DbCheck

`DbCheck` is a software package that auto-generates test cases against a method's pre-conditions, and run the method under test to verify its post-conditions. Tests are auto-generated according to a `json` file which specifies the pre- and post-conditions in a type specific way, so that some simple bounds of parameters could be made use of during the generation of argument lists.

Motivation

Test auto-generation has been a great tool in software industry for developers to find bugs and verify software design goals. It largely help software developers to automate the process of *software contract* verifications, where the pre- and post-conditions of a software module is stated, tested and verified.

The problem with most such test auto-generation tools is that the parameter space size grows exponentially with the size of the parameter lists, and also the branching statements that depends on the parameter values. There has been a lot of efforts put into the area of space reduction in literature, and another usual approach is for the users to supply their own data generators equipped with prior knowledge of the expected domain of parameters.

How it works

This tool works by reading a property file that specifies the argument name, types and pre- and post-conditions for a method under test. Unlike many other tools, the pre-conditions are parsed into AST and the tool tries to read and deduce the possible sub-domain of parameters, so that the argument generation can be more efficient. It can also be used to verify false negative cases for the method under test.

As a feedback, the tool will also trace the statement and code lines under test and determine the strength / quality of test data generated.

This tool uses python's built-in libraries including `ast` and `trace`, and supports common built-in types.

How to run

Call

```
python dbcheck.py choose_props.json
```

where the single argument is the `json` property file that defines the properties, i.e. pre-conditions and post-conditions for a certain module and method.

Calling the program will also generate a trace file for that specific module in the current directory, highlighting the test coverage of current run.

Related works

The work is inspired by Quickcheck, which was an automated testing framework written in Haskell. The tool has inspired a couple dozens of other language ports which all follow the similar idea of type-specific test generation.

Automated Whitebox Fuzz Testing by Godefroid et. al. is a related paper in 2008 that described how they start from a sample in the argument space and increase the test coverage by systematically changing some part of the parameters in order to hit different code logic branches, etc.