Mopsa at the Software Verification Competition

Raphaël Monat

SyCoMoRES team

rmonat.fr
Introduction
Research area: formal methods
Goal: improve confidence in software
Worked on two real-world systems
▶ Analysis of Python programs, and interoperability with C (LIP6)
▶ French income tax code (Inria Paris & MSR)
Research area: formal methods
Goal: improve confidence in software
Research area: formal methods
Goal: improve confidence in software

Worked on two real-world systems
- Analysis of Python programs, and interoperability with C (LIP6)
- French income tax code (Inria Paris & MSR)
SyCoMoRES team

Youngest team, hosted in ESPRIT.
SyCoMoRES team

Youngest team, hosted in ESPRIT.

Component-based design of real-time embedded systems
SyCoMoRES team

Youngest team, hosted in ESPRIT.

**Component-based design of real-time embedded systems**

- Programming language design
- Static analysis
- Real-time scheduling
- Computer-assisted formal proofs
SyCoMoRES team

Youngest team, hosted in ESPRIT.

Component-based design of real-time embedded systems

- Programming language design
- Static analysis
- Real-time scheduling
- Computer-assisted formal proofs

Members

- Patrick Baillot
- Clément Ballabriga
- Julien Forget
- Giuseppe Lipari
- Vlad Rusu
- Nordine Feddal
- Andrei Florea
- Sandro Grebant
- Leandro Gomes
- Ikram Senoussaoui
Software verification

Cheap approach: test $S[\text{prog}]$. Some bugs may go undetected! Would there be a way to automatically prove programs correct?
Software verification

$S[ prog ]$  
Bad states

Cheap approach: test $prog$. Some bugs may go undetected!

Would there be a way to automatically prove programs correct?
Software verification

Cheap approach: test \textit{prog}.

Some bugs may go undetected!

Would there be a way to automatically prove programs correct?
Software verification

Cheap approach: test \textit{prog}.

$S[ prog ]$
Software verification

Cheap approach: test \textit{prog}.
Software verification

Cheap approach: test prog.

Some bugs may go undetected!

Would there be a way to automatically prove programs correct?
Software verification

Cheap approach: test $prog$.

Some bugs may go undetected!
Software verification

Cheap approach: test \textit{prog}.
Some bugs may go undetected!
Software verification

Cheap approach: test $prog$. Some bugs may go undetected!

Would there be a way to automatically prove programs correct?
An impossibility theorem

All reported errors are true errors.

Sound

All true errors are reported.
An impossibility theorem

All reported errors are true errors

Complete

Sound

All true errors are reported
An impossibility theorem

Guaranteed Termination

Complete

Sound

All reported errors are true errors

All true errors are reported
An impossibility theorem

All reported errors are true errors

Guaranteed Termination

Complete

Sound

All true errors are reported

Rice’s theorem
An impossibility theorem

- All reported errors are true errors
- Complete
- Sound
- Guaranteed Termination
- Abstract Interpretation
- Rice’s theorem

- All true errors are reported
Abstract interpretation – the big picture

P. Cousot and R. Cousot. “Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints”. POPL 1977
Abstract interpretation – the big picture

P. Cousot and R. Cousot. “Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints”. POPL 1977
Abstract interpretation – the big picture

\[ D \] (concrete)

\[ S[\text{prog}] \]

Bad states

\[ \gamma \]

\[ S^\#[\text{prog}] \]

\[ D^\# \] (abstract)

Bad states

True alarm

P. Cousot and R. Cousot. “Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints”. POPL 1977
Abstract interpretation – the big picture

\[ S[\text{prog}] \]

\( \mathcal{D} \) (concrete)

Bad states

\( \gamma \)

\[ S^\#[\text{prog}] \]

\( \mathcal{D}^\# \) (abstract)

Bad states

P. Cousot and R. Cousot. “Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints”. POPL 1977
Abstract interpretation – the big picture

\[ \mathcal{D} (\text{concrete}) \]

\[ \mathcal{S}[\text{prog}] \]

\[ \mathcal{S}^\#[\text{prog}] \]

\[ \mathcal{D}^\# (\text{abstract}) \]

Bad states

Bad states

False alarm (Abstraction too coarse)

