Investigation and Visualization of K Framework

2130026 Hiroaki Osawa

This research project introduces the K Framework, a framework that can rigorously defines the syntax and semantics of a programming language using formal methods, and proposes a tool to effectively visualize the formal semantics of a programming language described using the K Framework. The framework is a framework that enables concise and systematic definitions of programming languages based on operational semantics and has the ability to automatically generate tools such as parsers, interpreters, and model checkers. This feature provides great convenience in formalizing existing programming languages and designing new languages, and it is widely used in software development for formal semantics of programming languages such as C, Java, JavaScript, and Ethereum smart contracts. Successful examples such as the formal specification of the EVM language confirm its practicality. However, understanding the formal semantics described in the K Framework requires a precise grasp of a large number of rewriting rules, the complexity of which is a major barrier for learners and practitioners.

Based on operational semantics, the K Framework defines the semantics of a programming language using its syntax, an initial state, and a set of rewriting rules. The syntax is represented by a context-free method using BNF, and the initial state is represented by a tree structure. The state is often the main body of the program or an environment for storing variables. Rewriting rules make it possible for a language designer to explicitly only describe part of an entire state (called a configuration) being changed and omit the other parts being unchanged.

One of the characteristics of the K Framework is the mixture of rewriting rules explicitly described by the designer and implicit rules generated by the framework itself. The latter is, for example, rewriting rules necessary to realize value calls in evaluation strategies, while programming language implicit rules are essential to fully define the behavior of the language, they can also be a source of ambiguity about the designer's intentions. Therefore, when studying the semantics of a language written in the K Framework, it is not easy to identify the rules that the designer considered important and separate them from other less-important information. To mitigate this problem, this study developed a dedicated tool to effectively understand the rewriting rules of the language described in the K Framework. This tool extracts rules explicitly specified by the designer and visualizes before and after states rewritten by the specified rewriting rules.

The proposed tool was applied to various language paradigms, including procedural programing language (IMP), functional programing language

(LAMBDA), object-oriented programing language (CLASS), and a concurrent programming language (THREAD), to demonstrate the usefulness of visualization according to their characteristics. CLASS traces the behavior of object creation and method invocation, helping the reader to understand the concepts specific to object-oriented programming. In THREAD, we provide a method to clarify the complex behavior of concurrency by detailing the state transitions associated with concurrent threads creation, synchronization mechanisms, and inter-threads conflicts. These case studies demonstrate the applicability of the proposed tool to various programming language paradigms.

Furthermore, the proposed tool is unique and superior to existing visualization tools (e.g., ShiViz, SMGA). In particular, the ability to selectively visualize rewriting rules that designers consider important is a feature not found in other tools. In addition, by visually showing the order of application of rewriting rules along a time axis, the tool is designed to intuitively understanding the interaction between rules and the priority of application. This approach lowers the hurdle for learning formal semantics and shows its potential for practical as well as academic applications.

The significance of this research is that it provides a new approach to effectively understanding the formal semantics of programming languages utilizing the K Framework. This tool reduces the complexity of the semantics described in the K Framework and increases the practicality of designing and describing the semantics of programming languages using formal methods. Furthermore, due to the inherent flexibility of the K framework, it can be applied to new programming language paradigms and more complex systems in the future and has the potential to further expand the range of applications of formal methods. This research is an important step toward the practical application and dissemination of language design using formal methods and represents a new direction in the field of formal semantics.