Effcient Software Model Checking with Block-Abstraction Memoization

Karlheinz Friedberger

LMU Munich, Germany

3. November 2021

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

SoSy-Lab
Software Systems

Outline

Introduction and Background

Block-Abstraction Memoization

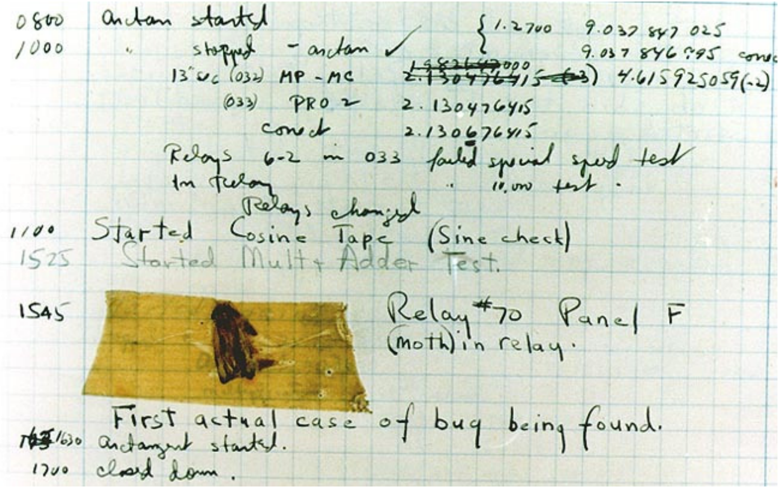
1. Parallel Block-Abstraction Memoization

2. Refinement for Block-Abstraction Memoization

3. Interprocedural Block-Abstraction Memoization

Conclusion

Software Quality



Computer's log of Mark II Aiken Relay Calculator, Grace Hopper, 1947

(Automated) Software Verification

Computer Program

```
int main() {
    int a = foo();
    int b = bar(a);
    if (a != b) error();
}
```

Specification

LTL(G ! call(error()))

Verification Tool

TRUE
i.e., specification is satisfied
→ proof

FALSE
i.e., bug found
→ counterexample

Configurable Program Analysis (CPA)

[Beyer/Henzinger/Théoduloz, 2007]

Fixed-point algorithm for exploring reachable abstract states

- termination based on coverage

Configurable Program Analysis (CPA)

[Beyer/Henzinger/Théoduloz, 2007]

Fixed-point algorithm for exploring reachable abstract states

- termination based on coverage

Operators defined for abstract domain:

- *transfer*: successor computation
- *merge*: combination of two abstract states
- *stop*: coverage of abstract states

Configurable Program Analysis (CPA)

[Beyer/Henzinger/Théoduloz, 2007]

Fixed-point algorithm for exploring reachable abstract states

- termination based on coverage

Operators defined for abstract domain:

- *transfer*: successor computation
- *merge*: combination of two abstract states
- *stop*: coverage of abstract states

domain	abstract state
location	l_3
callstack	$[f_1, f_2]$
explicit value	$\{a = 3, b = 5\}$
predicate	$(l < 4 \land m = 5) \lor n \neq 0$

Block-Abstraction Memoization

[Wonisch/Wehrheim, 2012]

Challenge:

- X computation of the complete abstract state space is expensive
- X analysis is not modular

Block-Abstraction Memoization

[Wonisch/Wehrheim, 2012]

Challenge:

- X computation of the complete abstract state space is expensive
- X analysis is not modular

Possible solution: **block summaries**

- divide and conquer strategy
- reuse intermediate results

Block-Abstraction Memoization

[Wonisch/Wehrheim, 2012]

Challenge:

- ✗ computation of the complete abstract state space is expensive
- ✗ analysis is not modular

Possible solution: **block summaries**

- divide and conquer strategy
- reuse intermediate results

Our contribution:

- ✓ independence of domain
- ✓ modular design and implementation

Block Abstraction

- input-output relation for a block

Block Abstraction

- input-output relation for a block

Examples for several domains:

domain	abstract input state	abstract output state
location	l_3	l_5
callstack	$[f_1, f_2]$	$[f_1, f_2]$
explicit value	$\{a = 3, b = 5\}$	$\{a = 4, b = 6, c = 9\}$
predicate	$(l < 4 \wedge m = 5) \vee n \neq 0$	$(l < 4 \wedge m = 6) \vee n > l + 1$

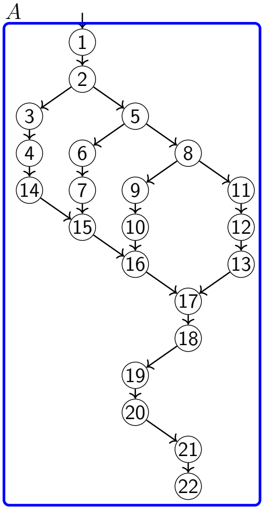
Block Abstraction

- input-output relation for a block

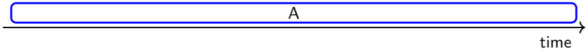
Examples for several domains:

domain	abstract input state	abstract output state
location	l_3	l_5
callstack	$[f_1, f_2]$	$[f_1, f_2]$
explicit value	$\{a = 3, b = 5\}$	$\{a = 4, b = 6, c = 9\}$
predicate	$(l < 4 \land m = 5) \lor n \neq 0$	$(l < 4 \land m = 6) \lor n > l + 1$

