From Concept to Prototype: Building a Retargetable Translator for Driving Simulation Languages

Marvin Schneider, Prof. Dr.-Ing. Jörn Schneider[1]

(1) Trier University of Applied Sciences, Germany

***Abstract** – Automated driving systems have introduced significant changes to the automotive industry, offering potential improvements in safety, efficiency, and convenience. The development of advanced driver assistance systems increasingly relies on integrating driving simulation into the systems and software engineering process. We highlight the pivotal role of driving simulation in meeting safety standards, particularly SOTIF (ISO 21448), and adhering to legal regulations. A key factor here is the ability to describe suitable driving scenarios as base of the simulation.*

Thus, one can design and deploy a wide range of scenarios for comprehensive testing in a safe and controlled setting. Though a challenge comes with the many different, often proprietary, driving simulation languages, that complicate the replication of comparable experiments across diverse simulation platforms. A tool that translates between different simulation languages has to be flexible enough to adapt to new languages, dialects, and standards but also has to guarantee a well-defined and correct translation.

We present a prototype of a retargetable translator designed for translating between different driving simulation languages. Furthermore, we have developed a concept for a formal semantics to verify the correctness of the translation process. This eliminates the need to formally verify the translator itself.

***Keywords:** Driving Simulator Languages, Retargetable Translator, Scenario Design, Formal Semantics*

Introduction

In recent years, driving simulation has become an essential component of automotive software engineering, supporting the development of driver assistance systems (DA) and automated driving functions (ADF). It offers advantages like early testing and validation, agile software development, and the creation of reproducible and safe virtual driving environments. Additionally, driving simulation plays a role in training AI algorithms, assessing user behaviour, and validating DA and ADF systems.

The development process for automotive software involves various contributors, from Tier-N-Suppliers to OEMs, and although it may follow agile methodologies, it still consumes significant time. This often results in diverse driving simulation languages and road networks, causing challenges in achieving consistency and replicability. Furthermore, vehicle lifecycles extend beyond the initial development, introducing legacy issues. Independent institutions, including government bodies, need to replicate driving simulation results for safety and type approval purposes.

While correct translation between different driving simulation languages alone does not guarantee reproducibility and comparability, it remains essential. The focus of this work is the description of road networks and the translation between different driving simulation languages. Proprietary formats used by various simulation software vendors have created compatibility issues. Efforts to harmonize the field are underway with the ASAM e.V. standards, including OpenDRIVE, which is continually evolving. But then there is also a need to translate between versions of the same language.

Two primary requirements for translation tools are adaptability to new source and target languages or versions and provable correctness of translation. This paper proceeds with a brief overview of the considered languages for the retargetable driving simulation language translator. The concept of a prototype of the described translator is presented, followed by an explanation of its translation process. Then our idea of translation correctness and the semantics of driving simulation languages in the context of formal programming language semantics is explained briefly. The paper concludes by summarizing the results presented.

Driving Simulation Languages

This chapter explores different languages and abstraction levels, each applied to distinct domains. Regardless of their differences, they all cover the fundamental static attributes of streets, including road configurations, lane separation, and traffic signage. However, differences become evident in data representation and the range of features they support. We examined three languages for the prototype, with a more in-depth focus on SILAB and ASAM OpenDrive.

Language: SILAB

Format: special configuration files with proprietary grammar

Domain: Driving Simulation

Description: SILAB is a simulation software that utilizes a proprietary language (which is also called SILAB in this paper for simplicity), initially developed for human-factors research. Its scope encompasses static depictions of roads, objects, and environments. Notably, SILAB excels in enabling dynamic road modifications during runtime, where the road network can be adjusted, allowing for dynamic road segment selection through algorithms or operators. It also features event definition, which can influence simulation states, and employs elements like "hedgehogs" as triggering points within lanes. The software facilitates the placement of objects with 3D models, alongside defining environmental aspects such as surface textures and mathematical terrain and tree distribution descriptions. Complex roads can be designed using specialized software tools that generate Area2 language elements, which can interconnect with other roads. Moreover, it allows integration with external software to enhance simulation functionality [WÜR].

