In-Place vs. Copy-on-Write CEGAR Refinement for Block Summarization with Caching

Dirk Beyer and Karlheinz Friedberger

LMU Munich, Germany

ISoLA, 2018-11-06

Introduction

Challenge:

- computation of abstract state space *at once* is expensive

Possible solution: block summaries

- split task into smaller problems and solve them separately
- use a cache for intermediate results

Requested requirements:

- independent of domain
- modular implementation: CEGAR, optimization and heuristics

Introduction

BAM in CPAchecker (ICFEM 2012, [3])

CFA divided into *blocks*

- functions or loops as block size
- block size defines entry and exit nodes

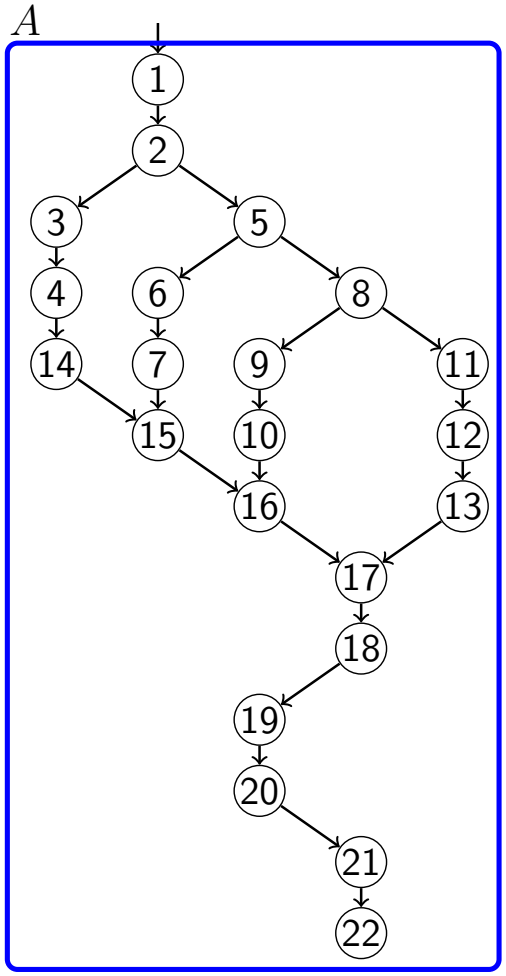
BAMCPA

- implemented as top-level CPA
- manage the analysis and the cache
- optimize cache access with a domain-specific *Reducer*

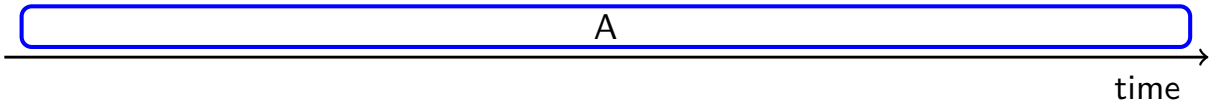
Combinable with...

- several CPAs: predicates, intervals, explicit values
- CEGAR: specialized refinement
- Exporter: state space, counterexample trace, witness

