicit for each verification task, by adding this data to the YAML task definition (similar to language, machine model).
- Idea about the proposal:
  i. ‘malloc always succeeds’ is not realistic.
  ii. But adjusting all of the verification tasks that assume malloc always succeeds can not be fixed easily
  iii. Users assume that verifiers check the actual semantics of the code, not have implicit assumptions about the behavior -> leads to false negatives.
- Alternative proposal: Always assume the actual semantics of malloc and let’s try to fix all of the existing tasks.
  i. Run verifiers good in memory-safety to figure out how many tasks are affected if we assume that malloc can fail for all tasks.
  ii. Just add if (pointer == null) abort(); everywhere
- **Decision: alloc, calloc, malloc, etc. get their original semantics. Thomas Lemberger prepares a MR.**

12. What information is missing from the website but linked here: https://gitlab.com/sosy-lab/sv-comp/sv-comp-2024 (Dirk, Marian)
    - **Decision: Marian Lingsch moves links at https://gitlab.com/sosy-lab/sv-comp/sv-comp-2024 to SV-COMP webpage**
13. Allowing un-preprocessed C programs (Simmo)
    - Overview and proposal: SV-COMP un-preprocessed programs
    - Hernan (will not attend the meeting): this will make porting deprecated __VERIFIER_atomic to real C code (c11 atomics, pthread mutexes, etc) much easier.
    - Potential issues:
      i. Which header and compiler versions to use?
      Answer: Current preprocessed files contain only declarations for functions, and their behavior is defined in the C standard (should be same for all headers)
      ii. Verifiers have to support more macros when they want to work on the actual C file
    - Potential advantages:
      i. Less maintenance burden on .c and .i files
      ii. Verifiers can decide to inject their own definitions of functions to the standard library, if this helps them
      iii. Easier to read tasks (usability)
    - **Decision: Simmo Saan prepares the proposal in SV-COMP un-preprocessed programs as a MR to the rules:** https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/537
14. Extend __VERIFIER_nondet_X to __VERIFIER_nondet_object(&o)?
15. Reintroduce walltime track (Thomas)
    - Walltime in benchcloud?
    - Smaller task set
    - What if a sequential tool wins? Bigger problem: if each sequential tools participates, that means a lot of wasted resources
    - Can meta-verifiers compete as normal tools here?

- **Decision: demo track**

16. The State of Benchmarks in SV-COMP: Production of new verification tasks for the SV-COMP format (Charles, Jon)
    - Intro, why we are here: we have a tool intending to gather and transform Java files into a suitable testing format for events such as SV-COMP
        i. As a proof of concept, we have created 20 tests in an MR seen here
    - Are the existing SV-COMP benchmarks fair? Too easy/hard?
    - Should we be focusing on specific categories? As of now Java is all one group
        i. How can this be extended - languages and categories of verification tasks?
    - Are these a good representation of real-world tests/problems?
    - **There will be a survey on the Google Group, please participate**
17. Should we make a separate category structure for Verification and Validation track?
    - **Decision: postpone to next year**
18. Suggest rules for AI-based tools (to prevent learning of the expected results)
    - Discussion and opinions:
        i. Goals of the competition: Compare algorithms?
        ii. No, compare verifiers.
        iii. SV-COMP is about evaluating algorithms and data-structures for symbolic reasoning about programs. Maybe the use of LLMs and their effectiveness on symbolic reasoning tasks can not be well evaluated on sv-benchmarks.
        iv. We do not want to discourage the use of AI in the competition, but we want to avoid overfitting.
        v. Just letting all AI-based tools participate hors concours is not a solution, because the result presentation still includes them without paying respect to strong overfitting. This produces a wrong impression of the power of AI tools compared to tools based on symbolic reasoning.
        vi. The goal of the competition should be to check techniques for program analysis that generalize beyond the verification tasks of sv-benchmarks. We can assume that this is true for algorithms based on symbolic reasoning, but not for LLMs.
    - Ideas:
        i. Prepare randomly selected training set (split the benchmarks)
            1. Trust that they do not train on these
            2. Or try to reproduce the training
        ii. Disallow to use the expected results or tool verdicts for training
        iii. There must be a declaration of AI use that declares what training data was used
        iv. Add “fresh”, hidden tasks that are not included in the preruns, but submitted by the participants
        v. Restrict training time of LLMs
        vi. Add diverse benchmark tasks to avoid overfitting
        vii. Use obfuscation on verification tasks to find differences in performance based on variable/function names
        viii. Negate conditions in assertions. This *should* toggle the expected verdict and LLMs *may* not realize

ix. Inject faults at some random location in true-tasks to check whether the tools can detect this or become unsound. Hypothesis: LLMs will keep the same, (now wrong) verdict
x. Forbid training on sv-benchmarks altogether

- **Decision: There must be a declaration of AI use that declares what training data was used. All training data and scripts must be provided. By default, all AI-based tools participate hors concours. They are allowed to persuade the competition jury during the tool-qualification process to be ranked as full participants.**

19. Do we want to show the virtually best verifier in the scores? If yes, how do we define it?
    - Definition of "virtually best verifier": Best result for each task over all participants. Definition of "best": Correct result with the least CPU time consumption.
    - Open questions:
      i. Include meta-verifiers?
         1. Reasons to include meta-verifiers:
            a. Show the actual best in the plots
            b. Show the state of the art of the whole SV-COMP community.
            c. Hypothesis: Meta verifiers are verifiers that do not add any significant analysis techniques. So they should not contribute a lot to the virtually best.
         2. Reason to exclude meta-verifiers: We want the virtually best to show us the potential of the basic techniques and algorithms used. This is a good baseline for meta-verifiers that combine tools.
      ii. Include inactive tools?
         1. Issue with including them: Many inactive tools produce negative scores. Hypothesis: They basically guess, so we don't want to include them in the virtually best.
      iii. For each verifier, show its contribution to the virtually best? (e.g., "Ultimate Automizer provides the best result across all verifiers in 15% of cases")
    - **Decision: Levi computes the following numbers:**
      i. Virtually best including all verifiers
      ii. Virtually best including all verifiers, but only include verifiers that have more than 0 in sum across the categories they participate in
      iii. Virtually best on non-meta verifiers
      iv. Virtually best on non-meta and active verifiers
      v. Virtually best on active verifiers
      vi. Any other combination Levi comes up with

organization committee, timeline, registration

20. who is willing to continue on the committee, who wants to enter? https://sv-comp.sosy-lab.org/2025/organization.php
    - Idea: People from outside LMU Munich could help with infrastructure, to make the team more robust
    - **Decision: Committee stays, but people are always welcome**