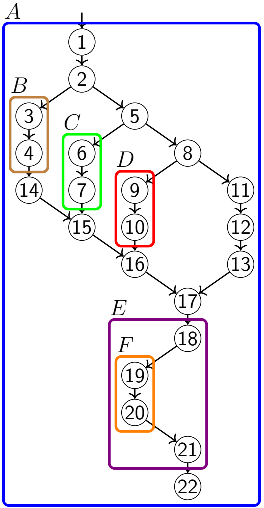
Schematic Example of an Analysis



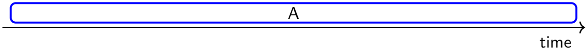
State-Space Exploration



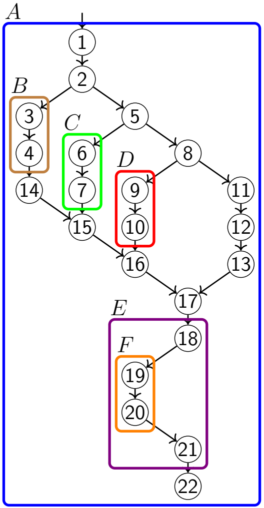
Schematic Example of an Analysis

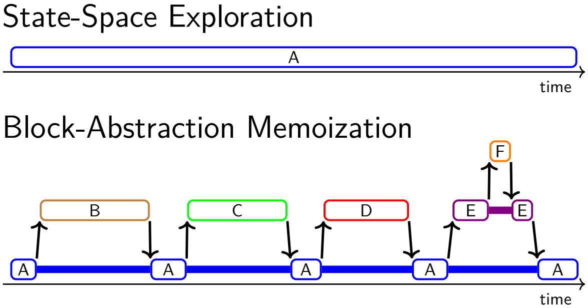


State-Space Exploration

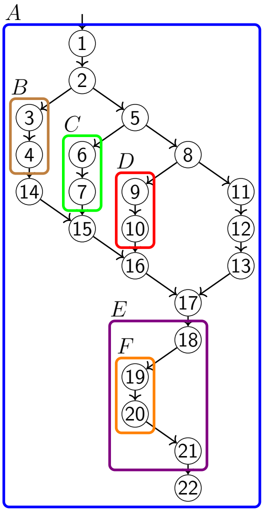


Schematic Example of an Analysis

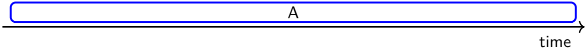




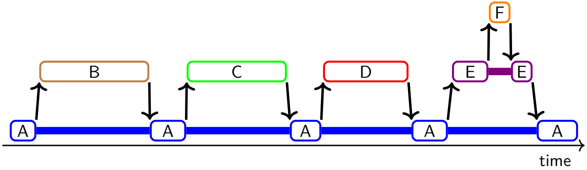
Schematic Example of an Analysis



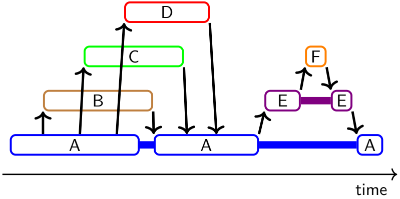
State-Space Exploration



Block-Abstraction Memoization



Parallel Block-Abstraction Memoization



Parallel Block-Abstraction Memoization

[Beyer/Friedberger, 2018]

Challenges with an efficient parallel algorithm:

- X program analysis strictly sequential (per block!)
- X control-flow dependencies between block abstractions

Parallel Block-Abstraction Memoization

[Beyer/Friedberger, 2018]

Challenges with an efficient parallel algorithm:

- ✗ program analysis strictly sequential (per block!)
- ✗ control-flow dependencies between block abstractions

Our contribution: Parallel Block-Abstraction Memoization

- ✓ combination of the existing CPA concept and a parallel application

Evaluation for Parallel Block-Abstraction Memoization

Research questions

- ? 1 processing unit: overhead of the parallel approach
- ? performance with more processing units

Evaluation for Parallel Block-Abstraction Memoization

Research questions

- ? 1 processing unit: overhead of the parallel approach
- ? performance with more processing units

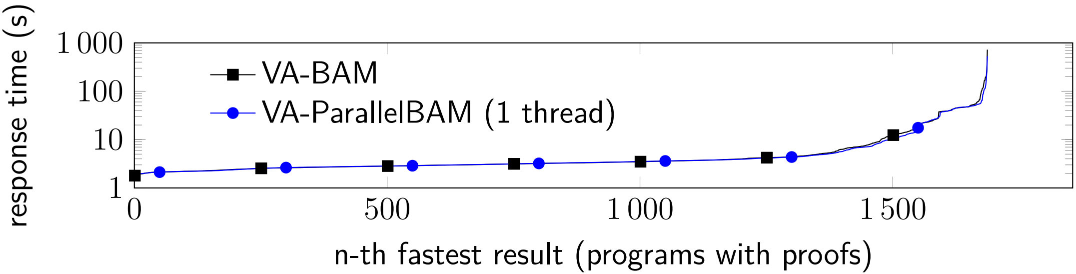
Configuration (CPAchecker r28809)

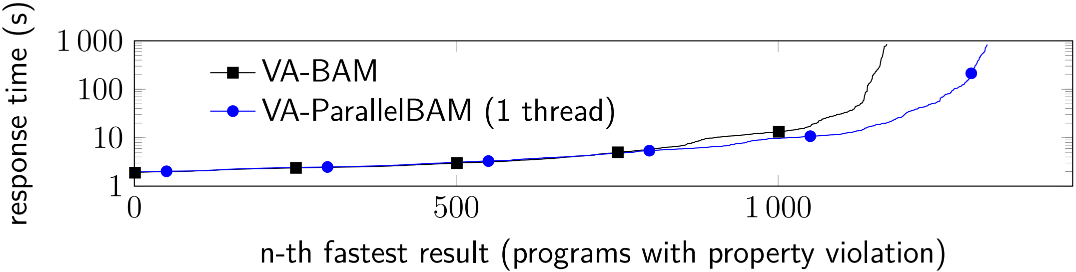
- explicit-value analysis (VA) with BAM or with ParallelBAM

Environment and tasks

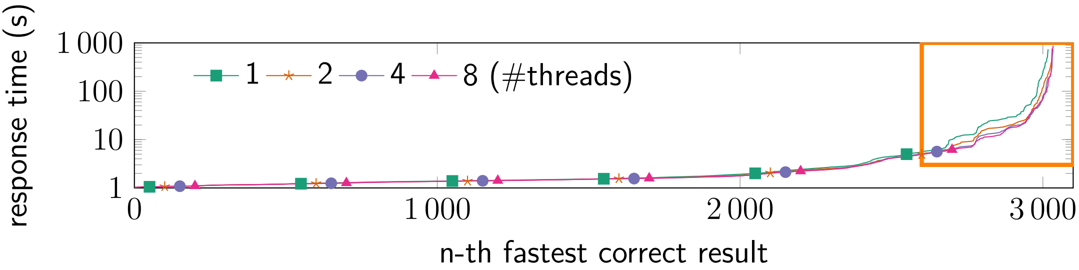
- Intel Xeon E3-1230 v5 with 3.40 GHz and 4 physical cores
- limitation of 15 GB RAM and 15 min of runtime
- 5400 tasks from SV benchmark suite

