Mopsa at the Software Verification Competition

Raphaël Monat
SyCoMoRES team
rmonat.fr
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.
Well-established & industrialized analysis of static programming languages

- Java: Julia (2010)
Well-established & industrialized analysis of static programming languages

- Java: Julia (2010)

What about

- Multiple languages?
Well-established & industrialized analysis of static programming languages

- Java: Julia (2010)

What about

- Multiple languages?
- Common abstractions?
Towards Mopsa

Well-established & industrialized analysis of static programming languages

- Java: Julia (2010)

What about

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

1. Introduction
2. Mopsa
3. SV-Comp
4. Mopsa at SV-Comp
5. Conclusion
Mopsa
Overview of Mopsa

**Modular Open Platform for Static Analysis**

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 Open Platform for Static Analysis**

[gitlab.com/mopsa/mopsa-analyzer](https://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 Open Platform for Static Analysis

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**

*gitlab.com/mopsa/mopsa-analyzer*

### Goals

<table>
<thead>
<tr>
<th>◀️</th>
<th>explore new designs</th>
<th>◀️</th>
<th>support multiple languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>◀️</td>
<td>ease development</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

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**

[gitlab.com/mopsa/mopsa-analyzer](gitlab.com/mopsa/mopsa-analyzer)

### Goals

<table>
<thead>
<tr>
<th>Goals</th>
<th>Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>- explore new designs</td>
<td>- support multiple languages</td>
</tr>
<tr>
<td>- ease development</td>
<td>- loosely couple abstractions</td>
</tr>
</tbody>
</table>

---

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**

[gitlab.com/mopsa/mopsa-analyzer](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

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

<table>
<thead>
<tr>
<th>Language</th>
<th>Benchmark Max. LoC</th>
<th>Time</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coreutils</td>
<td>550</td>
<td>20s</td>
<td>99.8%</td>
</tr>
<tr>
<td>Juliet</td>
<td>340,000</td>
<td>2.5h</td>
<td>98.9%</td>
</tr>
<tr>
<td>Python</td>
<td>1,792</td>
<td>1.3m</td>
<td>99.2%</td>
</tr>
<tr>
<td>PathPicker</td>
<td>2,560</td>
<td>3.0m</td>
<td>99.2%</td>
</tr>
<tr>
<td>Python+C</td>
<td>ahocorasick 4,800</td>
<td>1.0m</td>
<td>98.0%</td>
</tr>
<tr>
<td></td>
<td>bitarray 5,700</td>
<td>4.6m</td>
<td>94.6%</td>
</tr>
</tbody>
</table>

Work in progress
- Analysis for recursive ADTs, presented at JFLAs last week by Milla Valnet.


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

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

<table>
<thead>
<tr>
<th>Language</th>
<th>Benchmark</th>
<th>Max. LoC</th>
<th>≈ Time</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>C²</td>
<td>Coreutils</td>
<td>550</td>
<td>20s</td>
<td>99.8%</td>
</tr>
<tr>
<td>Juliet</td>
<td>340,000</td>
<td>2.5h</td>
<td>98.9%</td>
<td></td>
</tr>
</tbody>
</table>

Current public analyses in Mopsa

<table>
<thead>
<tr>
<th>Language</th>
<th>Benchmark</th>
<th>Max. LoC</th>
<th>Time</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>C²</td>
<td>Coreutils</td>
<td>550</td>
<td>20s</td>
<td>99.8%</td>
</tr>
<tr>
<td></td>
<td>Juliet</td>
<td>340,000</td>
<td>2.5h</td>
<td>98.9%</td>
</tr>
<tr>
<td>Python³</td>
<td>PyPerformance</td>
<td>1,792</td>
<td>1.3m</td>
<td>99.2%</td>
</tr>
<tr>
<td></td>
<td>PathPicker</td>
<td>2,560</td>
<td>3.0m</td>
<td>99.2%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Language</th>
<th>Benchmark</th>
<th>Max. LoC</th>
<th>≈ Time</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>C²</td>
<td>Coreutils</td>
<td>550</td>
<td>20s</td>
<td>99.8%</td>
</tr>
<tr>
<td></td>
<td>Juliet</td>
<td>340,000</td>
<td>2.5h</td>
<td>98.9%</td>
</tr>
<tr>
<td>Python³</td>
<td>PyPerformance</td>
<td>1,792</td>
<td>1.3m</td>
<td>99.2%</td>
</tr>
<tr>
<td></td>
<td>PathPicker</td>
<td>2,560</td>
<td>3.0m</td>
<td>99.2%</td>
</tr>
<tr>
<td>Python+C⁴</td>
<td>ahocorasick</td>
<td>4,800</td>
<td>1.0m</td>
<td>98.0%</td>
</tr>
<tr>
<td></td>
<td>bitarray</td>
<td>5,700</td>
<td>4.6m</td>
<td>94.6%</td>
</tr>
</tbody>
</table>

