Comparing Transparent Static Analyzers with Open Verification Dashboard

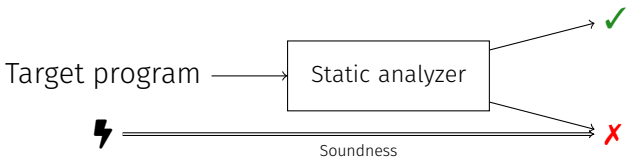
Tom Goalard, Karoliine Holter, Simmo Saan,
Vesal Vojdani, Raphaël Monat

`rmonat.fr/ecoop26_tsa`

ECOOP
3 July 2026

Inria

Static Analysis



Alarms

- Muske and Serebrenik [MS23]: 130 works on alarm postprocessing
- Alarms are only one side of the coin!

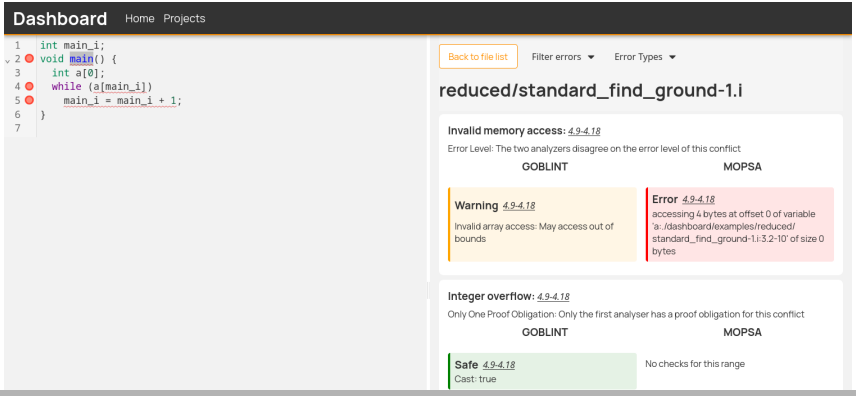
Transparency

- Log proof obligations (PO). Analogy with deductive verification.
- Enables semantics-directed, fine-grained output.

Comparing Static Analyzers

- Real-world software: no false/true alarm baseline
- Incomparable approximations between analyzers
- Transparency helps, yet harmonization efforts required

  `strcpy` misuse: buffer overflow or contract violation for the stub?



Outline

1. Transparent Static Analysis
2. Open Verification Dashboard
3. Experimental Evaluation

Transparent Static Analysis

Safety check categories

Example on toy imperative language with mathematical integers

$\mathcal{C} = \{\text{divByZero}\}$	*Safety check categories*
$\mathcal{L}$	*Program locations*
$\varphi \in \text{PO} = \mathcal{L} \times \mathcal{C}$	*Proof obligations*
$\Theta \subseteq \text{CH} = \text{PO} \times \{\checkmark, \times\}$	*Checks*

Concrete Collecting Semantics

$\mathbb{E}[\![e \in \text{Expr}]\!] : (\text{Var} \to \mathbb{Z}) \to 2^{\mathbb{Z}} \times 2^{\text{CH}}$ *Concrete semantics of e*

$$\mathbb{E}[\![e_1 /^{\ell} e_2]\!]\sigma =$$
$$\textbf{let } M_1, \Theta_1 = \mathbb{E}[\![e_1]\!]\sigma \textbf{ and } M_2, \Theta_2 = \mathbb{E}[\![e_2]\!]\sigma \textbf{ in}$$
$$\textbf{let } \Theta' = \{(\ell, \text{divByZero}), \textsf{check}(m_2 \neq 0) \mid m_2 \in M_2\} \textbf{ in}$$
$$\{m_1 \,/\, m_2 \mid m_i \in M_i, m_2 \neq 0, m_1 \,/\, m_2 \in \mathbb{Z}\}, \Theta_1 \cup \Theta_2 \cup \Theta'$$
$$\textsf{check}(b) = \textbf{if } b \textbf{ then } \checkmark \textbf{ else } \times$$

