Lars Grunske, Janet Siegmund, Andreas Vogelsang (Hrsg.): SE 2022,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2022 81

CiFi: Versatile Analysis of Class and Field Immutability

Tobias Roth, Dominik Helm, Michael Reif, Mira Mezini[1]

Abstract: This paper was accepted in 2021 at the 36th IEEE/ACM International Conference on Automated Software Engineering and proposes a model for immutability analysis. Reasoning about immutability is important for preventing bugs, e.g., in multi-threaded software. Static analysis to infer immutability properties has mostly focused on individual objects and references. Reasoning about fields and entire classes, while significantly simpler, has gained less attention. A consistently used terminology is missing, which makes it difficult to implement analyses that rely on immutability information. We propose a model for class and field immutability that unifies terminology for immutability flavors considered by previous work and covers new levels of immutability to handle lazy initialization and immutability dependent on generic type parameters. Using the OPAL static analysis framework, we implement CiFi, a set of modular, collaborating analyses for different flavors of immutability, inferring the properties defined in our model. We propose a benchmark of representative test cases for class and field immutability. We use the benchmark to showcase CiFi's precision and recall in comparison to state of the art and use CiFi to study the prevalence of immutability in real-world libraries, showcasing the practical quality and relevance of our model.

Keywords: class and field immutability; static analysis; lattice; Java

1 Summary

Immutability brings important guarantees and is, e.g., recommended for secure coding [Or20]. Intuitively, immutability means that a program element is unchangeable or not changed after its creation [Po13]. While there are multiple flavors and levels of immutability, current approaches only focus on a restricted set of immutability levels and additionally cannot handle lazy initialization and immutability in combination with generic types properly. Furthermore, there is not only no common terminology yet for different immutability levels, but also existing terminology is used inconsistently [Co17, Po00, NPN12, Po13].

To solve this problem, we propose a unified terminology and present it in our lattice model that focuses on class and field immutability. Analyzing class immutability is much simpler than analyzing object immutability while remaining sound [Co16, Co17]. The model is divided into four different lattices for *field assignability*, *field immutability*, *class immutability*, and *type immutability*. Because current approaches cannot handle lazy initialization of fields and immutability in combination with Java generics precisely, we introduced the novel immutability levels *(unsafe) lazy initialization* and *dependent immutability*. (Unsafe) lazy

[1] Technische Universität Darmstadt, FG Softwaretechnik, Germany
{roth,helm,mezini}@cs.tu-darmstadt.de,mi.reif.mr@gmail.com

initialization describes whether all reads of a field always return the same value. Dependent immutability describes cases where the immutability of a type, class or field depends on the concretization of a generic type. This model allows for modular immutability analyses for field assignability and field, class, and type immutability.

Based on our model, we implemented CiFi that encompasses four different analyses for field assignability, field, class, and type immutability in the Java Bytecode analysis framework OPAL [He20]. OPAL's blackboard architecture supports the composition of multiple decoupled interdependent analyses. In our evaluation, we show the expressiveness of our model and challenge CiFi with our CiFi-Benchmark for field and class immutability, which we created based on our immutability research. We show that CiFi handles most of the test-cases precisely and over-approximates remaining corner-cases soundly. The results of analyzing several real-world libraries with CiFi show the applicability and relevance of our immutability model in the real world. Furthermore, CiFi outperforms the state of the art in class- and field-immutability enforcement, Glacier [Co17].

2 Data Availability

CiFi: `https://github.com/opalj/opal/tree/feature/classFieldImmutability`
CiFi-Benchmark: `https://github.com/opalj/CiFi-Benchmark`
Artifact: `https://doi.org/10.5281/zenodo.5227231`

Bibliography

[Co16] Coblenz, Michael; Sunshine, Joshua; Aldrich, Jonathan; Myers, Brad; Weber, Sam; Shull, Forrest: Exploring language support for immutability. In: 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). pp. 736–747, 2016.

[Co17] Coblenz, Michael; Nelson, Whitney; Aldrich, Jonathan; Myers, Brad; Sunshine, Joshua: Glacier: Transitive class immutability for Java. In: 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE). IEEE, pp. 496–506, 2017.

[He20] Helm, Dominik; Kübler, Florian; Reif, Michael; Eichberg, Michael; Mezini, Mira: Modular collaborative program analysis in OPAL. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE'20, pp. 184–196, 2020.

[NPN12] Nelson, Stephen; Pearce, David J; Noble, James: Profiling field initialisation in Java. In: International Conference on Runtime Verification. Springer, pp. 292–307, 2012.

[Or20] Oracle: Secure Coding Guidelines for Java SE. https://www.oracle.com/java/technologies/javase/seccodeguide.html, September 2020.

[Po00] Porat, Sara; Biberstein, Marina; Koved, Larry; Mendelson, Bilha: Automatic detection of immutable fields in Java. In: CASCON. p. 10, 2000.

[Po13] Potanin, Alex; Östlund, Johan; Zibin, Yoav; Ernst, Michael D.: Immutability. In: Aliasing in Object-Oriented Programming, pp. 233–269. Springer-Verlag, Berlin, Heidelberg, 2013.