CPAchecker

A Tool for Configurable Program Analysis

SoSy-Lab Software Systems

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Daniel Baier, Dirk Beyer, Po-Chun Chien, Marek Jankola, Matthias Kettl, Nian-Ze Lee,
Thomas Lemberger, Marian Lingsch-Rosenfeld, Martin Spiessl, Henrik Wachowitz, and Philipp Wendler

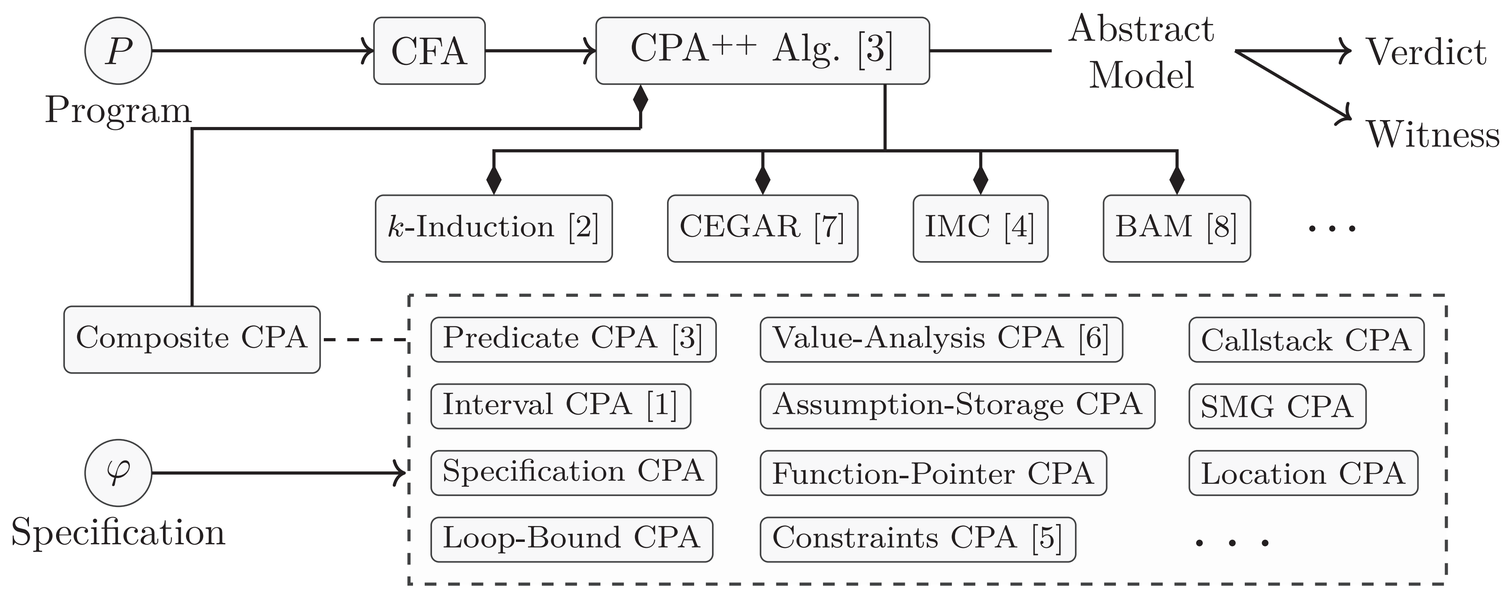
CPAchecker

CPAchecker

CPAchecker is a modern and versatile framework for building software-verification analyses from well-known concepts that match the user's requirements.

