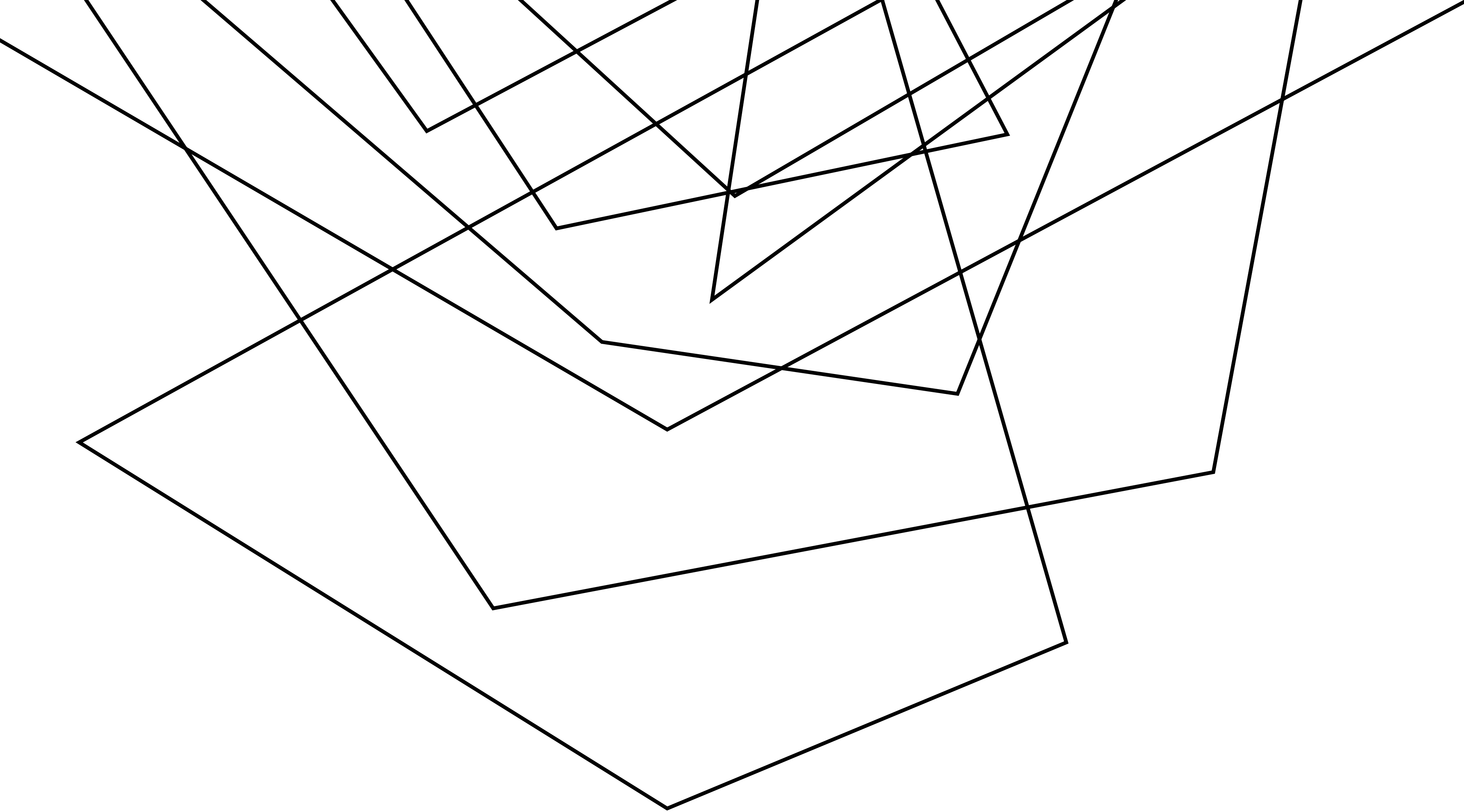
EXERCISE #4

DYNAMIC ANALYSIS REVIEW

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

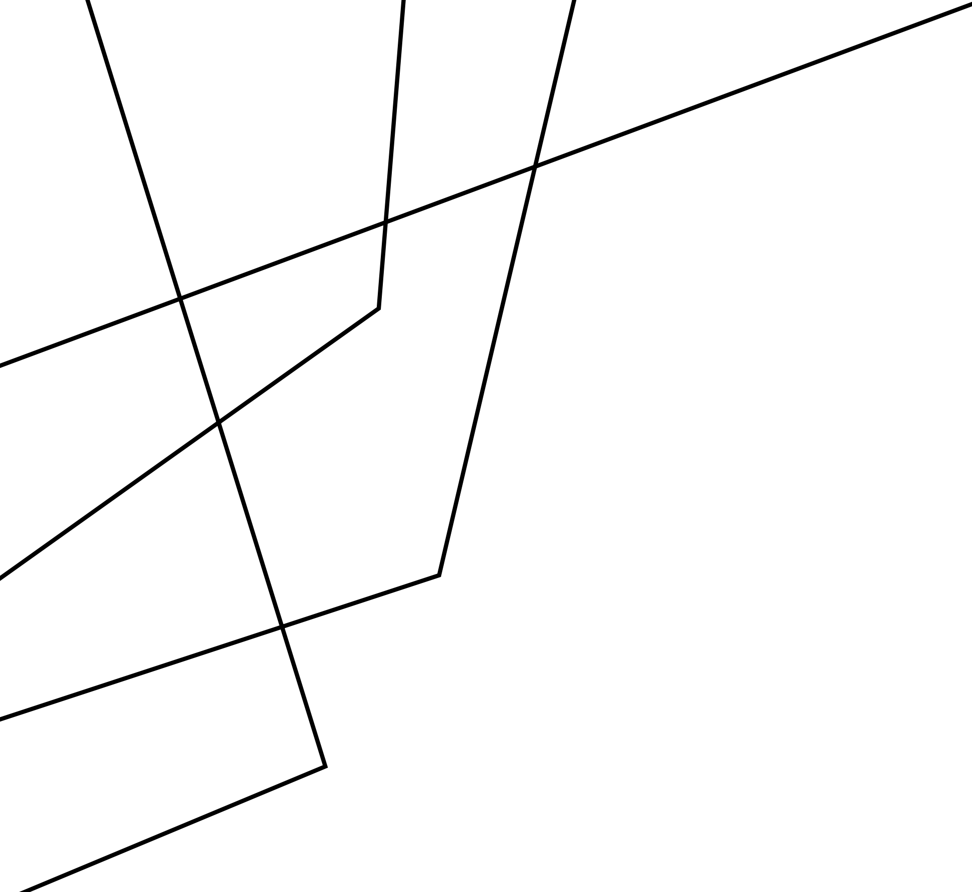
- Give a function and multiple input sets that collectively exercise 100% branch coverage on that function but less than 100% path coverage



STATIC ANALYSIS

