EvoSpex: An Evolutionary Algorithm for Learning Postconditions

Facundo Molina[1,3], Pablo Ponzio[1,3], Nazareno Aguirre[1,3], and Marcelo Frias[2,3]

[1] Universidad Nacional de Río Cuarto, Argentina
[2] Instituto Tecnológico de Buenos Aires, Argentina
[3] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina

Abstract

Software reliability is a primary concern in the construction of software, and thus a fundamental component in the definition of software quality. Analyzing software reliability requires a *specification* of the intended behavior of the software under analysis, and at the source code level, such specifications typically take the form of *assertions*. Unfortunately, software many times lacks such specifications, or only provides them for scenario-specific behaviors, as assertions accompanying tests. This issue seriously diminishes the analyzability of software with respect to its reliability, emphasizing the well known Oracle Problem.

We tackle this problem by proposing a technique that, given a Java method, automatically produces a postcondition assertion of the method's current behavior. This mechanism is based on generating *valid* and *invalid* pre/post state pairs (i.e., state pairs that represent, and do not represent, the method's current behavior, respectively), which guide EvoSpex, a genetic algorithm, to produce a JML-like assertion characterizing the valid pre/post pairs, while leaving out the invalid ones. The generation of valid pre/post pairs is based on executing the method on a *bounded exhaustive* test set, generated by exercising the method inputs' APIs. The invalid pairs, on the other hand, are obtained by *mutating* valid pairs modifying the post-states so that each resulting pair does not belong to the set of valid pairs. EvoSpex searchs for an adequate postcondition assertion expressed in a JML-like language that involves quantification, object navigation and reachability expressions, making our approach particularly well-suited for reference-based class implementations with (implicit) strong representation invariants, such as heap-allocated structural objects, and complex custom types.

We evaluated our technique on a benchmark of open source Java projects, featuring complex implementations of reference-based classes. In these case studies, EvoSpex was able to generate post-conditions that were stronger and more accurate, than those generated by related specification-inference approaches, as evaluated by an automated oracle assessment tool. Moreover, EvoSpex was also able to infer an important part of manually written rich postconditions (strong contracts used for verification) present in verified classes, and reproduce contracts for methods whose class implementations were automatically synthesized from specifications.

This work was published at the *43rd International Conference on Software Engineering (ICSE 2021)* held virtually on 25-28 May 2021. DOI