Definite Reference Mutability (Artifact)*

Ana Milanova
Department of Computer Science, Rensselaer Polytechnic Institute
110 8th Street, Troy NY, USA
milanova@cs.rpi.edu

Wei Huang[1]
Google
huangwe@google.com

Abstract

Related paper "Definite Reference Mutability" presents ReM (Re[ference] M[utability]), a type system that separates mutable references into (1) definitely mutable, and (2) maybe mutable, i.e., references whose mutability is due to inherent approximation. We have implemented ReM and applied it on a large benchmark suite. Results show that ≈ 86% of mutable references are definitely mutable.

This article describes the tool artifact from the related paper. The purpose of the article and artifact is to allow researchers to reproduce our results, as well as build new type systems upon our code.

2012 ACM Subject Classification Theory of computation → Program analysis, Software and its engineering → General programming languages

Keywords and phrases reference immutability, type inference, CFL-reachability

Digital Object Identifier 10.4230/DARTS.4.3.7

Related Article Ana Milanova, "Definite Reference Mutability", in Proceedings of the 32nd European Conference on Object-Oriented Programming (ECOOP 2018), LIPIcs, Vol. 109, pp. 25:1–25:31, 2018. `https://dx.doi.org/10.4230/LIPIcs.ECOOP.2018.25`

Related Conference 32nd European Conference on Object-Oriented Programming (ECOOP 2018), July 19–21, 2018, Amsterdam, Netherlands

1 Scope

In previous work we developed a framework for inference and checking of pluggable types [5, 4]. Users instantiate the framework with certain parameters to define a type system. The framework takes as input a program (typically only partially annotated or not annotated at all), infers types for all variables and type checks the inferred types. We have instantiated the framework with known type systems and new ones. These include classical Ownership types [1, 5], Universe types [2, 5], ReIm reference immutability types [8], Information flow types for the detection of privacy leaks in Android apps [6], and AJ types for data centric synchronization [10, 3, 7].

The artifact builds upon this framework. Package `edu.rpi` is the heart of the framework: it includes type annotation utilities, visitors and a generic constraint solver. It is built on top of Soot [9]. Package `edu.rpi.reim` contains instantiations of ReIm and ReM. An instantiation introduces type-system-specific type qualifiers, initialization rules and typing rules, possibly overriding default rules defined in generic `InferenceTransformer` in package `edu.rpi`. For the majority of cases, ReIm and ReM reuse rules from the generic transformer.

The key purpose of this artifact is to reproduce and validate the claims of the related ECOOP paper. In addition, we invite researchers to build new type systems upon our framework.

* This work was partially supported by NSF grant 1319384.
[1] Work done while author was a PhD student at Rensselear Polytechnic Institute.

2 Content

The artifact package includes:

- `bin` - directory contains compiled code
- `src` - directory contains all source code
- `lib` - directory contains all libraries: `soot-develop.jar` and `rt.jar` necessary to compile and run the code. We include the `rt.jar` from jdk1.7.0_75 for MacOS. (It can be downloaded from the Oracle website: `http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html`.) The artifact requires a Java 7 `rt.jar`.
- `bench` - directory contains all benchmarks from the related paper
- `run-tests` - a script that automatically runs tool with benchmarks
- `README` - a description of artifact

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition, the artifact is available at: `http://www.cs.rpi.edu/~milanova/soot-reim-definite.zip`.

Source code for the framework, including all type systems, is available on GitHub: `https://github.com/proganalysis/type-inference`.

4 Tested platforms

1. Mac OS X El Capitan, 2.8 GHz Intel Core i7, 16 GB RAM. Java version 1.8.0_71.
2. Ubuntu 16.04.4 LTS, Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz, 32 GB RAM. Java version 1.8.0_171.

The tool runs as is on these platforms using default maximal heap size.

5 License

6 MD5 sum of the artifact

590b9f9b3160a342dcf2d71abb6c7585

7 Size of the artifact

120397581 B

Acknowledgements. We thank the ECOOP 2018 Artifact Evaluation committee and the ECOOP 2018 Program committee for valuable suggestions, and the National Science Foundation for supporting our work under NSF grant 1319384.

References

1 David G. Clarke, John M. Potter, and James Noble. Ownership types for flexible alias protection. In *Proceedings of the 13th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, OOPSLA '98, pages 48–64, New York, NY, USA, 1998. ACM. URL: http://doi.acm.org/10.1145/286936.286947, doi:10.1145/286936.286947.

2 Werner Dietl and Peter Müller. Universes: Lightweight ownership for JML. *Journal of Object Technology*, 4(8):5–32, 2005.

3 Julian Dolby, Christian Hammer, Daniel Marino, Frank Tip, Mandana Vaziri, and Jan Vitek. A data-centric approach to synchronization. *ACM Transactions on Programming Languages and Systems*, 34(1):1–48, April 2012.

4 Wei Huang. *An inference and checking framework for context-sensitive pluggable types*. PhD thesis, Rensselaer Polytechnic Institute, 2014.

5 Wei Huang, Werner Dietl, Ana Milanova, and Michael D. Ernst. Inference and checking of object ownership. In *Proceedings of the 26th European Conference on Object-Oriented Programming*, ECOOP'12, pages 181–206, Berlin, Heidelberg, 2012. Springer-Verlag. URL: http://dx.doi.org/10.1007/978-3-642-31057-7_9, doi:10.1007/978-3-642-31057-7_9.

6 Wei Huang, Yao Dong, Ana Milanova, and Julian Dolby. Scalable and precise taint analysis for android. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis*, ISSTA 2015, pages 106–117, New York, NY, USA, 2015. ACM. URL: http://doi.acm.org/10.1145/2771783.2771803, doi:10.1145/2771783.2771803.

7 Wei Huang and Ana Milanova. Inferring AJ types for concurrent libraries. In *Workshop on the Foundations of Object-oriented Languages (FOOL)*, 2012.

8 Wei Huang, Ana Milanova, Werner Dietl, and Michael D. Ernst. Reim & reiminfer: Checking and inference of reference immutability and method purity. In *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications*, OOPSLA '12, pages 879–896, New York, NY, USA, 2012. ACM. URL: http://doi.acm.org/10.1145/2384616.2384680, doi:10.1145/2384616.2384680.

9 Raja Vallée-Rai, Phong Co, Etienne Gagnon, Laurie J. Hendren, Patrick Lam, and Vijay Sundaresan. Soot - a java bytecode optimization framework. In *Proceedings of the 1999 conference of the Centre for Advanced Studies on Collaborative Research, November 8-11, 1999, Mississauga, Ontario, Canada*, page 13, 1999. URL: http://doi.acm.org/10.1145/781995.782008, doi:10.1145/781995.782008.

10 Mandana Vaziri, Frank Tip, Julian Dolby, Christian Hammer, and Jan Vitek. A type system for data-centric synchronization. In *Proceedings of the 24th European Conference on Object-oriented Programming*, ECOOP'10, pages 304–328, Berlin, Heidelberg, 2010. Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=1883978.1884000.