

CPA-DF: A Tool for Configurable Interval Analysis to Boost Program Verification

Dirk Beyer, Po-Chun Chien, and Nian-Ze Lee



CPA '23 @ Luxembourg
LMU Munich, Germany

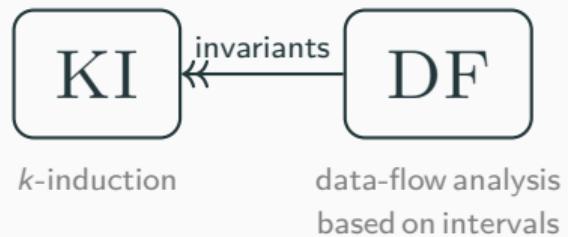


Highlights of CPA-DF

- Use cases:
 - as a standalone verifier for C programs
 - as a **performance booster** in a portfolio-based verifier
- Contributions:
 - discovery of a new aspect of an existing component in CPACHECKER [1]
 - large-scale evaluation
 - open-source data-flow analysis tool

Motivation

Cooperative Invariant Injection [1]



Motivation

Cooperative

Invariant Injection [1]



k-induction

data-flow analysis
based on intervals

vs.

Non-cooperative

Parallel Execution



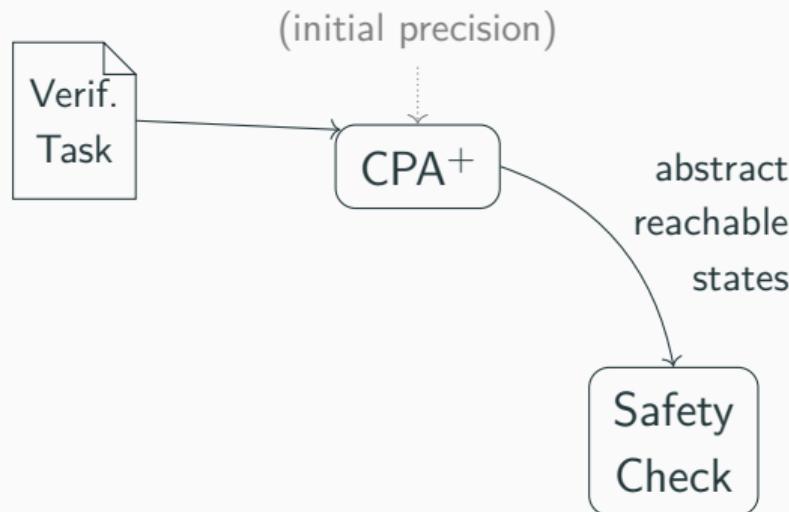
Configurable Program Analysis using Interval Expressions

- Interval CPA \mathbb{I}
 - Abstract domain: $\text{var} \mapsto \text{interval expression}$
 - E.g. $[l_1, u_1] \cup [l_2, u_2]$
 - Precision
 - Important variables: merge allowed only if expressions match on them
 - Widening [4]: whether to further relax an abstraction
- CPA⁺ [2]: a reachability algorithm exploring abstract states

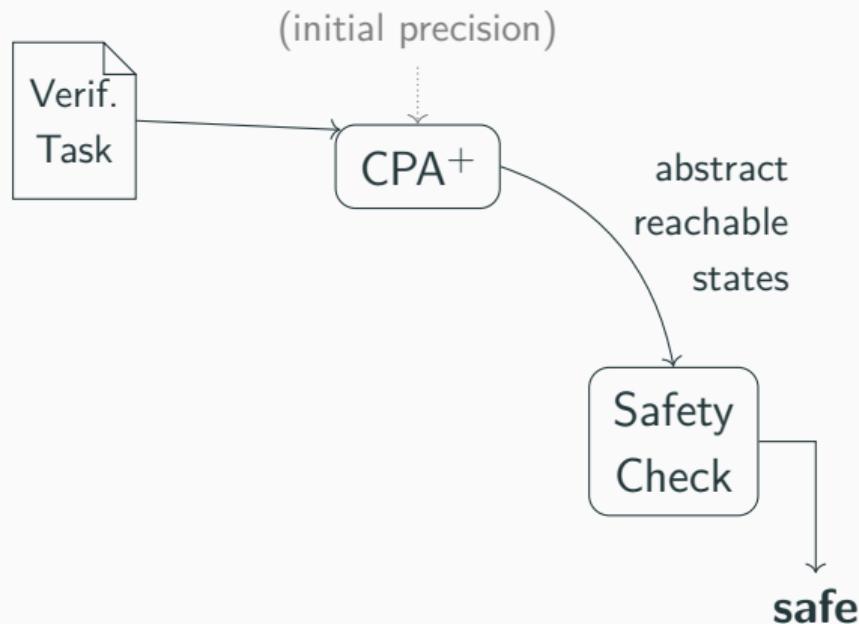
Interval Analysis with Precision Refinement