Evaluation: Sequential vs. Parallel Approach

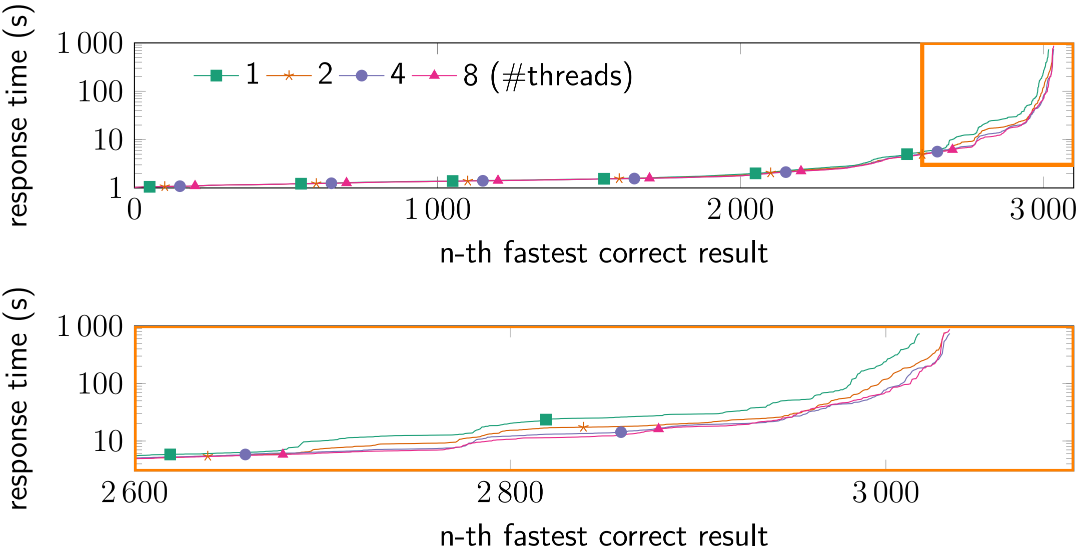




Evaluation: More Processing Units



Evaluation: More Processing Units



Refinement for Block-Abstraction Memoization

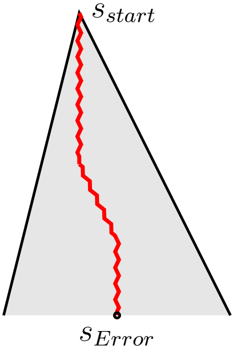
Counterexample-guided Abstraction Refinement (CEGAR)

- granularity of the analysis
- domain-independent approach

Refinement for Block-Abstraction Memoization

Counterexample-guided Abstraction Refinement (CEGAR)

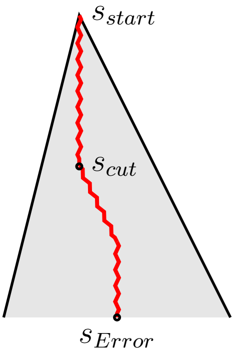
- granularity of the analysis
- domain-independent approach



Refinement for Block-Abstraction Memoization

Counterexample-guided Abstraction Refinement (CEGAR)

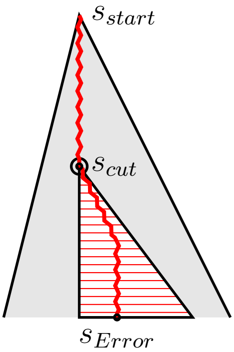
- granularity of the analysis
- domain-independent approach



Refinement for Block-Abstraction Memoization

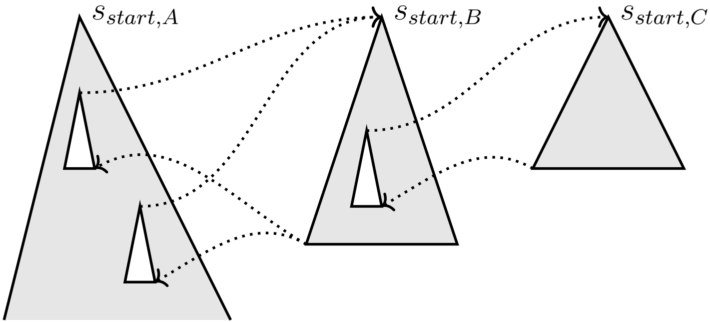
Counterexample-guided Abstraction Refinement (CEGAR)

- granularity of the analysis
- domain-independent approach

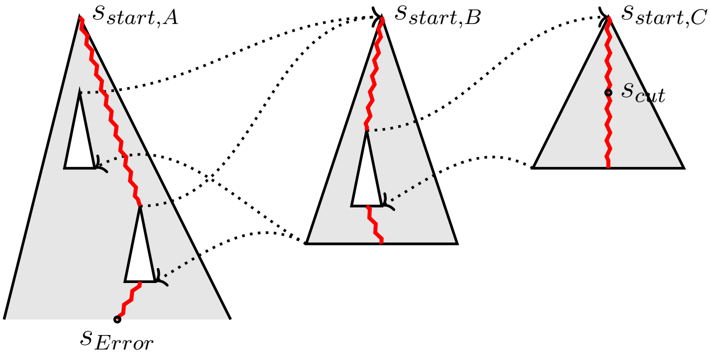


CEGAR with Lazy Refinement (with BAM)

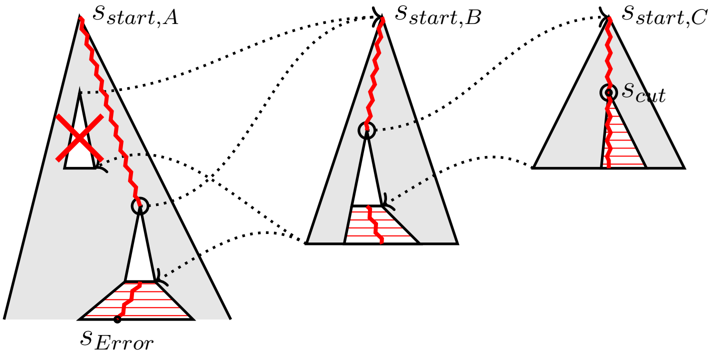
CEGAR with Lazy Refinement (with BAM)



CEGAR with Lazy Refinement (with BAM)



CEGAR with Lazy Refinement (with BAM)



Challenges with *In-Place* Refinement

[Beyer/Friedberger, 2018]

Missing information after the refinement

- ? re-compute nested blocks or take partial results from cache?

