

# CSC2/455 Software Analysis and Improvement

## Available Expressions, Reaching Definitions, Very Busy Expressions

Sreepathi Pai

URCS

February 3, 2020

# Outline

Review

Available Expressions Analysis

The Data Flow Analysis Framework

Reaching Definitions Analysis

Very Busy Expressions (Anticipable expressions)

Postscript

# Outline

Review

Available Expressions Analysis

The Data Flow Analysis Framework

Reaching Definitions Analysis

Very Busy Expressions (Anticipable expressions)

Postscript

# Data flow Analysis so far

- ▶ Live variable analysis
  - ▶ “Is there a read of this variable along any path?”
- ▶ Data flow equations
  - ▶ DEF and USE
- ▶ Iterative data flow analysis
- ▶ A brief look at quality of solutions
  - ▶ Path-based vs Iterative

# Outline

Review

**Available Expressions Analysis**

The Data Flow Analysis Framework

Reaching Definitions Analysis

Very Busy Expressions (Anticipable expressions)

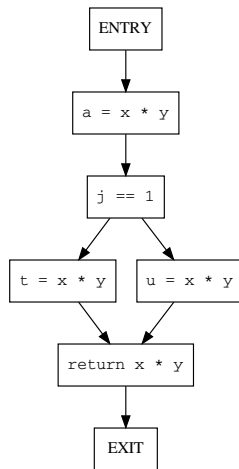
Postscript

# Goals

- ▶ Identify repeated calculations of the same expression (analysis)
- ▶ Remove these repeated calculations (optimization)

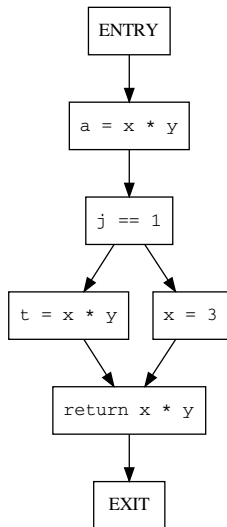
# Example 1

```
a = x * y;  
if(j == 1) {  
    t = x * y;  
} else {  
    u = x * y;  
}  
return x * y;
```



## Example 2

```
a = x * y;  
if(j == 1) {  
    t = x * y;  
} else {  
    x = 3;  
}  
return x * y;
```

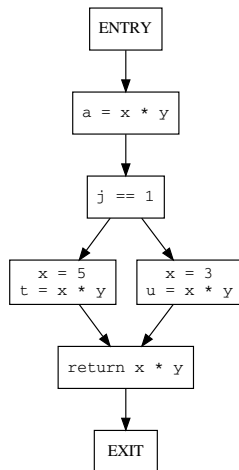




## Example 3

Is  $x * y$  available at the return statement?

```
a = x * y;
if(j == 1) {
  x = 5;
  t = x * y;
} else {
  x = 3;
  u = x * y;
}
return x * y;
```



# Definition

An expression  $e$  is available at a program point  $p$  iff:

- ▶ it has been calculated previously
  - ▶ these can be distinct calculations
- ▶ *all* paths from ENTRY to  $p$  contain the calculation
- ▶ no subexpression of  $e$  has been modified along these paths

Informally: Can we reuse the value computed previously by an expression instead of recalculating it?

# Differences from LIVE

- ▶ What are the similarities from LIVE variable analysis?
- ▶ What are the differences from LIVE variable analysis?

# Dataflow Equations: Basic Block

- ▶  $GEN(n)$ 
  - ▶ a block  $n$  generates these expressions
- ▶  $KILL(n)$ 
  - ▶ a block  $n$  *kills* these expressions

# Dataflow Equations: Incorporating Flow

- ▶ How does information flow for AVAIL?
- ▶ How do we “merge” information from multiple source blocks?

$$AVAIL\_IN(n) = ?$$

# Dataflow Equations: Incorporating Flow

- ▶ Information flows *forwards*
- ▶ Only expressions common to all *all* paths are available
  - ▶ Note: Live variables needed *any* path

# The Dataflow Equation

$$\text{AVAIL\_IN}(n) = \bigcap_{m \in \text{pred}(n)} \text{GEN}(m) \cup (\text{AVAIL\_IN}(m) - \text{KILL}(m))$$

# Initialization

- ▶  $AVAIL\_IN(ENTRY) = \emptyset$
- ▶  $AVAIL\_IN(n) = \{\text{all expressions}\}$



# Outline

Review

Available Expressions Analysis

**The Data Flow Analysis Framework**

Reaching Definitions Analysis

Very Busy Expressions (Anticipable expressions)

