TDDD04 Software testing Lab report 2
Jonathan Anderson, jonan099
Medhanie Weldemariam, medwe277

Concurrency testing 1: Deadlock detection

1. How would you have found the error in the program if you had not used JPF?

   With out using JPF we are not able to find any error.

2. How does JPF find the deadlock? Use information from the trace printouts from the program as well as the description of how JPF works to justify your answer.

   JPF uses an VM and listener to detect states looking for deadlocks properties, by changing the schedule in this case.

Concurrency testing 2: Race condition detection

3. How would you have found the error in the program if you had not used JPF?

   By running the software many times hoping to see a different output, and changing the load of the threads by adding wait or more work to a thread.

4. What happens if you make the Updater.run or Pair.update methods **synchronized**? Why?

   Case1: when you synchronize Pair.update

   we get no errors because it will synchronize the variables before use.

   Case2: When you synchronize Updater.run

   we get one error while the two threads has been locked. We also found the two threads to be locked when we synchronize the run method, and also one error found same as the first one which is not synchronized

Symbolic execution 1: Test case generation with exceptions

5. What happens if you change the Triangle.getType method to not throw exceptions on invalid parameters but return TriangleType.NaT instead?

   Case 1: With exception throw
   Here, we got one error and one test case

   Case 2: Without exception throw
   But here, no errors has been detected, and also we have got 25 test cases

6. What test cases are generated if you replace the exception?

   25 test cases has been generated for testing all the possible outputs and different variations of inputs.

7. What happens if you change getType(sym#sym#sym) to getType(con#con#sym)?

   When we change the first two 'sym#' into 'con#' they become concrete values as a result we have got four different test cases and no errors has been detected.

8. What does the notation getType(con#con#sym) mean and what is the effect on the generated test cases? How do you think it could be used when probing the program for execution paths?

   Con means concrete and makes the value fixed for the run so we get fewer possible outcomes but a shorter run time where as,

   Sym means Symbolic and it variates during the run which result in more test cases but a longer run time.

   So if we know what we want to test as part of the program i.e. we don't want to test everything, we could probe the program to only test the part that's wanted to be tested with concrete values.

Symbolic execution 2: Test case generation with coverage analysis

9. How does the SymbolicSequenceListener seem to select test case input values?

   It seems to choose values just one above the boundary of the if clauses.
   For example:
   if (usage > 100)
   it tests for the value 101.

10. Do you get 100% statement coverage from the test cases? If not, why do you think that is so?

    No, because the symbolic max value “symbolic.max_double=7.0” for a double is set to 7 at the standard run configuration and to test every part of the program it needs to be greater than 100.

11. Does the coverage indication provided by the CoverageAnalyzer seem correct to you? Justify your answer.

    The coverage seems correct since it has covered ten out of the programs fifteen lines of code, but the whole program has not been tested. So thats why everything isn't tested.

Last exercise: your own code

12. Can you perform symbolic execution with arguments of any datatype?

    We are not able to perform symbolic execution for any of datatypes other than integers.

13. How long does it take for JPF to find a possible race condition or deadlock with a more complicated setup than the examples above?

    A fair bit longer for our small test program, so a larger program it will be much more time consuming to test.