Mopsa at the Software Verification Competition

Raphaël Monat

SyCoMoRES team

`rmonat.fr`

ProgLang @ Inria
9 February 2023

Introduction

Conservative static program analysis

average.py

```python
def average(l):
  m = 0
  for i in range(len(l)):
    m = m + l[i]
  m = m // (i + 1)
  return s

r1 = average([1, 2, 3])
r2 = average(['a', 'b', 'c'])
```

TypeError: unsupported operand type(s) for '+': 'int' and 'str'

argslen.c

```c
#include <string.h>

int main(int argc, char *argv[]) {
  int i = 0;
  for (char **p = argv; *p; p++) {
    strlen(*p); // valid string
    i++; // no overflow
  }
  return 0;
}
```

No alarm

Specifications of the analyzer

Inference	of program properties such as the absence of run-time errors.
Semantic	based on a formal modelization of the language.
Automatic	no expert knowledge required.
Sound	covers all possible executions.

Towards Mopsa

Well-established & industrialized analysis of static programming languages

- C: Polyspace (1999), Astrée (2003), Frama-C (2008)
- Java: Julia (2010)
- JavaScript: Jensen, Møller, and Thiemann. “Type Analysis for JavaScript”. SAS 2009

Towards Mopsa

Well-established & industrialized analysis of static programming languages

- C: Polyspace (1999), Astrée (2003), Frama-C (2008)
- Java: Julia (2010)
- JavaScript: Jensen, Møller, and Thiemann. “Type Analysis for JavaScript”. SAS 2009

What about

- Multiple languages?

Towards Mopsa

Well-established & industrialized analysis of static programming languages

- C: Polyspace (1999), Astrée (2003), Frama-C (2008)
- Java: Julia (2010)
- JavaScript: Jensen, Møller, and Thiemann. "Type Analysis for JavaScript". SAS 2009

What about

- Multiple languages?
- Common abstractions?

Towards Mopsa

Well-established & industrialized analysis of static programming languages

- C: Polyspace (1999), Astrée (2003), Frama-C (2008)
- Java: Julia (2010)
- JavaScript: Jensen, Møller, and Thiemann. “Type Analysis for JavaScript”. SAS 2009

What about

- Multiple languages?
- Common abstractions?
- Precision and configurability?

Outline

Mopsa

Overview of Mopsa

Modular **O**pen **P**latform for **S**tatic **A**nalysis[1]

`gitlab.com/mopsa/mopsa-analyzer`

Goals

[1] Journault, Miné, Monat, and Ouadjaout. "Combinations of reusable abstract domains for a multilingual static analyzer". VSTTE 2019

Overview of Mopsa

Modular **O**pen **P**latform for **S**tatic **A**nalysis[1]

`gitlab.com/mopsa/mopsa-analyzer`

Goals

- explore new designs

[1] Journault, Miné, Monat, and Ouadjaout. "Combinations of reusable abstract domains for a multilingual static analyzer". VSTTE 2019

Overview of Mopsa

Modular **O**pen **P**latform for **S**tatic **A**nalysis[1]

`gitlab.com/mopsa/mopsa-analyzer`

Goals

- explore new designs
- ease development

[1] Journault, Miné, Monat, and Ouadjaout. "Combinations of reusable abstract domains for a multilingual static analyzer". VSTTE 2019

Overview of Mopsa

Modular Open Platform for Static Analysis[1]

`gitlab.com/mopsa/mopsa-analyzer`

Goals

- explore new designs
- ease development
- support multiple languages

[1] Journault, Miné, Monat, and Ouadjaout. "Combinations of reusable abstract domains for a multilingual static analyzer". VSTTE 2019

Overview of Mopsa

Modular **O**pen **P**latform for **S**tatic **A**nalysis[1]

`gitlab.com/mopsa/mopsa-analyzer`

Goals

- explore new designs
- ease development
- support multiple languages
- loosely couple abstractions

[1] Journault, Miné, Monat, and Ouadjaout. “Combinations of reusable abstract domains for a multilingual static analyzer”. VSTTE 2019

Overview of Mopsa

Modular **O**pen **P**latform for **S**tatic **A**nalysis[1]

`gitlab.com/mopsa/mopsa-analyzer`

Goals

- explore new designs
- ease development
- support multiple languages
- loosely couple abstractions

Contributors

- Antoine Miné
- Abdelraouf Ouadjaout
- Raphaël Monat
- *David Delmas*
- *Guillaume Bau*
- *Milla Valnet*
- Matthieu Journault