$\mathbb{E}[\![1 /^{\ell} \texttt{rand}(-1,1)]\!]\emptyset = \quad \{-1,1\}, \{(l, \text{divByZero}), \checkmark\} \cup \{(l, \text{divByZero}), \times\}$

Abstract Semantics

$\mathbb{E}^\sharp[\![e \in \text{Expr}]\!] : (\text{Var} \to \mathbb{Z}^\sharp) \to \mathbb{Z}^\sharp \times 2^{\text{CH}}$ *Abstract semantics of e*

$\mathbb{E}^\sharp[\![e_1 \,/^\ell\, e_2]\!]\sigma^\sharp =$

let $z_1^\sharp, \Theta_1 = \mathbb{E}^\sharp[\![e_1]\!]\sigma^\sharp$ **and** $z_2^\sharp, \Theta_2 = \mathbb{E}^\sharp[\![e_2]\!]\sigma^\sharp$ **in**

let $\Theta' = \{(\ell, \text{divByZero}), ✗ \mid \mathbb{E}^\sharp[\![e_2 == 0]\!]\sigma^\sharp \neq (\bot^\sharp, \_)\}$ **in**

let $\Theta'' = \{(\ell, \text{divByZero}), ✓ \mid \mathbb{E}^\sharp[\![e_2 \,!= 0]\!]\sigma^\sharp \neq (\bot^\sharp, \_)\}$ **in**

$z_1^\sharp \,/^\sharp\, z_2^\sharp, \Theta_1 \cup \Theta_2 \cup \Theta' \cup \Theta''$

$\mathbb{E}^\sharp[\![1 \,/^\ell\, \texttt{rand}(-1,1)]\!]\_ = \quad [-1,1], \{(l, \text{divByZero}), ✓\} \cup \{(l, \text{divByZero}), ✗\}$

Soundness

$$\forall e \in \text{Expr}, \forall \sigma \in \gamma(\sigma^\sharp), \begin{cases} \mathbb{E}^\sharp[\![e]\!]\sigma^\sharp = (z^\sharp, \Theta^\sharp) \\ \wedge\, \mathbb{E}[\![e]\!]\sigma = (M, \Theta) \end{cases} \implies M \subseteq \gamma_{\mathbb{Z}}(z^\sharp) \wedge \Theta \subseteq \Theta^\sharp$$

$\Theta \subseteq \Theta^\sharp$ concisely encodes:

- If a check fails at runtime, the analysis should say so *true alarm*
- If a check is reachable at runtime, the analysis should check it *safe check, false alarm*

Report postprocessing

From analysis checks (same location, multiple contexts) to analysis results:

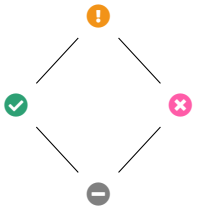
$$\textsf{checkStatus} : 2^{\{✓,✗\}} \to \text{Status}$$

$$\emptyset \mapsto ⊖$$

$$\{✓\} \mapsto ✓$$

$$\{✗\} \mapsto ✗$$

$$\{✓, ✗\} \mapsto !$$



$$\{(l, \text{divByZero}), ✓\} \cup \{(l, \text{divByZero}), ✗\} \rightsquigarrow (l, \text{divByZero}), !$$

