Efficient Bounded Exhaustive Input Generation from Program APIs

Mariano Politano[1,4], Valeria Bengolea[1], Facundo Molina[3], Nazareno Aguirre[1,4], Marcelo F. Frias[2,4], and Pablo Ponzio[1,4]

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

Abstract

Bounded exhaustive input generation (BEG) is an effective approach to reveal software faults. However, existing BEG approaches require a precise specification of the *valid* inputs, i.e., a `repOK`, that must be provided by the user. Writing `repOK`s for BEG is challenging and time consuming, and they are seldom available in software. Several studies show that BEG approaches are effective in revealing software failures. Furthermore, the *small scope hypothesis*, which states that most software faults can be revealed by executing the SUT on "small inputs", suggests that BEG approaches should discover most (if not all) faults in the SUT, if large enough scopes are used. The challenge that BEG approaches face is how to efficiently explore a huge search space, that often grows exponentially with respect to the scope. Thus, pruning parts of the search space involving invalid and redundant inputs is key to make BEG approaches scale up in practice.

In this work, we introduce BEAPI, an efficient approach that employs routines from the API of the software under test to perform BEG. Like API-based test generation approaches, BEAPI creates sequences of calls to methods from the API, and executes them to generate inputs. As opposed to existing BEG approaches, BEAPI does not require a `repOK` to be provided by the user. To make BEG from the API feasible, BEAPI implements three key pruning techniques: *(i)* discarding test sequences whose execution produces exceptions violating API usage rules, *(ii)* state matching to discard test sequences that produce inputs already created by previously explored test sequences, and *(iii)* the automated identification and use of a subset of methods from the API, called *builders*, that is sufficient to perform BEG.

Our experimental assessment shows that BEAPI's efficiency and scalability is competitive with existing BEG approaches, without the need for `repOK`s. We also show that BEAPI can assist the user in finding flaws in `repOK`s, by (automatically) comparing inputs generated by BEAPI with those generated from a `repOK`. Using this approach, we revealed several errors in `repOK`s taken from the assessment of related tools, demonstrating the difficulties of writing precise `repOK`s for BEG.

This work was published at *26nd International Conference on Fundamental Approaches to Software Engineering* on May 2023. `https://link.springer.com/chapter/10.1007/978-3-031-30826-0_6`