21. add qualification team
    - Idea: the current process of reviewers checking checkmarks does not need to be distributed to the full competition jury, being done 3x per tool.
    - Jan Strejček, Paulina Ayaziová, and Raphaël Monat, Malte Mues, Dirk Beyer plan to simplify qualification process
    - Qualification team still needed
22. what should be changed in organization?
    - Please provide suggestions via mail to Dirk or Jan
23. How to declare participation in categories: web form/merge request to existing/new file? (skipped in meeting)
24. Do you prefer to use registration via GoogleForm or via EasyChair? (skipped in meeting)
25. **The Book? (ASV '19)** **https://easychair.org/conferences2/overview?a=21840614**

validation track

26. Handover plaquettes (Dirk) ✔
27. Voting results (Jan) ✔
28. **[DONE]** dropping some witness format (correctness 1.0)?
    - Proposal: Drop all correctness witnesses in version 1.0 (affects unreach-call and no-overflow)
    - Reason: Voting mechanism of validation track shows that correctness witnesses 2.0 are of significantly higher quality, according to agreement of correctness validators
    - Issue: What about inactive tools?
        i. Solutions:
            1. Keep validation for version 1.0 for inactive tools
            2. Don't run inactive tools that don't support version 2.0
            3. Inject trivial correctness witnesses for tools that don't provide version 2.0 (for all tools, inactive and active)
    - **Decision: Drop correctness witnesses version 1.0. The organizers may decide whether to run inactive tools that don't support version 2.0 with trivial witnesses.**
    - **Additional note by Dirk and Jan: For Java, we have only one correctness witness validator and it supports only format 1.0. Hence, we keep correctness witness format 1.0 for Java.**
29. support witnesses in some new categories?
    - Demo tracks for new witnesses?
        i. Categories with existing witnesses: No need for demos, witnesses are immediately allowed and adapted. Old witnesses will still be supported (e.g. violation witnesses for C.termination.*)
        ii. Categories with no existing witnesses: Witnesses are not considered in score computation, but we add demo-tracks in validation competition
        (e.g. correctness witnesses for C.unreach-call.Concurrency)
30. if a validation task is voted incorrectly, override the expected result and keep it in the evaluation?
    - Current process: To remove a task from the score computation of validation track, it is not necessary to provide a fix to the wrongly labeled witness, but

only to convince the community that the witness is actually labeled with a wrong expected verdict.
    - Suggestion: Add fixed witnesses *as handcrafted witnesses* (in a reasonable amount)
    - **Decision: Add correctly labeled, original witnesses to the directory of handcrafted witnesses (in a reasonable amount) so that they can be rejected by validators**
31. **[DONE]** Reduce Timeout for Validation of Correctness witnesses (Dirk, Marian, Jan)
    - Suggestion: 300 s for correctness, keep 90s for violation
        i. Reducing validation of correctness witnesses to 300s reduces the number of confirmed witnesses by only a handful of tasks
    - Raphaël: I would really like to discuss this, following the previous point in the last meetings.
    - **Decision: reduce to 300s.**
32. cancel weighting between wrong/correct validation tasks?
    - **Decision: Everyone agrees**

2024-12-10 16:00 - 17:00 CET (Last meeting for SV-COMP25)

Participants: Simmon Saan, Frank Schüssele, Vesal Vojdani, Nils Lommen, Chris Meudec, Zsófia Adam, Marian Lingsch-Rosenfeld, Raphael Monat
Note Takers: Frank Schüssele, Marian Lingsch-Rosenfeld
Agenda (please add your name to your topic):

1. Introduction of people
2. Who will be taking notes today? Preferably multiple people
3. Pending TODO items (please update the list moved to a separate section above)
4. Report from organizers:
    a. Changes after deadline
        i. List of changes made (I think all covered by precedence): https://competition.zulipchat.com/#narrow/channel/471288-SV-COMP/topic/Organizer.20Report/near/486966804
        ii. LIV change: https://gitlab.com/sosy-lab/benchmarking/fm-tools/-/merge_requests/553 (The bug only occurs for rejected witnesses)
        (a) do not apply (then Goblint does not have validation with LIV, developers seem to be ok with this)
        (b) merge and rerun only Goblint validation
        (c) merge and rerun validation with LIV for all tools
        Organizer asks for advice from the jury/community.
        (I think voting is not necessary, if you can agree, each choice is reasonable.)
            1. **Conclusion**: Option b was chosen
    b. Paper deadline approaching (December 20): if you want to submit a paper, please make a new submission with the prefix “[paper] <toolname>:”
    (Easychair is currently closed for submissions, but will be re-opened in few days.)

i. **TODO (Dirk)**: Send an E-Mail to all Test-Comp 2025 jury members regarding this change, reopen Easychair for submissions. [Jan did it for SV-COMP 2025]

~~Raphaël: this is new, maybe it should be sent to all jury members? Also, easychair currently says that "New submissions are not allowed".~~

5. What should be changed for the next SV-COMP/Test-Comp?
    a. Marian: Do we want to require tools to have a Software Bill of Material (SBOM)?
        i. Could be machine readable and improve the qualification process
        ii. It is a template which makes it easier to look at
        iii. **Conclusion**: Could be interesting, but could make extra work for newcomers
    b. Raphaël: continuing last week's discussion on reducing the computational time of SV-Comp.
        i. Reducing verification time? Deduplicating witnesses?
        ii. Other suggestions from https://competition.zulipchat.com/#narrow/channel/464099-general/topic/SV-COMP.3A.20Verification.20vs.2E.20Validation/near/483985660
        iii. Re-discussing correctness validation time. https://arxiv.org/pdf/2310.16572 measures a ~25% reduction in validation compared to verification. Reduce correctness validation time to 675s first?
        dbeyer: I cannot be at the meeting tomorrow (lecture).
        For the verification: I would be in favor of keeping the verification time as is: 900 s. This is pretty established and used a lot.
        For the correctness-validation time, I would either keep it as is (900 s) or reduce considerably, e.g., to 300 s. A mild reduction does not have a scientific meaning to me. A significant reduction would send the signal that we have the goal to reduce the validation time (and that 'true' witnesses are not good witnesses). [**Jan**: I support a significant reduction to 300 or 450 s]
            1. For some categories like Software systems it is difficult to increase the performance of validators using only Invariants. Exclude them as arrays/heaps?
            2. For Goblint there are only 27 tasks which are validated by Goblint after 450 seconds (out of 7246)
            3. Could we have a data-driven approach to check that actual validators (doing more than verification) would not be impacted? Or which time limit would be interesting?
            4. **Conclusion**: There is interest in decreasing the time-limit but the concrete time-limit should be determined using data aided by excluding categories where witnesses do not increase validator performance
6. Does the community want another meeting next week at 10:00?
    a. **Conclusion**: There is no meeting next week
7. Marian: Should the validation track only use witnesses which were created by active verifiers?
    a. All witnesses are benchmarks for the validation track and usually having more benchmarks is good

    b. Inactive verifiers do not affect the scoring in the verification track maybe they should also not affect the scoring in the validation track
    c. **Conclusion:** Remove on a per tool basis, for example inactive verifiers that produce trivial witnesses should be excluded
8. Raphaël: official results date?
9. Will the community be in Canada?
    a. A hybrid meeting may be undesired
    b. Maybe we can use Zulip to coordinate specific meetings on certain topics preparing the meeting on ETAPS
    c. ~~**TODO (Marian)**~~ **DONE**: Talk to Dirk about this such that he sends an E-Mail with the official announcement