Challenges with *In-Place* Refinement

[Beyer/Friedberger, 2018]

Missing information after the refinement

- ? re-compute nested blocks or take partial results from cache?

Missing information after the analysis

- X export of incomplete data (witnesses, explored state space, statistics)
- X no guarantee for progress in the analysis

Challenges with *In-Place* Refinement

[Beyer/Friedberger, 2018]

Missing information after the refinement

- ? re-compute nested blocks or take partial results from cache?

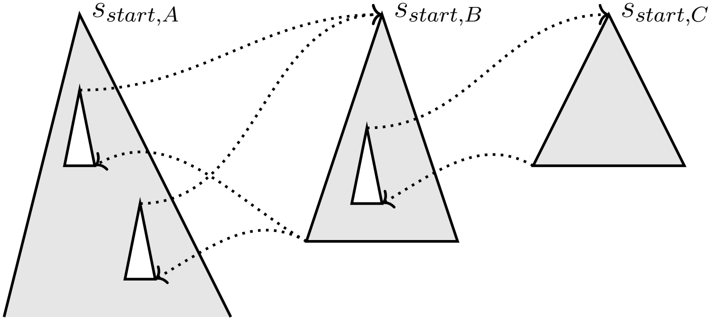
Missing information after the analysis

- X export of incomplete data (witnesses, explored state space, statistics)
- X no guarantee for progress in the analysis

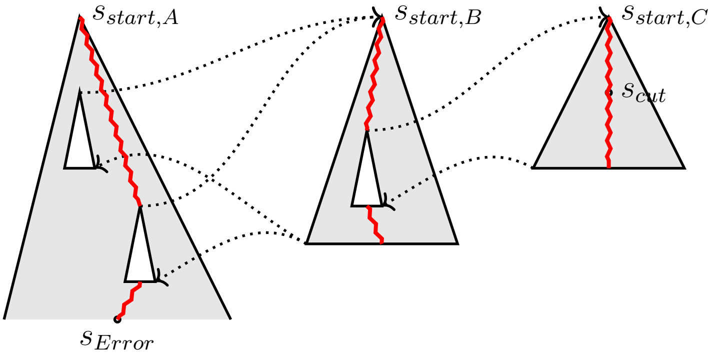
Our contribution: *Copy-on-Write* refinement for BAM

- no deletion of computed block abstractions

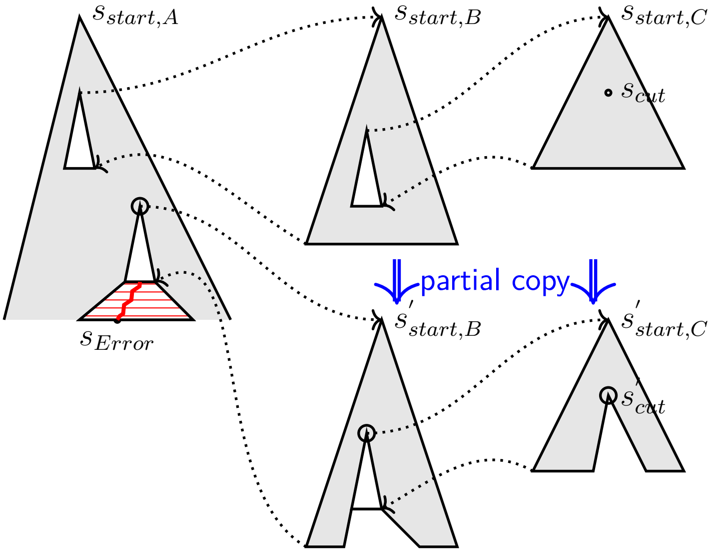
Copy-on-Write Refinement Strategy



Copy-on-Write Refinement Strategy



Copy-on-Write Refinement Strategy



Evaluation

Research questions

- ? different runtime for the analysis?
- ? different number of refinements?

Evaluation

Research questions

- ? different runtime for the analysis?
- ? different number of refinements?

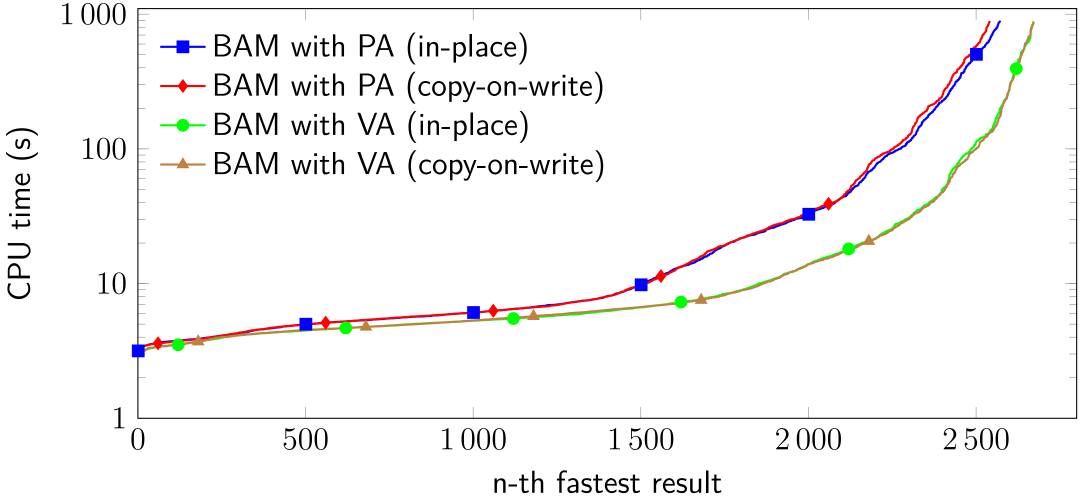
Configuration (CPAchecker r29066)

