

EXERCISE

CALL TARGET REVIEW

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

Draw the exploded supergraph for the following program:

```
class SupClass{
public:
    virtual int fun(SupClass * in){
        in->fun();
        return 1;
    }
};
class SubA : public SupClass{
    int fun(SupClass * in){
        return 2;
    }
};
class SubB : public SupClass{
    void fun(SupClass * in){
        return 3;
    }
};
int main(){
    SupClass * s = new SubA();
    s->fun();
}
```

EXERCISE

CALL TARGET ANALYSIS REVIEW

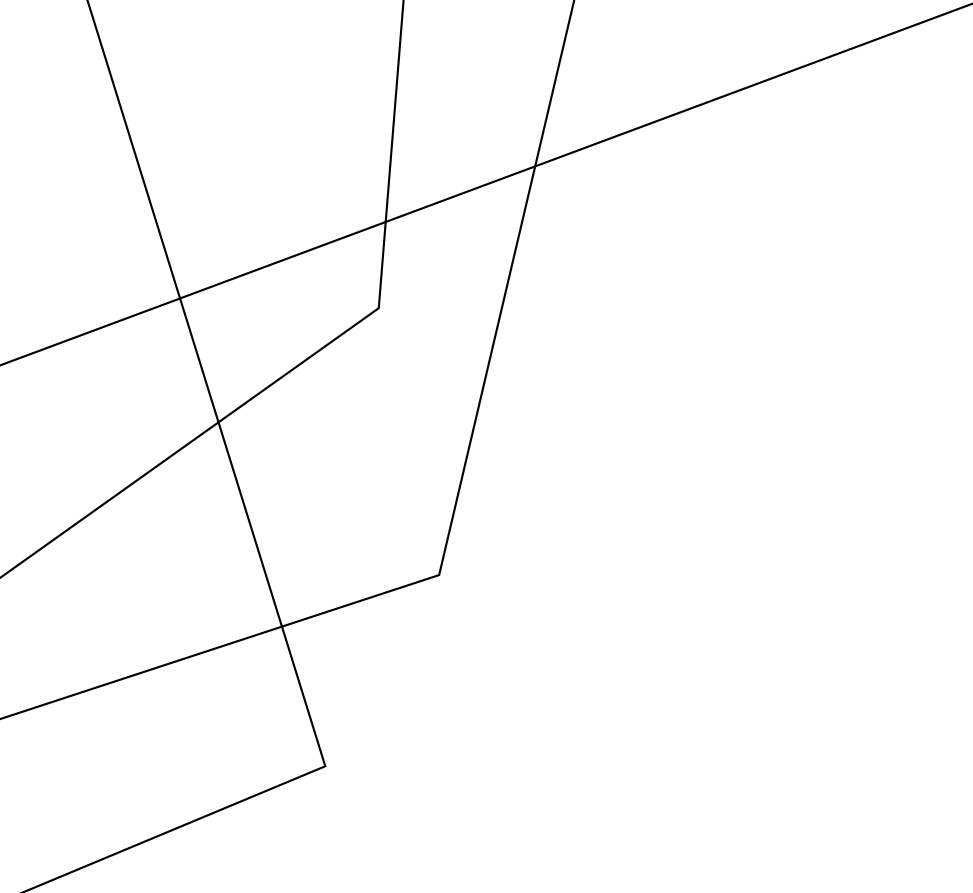
Write your name and answer the following on a piece of paper