2024-12-03 10:00 - 11:00 CET

Participants: Dirk Beyer, Frank Schüssele, Chris Meudec, Simmo Saan, Raphaël Monat, Marian Lingsch-Rosenfeld, Levente Bajczi, Martin Jonáš, Marcel Ebbinghaus, Tomáš Dacík, Manuel Bentele, Paulína Ayaziová, Martin Blicha, Dominik Klumpp
Note Takers: Marian Lingsch-Rosenfeld
Agenda (please add your name to your topic):

1. Introduction of people
2. Who will be taking notes today? Preferably multiple people
3. Pending TODO items: all of them
4. Report from organizers
    a. The final runs for SV-COMP are currently running
    b. Once the final runs are done, there will be 48 hours on working days to submit changes to SV-Benchmarks before announcing the final results
5. Jan: What should be changed for the next SV-COMP/Test-Comp?
    a. Marian: Do we want to deprecate Witnesses V1 for the categories for which V2 witnesses exist for SV-COMP26?
        i. Advantages:
            1. remove redundancies from pre-runs and final runs,
            2. V2 is better and should be used
            3. Users should get V2 witness and no confusion with the versions
        ii. **Conclusion**: The version of witnesses (n + 1) is run on SV-COMP (X) officially in the competition alongside witness version (n) then in SV-COMP (X + 1) only witnesses version (n + 1) is used.
        iii. **TODO (Jan)**: Add this decision into the page with the rules for the validation track
    b. Marian: Reduce time limit for validation of correctness witnesses?
    Raphaël: Why?
        - **Advantages:**
            - Less time consumption of pre-runs and final-runs
            - (added after the meeting: Pushes witnesses to provide useful information)
        - **Disadvantages**:
            - Makes it more difficult to integrate new tools into SV-COMP
            - May reduce score of most scalable tools.

- **Conclusion**: We will reduce it to some number between 90 and 450 depending on the results, depending on data from correctness validation and quantile plots

c. Dirk: Aggregate validation results also for pre-runs and make available to developers and jury. Provide up-to-date witness store more timely.
d. Chris: Reduce time limit for Test-Comp tests generation to move towards more useful tools for developers.
    i. **Advantages**:
        1. Usually the tools find something quite quickly or not at all
    ii. **Disadvantages**:
        1. Fuzzers usually want higher time-limits
    iii. **Conclusion**: Discuss further
e. Raphaël (but mentioned previously by Simmo, Hernan): un-preprocessed files? https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/489#note_2207944643
    i. **Advantages**:
        1. It is difficult to keep some tasks in sync between their original and preprocessed version
    ii. **Disadvantages**:
        1. Changing everything at once is difficult
    iii. **Conclusion**: Will be discussed by interested people and the benchmark quality assurance team in: https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/489#note_2207944643
    iv. **Conclusion**: Should be done incrementally i.e. by allowlisting
f. Marian: Virtually best verifier? SAT does it already: https://satcompetition.github.io/2024/results.html
    i. It is unclear how to define the Virtual Best Verifier, since a verifier which always returns true and one which always returns false would produce an unachievable virtual best verifier (idea: include only positive-scoring tools or tools that have less than 1% wrong results)
    ii. **TODO (Levente):** Will try to write a script which generates the XML files for the virtually best verifier based on one of the metrics given above
    DONE (Levente): First pass script done (takes a folder of results, and outputs another folder with virtual-best selection for each (sub)category), improvements and ideas welcome: https://gitlab.com/sosy-lab/benchmarking/competition-scripts/-/merge_requests/139

6. Wit4Java: MR https://gitlab.com/sosy-lab/benchmarking/fm-tools/-/merge_requests/535
7. Jan: The book? Dirk: Instructions for camera-ready will be send out in the next two weeks, with a short deadline, which is interpreted as soft deadline.
8. Dominik: final results already sent out?
    a. For tools which are finished the final results have been sent, for UAutomizer: https://sv-comp.sosy-lab.org/2025/results/results-verified/uautomizer.results.SV-COMP25.table.html

2024-11-26 16:00 - 16:48 CET

Participants: Henrik Wachowitz, Manuel Bentele, Nils Lommen, Frank Schüssele, Simmo Saan, Jan Strejček, Raphaël Monat, Dominik Klumpp, Marian Lingsch-Rosenfeld, Levente Bajczi, Tomáš Dacík, Thomas Lemberger
Note Takers: Marian Lingsch-Rosenfeld, Jan Strejček
Agenda (please add your name to your topic):

1. Introduction of people
2. Who will be taking notes today? Preferably multiple people
3. Jan: Test-Comp
    a. **TODO (Dirk)**: Send an E-Mail regarding the timeline for Test-Comp and update the website to reflect the changes
4. Marian: Till when will we have SV-COMP meetings?
    a. The last meeting is scheduled for December 10
5. Dirk: After deadline requests:
    a. ~~**TODO**~~ **DONE (Jan, Dirk):** ~~set up a voting for jury~~ we use the precedent used also in the last year: technical fixes are accepted, jury is only informed (and can protest).
    b. MLB submission after deadline: proposal ~~reject~~ accept (wrong java path) https://gitlab.com/sosy-lab/benchmarking/fm-tools/-/merge_requests/520
    c. LIV submission after deadline: proposal accept (technical problems) https://gitlab.com/sosy-lab/benchmarking/fm-tools/-/merge_requests/511
    d. Benchmark definitions (already merged, not subject of voting): https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/507
    e. SWAT resubmission: (proposal accept) https://gitlab.com/sosy-lab/benchmarking/fm-tools/-/merge_requests/524
6. Jan: how can meta-verifiers participate? We have meta-verifiers that would like to participate competitively (e.g., nacpa, pesco).
    a. **Conclusion**: let them compete, mark them “meta-verifiers” and they cannot get any medal (basically as “hors concours”, but with a different label). The labels “hors concours” and “meta-verifier” can be combined in future.
    ~~b. Suggestion 2: If there is only one actively participating meta-verifier, we can let it compete with other verifiers and say in the report that this was an exception (to not set up a precedent for meta-verifiers competing in future).~~
7. Jan and Paulína: what about setting the re-submission deadline for validators a few days later than for verifiers? (Otherwise, the witnesses are still changing.)
    a. **Conclusion**: solve ad-hoc (if needed), consider to change the schedule for the next year
8. Marian: What happens with witnesses whose syntax is accepted by WitnessLint but rejected by tools? What happens if the witness is provided by a Verifier? What happens if it is part of the witness benchmarks?
    a. Dirk: The usual procedure: touch the task to exclude it
    b. **Conclusion** For the hand-crafted witnesses: There should be no benchmarks with witnesses which do not conform to the format
    c. **Conclusion** for the witnesses produced by verifiers: if there is a syntax error in the witnesses then WitnessLint is adjusted to reject these witnesses (and re-run on the tool to disqualify those witnesses (actually re-run on all tools as this change can disqualify some witnesses by other tools as well)
9. Merge Fuzzle into Recursive (sv-benchmarks!1599)?

    a. **Conclusion**: Let's merge it after SV-COMP 2025 and before SV-COMP 2026
10. Chris: confirmation of TestComp new deadlines. Zulip message suggests Nov. 25, 2024 Notifications and tool reviews; list of qualified competition candidates published and 2024 Nov. 29, 2024 Third tool deadline: Competition candidates can resubmit a new version of the verifier (evaluation phase starts)

Appendix: Citation of minutes from 2022-11-08

- Definition for Meta-Verifier (19:37-21:08)
  Proposal from meeting on 2022-11-01 is refined to: A meta verifier is a combination of at least two existing verification components such that each result produced by the combination can be computed by some of its components alone. A verifier is the result of research and engineering in verification algorithms and approaches, while a typical meta verifier selects a verification component to run, sets up its parameters, and potentially post-processes its output.