Language: ASAM OpenDrive

Format: XML

Domain: Driving Simulation, Road Description

Description: ASAM **Open D**ynamic **R**oad **I**nformation for **V**ehicle **E**nvironment (ASAM OpenDRIVE), an XML-based open standard, evolved from a proprietary predecessor. It offers extensive capabilities for describing road and rail networks and supports complex connections between multiple roads and lanes. Additionally, it provides the specification of traffic signs, traffic lights, and specific road markings [ASS21].

Language: Okstra®

Format: XML

Domain: German Road Construction

Description: OKSTRA® is a German standard focussing on road design, documentation, and traffic data acquisition. It introduces an XML file format employed in the design, construction, and documentation of road networks. Unlike the other introduced languages, OKSTRA® was not originally designed for simulation purposes, leading to substantial distinctions in its utilization and purpose [BUN21].

These languages highlight various aspects of the real world through their different emphases. A complete representation of the real world with all its physical properties will not be possible, which is why such specialisations are necessary. Thus, when translating from one language to another, properties may be lost or properties that did not exist in the other language may be replaced by sensible default values. A similar behaviour can be observed when translating from one language into different versions.

Retargetable Translator

Our method is a system designed in a manner similar to compilers for programming languages. The front-end operates by decomposing the source file to construct an intermediate representation (IR), which is subsequently translated into the target file through the back-end. This well-established approach provides our system with adaptability, allowing seamless integration of new source or target languages. This versatility turns out beneficial in addressing evolving requirements and scenarios. For instance, the inclusion of a new source file language simply requires the development of a dedicated front-end for that particular format, leaving all other system components unaffected.

Concept

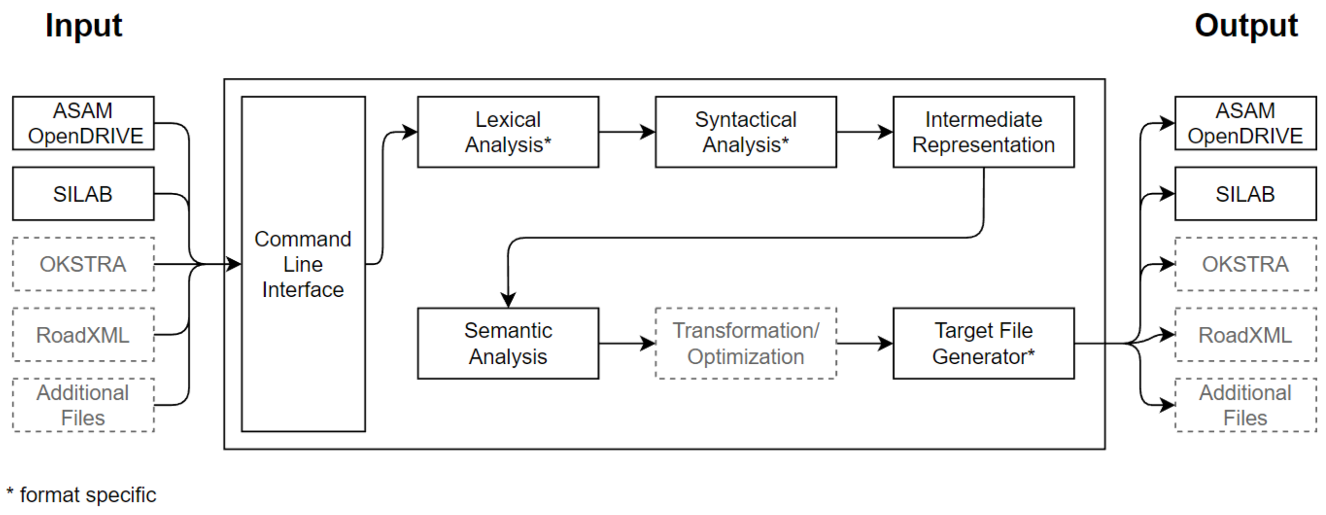

Figure 1. Basic program structure

Fig. 1. shows a schematic representation of the fundamental program structure of our prototype. Positioned on the left are the source files. Next to them is the command line interface enabling user interaction. The compilation process starts with lexical and syntactical analyses specific to the provided format. Thereafter, an intermediate representation is generated, facilitating semantic analysis and enabling subsequent optimization procedures. The final stage involves the generation of the target file, which, once again, is specific to the target format. Notably, the file formats enclosed by dashed lines, as well as the optimization processes, have yet to be implemented.

