

### 631 practice exam

This represents far more work that would actually be on the midterm. However, it is representative of the questions that might appear on the midterm (also possibly some questions on OCAML).

This program is to be used in questions 1-3.

```
S0      if (z <= 1) goto L1
S1      x = 1
S2      if (z > 2) goto L2
S3      y = x+1
S4      goto L3
S5 L1   x = 2
S6 L2   z = x - 3
S7      x = 4
S8      z = x+7
S9 L3   print x,y,z
```

1. Control flow analysis
  - (a) What are the basic blocks?
  - (b) Show the control flow graph
  - (c) Number the nodes of the CFG in reverse postorder.
  - (d) Give the dominator tree for this program
2. Control dependence - which block(s) are control-dependent on which block(s)?
3. Dominance frontiers and SSA form
  - (a) For each block, give the dominance frontier and the iterated dominance frontier.
  - (b) Give the (minimal) SSA form of the program.
4. Data flow lattices
  - (a) Remember that we define our  $\sqsubseteq$  operator as  $u \sqsubseteq v \equiv u = u \sqcap v$ . A frame work is monotone if and only if  $u \sqsubseteq v \Rightarrow f(u) \sqsubseteq f(v)$ .  
Prove or disprove that even if a framework is not distributive (i.e.,  $f(u \sqcap v) = f(u) \sqcap f(v)$ ), monotonicity guarantees that  $f(u \sqcap v) \sqsubseteq f(u) \sqcap f(v)$ .
  - (b) Informally/intuitively, describe what would be true of a system that is not monotone.
5. Number the nodes on the graph according to a preorder depth first traversal. Label edges as t/f/b/c (tree/forward/back/cross) edges. Draw the dominator tree for this graph.

6. We want to be able to determine if a pointer might be null. We want to warn the user and refuse to compile when we find a statement that is a definite error, and omit run-time null pointer checks whenever possible.

Assume that doing a reference through a null pointer throws an exception that is not caught within the code we are analyzing.

For our analysis, try to compute accurate information for copy statements (e.g., after `p = q`, whatever we had previously known about `q` we now know about `p`) and pointer uses (e.g., after `p = q.x` we know that `q` is non-null and don't know anything about `p`).

Formulate this problem as one or more data flow problems. There are several different ways to do this; the more accurate your solution the higher your grade.

- (a) What is your lattice (e.g., the type of the values you compute at each point)?
  - (b) Discuss the ways in which your analysis is conservative, how it generate incorrect (but conservative) answers, and why the decisions we might make on it are safe.
  - (c) What is your meet function?
  - (d) Is your solution a forwards or backwards problem?
  - (e) If your framework monotone? Is it distributive? Justify your answers.
  - (f) What value should be used to initialize In/Out values (other than those for Entry/Exit)?
  - (g) Describe how to compute the transfer functions.
  - (h) Briefly describe the changes that would need to be made to take into account conditional tests such as `if p == null goto L5`.
7. Consider the following program:

```
X := read()
Z := read()
```

```

if (...) goto L1
L2:
Y := X*X
write Y
L3:
if (...) goto L4
W := Z*Z
X := X+W
goto L2
L1:
Y := X+X
write Y
goto L3
L4:
Y := X*X
write Y
if (...) goto L5
Y := X+X
L5:
write Y

```

- (a) Give the control flow graph for this program.
  - (b) Give the dominator tree for this program.
  - (c) Give the control dependence relation (which block is control dependence on which block) for this program.
  - (d) Give the iterated dominance frontier for each block.
8. Put the following program in SSA form (you may draw a control flow graph to illustrate your solution):

```

x := 0;
do {
  x := x + 1;
  z := x;
  y := 0;
  if (...) {
    y := 1;
  }
  w := y + z;
} while (...);
print(x, y, z, w);

```

9. Type inference through data flow analysis. We have a dynamically typed language with a simple type system:

$$\text{Type} = \text{INT} \text{ --- } \text{FLOAT} \text{ --- } \&\text{Type} \text{ --- } \text{unknown}$$

The third possibility indicates a pointer type (e.g., `&int` (pointer to an int) or `&&float` (pointer to a pointer to a float)).

Although this system does not require static types for variables, we want to deduce static types when possible so that we will not have to generate run-time type-checks when possible.

Assume that the language allows simple arithmetic and comparison operators on ints and floats (it suffices to discuss just  $+$  in your answer), dereferencing a pointer and taking the address of a variable. For arithmetic operations, ints are converted to floats when combined with floats. Dereferencing an int or pointer arithmetic generates a run-time exception.

Define and describe an appropriate data flow analysis framework to compute static types when appropriate. Discuss how your framework can be inaccurate. Do you compute a meets-over-all-paths answer?

I haven't spelled out all the details. Treat this as though you were being asked to come up with an example for a textbook. Fill in details that illuminate the problem, and omit those that do not.

10. Exercise 1.3 from handout.

11. Exercise 1.4 from handout.

12. Consider an abstract interpretation that abstracts integers as either even or odd. In particular, the environment of a program, rather than mapping variables to specific integer values, maps each variable to either odd or even.

- Give the abstraction ( $\alpha$ ) and concretization ( $\gamma$ ) function. Show that  $(\alpha, \gamma)$  is a Galois connection (e.g., that  $\alpha(X) \subseteq Y \Rightarrow X \subseteq \gamma(Y)$ )
- Give the abstract meaning of  $+$ ,  $*$  and  $/$ .
- Give the abstract value of the environment at the end of the following code:

```
x := 5; y := 4 z := 21; while (...) x++; y--; z := x+y;
```

Why does the analysis not determine that  $z$  is always odd?

- Specify a more precise abstraction that allows derivation of the fact that  $z$  is odd. In particular, you should allow abstract states such as "either  $x$  is odd and  $y$  is even, or  $x$  is even and  $y$  is odd". Give the revised abstraction ( $\alpha$ ) and concretization ( $\gamma$ ) function. Show that  $(\alpha, \gamma)$  is a Galois connection (e.g., that  $\alpha(X) \subseteq Y \Rightarrow X \subseteq \gamma(Y)$ ).