講演番号	030

Sound Static Analysis for Safety and Security

Daniel Kästner [1)] **Reinhard Wilhelm** [1)] **Christian Ferdinand** [1)] **Adrian Dapprich** [2)]

1) AbsInt GmbH
Science Park 1, 66123 Saarbrücken, Germany (E-mail: info@absint.com)
2) AbsInt GmbH
16-4-16 Numabukuro, Nakano, Tokyo, 165-0025, Japan (E-mail: dapprich@absint.com)

KEY WORDS: **software and its underlying technologies, ISO-26262, cybersecurity,** static analysis, SAST tools (E3)

We discusse the role of sound static analysis in ensuring the safety and security of modern embedded software, particularly in the automotive domain. As software systems grow in size and complexity the risk of runtime errors increases significantly. These errors, including buffer overflows, division by zero, and data races can lead to system failures or cybersecurity breaches, making their prevention critical.

Static analysis is a technique that examines software without executing it. While basic (syntactic) analysis checks coding rules, more advanced semantic static analysis analyzes program behavior, considering variable values and control flow. Static analysis tools are distinguished between unsound and sound: unsound methods may miss errors, whereas sound analysis guarantees that all potential runtime errors are detected (no false negatives), though it may produce false positives.

A key method enabling sound analysis is abstract interpretation, which approximates all possible program executions using simplified models (abstract semantics). This allows scalable analysis while ensuring coverage of all execution paths. An example of a sound analysis tool is the Astrée static analyzer. It uses multiple abstract domains (e.g., for numeric values, relationships between values, pointers, etc.) and supports concurrent program analysis, enabling detection of data races and deadlocks. To improve precision and reduce false alarms, the analyzer allows fine-tuning via techniques such as loop unrolling, value partitioning, and control partitioning. These methods help distinguish different execution paths and variable states more accurately. Importantly, increasing precision does not compromise soundness and can even improve performance by eliminating infeasible paths.

Experimental results on large-scale industrial AUTOSAR projects (up to 40 million lines of preprocessed code) as shown in Tab. 1, demonstrate that sound static analysis is feasible in practice. Analysis times ranged from hours to about a day, with manageable memory usage. Precision tuning reduced false alarms by up to 36% (cf. Tab. 2) and improved detection of real issues without additional verification burden.

In conclusion, sound static analysis, particularly using abstract interpretation, is a powerful and practical approach for ensuring the absence of runtime errors and verifying safety and security requirements in complex embedded systems. It supports compliance with industry standards and provides comprehensive insights into data and control flow.

Tab. 1 Analysis Results on AUTOSAR Projects

Project	PPLOC	Time	Memory
P1	5 M	12h	19 GB
P2	6 M	12h	22 GB
P3	4 M	18h	20 GB
P4	4 M	15h	13 GB
P5	3 M	10h	37 GB
P6	3 M	13h	12 GB
P7	5 M	11h	29 GB
P8	12 M	1d 12h	61 GB
P9	9 M	17h	47 GB
P10	40 M	1d 3h	32 GB

Tab. 2 Impact of precision-tuning annotations

Project	Annotations	ΔRTE	ΔRaces	ΔT	ΔM
P1	432	-21%	-3%	-6%	-1%
P2	139	-3%	0%	-11%	-1%
P3	741	-24%	-1%	-30%	-16%
P4	428	-23%	1%	+150%	+43%
P5	919	-36%	-21%	-60%	-27%
P6	53	-12%	-21%	-51%	-75%
P7	421	-12%	-3%	-50%	-18%
P8	498	-29%	-21%	+27%	+40%
P9	1166	-21%	-20%	+22%	+77%
P10	528	-26%	-1%	-53%	-17%