Università
della
Svizzera
italiana

Python test generator

Paolo Tonella (Software Institute, Università della Svizzera italiana, Lugano, Switzerland)

Goal of the project

Write a search based automated test generator for Python. *The generator shall maximize condition coverage of the functions under test and will be compared against a random fuzzer used as baseline.*

1. Write an instrumentation script that transforms the Python code under test to enable computation of the coverage fitness function
2. Develop a fuzzer that generates new test cases randomly or by mutating/crossing over previously created tests
3. **Use the library Deap to define a genetic algorithm that evolves test case inputs so as to maximize condition coverage**
4. Use the tool MutPy to inject artificial faults (mutations) into the benchmark functions under test and evaluate the fault detection capability of the genetic algorithm, considering the random fuzzer as baseline

Deap’s configuration

- **Example of Deap’s configuration**: sb_cgi_decode.py
- **Fitness function**:
    - sum of normalised branch distances computed only for yet uncovered branches (i.e., branches not in the archive of solutions); normalisation of branch distance d is equal to: d / (1 + d)
    - fitness to be minimized (weights = (-1.0,))
    - overall set of branches to be covered is computed during instrumentation
- **Individual**:
    - subclass of list
    - three types of individuals, corresponding to **three types of signatures**:
        - **1 to 3 int parameters**: repeat of int, with initialisation, crossover and mutate defined for list[int] individuals
        - **1 to 3 str parameters**: repeat of str, with initialisation, crossover and mutate defined for list[str] individuals
        - **a pair (str, int) of parameters**: list of one pair (str, int), with initialisation, crossover and mutate defined for individuals of type [(str, int)]
- **Test execution**:
    - as with the Fuzzer, after loading the instrumented file, call the function under test using: `globals()[‘f_instrumented’](270, 966)`
    - upon execution, collect inputs and output into a dictionary e.g. `out[in] = 966`
- **Test generation**:
    - Only test cases that increase coverage are kept in the archive and are reported as test case outputs
    - In the generated tests, the original function (e.g., `f)`, not the instrumented one (e.g., `f_instrumented`) is called

Python test generator

Paolo Tonella
Software Institute, Università della Svizzera italiana, Lugano, Switzerland