2024-11-19 10:00 - 11:00 CET

Participants: Chris Meudec, Frank Schüssele, Simmo Saan, Marian Lingsch-Rosenfeld, Henrik Wachowitz, Manuel Bentele, Nils Lommen, Po-Chun Chien, Tomáš Dacík, Vesal Vojdani, Jan Strejček, Kaled Alshmrany, Paulína Ayaziová, Raphaël Monat, Marcel Ebbinghaus, Soha Hussein
Note Takers: Marian Lingsch-Rosenfeld, Jan Strejček
Agenda (please add your name to your topic):

1. Introduction of people
2. Who will be taking notes today? Preferably multiple people
3. Pending TODO items:
    a. Dirk (2024-10-08.9 - updating Contributing.md)
    b. Jan (2024-10-29.6.c - writing rules for validation track)
    ~~c. Jan (2024-11-05.12.f - checking the status of Witness Format 2.0)~~
    d. Dirk (2024-11-05.8 - add second test-suite using gcc and clang)
    e. Dirk (2024-11-05.9 - rules page out of date)
    f. Dirk (2024-11-05.10.b - disqualification requests)
    g. Dirk (2024-11-05.10.c - disqualification requests)
    h. Dirk (2024-11-05.14.a - update the scripts to validate violation witnesses for no-data-race)
    i. Dirk (2024-11-12.6 - Setup PCs Test-Comp and on the webpage)
    j. Dirk (2024-11-12.7 - mark freezed benchmarks as "svcomp25-freeze")
    k. Jan (2024-11-12.8 - Add the tables present in 2024-11-05.12 into the rules in an appropriate place)
    l. Dirk (2024-11-12.9 - merge the MR with a more precise formulation of valid-memtrack)
    m. Dirk (2024-11-12.10 - add link to Zulip to the web)
~~4. Nian-Ze: update the Intel TDX tasks after the freezing deadline (MR 1591)~~
    a. We will prepare a precise fix that only updates affected tasks so unaffected ones can remain in the competition this year
    b. Affected tasks: https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/issues/1424#note_2213183654
5. Meeting minutes are not online, do we want to add them to the webpage?
    a. **Conclusion**: Will not be added, since it should not be public information

6. Jan: please deliver the reviews
7. Marian: Will there be a deadline extension?
    a. **Conclusion**: There will be a deadline extension of 2 - 3 days for the final version of tools for Test-Comp
8. Jan: How to handle violation witnesses of valid-memtrack as we have changed the description of the bug only a few days before the final deadline? See: https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/471
    a. **Conclusion**: use the version in the MR, proceed as usual
9. Jan: The **book**? (Dirk: no progress since last week)
10. Kaled: Can FuSeBMC team participate only with one tool? Sure it can. (and a sequence of other questions followed, not written down)
11. Soha: Wit4java handles witnesses by GDart differently (its witnesses contain assumptions with method calls). Unfortunately, the GraphML witness format does not allow function calls in assumptions: *Valid values:* Expression that (1) evaluates to a value of the equivalent of a Boolean type in the programming language of the verification task and (2) may consist of conjunctions or disjunctions, but **not function calls**, see https://gitlab.com/sosy-lab/benchmarking/sv-witnesses/-/blob/main/doc/README-GraphML.md.
    a. **TODO (Soha)**: Check the format description and suggest a solution (probably some modification of the format)

2024-11-12 16:00 - 16:45 CET

Participants: Chris Meudec, Frank Schüssele, Manuel Bentele, Dominik Klumpp, Levente Bajczi, Marian Lingsch-Rosenfeld, Malte Paul Mues, Matthias Heizmann, Simmo Saan, Vesal Vojdani, Raphaël Monat, Tomáš Dacík, Jan Strejček
Note Takers: Marian Lingsch-Rosenfeld, Jan Strejček
Agenda (please add your name to your topic):

1. Introduction of people (skipped)
2. Who will be taking notes today? Preferably multiple people
3. Pending TODO items:
    a. Dirk (2024-10-08.9 - updating Contributing.md)
    b. Jan (2024-10-29.6.c - writing rules for validation track)
    ~~c. Jan (2024-11-05.12.f - checking the status of Witness Format 2.0)~~
    d. Dirk (2024-11-05.8 - add second test-suite using gcc and clang)
    e. Dirk (2024-11-05.9 - rules page out of date)
    f. Dirk (2024-11-05.10.b - disqualification requests)
    g. Dirk (2024-11-05.10.c - disqualification requests)
    h. Dirk (2024-11-05.14.a - update the scripts to validate violation witnesses for no-data-race)
4. (Moved from one week ago) Marian and Jan: Use of witnesses in verification and validation tracks
    a. ~~**TODO**~~ **DONE (Simmo)**: Create an MR in SV-Witnesses which describes the result of the discussion about the semantics of Witnesses (See: https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/452#note_2169739276): https://gitlab.com/sosy-lab/benchmarking/sv-witnesses/-/merge_requests/85

5. Simmo: “false” vs “false(no-overflow)” — violated subproperty name is needed for valid-memsafety, but also for others? Seems to affect scoring if verifier and validator have differing output.
    a. ~~**TODO**~~ **DONE (Simmo)**: Open Issue: https://gitlab.com/sosy-lab/benchmarking/competition-scripts/-/issues/37
6. Raphaël: jury assignment in Easychair?
    a. **TODO (Dirk, Jan)**: Set up PCs (SV-COMP and Test-Comp) in EasyChair and ask for reviews
7. **TODO (Dirk)**: mark freezed benchmarks as “svcomp25-freeze”
8. **TODO (Jan)**: Add the tables present in 2024-11-05.12 into the rules in an appropriate place
9. **TODO (Dirk)**: merge the MR with a more precise formulation of valid-memtrack, see: https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/471
10. **TODO (Dirk)**: add link to zulip to the web