Schematic Example of an Analysis

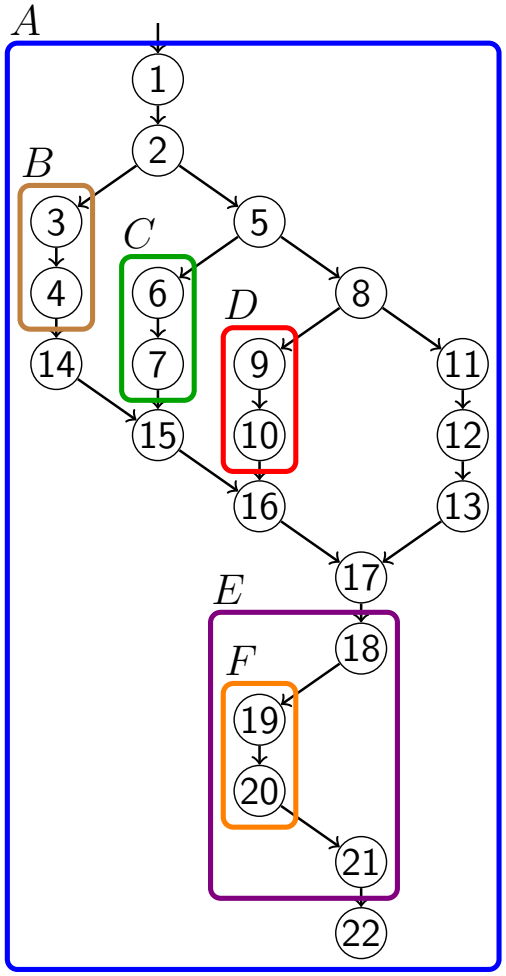


CFA

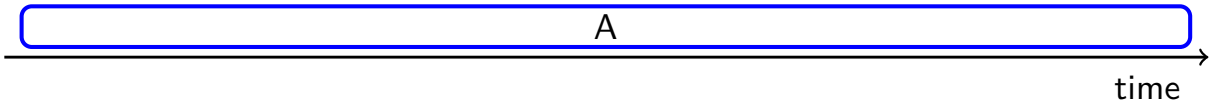


analysis without BAM

Schematic Example of an Analysis

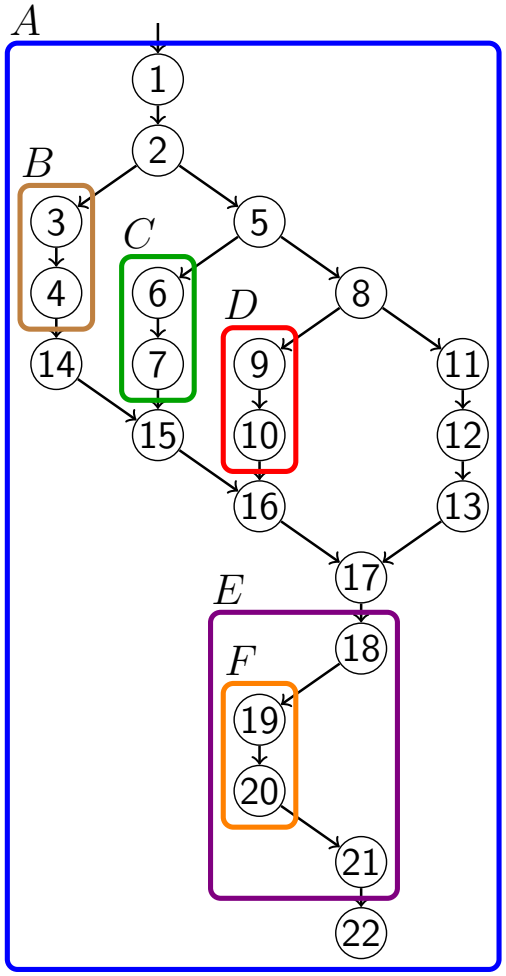


CFA

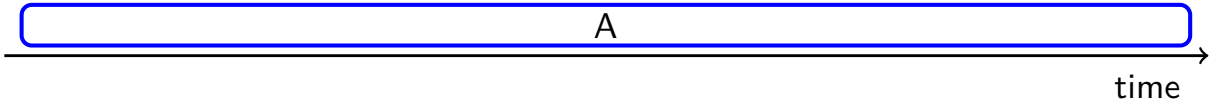


analysis without BAM

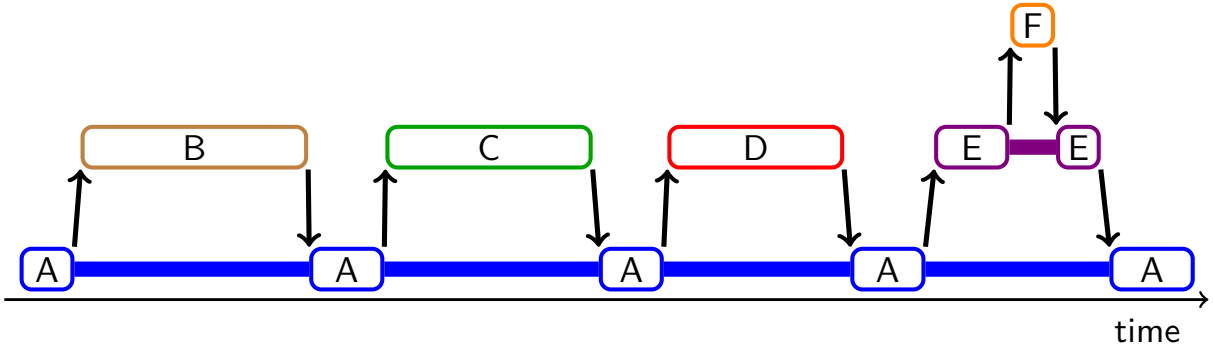
Schematic Example of an Analysis



CFA



analysis without BAM

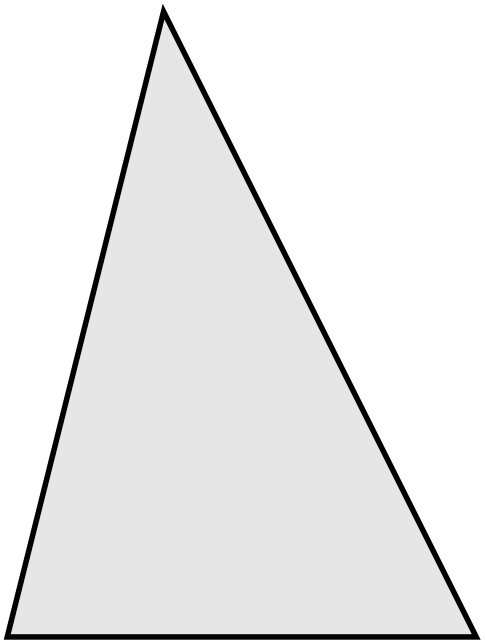


analysis with BAM

CEGAR with Lazy Refinement (without BAM)

Spurious error path found (see BLAST, [2])

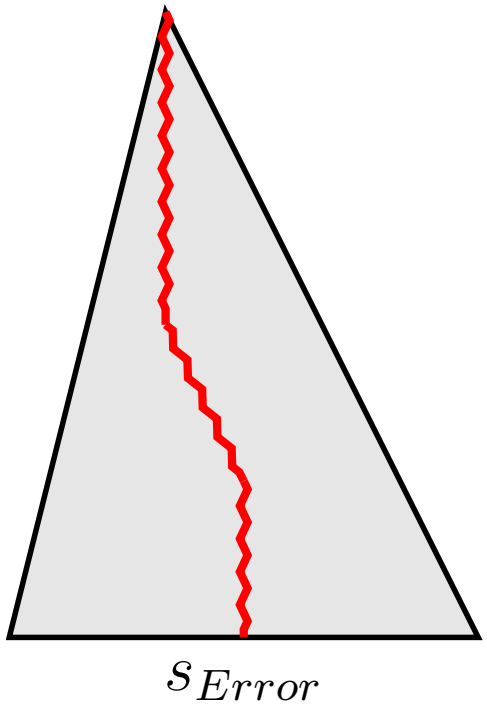
- start refinement procedure
- determine a new precision and a cutpoint
- remove only a "minimal" part of the ARG



CEGAR with Lazy Refinement (without BAM)

Spurious error path found (see BLAST, [2])

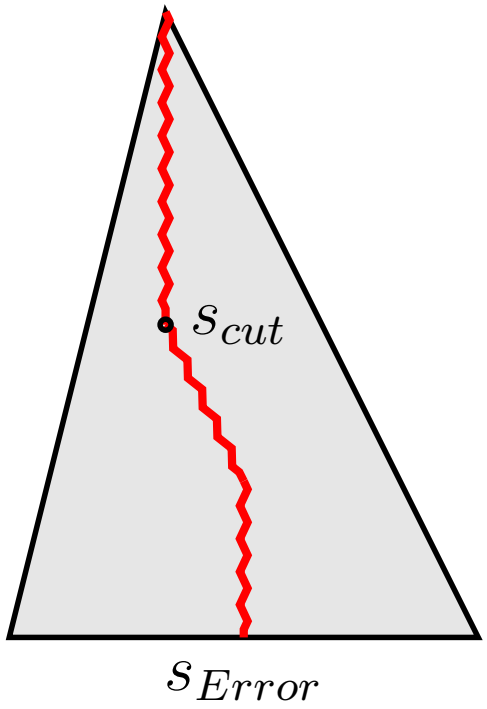
- start refinement procedure
- determine a new precision and a cutpoint
- remove only a "minimal" part of the ARG



