Philipps-Universität Marburg
Fachbereich Mathematik und Informatik
AG Programmiersprachen und -werkzeuge

Philipps Universität Marburg

Domain-Specific Consistency Constraints for Java Bytecode

Motivation

Java bytecode is a very versatile intermediate representation to which several languages (Java, Scala, Groovy, etc.) are compiled. For all available libraries, even if they are closed-source, the bytecode is always present, which is why many code analysis tools today process Java bytecode. The programming languages and tools group is developing a toolkit for program analyses and transformations of Java bytecode. The toolkit is named Modular Bytecode Engineering and Analysis based on Models, or ModBEAM for short. It contains a metamodel of Java bytecode defined in the ECORE format of the "Eclipse Modeling Framework" (EMF) which is the prevalent standard in model-driven engineering (MDE). ModBEAM also provides and Eclipse plug-in which can represent Java bytecode files as models according to its metamodel. This allows the usage of a wide variety of model analysis and transformation tools, available in the EMF ecosystem, to inspect and possibly modify the bytecode model; this facilitates developing modular and powerful bytecode-based tools. Afterwards, ModBEAM can again generate regular Java bytecode from the model, such that the possibly modified bytecode can be executed.

Assignment

Java bytecode has some consistency constraints (for example the type of the returned value must fit the declared method result type) which can easily be violated by code transformations. The ECORE formalism for metamodels already allows defining consistency constraints in terms of predicates in the Object Constraint Language (OCL). A number of OCL predicates have already been defined for the ModBEAM metamodel corresponding to basic constraints from the Java Virtual Machine Specification. In this assignment further constraints should be identified and defined in OCL. Possible sources of such advanced constraints are architectural patterns (e.g., in the *Abstract Factory* pattern, only the factory is allowed to call the constructor of certain classes) or permissions in defined in Java modules.

Further Reading

- Christoph Bockisch, Gabriele Taentzer, Nebras Nassar, and Lukas Wydra. Java bytecode verification with ocl why, how and when? *Journal of Object Technology*, 19(3):3:1–16, October 2020. Special Issue dedicated to Martin Gogolla on his 65th Birthday. doi:10.5381/jot.2020.19.3.a13
- Jordi Cabot and Martin Gogolla. Object constraint language (ocl): A definitive guide. volume 7320, pages 58–90, 06 2012. doi:10.1007/978-3-642-30982-3_3

Info

Bytecode, MDE

Java, EMF, OCL

Bachelor or Master Thesis

Theory
Practice

Contact

Prof. Dr. Christoph Bockisch

bockisch@
mathematik.
uni-marburg.de