EECS 677: Software Security Evaluation

Drew Davidson



ADMINISTRIVIA AND ANNOUNCEMENTS

Quiz dates are now posted

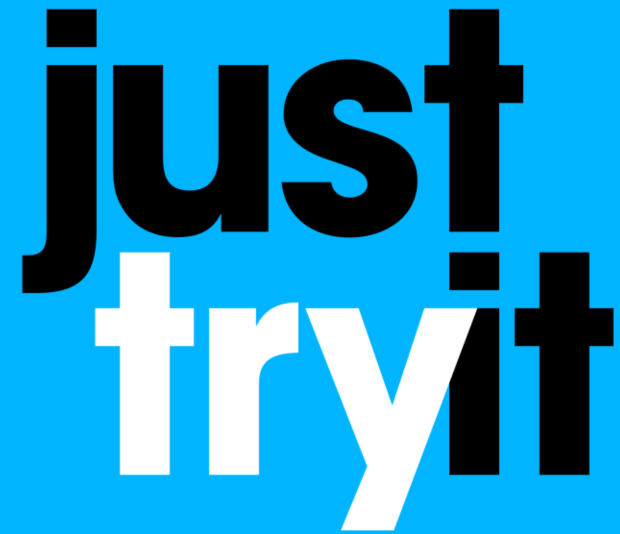
Quiz 1 is NEXT FRIDAY (in class)

LAST TIME: DYNAMIC ANALYSIS

REVIEW: DYNAMIC ANALYSIS

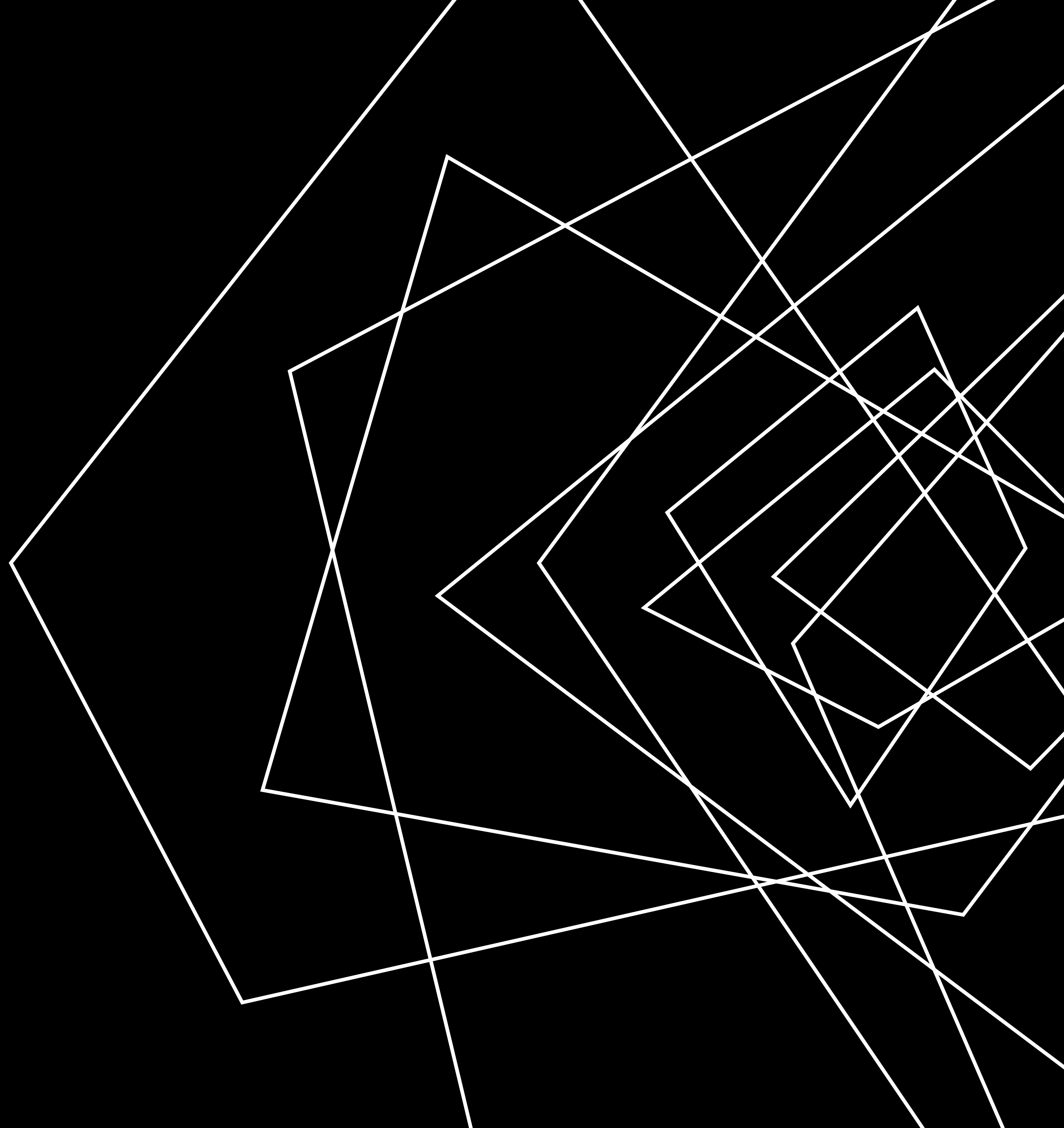
High-Level Overview of a classic auditing technique: testing