2024-11-05 10:00 - 11:34 CET

Participants: Daniel Baier, Raphaël Monat (until 11:05), Jan Strejček, Dirk Beyer, Nils Lommen, Simmo Saan, Marian Lingsch-Rosenfeld, Hernan, Vesal Vojdani, Tomáš Dacík, Marcel Ebbinghaus, Levente Bajczi, Martin Jonáš, Chris Meudec, Frank Schüssele, Manuel Bentele, Soha Hussein
Note Takers: Marian Lingsch-Rosenfeld, Jan Strejček
Agenda (please add your name to your topic):

1. Introduction of people
2. Who will be taking notes today? Preferably multiple people
3. Pending TODO items:
    a. Dirk (2024-10-08.9 - updating Contributing.md)
    b. Jan (2024-10-29.6.c - writing rules for validation track)
    c. Raphäel (2024-10-29.7.a - prepare MR for SV-COMP 2026)
4. Raphaël: new memcleanup subcategory? (e.g. SoftwareSystems-uthash-MemCleanup) Why is it separate from memsafety?
    a. Summary in https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/issues/160
5. Report by Dirk on Preruns: All requests satisfied, now increasing to 900 s (but only 2 CPUs and 7 GB)
    a. Whenever a new FM-Tools entry is merged a new pre-run is started
    b. To get pre-runs without a new version contact Dirk, this needs to be started manually
6. Tagging the freeze version
    a. What to do with the unmerged MRs tagged SV-COMP 2025?
    Conclusion: Let’s extend the freeze deadline to 5-Nov-2024 AoE. If someone protests within one week, there will be a formal voting about the deadline extension.
    ~~**TODO**~~ **DONE (Dirk)**: change the deadline on web.
7. Daniel: Discuss ~~(and vote on)~~ MR 1540, 1584 and 1583
    a. 1540 adds the subproperties for lots of Juliet task definitions and splits some tasks into 2 tasks if there were multiple subproperties valid.
    b. 1584 adds 86 additional subproperties in tasks added by 1540 that were forgotten. (Only affects task definitions)
    c. 1583 merges some task definitions that only differed by a distinct property but were otherwise equal and also fixes some incorrect task names (those task

names were also fixed inside of the tasks, but did not change any tasks or task names)

8. **TODO** (**Dirk)**: Test-Comp: Add second test-suite validator for Test-Comp (one based on gcc, one based on clang)
9. **TODO** (**Dirk)**: Simmo: Rules page out of date:
    a. Ubuntu version change, no overflow specification improvement and atomic rule changes were merged
    b. ~~Still open~~ (merged on Nov 10): https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/452
    c. Still open: https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/471
10. Dirk: Discuss how to handle disqualification requests from April for Deagle.
    a. Should we adjust the results to the facts that were found out after they were published?
    b. **Conclusion/TODO (Dirk):** Add to rules: We do not remove the numbers/Tool but put an asterisk which explains the situation
    c. **Conclusion/TODO (Dirk):** Add to rules: In the future if a tool does not conform to the rules in one category they are excluded from all categories. The tool will also be removed from all columns
11. Jan: The **book**? Dirk was at a conference and failed to make progress. ☹
12. Marian and Jan: Use of witnesses in verification and validation tracks
    a. Properties for which correctness witnesses are currently expressive enough to provide useful information for the validation of C programs:

Property	Fully Supported in Witness Version 1.0	Fully Supported in Witness Version 2.0	Run in Verification Track	Run in Validation Track
ReachSafety	x	x	x	x
ReachSafety-Recursive	x (partly)	x (partly)	x	x
ReachSafety-Arrays,-Floats,-Heap				
ConcurrencySafety				
NoDataRace				
MemSafety				
NoOverflows	x	x	x	x
Termination				

    b. Properties for which violation witnesses are currently expressive enough to provide useful information for the validation of C programs:

Property	Fully Supported in Witness Version 1.0	Fully Supported in Witness Version 2.0	Run in Verification Track	Run in Validation Track
ReachSafety	x	x	x	x
ReachSafety-Arrays,-Floats,-Heap	x	x	x	x
ConcurrencySafety (ReachSafety, NoOverflows)	x	-	x (only for witnesses 1.0)	x (only for witnesses 1.0)
ConcurrencySafety (MemSafety)	x	-	x (only for witnesses 1.0)	x (only for witnesses 1.0)
NoDataRace	x	-	x (only for witnesses 1.0)	x (only for witnesses 1.0)
MemSafety - valid-deref,valid-free	x	x	x	x
MemSafety - valid-memtrack, valid-memcleanup	x	-	x (only for witnesses 1.0)	x (only for witnesses 1.0)
NoOverflows	x	x	x	x
Termination	x	-	x (only for witnesses 1.0)	x (only for witnesses 1.0)

   c. The results of the validation track 2024 show that many witnesses for MemSafety (both violation and correctness) were validated. But how many of them are trivial? Validating a trivial witness is the same as verification. Do we want this in the validation track competition?
   d. Suggestions: Maybe we should express in the rules that we need better witnesses?
   e. ~~**TODO**~~ **DONE (Marian, Frank)**: Ask the authors of witness format 1.0 for the properties the format supports and fill it in the table (use “-” for negative answer).
   f. ~~**TODO**~~ **DONE (Jan)**: Fill in the properties currently supported by witness format 2.0.
13. Hernan: use un-preprocessed sources. The topic is at least 4 years old. Many people seem to be in favor. We keep saying “let's discuss it in person in next TACAS” and nothing ever happens. Can we already decide to move forward and agree that this will be implemented for SVCOMP 2026?
    Draft: Define concurrency tasks on un-preprocessed .c sources where available (!1359) · Merge requests · SoSy-Lab / Benchmarking / SV-Benchmarks · GitLab
    a. Simmo: The first step would be to agree on a rules change allowing unpreprocessed programs for SV-COMP 2026 as early as possible (after

2025). Then it's possible to make MRs that change subsets of tasks over/require new tasks to be unpreprocessed.
    b. ~~**TODO (Hernan):**~~ Open MR that will remove from competition rules the request to not have `include` in benchmarks.
14. Hernan: Competition scripts need update now that no-data-race requires violation witnesses [Data race violations with no witness (#36) · Issues · SoSy-Lab / Benchmarking / Competition Scripts · GitLab]
    a. **Conclusion/TODO (?Dirk?):** update the scripts to validate violation witnesses for no-data-race
15. Hernan: How to access witnesses from preruns?
    a. ~~**TODO (Hernan):**~~ send email to Dirk.