Open Verification Dashboard

Cross-Analyzer Comparison

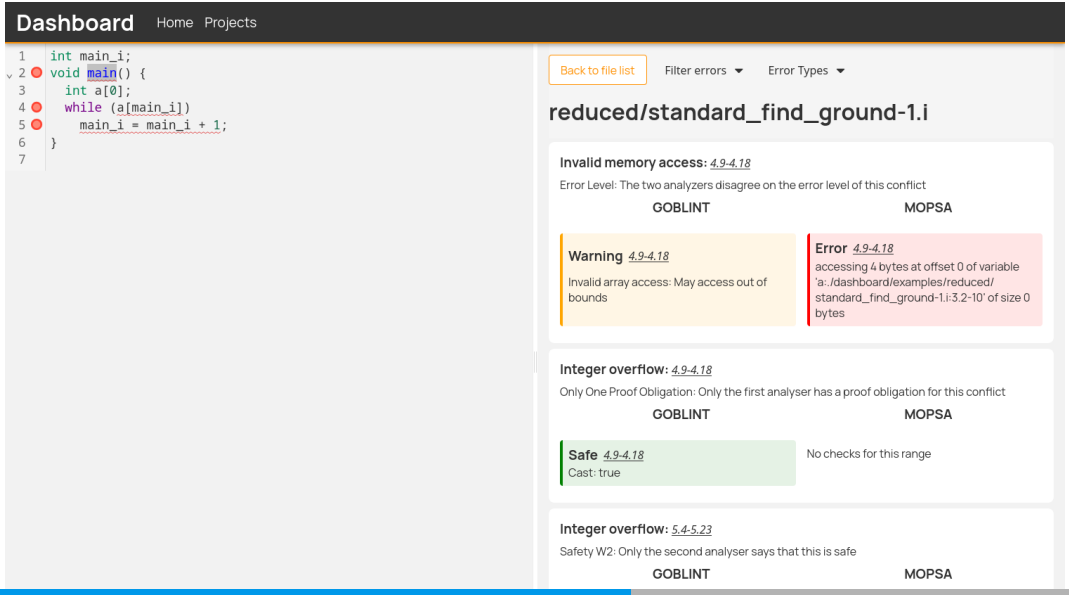
Status$_1$ \ Status$_2$	!	✗	✓	−
!	A^-	P	P	D
✗	P	A^-	C	D
✓	P	C	A^+	D
−	D	D	D	A^+

- Positive or negative **agreement** (A^+ / A^-)
- Coverage **disagreement** (D)
- Precision **asymmetry** (P)
- **Contradiction** (C)

Open Verification Dashboard

- JSON-based interchange format (simplified integration)
- CLI for automated workflows
- GUI interface for interactive exploration
  - Global statistics view (cf. paper)
  - Detailed view

Open Verification Dashboard GUI – Detailed view



Integrating a New Analyzer

Proof Obligation Outcomes

- Report successful checks in addition to alarms
- JSON interchange compatibility
- Check alignment

Standardized Integer Overflow Categories

Following a systematic comparison, 6 categories identified:

(signed|unsigned) integer overflow in (arithmetic operator|explicit cast|implicit cast)

Program Range Reporting

Multiple choices (line only, line+column, line+column**s**) in IKOS, Goblint, Mopsa

$\rightsquigarrow$ Heuristic location alignment (details in paper)

Experimental Evaluation

Protocol

Benchmarks

- SV-COMP no-overflow (8,358 tasks)
- coreutils `cut, env, uniq` $\simeq 200k$LOC

