

# CS 671 Graduate Seminar

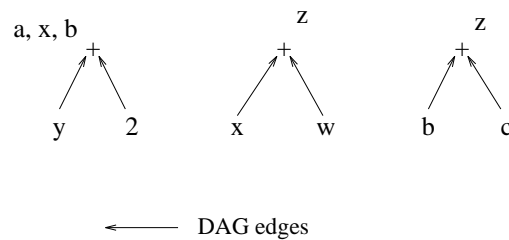
## Challenge Problem 1

### Local Common Subexpression Elimination

In lecture 2, we talked about the DAG construction algorithm for local common subexpression elimination.

- Show the DAG for the following piece of code

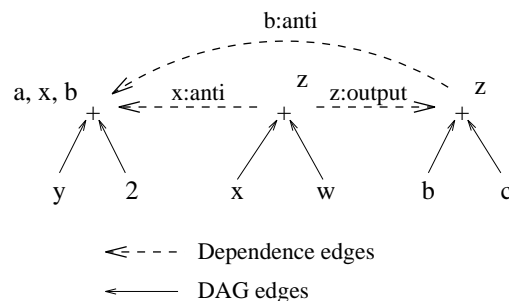
1.  $a = y + 2$
2.  $z = x + w$
3.  $x = y + 2$
4.  $z = b + c$
5.  $b = y + 2$



Use basic algorithm as discussed in lecture 2, and in ASU p. 548. The problem here is that the “naive” code generation does not enforce any ordering on the independent expression DAGS.

- Describe an algorithm that generates *correct* code from such a DAG.

Here is one possible approach: Introduce dependence edges between internal operation nodes. Code generation process uses rPOSTORDER to generate code for the subtrees. Depth-first search starts at nodes that do not have any incoming dependence edges.



- Show the code generated for the example code.

```
z = x + w // middle DAG
z = b + c // right DAG
a = y + 2 // left DAG
x = a
b = a
```

Of course, dead code elimination should be applied here to get rid of the first assignment to variable z.