[1] Journault, Miné, Monat, and Ouadjaout. "Combinations of reusable abstract domains for a multilingual static analyzer". VSTTE 2019

Current public analyses in Mopsa

Semantic property
Runtime error detection

Current public analyses in Mopsa

Semantic property
Runtime error detection

Language	Benchmark	Max. LoC	$\simeq$ Time	Selectivity
C[2]	Coreutils	550	20s	99.8%
	Juliet	340,000	2.5h	98.9%

[2] Ouadjaout and Miné. “A Library Modeling Language for the Static Analysis of C Programs”. SAS 2020

Current public analyses in Mopsa

Semantic property
Runtime error detection

Language	Benchmark	Max. LoC	$\simeq$ Time	Selectivity
C[2]	Coreutils	550	20s	99.8%
	Juliet	340,000	2.5h	98.9%
Python[3]	PyPerformance	1,792	1.3m	99.2%
	PathPicker	2,560	3.0m	99.2%

[2] Ouadjaout and Miné. “A Library Modeling Language for the Static Analysis of C Programs”. SAS 2020
[3] Monat, Ouadjaout, and Miné. “Static Type Analysis by Abstract Interpretation of Python Programs”. ECOOP 2020

Current public analyses in Mopsa

Semantic property

Runtime error detection

Language	Benchmark	Max. LoC	$\simeq$ Time	Selectivity
C[2]	Coreutils	550	20s	99.8%
	Juliet	340,000	2.5h	98.9%
Python[3]	PyPerformance	1,792	1.3m	99.2%
	PathPicker	2,560	3.0m	99.2%
Python+C[4]	ahocorasick	4,800	1.0m	98.0%
	bitarray	5,700	4.6m	94.6%

[2] Ouadjaout and Miné. “A Library Modeling Language for the Static Analysis of C Programs”. SAS 2020
[3] Monat, Ouadjaout, and Miné. “Static Type Analysis by Abstract Interpretation of Python Programs”. ECOOP 2020
[4] Monat, Ouadjaout, and Miné. “A Multilanguage Static Analysis of Python Programs with Native C Extensions”. SAS 2021

Current public analyses in Mopsa

Semantic property

Runtime error detection

mopsa-build

- Utility handling multi-file projects and compilation flags
- Significantly simplifies user experience

Language	Benchmark	Max. LoC	$\simeq$ Time	Selectivity
C[2]	Coreutils	550	20s	99.8%
	Juliet	340,000	2.5h	98.9%
Python[3]	PyPerformance	1,792	1.3m	99.2%
	PathPicker	2,560	3.0m	99.2%
Python+C[4]	ahocorasick	4,800	1.0m	98.0%
	bitarray	5,700	4.6m	94.6%

[2] Ouadjaout and Miné. “A Library Modeling Language for the Static Analysis of C Programs”. SAS 2020
[3] Monat, Ouadjaout, and Miné. “Static Type Analysis by Abstract Interpretation of Python Programs”. ECOOP 2020
[4] Monat, Ouadjaout, and Miné. “A Multilanguage Static Analysis of Python Programs with Native C Extensions”. SAS 2021

Current public analyses in Mopsa

Semantic property

Runtime error detection

mopsa-build

- Utility handling multi-file projects and compilation flags
- Significantly simplifies user experience

Language	Benchmark	Max. LoC	$\simeq$ Time	Selectivity
C[2]	Coreutils	550	20s	99.8%
	Juliet	340,000	2.5h	98.9%
Python[3]	PyPerformance	1,792	1.3m	99.2%
	PathPicker	2,560	3.0m	99.2%
Python+C[4]	ahocorasick	4,800	1.0m	98.0%
	bitarray	5,700	4.6m	94.6%

Work in progress

Analysis for recursive ADTs, presented at JFLAs last week by Milla Valnet.

