Department of Computer Science
Graduate Comprehensive Exam
in
Software Engineering
Fall 2019

- **Answer the questions on the paper supplied.**
- **Answer question 1, and answer question 2, and answer question 3. You should answer a total of three questions.**
- **Start each question on a new page. Write on only one side of the paper.**
- **Write your SIX-DIGIT Texas State ID in the top right corner of each page of your answer. Do NOT put your name anywhere on the answers.**
- **Put the number of the question being answered in the top left corner of each answer page.**
- **If the answer to a question is written on more than one page, number the pages consecutively.**

1. CS5391 Survey of Software Engineering
   {*from Dr. Chen*}

   Explain the following software process models and point out their pros and cons:

   (a) Formal method development model

   (b) Agile development model

2. CS5392 Formal Methods in Software Engineering
   {*from Dr. Yang*}
   1. Prove the validity of $p \to q \wedge r \vdash (p \to q) \wedge (p \to r)$.

$$\frac{\Phi \quad \Psi}{\Phi \wedge \Psi}\ \wedge\mathrm{i} \qquad \frac{\neg\neg\Phi}{\Phi}\ \neg\neg\mathrm{e} \qquad \frac{\boxed{\begin{array}{c}\Phi\\ \vdots\\ \Psi\end{array}}}{\Phi \to \Psi}\ \to\mathrm{i}$$

$$\frac{\Phi \wedge \Psi}{\Phi}\ \wedge\mathrm{e1} \qquad \frac{\Phi}{\neg\neg\Phi}\ \neg\neg\mathrm{i}$$

$$\frac{\Phi \wedge \Psi}{\Psi}\ \wedge\mathrm{e2} \qquad \frac{\Phi \quad \Phi \to \Psi}{\Psi}\ \to\mathrm{e}$$

$$\frac{\Phi \quad \Phi \to \Psi}{\Psi}\ \to\mathrm{e} \qquad \frac{\Phi \to \Psi \quad \neg\Psi}{\neg\Phi}\ \mathrm{MT}$$

2. Conduct symbolic evaluation of path 1,2,4,5,7,8 for the code fragment below. Explain if this path is feasible. Provide the intermediate and final results for path condition, domain and computation.

```
1)  input x,z;
2)  if x > z then
3)    y := 3;
    else
4)    x := z + 2;
    end if;
5)  if x > z + 1 then
6)    w := 3;
    else
7)    w := 2
    end if;
8)  write w;
```

3. Find appropriate predicates and their specification to translate the following into predicate logic:
(a) Every child is younger than its father.
(b) No animal is both a bird and a dog.
(c) Not all birds can fly.

3. CS5393 Software Quality
{ *from Dr. Yang*}

1. Recall a *du-path* with respect to a variable v is a simple path that is *def-clear* with respect to v from node n_i for which v is in $def(n_i)$ to a node n_j for which v is in $use(n_j)$.

Consider the following code fragment and answer the questions that follow using the node numbers given as comments:

```
y = m;              // node 1
if(x>0){            // node 2
      w++;          // node 3
}
else{
      w = 2*w;      // node 4
}
while(y<10){        // node 5
      x++;          // node 6
      y++;          // node 7
}
z = w + x;          // node 8
```

node 8 is a final node.

(b) Which nodes have defs for variable *w*? Which nodes have uses for variable *w*?

(c) Give a set of paths that would be needed to achieve *All-du-paths Coverage* with respect to variable *y*. Show the def-use relations and indicate which paths cover which relationships.

2. Consider the method *Min* and its 6 mutants in Figure 1. For each of mutants Δ1, Δ3, Δ5, provide reachability condition, infection condition, propagation condition, test case values to strongly kill it, and test case values to weakly kill it.

Original Method	With Embedded Mutants

int Min (int A, int B)
{
int minVal;
minVal = A;
if (B < A)
{
minVal = B;
}
return (minVal);
} // end Min
	int Min (int A, int B)
	{
	int minVal;
	minVal = A;
	Δ1 minVal = **B**;
	if (B < A)
	Δ2 if (B > A)
	Δ3 if (B < **minVal**)
	{
	minVal = B;
	Δ4 **Bomb()**;
	Δ5 minVal = **A**;
	Δ6 minVal = **failOnZero (B)**;
	}
	return (minVal);
	} // end Min

Figure 1: Mutants