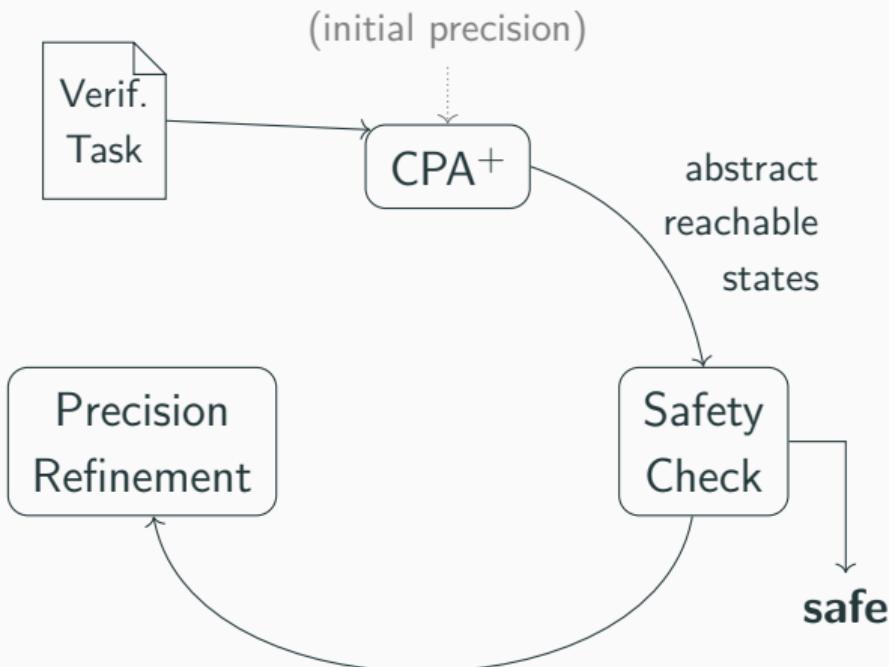
Interval Analysis with Precision Refinement



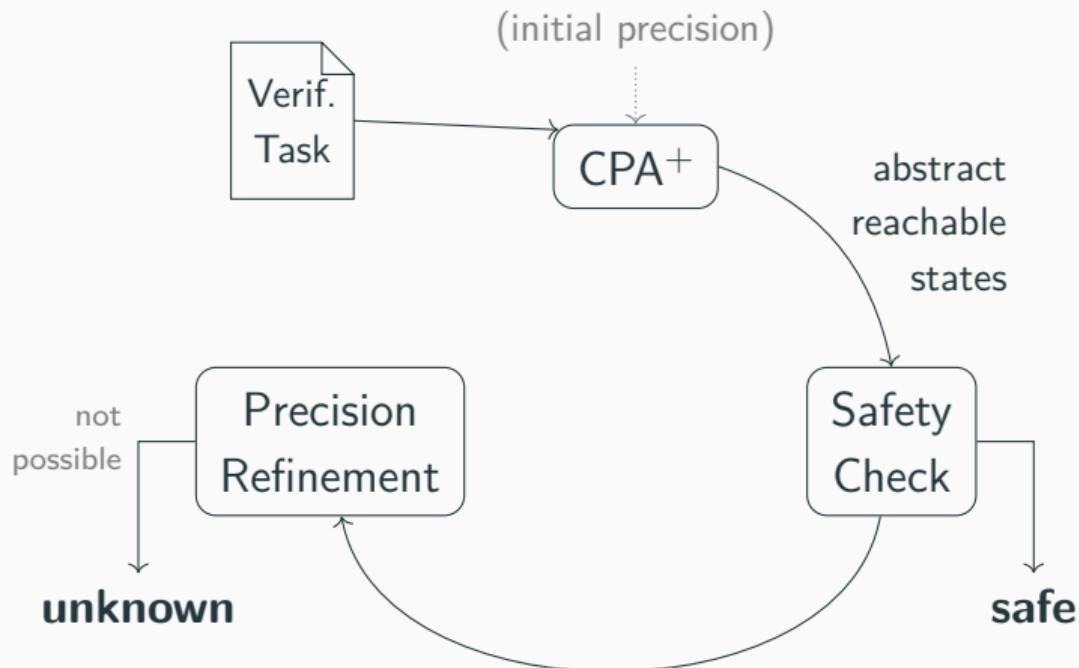
Interval Analysis with Precision Refinement