[2]Ouadjaout and Miné. “A Library Modeling Language for the Static Analysis of C Programs”. SAS 2020

[3]Monat, Ouadjaout, and Miné. “Static Type Analysis by Abstract Interpretation of Python Programs”. ECOOP 2020

[4]Monat, Ouadjaout, and Miné. “A Multilanguage Static Analysis of Python Programs with Native C Extensions”. SAS 2021

SV-Comp

Presentation of SV-Comp

Software-Verification Competition

- Yearly, since 2012

Presentation of SV-Comp

Software-Verification Competition

- Yearly, since 2012
- Part of ETAPS

Presentation of SV-Comp

Software-Verification Competition

- Yearly, since 2012
- Part of ETAPS
- Organized by Dirk Beyer (Munich)

Presentation of SV-Comp

Software-Verification Competition

- Yearly, since 2012
- Part of ETAPS
- Organized by Dirk Beyer (Munich)
- 50 participating tools in 2023

Presentation of SV-Comp

Software-Verification Competition

- Yearly, since 2012
- Part of ETAPS
- Organized by Dirk Beyer (Munich)
- 50 participating tools in 2023
- Initially for model checkers

Presentation of SV-Comp (II)

Workflow

- **Input** check if a given program satisfies a property

Presentation of SV-Comp (II)

Workflow

- **Input** check if a given program satisfies a property
- **Constraints** 15 minutes CPU time, 8GB RAM

Presentation of SV-Comp (II)

Workflow

- **Input** check if a given program satisfies a property
- **Constraints** 15 minutes CPU time, 8GB RAM
- **Output** result (true, false or unknown) & witness

Presentation of SV-Comp (II)

Workflow

- **Input** check if a given program satisfies a property
- **Constraints** 15 minutes CPU time, 8GB RAM
- **Output** result (true, false or unknown) & witness
- **Scoring** discussed later

Presentation of SV-Comp (II)

Workflow

- **Input** check if a given program satisfies a property
- **Constraints** 15 minutes CPU time, 8GB RAM
- **Output** result (true, false or unknown) & witness
- **Scoring** discussed later

Programs

Presentation of SV-Comp (II)

Workflow

- **Input** check if a given program satisfies a property
- **Constraints** 15 minutes CPU time, 8GB RAM
- **Output** result (true, false or unknown) & witness
- **Scoring** discussed later

Programs

- Preprocessed C programs

Presentation of SV-Comp (II)

Workflow

- **Input** check if a given program satisfies a property
- **Constraints** 15 minutes CPU time, 8GB RAM
- **Output** result (true, false or unknown) & witness
- **Scoring** discussed later

Programs

- Preprocessed C programs
- Lots of handcrafted or small examples

Presentation of SV-Comp (II)

Workflow

- **Input** check if a given program satisfies a property
- **Constraints** 15 minutes CPU time, 8GB RAM
- **Output** result (true, false or unknown) & witness
- **Scoring** discussed later

Programs

- Preprocessed C programs
- Lots of handcrafted or small examples
- “SoftwareSystems” category, more realistic

Presentation of SV-Comp (II)

Workflow

- **Input** check if a given program satisfies a property
- **Constraints** 15 minutes CPU time, 8GB RAM
- **Output** result (true, false or unknown) & witness
- **Scoring** discussed later

Programs

- Preprocessed C programs
- Lots of handcrafted or small examples
- “SoftwareSystems” category, more realistic
- Community-curated

Presentation of SV-Comp (II)

Workflow

- **Input** check if a given program satisfies a property
- **Constraints** 15 minutes CPU time, 8GB RAM
- **Output** result (true, false or unknown) & witness
- **Scoring** discussed later

Programs

- Preprocessed C programs
- Lots of handcrafted or small examples
- “SoftwareSystems” category, more realistic
- Community-curated

Properties

Presentation of SV-Comp (II)

Workflow

- **Input** check if a given program satisfies a property
- **Constraints** 15 minutes CPU time, 8GB RAM
- **Output** result (true, false or unknown) & witness
- **Scoring** discussed later

Programs

- Preprocessed C programs
- Lots of handcrafted or small examples
- “SoftwareSystems” category, more realistic
- Community-curated

Properties

- Reachability

Presentation of SV-Comp (II)

Workflow

- **Input** check if a given program satisfies a property
- **Constraints** 15 minutes CPU time, 8GB RAM
- **Output** result (true, false or unknown) & witness
- **Scoring** discussed later

Programs

- Preprocessed C programs
- Lots of handcrafted or small examples
- “SoftwareSystems” category, more realistic
- Community-curated

Properties

- Reachability
- Memory safety

Presentation of SV-Comp (II)

Workflow

- **Input** check if a given program satisfies a property
- **Constraints** 15 minutes CPU time, 8GB RAM
- **Output** result (true, false or unknown) & witness
- **Scoring** discussed later

Programs

- Preprocessed C programs
- Lots of handcrafted or small examples
- “SoftwareSystems” category, more realistic
- Community-curated

Properties

- Reachability
- Memory safety
- Integer overflows

Presentation of SV-Comp (II)

