Dataflow Analysis
Program optimizations

• So far we have talked about different kinds of optimizations
  • Peephole optimizations
  • Local common sub-expression elimination
• What about *global optimizations*
  • Optimizations across multiple basic blocks (usually a whole procedure)
    • Conditionals and loops
Useful optimizations

• Common subexpression elimination (global)
  • Need to know which expressions are available at a point

• Dead code elimination
  • Need to know if the effects of a piece of code are never needed, or if code cannot be reached

• Constant folding
  • Need to know if variable has a constant value

• So how do we get this information?
Dataflow analysis

• Framework for doing compiler analyses to drive optimization
• Works across basic blocks
• Examples
  • Constant propagation: determine which variables are constant
  • Liveness analysis: determine which variables are live
  • Available expressions: determine which expressions have valid computed values
  • Reaching definitions: determine which definitions could “reach” a use
Example: constant propagation

- **Goal:** determine when variables take on constant values

- **Why?** Can enable many optimizations

- **Constant folding**
  
  ```
  x = 1;
  y = x + 2;
  if (x > z) then y = 5
  ...
  
  x = 1;
  y = 3;
  if (1 > z) then y = 5
  ...
  ```

- **Create dead code**
  
  ```
  x = 1;
  y = x + 2;
  if (y > x) then y = 5
  ...
  
  x = 1;
  y = 3;  //dead code
  if (true) then y = 5  //simplify!
  ...
  ```
How can we find constants?

• Run program and see which variables are constant?
  
  • Problem: variables can be constant with some inputs, not others – need an approach that works for all inputs!

  • Problem: program can run forever (infinite loops?) – need an approach that we know will finish

• Idea: run program symbolically

• Essentially, keep track of whether a variable is constant or not constant (but nothing else)
next: constant propagation