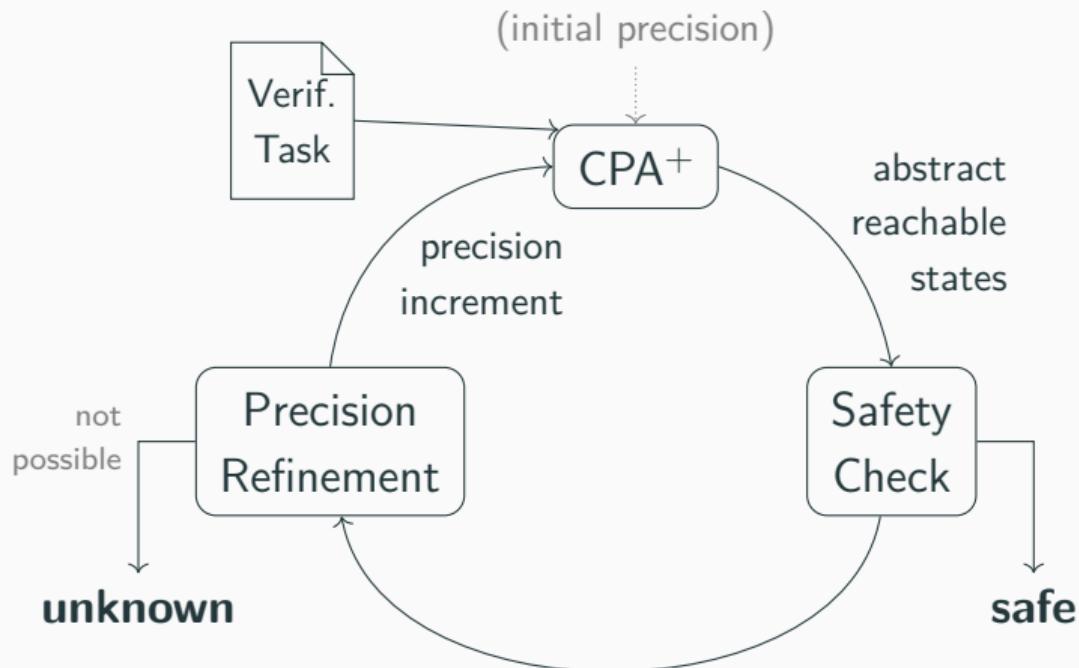
Interval Analysis with Precision Refinement



Interval Analysis with Precision Refinement



Interval Analysis with Precision Refinement



Evaluation

We conducted experiments to answer the following research questions:

- **RQ1:** Does dynamic precision adjustment help CPA-DF?
- **RQ2:** How is $KI \parallel DF$ compared to $KI \ominus DF$ in CPACHECKER?
- **RQ3:** Can CPA-DF complement other verifiers in a parallel portfolio?