Draw the call graph of the following program according to CHA

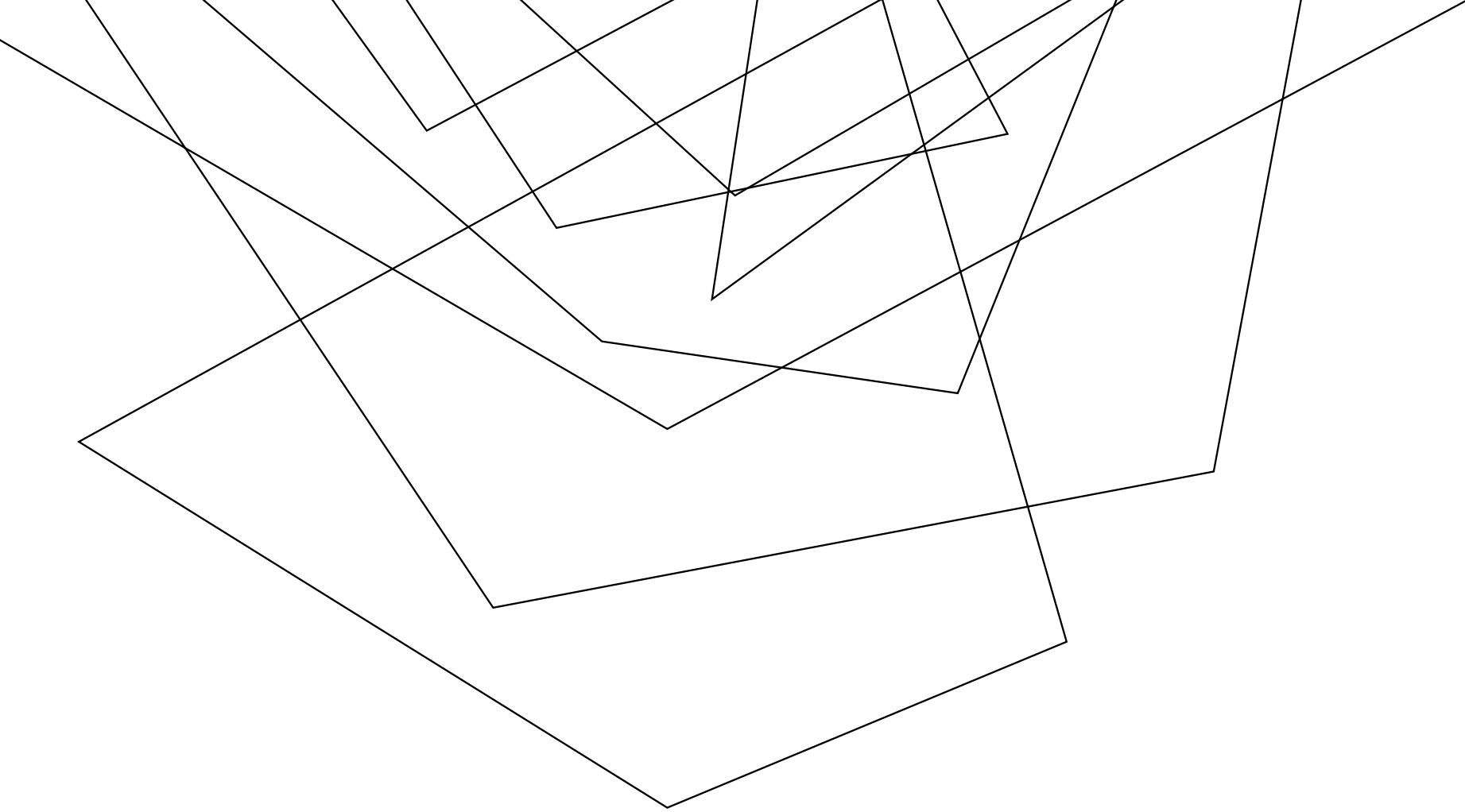
```
1: class SupClass{
2: public:
3:     virtual int fun(SupClass * in){
4:         in->fun(in);
5:     }
6: };
7:
8: class SubA : public SupClass {
9:     virtual int fun(SupClass * in){
10:         in->fun(in);
11:     }
12: };
13: class SubB : public SupClass {
14:     virtual int fun(SupClass * in){
15:         in->fun(in);
16:     }
17: };
18: int main(){
19:     SupClass * s = new SubA();
20:     s->fun(s);
21: }
```

EXERCISE SOLUTION

CALL TARGET ANALYSIS REVIEW

Abstract geometric lines in the top left corner, consisting of several thin black lines forming a series of overlapping, tilted rectangular shapes.