- Try the program, see what happens!
- A sound analysis: if you saw it, it happened
- Challenge: exercising all behavior



LECTURE OUTLINE

- Static Analysis
- Control Flow Graph



BEYOND TESTING

INTRO: STATIC ANALYSIS

What if we didn’t have to “guess” at an input?

- Extract the “rules” of the program
- Examine the effect of the program without providing explicit values

SOME FORMS OF STATIC ANALYSIS

CATEGORIZING ANALYSES

Syntax Analysis

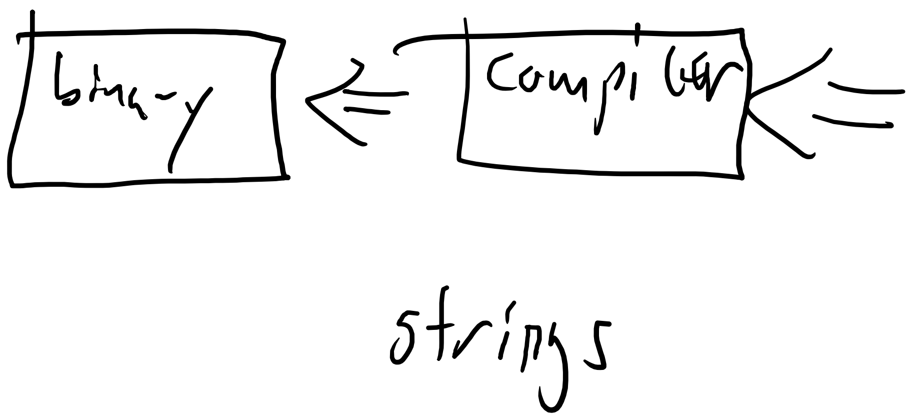
Model Checking

Dataflow Analysis

SYNTAX ANALYSIS

OVERVIEW: STATIC ANALYSIS

Some troubling behavior of a program may be discoverable via simply observing syntactic structure




```
if (password ="12345"){
    // secure thing
}
```

MODEL CHECKING

OVERVIEW: STATIC ANALYSIS

Extract a (finite) state system that approximates the analysis target

Example:

- States: configuration of the system
- Edges: transitions within the system

Check if the system can violate some correctness property

Each state indicates the value of a memory bit

MODEL CHECKING

STATIC ANALYSIS - MODEL CHECKING

State space explosion!

(SYMBOLIC) MODEL CHECKING

STATIC ANALYSIS – MODEL CHECKING

Extract a (finite) state system that approximates the analysis target

- States: configuration**s** of the system
- Edges: transitions within the system

Check if the system can violate some correctness property

Each state indicates a set of values or the truth of some abstract predicate

CEGAR

STATIC ANALYSIS – MODEL CHECKING

Counter**e**xample-**g**uided **a**bstraction **r**efinement

- Begin with a coarse, over-approximate abstraction of the system
- Check system correctness
- If a violation is reported, verify it!
    - If its a true positive – report it
    - If it's a false positive – refine the model to exclude it and check the new model

DATAFLOW ANALYSIS

OVERVIEW: STATIC ANALYSIS

Capture the effect of each statement on the program's data

- Compose the statements together to determine the aggregate effect of the program

ANALYSIS SPECIFICITY

STATIC ANALYSIS: DATAFLOW

```
int f(bool b) {
    Obj * o = null;
    int v = 2;
    if (b) {
        o = new Obj ();
        v = rand_int();
    }
    if (v == 2){
        o->setInvalid()
    }
    return o->property();
}
```

