The Project Part 2

CS569

Chao Zhang

Prof Groce Alex

1. Introduction

   For the part 2 of this project, I focus on the basic of this project, as the request, I need to implement a novel test generation algorithm using the TSTL API. It should include the timeout, seed, depth, width, faults, coverage and running. So this part will finish the basic function and made some improve. In the

2. Implementation

   Like the article *Adaptive Random Testing* said, the adaptive random testing is based on the intuition. But the biggest problem is how to implement this algorithm with the TSTL API. The most important part of this in the F-measure. In this part, I didn't fully implement the Adaptive random testing because it is too hard to be finished. So I did some BFS in this part of the project, but I will still keep trying on the implement of the ART. The first step is to implement the basic function as mentioned in requirement which are those seven arguments.

   ```
   def parse_args():
       parser = argparse.ArgumentParser()
       parser.add_argument('timeout', type=int, default=30)
       parser.add_argument('seed', type=int, default=None)
       parser.add_argument('depth', type=int, default=100)
       parser.add_argument('width', type=int, default=1)
       parser.add_argument('faults', type=int, default=0)
       parser.add_argument('coverage', type=int, default=1)
       parser.add_argument('running', type=int, default=1)
       parsed_args = parser.parse_args(sys.argv[1:])
       return (parsed_args, parser)
   ```

   The argparse is the way I used to implement those seven arguments from the command lines. So the user can use the command line to make the specific limit to each of them. I also have random function to generate the random number in the project.

   ```
   print bugs,"FAILED"
   print "TOTAL ACTIONS",actCount
   print "TOTAL RUNTIME",time.time()-start
   ```

   It also print the number of bugs founded and the number of action had been take.

3. Future Plan

   This part did not do much on the project, but in the future, I will try to improve the rest part of the tester, like the depth, I may make the tester work like it can get the most depth time and do some more about that. In additional, I will still try to work on the ART algorithm because that is my plan in the beginning. For that, I may try to implement the algorithm 2 shown in the article *Adaptive Random Testing.*

   ```
   Algorithm 2:
   initial test data := randomly generate a test data from the input domain;
   selected set := { initial test data };
   counter := 1;
   total number of candidates := 10;
   ```

use initial test data to test the program;
if (program output is incorrect) then
reveal failure := true;
else
reveal failure := false;
end if
while (not reveal failure) do
candidate set := {};
test data := Select The Best Test Data(selected set, candidate set, total number of candidates);
use test data to test the program;
if (program output is incorrect) then reveal failure := true;
else selected set := selected set + { test data };
counter := counter + *1;*
end if
end while
But this is very hard so all I can do is try to work on it.

4. Reference

I. M. T.Y. Chen, H. Leung. Adaptive random testing, 2004

Discussed with Zhou Zheng.