CEGAR with Lazy Refinement (without BAM)

Spurious error path found (see BLAST, [2])

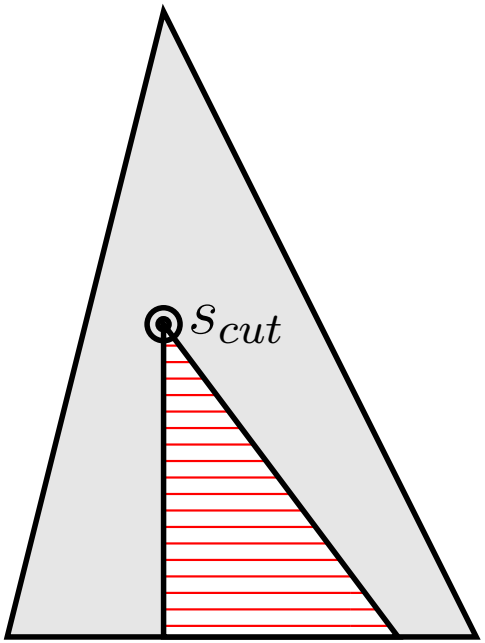
- start refinement procedure
- determine a new precision and a cutpoint
- remove only a "minimal" part of the ARG



CEGAR with Lazy Refinement (without BAM)

Spurious error path found (see BLAST, [2])

- start refinement procedure
- determine a new precision and a cutpoint
- remove only a "minimal" part of the ARG



CEGAR with Lazy Refinement (**with** BAM)

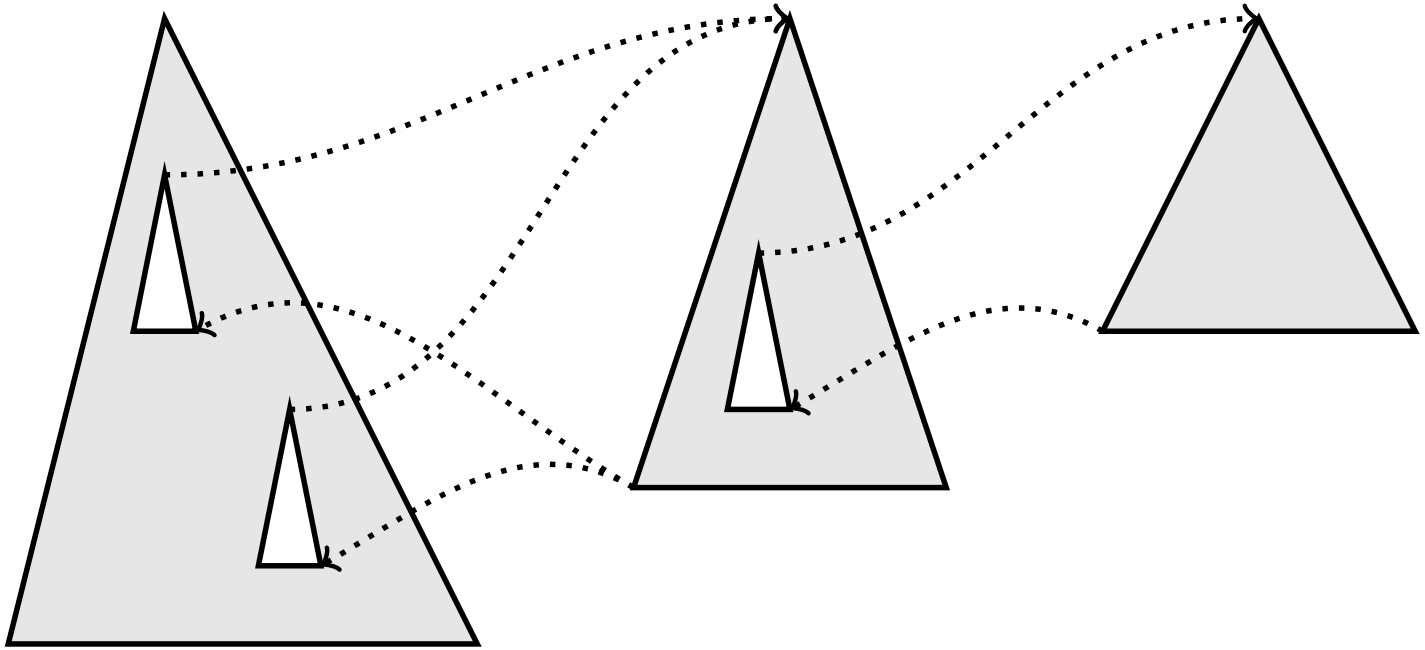
Spurious error path found (see BAM, [3])

- start refinement procedure
- determine precisions and cutpoints over several ARGs
- remove only a "minimal" part of ... ?