Benchmark Tasks and Tools

- 9537 ReachSafety tasks from SV-COMP '23 benchmark set
 - 3153 are unsafe (contain property violation)
 - 6386 are safe
- Top contenders from SV-COMP '23
 - CPACHECKER [3]: DF, KI, $KI \parallel DF$, $KI \ominus DF$
 - ESBMC [5]
 - SYMBIOTIC [6]

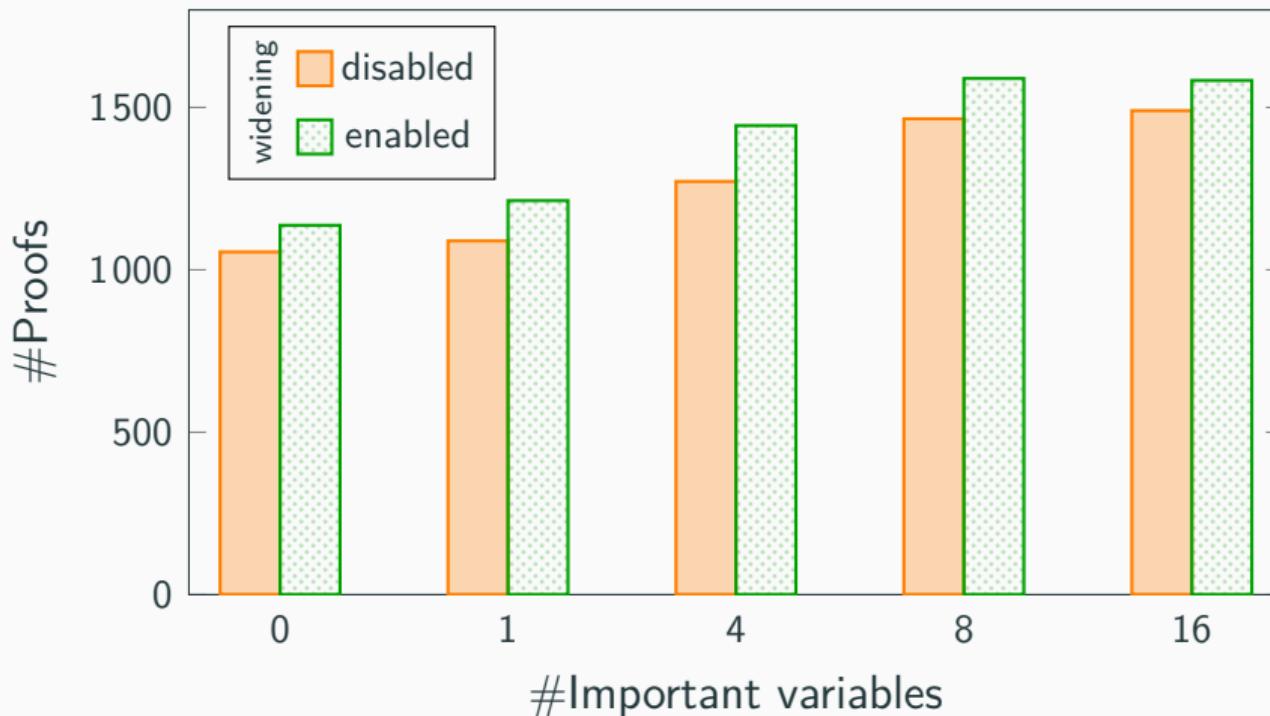
Experimental Setup

- OS: Ubuntu 22.04 (64 bit)
 - Machine: 3.4 GHz CPU (8 cores) and 33 GB of RAM
 - Each task is limited to
 - 4 CPU cores
 - 15 min of CPU time
 - 15 GB of RAM
- (reliable resource management by BENCHEXEC¹)