2024-10-29 16:00 - 17:00 CET

Participants: Simmo Saan, Chris Meudec, Dominik Klumpp, Zsófia Adam, Frank Schüssele, Marian Lingsch-Rosenfeld, Jan Strejček, Raphael Monat, Daniel Baier, Kaled Alshmrany
Note Takers: Marian Lingsch-Rosenfeld, Jan
Agenda (please add your name to your topic):

1. Introduction of people (Skip)
2. Who will be taking notes today? Preferably multiple people
3. Pending TODO items:
    a. Dirk (2024-10-08.9 - updating Contributing.md))
    b. Marian (2024-10-22.8b)
4. Task freezing deadline on 2024-11-01, **open MRs need reviews**: https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/merge_requests?milestone_title=SV-COMP+2025
    a. Tasks changed after this deadlines are excluded from the results for this year
    b. **Conclusion**: Every MR will get a comment that the deadline is on Friday and people should review
    c. ~~**TODO**~~ **DONE (Jan):** send email
5. Update from Dirk:
   - Merge requests for FM-Tools and Benchmark Definitions almost all done.
   - Preruns can most likely start tomorrow morning.
   - Jury can be extracted from FM-Tools data once all merged (ETA: tonight hopefully).
6. Marian: Update the rules to indicate what types of witnesses should be part of the scoring in the validation track. Witnesses which are not relevant for the verification track or for which no witness format exists should not be evaluated. See also points 15. and 16. Under 2024-04-08 in https://docs.google.com/document/d/1pyh9Ln_a8AZGr1CSmzPee6MkiHK0fBhb7Kd4ZzOyOEk/edit?tab=t.0
    a. **Conclusion**: We should not run benchmarks in the verification track for which no witness format exists. A group of people interested in this will discuss it.
    b. ~~**TODO**~~ **DONE (Jan, Marian)**: Meet to discuss what categories should be run
    c. **TODO (Jan)**: Write rules for validation track.
7. Raphaël: Busybox benchmarks contain stubs for some libc functions (getutent, getopt, getopt_long, read, vasprintf, write). However those stubs are unnecessary and inaccurate. Would anyone oppose removing those stubs?
    a. **TODO (Raphaël)**: prepare MR for SV-COMP 2026

8. ~~Raphaël: Easychair assignment for jury members? dbeyer: Jan or Dirk has to take care.~~ (addressed by 5.)
9. Raphaël: new memcleanup subcategory? (e.g. SoftwareSystems-uthash-MemCleanup) Why is it separate from memsafety?
    a. Should it be merged with SoftwareSystems-uthash-MemSafety?
    b. ~~**TODO**~~ **DONE (Raphaël)**: Check whether there are any technical reasons to keep it separate and if not, prepare a MR to merge the two subcategories. Cf. https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/issues/160
10. Frank: Proposed rule change: Mark __VERIFIER_atomic as deprecated (dbeyer: merged)
11. Marian: Reviewers for: https://github.com/sosy-lab/benchexec/pull/1093

2024-10-22 10:00 - 11:00 CEST

Participants: Philipp Wendler, Simmo Saan, Dominik Klumpp, Marian Lingsch-Rosenfeld, Raphael Monat, Vesal Vojdani, Dirk Beyer, Frank Schüssele, Marcel Ebbinghaus, Hernan, Chris Meudec, Martin Blicha, Malte Paul Mues, Martin Jonáš, Jan Strejček, Paulína Ayaziová, Manuel Bentele, Levente Bajczi
Note Takers: Jan, Levente Bajczi, Marian Lingsch-Rosenfeld
Agenda (please add your name to your topic):

1. Introduction of people
2. Who will be taking notes today? Preferably multiple people
3. Pending TODO items:
   Dirk (2024-10-08.9 (updating Contributing.md))
   Dirk (2024-10-01.9.b (Zulip stream)), **DONE**: https://competition.zulipchat.com/
4. Dirk: Clarify who can and is supposed to merge
    a. Quality Assurance Team should be allowed to Merge (Simmo accepts, Raphael has some reservations but accepts, Frank accepts), rights already given
    b. Changing the category structure should be discussed with chairs
    c. ~~**TODO**~~ **DONE**: Ask Zsófia if she also wants be allowed to Merge. Zsófia thinks that it is not necessary.
    d. **Optional (Team of Mergers):** Document informal merge rules (which cannot be handled by gitlab rules) if needed
5. Task freezing deadline on 2024-11-01, **open MRs need reviews**: https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/merge_requests?milestone_title=SV-COMP+2025
    a. Tasks changed after this deadlines are excluded from the results for this year
    b. 10 Floats task submitted ~15min after deadline: allow if gets reviewed or postpone to SV-COMP 2026? Conclusion: Let's treat them as within the deadline.
6. Marian (Moved from three weeks ago): Witness Benchmark Set for the validation track/competition, see: https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/tree/witness-benchmark-set?ref_type=heads. This would require adjustment for all the tool-info modules for Validators, as reference for CPAchecker there is this PR: https://github.com/sosy-lab/benchexec/pull/1078
    a. There are doubts about maintainability of this witness set.

    b. Jan: let's try with the relatively low number of handcrafted witnesses and we will see.
    c. Will be implemented as new task files (witness validation task)
    d. There was a discussion about naming/placement of these tasks
    e. ~~**TODO**~~ **DONE:** Marian will create a MR for handling handcrafted witnesses (will be stored in the same directory as the input programs, will be discussed internally with Dirk and Philipp)
    f. Tool info modules for validators should be changed to get the witnesses from the yaml files. ~~**TODO**~~ **DONE:** Marian will create a MR for all validators
7. Jan: The **book**? Dirk: no progress in the last weeks.
8. Levente: ~~sv-benchmarks setfile names changed (https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/commit/edbf5c067040778afba1e4334ba6c81ce2bbdc75), but benchmark-defs still use old version, even in (some) new MRs. Will these be changed all at once, or should everyone change it one-by-one?~~
    a. Simmo: https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/447 changed all *existing* ones at once, but MRs will have to fix conflicts/adapt
    b. ~~**TODO**~~ **DONE (Marian):** SV-COMP24 → SV-COMP25 in all rundefinition names as well. MR: https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/481
9. Hernan: when can we expect the first pre-runs?
10. Frank: (see !1559 and !1561) Are calls to undefined non-standard functions (that are still declared) allowed? What if the call is not reachable from main and the caller-function of the undefined function has the inline keyword?
    a. ~~**TODO**~~ **DONE:** Frank will prepare a MR that introduces function definitions with falling assertion inside (maybe with a comment that the body of these functions was not originally available).
11. Jan: Will SV-COMP 2025 run on Ubuntu 22.04 or Ubuntu 24.04?
    a. Rules page says 22.04 in one place.
    b. Raphaël: I interpret rules as saying Ubuntu 24.04… "(b) works on the GNU/Linux platform (more specifically, it must run on an x86_64 machine with the latest Ubuntu LTS)"
    c. SV-COMP 2025 will run on 24.04. The cloud will hopefully be updated by the end of this week.
    d. ~~**TODO**~~ **DONE** (also in Test-Comp): Jan will change rules page (refer to the latest LTS)

2024-10-15 16:00 - 17:00 CEST

Participants: Philipp Wendler, Jan Strejček, Simmo Saan, Frank Schüssele, Nils Lommen, Chris Meudec, Raphael Monat, Vesal Vojdani, Dominik Klumpp, Manuel Bentele, Malte Paul Mues, Matthias Heizmann, Marcel Ebbinghaus
Note Takers: Philipp (and others)
Agenda (please add your name to your topic):