CEGAR with Lazy Refinement (**with** BAM)

Spurious error path found (see BAM, [3])

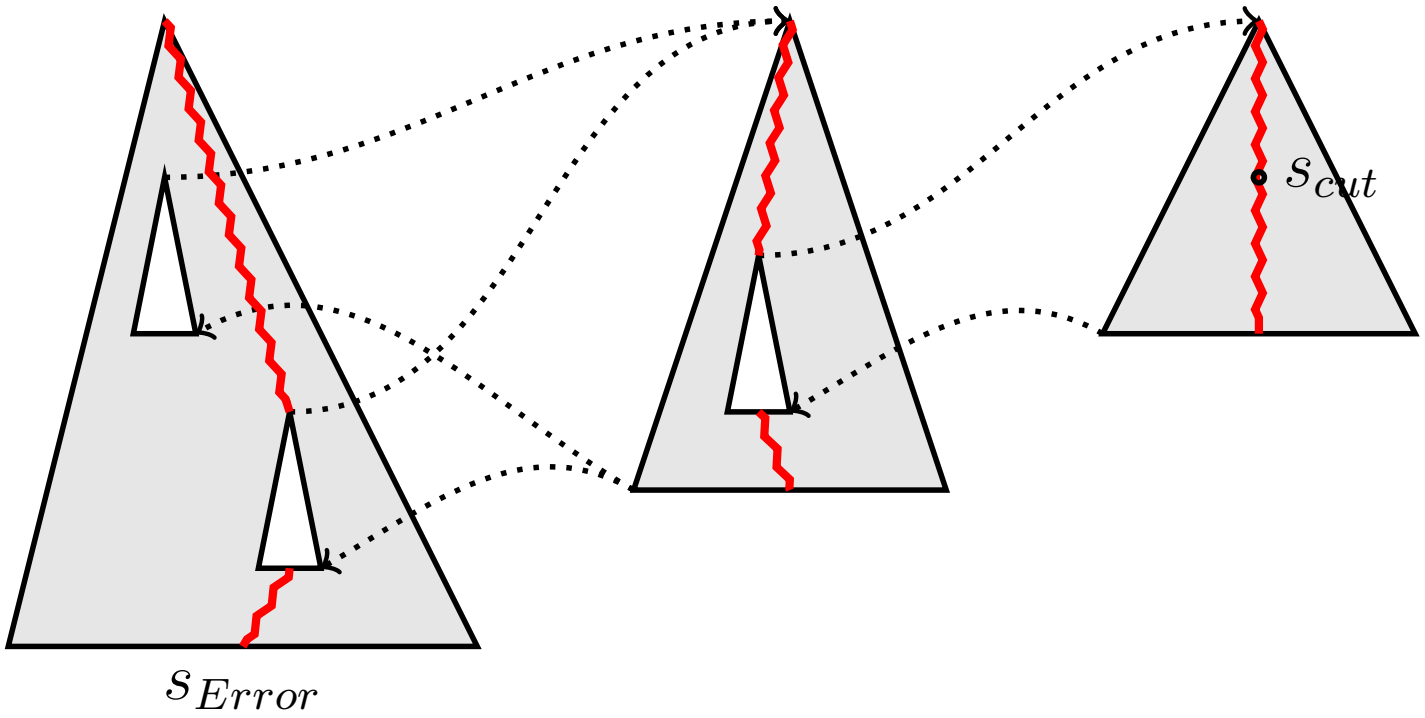
- start refinement procedure
- determine precisions and cutpoints over several ARGs
- remove only a "minimal" part of ... ?



CEGAR with Lazy Refinement (**with** BAM)

Spurious error path found (see BAM, [3])

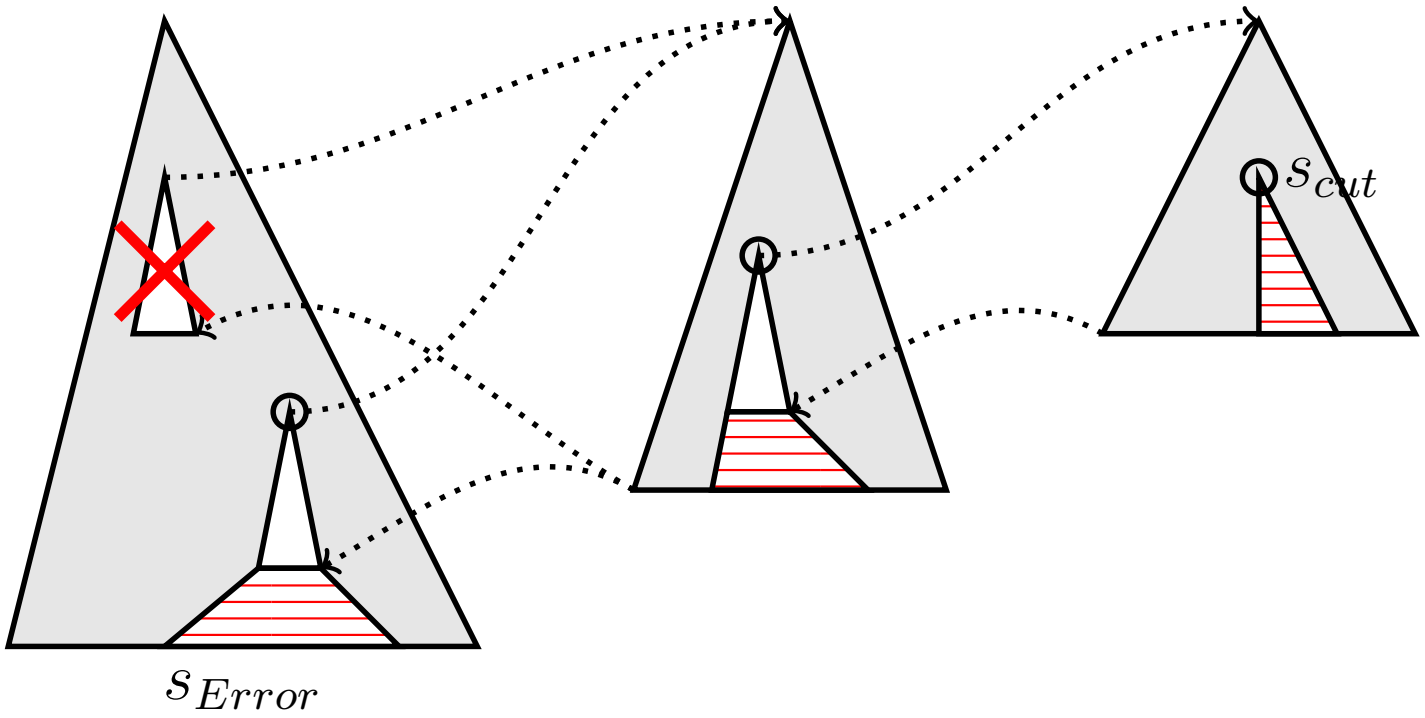
- start refinement procedure
- determine precisions and cutpoints over several ARGs
- remove only a "minimal" part of ... ?



