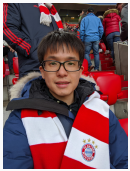
CPA-DF: A Tool for Configurable Interval Analysis to Boost Program Verification

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



ASE '23 @ Luxembourg
LMU Munich, Germany

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

SoSy-Lab Software Systems

CPA

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]

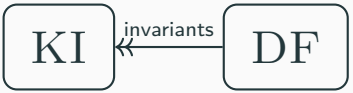
k-induction

data-flow analysis
based on intervals

Motivation

Cooperative

Invariant Injection [1]



k-induction

data-flow analysis based on intervals

vs.

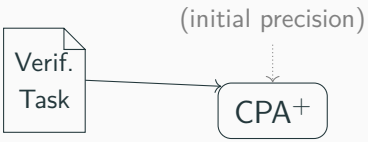
Non-cooperative

Parallel Execution

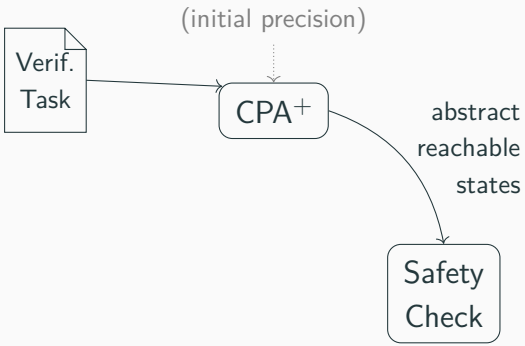
Configurable Progam Analysis using Interval Expressions

- Interval CPA $\mathbb{I}$
  - Abstract domain: *var* $\mapsto$ *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

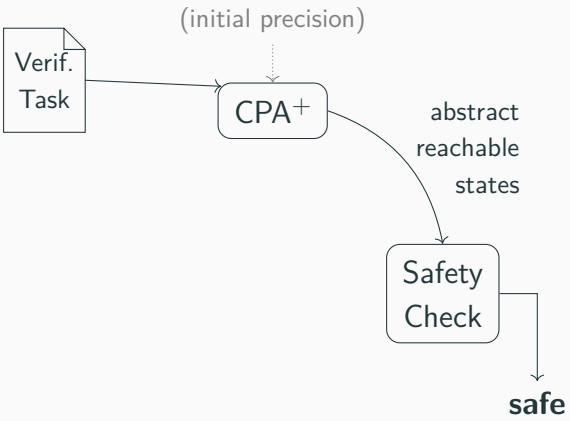
Interval Analysis with Precision Refinement



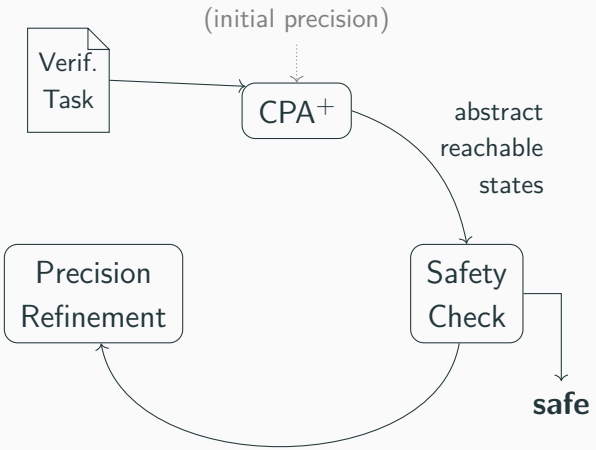
Interval Analysis with Precision Refinement



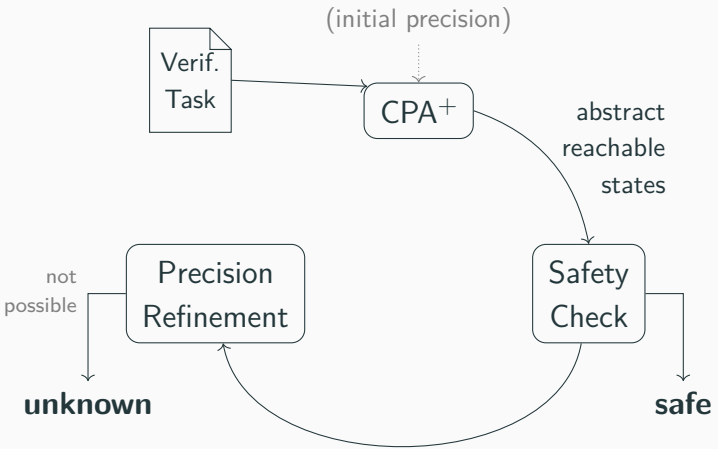
Interval Analysis with Precision Refinement