1. Introduction of people
2. Who will be taking notes today? Preferably multiple people
3. Pending TODO items: Dirk (2024-10-01.9.b (Zulip stream)), Dirk (2024-10-08.9 (updating Contributing.md))

4. ~~**TODO**~~ **DONE:** Jan will update the rules of Test-Comp to reflect two versions of TestCov (one compiled with gcc and the other with clang)
5. Jan: Should we reconsider the use of GNU C11 as the standard for all benchmarks? (see Philipp's comment)
    a. Simmo: tried in CI, minor incompatibilities between c11 and gnu11 arise
    b. Simmo: specifying C standard in each task definition (like data_model) can be done in backwards-compatible manner: all existing tool info modules simply will not use this information.
    c. Background: clarification for overflow property necessary (overflows of signed integers as defined in C11)
    d. In principle incompatibilities exist between standard versions, but currently not a problem for tools. So we can wait with labeling each task with its C version until someone pushes for this.
    e. Decision: keep status quo as in previous year, do not merge bench-defs!448 and sv-benchmarks!1550
    ~~**TODO**~~ **DONE**: clarify rules (definition of overflows in bench-defs!449, maybe explain that different tasks can conform to different C versions)
6. Hernan (I won't be able to attend, but please discuss): please merge Rename *.set file names to match PR 1542 (!1553) · Merge requests · SoSy-Lab / Benchmarking / SV-Benchmarks · GitLab to avoid "false alarms" in the CI for concurrency tasks.
    ~~**TODO**~~ **DONE:** Dirk: merge the MR
    ~~**TODO**~~ **DONE:** Dirk clarify the rules, who can/is supposed to merge (Benchmark Quality Assurance members cannot merge for now)
7. Hernan (I won't be able to attend, but please discuss): deadline "Submission deadline for verification tasks" is not clear enough. I assume "new tasks" are not allowed, but we can still open PRs to fix existing benchmarks. What about PRs which (strictly speaking) are not fixes? E.g., Draft: Replace __VERIFIER_atomic by atomic types/operations (!1554) · Merge requests · SoSy-Lab / Benchmarking / SV-Benchmarks · GitLab
    You can open PR all the time around the year. Changes like the mentioned one are not fixes, so the original benchmarks will be used in the competition even if the PR is not merged before benchmark freezing deadline (Nov 1).
8. Raphaël: best way to signal which MRs are ready to merge in sv-benchmarks? (E.g, !1533, and I guess much more will be coming!)
9. Jan: Suggestion for a better definition of valid-memtrack: All allocated memory blocks are **tracked**. The set of tracked blocks is defined as the smallest set of blocks satisfying the following two rules:
    a. A block is tracked whenever there is a pointer to this block (not necessarily pointing to the beginning of the block) or to the first address after this block (add pointer to the standard here) stored in a program variable. The variable can be of a pointer type or of a compound type containing a pointer. The variable does not have to be in the current scope, it can be on the call stack.
    b. If some pointer in a tracked block points to another block (again, not necessarily to the beginning of the block) or to the first address after this block, this pointed block is also tracked.

    ~~**TODO**~~ **DONE:** Jan: turn this into MR.

2024-10-08 10:00 - 11:00 CEST

Participants: Dirk Beyer, Jan Strejček, Dominik Klumpp, Frank Schüssele, Marian Lingsch-Rosenfeld, Chris Meudec, raphael Monat, Hernan Ponce de Leon, Paulina Ayaziová, Simmo Saan, Manuel Bentele
Note Takers: Raphaël, Marian Lingsch-Rosenfeld
Agenda (please add your name to your topic):

1. ?Introduction of people?
2. Who will be taking notes today? Preferably multiple people
3. Pending TODO from last meeting: Jan (2024-10-01.4.f), Dirk (2024-10-01.9.b (Zulip stream))
4. Raphaël: easychair is not open/linked but deadline is in a week?
    a. https://easychair.org/conferences/?conf=svcomp2025 (linked on Submission page)
    dbeyer: Should work since Oct 2
    b. ~~**TODO**~~ **(DONE):** for Dirk: Add link to submission page on dates page
5. Marian (Follow up on last week): Currently we are using C11 for the tasks. This generates two related questions. See discussion in MR: https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/448
    a. Do we want to update to C23
    dbeyer: I am also in favor of sticking to C11
    Raphaël: I would be in favor of keeping C11 this year, to stabilize the whole sv-benchmarks repo.
    Marian: Currently the CI of SV-Benchmarks considers multiple different C standards as pointed out by Simmo: https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/merge_requests/448#note_2142610023
    b. Do we want to specify the standard for which we are verifying a task in the task-definition files?

    => All CI should use GNU C11, and discuss in the plenary meeting if C standard should be file defined in the yml. C23 is too recent?
    ~~**TODO**~~ **(DONE):** Simmo will try it and report if breakages are created.
6. Marian (Moved from last week): Witness Benchmark Set for the validation track/competition, see: https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/tree/witness-benchmark-set?ref_type=heads. This would require adjustment for all the tool-info modules for Validators, as reference for CPAchecker there is this PR: https://github.com/sosy-lab/benchexec/pull/1078
    => Postponed
7. Simmo (Copied from last week due to no decision/progress): no-overflow definition: are casts signed overflows?
    a. **Contradictory expected verdicts**: tools cannot avoid wrong verdicts (unless they answer “unknown” for most tasks…)
    b. Why is no-overflow property only about *signed* overflows? Because signed overflows are UB.
    c. Casts to signed types are not UB (C11 6.3.1.3.3):
    “Otherwise, the new type is signed and the value cannot be represented in it; either the result is *implementation-defined* or an implementation-defined signal is raised.”
    d. Industry standard: CWE-681: Incorrect Conversion between Numeric Types is separate from (i.e. not ChildOf) CWE-190: Integer Overflow or Wraparound.

e. **Option 1:** (casts don't violate no-overflow)
    i. Revert ldv-regression/test_overflow verdict change.
    ii. Clarify rules that no-overflow property is only concerned with UB.

   Raphaël: in favor of option 1. Nobody in favor of option 2.
f. **Option 2:** (casts violate no-overflow)
    i. Fix 623+ tasks where they currently don't. (Who is going to do this?)
    ii. Clarify rules that no-overflow property covers signed overflow UB and cast-to-signed overflow IB. (Where else is such a definition used?)
    iii. Require ~22 tools to change their current behavior.
g. If no-overflow is no longer about signed overflow UB, then why would we exclude *unsigned* overflows? They are defined behavior but also covered by CWE-190 (*wraparound!*). (But this would be an even bigger change…)

=> **option 1 chosen**; Simmo will make a MR.