Flow Sensitive

Combine together effect of multiple paths, lose precision

ANALYSIS SPECIFICITY

STATIC ANALYSIS: DATAFLOW

```
int f(bool b) {
    Obj * o = null;
    int v = 2;
    if (b) {
        o = new Obj ();
        v = rand_int();
    }
    if (v == 2){
        o->setInvalid()
    }
    return o->property();
}
```

Path Sensitive

ABSTRACT INTERPRETATION

CATEGORIZING ANALYSES

(Over)approximate the state of the program
(Over)approximate the domain of values

```
a = 0;            0
a: 0  →  if (a <= 0) {
              a = 4;
a: +  →   }
a: +
          1/a;
```

ABSTRACT INTERPRETATION

CATEGORIZING ANALYSES

(Over)approximate the state of the program
(Over)approximate the domain of values

> Anything that isn't crystal clear to a static analysis tool probably isn't clear to your fellow programmers, either. The classic hacker disdain for "bondage and discipline languages" is short-sighted – the needs of large, long-lived, multi-programmer projects are just different than the quick work you do for yourself
>
> \- John Carmack

OVERVIEW DONE!

CATEGORIZING ANALYSES

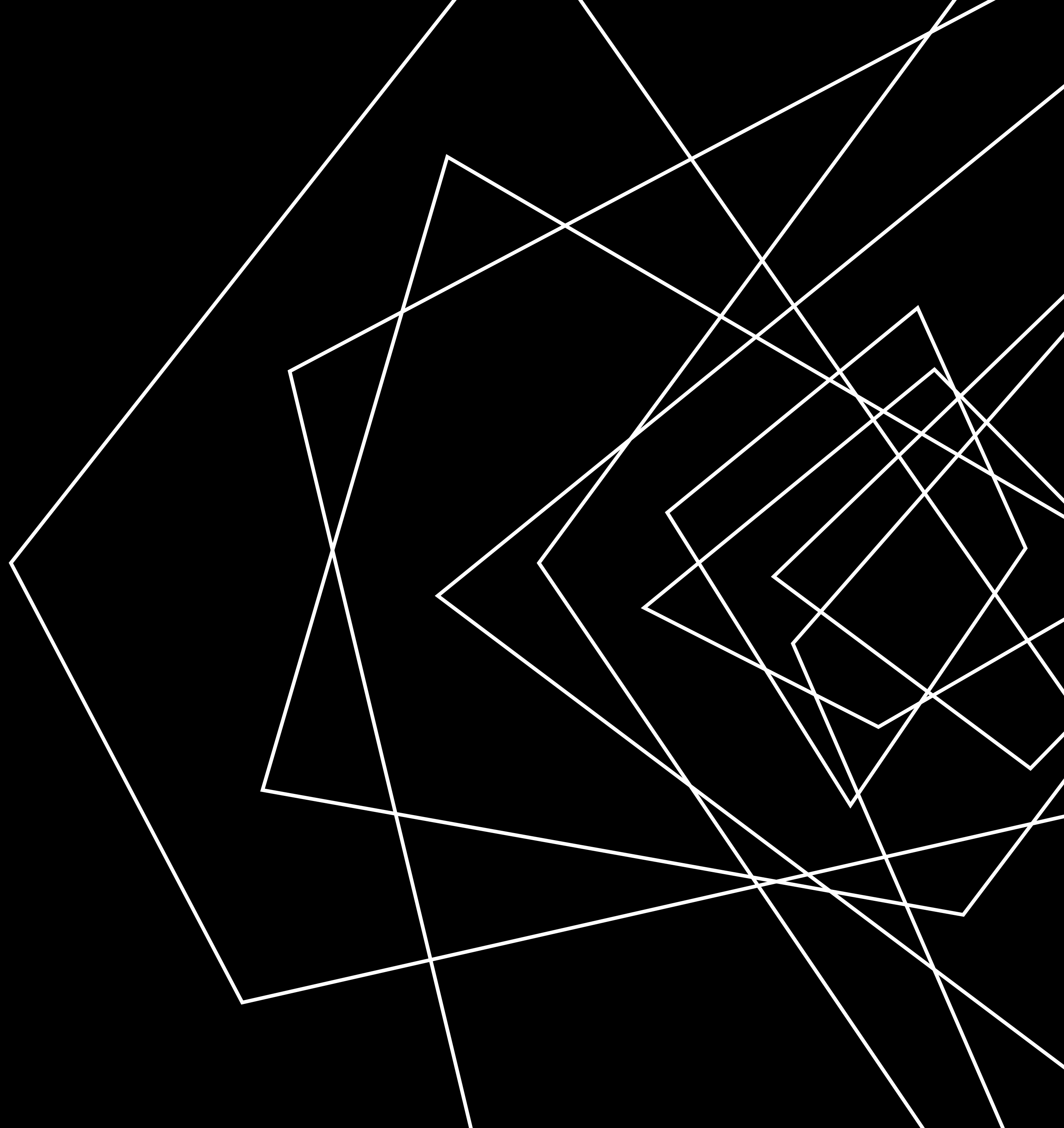
We’ll cover many of these techniques (and more!)

