EXERCISE #5

STATIC ANALYSIS REVIEW

Write your name and answer the following on a piece of paper

- Show the instruction flowchart of the following function

```
void v(int a){
  if (a < 2){
     while (c < 3){
       c++;
     }
    if (b > 3){
      c = 12;
    }
  }
  return;
}
```

ADMINISTRIVIA AND ANNOUNCEMENTS

CONTROL FLOW GRAPHS

EECS 677: Software Security Evaluation

Drew Davidson

CLASS PROGRESS

OVERVIEWED TWO ANALYSIS APPROACHES:

- DYNAMIC ANALYSIS: ANALYSIS THAT USES A RUN OF THE PROGRAM
- STATIC ANALYSIS: ANALYSIS WITHOUT RUNNING THE PROGRAM

CONTINUE TO EXPLORE STATIC ANALYSIS

CLASS PROGRESS

Look into concrete forms of Static Analysis

- Particularly interested in dataflow analysis for now
- Building up the underlying abstractions / techniques to perform such analysis

OPPORTUNITIES OF STATIC ANALYSIS

CLASS PROGRESS

Finite Abstractions of Unbounded State Space

- Unnecessary to supply a given program input
- Summarize the behavior of the program under ANY input

LAST TIME: STATIC ANALYSIS

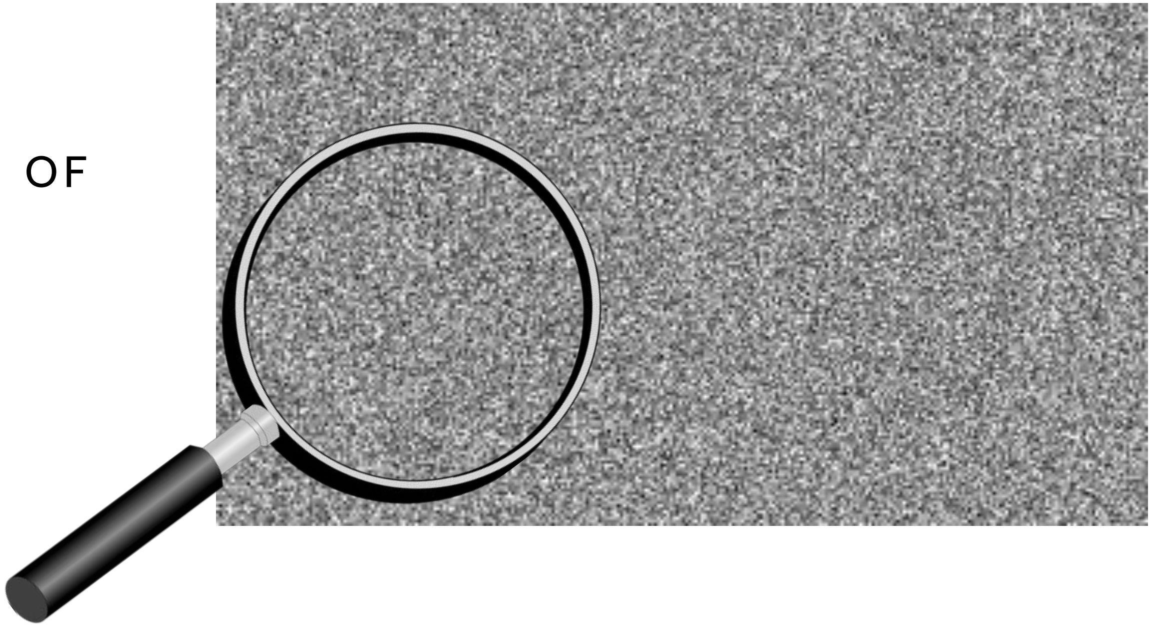
REVIEW: STATIC ANALYSIS

Mentioned some static analysis Techniques

- Syntactic Analysis
- Dataflow Analysis
- Model Checking

Started Building a Fundamental Unit of Static Analysis: the Basic Block

- Sequence of code that executes… sequentially

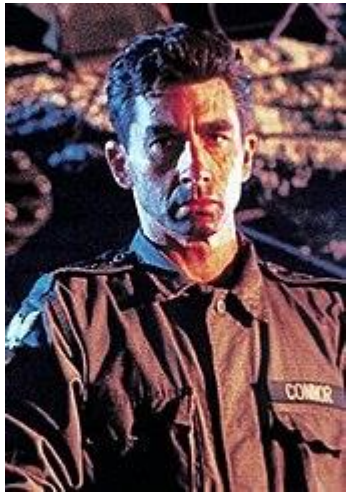


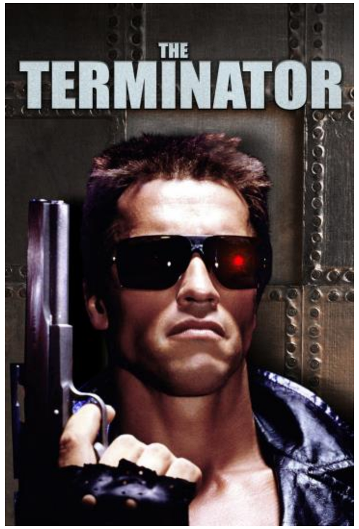
BASIC BLOCKS BOUNDARIES

REVIEW: STATIC ANALYSIS

TWO DISTINGUISHED INSTRUCTIONS IN A BLOCK (MAY BE THE SAME INSTRUCTION)

- Leader: An instruction that begins the block
- Terminator: An instruction that ends the block





BASIC BLOCKS BOUNDARIES

REVIEW: STATIC ANALYSIS

TWO DISTINGUISHED INSTRUCTIONS IN A BLOCK (MAY BE THE SAME INSTRUCTION)

- Leader: An instruction that begins the block

  The first instruction in the procedure

  The target of a jump

  The instruction after an terminator

- Terminator: An instruction that ends the block

  The last instruction of the procedure

  A jump (ifz, goto)

  A call (We'll use a special LINK edge)

BASIC BLOCKS EXAMPLE

STATIC ANALYSIS: CONTROL FLOW GRAPHS

```
y = 0;
if ( g ) {
  x = 1;
  x = 2;
} else {
  x = 3;
  if (g2) {
    x = y;
  }
  x = 4;
}
z = x;
```