cpachecker.sosy-lab.org

Overview



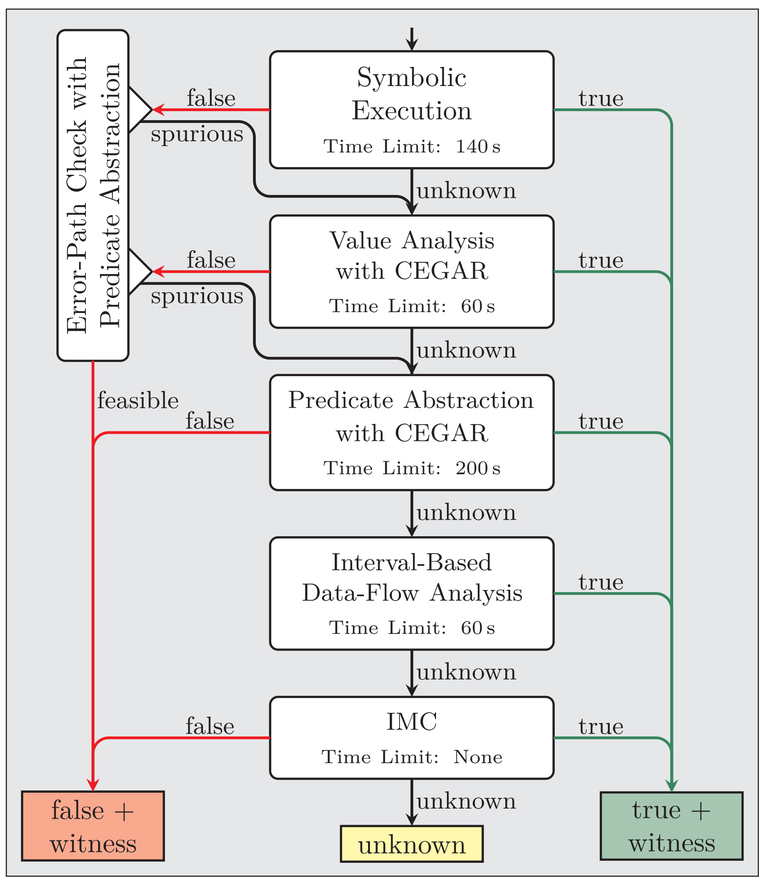
Competition Contribution

"CPAchecker 2.3 with Strategy Selection" is our latest paper describing new developments and configurations used in SV-COMP 2024.

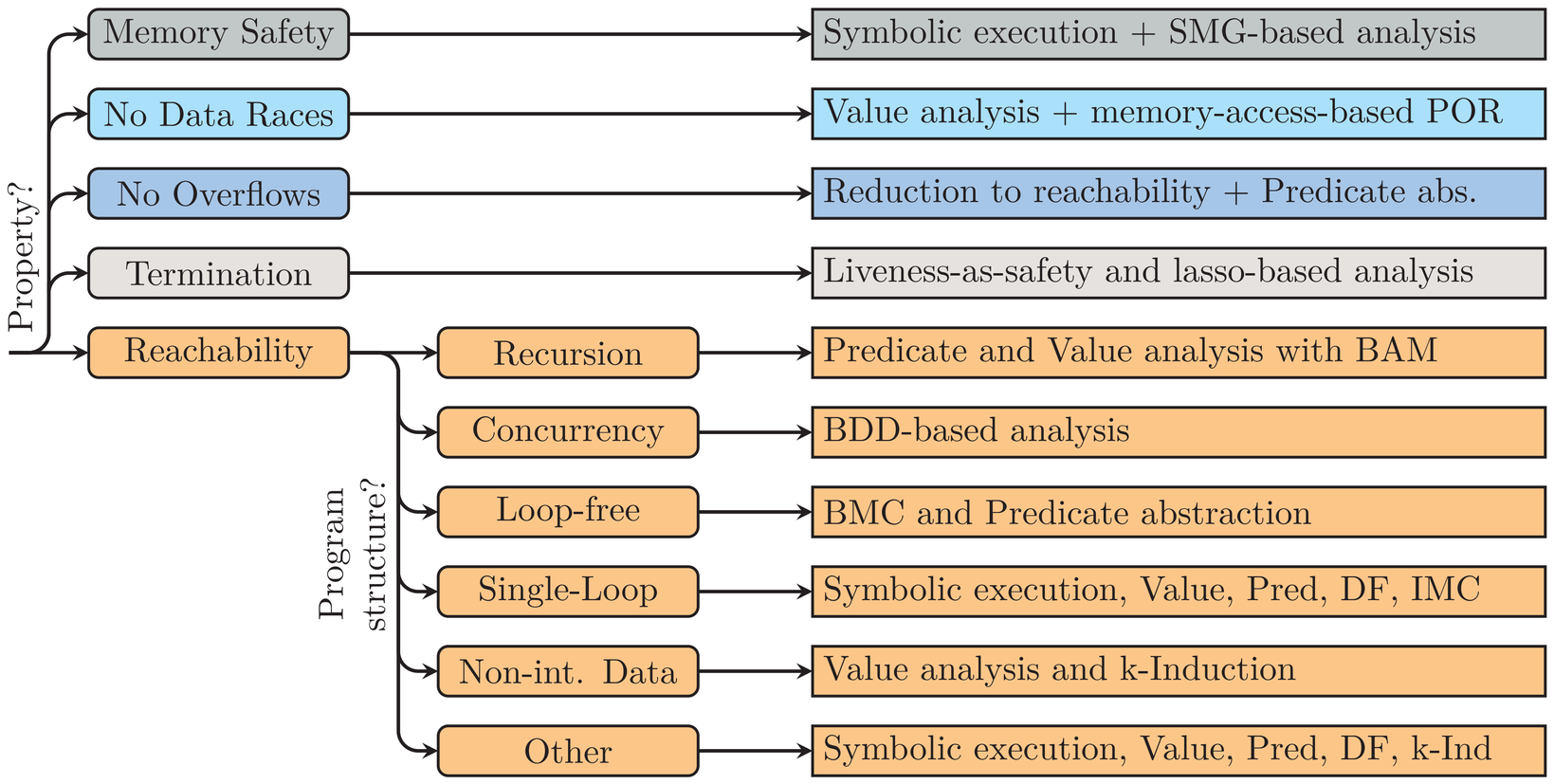
- Utilize strategy selection to predict a sequential portfolio of analyses
- Support all properties and categories of C programs
- 1st place in category *FalsificationOverall*
- 2nd place in category *Overall*
- 3rd place in category *ReachSafety*
- 17 968 validated results in total (the most among all participants)
- Only 17 wrong results (0.06 % of all tasks)
- New and improved analyses for:
  - Reachability
  - Memory safety
  - Termination
  - Overflows
  - Data races