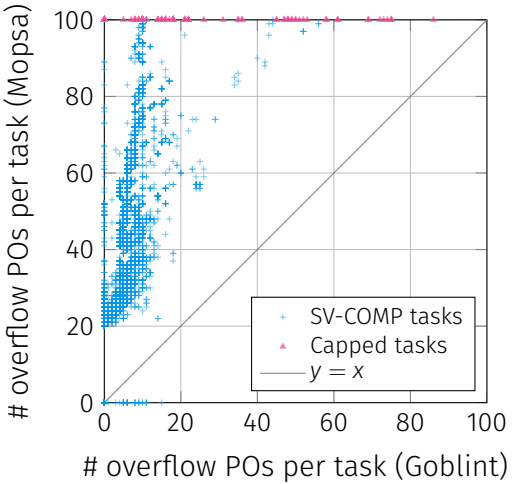
Transparent Tools

- Goblint
- Mopsa

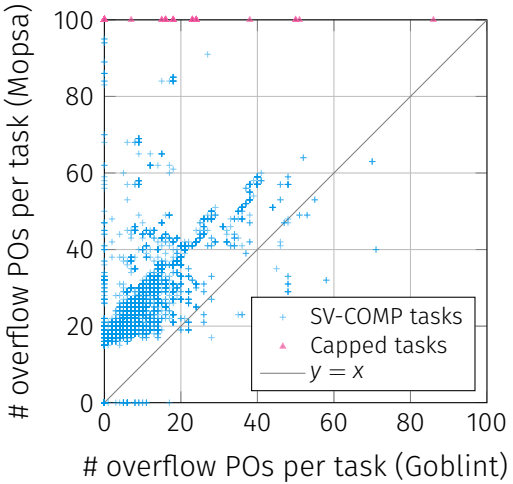
Research Questions

1. Complementarity of Goblint and Mopsa?
2. Joint Benefit of Virtual Combination?
3. Distillation of Disagreements through Testcase Reduction?

SV-COMP Benchmark Validity?



Before alignment



After alignment

N.B: Mopsa performs callstack-sensitive PO reporting.

RQ1: Complementarity

Mopsa	Goblint			
	8122	0	2268	159
	1	0	0	6
	222	0	11 700	23 754
	949	0	1804	0

Table 1: **before** PO alignment fixes.

Mopsa	Goblint			
	8123	0	2255	**146**
	1	**0**	**0**	**0**
	222	**0**	**32226**	**652**
	944	**0**	**2126**	**0**

Table 2: **after** PO alignment fixes.

- No contradictions
- Overall agreement
- Improved coverage consistency

Issue	Mopsa	Goblint
Missing check		#1933, cil#215, cil#216
Wrong location	#248	cil#211
Spurious check	**#251**	**#1909, #1910, #1932**

RQ2: Joint Benefit (Coreutils `uniq`)

	Goblint			
Mopsa	!	✗	✓	−
!	16	0	3	0
✗	0	0	0	0
✓	8	0	46	935
−	2	0	14	0

Joint Benefit

- Virtual combination improves precision
- But currently insufficient to prove whole program correct

RQ3: Distillation

Automated testcase reduction oracles

- Verdict-based: boolean output of SV-COMP verification
- Dashboard-based: specific safety check category

	Task Metrics			Verdict-Based			Dashboard-Based	
Folder	#	Lines	Conflicts	Final lines	Pres. (%)	Time (s)	Final lines	Time (s)
`float-benchs`	6	196.17	1.00	8.67	100.00	184.50	8.67	233.17
`nla-digbench`	4	42.25	5.75	5.00	0.00	104.25	6.00	123.00
`termination-crafted`	11	26.18	1.55	3.73	18.18	119.73	5.91	156.45
`termination-crafted-lit`	11	27.18	1.73	5.27	45.45	92.00	7.00	102.73
`termination-numeric`	3	30.00	1.00	4.00	0.00	124.00	5.67	190.00
`termination-restricted-15`	13	21.62	1.00	4.31	7.69	83.62	8.77	98.62

Dashboard Strategy

- Less aggressive in reducing size,
- 100% preservation of disagreements.

Conclusion

Related Work

- Sound(i)ness [Liv+15]
- Textbook static analyses focus on computing reachable states [MS24; Min17]
- Exchanging results: SARIF [OAS20], SV-COMP's witnesses [Bey+16; Aya+24]

Conclusion

- Transparent analyses going beyond alarm-based reporting
- Enables fine-grained comparison, leveraged by Open Verification Dashboard
- 14 bugs improving precision & reporting quality of Goblint+Mopsa
- More details in the paper, available+reusable artefact

⇝ standard for reporting C runtime errors detected by static analyzers.

⇝ spreading transparent reporting? Helps with UX, comparisons & improvements.

`rmonat.fr/ecoop26_tsa`