ADMINISTRIVIA AND ANNOUNCEMENTS



POINTS-TO ANALYSIS

EECS 677: Software Security Evaluation

Drew Davidson

LAST TIME: INTERPROCEDURAL ANALYSIS

REVIEW: LAST LECTURE

CONSIDER THE EFFECT OF MULTIPLE FUNCTIONS

Simplistic

- Function overturn all global / aliased facts

Supergraph / Context String

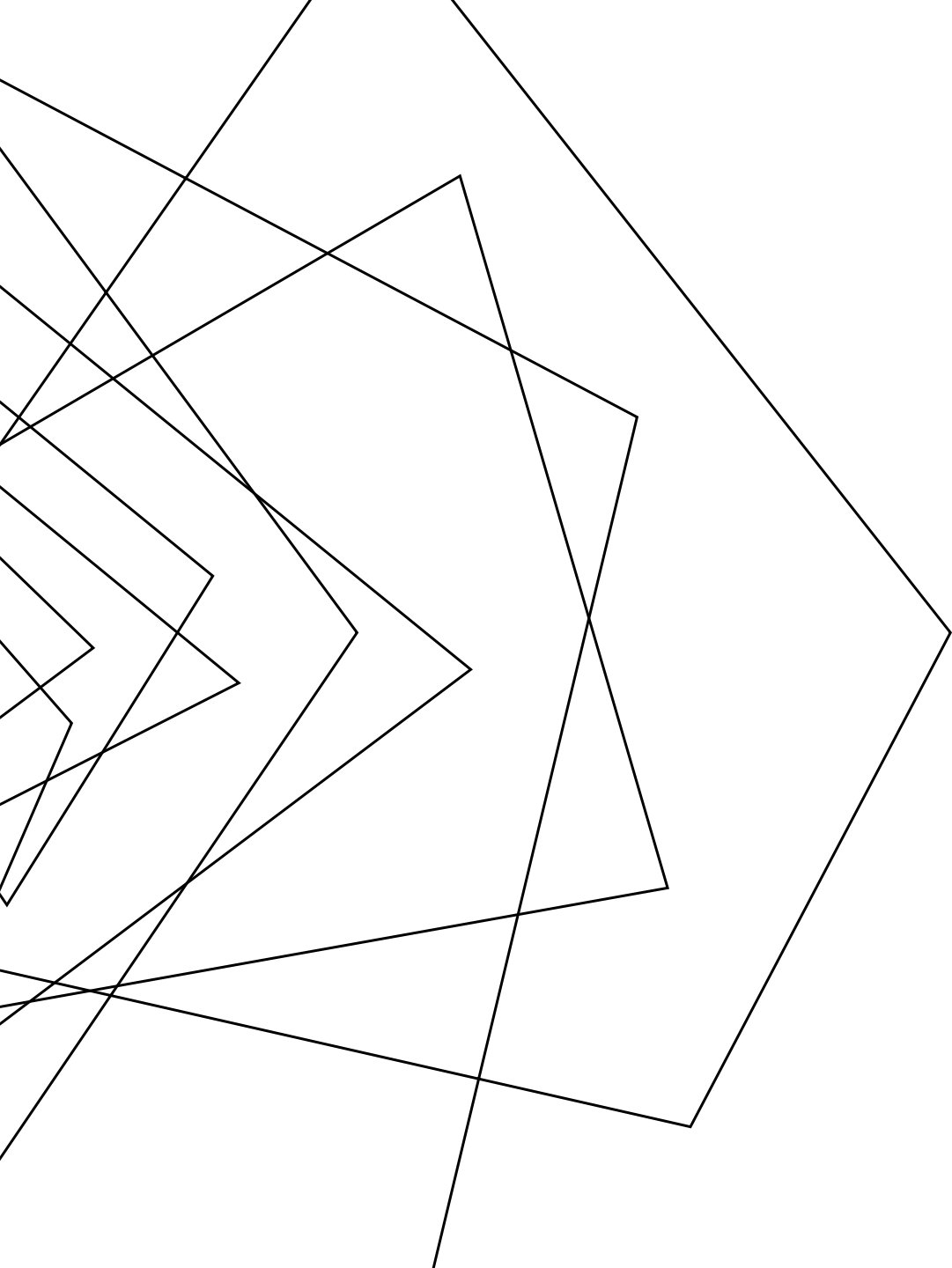
- 1-CFA (use a call-chain of 1)

Summary Information

- Use GMOD and GREF to cut down on imprecision

```
int g;
int v1;
int v2;
int fn(int a){
    if (a > 1){
        return 0;
    }
    return 1;
}

int main(){
    g = 1;
    v1 = fn(1);
    v1 = fn(v1);
    v2 = v1 / g;
    return v2 / v1;
}
```