---

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

<table>
<thead>
<tr>
<th>Language</th>
<th>Benchmark</th>
<th>Max. LoC</th>
<th>≈ Time</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>C²</td>
<td>Coreutils</td>
<td>550</td>
<td>20s</td>
<td>99.8%</td>
</tr>
<tr>
<td></td>
<td>Juliet</td>
<td>340,000</td>
<td>2.5h</td>
<td>98.9%</td>
</tr>
<tr>
<td>Python³</td>
<td>PyPerformance</td>
<td>1,792</td>
<td>1.3m</td>
<td>99.2%</td>
</tr>
<tr>
<td></td>
<td>PathPicker</td>
<td>2,560</td>
<td>3.0m</td>
<td>99.2%</td>
</tr>
<tr>
<td>Python+C⁴</td>
<td>ahocorasick</td>
<td>4,800</td>
<td>1.0m</td>
<td>98.0%</td>
</tr>
<tr>
<td></td>
<td>bitarray</td>
<td>5,700</td>
<td>4.6m</td>
<td>94.6%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Language</th>
<th>Benchmark</th>
<th>Max. LoC</th>
<th>Approx. Time</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>C²</td>
<td>Coreutils</td>
<td>550</td>
<td>20s</td>
<td>99.8%</td>
</tr>
<tr>
<td></td>
<td>Juliet</td>
<td>340,000</td>
<td>2.5h</td>
<td>98.9%</td>
</tr>
<tr>
<td>Python³</td>
<td>PyPerformance</td>
<td>1,792</td>
<td>1.3m</td>
<td>99.2%</td>
</tr>
<tr>
<td></td>
<td>PathPicker</td>
<td>2,560</td>
<td>3.0m</td>
<td>99.2%</td>
</tr>
<tr>
<td>Python+C⁴</td>
<td>ahocorasick</td>
<td>4,800</td>
<td>1.0m</td>
<td>98.0%</td>
</tr>
<tr>
<td></td>
<td>bitarray</td>
<td>5,700</td>
<td>4.6m</td>
<td>94.6%</td>
</tr>
</tbody>
</table>

**Work in progress**
Analysis for recursive ADTs, presented at JFLAs last week by Milla Valnet.

---

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
Software-Verification Competition

▶ Yearly, since 2012
Software-Verification Competition

- Yearly, since 2012
- Part of ETAPS
Software-Verification Competition

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

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

Programs

- Preprocessed C programs
- Lots of handcrafted or small examples
- "Software Systems" category, more realistic
- Community-curated

Properties

- Reachability
- Memory safety
- Integer overflows
- Termination
- Data race
## 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 (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)

<table>
<thead>
<tr>
<th>Workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong> check if a given program satisfies a property</td>
</tr>
<tr>
<td><strong>Constraints</strong> 15 minutes CPU time, 8GB RAM</td>
</tr>
<tr>
<td><strong>Output</strong> result (true, false or unknown) &amp; witness</td>
</tr>
<tr>
<td><strong>Scoring</strong> discussed later</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preprocessed C programs</td>
</tr>
</tbody>
</table>
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
### 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