Next up:

- Start looking at toolsets to build our analyses
- Looking at the kinds of program flaws that can cause problems

LECTURE OUTLINE

- Static Analysis Overveiw
- Control Flow Graphs



CONTROL FLOW GRAPHS

STATIC ANALYSIS: CONTROL FLOW GRAPHS

Program analysis relies heavily on two questions

- (How) can we get to a particular program point?
- What is the program configuration at a given point?

Helpful to structure program instructions as a graph

- Visualize transfer of control
- Avail ourselves of graph analyses (e.g. reachabilty)

FLOWCHARTS

STATIC ANALYSIS: CONTROL FLOW GRAPHS

NOTATION

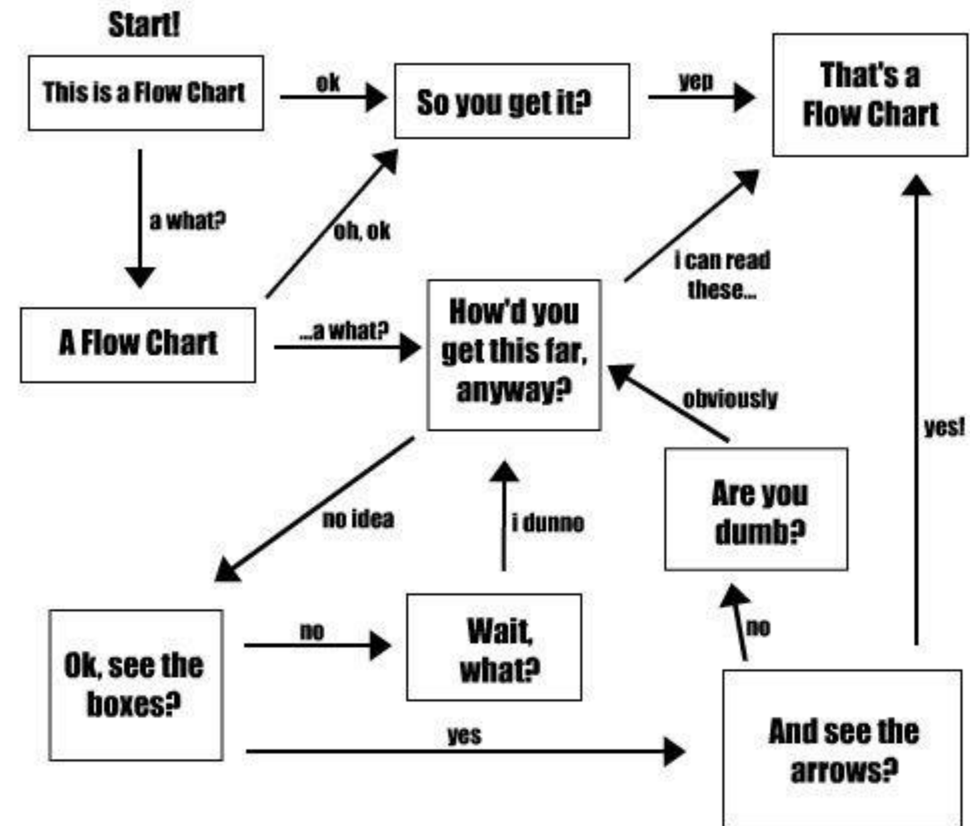
NODES ARE INSTRUCTIONS

EDGES GO TO SUCCESSOR NODES UNDER APPROPRIATE CONDITION

OPERATION

EXECUTE CURRENT INSTRUCTION

PROCEED TO THE RIGHT SUCCESSOR



CODE FLOWCHARTS

STATIC ANALYSIS: CONTROL FLOW GRAPHS

NOTATION

NODES ARE INSTRUCTIONS

EDGES GO TO SUCCESSOR NODES UNDER APPROPRIATE CONDITION