ASAM OpenDRIVE and SILAB are both partially implemented in their respective front- and back-end components, thereby enabling the potential for bidirectional translation.

Process of Translation

The initial phase involves lexical analysis, an essential process wherein language components are categorized into distinct tokens. This facilitates subsequent comprehension and manipulation of the language's structural elements. Once lexical analysis is done, the syntactic analysis proceeds, where the tokens are parsed through the application of a language-specific grammar. This grammar not only defines the syntax but also has information about the structure and arrangement of the language's components. It serves as the foundational framework for comprehending and processing the language, ensuring precise adherence to syntactic rules.

In the context of ASAM OpenDRIVE, the front-end leverages the ad-xolib7 XMLParser. In the case of SILAB, we undertook a reengineering of the grammar governing the supported constructs and proceeded to develop a bespoke parser. The implementation incorporated Flex and Bison for the lexer and parser, respectively.

To enhance system flexibility, we drafted an intermediate language. The objective of this is to achieve comprehensive coverage of the semantic aspects inherent in both the source and target languages, thereby ensuring a robust foundation for the translation process. The central advantage of employing such an intermediate representation lies in its capacity to facilitate the seamless integration of new languages into the front- and back-end components. Additionally, it provides an opportunity for optimization to be conducted on the intermediate representation itself, thus offering language-independence.

The semantic analysis examines the intermediate representation to determine the semantic accuracy of the constructed entities, ensuring that each element adheres to the intended semantic framework. Additionally, this analysis contemplates the need for potential transformations or augmentations to adjust any discrepancies, thereby aligning the elements within the intermediate representation with the desired semantic structure. It is crucial to identify any components within the intermediate representation that deviate from correct semantics, as such could potentially lead to the emergence of error states or disrupt the integrity of the system.

Within the domain of file formats, transformations or optimizations is aimed at enhancing the efficiency and effectiveness of data processing. It involves the reduction of redundant elements within the formats to reduce file size and complexity. Furthermore, the potential for optimization extends to the structural improvement of individual elements within the formats, e.g., to make the target files more organized, and making it easier to use them efficiently in simulations. Such optimization, for instance, might involve the reorganization of data within the file or prioritizing data segments that are more frequently accessed during simulation. This strategic adjustment aligns the file's layout with the demands of the simulation, resulting in simplified data retrieval and processing. The optimization process is currently only a theoretical concept, the practical implementation and evaluation remains pending.

The last phase of the process involves the transformation of the intermediate representation into the target language constructs. This is achieved through a systematic iteration over the various components within the IR to generate the requisite elements of the target format. This method ensures the accurate reconstruction of the target language, assuming that the formal semantic approach is correct. This phase is like the lexical and syntactical analysis language specific.

Intermediate Representation

The intermediate representation is an in-memory data structure and the result of the front-end. It offers a high level of flexibility. The core components of the intermediate representation are Street, StreetType, Environment, Module, and Map. Street stores information about individual roads, encapsulating details that are unique to each road segment. StreetType, on the other hand, takes on a more versatile role, as it can be linked to one or multiple Streets. Within StreetType, valuable data about the lanes, such as their specifications and characteristics, finds its place. The Environment component extends beyond the road itself, providing a detailed description of the surrounding area adjacent to the road. This includes information about the environment that can impact the driving experience, such as terrain, landmarks, and other contextual elements. Modules are used to organize and manage the road network, serving as containers for the structured representation of interconnected roads. These Modules offer a complete view of the road network, allowing for comprehensive analyses and translations. The Map is for the connections of different sections of the road network responsible. This linking mechanism enables the construction of a continuous representation of the entire network.

