272 – Homework Assignment 3

Fall 2018

Due: Thursday, December 6th, 11:00AM in class

Do not discuss the problems with anyone other than the instructor.

1. Assume that a program crashes with the input string "`(abcde)f`". Also assume that the program crashes with any input string that contains two matching parentheses and it does not crash with any other input. Show an execution of the delta debugging algorithm in this case (show the sequence of inputs that delta debugging algorithm will generate and the classification of each input). Assume that the minimal change is adding one character to the input string, and start with an initial partition where each set adds half of the failure inducing input string given above.

2. (a) Assume that there are two variables x and y in a program. Given the following data trace for (x, y):
(5, 1), (3, 0), (7, 2), (9, 3)
is it possible to infer an invariant of the following form: $x = ay + b$? Give the values for the constants a and b if your answer is yes.

(b) Consider an invariant of the form $x = a$ where x is a variable and a is a constant. If we assume that the initial value of x is selected from the interval $[1, 10]$ with uniform distribution, what are the chances of above invariant holding for a sequence of length four? If the user defined confidence level is 0.01 (i.e., 1%,) will Daikon report such a property as an invariant with a sequence length of four?

3. Consider a channel class with four methods: *open*, *send*, *receive*, *close*, and *status*. It is possible to send to or receive from a channel and check its status only after the channel is opened and before it is closed. A channel can be closed only after it is opened. A channel can be reopened only after it is closed. The *status* method is a query method.

(a) Draw the interface machine for this class using the interface machine model from the paper by Whaley et al.

(b) Assume that following constraint is added to the above interface specification: At least two messages have to be sent to a channel before it is closed. Can this be expressed using the state machine model used by Whaley et al.? Draw a state machine for this modified interface specification using the interface machine model from the paper by Shoham et al.

4.

(a) Using Linear Temporal Logic (LTL) write following properties for calls to a lock class with *acquire*, *release*, and *poll* methods. methods. 1) Immediately after each call to *acquire*, *release* must be called. 2) Once *acquire* method is called, eventually *release* method must be called. 3) Once *poll* method is called, only *poll* method must be called until *acquire* method is called. 4) *acquire* method cannot be called twice in a row.

(b) Assume that a reentrant lock class has an *acquire*, and a *release* method. The lock class allows arbitrary nesting of *acquire* and *release* calls as long as there is a corresponding prior *acquire* call

for each *release* call during the execution, and the number of calls to *acquire* and *release* methods match at the end of the execution. Write a context free pattern for this specification. Show the derivation of the following call sequence from the grammar you specified: *acquire, acquire, release, acquire, release, release*

(c) According to the *total matching* rule from the paper by Meredith et al., when is a trace a valid trace? According to the *suffix matching* rule from the same paper, when is a trace a valid trace?

5.

(a) Consider the following program segment:

```
int test1(int x, int y) {
  int z;
  if (x > y )
    z = x;
  else
    z = y;

  if (z > 0)
    return 1;
  else
    return 0;
}
```

Show how classic symbolic execution starting from the symbolic state $x = m, y = n$ (without using any concrete values) would compute the path conditions for this program. Show the symbolic state and the generated constraints.

(b) Consider the following program segment:

```
int test2(int x, int y) {
  int z;
  if (x > 100) {
    if (x + y = 20)
      z = 1;
  } else
    z = 0;

  if (z = 1) {
    if (y > 0)
      z = 2;
    else if (-y > 200)
      z = 3;
  }

  return z;
}
```

Show how concolic execution starting from the concrete state $x = 5, y = 8$ would compute the path conditions for this program. Show the concrete state, symbolic state and the generated constraints.