8. Simmo (Moved from last week): last year it was decided that NoDataRace will require *violation* witnesses this year, the easiest way would be to use GraphML violation witnesses with concurrency for this.
   => Indeed, Dartagnan supports validation of NoDataRace violation witnesses in the GraphML format. Tools should generate these witnesses to get points.
9. Hernan (Moved from last week): Usual runners do not work for external contributors of tasks (see Introduce c/libvsync concurrency benchmarks (!1548) · Merge requests · SoSy-Lab / Benchmarking / SV-Benchmarks · GitLab).
   => Document this in a place which is easier to find than an Issue. Ideally in Contributing.md
   TODO: Dirk adds this information to the Contributing.md file in SV-Benchmarks
10. Jan (Moved from last week): **The book?**
    => Dirk is currently testing the Style and will contact the Authors regarding this. Horizon is "in a couple of days"
11. Jan: Issue in Test-Comp: The C standard does not specify the order of evaluation of subexpressions or function parameters. Hence, if the benchmark contains foo(__verifier_nondet_int(),__verifier_nondet_int()), then some test generators generate inputs in one order and some generators in the opposite order, but TestCov assumes only one of these orders (it is based on gcc, therefore it uses the evaluation order of gcc, but e.g. clang uses the opposite order). What should we do about it? Identify and remove/change benchmarks where the evaluation order changes the order of reading the input? Or give the information about the expected evaluation order to tests? Or develop another tool measuring test coverage and consider the max value of this tool and TestCov? Maybe TestCov compiled with clang can do the job. **TODO**: discuss this further.

2024-10-01 16:00 - 17:00 CEST

Participants: Dirk Beyer, Jan Strejcek, Dominik Klumpp, Frank Schüssele, Manuel Bentele, Marcel Ebbinghaus, Chris Meudec, Marian Lingsch-Rosenfeld, Hernan, Zsofi Adam, Nils Lommen, Raphaël Monat, Vesal Vojdani, Simmo Saan
Note Takers: Marian Lingsch-Rosenfeld, Jan Strejcek
Agenda (please add your name to your topic):

1. Introduction of people
2. Who will be taking notes today? Preferably multiple people

3. Report from Dirk
   (Web site is online, meetings announced https://sv-comp.sosy-lab.org/2025/dates.php, mail to group sent)
4. Marian and Jan: For memory memtrack, what happens if there is a reversible operation on the last pointer pointing to some memory
    a. Simmo: interesting point, but reversibility might be very difficult to determine: if there only is a pointer to the middle of some array, are all values that determined the offset still accessible from some variables (or whatever computation on them) to know what to subtract in all cases?
       E.g. int *p = arr[nondet()].
    b. Vesal: specifically, the way the rules state this as an LTL formally with globally memories being tracked, so what should be the verdict for the following program where Valgrind reports in the end that everything is fine:

```
void *p;
int main() {
        p = malloc(10);
        p += 5;  // possibly lost
        p += 15; // definitely lost ← so we have a violation?
        p -= 15; // possibly lost
        p -= 5;  // still reachable
        free(p); return 0; }
```

    c. Jan: When changing the format of the pointer, e.g. as a string, saving it to a file is the property violated?
    d. The property is not equivalent to memory-cleanup, since there can be a non-terminating program which always loses track of memory, but this would not be a violation of memory-cleanup due to the program not terminating
    e. **Conclusion**: change the definition to have the meaning ”All allocated memory is tracked, i.e., pointed (in)to (transitively) from some stack or global variable.”, but make it more precise.
    f. ~~**TODO**~~ **(DONE)**: Jan will prepare a merge request for rule change.
5. Simmo: no-overflow definition: are casts signed overflows?
    a. Dirk thinks casts should be overflows, but standard needs to be studied.
    b. ~~**TODO**~~ **(DONE)**: Study C standard on this.
6. Marian: Signed overflow was defined in C23 using the 2 complement representation of integers. See: https://www.open-std.org/jtc1/sc22/wg14/www/docs/n2412.pdf, https://en.cppreference.com/w/c/23. Currently we only allow at most one type of undefined behavior per task. Do we still consider signed overflow undefined behavior i.e. what C standard are we using?
    a. **Conclusion**: We use C11
    b. ~~**TODO**~~ **(DONE)**: Marian does an MR to explicitly state that we are using C11 and updates the mention of C99 to reference C11
    c. ~~**TODO**~~ **(DONE)**: Marian adds to the agenda for next week if we want to switch to C23
7. Link to EasyChair in Important dates is not working
    a. ~~**TODO**~~ **(DONE):** Dirk should fix it once the EasyChair instance is obtained
8. ~~Next meeting date/time? -> see https://sv-comp.sosy-lab.org/2025/dates.php~~
9. Communication channels

a. Vesal has a slight concern that Dirk's second email was blocked by our Microsoft Defender spam filter, maybe this happened for others as well?
   -> Zoom link now on web site: https://sv-comp.sosy-lab.org/2025/dates.php
b. Maybe consider using a jury mailing list / SV-COMP Zulip.
   i. ~~**TODO**~~ **(DONE):** Dirk setups a Zulip stream at the zulip cloud infrastructure

2024-09-24 18:00 - 18:50 CEST

Participants: Dirk Beyer, Levente Bajczi, Raphaël Monat, Dominik Klumpp, Thomas Lemberger, Vesal Vojdani, Simmo Saan, Marian Lingsch-Rosenfeld, Frank Schüssele
Note Takers:
Agenda:

1. Who will be taking notes today? Preferably multiple people
2. Discussion of deadlines
3. Do we want to have the SV-COMP community meeting at ETAPS hybrid this year, due to the amount of traveling?
4. Next meeting date/time?

Discussion:

- Deadline discussion:
  - Time between freezing and last submission could be widened to provide more times to inspect results
  - Levi proposes to have time between qualification and evaluation
    Decided: Nov. 18
  - Enforce that reviewer assignment is actually done directly after registration deadline
  - Jury is known on Oct 12
- Organization Committee:
  - Simmo and Frank to be added to the Benchmark Quality Assurance
- **Next Meetings**:
  - Every Tuesday from now on, alternating 10:00 and 16:00 (CET), starting next week with 16:00
- Someone to enforce deadlines? -> Jan (second chair)
- SV-COMP community meeting at ETAPS
  - Hybrid is difficult/discriminating for online people
  - We don't need to decide now

Full Timeline for SV-COMP (AoE):

Oct. 11, 2024 Registration deadline for tools (basic submission via EasyChair)
Oct. 15, 2024 Submission deadline for verification tasks (via pull request in sv-benchmarks repo)
Oct. 20, 2024 Abstract deadline (paper abstracts via EasyChair; register and submit participating system for a qualification run)
Nov. 1, 2024 Freezing of verification tasks (pull requests after deadline NOT merged)
Nov. 18, 2024 Notifications and tool reviews; list of qualified competition candidates published
Nov. 22, 2024 Competition candidates can resubmit a new version of the verifier (evaluation phase starts)
Dec. 2, 2024 Offline benchmarking phase completed and competition results communicated to the participants (evaluation phase ends)

Dec. 6, 2024 Publication of competition results
Dec. 20, 2024 Paper deadline for submission of system descriptions
Jan. 20, 2025 Paper notification
Jan. 24, 2025 Camera-ready paper deadline for summary papers of qualified competition candidates