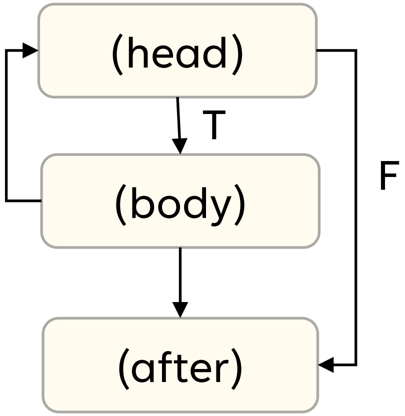
BENEFITS OF BASIC BLOCKS

STATIC ANALYSIS: CONTROL FLOW GRAPHS

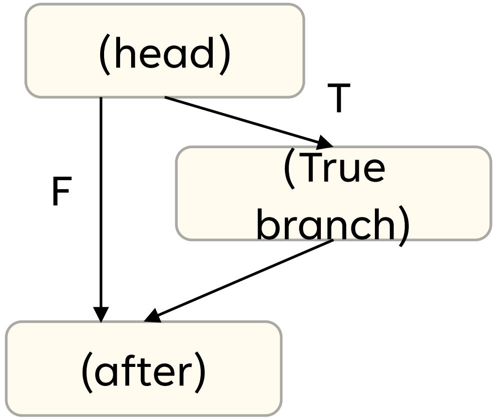
AN ADDITIONAL ABSTRACTION LAYER

- Leader: An instruction that begins the block

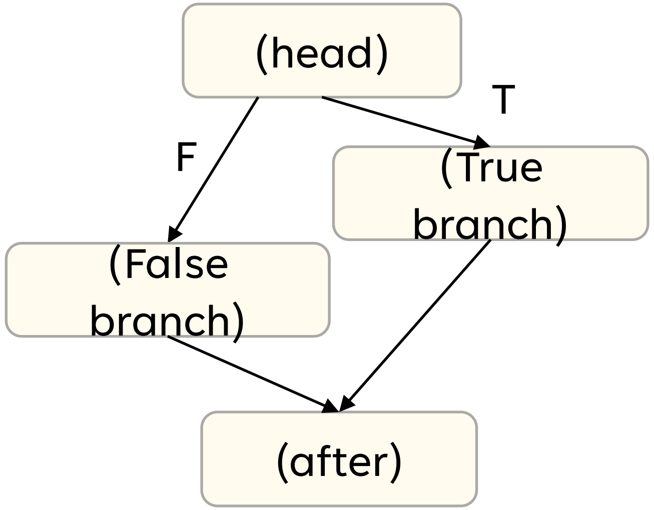
Loops



If-stmt



If-else



CFGS: A PER-FUNCTION ABSTRACTION

STATIC ANALYSIS: CONTROL FLOW GRAPHS

BY DEFINITION, A CFG NEVER INCLUDES MULTIPLE FUNCTIONS

Call instruction simply has a special “link” edge to its successor

CFG-Like analysis is possible on multiple functions, but requires special care to avoid infeasible paths

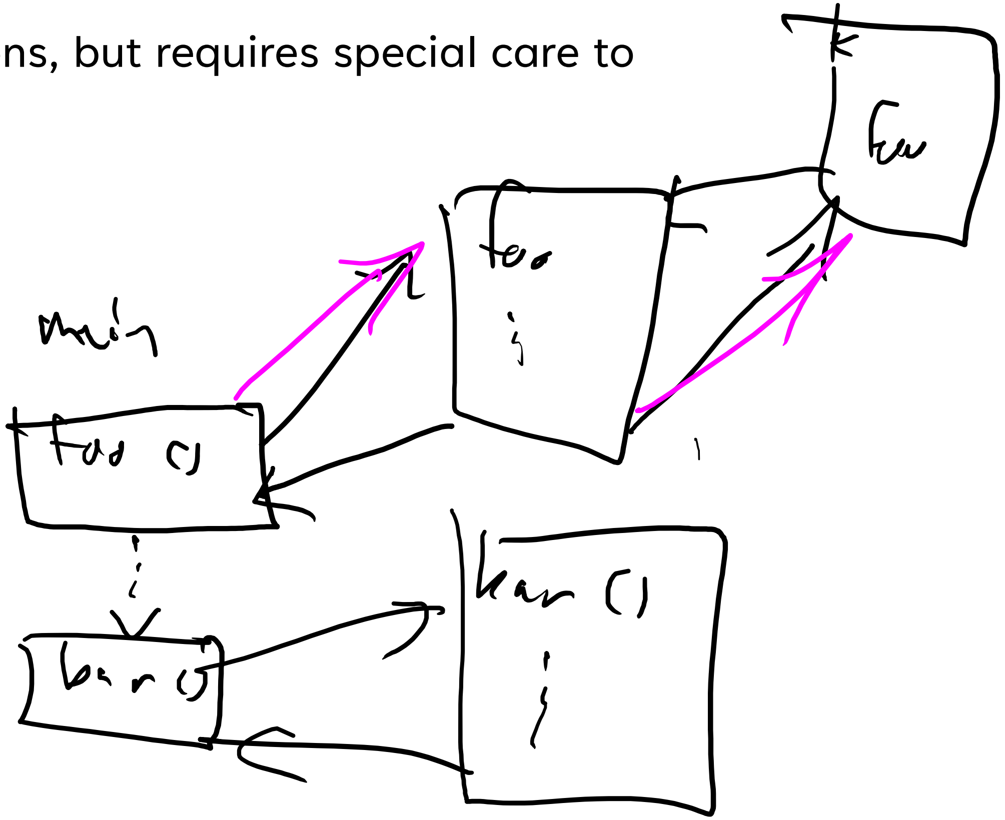
```
foo() {


}

bar() {


}

main() {
    foo();
    bar();
}
```



within a BBL

LECTURE OUTLINE

- (Local) Dataflow analysis
- Global dataflow analysis

DATAFLOW ANALYSIS: BIG IDEA

DATAFLOW ANALYSIS

View Each Statement as a Data Transfer Function

- Transform a program state into a new (updated) program state
- Simple idea: concrete program state into a new concrete program state

state M

y has the value 1

Stmt$_1$: x = y ;

state M’

x has the value 1
y has the value 1

COMPOSING TRANSFER FUNCTIONS

DATAFLOW ANALYSIS

STATEMENTS COMPOSE NATURALLY WITH EACH OTHER* ← they share a BBL

state M

y has the value 1

Stmt$_1$: x = y ;

