Deductive verification of the JavaBIP engine

Simon Bliudze (simon.bliudze@inria.fr)

Marieke Huisman (m.huisman@utwente.nl)

The main goal of this project is to prove the correctness of one or several encoders composing the JavaBIP coordination engine, using the VerCors deductive verifier.

Context

JavaBIP [1] is a Java-based implementation of the BIP [2] mechanisms for the coordination of concurrent software components. A runnable system in JavaBIP consists of two major parts: the *engine* and several *modules*, one for each component to be coordinated (see Figure 1). In a nutshell, a JavaBIP component extends a Java class with a behavioural *BIP specification* describing a Finite State Machine through Java annotations.

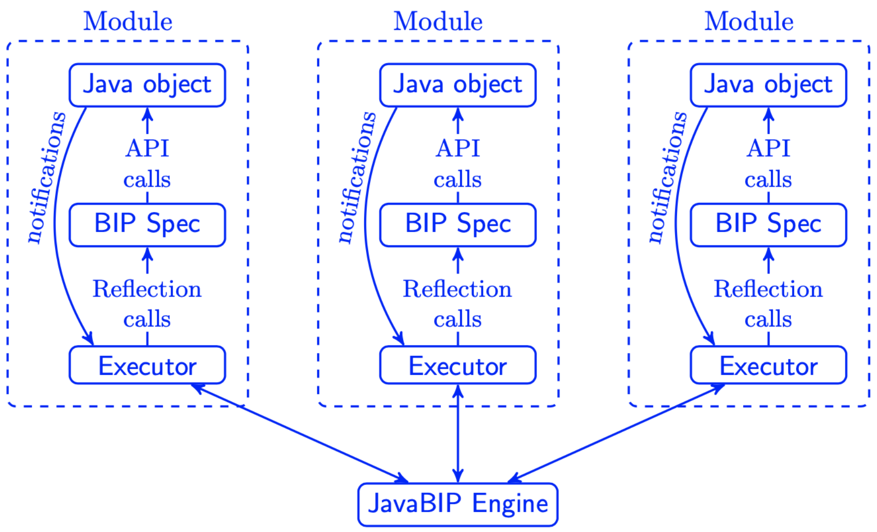

Figure 1: High-level view of the JavaBIP runnable system architecture

The coordination constraints are specified in terms of *glue* and *data wires*. The glue consists of synchronisation constraints encoding the set of possible interactions among the ports of the components. Data transfer is specified as a set of data wires connecting required inputs with provided outputs of the components.

The behaviour specification of each component along with the glue and data wire specifications are provided to the engine. The engine orchestrates the overall execution of the system by 1) deciding which component transitions must be executed at each cycle, 2) transferring the necessary data, and 3) triggering the execution of the selected transitions.

The JavaBIP engine consists of a collection of dedicated encoders transforming specification elements (behaviour, glue, data wires) into Boolean constraints (see Figure 2). The engine kernel stores the conjunction of these constraints as a Binary Decision Diagram (BDD) used to compute the interactions to be fired at each execution cycle.

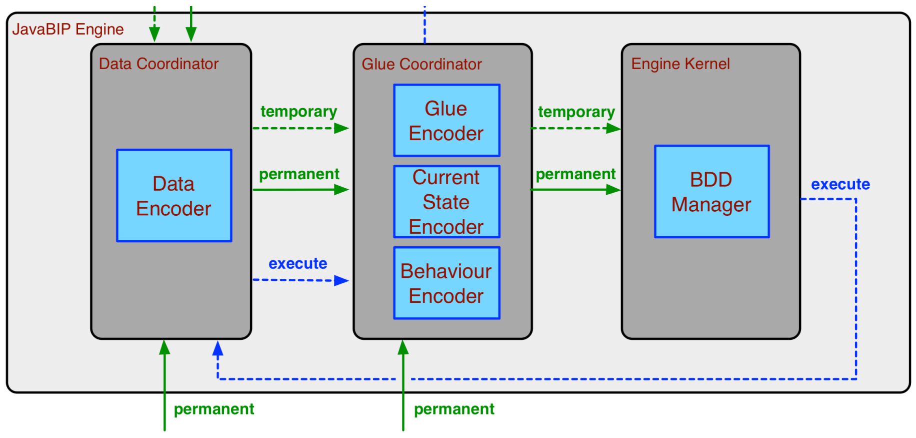

Figure 2: JavaBIP engine architecture

Project goals

The BDD representation used for the encoding of JavaBIP models allows the efficient execution of JavaBIP systems but makes it very hard to test and debug the engine. On the other hand, such Boolean encodings and the modular architecture of the engine make it well suited for deductive verification. The main goal of this project is, thus, to prove the correctness of one or several encoders composing the JavaBIP engine, using the VerCors deductive verifier [3]. After studying the source code of the selected encoder, the student will have to 1) formalise the equivalence between the input model and the Boolean encoding, 2) design the require and gurantee annotations necessary to carry out deductive verification using VerCors, and 3) interpret the results and, if necessary, propose appropriate bug fixes.

Benefits

You will learn the principles of correct-by-construction design of concurrent software based on formal operational semantics and get an in-depth understanding of BIP, a state-of-the-art component-based framework. Successful project realisation can lead to a research publication.

Required skills

Good analytical skills will definitely be required. The candidate must have good understanding of Labelled Transition Systems and Finite State Automata. The work will be based on Java sources, so proficiency in Java is a must.

Contact and application

For additional information and to apply please send an e-mail to Simon Bliudze and Marieke Huisman with the subject “JavaBIP verification”.

References

1. Simon Bliudze, Anastasia Mavridou, Radoslaw Szymanek, and Alina Zolotukhina. Exogenous coordination of concurrent software components with JavaBIP. *Software: Practice and Experience*, **47**(11):1801–1836, November 2017. PDF
2. Ananda Basu, Saddek Bensalem, Marius Bozga, Jacques Combaz, Mohamad Jaber, Thanh-Hung Nguyen, Joseph Sifakis. Rigorous component-based system design using the BIP framework. *IEEE Software* **28**(3) (2011) 41–48 PDF
3. Stefan Blom, Saeed Darabi, Marieke Huisman, Wytse Oortwijn. The VerCors toolset: Verification of parallel and concurrent software. In: *IFM*. Lecture Notes in Computer Science, vol. 10510, pp. 102–110. Springer (2017), DOI: 10.1007/978-3-319-66845-1_7