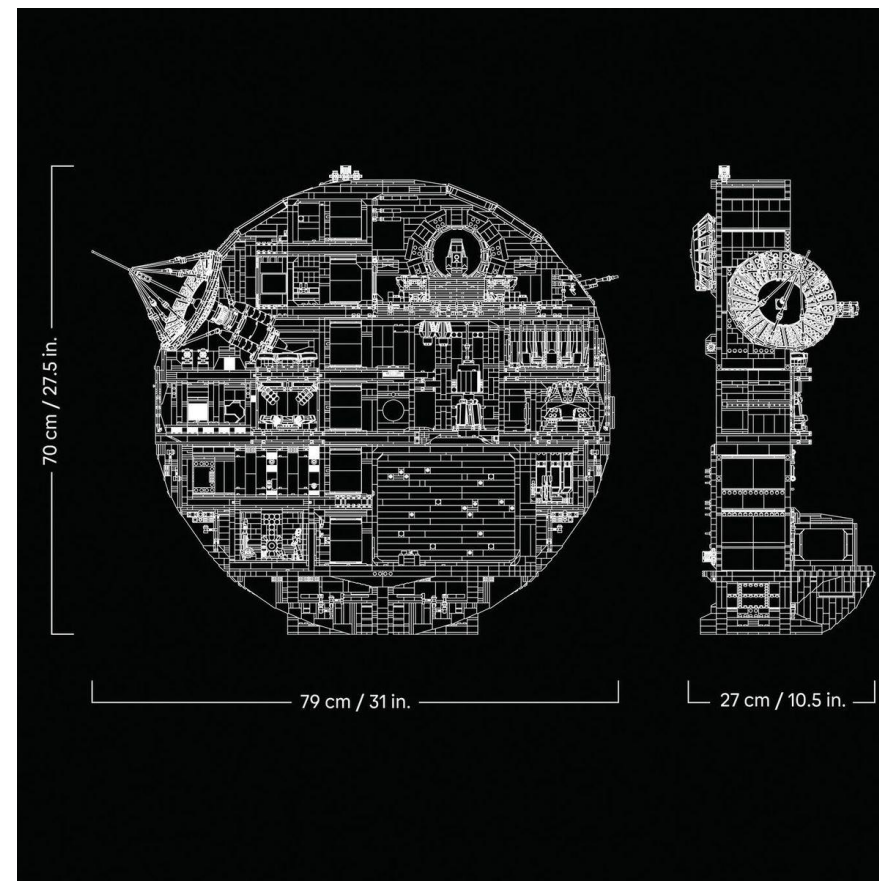


VIBE CHECK

WE'RE BUILDING A FULLY-OPERATIONAL
STATIC ANALYSIS

VIBE CHECK

WE'RE BUILDING A FULLY-OPERATIONAL
STATIC ANALYSIS



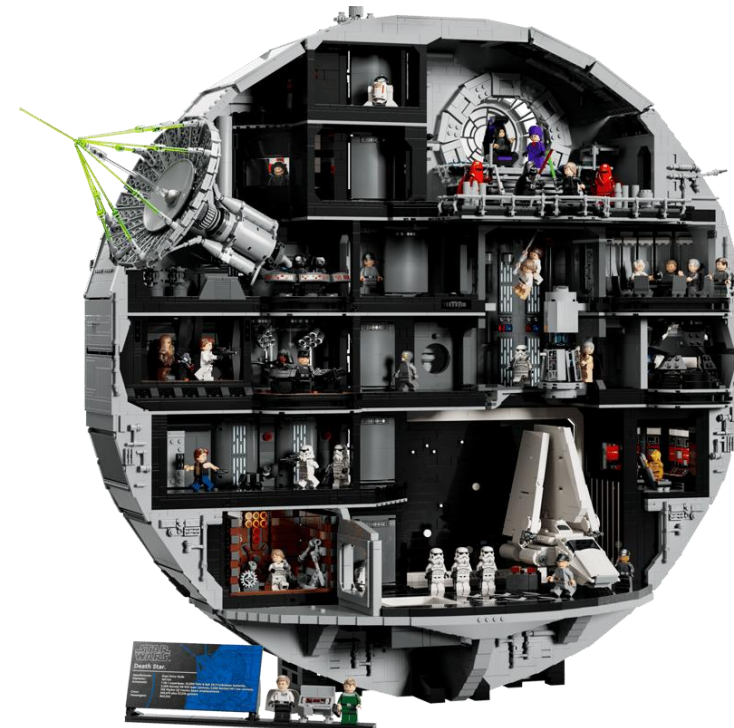
A TOOL CAPABLE OF ANSWERING A HOST
OF QUESTIONS ABOUT PROGRAMS

VIBE CHECK

WE'RE BUILDING A FULLY-OPERATIONAL
STATIC ANALYSIS

A TOOL CAPABLE OF ANSWERING A HOST
OF QUESTIONS ABOUT PROGRAMS

BUILD UP OUR ANALYSIS TO ENCAPSULATE
FEATURES OF NON-TOY PROGRAMS



EXTENDING DATAFLOW



Control-Flow Graphs



Call Graphs



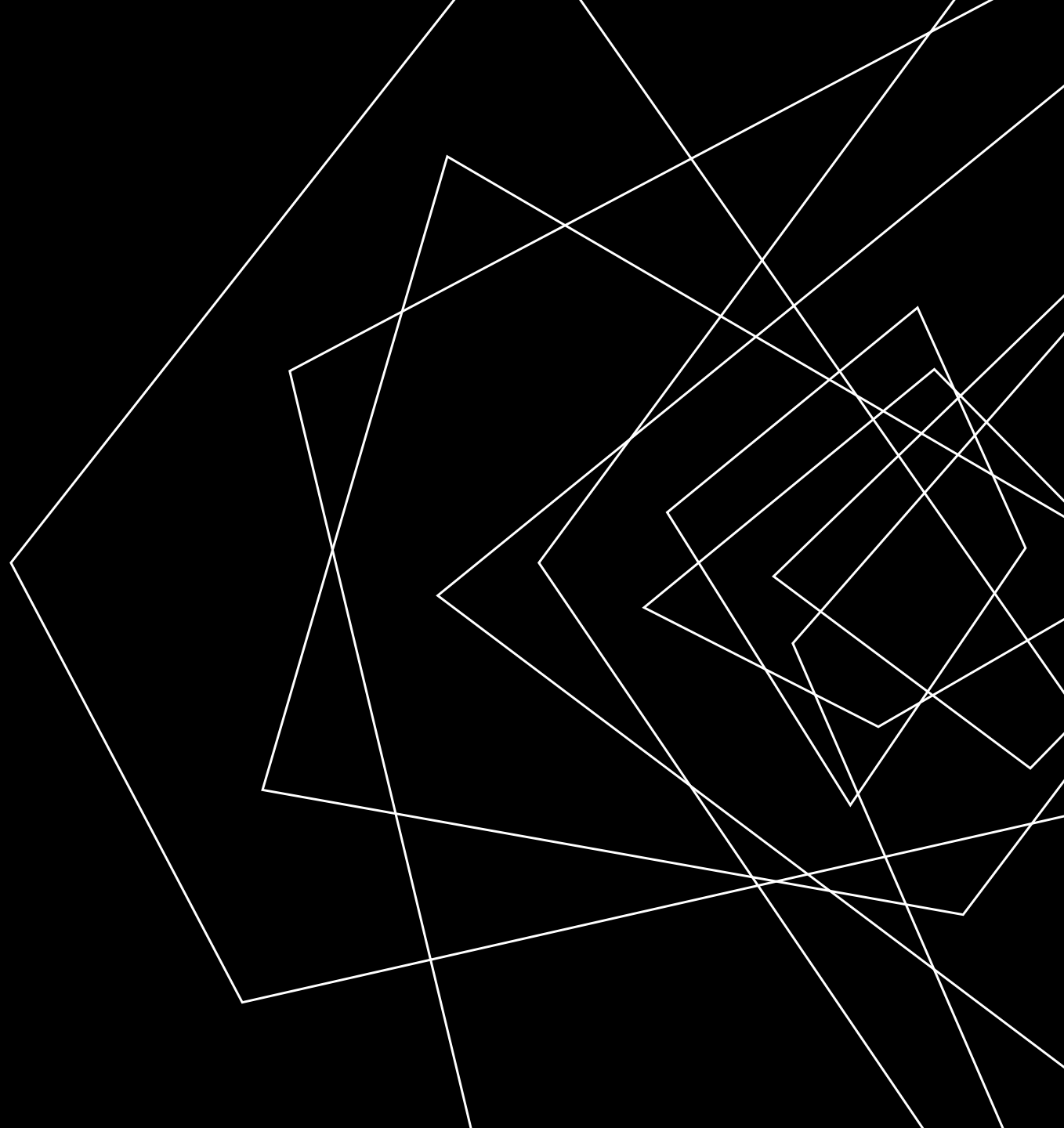
