Evaluating Kiama Abstract State Machines for a Java Implementation

by
Pongsak Suvanpong, Anthony M. Sloane

Abstract

This study evaluates the Abstract State Machines (ASM) component in Kiama. ASM is a formal method for specifying a system using the state machine concept (state-based). The specification of a model is done using the ASM notations with a clear and precise programming language constructs. There are number of tools for specifying and executing ASM models in computer. They are implemented as programming languages or modifying an existing programming language to support ASM execution model.

Kiama is a lightweight language processing library in Scala. It provides classes for language processing paradigms such as attribute grammars, strategic term rewriting and Abstract State Machines (ASM) which can be used for analyzing, translating and executing languages. Most of the components in Kiama have been evaluated and tested in number of case studies except the ASM component.

To evaluate Kiama ASM, we implement several complex ASM for executing the dynamic semantics of the Java language and the JVM. We use the book *Java and the Java Virtual Machine: Definition, Verification and Validation* by R. Stärk, J. Schmid and E. Börger, as our reference. The book has developed the formal ASM models for the dynamic semantics of the Java language and the Java Virtual Machines. The aim of our study is to closely replicate the machine definitions described in the book.

We are able to implement the machines without any modifications to Kiama's ASM. The combination of Scala and Kiama allows us to closely replicate the book's ASM notations in executable code. The transition rules in our code map one-to-one to the rules in the JBOOK. The techniques that we use to accomplish are Scala pattern matching, implicit functions and extractor patterns. The Kiama ASM library provides us with the ASM execution model and state.

The significance of this study is one can take ASM mathematical definitions and code them up in Scala to execute them and be able to use the code in an application development. We believe that the techniques that we have used to accomplish the goals in this study can be applied to other applications of ASM.