OPERATION

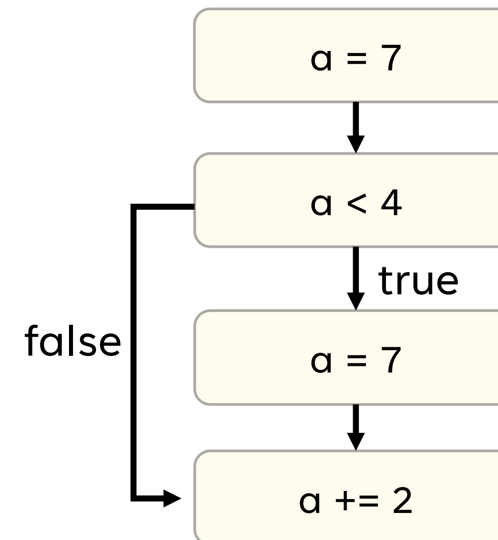
EXECUTE CURRENT INSTRUCTION

PROCEED TO THE RIGHT SUCCESSOR

source code

```
a = 7;
if (a < 4){
  a = 7;
}
a += 2;
```

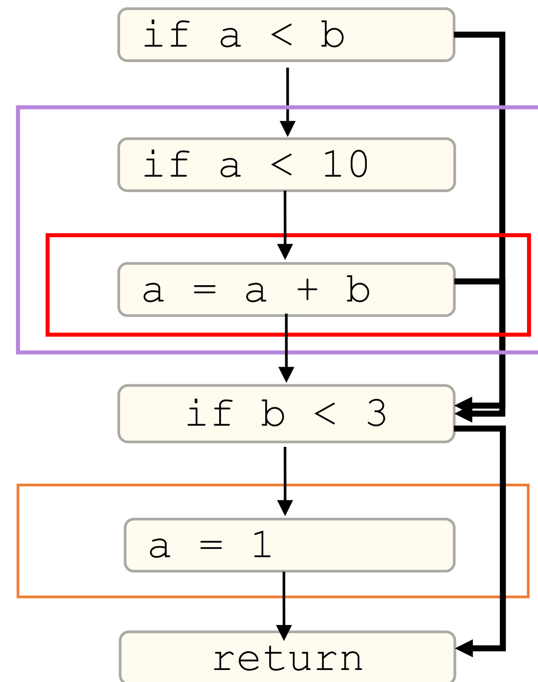
Instruction Flowgraph



FLOWCHARTS: VISUALIZING CONTROL

STATIC ANALYSIS: CONTROL FLOW GRAPHS

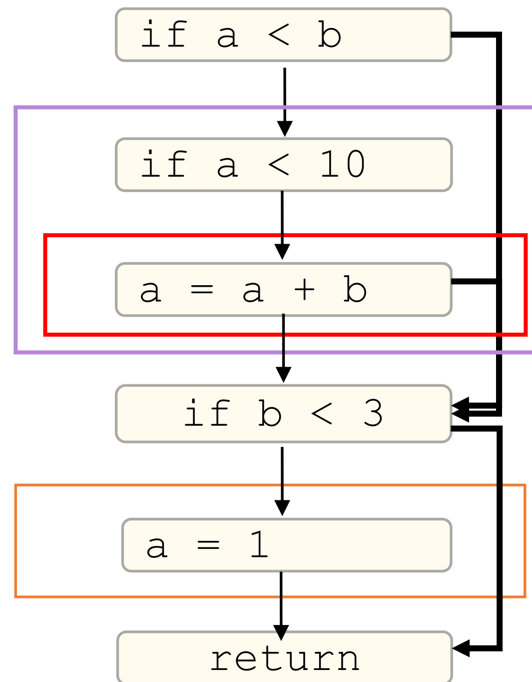
```
void funk(int a, int b){
    if (a < b){
        if (a < 10){
            a = a + b;
        }
    }
    if (b < 3){
        a = 1;
    }
    return;
}
```

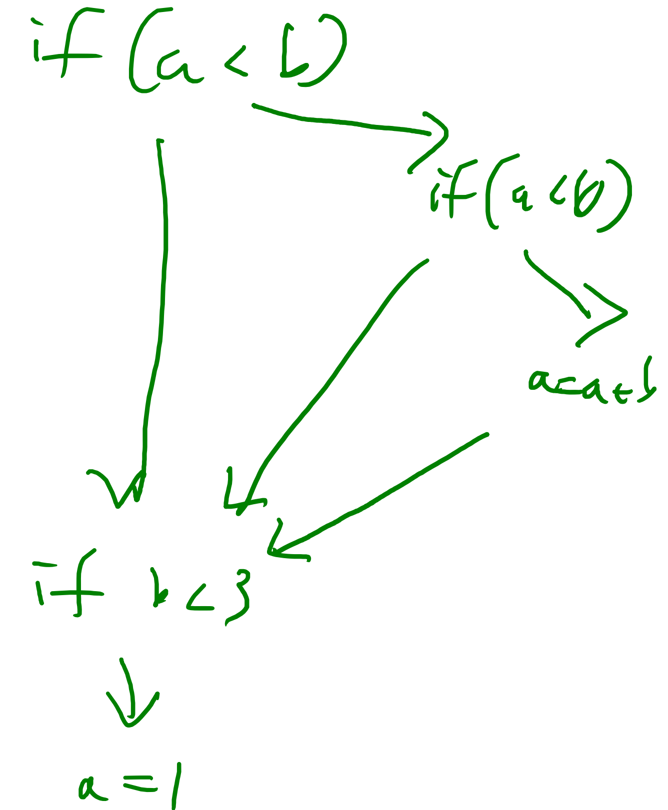


FLOWCHARTS: VISUALIZING CONTROL

STATIC ANALYSIS: CONTROL FLOW GRAPHS

```
void funk(int a, int b){
    if (a < b){
        if (a < 10){
            a = a + b;
        }
    }
    if (b < 3){
        a = 1;
    }
    return;
}
```





FLOWCHARTS: A USEFUL TOOL

STATIC ANALYSIS: CONTROL FLOW GRAPHS

MAYBE THIS IS HOW YOU LEARNED TO THINK ABOUT CODE!

