Bachelor Thesis Description: Synthesizing Method Sequences to Detect Object Invariant Violations

Timon Gehr

1 Abstract

During my bachelor thesis, I will develop an under-approximating software validation tool based on dynamic symbolic execution with special emphasis on detecting possible violations of visible-state object invariants, and the effects of these violations. More specifically, the tool should be able to synthesize a sequence of public operations after execution of which the negation of the invariant of a live object holds in case this is possible. Furthermore, the tool should attempt to find a method that breaks when the invariant is violated in this way. As these problems are undecidable in general, the tool should apply heuristics in order to cover a subset of cases. The developed approach shall be documented and its effectiveness investigated by the means of a case study of significant size.

2 Setup

An academic licence for the source code of the concolic testing tool Pex, which is developed at Microsoft Research, has been acquired and has been made available to us. The tool will be developed as an extension to Pex, and the final product will therefore be CLI based. It will leverage specifications expressed in Code Contracts, which Pex already supports.

3 Project Definition

The project is subdivided spatially into a core part and multiple extensions that improve on the former. The basis of the core part consists in the exhaustive generation of all method sequence skeletons on a single receiver up to a user-specified bound. The existing tool Pex can then be run on them in order to find suitable argument values. Method sequences that indeed result in an invariant violation can then be extended by one additional method call in order to attempt to cause a crash in that method. On that basis, more elaborate strategies are built. The following are included in the core part: Specifically exploit public fields. Also use method return values as further arguments and receivers. Specifically exploit subclasses with stronger invariants. Detect and exploit invariant-breaking patterns. Specifically treat multi-object invariants. Use the approach recursively for test input generation. Possible extensions are: Prune methods based on read and write effects. Prune paths based on read and write effects. Prioritize methods other tools cannot verify to be invariant-preserving. Prune methods that are subsumed by other methods. Optimize performance, for example by profiling and modification, or parallelization. The project is subdivided temporally into the following phases:

3.1 Reading Phase

Description I will read research papers that have a goal similar to mine in order to get familiar with related work. Furthermore, I will study some of the Pex source code in order to obtain an understanding of how an extension may or may not be implemented.

Projected time frame October 2012.

Deliverables Initial bachelor thesis presentation.

3.2 Implementation Phase

Description The goal of this phase is to obtain a working tool. It should be tested using plausible real-world examples as soon as possible. Time is allocated for the implementation of extensions.

Projected time frame November and December 2012.

Deliverables Implementation for the core part. Additional improvements.

3.3 Evaluation Phase

Description During the evaluation phase, the test suite is extended so that it can be used to obtain clues about the effectiveness of the developed approach. The tool created during the implementation phase is evaluated and the results are collected. Optionally, the tool may be extended such that it can be compared to existing work whose goal is to maximize code coverage. Time is allocated for the implementation of further extensions.

Projected time frame January 2013.

Deliverables Evaluation results.

3.4 Writing Phase

Description Finally, the foregone efforts are documented in the final report. The evaluation results are presented and interpreted.

Projected time frame February 2013.

Deliverables Final report. Final presentation.