Properties

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

<table>
<thead>
<tr>
<th>Category</th>
<th># tasks</th>
<th>Median loc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReachSafety</td>
<td>6282</td>
<td>1267</td>
</tr>
<tr>
<td>MemSafety</td>
<td>6280</td>
<td>86</td>
</tr>
<tr>
<td>ConcurrencySafety</td>
<td>2370</td>
<td>127</td>
</tr>
<tr>
<td>NoOverflows</td>
<td>6539</td>
<td>49</td>
</tr>
<tr>
<td>Termination</td>
<td>3324</td>
<td>901</td>
</tr>
<tr>
<td>SoftwareSystems</td>
<td>5825</td>
<td>6655</td>
</tr>
</tbody>
</table>

Subcategories in SoftwareSystems
## Presentation of SV-Comp (III)

<table>
<thead>
<tr>
<th>Category</th>
<th># tasks</th>
<th>Median loc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReachSafety</td>
<td>6282</td>
<td>1267</td>
</tr>
<tr>
<td>MemSafety</td>
<td>6280</td>
<td>86</td>
</tr>
<tr>
<td>ConcurrencySafety</td>
<td>2370</td>
<td>127</td>
</tr>
<tr>
<td>NoOverflows</td>
<td>6539</td>
<td>49</td>
</tr>
<tr>
<td>Termination</td>
<td>3324</td>
<td>901</td>
</tr>
<tr>
<td>SoftwareSystems</td>
<td>5825</td>
<td>6655</td>
</tr>
</tbody>
</table>

Subcategories in SoftwareSystems

- AWS C commons
### Presentation of SV-Comp (III)

<table>
<thead>
<tr>
<th>Category</th>
<th># tasks</th>
<th>Median loc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReachSafety</td>
<td>6282</td>
<td>1267</td>
</tr>
<tr>
<td>MemSafety</td>
<td>6280</td>
<td>86</td>
</tr>
<tr>
<td>ConcurrencySafety</td>
<td>2370</td>
<td>127</td>
</tr>
<tr>
<td>NoOverflows</td>
<td>6539</td>
<td>49</td>
</tr>
<tr>
<td>Termination</td>
<td>3324</td>
<td>901</td>
</tr>
<tr>
<td>SoftwareSystems</td>
<td>5825</td>
<td>6655</td>
</tr>
</tbody>
</table>

#### Subcategories in SoftwareSystems
- AWS C commons
- BusyBox (coreutils alternative)
### Presentation of SV-Comp (III)

<table>
<thead>
<tr>
<th>Category</th>
<th># tasks</th>
<th>Median loc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReachSafety</td>
<td>6282</td>
<td>1267</td>
</tr>
<tr>
<td>MemSafety</td>
<td>6280</td>
<td>86</td>
</tr>
<tr>
<td>ConcurrencySafety</td>
<td>2370</td>
<td>127</td>
</tr>
<tr>
<td>NoOverflows</td>
<td>6539</td>
<td>49</td>
</tr>
<tr>
<td>Termination</td>
<td>3324</td>
<td>901</td>
</tr>
<tr>
<td>SoftwareSystems</td>
<td>5825</td>
<td>6655</td>
</tr>
</tbody>
</table>

**Subcategories in SoftwareSystems**

- AWS C commons
- BusyBox (coreutils alternative)
- Linux Device Drivers
## Presentation of SV-Comp (III)

<table>
<thead>
<tr>
<th>Category</th>
<th># tasks</th>
<th>Median loc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReachSafety</td>
<td>6282</td>
<td>1267</td>
</tr>
<tr>
<td>MemSafety</td>
<td>6280</td>
<td>86</td>
</tr>
<tr>
<td>ConcurrencySafety</td>
<td>2370</td>
<td>127</td>
</tr>
<tr>
<td>NoOverflows</td>
<td>6539</td>
<td>49</td>
</tr>
<tr>
<td>Termination</td>
<td>3324</td>
<td>901</td>
</tr>
<tr>
<td>SoftwareSystems</td>
<td>5825</td>
<td>6655</td>
</tr>
</tbody>
</table>

### Subcategories in SoftwareSystems
- AWS C commons
- BusyBox (coreutils alternative)
- Linux Device Drivers
- OpenBSD
## Presentation of SV-Comp (III)