Paper available here

Config for Reachability Single-Loop

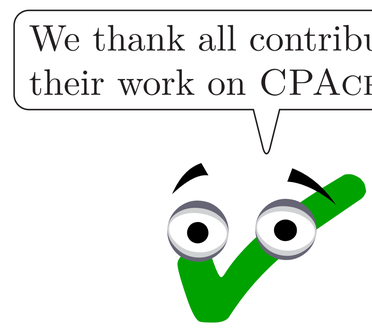


Verification Strategy for SV-COMP 2024



Contributors

CPAchecker is an open-source project, mainly developed by the Software and Computational Systems Lab at LMU Munich, and is used and extended by international associates from U Passau, U Oldenburg, U Paderborn, ISP RAS, TU Prague, TU Vienna, TU Darmstadt, and VERIMAG in Grenoble, along with several other universities and institutes.



References

[1] Beyer, D., Chien, P.C., Lee, N.Z.: CPA-DF: A tool for configurable interval analysis to boost program verification. In: Proc. ASE. pp. 2050–2053. IEEE (2023). https://doi.org/10.1109/ASE56229.2023.00213

[2] Beyer, D., Dangl, M., Wendler, P.: Boosting k-induction with continuously-refined invariants. In: Proc. CAV. pp. 622–640. LNCS 9206, Springer (2015). https://doi.org/10.1007/978-3-319-21690-4_42

[3] Beyer, D., Dangl, M., Wendler, P.: A unifying view on SMT-based software verification. J. Autom. Reasoning **60**(3), 299–335 (2018). https://doi.org/10.1007/s10817-017-9432-6

[4] Beyer, D., Lee, N.Z., Wendler, P.: Interpolation and SAT-based model checking revisited: Adoption to software verification. arXiv/CoRR **2208**(05046) (July 2022). https://doi.org/10.48550/arXiv.2208.05046

[5] Beyer, D., Lemberger, T.: Symbolic execution with CEGAR. In: Proc. ISoLA. pp. 195–211. LNCS 9952, Springer (2016). https://doi.org/10.1007/978-3-319-47166-2_14

[6] Beyer, D., Löwe, S.: Explicit-state software model checking based on CEGAR and interpolation. In: Proc. FASE. pp. 146–162. LNCS 7793, Springer (2013). https://doi.org/10.1007/978-3-642-37057-1_11

[7] Clarke, E.M., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement for symbolic model checking. J. ACM **50**(5), 752–794 (2003). https://doi.org/10.1145/876638.876643

[8] Friedberger, K.: CPA-BAM: Block-abstraction memoization with value analysis and predicate analysis (competition contribution). In: Proc. TACAS. pp. 912–915. LNCS 9636, Springer (2016). https://doi.org/10.1007/978-3-662-49674-9_58