How Bosch brings Formal Methods to Autonomous Systems Engineering

Presenter: Michaela Klauck and Christian Heinzemann,
Robert Bosch GmbH – Bosch Research
Robert-Bosch-Campus 1
71272 Renningen, Germany
michaela.klauck@de.bosch.com, christian.heinzemann@de.bosch.com
Co-Author: Ralph Lange, ralph.lange@de.bosch.com

At Bosch Research, we strive at making formal methods efficiently applicable in industry, especially in autonomous driving and robotics. We report on our experiences and findings gained in four industrial research projects in which we developed and analyzed formal methods tooling for the application in industrial case studies. These works provided insights how formal methods can be made attractive for industry. At the same time, they raised challenging research questions to overcome recurring problems which often still prevent practical applicability of formal methods in industry.

We applied search-based testing (SBT) in automated driving control applications in several use cases [1]. We used formal specifications based on Signal Temporal Logic (STL) with robust semantics for the system under test and applied global optimization to search for executions that violate the specification. From our case studies, we derived lessons learned for the adoption of SBT methods and tools in industry. The core findings are: (1) SBT is helpful to find relevant errors and provides valuable feedback to developers, but user acceptance depends on seamless integration into existing toolchains with a feedback loop to apply the results to the system. (2) Good support for writing and debugging the formal specifications is essential.

In addition, we developed a formally verifiable DSL, called vTSL, for the specification of robot tasks by end-users [2], because user-defined tasks often have to meet safety and integrity constraints to protect the robotic platform and its users. vTSL allows to automatically prove that these constraints are met by transforming task trees specified in vTSL into Promela models for verification with the Spin model checker. The largest potential for unwanted behavior of robotic systems, however, is not in individual parameters or inputs, but in the concurrent execution of different underlying skills, subsystems of the robot platform and the interaction with the environment. For this purpose, vTSL is attractive to use because it interfaces with the popular Robot Operating System. A key lesson that we learned during the application of vTSL is that formal methods should integrate as seamlessly as possible into the normal development workflow of engineers. If the formal model is a separate artifact that needs to be created and maintained by hand, there is little to no acceptance in industrial practice.

For this reason, we took a different approach in our most recent publication [3]. We automatically generate a formal model from the system code itself. This enables us to apply model checking during the development of a behavior planner for autonomous vehicles and to integrate model checking techniques into the development process in the Automated Driving Alliance between Bosch and Cariad. The key idea is to use an automated extraction mechanism that, starting from the (mildly annotated) C++ code of the planner, generates a higher-level model of the underlying logic. This model, composed in closed loop with environment descriptions, can then be exhaustively analyzed with model checking instead of very large amounts of test drives. The approach was exemplarily deployed in series development, and successfully found relevant issues in intermediate versions of the planner at development time. Using automated extraction of the formal model from production code and the seamless integration into the development environment of the engineers were key success factors.

Formal methods are also of great help to achieve robust deliberation capabilities of autonomous robots [4]. This is one of the core topics of CONVINCE, funded by the EU Horizon Europe Program. In this project, we develop formal tools that ensure correct execution of behaviors and contingency plans instan-

tiated by a cognitive deliberation system both at design time and run time. Those tools will be integrated into a model-driven software toolchain to automate the analysis of behaviors to ensure that they are safe and secure. To achieve our goals, modeling languages to formally describe the entire robotic system, and tools to formally verify robot behavior are under development. Common modeling languages established in the community are of interest because then model checking tooling is available right away. To minimize overhead for developers and modeling discrepancies, we aim at directly using parts of the robot's software, e.g., Behavior Trees in the model. Algorithmically, we explore the use of statistical model checking (SMC) to handle real-world scenarios. An important goal is to bring together existing model checking, robotic tools, and modeling languages by an open-source framework which also commercial providers can link their solutions to. For this, our framework will provide lightweight adapters and small tools on top of existing software.

To conclude, we used formal methods in different projects and at different stages in design, specification, development, validation, and verification of software systems. Our results show that formal methods can bring a huge benefit to industry but also highlighted remaining challenges. The lessons learned from these projects can be summarized as follows:

1) Applying formal specifications requires appropriate support for engineers in terms of modeling languages expressive enough for industrial settings based on logics for which efficient tooling is available that supports design and debugging support.
2) This includes support for model extraction, generation, or writing to bridge the gap between a running industrial system and a model of it, to make use of formal methods. This goes hand in hand with model validation to make sure that the model is coherent with the system.
3) Tooling for formal methods needs to be tightly integrated into the day-to-day development environment of engineers, such that it is easily accessible for them, and necessary changes can be quickly recognized and implemented as development progresses. Completely decoupled tooling is a problem, because then inputs and outputs need to be converted, the usage is unclear, and in general the inhibition barrier to use the tooling is too high.

In the currently running EU-funded project CONVINCE, we are open for collaborations (Master theses, PhD sabbaticals, visits, etc.) on tooling development to bridge existing model checking tools to industry and integrate them into the CONVINCE toolchain for robust robot deliberation using formal methods. This includes algorithmic improvements, e.g., in statistical model checking engines, but also feature contributions to common modeling languages and formats. In addition, we can of course offer very interesting and challenging industrially relevant case-studies to test existing formal approaches.

[1] Gladisch, C., Heinz, T., Heinzemann, C., Oehlerking, J., von Vietinghoff, A., & Pfitzer, T. (2019, November). Experience paper: Search-based testing in automated driving control applications. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*.
[2] Heinzemann, C., & Lange, R. (2018, October). vTSL - A Formally Verifiable DSL for Specifying Robot Tasks. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
[3] König L., Heinzemann C., Griggio A., Klauck M., Cimatti A., Henze F., Tonetta S., Küperkoch S., Fassbender D., & Hanselmann M. (2024, April). Towards safe Autonomous Driving: Model Checking a Behavior Planner during Development. *In 2024 International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, accepted, to be published.
[4] Klauck M., Lange R., Henkel C., Kchir S., & Palmas M. (2024, March). Towards Robust Autonomous Robots using Statistical Model Checking. *In 2024 Springer Proceedings in Advanced Robotics (SPAR), European Robotics Forum (ERF)*, accepted, to be published.