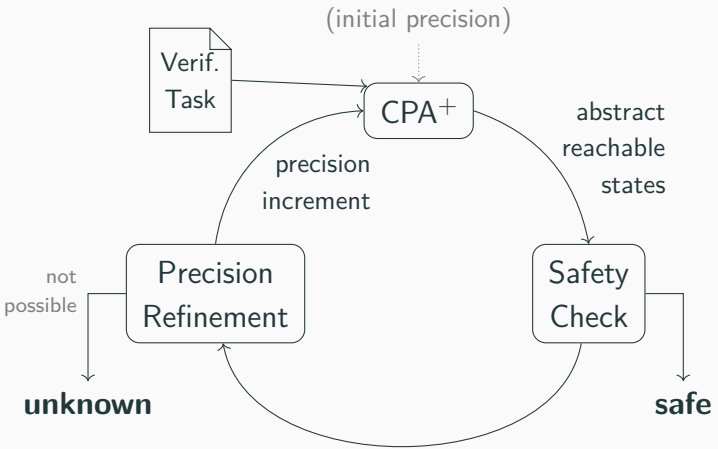
Interval Analysis with Precision Refinement



Interval Analysis with Precision Refinement



Interval Analysis with Precision Refinement



Evaluation

We conducted experiments to answer the following research questions:

- **RQ1:** How is KI || DF compared to KI⟲DF in CPAchecker?
- **RQ2:** Can CPA-DF complement other verifiers in a parallel portfolio?

Benchmark Tasks and Tools

- 6386 ReachSafety tasks from SV-COMP '23 benchmark set
  - all safe, i.e. without known property violation
- Top contenders from SV-COMP '23
  - CPAchecker [3]: DF, KI, KI || DF, KI ⇢ DF
  - Esbmc [5]
  - Symbiotic [6]

Experimental Setup

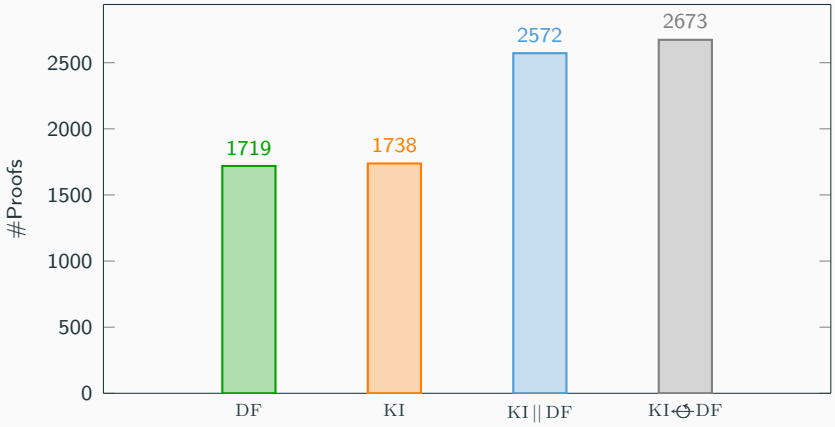
- OS: Ubuntu 22.04 (64 bit)
- Machine: 3.4 GHz CPU (8 cores) and 33 GB of RAM
- Each task is limited to
  - 4 CPU cores
  - 15 min of CPU time
  - 15 GB of RAM

  (reliable resource management by BenchExec[1])

[1] `https://github.com/sosy-lab/benchexec`

RQ1: How is KI || DF compared to KI⟲DF in CPAchecker?

Parallel Portfolio vs. Invariant Injection

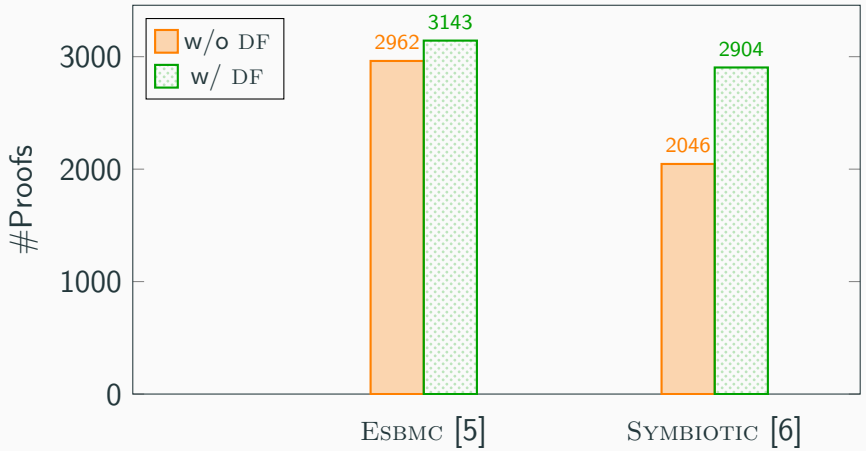


Overall Statistics of KI || DF vs. KI⇜DF

Algorithm	DF	KI	KI || DF	KI⇜DF
Correct proofs	1719	1738	2572	2673
Wrong proofs	0	0	0	**15**
Unknowns	6537	6277	5472	5356

RQ2: 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/

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