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

A Scalable Representation of Java Bytecode Models

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

EMF provides a standard representation for models, which is also available for ModBEAM models, based on a tree-structure. However, models in general and bytecode models in specific rather have a graph structure than a tree. In particular the bytecode models of ModBEAM contain the control-flow graph formed by the instructions in the bytecode. All in all, the standard visualization is not very easy to read and also not very scalable: each instruction in the bytecode corresponds to at least two model objects and therefore requires at least two lines in the default tree view.
In this project, a new visualization should be developed that (ideally) combines a compact textual representation for blocks of sequentially executed instructions and a graphical representation of the control flow. For other entities like classes, methods and fields also an appropriate visualization should be developed.
Within the EMF ecosystem, there are several frameworks available for generating custom visualizations of models. Most prominent are xText for textual representations or GEF and GMP for graphical representations. The main purpose of this visualization is to quickly generate images of ModBEAM models for illustration. Therefore, it should also be possible to manually edit the layout of the generated (graphical) representation. Being able to edit the model through representation is only an optional requirement.

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`

- Bugra M. Yildiz, Christoph Bockisch, Arend Rensink, and Mehmet Aksit. An mde approach for modular program analyses. In *Companion Proceedings of the 1st International Conference on the Art, Science, and Engineering of Programming*, Programming '17, New York, NY, USA, 2017. Association for Com-

Info

Bytecode, MDE

Java, EMF

Bachelor or Master Thesis

Theory ▰▰▱▱▱
Practice ▰▰▰▱▱

Contact

Prof. Dr. Christoph Bockisch

`bockisch@mathematik.uni-marburg.de`

puting Machinery. `doi:10.1145/3079368.3079392`

- Homepage of the Xtext project. `https://eclipse.dev/Xtext/`. Accessed: 2023-10-26
- Homepage of the Graphical Editing Framework (GEF) project. `https://projects.eclipse.org/projects/tools.gef`. Accessed: 2023-10-26
- Homepage of the Graphical Modeling Project (GMP). `https://eclipse.dev/modeling/gmp/`. Accessed: 2023-10-26