Assignment 3 Solutions
More dataflow analysis
Due: February 4, 11:00 am

This is a written assignment, every student must hand in his or her homework. Bring your homework to class on February 4th. SCPD students may submit their homework by e-mail via scpd-distribution@lists.stanford.edu or give your homework to the courier. There is also an online-component on Gradiance that must be finished before the deadline.

1. [20 points] Consider a language which has only non-negative integer variables in the range [0,255]. This language has the following types of statements.

**Assignments**
\[ x = k \]
(k is a constant)

**Expressions**
\[ x = a \ op \ b \]
(a is a variable and b is a variable or constant and \( \text{op} \) is either \(+,-,\times,/\))

**Gotos and conditional branches**
(No assignments are done to any variables in conditional branch statements.)

We would like to issue two types of warnings:
WARNING I: Possible overflow
WARNING II: Possible underflow

Example:

\[ y = 16 \]
\[ x = y * 16 \] (Warning I)

\[ y = 5 \]
\[ x = y - 10 \] (Warning II)

**SOLUTION :**

a. [2 points] Explain what each value in the dataflow analysis represents?

Each value for a variable should store the maximum and minimum possible value for that variable.

b. [2 points] What is the domain?

Domain is the set of \((a : min_a, max_a)\)
c. **[2 points]** What is the direction of data flow?

  Forward

d. **[4 points]** What is the transfer function?

  For assignments \( x = k \), we get \((x : k, k)\)
  
  For addition \( x = a + b \) we get \((x : \min_a + \min_b, \max_a + \max_b)\)
  
  Multiplication is similar to addition.
  
  For subtraction \( x = a - b \) we get \((x : \min_a - \max_b, \max_a - \min_b)\)
  
  Division is similar to subtraction.
  
  For overflow/underflow, we set the value to \((x : 0, 255)\) and issue warnings correspondingly.

e. **[2 points]** What is the meet operator?

\[
(a : \min_1, \max_1) \land (a : \min_2, \max_2) = (a : \min(\min_1, \min_2), \max(\max_1, \max_2))
\]

f. **[2 points]** What are the initialization conditions?

  \(\text{OUT}[B] = (x : 255, 0)\) (top element). \(\text{OUT}[\text{Entry}] = (x : 0, 255)\)

  Note that some students initialize to undefined, which is also acceptable if the meet operator on undefined is correct.

g. **[3 points]** How many iterations can it take to converge?

  255-258 are all acceptable.

h. **[3 points]** If the range of our numbers were \([0,4294967295]\) (unsigned long). Is it feasible to wait for convergence? If not, what do you propose to do?

  No, it’s not feasible to wait for convergence. Run for a fixed number of iterations say 100 or 1000, if value keeps increasing still, mark as overflow. If value keeps decreasing mark as underflow. This is an open-ended question. Other reasonable approaches are also acceptable.

**GRADER COMMENTS:**

- Some students keep track of all possible values of each variable. Though this leads to correct answer, it’s too inefficient.

- Some students initialize \(\text{OUT}[B]\) to \([0, 255]\), which can generate unnecessary warnings (Consider a loop).
2. [10 points] Run PRE on the following block of code and draw the output after the final stage. You may add new empty basic blocks wherever necessary.

SOLUTION: 

```
entry

  t = x + y

  t = x + y

  a = t

  \( x = \text{input}() \)

  y = \text{input}()

  t = x + y

  b = t

  c = x + y

exit
```
3. [14 points] You are given the task of optimizing the code given below but you’re only allowed to use the following three optimization techniques:

- PRE (as discussed in class)
- Constant propagation (as discussed in class)
- Copy Propagation (Discussed in Section 9.1.5, )

in any order and multiple times if necessary.

```
entry

# t1 = q+2

# t2 = a+t1

# r=2

# p=t1

# x=t1

# z=t2

# y=t1

# m=t2

# w=a+x

# r=input()

# exit
```

SOLUTION:
a. **[4 points]** What is the order in which you executed them to produce the best optimized code by running a minimum number of analysis?

Constant propagation, PRE, Copy propagation, PRE again.

b. **[10 points]** What is the final optimized program?

**GRADER COMMENTS:**
- Copy propagation is not the same as constant
- You need to run each analysis/phase of PRE till convergence. Just one iteration of each analysis produces wrong results.
- Try to do it with your understanding of the algorithm instead of direct simulation. That way most people wouldn’t have missed the loop invariant code motion.