- BAM with predicate analysis (PA) and BAM with explicit-value analysis (VA)
- *in-place* vs. *copy-on-write* refinement

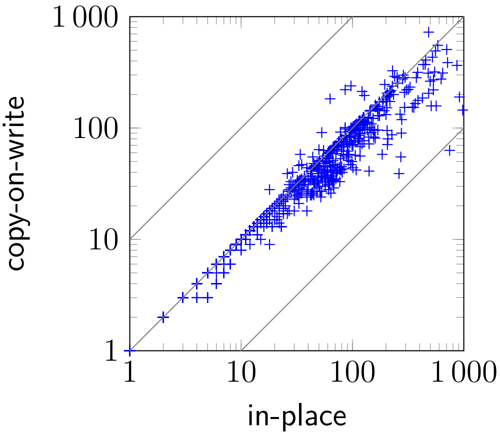
Environment and tasks

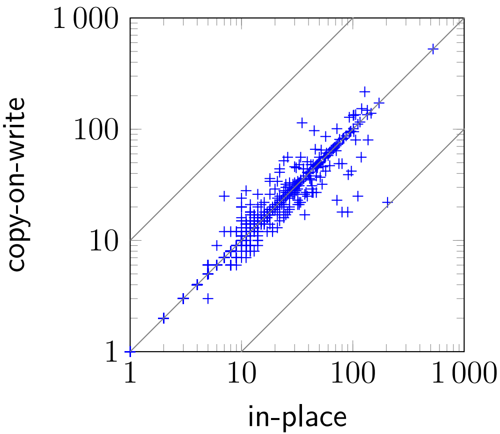
- Intel Xeon E3-1230 v5 with 3.40 GHz and 4 physical cores
- limitation of 15 GB RAM and 15 min of runtime
- 5400 tasks from SV benchmark suite

Evaluation: Runtime of Different Refinement Approaches



Evaluation: #Refinements for Different Analyses





Number of refinements for (1) predicate analysis and (2) explicit-value analysis

Interprocedural Block-Abstraction Memoization

[Beyer/Friedberger, 2020]

Challenges with (**intra**procedural) block abstractions:

- X dependent on program context
- X missing support for recursive procedures

Interprocedural Block-Abstraction Memoization

[Beyer/Friedberger, 2020]

Challenges with (**intra**procedural) block abstractions:

- ✗ dependent on program context
- ✗ missing support for recursive procedures

Our contribution: **Inter**procedural Block-Abstraction Memoization

- block abstractions are (mostly) independent of the calling context
- fixed-point algorithm for soundly analyzing recursive procedures

Interprocedural Block-Abstraction Memoization

[Beyer/Friedberger, 2020]

Challenges with (**intra**procedural) block abstractions:

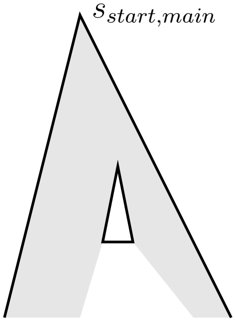
- ✗ dependent on program context
- ✗ missing support for recursive procedures

Our contribution: **Inter**procedural Block-Abstraction Memoization

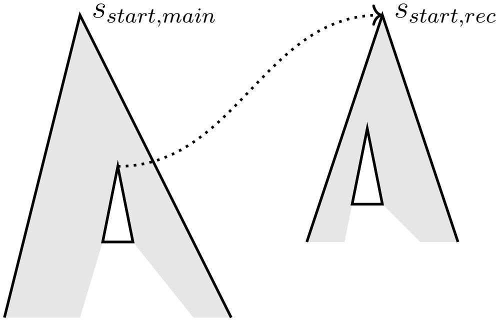
- block abstractions are (mostly) independent of the calling context
- fixed-point algorithm for soundly analyzing recursive procedures