P. Cousot and R. Cousot. “Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints”. POPL 1977
Abstract interpretation – the big picture

\[ D (\text{concrete}) \rightarrow \gamma \rightarrow D^\# (\text{abstract}) \]

Unsound analysis
(shouldn’t happen)

P. Cousot and R. Cousot. “Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints”. POPL 1977
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 m

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.
Critical software certification through static analysis

Bertrane, P. Cousot, R. Cousot, Feret, Mauborgne, Miné, and Rival. “Static analysis and verification of aerospace software by abstract interpretation”. AIAA Infotech@Aerospace (I@A 2010) 2010
Critical software certification through static analysis

Bertrane, P. Cousot, R. Cousot, Feret, Mauborgne, Miné, and Rival. “Static analysis and verification of aerospace software by abstract interpretation”. AIAA Infotech@Aerospace (I@A 2010) 2010
Critical software certification through static analysis

Democratizing static analysis?
- Multiple languages?
- Precision and configurability?

Bertrane, P. Cousot, R. Cousot, Feret, Mauborgne, Miné, and Rival. “Static analysis and verification of aerospace software by abstract interpretation”. AIAA Infotech@Aerospace (I@A 2010) 2010
Mopsy
Overview of Mopsa

Modular Open Platform for Static Analysis\(^1\)
gitlab.com/mopsa/mopsa-analyzer

2016-2021: ERC Consolidator Grant, awarded to Antoine Miné.

\(^1\)Journault, Miné, Monat, and Ouadjaout. “Combinations of reusable abstract domains for a multilingual static
Overview of Mopsa

Modular Open Platform for Static Analysis

gitlab.com/mopsa/mopsa-analyzer

2016-2021: ERC Consolidator Grant, awarded to Antoine Miné.

Goals

1Journault, Miné, Monat, and Ouadjaout. “Combinations of reusable abstract domains for a multilingual static
Overview of Mopsa

**Modular Open Platform for Static Analysis**

gitlab.com/mopsa/mopsa-analyzer

2016-2021: ERC Consolidator Grant, awarded to Antoine Miné.

**Goals**

▶ explore new designs

---

\(^1\)Journault, Miné, Monat, and Ouadjaout. “Combinations of reusable abstract domains for a multilingual static...
Overview of Mopsa

**Modular Open Platform for Static Analysis**

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

2016-2021: ERC Consolidator Grant, awarded to Antoine Miné.

**Goals**

- explore new designs
- ease development/prototyping

---

Overview of Mopsa

Modular Open Platform for Static Analysis\(^1\)
gitlab.com/mopsa/mopsa-analyzer

2016-2021: ERC Consolidator Grant, awarded to Antoine Miné.

Goals

- explore new designs
- ease development/prototyping
- support multiple languages

\(^1\)Journault, Miné, Monat, and Ouadjaout. “Combinations of reusable abstract domains for a multilingual static
Overview of Mopsa

**Modular Open Platform for Static Analysis**

gitlab.com/mopsa/mopsa-analyzer

2016-2021: ERC Consolidator Grant, awarded to Antoine Miné.

**Goals**

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

---

1Journault, Miné, Monat, and Ouadjaout. “Combinations of reusable abstract domains for a multilingual static
Overview of Mopsa

Modular Open Platform for Static Analysis

gitlab.com/mopsa/mopsa-analyzer

2016-2021: ERC Consolidator Grant, awarded to Antoine Miné.

Goals

▶ explore new designs
▶ ease development/prototyping
▶ support multiple languages
▶ loosely couple abstractions

Contributors

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

¹Journault, Miné, Monat, and Ouadjaout. “Combinations of reusable abstract domains for a multilingual static
Current public analyses in Mopsa

<table>
<thead>
<tr>
<th>Semantic property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime error detection</td>
</tr>
</tbody>
</table>

\[ \approx 50,000 \text{ lines of OCaml code} \]

# Language Benchmark Max. LoC

- Coreutils 550 20s 99.8%
- Juliet 340,000 2.5h 98.9%
- PyPerformance 1,792 1.3m 99.2%
- PathPicker 2,560 3.0m 99.2%
- ahocorasick 4,800 1.0m 98.0%
- bitarray 5,700 4.6m 94.6%