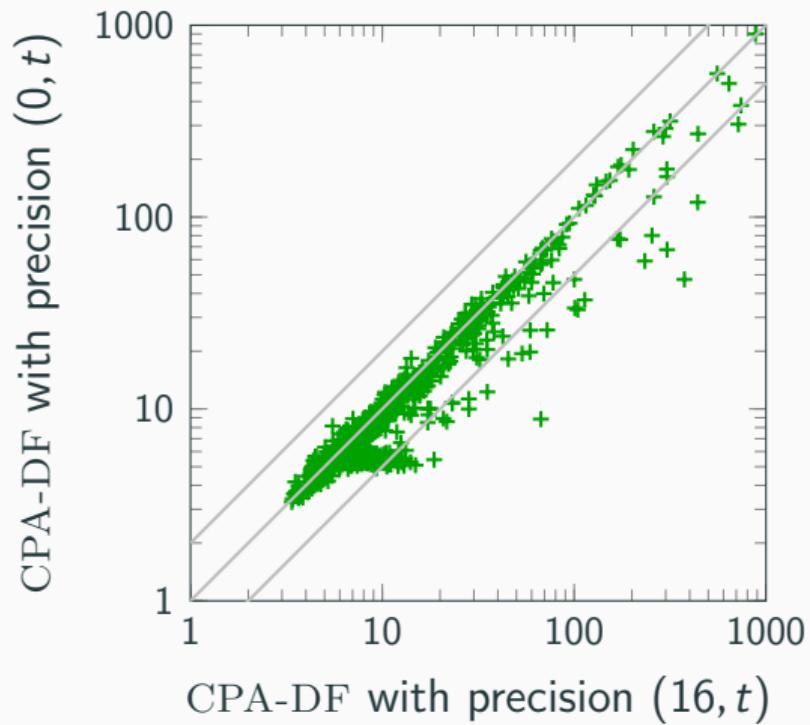
¹<https://github.com/sosy-lab/benchexec>

**RQ1: Does dynamic precision
adjustment help CPA-DF?**

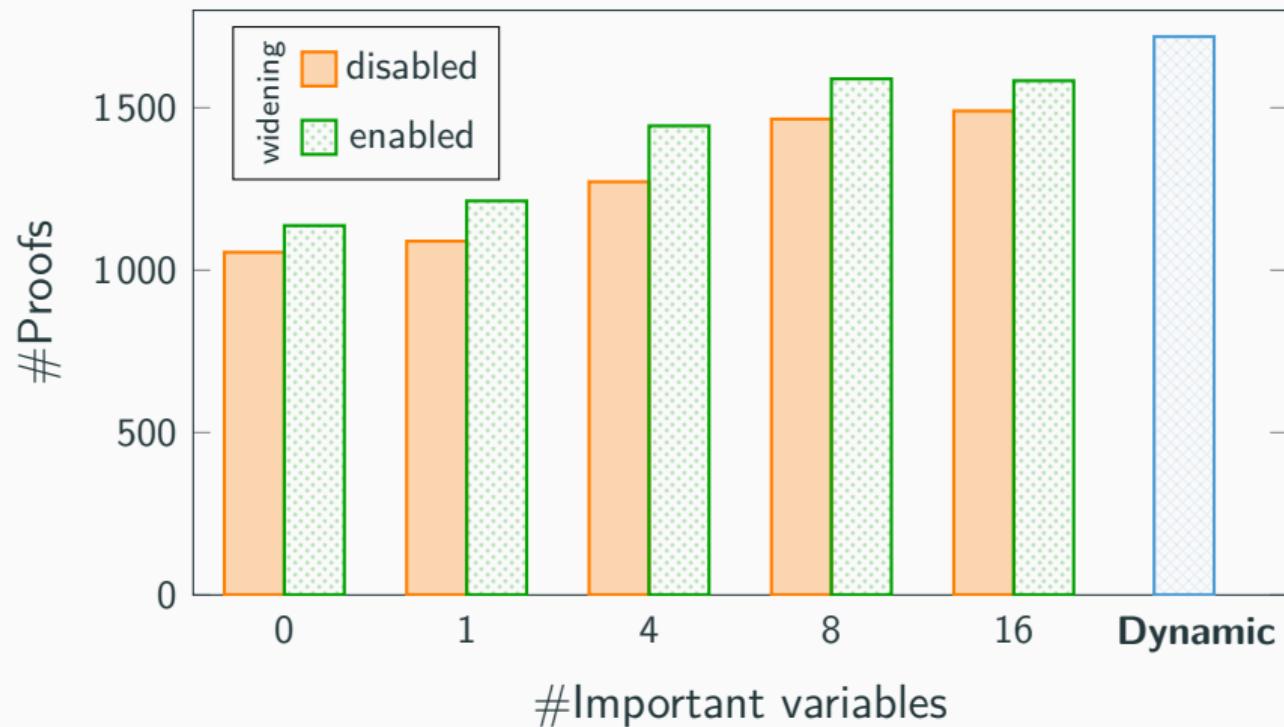
Performance of CPA-DF under Different Precisions



