**Integrating Abstract Interpretation in Agentic Workflows to Automatically ensure Memory Safety and prevent Cybersecurity Vulnerabilities**

*Dr. Daniel Kästner, CTO, AbsInt GmbH*

Abstract Interpretation is a formal method for static program analysis ensuring soundness in a mathematically rigorous way. A sound static runtime error analysis at the C/C++ level finds all runtime errors due to undefined/unspecified behaviors of the programming language, and can prove their absence.

Examples of such runtime errors are divisions by zero, buffer overflows, data races and deadlocks. They cannot only compromise the safety of the system, but also constitute critical cybersecurity vulnerabilities. Runtime error analysis is also the basis of more advanced program analyses, e.g., for data and control coupling and non-interference analysis, which also have applications in the safety and cybersecurity domain.

Soundness of the analysis means that there are no false negatives (missed defects), but there can be false positives. The findings of the analyzer have to be investigated to determine whether a given finding is a false alarm or a real defect in the program. Performing this review requires expert knowledge about the underlying software and the analysis methodology.

Due to the quickly evolving capabilities of LLM engines, usage of AI copilots to automatically generate code has gained significant momentum outside the classical safety-/security-critical fields. In the safety domain, the stochastic behavior and inherent non-determinism of LLMs need to be taken into account, so the qualification of LLM-generated code remains an open problem.

In the presentation we will give an overview of ongoing research to integrate the Astrée analyzer in an orchestrated agentic AI framework. There are two main goals: the first one is to ensure absence of runtime errors in AI-assisted code generation processes. The second one is to apply AI agents to review findings and fix defects. We have developed skill files and support scripts that enable AI agents to create Astrée configurations, import analysis results, and that provide a knowledge base for AI agents to assist in reviewing and explaining findings. The vision is to create a feedback loop that achieves predefined quality goals (e.g., zero runtime errors) within given analysis time or memory limits, and within given LLM usage costs (e.g., token usage, model size). One of the open challenges is to enable AI agents to fine-tune the precision of the analyzer to avoid false alarms. The fine-tuning is based on formal directives that parameterize the abstract domains of Astrée for a given scope of the program. Another challenge is to optimize the analysis configuration for execution time or memory bounds. It is also necessary to evaluate the quality of LLM-suggested code fixes and LLM-generated alarm explanations in realistic environment and develop strategies to achieve improvements.

The benefits of agentic use of abstract interpretation-based analysis are significant: a successful automation makes it possible to determine all runtime errors in LLM-generated code, or to demonstrate their absence. The evidence about absence of runtime errors can be leveraged wrt qualification criteria of LLM-generated code. The process of reviewing and classifying findings can be partially automated, which is a huge benefit for applying sound static analysis in CI/CD and devOps environments and for scaling to large user bases, even in absence of AI-generated code.