Postscript

# Building Blocks

- ▶ How to generate basic block facts
  - ▶ GEN (facts generated by this basic block)
  - ▶ KILL (facts killed by this basic block)
- ▶ How to merge facts across basic blocks
  - ▶ Use a control-flow graph to abstract control flow
  - ▶ Direction: forwards/backwards
  - ▶ Path requirement: “may” (any) or “must” (all)
  - ▶ Determines whether you obtain facts from predecessors or successors, and whether you use union or intersection
  - ▶ End goal: a data flow equation for each basic block that needs to be solved
- ▶ How to determine initial values
  - ▶ Usually analysis specific
- ▶ How to solve: iteration until fix point is reached
  - ▶ Other methods possible (later in course)

# Outline

Review

Available Expressions Analysis

The Data Flow Analysis Framework

**Reaching Definitions Analysis**

Very Busy Expressions (Anticipable expressions)

Postscript

## Example

(Note: The #n notation identifies a definition)

```
y#0 = 0;
x#0 = 1995;

if(y == 0) {
    x#1 = 2000;
    y#1 = 50;
} else {
    x#2 = 3000;
}
a#0 = x + y;
```

Here, the value of  $y$  in the last line comes from  $y\#0$  or  $y\#1$ , while the value of  $x$  comes from  $x\#1$  or  $x\#2$ .

# Definition

A definition of variable  $x$  at a point  $p$  reaches the use of  $x$  at a point  $q$  iff: there is a path from  $p$  to  $q$  along which  $x$  is not redefined.

# Setting up the framework

- ▶ What is the domain of the set of dataflow facts?
- ▶ What are the basic block facts (i.e. GEN and KILL)?
- ▶ How does information flow?
- ▶ How is information merged?

# Setting up the framework

- ▶ What is the domain of the set of dataflow facts?
  - ▶ The definitions
- ▶ What are the basic block facts (i.e. GEN and KILL)?
  - ▶ GEN is the set of definitions
  - ▶ KILL is the set of definitions
- ▶ How does information flow?
  - ▶ From a block's predecessors to it
- ▶ How is information merged?
  - ▶ Union

# Outline

Review

Available Expressions Analysis

The Data Flow Analysis Framework

Reaching Definitions Analysis

**Very Busy Expressions (Anticipable expressions)**

Postscript



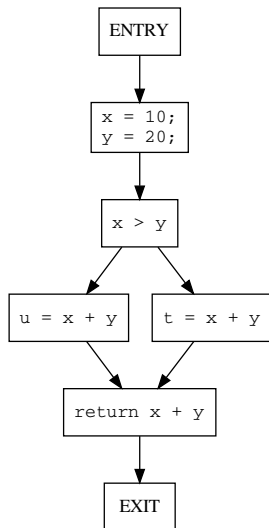
## Example

```
x = 10;
y = 20;

if(x > y) {
    u = x + y;
} else {
    t = x + y;
}

return x + y;
```

- ▶ Can you move  $x + y$  out of the if to (say) reduce code size?
- ▶ Where would you move it?
- ▶ What would AVAIL do here?



# Definition

An expression  $e$  is *very busy* at the end of a basic block  $b$  iff:

- ▶  $e$  is calculated on every path out of  $b$ ,
- ▶  $e$  could be calculated in  $b$  without changing the results of calculating  $e$  in succeeding blocks

Note:  $e$  may not be currently calculated in  $b$ . Informally, we're trying to compute an expression earlier than it appears in the code.

# Setting up the framework

- ▶ What is the domain of the set of dataflow facts?
- ▶ What are the basic block facts (i.e. GEN and KILL)?
- ▶ How does information flow?
- ▶ How is information merged?

# Setting up the framework

- ▶ What is the domain of the set of dataflow facts?
  - ▶ Expressions
- ▶ What are the basic block facts (i.e. GEN and KILL)?
  - ▶ GEN is the expressions evaluated in this block
  - ▶ KILL is the expressions redefined by this block
- ▶ How does information flow?
  - ▶ From the block's successors to it
- ▶ How is information merged?
  - ▶ Intersection

# Outline

Review

Available Expressions Analysis

The Data Flow Analysis Framework

Reaching Definitions Analysis

Very Busy Expressions (Anticipable expressions)

Postscript

# References

- ▶ Chapter 9 of Cooper and Turczon
  - ▶ Section 9.2.4
- ▶ Also recommended (more detailed coverage here):
  - ▶ Aho, Lam, Sethi and Ullman, Chapter 9, Section 9.2.6 (Available Expressions)
  - ▶ Aho, Lam, Sethi and Ullman, Chapter 9, Section 9.2.4 (Reaching Definitions)