Call Targets



Pointer Analysis

LECTURE OUTLINE

- Aliasing Relationships
- Representing Points-To
- Analyses



ALIASES

ALIASING RELATIONSHIPS

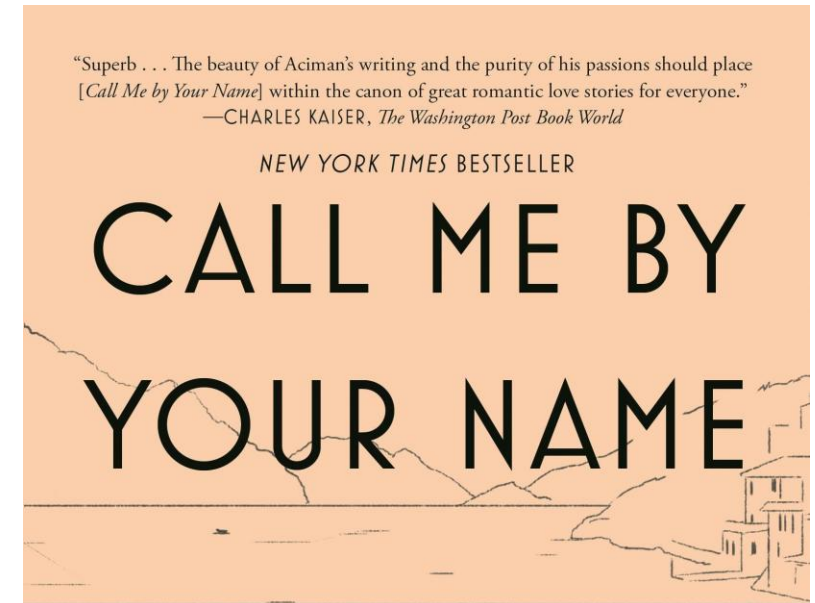
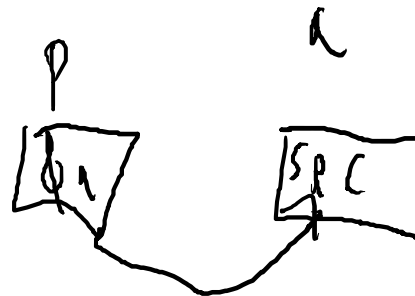
MANY REFERENCES, ONE ENTITY

Machine Code (and many source languages) allow data values that address other locations (it's all integers, after all)

AWARENESS NEEDED IN DATAFLOW FACTS!

Pseudocode

```
p = &a;
*p = SOURCE();
SINK(a);
```



MANY FORMS OF ALIASING

ALIASING RELATIONSHIPS

THERE'S LOTS OF WAYS THAT
ALIASING HAPPENS

Pass-by-reference

Object References

$k_{one} = \text{new } k()$
 $k_{two} = \text{new } k()$
 $one = two;$

$\text{foo}(\text{int} \& a, \text{int} \& b) \{$

$\}$
 $\text{bar}() \{$
 $\text{int } c;$
 $\text{foo}(c, c);$
 $\}$

"Some people call me
the space cowboy."

leans in

"Some call me the
gangster of love."

"Some people call me M—"

Barista: I'm just gonna put
'Steve' on the cup, ok?

[Credit](#)

ALIASES AND SAFE ANALYSIS

ALIASING RELATIONSHIPS

MAY-POINT(P)

The set of locations to which P *might* refer (i.e. point-to)