CEGAR with Lazy Refinement (**with** BAM)

Spurious error path found (see BAM, [3])

- start refinement procedure
- determine precisions and cutpoints over several ARGs
- remove only a "minimal" part of ... ?



Problems with the *In-Place* Refinement

Missing information after analysis

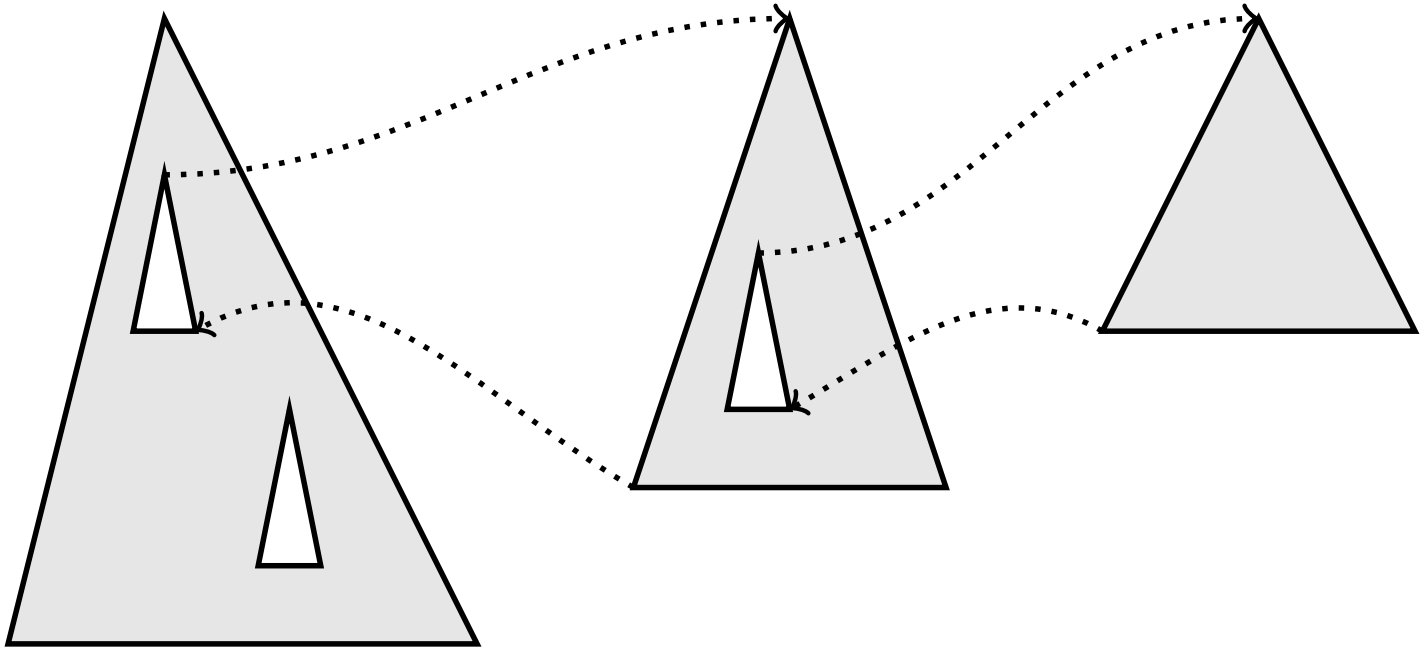
- exporting incomplete data (witnesses, ARGs, statistics)
- re-compute nested blocks or take from cache? precision?

Repeated counterexample

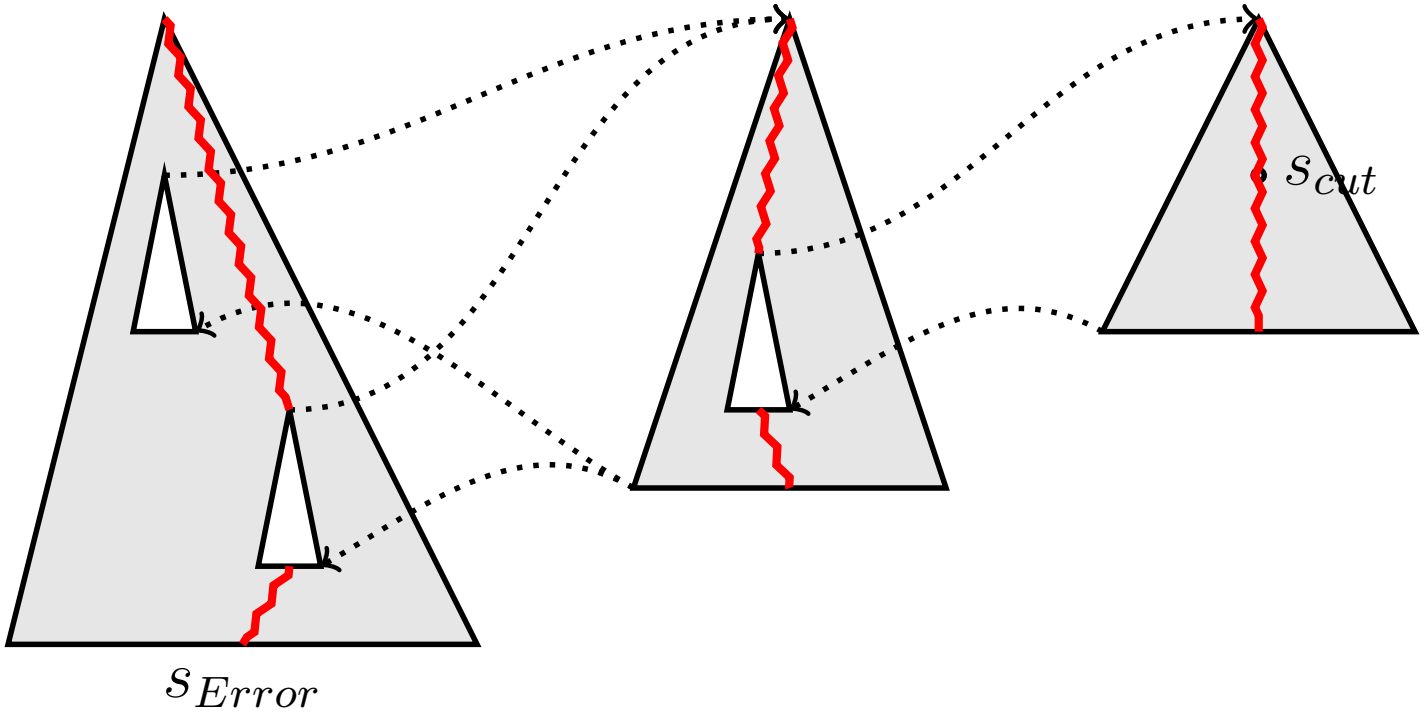
- problem mostly on "large" programs,
  e.g., with many blocks and several refinements
- an error path cannot be excluded from repeated exploration
- cycles of error paths (and refinements)
  $\rightarrow$ no progress in CEGAR