Stmt$_2$: z = x ;

state M'

x has the value 1 y has the value 1 z has the value 1

For now, we'll only think about analysis within a BBL

AN EARLY WIN

DATAFLOW ANALYSIS

EVEN WITH THIS VERY SIMPLE CONCEPT, MIGHT BE ABLE TO DETECT SOME ISSUES

<u>state M</u>

y has the value 1

Stmt$_1$: x = y ;

y = 1

x = 1

Stmt$_2$: z = 0 ;

y = 1

x = 1

z = 0

Stmt$_3$: p = $\underline{1 / z}$;

FORMALIZING TRANSFER FUNCTIONS

DATAFLOW ANALYSIS

If we want to Build an Automated (Local) Dataflow analysis, we need programmatic Precision

- Some sort of specification of what a statement does
- A statement is a memory state transformer

Memory state M

Stmt$_1$: k += 1 ;

Memory state M’

<u>Need a semantics!</u>

Representation mapping (large) set of memory states to each other

Depend somewhat on the analysis

Goals:

- Keep states manageable
- Handle the uncertainty inherent in static analysis

MEMORY AS VALUE SETS

DATAFLOW ANALYSIS

Let each memory Location correspond to a Set of Values it might Contain

- Define (informally) transfer functions as mapping elements of M to elements of M’

We’re still kinda-dodging the larger semantic questions here, for now lets just say we’re using a big ol’ if statement to define an operator

Memory state M	$\langle k: \{1\}\rangle$	$\langle k: \{3,4\}\rangle$
Stmt$_1$: k += 1 ;		
Memory state M’	$\langle k: \{2\}\rangle$	$\langle k: \{4,5\}\rangle$

COMPOSING VALUE SETS

DATAFLOW ANALYSIS

$\langle y : \{1, \overset{0}{\cancel{2}}\} \rangle$

Stmt$_1$: x = y ;

$\langle y : \{1, \overset{0}{\cancel{2}}\}, x : \{1, \overset{0}{\cancel{2}}\} \rangle$

Stmt$_2$: z = ~~0~~ ;

$\langle y : \{1, \overset{0}{\cancel{2}}\}, x : \{1, \overset{0}{\cancel{2}}\}, z : \{0\} \rangle$

Stmt$_3$: p = 1 / z ;

MODELLING UNCERTAINTY

DATAFLOW ANALYSIS

WE CAN NOW HANDLE OPAQUE DATA SOMEWHAT CLEANLY

z = 0;

Stmt$_1$: x = y ;

< z : {0} >

Stmt$_2$: z = USER_INPUT ;

< z : { MIN_INT - MAX_INT} >

Stmt$_3$: p = 1 / z ;

Stmt$_1$: x = y ;

Stmt$_2$: z = global ;

Stmt$_3$: p = 1 / z ;

LECTURE OUTLINE

- (Local) Dataflow analysis
- Global dataflow analysis

COMPOSING BLOCKS

DATAFLOW: TRANSFER FUNCTIONS

VALUE-SET MODEL OF MEMORY IMPLIES AN EASY WAY TO EXTEND BEYOND LOCAL ANALYSIS

```
int x = 2;
if ( g ){
  x = x - 1;
  if ( g2 ){
    x = x - 1;
  }
}
return 1 / x;
```