CPU-time Comparison: Low vs. High Precisions

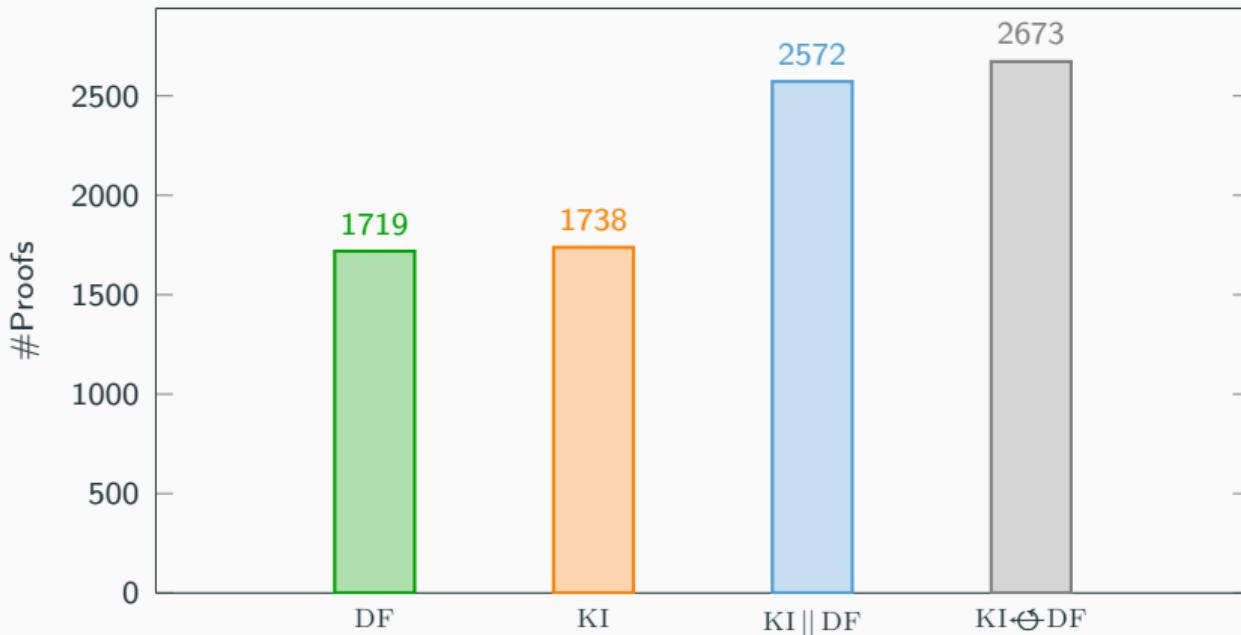


Static vs. Dynamic Precisions



RQ2: How is KI || DF compared to
KI \ominus DF in CPAchecker?

