# Static Analysis for Software Engineering (ECE750-T05): Final[*]

July 31, 2012

This final exam has 4 questions of equal weight. Please do not collaborate; if you have a question, you may contact me (only) and I will get back to you. You may talk to other people about which questions you have completed and how long it took you to do each question.

## Question 1:

This question is about "Effective Typestate Verification in the Presence of Aliasing," by Fink et al. We will use the automaton in Figure 1. The following code differs slightly from the code in Figure 2: I've added `s.close()` on line 16.

```
 1  public static void talk(Socket s) { s.getOutputStream(); }
 2  public static Socket createSockets() {
 3    Collection result = new LinkedList();
 4    for (int i = 0; i < 5; i++) {
 5      result.add(new Socket());
 6    }
 7    return result;
 8  }
 9
10  public static void example() throws IOException {
11    Collection sockets = createSockets();
12    for (Iterator it = sockets.iterator(); it.hasNext(); ) {
13      Socket s = (Socket) it.next();
14      s.connect(...);
15      talk(s);
16      s.close();
17    }
18  }
```

Prove or disprove: the Integrated Verifier analysis in the paper will identify the call to s.close() as a potential point of failure.

[*]version 1, July 23

## Question 2:

Recall the paper by Kulkarni et al on Galois, "Optimistic Parallelism requires Abstractions". Figure 9 of that paper presents the interface for their Set class. In this question, we will explore graphs in Galois. **Please do not refer to the Graph class in the Galois implementation. You may look at any other part of the implementation.** This question has two parts. Part (a) counts for 2/3rd of the mark for this question, while part (b) counts for 1/3rd.

### part (a)

Consider the following code, which uses a Galois graph implementation.

```
struct Node {
  ... // Definition of node data
};

typedef Galois::Graph::FirstGraph<Node,int,true> Graph;

// Create graph
Graph g;
Node n1, n2;
Graph::GraphNode a, b;
a = g.createNode(n1);
g.addNode(a);
b = g.createNode(n2);
g.addNode(b);
g.addEdge(a, b, 5);

// Traverse graph
for (Graph::active_iterator i=g.active_begin(), iend=g.active_end();
     i != iend; ++i) {
  Graph::GraphNode src = *i;
  for (Graph::neighbor_iterator j = g.neighbor_begin(src),
                                jend = g.neighbor_end(src);
       j != jend; ++j) {
    Graph::GraphNode dst = *j;
    int edgeData = g.getEdgeData(src, dst);
    assert(edgeData == 5);
  }
}
```

Write down a Galois interface for a Graph class which could be linked against the above code. Include any other methods that you think are important. Provide a pseudocode implementation of the class and use it to argue for your choices of calls, commutes and inverse declarations in your interface.

### part (b)

Observe that Figure 9 does not include any provision for iteration. Explain how Galois supports iteration on this Set declaration. (You may consult existing Galois code to answer this question; just don't look at the Graph implementation.)

## Question 3:

This question is about the paper by Yu et al, "Patching Vulnerabilities with Sanitization Synthesis". I've included one of the benchmark extracts below[1]. Show me the operation of the vulnerability analysis (Section 4) and the vulnerability signature generation algorithm (Section 5) on the following example. I'd like to see the dependency graph, sink nodes, and the attack pattern that you're using. Write out the PRE and POST sets at the fixed point. Include any explanations that you think will help me understand your solutions.

```
$cp9 = "Show this entry to everybody";
$cp10 = "Show this entry only to registered users";

$sql = "SELECT id,title,date,private FROM blog ORDER BY id DESC";
$result = mysql_query($sql,$connection);
while ($row = mysql_fetch_assoc($result)) {
   $row["title"] = addslashes($row["title"]);
   $row["date"] = addslashes($row["date"]);
   $row["id"] = addslashes($row["id"]);

   print("<tr>\n<td class=\"text\">{$row["date"]}</td>\n".
         "<td class=\"text\">{$row["title"]}</td>\n<td align=\"center\">\n");
   if($row["private"]) {
       print("<a href=\"priv.php?id={$row["id"]}\"><img alt=\"{    $cp9}\" ".
             "src=\"images/admin/private.gif\" border=\"0\"></a>");
   }
   else {
       print("<a href=\"priv.php?id={$row["id"]}\"><img alt=\"{    $cp10}\" ".
             "src=\"images/admin/public.gif\" border=\"0\"></a>");
   }
   print("</td>");
}
```

## Question 4 (Programming Question):

Develop a JavaCOP type system extension to prevent calls to `getOutputStream()` after `close` on a Socket.

```
public static void example() throws IOException {
  Socket s = new Socket();
  s.close();
  s.getOutputStream(); // not allowed!
}
```

You'll need to implement a flow analysis. You do not have to handle aliasing or interprocedural control flow. (Document your automaton and the assumptions that you're making about the code.) Submit code and test cases to me by email.

[1]Benchmark extracts at `http://www.cs.ucsb.edu/~vlab/stranger/benchmarks.tar.gz`.