Idea: do not delete computed block abstractions

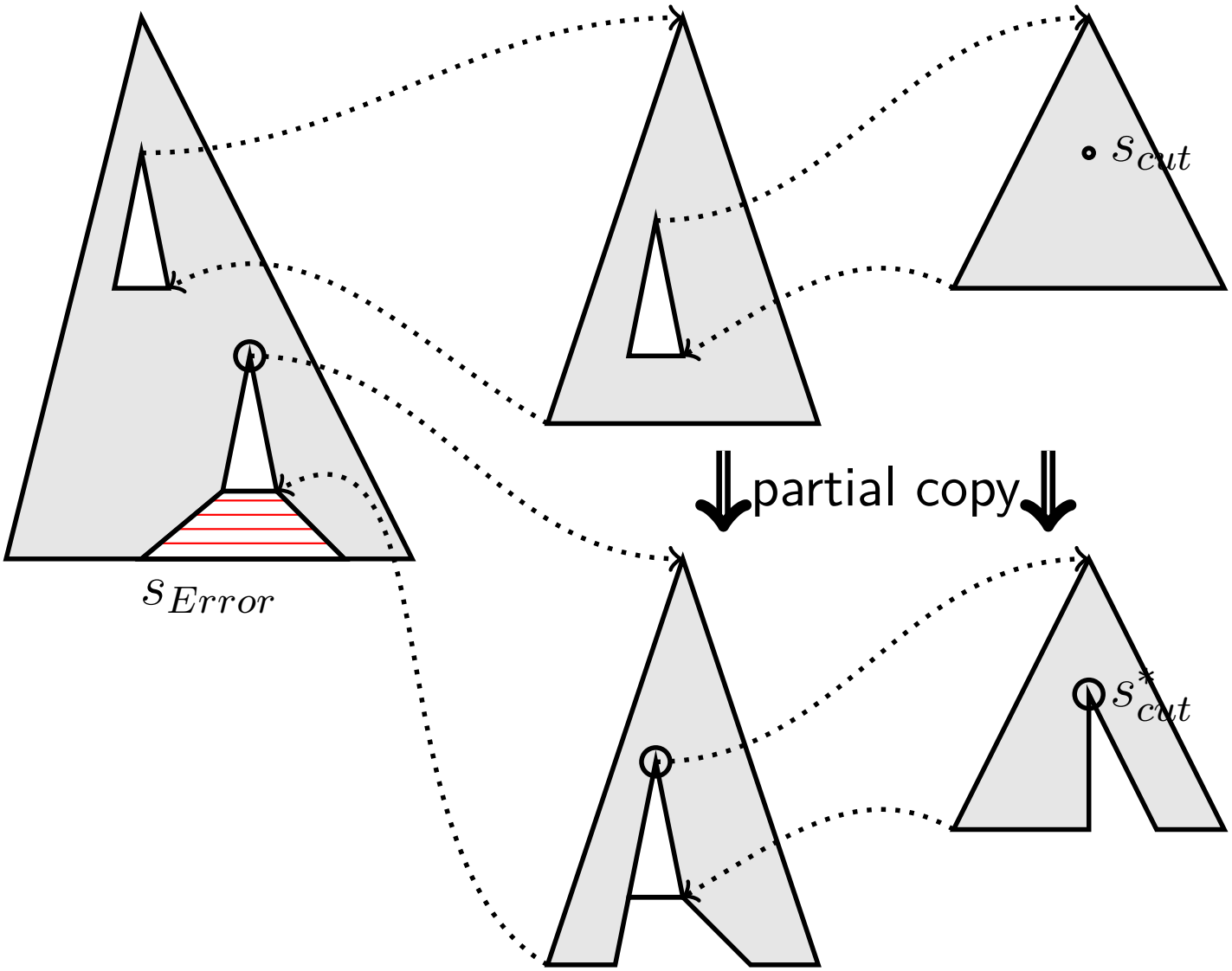
Copy-on-Write Refinement Strategy for the ARG



Copy-on-Write Refinement Strategy for the ARG



Copy-on-Write Refinement Strategy for the ARG



(Idea similar to data structures, file systems, etc.)

Refinement Strategy

Computational overhead?