Workflow

- **Input** check if a given program satisfies a property
- **Constraints** 15 minutes CPU time, 8GB RAM
- **Output** result (true, false or unknown) & witness
- **Scoring** discussed later

Programs

- Preprocessed C programs
- Lots of handcrafted or small examples
- “SoftwareSystems” category, more realistic
- Community-curated

Properties

- Reachability
- Memory safety
- Integer overflows
- Termination

Presentation of SV-Comp (II)

Workflow

- **Input** check if a given program satisfies a property
- **Constraints** 15 minutes CPU time, 8GB RAM
- **Output** result (true, false or unknown) & witness
- **Scoring** discussed later

Programs

- Preprocessed C programs
- Lots of handcrafted or small examples
- “SoftwareSystems” category, more realistic
- Community-curated

Properties

- Reachability
- Memory safety
- Integer overflows
- Termination
- Data race

Presentation of SV-Comp (III)

Category	# tasks	Median loc.
ReachSafety	6282	1267
MemSafety	6280	86
ConcurrencySafety	2370	127
NoOverflows	6539	49
Termination	3324	901
SoftwareSystems	5825	6655

Subcategories in SoftwareSystems

Presentation of SV-Comp (III)

Category	# tasks	Median loc.
ReachSafety	6282	1267
MemSafety	6280	86
ConcurrencySafety	2370	127
NoOverflows	6539	49
Termination	3324	901
SoftwareSystems	5825	6655

Subcategories in SoftwareSystems

- AWS C commons

Presentation of SV-Comp (III)

Category	# tasks	Median loc.
ReachSafety	6282	1267
MemSafety	6280	86
ConcurrencySafety	2370	127
NoOverflows	6539	49
Termination	3324	901
SoftwareSystems	5825	6655

Subcategories in SoftwareSystems

- AWS C commons
- BusyBox (coreutils alternative)

Presentation of SV-Comp (III)

Category	# tasks	Median loc.
ReachSafety	6282	1267
MemSafety	6280	86
ConcurrencySafety	2370	127
NoOverflows	6539	49
Termination	3324	901
SoftwareSystems	5825	6655

Subcategories in SoftwareSystems

- AWS C commons
- BusyBox (coreutils alternative)
- Linux Device Drivers

Presentation of SV-Comp (III)

Category	# tasks	Median loc.
ReachSafety	6282	1267
MemSafety	6280	86
ConcurrencySafety	2370	127
NoOverflows	6539	49
Termination	3324	901
SoftwareSystems	5825	6655

Subcategories in SoftwareSystems

- AWS C commons
- BusyBox (coreutils alternative)
- Linux Device Drivers
- OpenBSD

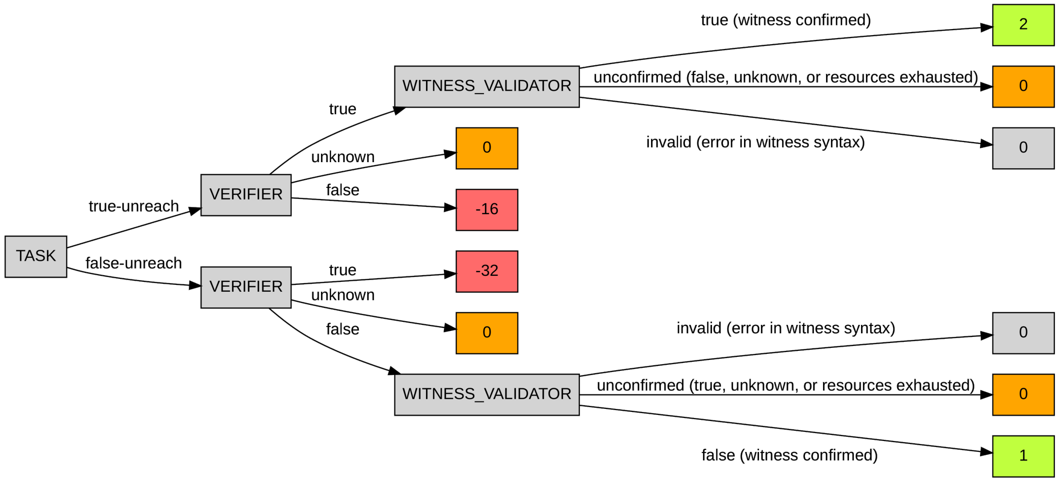
Presentation of SV-Comp (III)

