TestFul: using a Hybrid Evolutionary Algorithm for Testing Stateful Systems

Matteo Miraz, Pier Luca Lanzi, Luciano Baresi
Politecnico di Milano
{miraz, lanzi, baresi}@elet.polimi.it

Stateful Systems

* Widespread, in several flavours (e.g., Java classes, Components, Web Services, ...)
* The outcome of an operation depends on the input parameters and on the internal state
* Evolving the state towards an interesting configuration is dificult

Random Test Generation

Performs a random sequence of invocations

+ very easy to implement
+ successfully applied on real systems
- unguided => expensive
- the search space is not explored properly

Evolutionary Test Generation

Uses evolutionary techniques, to find the sequence of method invocations that exercises a pre-selected element (statement, branch, path, ...)

+ guided => reach the selected unit efficiently
- related elements are handled independently
- internal state is not valorized properly
- only consider a single objective
  (e.g., branch coverage)

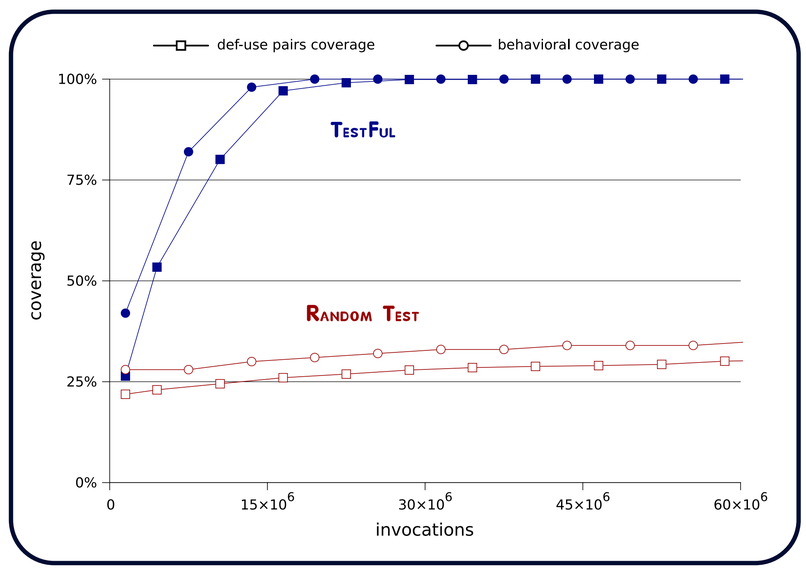
TestFul

Applies Multi-Objective Evolutionary Computation techniques at a higher level
* to avoid unnecessary effort for exercising related features
* to reuse interesting state configurations
* to combine three coverage criteria with a compactness measure to guide the test generation
  - **condition coverage**: to ensure that all conditions in the system are exercised
  - **def-use pairs coverage**: to ensures that the data flow graph is covered properly; to spot dependencies among different parts of a stateful system
  - **behavioral coverage**: to emphasize useful state configurations (NEW!)
  - **Occam's razor**: given two tests with the same coverage criteria, the shorter is preferred

Preliminary Results

Does TestFul tackle the test generation better?
* Is it able to evolve the state fruitfully?
* Is it able to generate good tests?
* Is it faster than similar approaches?

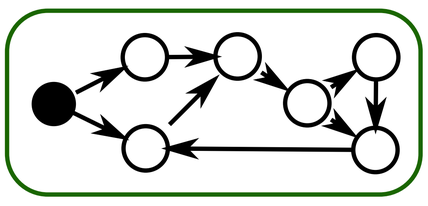
Preliminary results are encouraging
* TestFul achieves better average results, in less time, with less variance



Future Work

Behavioral Coverage

A behavioral model of the system is inferred monitoring tests execution

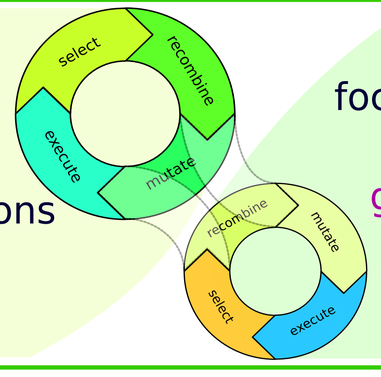


the more the model is complete, the more the test is valuable

Hybridization

Higher loop
focus: whole system

recombine operations to achieve the utmost coverage



Lower loop
focus: uncovered elem.

generate an operation to reach an uncovered element
(e.g. a new condition)