- *in-place*: removing a subtree needs $O(N)$ time
- *copy-on-write*: copying a subtree needs $O(N)$ time
- only small increase in memory consumption:
  $\rightarrow$ *flat copy* of ARG states

Benefits

- no need to re-computate deleted blocks
- *all* information available at end of analysis
- immutable ARGs (after finished sub-analysis)

Evaluation

Benchmarks and Environment

- SV-COMP 2018 benchmark suite
- Intel Xeon E3-1230 v5 with 3.40 GHz
- 15 GB Ram, 15 min run time

Configurations

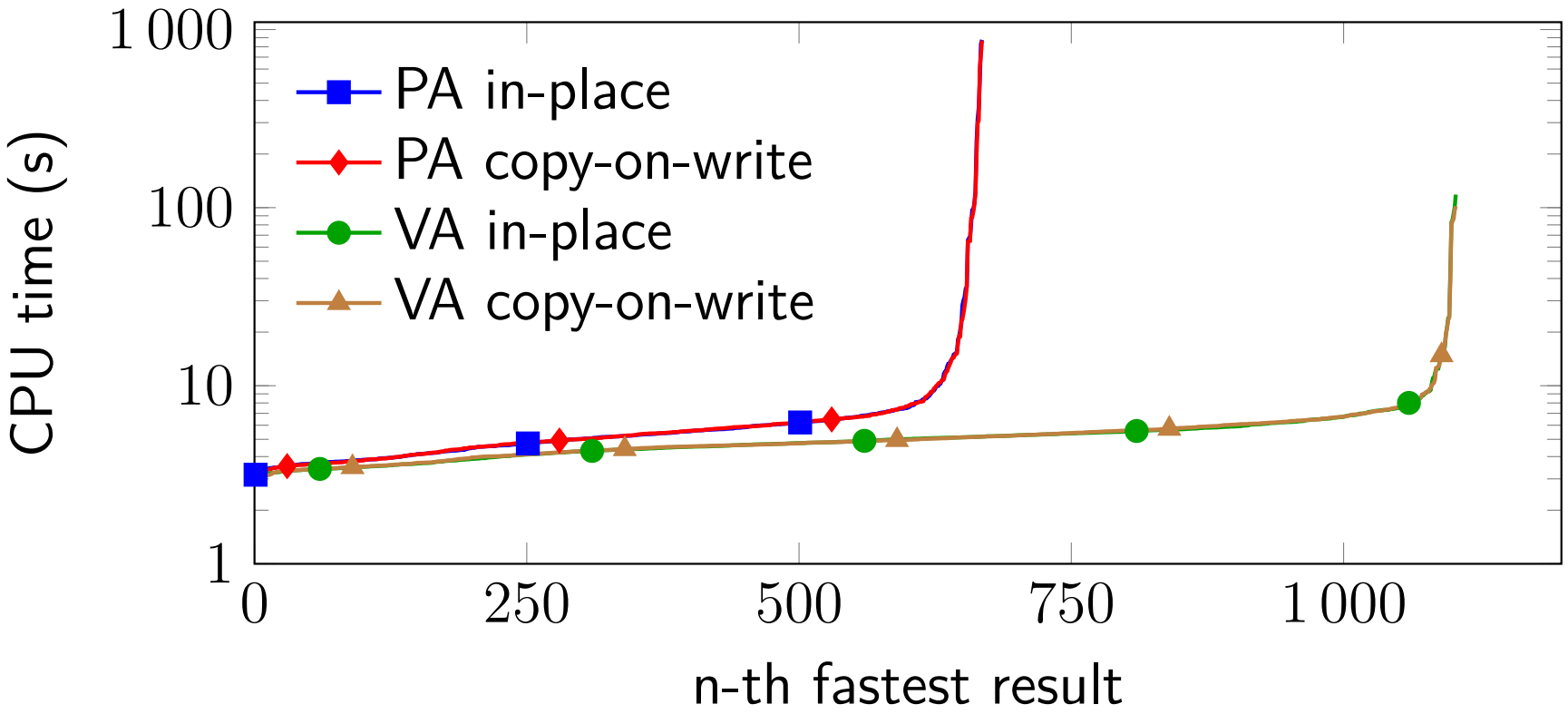
- BAM with predicate analysis,
- BAM with value analysis
- *in-place* vs. *copy-on-write*

Expectations

- tasks with up to one refinement
- tasks with more than one refinement

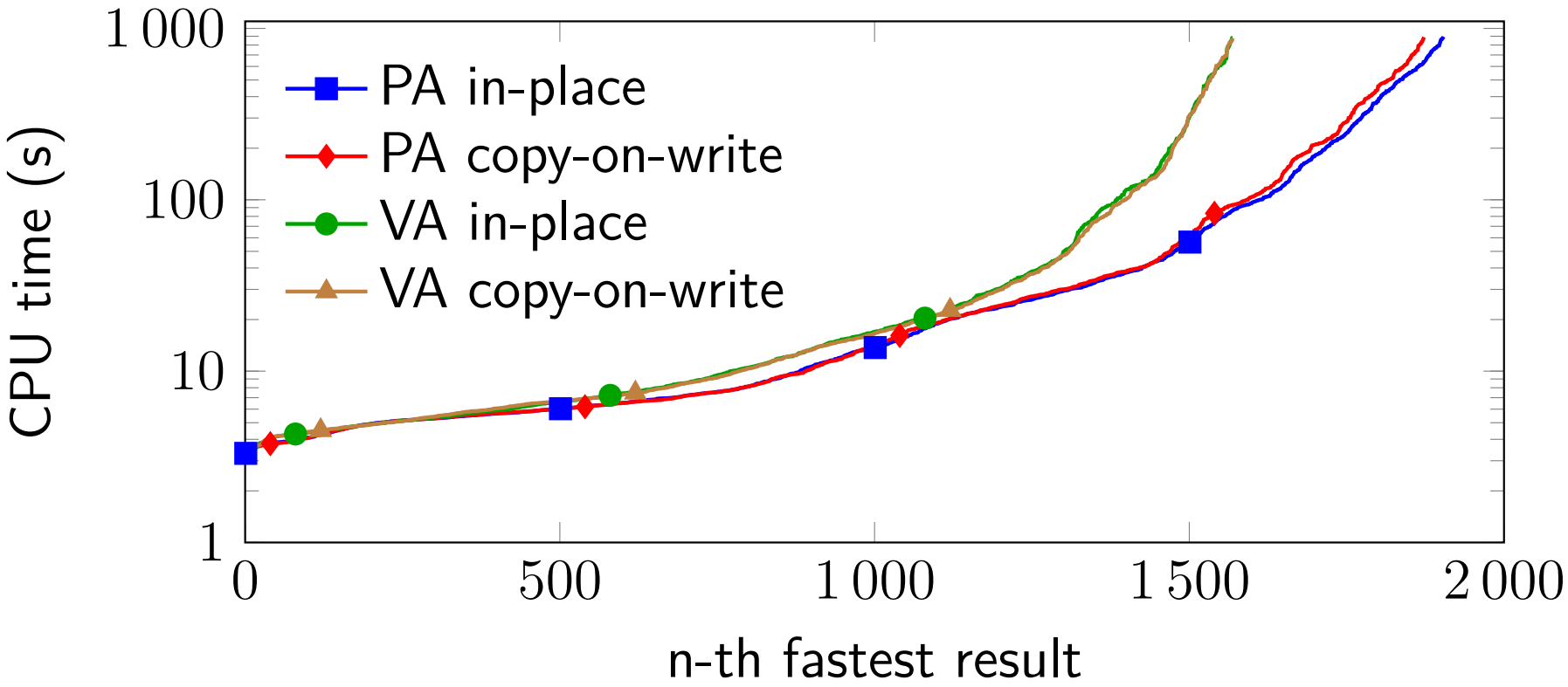
Evaluation ($\leq$1 refinements)