The selection of these specific components for the IR is not arbitrary; rather, it reflects a strategy to accommodate the diverse approaches of individual formats. Each of these components serves a distinct purpose and collectively forms a robust foundation for handling the details of road data in the translation process. However, it's important to note that the intermediate representation presented here is not exhaustive. To enhance the intermediate representation, additional analysis of languages and application areas is required.

Translation Correctness through Formal Semantics

The challenge of ensuring the accuracy of translations between different driving simulation languages shares common ground with issues encountered in the domain of compiler design. In the latter context, the translation of a high-level programming language into one or more low-level languages (e.g., assembler) requires a reliable preservation of the original code's functionality. Similarly, translating between distinct high-level programming languages presents identical challenges.

In the domain of compiler design, formal semantics for programming languages are well-established and serve as the foundation for ensuring the correctness of translations. The semantics precisely details how each language construct affects the system's state. While the question of which formal mathematical model could adequately address this aspect in the domain of driving simulation remains open, we can draw insights from the established practices of language translation within the programming domain. It is worth noting that the successful application of formal verification to commercial optimizing compilers [LER16] shows the practical feasibility of this approach.

Formal Semantics of Driving Simulator Languages

In compiler design the formal semantics of a programming language describes the effects of language elements on the state of the executing machine. Instead of the real machines with their messy details, abstract machines are used. A similar abstraction for a driving simulator would allow to use the established approaches from compiler design. However, there is no straight-forward way of coming up with an abstract driving simulator that would provide the necessary fidelity to real world driving for all use case of driving simulation. Consider for instance the training of a deep learning network for situational awareness based on LIDAR or video data.

In lack of suitable abstract machines we chose to consider the transformations from one language to another or more precisely the transcription as foundation for the formal semantics.

Instead of pursuing a 'universal' formal semantics, we proposed a universal mathematical model to describe translation semantics. This offers two advantages: it mitigates mismatches between semantic concepts in different formats and avoids the need for a unified abstract machine for driving simulators. This approach aligns with translation validation, ensuring correctness of each individual translation step [PNU98]. A more profound explanation with examples can be found in [SCH22].

Conclusion

In summary, the landscape of driving simulation languages is gaining relevance within the automotive software engineering domain. Its significance not only from its role in software development but also from its essential contribution to the future type approvals of automated driving functions. Given the current infeasibility to formally verify all components, including artificial neural networks, other concept must be examined to ensure safety and uphold professional standards. This is particularly relevant in the context of toolchains used for safety-related functionalities.

As our contribution to this dynamic field, we introduced a semantics-driven, adaptable translator for driving simulation languages. We briefly discussed the intersection of formal programming language semantics and driving simulation semantics. Future efforts could focus on developing a tool capable of demonstrating the correctness of translations on a per-instance basis, similar to approaches employed in the programming languages domain, following concepts established in prior research [PNU98] and recent developments in that field.

References

Association for Standardization of Automation and Measuring Systems. (Version 1.7.0, 2021). **ASAM OpenDRIVE V1.7.0 User Guide**.URL:https://www.asam.net/index.php?eID=dumpFile&t=f&f=4422&token=e590561f3c39aa2260e5442e29e93f6693d1cccd

Würzburger Institut für Verkehrswissenschaften. **SILAB Documentations Version SILAB 5**.

Bundesanstalt für Straßenwesen (last visited Nov, 2021). **Objektkatalog für das Straßen- und Verkehrswesen**. URL: https://www.okstra.de/

Leroy, X., Blazy, S., Kästner, D., Schommer, B., Pister, M., Ferdinand, C. (Jan. 2016). **CompCert - A Formally Verified Optimizing Compiler**. *ERTS 2016: Embedded Real Time Software and Systems*

Pnueli, A., Siegel, M., Singerman, E. (Mar. 1998). **Translation Validation**. *Springer Berlin Heidelberg: Tools and Algorithms for the Construction and Analysis of Systems.*, pp. 151-166

Schneider, J., Schneider, M. (Jan. 2022). **A Translation Semantics for Driving Simulation Languages.** *19th Workshop Automotive Software Engineering*