<table>
<thead>
<tr>
<th>Category</th>
<th># tasks</th>
<th>Median loc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReachSafety</td>
<td>6282</td>
<td>1267</td>
</tr>
<tr>
<td>MemSafety</td>
<td>6280</td>
<td>86</td>
</tr>
<tr>
<td>ConcurrencySafety</td>
<td>2370</td>
<td>127</td>
</tr>
<tr>
<td>NoOverflows</td>
<td>6539</td>
<td>49</td>
</tr>
<tr>
<td>Termination</td>
<td>3324</td>
<td>901</td>
</tr>
<tr>
<td>SoftwareSystems</td>
<td>5825</td>
<td>6655</td>
</tr>
</tbody>
</table>

Subcategories in SoftwareSystems

- AWS C commons
- BusyBox (coreutils alternative)
- Linux Device Drivers
- OpenBSD
- uthash
SV-Comp’s Scoring System

True (witness confirmed)

0

Unconfirmed (false, unknown, or resources exhausted)

-16

Invalid (error in witness syntax)

-32

0

Invalid (error in witness syntax)

0

Unconfirmed (true, unknown, or resources exhausted)

0

False (witness confirmed)

1

Community-based curation of verdicts

187 manual fixes on my end
SV-Comp’s Scoring System

- Remarks
  - Community-based curation of verdicts
  - 187 manual fixes on my end

Diagram:
- TASK
  - true-unreach
    - true
      - WITNESS_VALIDATOR
        - true (witness confirmed)
        - 2
    - unknown
      - 0
    - false
      - -16

- false-unreach
  - true
    - WITNESS_VALIDATOR
      - invalid (error in witness syntax)
      - 0
    - unknown
      - 0
    - false
      - invalid (error in witness syntax)
      - 0

- verdict
  - true
    - WITNESS_VALIDATOR
      - unconfirmed (true, unknown, or resources exhausted)
      - 0
    - unknown
      - 0
    - false
      - false (witness confirmed)
      - 1
SV-Comp’s Scoring System

Remarks

- community-based curation of verdicts
- 187 manual fixes on my end
Categories are divided into subcategories (a family of benchmarks).
Categories are divided into subcategories (a family of benchmarks).

Scoring incentive for balanced results among subcategories.

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

Motivation

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

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

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)**

---

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\(^5\)

---


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%

---


<table>
<thead>
<tr>
<th>SV-Comp’s “ Witnesses”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Motivation</strong></td>
</tr>
<tr>
<td>▶ Ensure that results can be validated, at a reduced computational cost</td>
</tr>
<tr>
<td>▶ Improve interoperability between verifiers?</td>
</tr>
<tr>
<td><strong>Witnesses</strong></td>
</tr>
<tr>
<td>Automata where edges contain program invariants and control choices</td>
</tr>
<tr>
<td><strong>Issues (in my opinion)</strong></td>
</tr>
<tr>
<td>▶ Interprocedural encoding to be improved(^5)</td>
</tr>
<tr>
<td>▶ Cross-validator scores can be low(^6) – 45%</td>
</tr>
<tr>
<td>▶ 96.4% of Mopsa’s trivial witnesses are validated</td>
</tr>
</tbody>
</table>

Mopsa at SV-Comp
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
Our approach

1. Analyze the target program with Mopsa
2. Postprocess Mopsa’s result to decide whether the property of interest holds
   - Yes? finished!
## 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 all runtime errors
- ⇒ We could at least add slicing
- **New analyses restart from scratch**
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
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 all 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 all runtime errors
   → 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 all runtime errors
  → 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

Conf. Check-Circle CLOCK

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).
## Portfolio of analyses used

### Analyses used

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intervals, small structs initialized</td>
</tr>
<tr>
<td>2</td>
<td>+ string-length domain, medium structs initialized</td>
</tr>
<tr>
<td>3</td>
<td>+ polyhedra with static packing</td>
</tr>
</tbody>
</table>

21,220 tasks in total, 12,636 correctness tasks

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

### Analyses used

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intervals, small structs initialized</td>
</tr>
<tr>
<td>2</td>
<td>+ string-length domain, medium structs initialized</td>
</tr>
<tr>
<td>3</td>
<td>+ polyhedra with static packing</td>
</tr>
<tr>
<td>4</td>
<td>+ congruences &amp; widening tweaks: thresholds, delay</td>
</tr>
</tbody>
</table>