Category	# tasks	Median loc.
ReachSafety	6282	1267
MemSafety	6280	86
ConcurrencySafety	2370	127
NoOverflows	6539	49
Termination	3324	901
SoftwareSystems	5825	6655

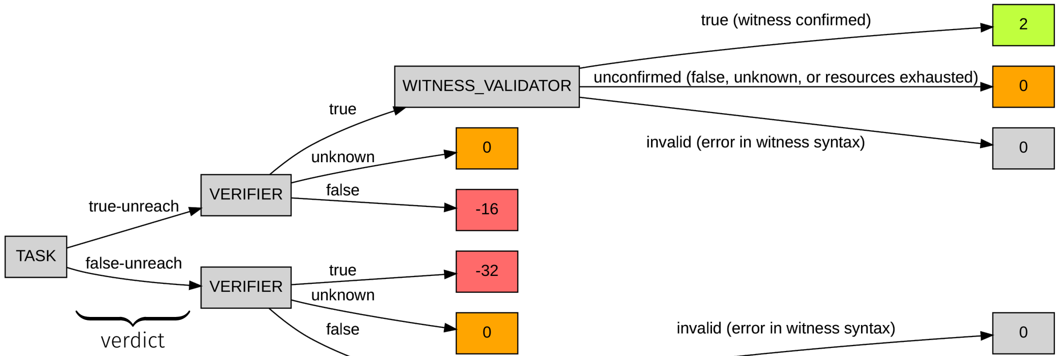
Subcategories in SoftwareSystems

- AWS C commons
- BusyBox (coreutils alternative)
- Linux Device Drivers
- OpenBSD
- uthash

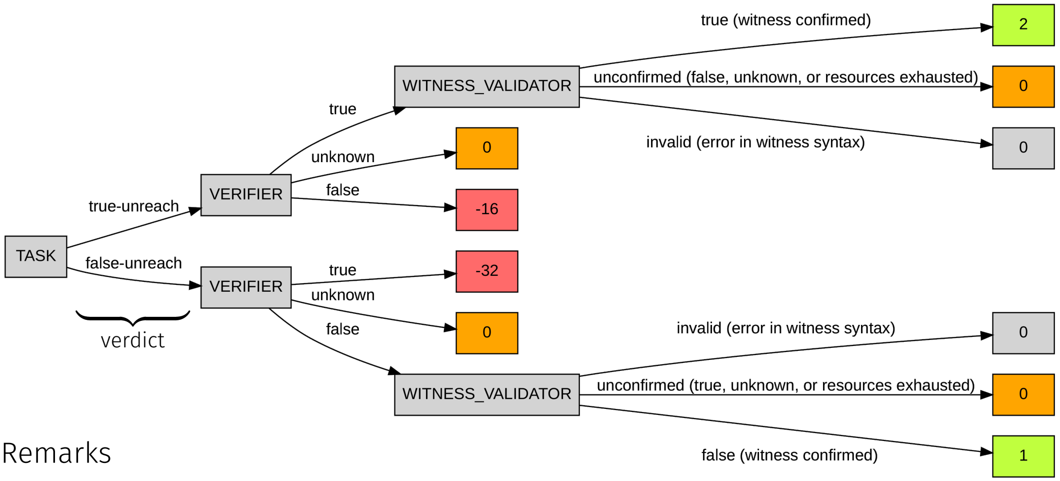
SV-Comp's Scoring System


TASK
true-unreach
false-unreach
VERIFIER
true
unknown
false
WITNESS_VALIDATOR
true (witness confirmed)
2
unconfirmed (false, unknown, or resources exhausted)
0
invalid (error in witness syntax)
0
0
-16
VERIFIER
true
-32
unknown
0
false
WITNESS_VALIDATOR
invalid (error in witness syntax)
0
unconfirmed (true, unknown, or resources exhausted)
0
false (witness confirmed)
1

SV-Comp's Scoring System


TASK
true-unreach
false-unreach
VERIFIER
VERIFIER
true
unknown
false
WITNESS_VALIDATOR
0
-16
true (witness confirmed)
2
unconfirmed (false, unknown, or resources exhausted)
0
invalid (error in witness syntax)
0
true
unknown
false
-32
0
WITNESS_VALIDATOR
invalid (error in witness syntax)
0
unconfirmed (true, unknown, or resources exhausted)
0
false (witness confirmed)
1
verdict

SV-Comp's Scoring System


TASK
true-unreach
false-unreach
verdict
VERIFIER
true
unknown
false
WITNESS_VALIDATOR
true (witness confirmed)
2
unconfirmed (false, unknown, or resources exhausted)
0
invalid (error in witness syntax)
0
0
-16
VERIFIER
true
-32
unknown
0
false
WITNESS_VALIDATOR
invalid (error in witness syntax)
0
unconfirmed (true, unknown, or resources exhausted)
0
false (witness confirmed)
1

Remarks

- community-based curation of verdicts
- 187 manual fixes on my end

SV-Comp’s Scoring System (II)

Categories are divided into subcategories (a family of benchmarks).