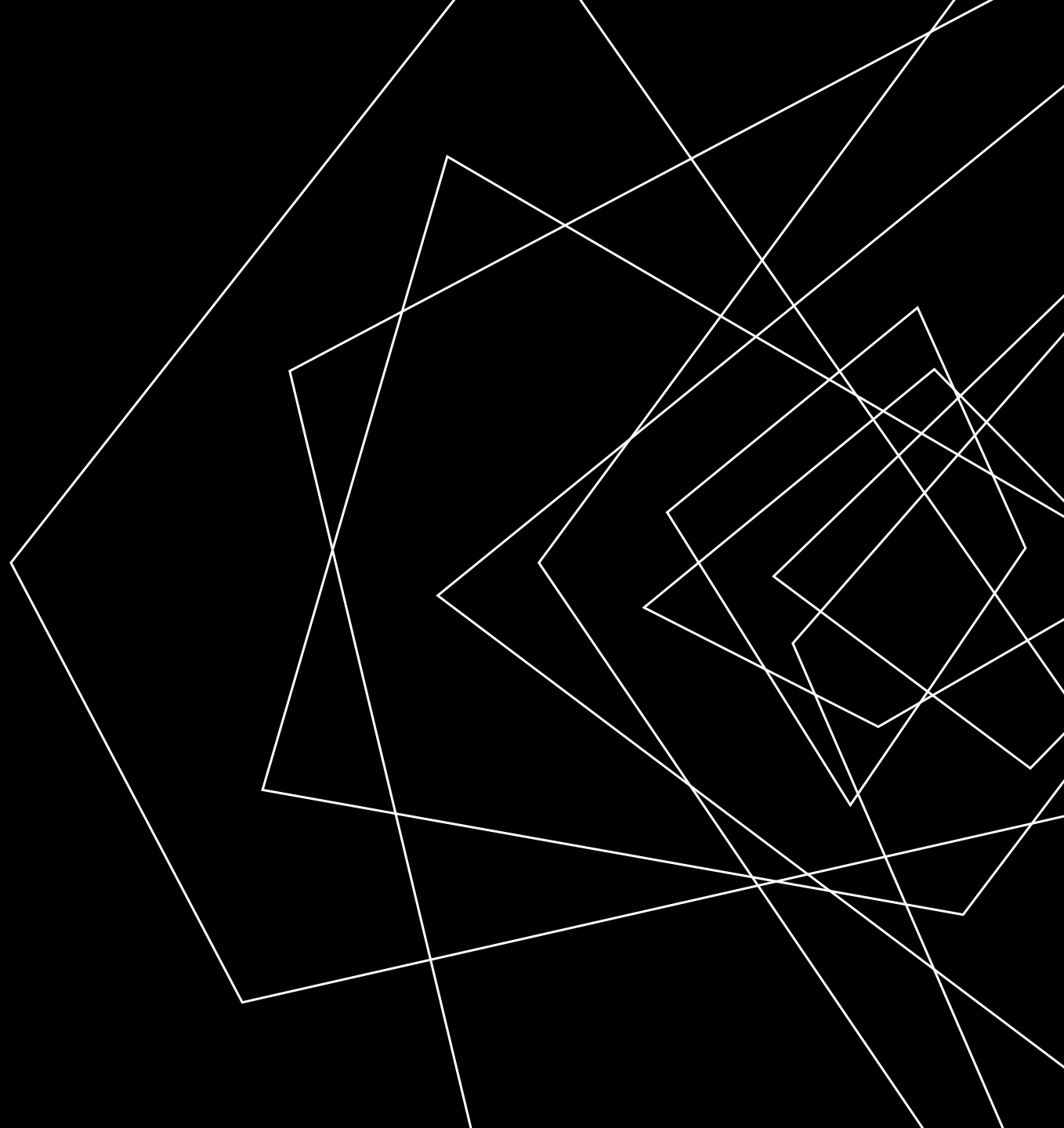
MUST-POINT(P)

The set of locations to which P *definitely* refers (i.e. points-to)

```
p = &a;  
q = p;  
p = &b;  
r = p;
```

