Prof. Dr. P. Liggesmeyer
M.Sc. Z. Guo

University of Kaiserslautern
Dept. of Computer Sciences
AG Software Engineering: Dependability

Software Quality Assurance
(WS 08/09)

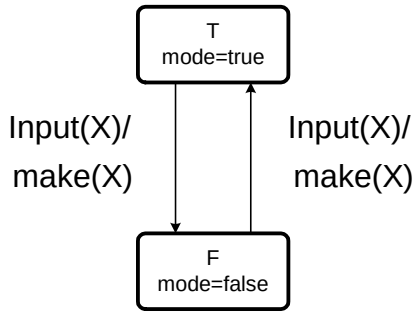
Problem Set 7

Due Thursday, February 12th, 2009

Problem 1: Dataflow Anomaly Analysis

Consider the following Java implementation of the class `Switch`. The class is a module of a technical application that is classified as safety-critical.

A switch provides a return value that depends on the internal state and the given input value. The return value is true if input and internal state coincide, else it is false. Every usage modifies the internal state of the switch.




```
public class Switch {
  private boolean mode;
  public Switch() {
    init();
  }
  private void init() {
    mode=true;
  }
  public boolean toggle(boolean mode) {
    boolean result;
    if((this.mode&&mode)||(!this.mode&&!mode))
      result=true;
    else
      result=false;
    if (this.mode)
      this.mode=!mode;
    else
      mode=mode;
    return result;
  }
}
```

Perform a dataflow anomaly analysis for the operation `toggle`.

Problem 2: Dataflow Anomaly Analysis

A software company develops software packages for commercial animal housing. A particular function, which is implemented in the C programming language, computes the daily amount of feed for different animal species depending on their individual weight.

Until now, this function has only been part of a software package for farms and has worked without failures for years. Recently, it has also been included in a software package for zoos, and now it produces wrong output in some cases. By performing a dataflow analysis, the faults shall be revealed.

```
/* Own data type for enumeration of animal species */
typedef enum {COW, HORSE, PIG, ELEPHANT} Animal_A;

/* Function for determining the daily amount of feed depending
 * on the animal species and the individual weight
 */
float feedamount(Animal_A species, float weight)
{

   float amount, factor;

   switch (species)
   {
      case COW:
      {
         factor = 0.05;
         break;
      }
      case HORSE:
      {
         factor = 0.1;
         break;
      }
      case PIG:
      {
         factor = 0.02;
         break;
      }
   } // end switch

   amount = factor * weight;

return amount;

} // end feedamount
```

a) Which mistakes were made and how could the consequences have been avoided?

b) Perform a dataflow anomaly analysis for the operation `feedamount`.

Problem 3: Dataflow Anomaly Analysis

Consider the following Java implementation of the operation ALL_POSITIVE, which checks whether all elements of a one-dimensional array are positive. As parameters, the field and its length are given.

```
boolean ALL_POSITIVE(int[] array,int len) {
  boolean result;
  int i,tmp;
  i=0;
  result=true;
  while (i<len&&result) {
    tmp=array[i];
    if (tmp<=0)
      result=false;
    i++;
  }
  return result;
}
```

Perform a dataflow anomaly analysis for the operation ALL_POSITIVE.

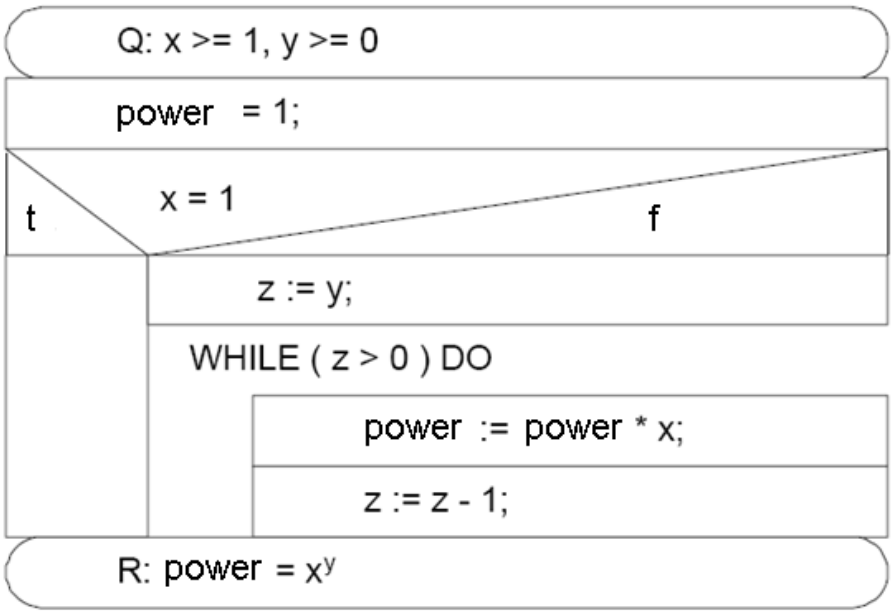
Problem 4: Hoare Calculus and Verification

$P_e^v\{v = e;\}P$ is defined as the weakest precondition. Give a weakest precondition for the following expressions:

a) {z = x + y;} z = 10
b) {x = f(x);} x = u
c) {a = 0;} a = 0 ∧ b ≥ 0

Problem 5: Hoare Calculus and Verification

Consider the following program:
Let x and y be integer variables. The shown operation should compute the power X^Y



a) Construct a loop invariant for verification!
b) Show correctness by completion of the annotations!

Problem 6: Algebraic Specification

A tree is defined as the following:

Functions:

new:	CHAR	→ TREE
create:	TREE x CHAR x TREE	→ TREE
left:	TREE	→ TREE
right:	TREE	→ TREE
value:	TREE	→ CHAR
isLeaf:	TREE	→ BOOLEAN
node:	TREE	→ INTEGER

Axioms:

∀ b, b_1, b_2 in TREE, ∀ c in CHAR		
1. value (new (c))	=	c
2. value (create (b1, c, b2))	=	c
3. isLeaf (new (c))	=	true
4. left (b)	=	if b = create (b1, c, b2) then b1 else error
5. right (b)	=	if b = create (b1, c, b2) then b2 else error
6. node (b)	=	if isLeaf (b) then 0 else node (left (b)) + node (right (b)) + 1

a) Simplify the following terms by using the axioms given above. State out the used axioms in each step.

   a. `value (right (create (create (new ('a'), 'b', new ('c')), 'd', left (create (new ('e'), 'f', new ('g'))))))`
   b. `isLeaf (left (create (new ('a'), 'b', new ('c'))))`
   c. `node (create ( create ( new ('a'), 'b', new ('c')), 'd', new ('e')))`

b) Visualize the following term: `create (create (new ('f'), 'b', right ( create ( new ('c'),'g', create (new ('d'), 'i', new ('e'))))), 'h', new ('a'))`