Philipps-Universität Marburg
Department of Mathematics and Computer Science
Programming Languages and Tools group

Semantic Comparison of Java Bytecode

Context: The ModBEAM Project

Java bytecode is a very versatile intermediate representation to which several languages (Java, Scala, Groovy, etc.) are compiled. It is always available even for closed-source code therefore targeted by many modern code analysis tools. The programming languages and tools group is developing a toolkit for program analyses and transformations of Java bytecode, named *Modular Bytecode Engineering and Analysis based on Models* (*ModBEAM*) [2, 4]. It contains a *meta model*[1] of Java bytecode defined in the ECORE format of the "Eclipse Modeling Framework" (EMF), the prevalent standard in model-driven engineering. ModBEAM also provides and *Eclipse plug-in* which can represent Java bytecode files as *instance models*[2] according to its meta model. This allows the usage of a wide variety of powerful model analysis and transformation tools, available in the EMF ecosystem, to inspect and possibly modify the bytecode model. For possibly transformed bytecode models, the ModBEAM plug-in can again generate regular Java bytecode.

Motivation

Java bytecode is produced by multiple different tools, for example, there are multiple Java source code compilers, compilers for other languages or other code generators. One particularly interesting use case are tools which rewrite bytecode in certain ways; popular examples are code instrumentations for purposes such as profiling or dependency injection. To test such tools, the produced bytecode can be compared to the expected bytecode. However, comparing the bytecode for identity may be too restrictive. For example, the order in which entities like methods are defined in the bytecode may be different without affecting the meaning of the bytecode.

Assignment

To facilitate a more lenient comparison between two Java bytecode files, a Java library should be developed that tolerates small differences in the code which are known not to affect the meaning of bytecode. To read and process Java bytecode, several tools are available, probably the most suitable of these is the ASM bytecode toolkit [1]. The developed library should be able to read two different Java bytecode files and compare them, whereby it should be possible to configure which differences are tolerated. In case an unacceptable difference is found, a meaningful message should be created that describes the location of the difference in the bytecode (i.e., which method, which instruction, etc.), what was expected and what was found instead. The developed library should be able to compare all language constructs (e.g. annotations, generics, etc.) supported by the most recent Java version.
The tool SootDiff [3] has a similar goal and can even in some cases recognize of different instruction sequences are semantically equivalent. However, it only supports Java 7, and it may be impossible to also opt for stricter comparison with SootDiff. The suitability of SootDiff should, nevertheless, be investigated in this project and the newly developed library should be compared with SootDiff. Other related approaches are comparison tools for JSON objects with different comparison modes or more general diff algorithms.

References

Info

Java bytecode, language processors

Java

ASM

Research
Develop

Thesis (BSc, MSc)

Prof. Christoph Bockisch

[1]If the term meta model is not known, think of is as the classes in an object-oriented program.
[2]Assuming a meta model is comprised of classes, think of an instance model as thier runtime objects in a specific execution.

[1] Homepage of the ASM bytecode manipuation and analysis framework. `https://asm.ow2.io/`. Accessed: 2024-06-09.

[2] 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`.

[3] Andreas Dann, Ben Hermann, and Eric Bodden. Sootdiff: Bytecode comparison across different java compilers. In *Proceedings of the 8th ACM SIGPLAN International Workshop on State of the Art in Program Analysis*, pages 14–19, 2019.

[4] Bugra M. Yildiz, Christoph Bockisch, Arend Rensink, and Mehmet Aksit. An MDE approach for modular program analyses. In *Companion Proceedings of Programming' 17*. ACM, 2017. `doi:10.1145/3079368.3079392`.