Parallel Portfolio vs. Invariant Injection

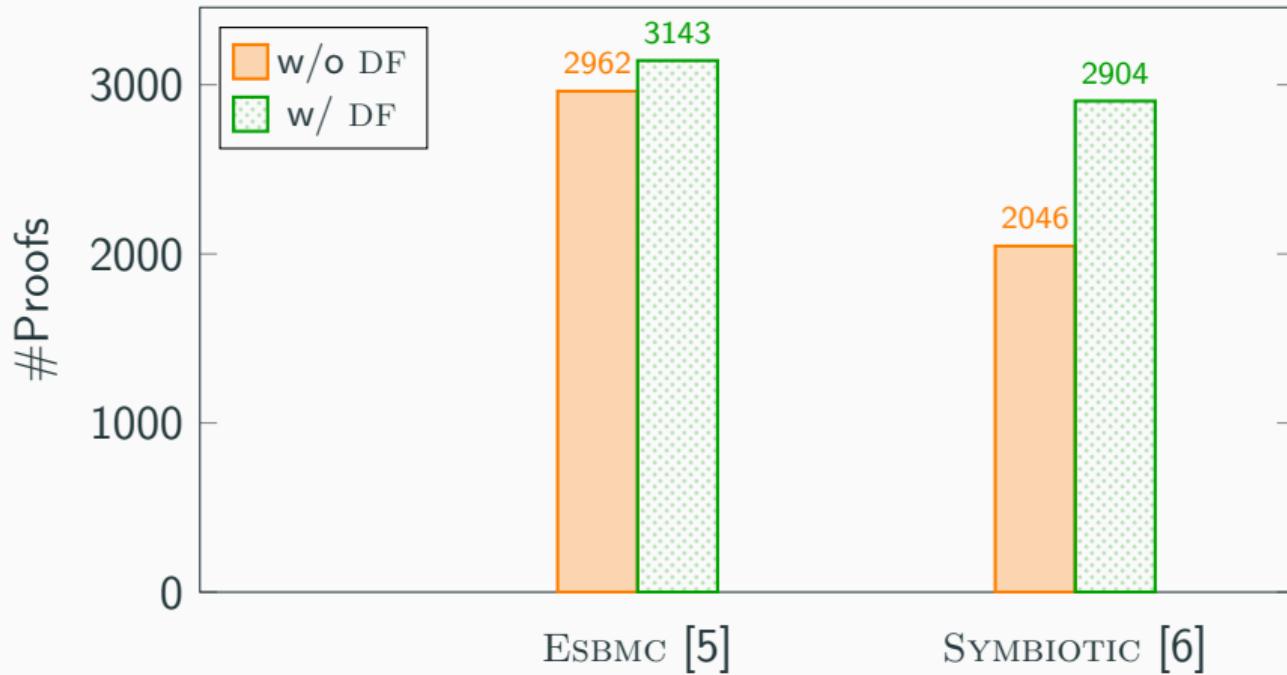


Overall Statistics of KI || DF vs. KI \ominus DF

| Algorithm | DF | KI | KI DF | KI \ominus DF |
|-----------------|------|------|----------|-----------------|
| Correct results | 1719 | 3258 | 4063 | 4164 |
| proofs | 1719 | 1738 | 2572 | 2673 |
| alarms | 0 | 1520 | 1491 | 1491 |
| Wrong proofs | 0 | 0 | 0 | 15 |
| Wrong alarms | 0 | 2 | 2 | 2 |
| Unknowns | 6537 | 6277 | 5472 | 5356 |

**RQ3: Can CPA-DF complement other
verifiers in a parallel portfolio?**

Boosting Program Verification with CPA-DF



Conclusion

- CPA-DF: a data-flow analysis tool based on interval expressions
- In our evaluation,
 - running CPA-DF in parallel to KI achieved a comparable performance as running them cooperatively, and
 - CPA-DF complemented the other well-established verifiers well and boosted the overall performance.

For more info., visit:



[www.sosy-lab.org/
research/cpa-df/](http://www.sosy-lab.org/research/cpa-df/)

References i

- [1] 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
- [2] Beyer, D., Henzinger, T.A., Théoduloz, G.: Program analysis with dynamic precision adjustment. In: Proc. ASE. pp. 29–38. IEEE (2008).
<https://doi.org/10.1109/ASE.2008.13>
- [3] Beyer, D., Keremoglu, M.E.: CPACHECKER: A tool for configurable software verification. In: Proc. CAV. pp. 184–190. LNCS 6806, Springer (2011).
https://doi.org/10.1007/978-3-642-22110-1_16
- [4] Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for the static analysis of programs by construction or approximation of fixpoints. In: Proc. POPL. pp. 238–252. ACM (1977). <https://doi.org/10.1145/512950.512973>

References ii

- [5] Gadelha, M.R., Monteiro, F.R., Morse, J., Cordeiro, L.C., Fischer, B., Nicole, D.A.: ESBMC 5.0: An industrial-strength C model checker. In: Proc. ASE. pp. 888–891. ACM (2018).
<https://doi.org/10.1145/3238147.3240481>
- [6] Slabý, J., Strejček, J., Trtík, M.: Checking properties described by state machines: On synergy of instrumentation, slicing, and symbolic execution. In: Proc. FMICS. pp. 207–221. LNCS 7437, Springer (2012). https://doi.org/10.1007/978-3-642-32469-7_14