tasks with up to one refinement
$\rightarrow$ no difference expected!

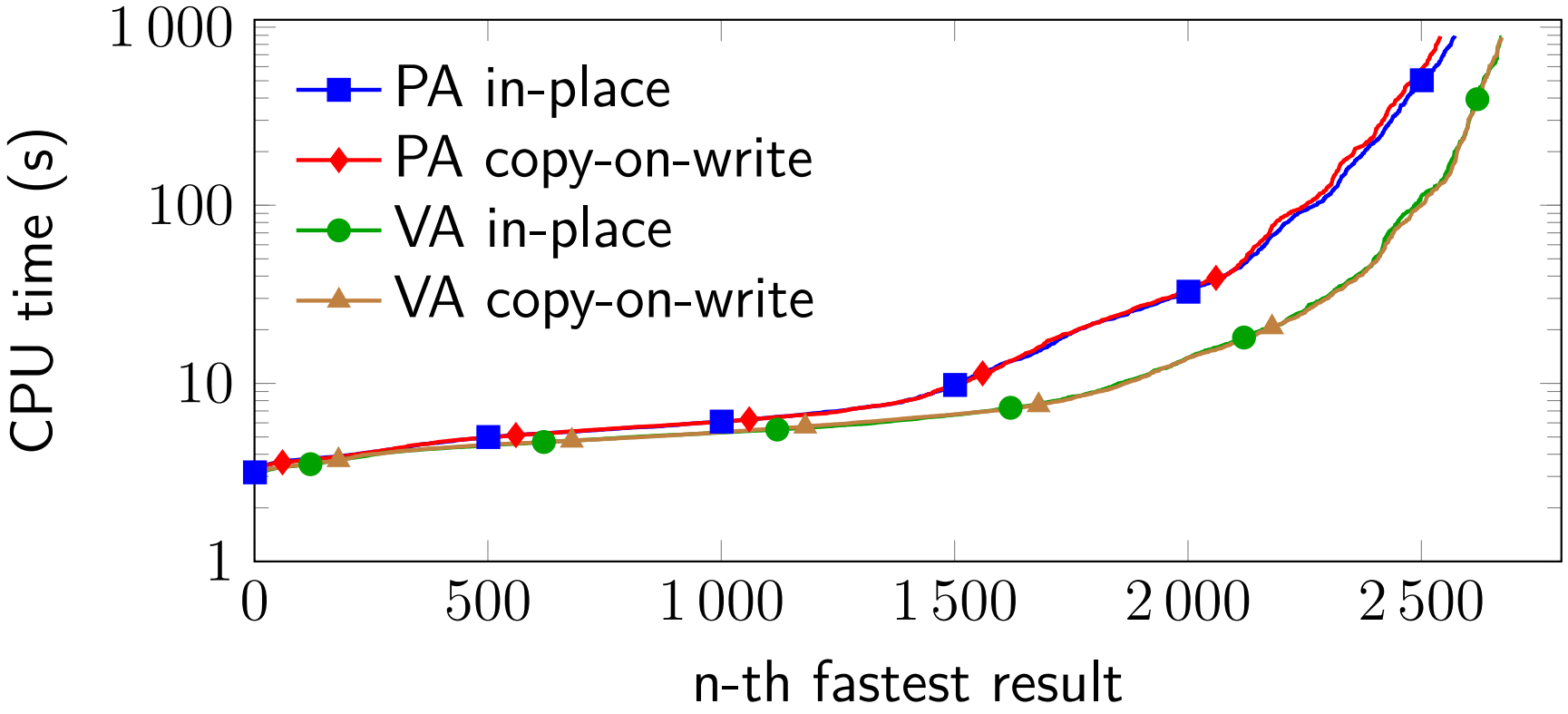


Evaluation (>1 refinements)

tasks with more than one refinement
→ performance difference expected, but...



Evaluation ($\leq$1 and $>$1 refinements combined)



Conclusion

Current status:

- nearly no difference in runtime and number of solved tasks
- exported correctness proof as precise as during analysis

Future work:

- some refinement heuristics might no longer be beneficial
- how to choose from several cache-entries for the same key?
- CEGAR within BAM vs. BAM within CEGAR?

References

D. Beyer and K. Friedberger.
In-place vs. copy-on-write cegar refinement for block summarization with caching.
In *Proc. ISoLA*, LNCS 11245, pages 197–215. Springer, 2018.

D. Beyer, T. A. Henzinger, R. Jhala, and R. Majumdar.
The software model checker BLAST.
Int. J. Softw. Tools Technol. Transfer, 9(5-6):505–525, 2007.

D. Wonisch and H. Wehrheim.
Predicate analysis with block-abstraction memoization.
In *Proc. ICFEM*, LNCS 7635, pages 332–347. Springer, 2012.