✓ based on Intraprocedural Block-Abstraction Memoization

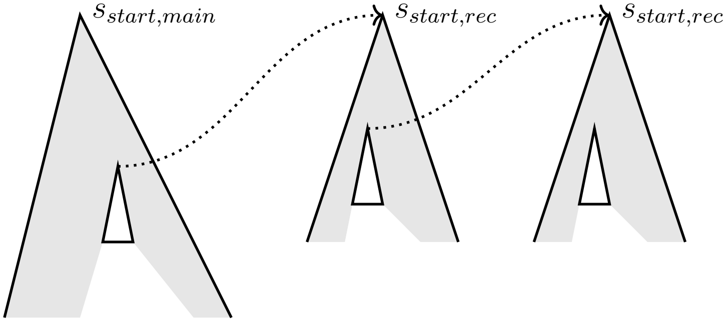
Analysis of Recursive Procedures



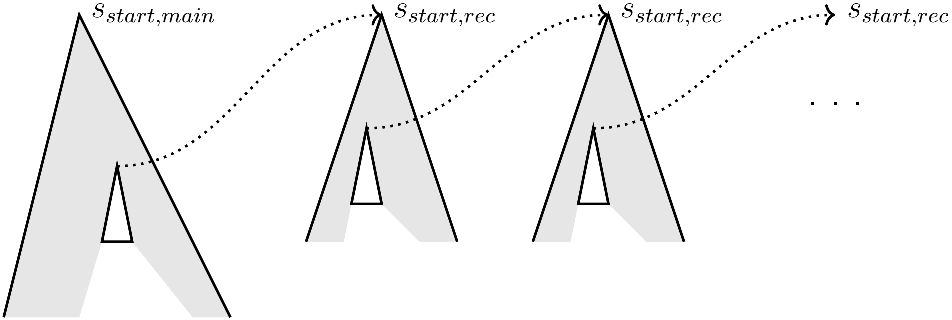
Analysis of Recursive Procedures



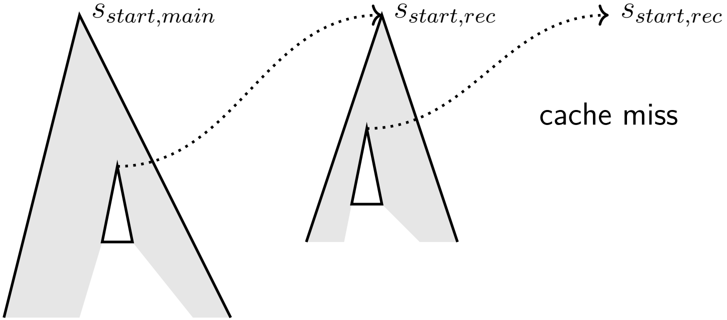
Analysis of Recursive Procedures



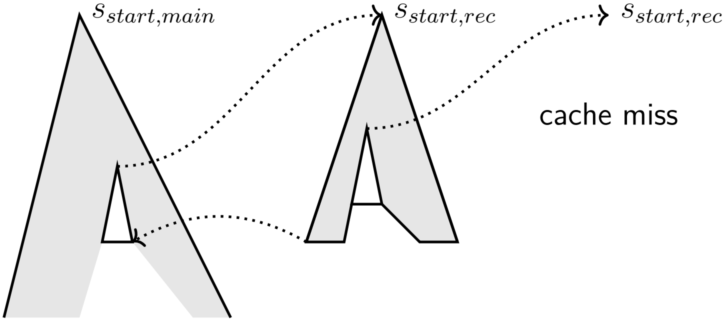
Analysis of Recursive Procedures



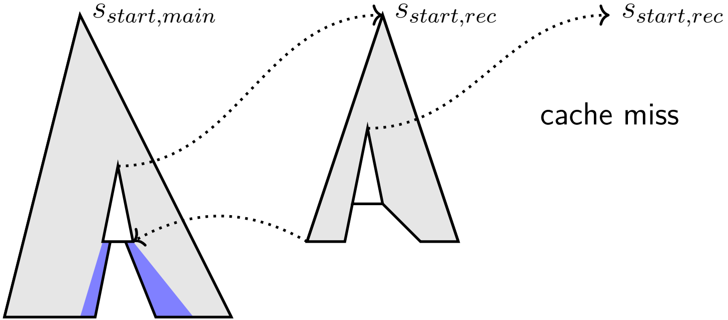
Analysis of Recursive Procedures (Interprocedural BAM)



Analysis of Recursive Procedures (Interprocedural BAM)

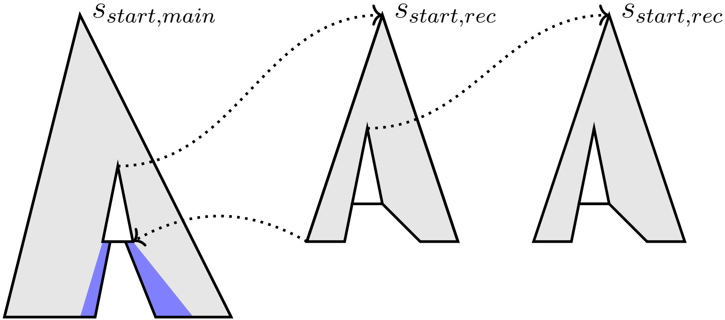


Analysis of Recursive Procedures (Interprocedural BAM)

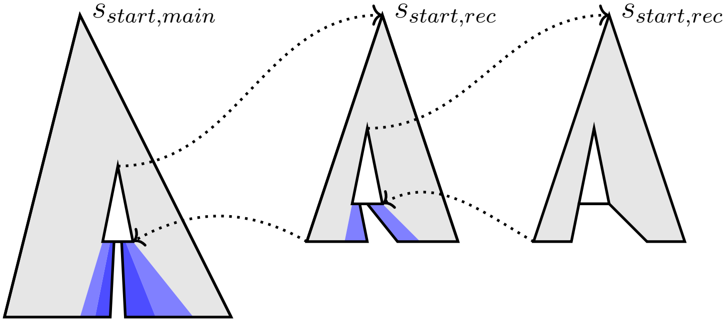


cache miss

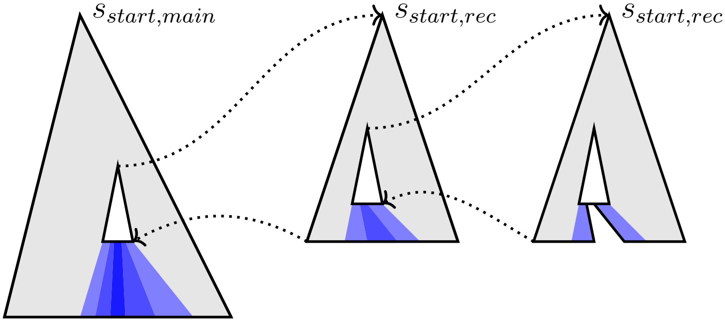
Analysis of Recursive Procedures (Interprocedural BAM)



Analysis of Recursive Procedures (Interprocedural BAM)



Analysis of Recursive Procedures (Interprocedural BAM)



Evaluation

Research questions

- ? effectiveness and efficiency against **Intra**procedural BAM
- ? effectiveness and efficiency against other state-of-the-art tools

Evaluation

Research questions

- ? effectiveness and efficiency against **Intra**procedural BAM
- ? effectiveness and efficiency against other state-of-the-art tools

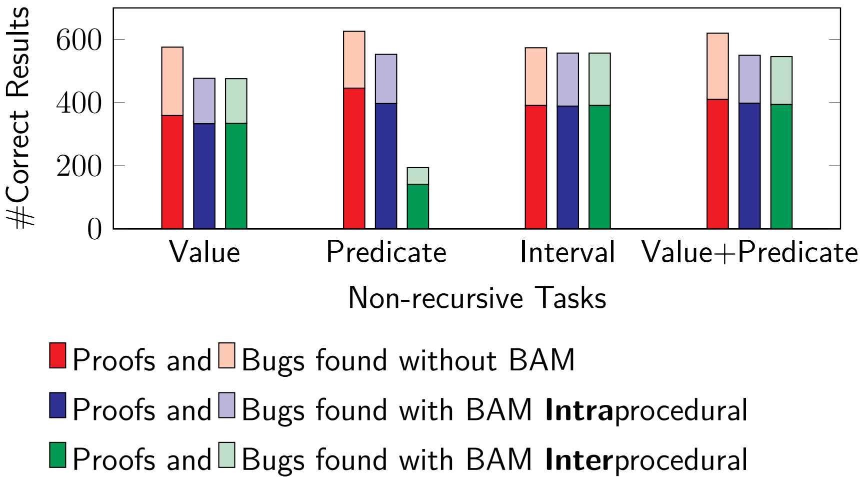
Tools and configurations

- CPAchecker v1.9 with different analyses and different domains
- several participants of SV-COMP'20

