Program Tailoring: Slicing by Sequential Criteria (Artifact)

Tian Tan[1], Yue Li[2], Yifei Zhang[3], and Jingling Xue[4]

1 **School of Computer Science and Engineering, UNSW Australia**
`tiantan@cse.unsw.edu.au`
2 **School of Computer Science and Engineering, UNSW Australia**
`yueli@cse.unsw.edu.au`
3 **School of Computer Science and Engineering, UNSW Australia**
`yzhang@cse.unsw.edu.au`
4 **School of Computer Science and Engineering, UNSW Australia**
`jingling@cse.unsw.edu.au`

Abstract

Protocol and typestate analyses often report some sequences of statements ending at a program point P that needs to be scrutinized, since P may be erroneous or imprecisely analyzed. Program slicing focuses only on the behavior at P by computing a slice of the program affecting the values at P. In our companion paper "Program Tailoring: Slicing by Sequential Criteria", we propose to focus on the subset of that behavior at P affected by one or several statement sequences, called a *sequential criterion (SC)*. By leveraging the ordering information in a *SC*, e.g., the temporal order in a few valid/invalid API method invocation sequences, we introduce a new technique, *program tailoring*, to compute a tailored program that comprises the statements in all possible execution paths passing through at least one sequence in *SC* in the given order.

This artifact is based on TAILOR[1], a prototyping implementation of program tailoring, to evaluate the usefulness of TAILOR in practice. The provided package is designed to support repeatability of all the experiments of our companion paper. Specifically, it allows users to reproduce the results for all the three research questions addressed in the evaluation section of our companion paper. In addition, an extensive set of extra results, which are not described in the companion paper, are also included, in order to help users better understand this work.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages - Program Analysis, D.2.5 Testing and Debugging - Code inspections and Debugging aids

Keywords and phrases Program Slicing, Program Analysis, API Protocol Specification

Digital Object Identifier 10.4230/DARTS.2.1.8

Related Article Yue Li, Tian Tan, Yifei Zhang, and Jingling Xue, "Program Tailoring: Slicing by Sequential Criteria", in Proceedings of the 30th European Conference on Object-Oriented Programming (ECOOP 2016), LIPIcs, Vol. 56, pp. 15:1–15:27, 2016.
`http://dx.doi.org/10.4230/LIPIcs.ECOOP.2016.15`

Related Conference 30th European Conference on Object-Oriented Programming (ECOOP 2016), July 18–22, 2016, Rome, Italy

1 Scope

This artifact is provided to enable the results for all three research questions (RQs 1 - 3) in our companion paper to be reproduced. In the paper, we have conducted a thorough evaluation with three sets of experiments, including two large cases studies for two different application scenarios

[1] TAILOR is available at `http://www.cse.unsw.edu.au/~corg/tailor`.

in order to evaluate Tailor's usefulness (RQ1 and RQ2) and one stress test in order to measure Tailor's scalability (RQ3). In RQ1, we show how Tailor can improve the precision of thin slicer for program debugging and understanding. In RQ2, we show how Tailor can make program analysis scale better with improved precision. In RQ3, we provide a stress test to investigate how well Tailor scales, in practice.

Please note that this artifact is large and complex. It involves two clients (Clara [1] and Solar [2]), two slicing tools (Thin slicer and Tailor), three static analysis frameworks (Doop [3], Wala [4] and Soot [5]), with complicated interactions among these clients and tools. In addition, sophisticated and memory-consuming whole-program pointer analyses will be performed on a set of seven large real-world Java applications under a large library.

2 Content

The artifact package includes:

- `index.html`: Detailed instructions about how to use this artifact.
- `artifactStructure.pdf`: A figure describing the structure of this artifact.
- `run.py`: A Python script for driving all the provided analyses.
- `executable`: The folder containing all the tools used, together with JRE 1.6 and the applications analyzed in our evaluation. In particular, these tools are Tailor, the thin slicer implemented in Wala, Clara (a tool that generates *SCs* for RQ1), and Solar (a tool that generates *SCs* for RQ2).
- `SCs`: The folder containing all *SCs* used in RQs 1 - 3.
- `screenshots`: The folder containing the screenshots of the outputs of this artifact.
- `outputs(given)`: The folder containing all the detailed experimental results which are given in advance for users to reference conveniently without running the 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). The latest version of the artifact and the code of Tailor are available at: `http://www.cse.unsw.edu.au/~corg/tailor`.

4 Tested platforms

This artifact is known to work on Linux. It requires a Java 1.8 distribution and a Python interpreter (between 2.7 and 3.0). As explained in Section 1, the artifact is large and complex. Therefore, a machine with a large memory size (with ours being 64GB) is required in order to reproduce all the results reported in our paper. We performed all the experiments described in the companion paper on a Ubuntu 14.04 LTS machine with Xeon E5-2650 2GHz CPU that is equipped with 64GB RAM. We used Java 1.8.0_25 and Python 2.7.6.

5 License

GPL v3 (`http://www.gnu.org/licenses/gpl.html`)

6 MD5 sum of the artifact

e6fccf02cf279d8a7a4c442919aca52f

7 Size of the artifact

569 MB

References

1 Clara. http://www.bodden.de/clara.
2 Solar. http://www.cse.unsw.edu.au/∼corg/solar.
3 Doop. http://doop.program-analysis.org.
4 Wala. http://wala.sf.net.
5 Soot. https://sable.github.io/soot.