# safe operations

- 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

≃ 50,000 lines of OCaml code

<table>
<thead>
<tr>
<th>Language</th>
<th>Benchmark</th>
<th>Max. LoC</th>
<th>≃ Time</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C^2 )</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>
</tbody>
</table>

Current public analyses in Mopsa

Semantic property
Runtime error detection

≃ 50,000 lines of OCaml code

<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>
</tbody>
</table>

Current public analyses in Mopsa

Semantic property
Runtime error detection

$\approx 50,000$ lines of OCaml code

<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>C$^2$</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>Python$^3$</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

≃ 50,000 lines of OCaml code

<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
SV-Comp
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
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
## 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)

<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>
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
- 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
- Community-curated
- Programs can be added over the years
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
- Community-curated
- Programs can be added over the years
## 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
- Community-curated
- Programs can be added over the years

### Properties

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

<table>
<thead>
<tr>
<th>Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Preprocessed C programs</strong></td>
</tr>
<tr>
<td><strong>Lots of handcrafted or small examples</strong></td>
</tr>
<tr>
<td><strong>Community-curated</strong></td>
</tr>
<tr>
<td><strong>Programs can be added over the years</strong></td>
</tr>
</tbody>
</table>
**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
- Community-curated
- Programs can be added over the years

**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

Properties

- Reachability
- Memory safety
- Integer overflows
- Termination

Programs

- Preprocessed C programs
- Lots of handcrafted or small examples
- Community-curated
- Programs can be added over the years
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**
- Reachability
- Memory safety
- Integer overflows
- Termination
- Data race

**Programs**
- Preprocessed C programs
- Lots of handcrafted or small examples
- Community-curated
- Programs can be added over the years
## 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 Commons
- BusyBox (coreutils alternative)
- Linux Device Drivers
- OpenBSD
- uthash
### 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

- **Verdicts**:
  - **True (Witness Confirmed)**: 2
  - **Unconfirmed (False, Unknown, or Resources Exhaustrated)**: 0
  - **Invalid (Error in Witness Syntax)**: 0

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

- **Diagram**:
  - **Task**: True-unreach
  - **Verifier**:
    - True: -32
    - Unknown: 0
    - False: -16
  - **Witness Validator**:
    - True: 2
    - Unconfirmed: 0
    - False: 1
SV-Comp’s Scoring System

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

Diagram:
- **Task**
  - true-unreach
  - false-unreach
- **Verifier**
  - true
  - unknown
  - false
- **Witness Validator**
  - true (witness confirmed)
  - unconfirmed (false, unknown, or resources exhausted)
- **Result**
  - 2
  - 0
  - -16
  - -32
  - 0
  - 0
  - 0
  - 0
  - 0
  - 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
SV-Comp’s “Witnesses”

Motivation

► Ensure that results can be validated, at a reduced computational cost
► Improve interoperability between verifiers?
SV-Comp’s “Witnesses”

<table>
<thead>
<tr>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Ensure that results can be validated, at a reduced computational cost</td>
</tr>
<tr>
<td>▶ Improve interoperability between verifiers?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Witnesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automata where edges contain program invariants and control choices</td>
</tr>
</tbody>
</table>
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)**

---

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\(^5\)

---


\(^6\) 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%

\(^6\) Beyer, Dangl, Dietsch, Heizmann, Lemberger, and Tautschnig. “Verification Witnesses”. 2022
<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>

\(^5\)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

   - 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
### 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
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 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
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**

<table>
<thead>
<tr>
<th>Analyses used</th>
<th>Conf. Check-Circle CLOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Intervals, small structs initialized</td>
<td>21220 tasks in total, 12636 correctness tasks</td>
</tr>
<tr>
<td>2. + string-length domain, medium structs initialized</td>
<td></td>
</tr>
<tr>
<td>3. + polyhedra with static packing</td>
<td></td>
</tr>
<tr>
<td>4. + congruences &amp; widening tweaks: thresholds, delay</td>
<td></td>
</tr>
</tbody>
</table>

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>✔️</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
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/
Mopsa’s Results

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

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

⁷ other active abstract interpreter
Mopsa’s Results

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

Reachability
Mopsa scores a bit below Goblint. 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!
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>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 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
Mopsa as a stable academic static analyzer,
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.
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 Questions

Raphaël Monat
SyCoMoRES team
rmonat.fr

30 minutes of Science
10 March 2023