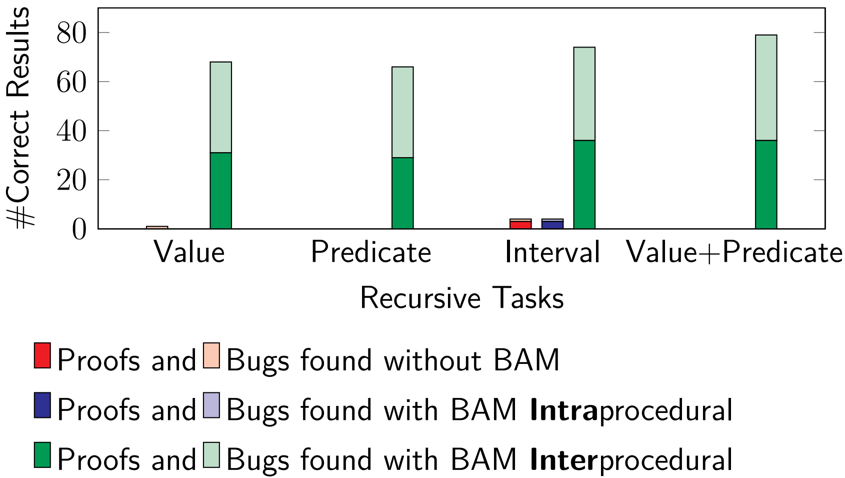
Environment and tasks

- Intel Xeon E3-1230 v5 with 3.40 GHz and 4 physical cores
- limitation of 15 GB RAM and 15 min of runtime
- $>$1000 non-recursive tasks from SV benchmark suite
- $>$100 recursive tasks from SV benchmark suite

Evaluation: CPAchecker with Different Analyses



Evaluation: CPAchecker with Different Analyses



Evaluation: CPAchecker vs. Tools of SV-COMP'20 on Recursive Tasks

Verifier	CPU time (s)	Proofs	Bugs
Cbmc	662	32	47
CPAchecker (Value+Predicate)	2 180	37	46
Divine	1 190	32	42
Esbmc	941	33	47
Map2Check	23 600	34	37
Pinaka	237	31	31
Symbiotic	138	33	45
UAutomizer	2 160	41	37
VeriAbs	7 630	41	46

Conclusion

Block Abstraction Memoization

- domain-independent approach for software verification

Parallel Block Abstraction Memoization

- simple and efficient approach

Refinement for Block Abstraction Memoization

- insights into refinement in the context of BAM

Interprocedural Block Abstraction Memoization

- support for recursive procedures
- competitive performance

Ideas and Future Research Directions

Block Abstraction Memoization

- combine with backward analysis for bi-directional state-space exploration
- more domain

Parallel Block Abstraction Memoization

- multiple processes (machines) instead of multiple threads

Interprocedural Block Abstraction Memoization

- pointer handling and heap manipulation is currently unsolved

References and Data Availability

- Domain-Independent Multi-threaded Software Model Checking
  Dirk Beyer and Karlheinz Friedberger, ASE 2018
  Supplement: https://www.sosy-lab.org/research/bam-parallel/

- In-Place vs. Copy-on-Write CEGAR Refinement for Block Summarization with Caching
  Dirk Beyer and Karlheinz Friedberger, ISoLA 2018
  Supplement: https://www.sosy-lab.org/research/bam-cow-refinement/

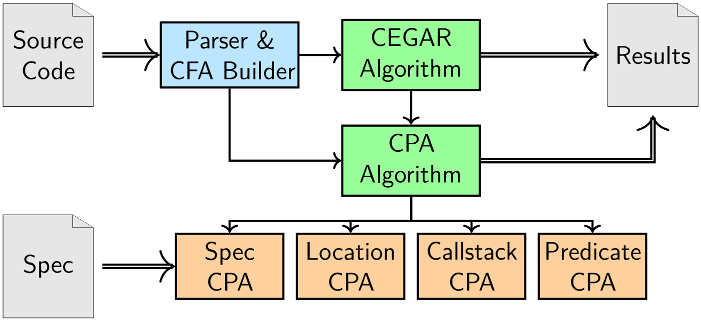
- Domain-Independent Interprocedural Program Analysis using Block-Abstraction Memoization
  Dirk Beyer and Karlheinz Friedberger, FSE 2020
  Reproduction Package: https://doi.org/10.5281/zenodo.4024268

Tools

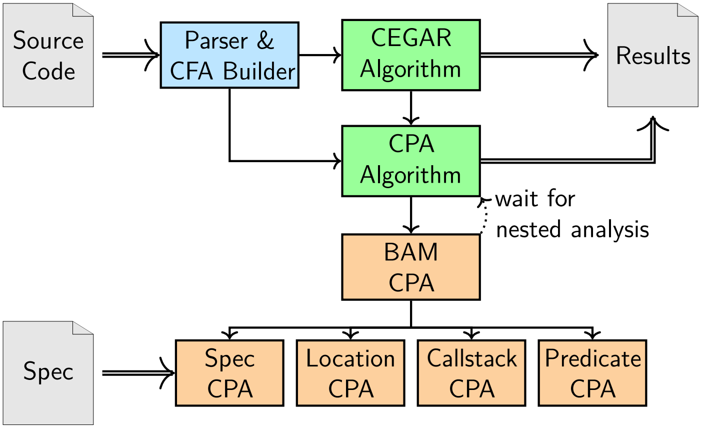
- CPAchecker - The Configurable Software-Verification Platform
  https://cpachecker.sosy-lab.org
- BenchExec - Reliable Benchmarking and Resource Measurement
  https://github.com/sosy-lab/benchexec
- JavaSMT - Unified Java API for SMT Solvers
  https://github.com/sosy-lab/java-smt
- SV-Benchmarks - Collection of Verification Tasks
  https://github.com/sosy-lab/sv-benchmarks

Questions?