IT'S A NICE WAY TO VISUALIZE THE *CONTROL FLOW* OF THE PROGRAM

WE CAN EXTEND THIS INTUITION FOR PROGRAM ANALYSIS

COMPACTING THE FLOW CHART

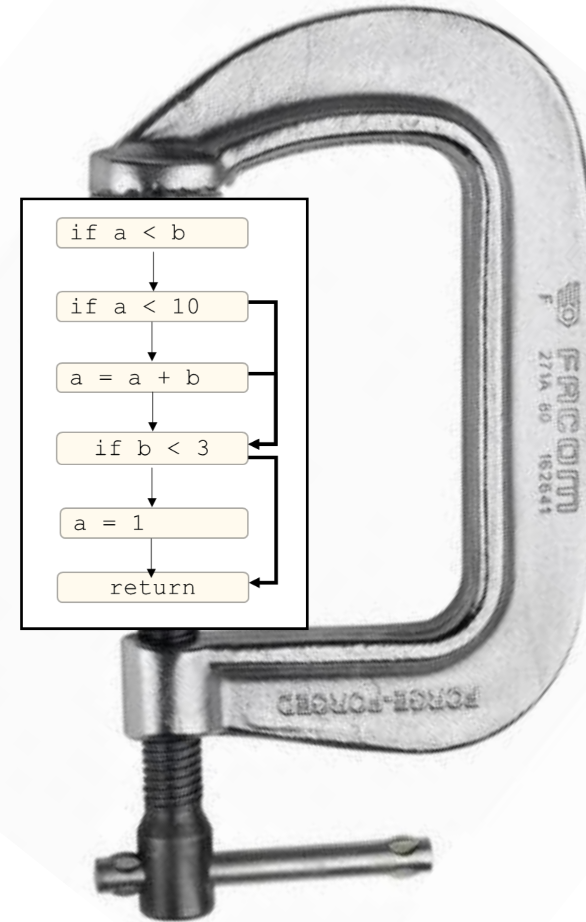
STATIC ANALYSIS: CONTROL FLOW GRAPHS

FROM FLOWCHARTS TO CONTROL FLOW GRAPHS

- This graph is needlessly verbose
- Too many nodes that communicate nothing

FWHAT IF WE ELIMINATE THE 1 INSTRUCTION PER NODE CONSTRAINT?