COMPOSING BLOCKS

GLOBAL DATAFLOW ANALYSIS

Value-Set Model of memory implies an easy way to extend beyond Local Analysis

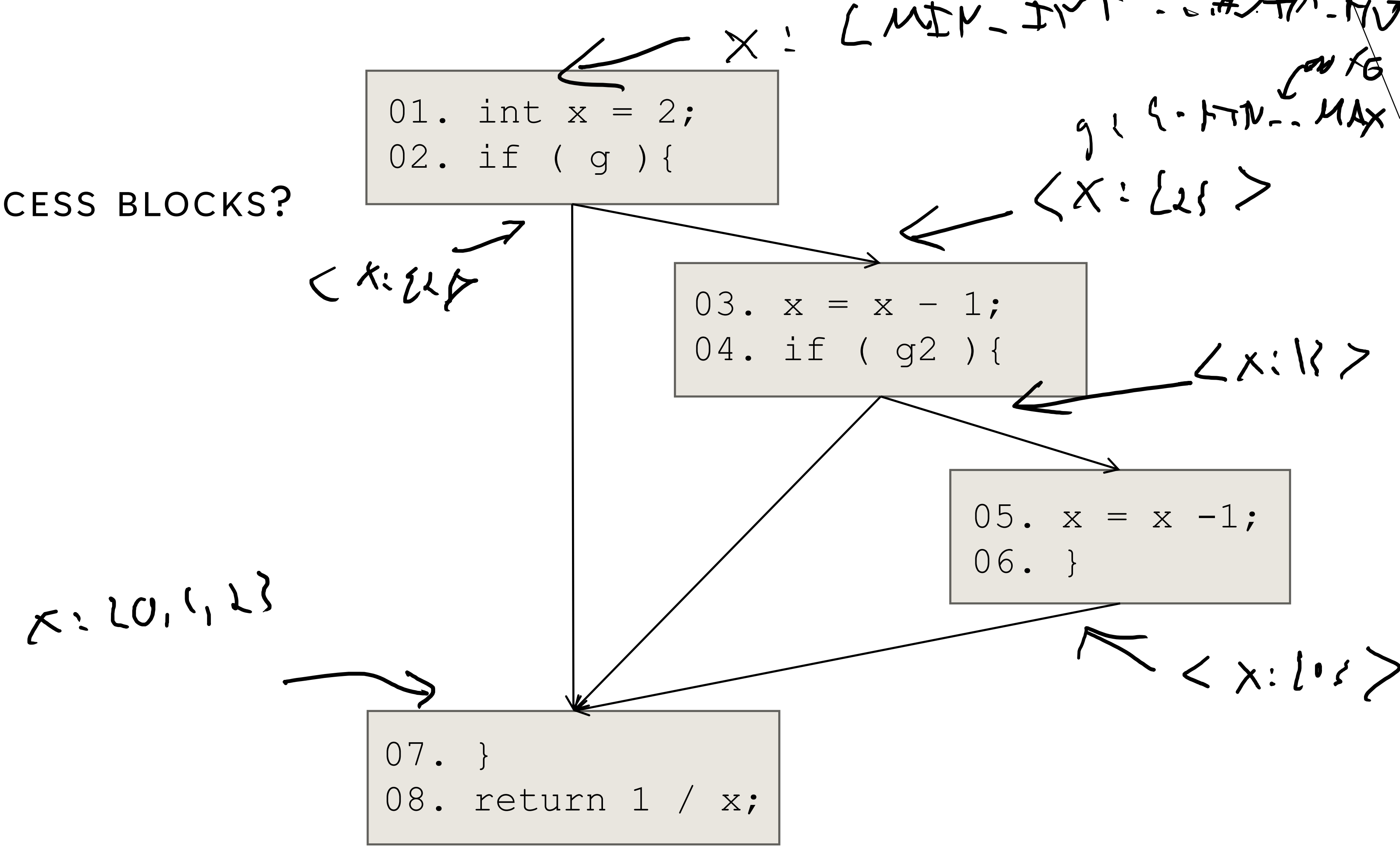
```
int x = 2;
if ( g ){
   x = x - 1;
   if ( g2 ){
     x = x – 1;
   }
}
return 1 / x;
```

CHAOTIC ITERATION

GLOBAL DATAFLOW ANALYSIS

In what order do we process blocks?

```
01. int x = 2;
02. if ( g ){
03.    x = x - 1;
04.     if ( g2 ){
05.       x = x - 1;
06.     }
07. }
08. return 1 / x;
```



```
01. int x = 2;
02. if ( g ){
```

```
03. x = x - 1;
04. if ( g2 ){
```

```
05. x = x -1;
06. }
```

```
07. }
08. return 1 / x;
```

TROUBLE ON THE HORIZON

GLOBAL DATAFLOW ANALYSIS

Loops

LOOPS ARE TOUGH TO HANDLE!

GLOBAL DATAFLOW ANALYSIS

Issues with Loops

- Generate lots of paths
- Cyclic data dependency

Oh, brother! You may have some loops

LECTURE END!

- Local Dataflow analysis
- Global Dataflow analysis