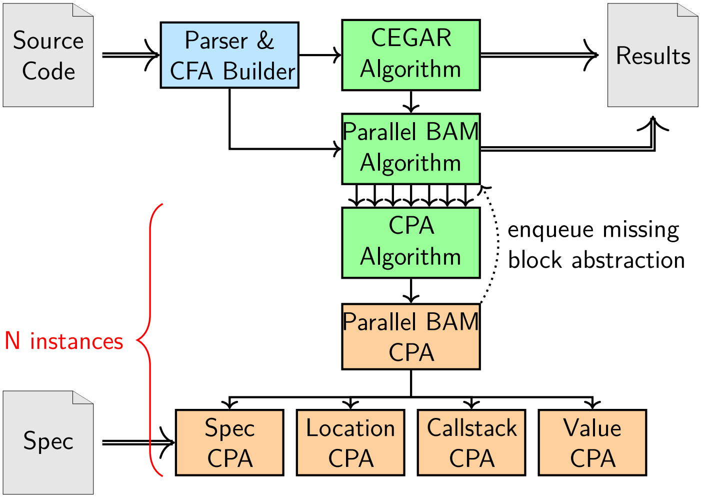
CPAchecker Framework



BAM in CPAchecker



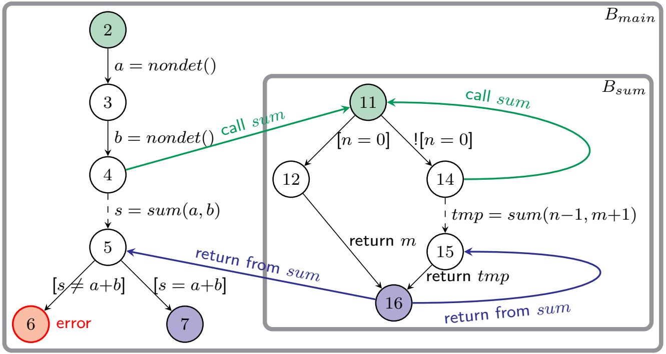
Parallel BAM in CPAchecker



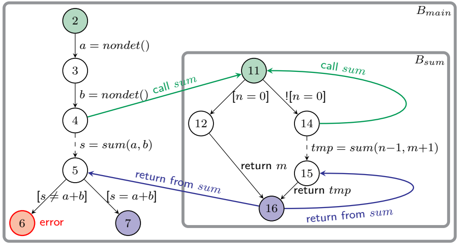
Example of an Analysis

```
void main(void) {
  uint a = nondet();
  uint b = nondet();
  uint s = sum(a, b);
  if (s != a + b) {
    error ();
  }
}

uint sum(uint n, uint m) {
  if (n == 0) {
    return m;
  } else {
    uint tmp = sum(n − 1, m + 1);
    return tmp;
  }
}
```

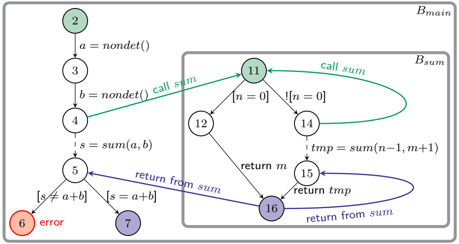


Example of an Analysis

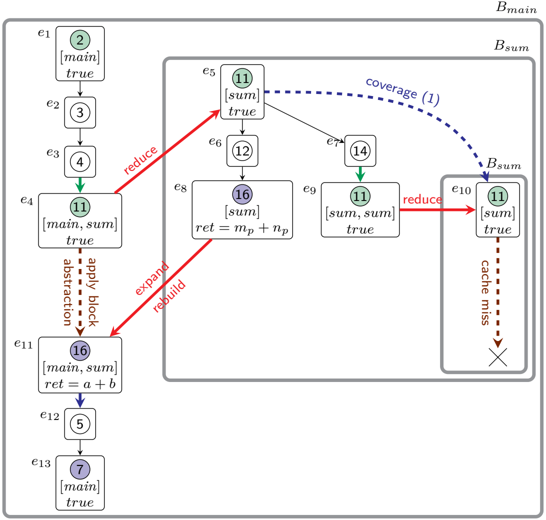


control-flow automaton

Example of an Analysis



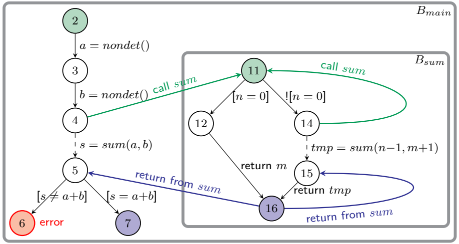
control-flow automaton



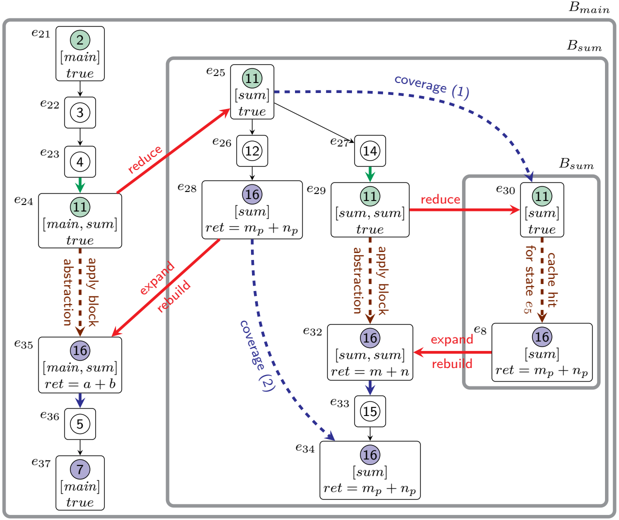
abstract reachability graph

Fixed-point algorithm
(first iteration)

Example of an Analysis



control-flow automaton



abstract reachability graph

Fixed-point algorithm
(second iteration)