SV-Comp's Scoring System (II)

Categories are divided into subcategories (a family of benchmarks).

Scoring incentive for balanced results among subcategories.

$$\text{overall score} \propto \sum_{s \in \text{subCategory}} \frac{\text{raw score in s}}{\#\ \text{tasks in s}}$$

SV-Comp’s Scoring System (II)

Categories are divided into subcategories (a family of benchmarks).

Scoring incentive for balanced results among subcategories.

$$\text{overall score} \propto \sum_{s \in \text{subCategory}} \frac{\text{raw score in s}}{\#\ \text{tasks in s}}$$

You may have a high raw score but not so good overall score.

SV-Comp's "Witnesses"

Motivation

- Ensure that results can be validated, at a reduced computational cost

[5] Saan. Witness Generation for Data-flow Analysis. 2020.
[6] Beyer, Dangl, Dietsch, Heizmann, Lemberger, and Tautschnig. "Verification Witnesses". 2022.

SV-Comp's "Witnesses"

Motivation

- Ensure that results can be validated, at a reduced computational cost
- Improve interoperability between verifiers?

[5] Saan. Witness Generation for Data-flow Analysis. 2020.
[6] Beyer, Dangl, Dietsch, Heizmann, Lemberger, and Tautschnig. "Verification Witnesses". 2022.

SV-Comp's "Witnesses"

Motivation

- Ensure that results can be validated, at a reduced computational cost
- Improve interoperability between verifiers?

Witnesses

Automata where edges contain program invariants and control choices

[5] Saan. Witness Generation for Data-flow Analysis. 2020.
[6] Beyer, Dangl, Dietsch, Heizmann, Lemberger, and Tautschnig. "Verification Witnesses". 2022.

SV-Comp's "Witnesses"

Motivation

- Ensure that results can be validated, at a reduced computational cost
- Improve interoperability between verifiers?

Witnesses

Automata where edges contain program invariants and control choices

Issues (in my opinion)

[5] Saan. Witness Generation for Data-flow Analysis. 2020.

[6] Beyer, Dangl, Dietsch, Heizmann, Lemberger, and Tautschnig. "Verification Witnesses". 2022.

SV-Comp's "Witnesses"

Motivation

- Ensure that results can be validated, at a reduced computational cost
- Improve interoperability between verifiers?

Witnesses

Automata where edges contain program invariants and control choices

Issues (in my opinion)

- Interprocedural encoding to be improved[5]

[5] Saan. Witness Generation for Data-flow Analysis. 2020.

[6] Beyer, Dangl, Dietsch, Heizmann, Lemberger, and Tautschnig. "Verification Witnesses". 2022.

SV-Comp's "Witnesses"

Motivation

- Ensure that results can be validated, at a reduced computational cost
- Improve interoperability between verifiers?

Witnesses

Automata where edges contain program invariants and control choices

Issues (in my opinion)

- Interprocedural encoding to be improved[5]
- Cross-validator scores can be low[6] – 45%

[5] Saan. Witness Generation for Data-flow Analysis. 2020.
[6] Beyer, Dangl, Dietsch, Heizmann, Lemberger, and Tautschnig. "Verification Witnesses". 2022.

SV-Comp's "Witnesses"

Motivation

- Ensure that results can be validated, at a reduced computational cost
- Improve interoperability between verifiers?

Witnesses

Automata where edges contain program invariants and control choices

Issues (in my opinion)

- Interprocedural encoding to be improved[5]
- Cross-validator scores can be low[6] – 45%
- 96.4% of Mopsa's trivial witnesses are validated

[5] Saan. Witness Generation for Data-flow Analysis. 2020.

[6] Beyer, Dangl, Dietsch, Heizmann, Lemberger, and Tautschnig. "Verification Witnesses". 2022.

