Quiz 2, CS6013

Maximum marks = 60, Time: 60 min

22-Mar-2024

Read all the instructions and questions carefully. You can make any reasonable assumptions that you think are necessary; but state them clearly. There are total four questions, totalling 60 marks. You will need approximately 10 minutes for answering a 10 marks question (plan your time accordingly). For questions with sub-parts, the division for the sub-parts are given in square brackets. Start each question on a new page. The maximum space allowed for any question is specified at the beginning of the question.

1. [15 marks, 2 pages] **Flow Insensitive and Sensitive Analysis.**
   Consider a sub-graph of a CFG, with three basic blocks B1, B2, and B3, such that B3 has two predecessors B1 and B2. Using flow-sensitive constant propagation as an example, state how the IN map for B3 is calculated based on the IN maps of B1 and B2. [5 marks]
   In the case of flow-insensitive analysis, (a) give the flow function used to compute the variable-constant map for a statement of the form `x = y + z`. [5 marks] (b) state how is the IN map for any block computed correctly, without having any knowledge of predecessor and successor information of the basic-blocks. [5 marks] For convenience, you may assume that each basic-block consists of exactly one statement.

2. [15 marks, 2 pages] **Simple Constant Propagation.**
   Consider the following C code.

```
L1: void foo (int a) {
L2:   int x, y, z;
L3:   x = 1;
L4:   y = a;
L5:   z = 1;
L6:   p = 1;
L7:   while (x < y) {
L8:      p = p + 1;
L9:      z = z * x;
L10:     x = x + 1;
L11:     p = p - 1; }
L13:  printf ("%d%d%d%d",x, y, z, p); }
```

   For above code, use the simple constant propagation algorithm to show the variable-constant map at each program point, after each step. Clearly show the statement processed and the resulting variable-constant map. [10 marks]

   Write a C code (without the use of any if-statement or if-else statement) that shows that constant propagation is not distributive. [5 marks]

3. [15 marks, 2 pages] **Control Tree based Data Flow Analysis.**
Consider the following C code.

```
L1: void foo () {
L2:   int x, p;
L3:   x = 1;
L6:   p = 1;
L7:   while (x < 10) {
L8:      p = p * 1;
L10:     x = x + 1;
L12:  }
L13:  printf ("%d%d",x, p);
L14:}
```

For the above C code, draw the CFG (2 marks), control-tree (3 marks). Further, in the context of constant propagation, write the flow functions for each node in the control tree and equations to compute the IN value for each node [7 marks]. Use these equations to compute the constants at each statement [3 marks].

4. [15 marks, 2 page] **Interprocedural Analysis.**
Consider the following Java code.

```
1. public class A{
2.         public static void main(String[] args){
3.                 B x;
4.                 x = new B();
5.                 x.foo();
6.                 x = new C();
7.                 x.foo();
8.                 if (*) x = new C();
9.                 x.foo();
10.        }
11.        public void foo(){
12.                bar(); }
13.        public void bar(){}
14.}
15.public class B extends A{
16.        public void bar(){}
17.}
18.public class C extends B{
19.        public void bar(){}
20.        public void foo(){}
21.}
```

(a) Draw the call-graph based on CHA. [5 marks]
(b) Give an example C code, where summary based (context-insensitive) interprocedural constant propagation leads to improved precision compared to intraprocedural constant propagation; also show the obtained constants. [5 marks]
(c) Give an example C code (with loops, but no if-statements/if-else statements), where conditional constant propagation leads to improved precision compared to simple constant propagation algorithm; also show the obtained constants. [5 marks]