Conf. Check-Circle CLOCK:

- 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).
**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

<table>
<thead>
<tr>
<th>Conf.</th>
<th>Tasks Valid</th>
<th>Tasks Invalid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5695</td>
<td>279</td>
</tr>
<tr>
<td>2</td>
<td>6433 (+738)</td>
<td>365 (+86)</td>
</tr>
<tr>
<td>3</td>
<td>6885 (+452)</td>
<td>1844 (+1479)</td>
</tr>
<tr>
<td>4</td>
<td>6909 (+24)</td>
<td>2009 (+165)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Conf.</th>
<th>✓</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5695</td>
<td>279</td>
</tr>
<tr>
<td>2</td>
<td>6433 (+738)</td>
<td>365 (+86)</td>
</tr>
<tr>
<td>3</td>
<td>6885 (+452)</td>
<td>1844 (+1479)</td>
</tr>
<tr>
<td>4</td>
<td>6909 (+24)</td>
<td>2009 (+165)</td>
</tr>
</tbody>
</table>

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/

Reachability
Mopsa scores a bit below Goblint.

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!
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/

<table>
<thead>
<tr>
<th>Reachability</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mopsa scores a bit below Goblint.(^7)</td>
<td></td>
</tr>
<tr>
<td>Might be a bad configuration choice?</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mopsa is the only abstract interpreter participating in this category.</td>
<td></td>
</tr>
</tbody>
</table>

\(^7\) other active abstract interpreter
## Mopsa’s Results

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reachability</td>
<td>Mopsa scores a bit below Goblint.⁷</td>
</tr>
<tr>
<td></td>
<td>Might be a bad configuration choice?</td>
</tr>
<tr>
<td>Memory</td>
<td>Mopsa is the only abstract interpreter participating in this category.</td>
</tr>
<tr>
<td>Overflow</td>
<td>Ranks 6th/19, before Frama-C and Goblint.</td>
</tr>
<tr>
<td></td>
<td>Mopsa is on par with the winner for the number of programs proved correct!</td>
</tr>
</tbody>
</table>

⁷other active abstract interpreter
Bronze medal in the *SoftwareSystems* category!
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.

<table>
<thead>
<tr>
<th>Verifier</th>
<th>Bubaak</th>
<th>CPAchecker</th>
<th>Goblint</th>
<th>Mopsa</th>
<th>Symbiotic</th>
<th>Ultimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proved correct</td>
<td>291</td>
<td>1,651</td>
<td>1,256</td>
<td>1,610</td>
<td>942</td>
<td>1,423</td>
</tr>
<tr>
<td>Proved incorrect</td>
<td>143</td>
<td>59</td>
<td>0</td>
<td>0</td>
<td>84</td>
<td>2</td>
</tr>
<tr>
<td>CPU Time (s)</td>
<td>2,000,000</td>
<td>730,000</td>
<td>800,000</td>
<td>580,000</td>
<td>400,000</td>
<td>1,400,000</td>
</tr>
<tr>
<td>Rank</td>
<td>2</td>
<td>6</td>
<td>10</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>
Mopsa’s Results

Bronze medal in the *SoftwareSystems* category!

19 participants. First French participation.

<table>
<thead>
<tr>
<th>Verifier</th>
<th>Bubaak</th>
<th>CPAchecker</th>
<th>Goblint</th>
<th>Mopsa</th>
<th>Symbiotic</th>
<th>Ultimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proved correct</td>
<td>291</td>
<td><strong>1,651</strong></td>
<td>1,256</td>
<td>1,610</td>
<td>942</td>
<td>1,423</td>
</tr>
<tr>
<td>Proved incorrect</td>
<td>143</td>
<td>59</td>
<td>0</td>
<td>0</td>
<td>84</td>
<td>2</td>
</tr>
<tr>
<td>CPU Time (s)</td>
<td>2,000,000</td>
<td>730,000</td>
<td>800,000</td>
<td>580,000</td>
<td><strong>400,000</strong></td>
<td>1,400,000</td>
</tr>
<tr>
<td>Rank</td>
<td>2</td>
<td>6</td>
<td>10</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

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