Mopsa at SV-Comp

Adapting Mopsa to SV-Comp’s Framework

Our approach

1. Analyze the target program with Mopsa

Adapting Mopsa to SV-Comp's Framework

Our approach

1. Analyze the target program with Mopsa
2. Postprocess Mopsa's result to decide whether the property of interest holds

Adapting Mopsa to SV-Comp's Framework

Our approach

1. Analyze the target program with Mopsa
2. Postprocess Mopsa's result to decide whether the property of interest holds
    - **Yes?** finished!

Adapting Mopsa to SV-Comp's Framework

Our approach

1. Analyze the target program with Mopsa
2. Postprocess Mopsa's result to decide whether the property of interest holds
    - **Yes?** finished!
    - **No?** restart with a more precise analysis

Adapting Mopsa to SV-Comp's Framework

Our approach

1. Analyze the target program with Mopsa
2. Postprocess Mopsa's result to decide whether the property of interest holds
    - **Yes?** finished!
    - **No?** restart with a more precise analysis

Suboptimal strategy

Adapting Mopsa to SV-Comp's Framework

Our approach

1. Analyze the target program with Mopsa
2. Postprocess Mopsa's result to decide whether the property of interest holds
   - **Yes?** finished!
   - **No?** restart with a more precise analysis

Suboptimal strategy

- Task: decide if a property holds on a program

Adapting Mopsa to SV-Comp's Framework

Our approach

1. Analyze the target program with Mopsa
2. Postprocess Mopsa's result to decide whether the property of interest holds
    - **Yes?** finished!
    - **No?** restart with a more precise analysis

Suboptimal strategy

- Task: decide if a property holds on a program

  But Mopsa analyzes full programs and detects <u>all</u> runtime errors

Adapting Mopsa to SV-Comp's Framework

Our approach

1. Analyze the target program with Mopsa
2. Postprocess Mopsa's result to decide whether the property of interest holds
   - **Yes?** finished!
   - **No?** restart with a more precise analysis

Suboptimal strategy

- Task: decide if a property holds on a program

  But Mopsa analyzes full programs and detects <u>all</u> runtime errors

  $\Longrightarrow$ We could at least add slicing

Adapting Mopsa to SV-Comp's Framework

Our approach

1. Analyze the target program with Mopsa
2. Postprocess Mopsa's result to decide whether the property of interest holds
    - **Yes?** finished!
    - **No?** restart with a more precise analysis

Suboptimal strategy

- Task: decide if a property holds on a program
  But Mopsa analyzes full programs and detects <u>all</u> runtime errors
  $\Longrightarrow$ We could at least add slicing
- New analyses restart from scratch

Portfolio of analyses used

Analyses used

1. Intervals, small structs initialized

Portfolio of analyses used

Analyses used

1. Intervals, small structs initialized
2. + string-length domain, medium structs initialized

Portfolio of analyses used

Analyses used

1. Intervals, small structs initialized
2. + string-length domain, medium structs initialized
3. + polyhedra with static packing

Portfolio of analyses used

Analyses used

1. Intervals, small structs initialized
2. + string-length domain, medium structs initialized
3. + polyhedra with static packing
4. + congruences & widening tweaks: thresholds, delay

Portfolio of analyses used

Analyses used

1. Intervals, small structs initialized
2. + string-length domain, medium structs initialized
3. + polyhedra with static packing
4. + congruences & widening tweaks: thresholds, delay

Conf.	✓		🕓	
1	5695		279	
2	6433	(+738)	365	(+86)
3	6885	(+452)	1844	(+1479)
4	6909	(+24)	2009	(+165)

21220 tasks in total, 12636 correctness tasks

Portfolio of analyses used

Analyses used

1. Intervals, small structs initialized
2. + string-length domain, medium structs initialized
3. + polyhedra with static packing
4. + congruences & widening tweaks: thresholds, delay

Conf.	✓		🕓	
1	5695		279	
2	6433	(+738)	365	(+86)
3	6885	(+452)	1844	(+1479)
4	6909	(+24)	2009	(+165)

21220 tasks in total, 12636 correctness tasks

Mopsa validates 54% of correct tasks (61% for overall winner, UAutomizer).

Mopsa’s Results

`https://sv-comp.sosy-lab.org/2023/results/`

Mopsa’s Results

`https://sv-comp.sosy-lab.org/2023/results/`

Reachability

Mopsa scores a bit below Goblint.[7]

Might be a bad configuration choice?

[7] other active abstract interpreter

Mopsa’s Results

`https://sv-comp.sosy-lab.org/2023/results/`

Reachability

Mopsa scores a bit below Goblint.[7]

Might be a bad configuration choice?