- Attempt to use as few edges as possible



BASIC BLOCKS

STATIC ANALYSIS: CONTROL FLOW GRAPHS

DEFINITION: SEQUENCE OF INSTRUCTIONS GUARANTEED TO EXECUTE WITHOUT INTERRUPTION

BASIC BLOCKS BOUNDARIES

STATIC ANALYSIS: CONTROL FLOW GRAPHS

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

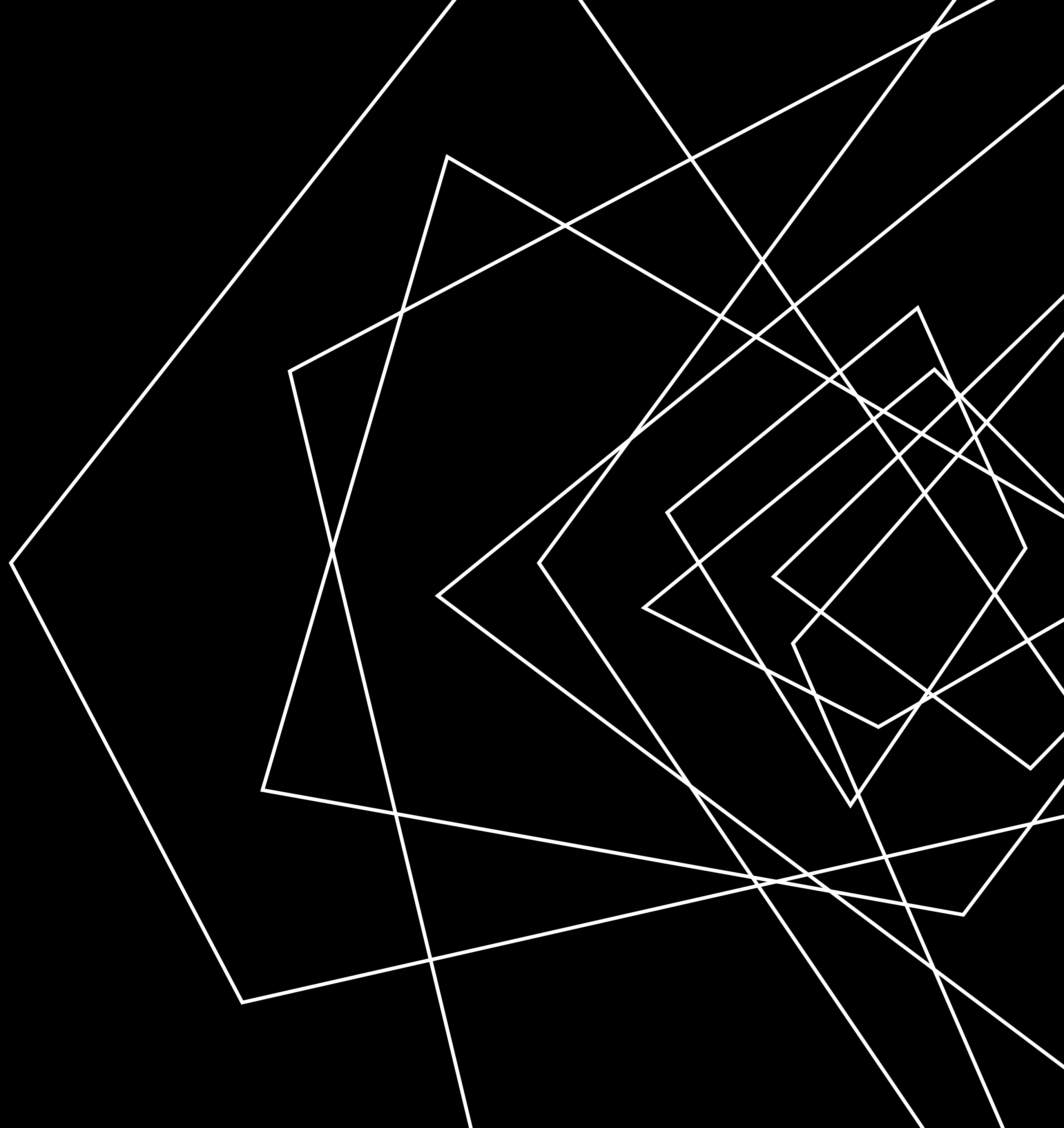
STATIC ANALYSIS: CONTROL FLOW GRAPHS

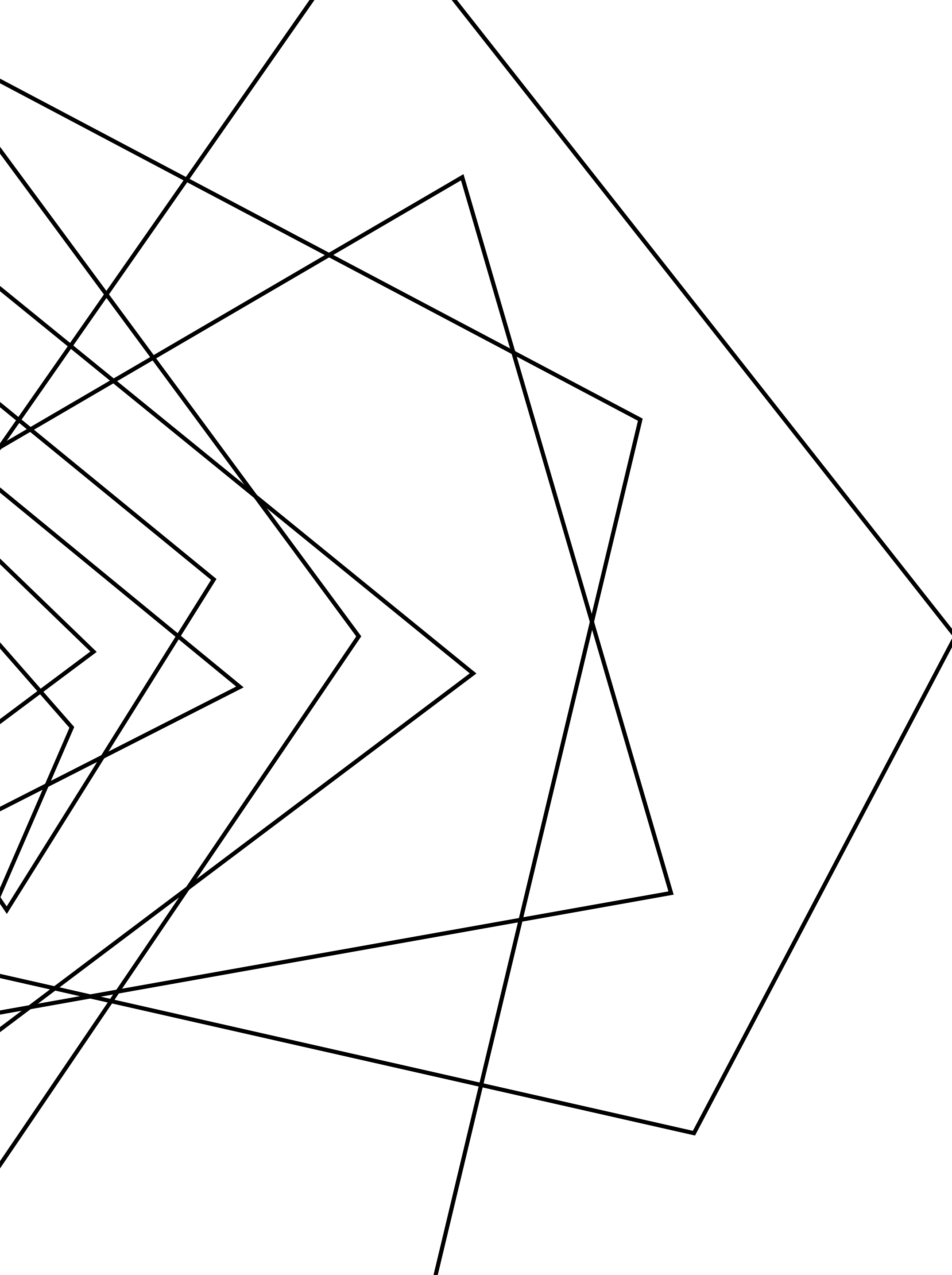
TWO DISTINGUISHED INSTRUCTIONS IN A BLOCK (MAY BE THE SAME INSTRUCTION)

- 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)*

LECTURE END!

- Static Analysis
- Control Flow Graphs





NEXT TIME

EXPLORE THE USE OF THE CONTROL FLOW GRAPH FOR FINDING VULNERABILITIES

SHOW ADDITIONAL PROGRAM ABSTRACTIONS TO SIMPLIFY ANALYSIS, IN PARTICULAR SSA FORM