Memory

Mopsa is the only abstract interpreter participating in this category.

[7] other active abstract interpreter

Mopsa's Results

`https://sv-comp.sosy-lab.org/2023/results/`

Reachability

Mopsa scores a bit below Goblint.[7]

Might be a bad configuration choice?

Memory

Mopsa is the only abstract interpreter participating in this category.

Overflow

Ranks 6th/19, before Frama-C and Goblint.

Mopsa is on par with the winner for the number of programs proved correct!

[7] other active abstract interpreter

Mopsa’s Results

Bronze medal in the *SoftwareSystems* category!

Mopsa’s Results

Bronze medal in the *SoftwareSystems* category!

19 participants.

Mopsa’s Results

Bronze medal in the *SoftwareSystems* category!

19 participants. First French participation.

Mopsa’s Results

Bronze medal in the *SoftwareSystems* category!

19 participants. First French participation.

Verifier	Bubaak	CPAchecker	Goblint	Mopsa	Symbiotic	Ultimate
Proved correct	291	**1,651**	1,256	1,610	942	1,423
Proved incorrect	143	59	0	0	84	2
CPU Time (s)	2,000,000	730,000	800,000	580,000	**400,000**	1,400,000
Rank	2	6	10	3	**1**	7

Mopsa’s Results

Bronze medal in the *SoftwareSystems* category!

19 participants. First French participation.

Verifier	Bubaak	CPAchecker	Goblint	Mopsa	Symbiotic	Ultimate
Proved correct	291	**1,651**	1,256	1,610	942	1,423
Proved incorrect	143	59	0	0	84	2
CPU Time (s)	2,000,000	730,000	800,000	580,000	**400,000**	1,400,000
Rank	2	6	10	3	**1**	7

Mopsa ranks second on raw scores.

Benefits of participation

- Fun! (up-to exhaustion)

Benefits of participation

- Fun! (up-to exhaustion)
- Good time for software improvements

Benefits of participation

- Fun! (up-to exhaustion)
- Good time for software improvements
  - 20 issues fixed

Benefits of participation

- Fun! (up-to exhaustion)
- Good time for software improvements
  - 20 issues fixed
  - We already have a 2024 feature wishlist

Benefits of participation

- Fun! (up-to exhaustion)
- Good time for software improvements
  - 20 issues fixed
  - We already have a 2024 feature wishlist
- Interaction and comparison with other tools from a broad community

Benefits of participation

- Fun! (up-to exhaustion)
- Good time for software improvements
  - 20 issues fixed
  - We already have a 2024 feature wishlist
- Interaction and comparison with other tools from a broad community
- Better understanding of the benchmarks

Benefits of participation

- Fun! (up-to exhaustion)
- Good time for software improvements
  - 20 issues fixed
  - We already have a 2024 feature wishlist
- Interaction and comparison with other tools from a broad community
- Better understanding of the benchmarks
  - Becoming a de facto standard

Benefits of participation

- Fun! (up-to exhaustion)
- Good time for software improvements
  - 20 issues fixed
  - We already have a 2024 feature wishlist
- Interaction and comparison with other tools from a broad community
- Better understanding of the benchmarks
  - Becoming a de facto standard
  - Always ongoing benchmark curation

Benefits of participation

- Fun! (up-to exhaustion)
- Good time for software improvements
  - 20 issues fixed
  - We already have a 2024 feature wishlist
- Interaction and comparison with other tools from a broad community
- Better understanding of the benchmarks
  - Becoming a de facto standard
  - Always ongoing benchmark curation
- Brings new research questions

Conclusion

Conclusion

Mopsa as a stable academic static analyzer,

Conclusion

Mopsa as a stable academic static analyzer,
able to analyze C and Python programs,

Conclusion

Mopsa as a stable academic static analyzer,
able to analyze C and Python programs,
competing with cutting-edge verifiers.

Conclusion

Mopsa as a stable academic static analyzer,
able to analyze C and Python programs,
competing with cutting-edge verifiers.

Some SV-Comp related research questions

- Best configuration to analyze a given program under resource constraints

Conclusion

Mopsa as a stable academic static analyzer,
able to analyze C and Python programs,
competing with cutting-edge verifiers.

Some SV-Comp related research questions

- Best configuration to analyze a given program under resource constraints
- Synergy with symbolic execution tools

Mopsa at the Software Verification Competition

Raphaël Monat

SyCoMoRES team

`rmonat.fr`

ProgLang @ Inria
9 February 2023