LECTURE OUTLINE

- Aliasing Relationships
- Representing Points-To
- Analyses



REPRESENTING POINTS-TO RELATIONSHIPS

REPRESENTING POINTS-TO

APPROACH #1: CONSTRAINT SETS

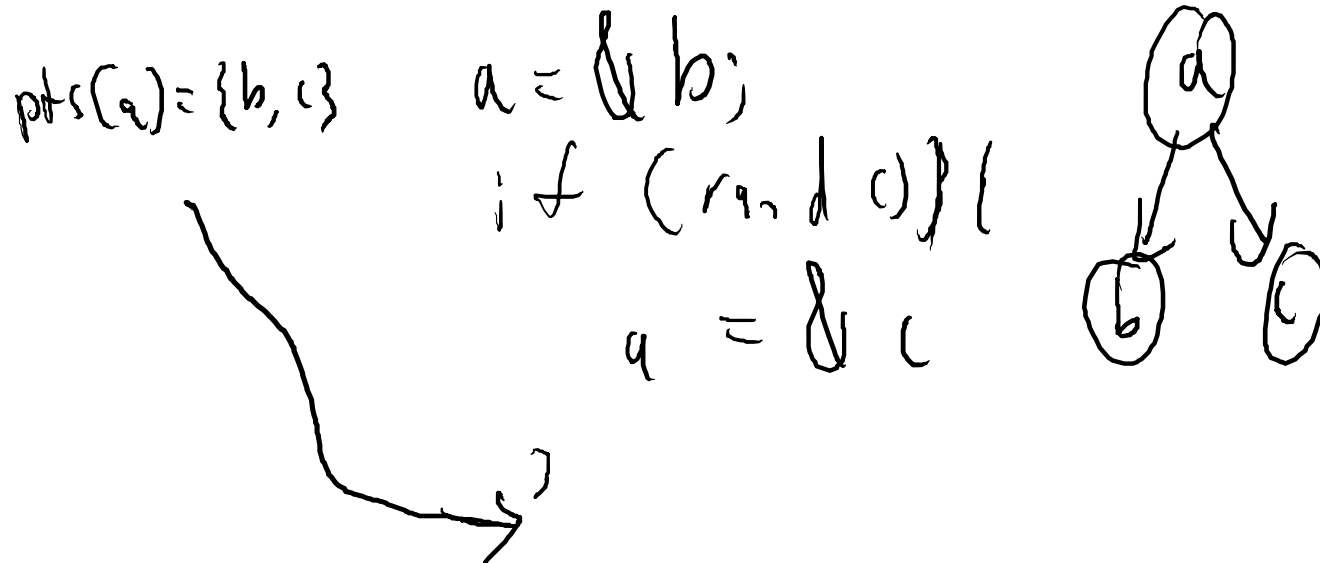
Constraint sets

$\text{loc}(X)$: the memory location of X itself

$\text{pts}(X)$: the memory location X may address

APPROACH #2: POINTS-TO GRAPH

A node-and-edge representation of the may-point relationship



ISSUES COMPUTING POINTS-TO

REPRESENTING POINTS-TO

PROBLEMS

MUTUAL DEPENDENCE

Dataflow needed to compute points-to

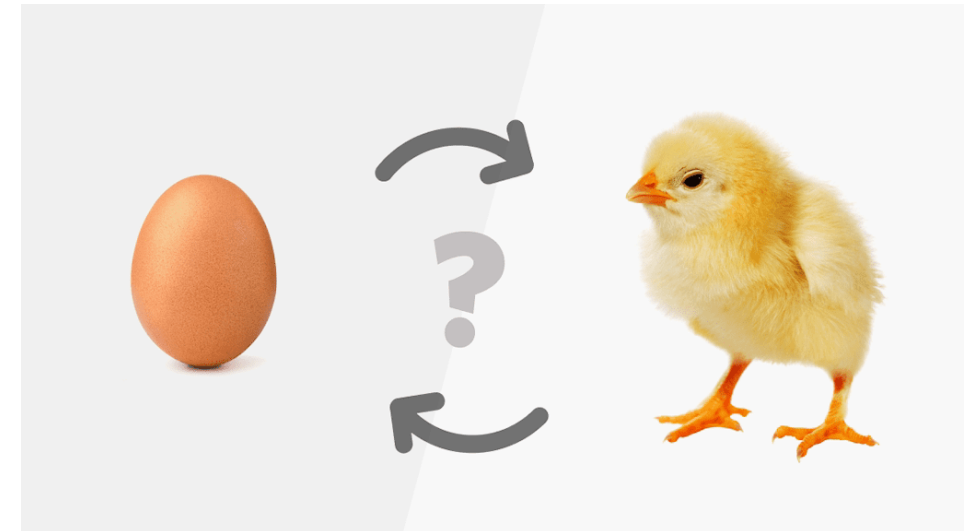
Points to needed to compute dataflow

SCALABILITY

A program spanning graph at each instruction point?!

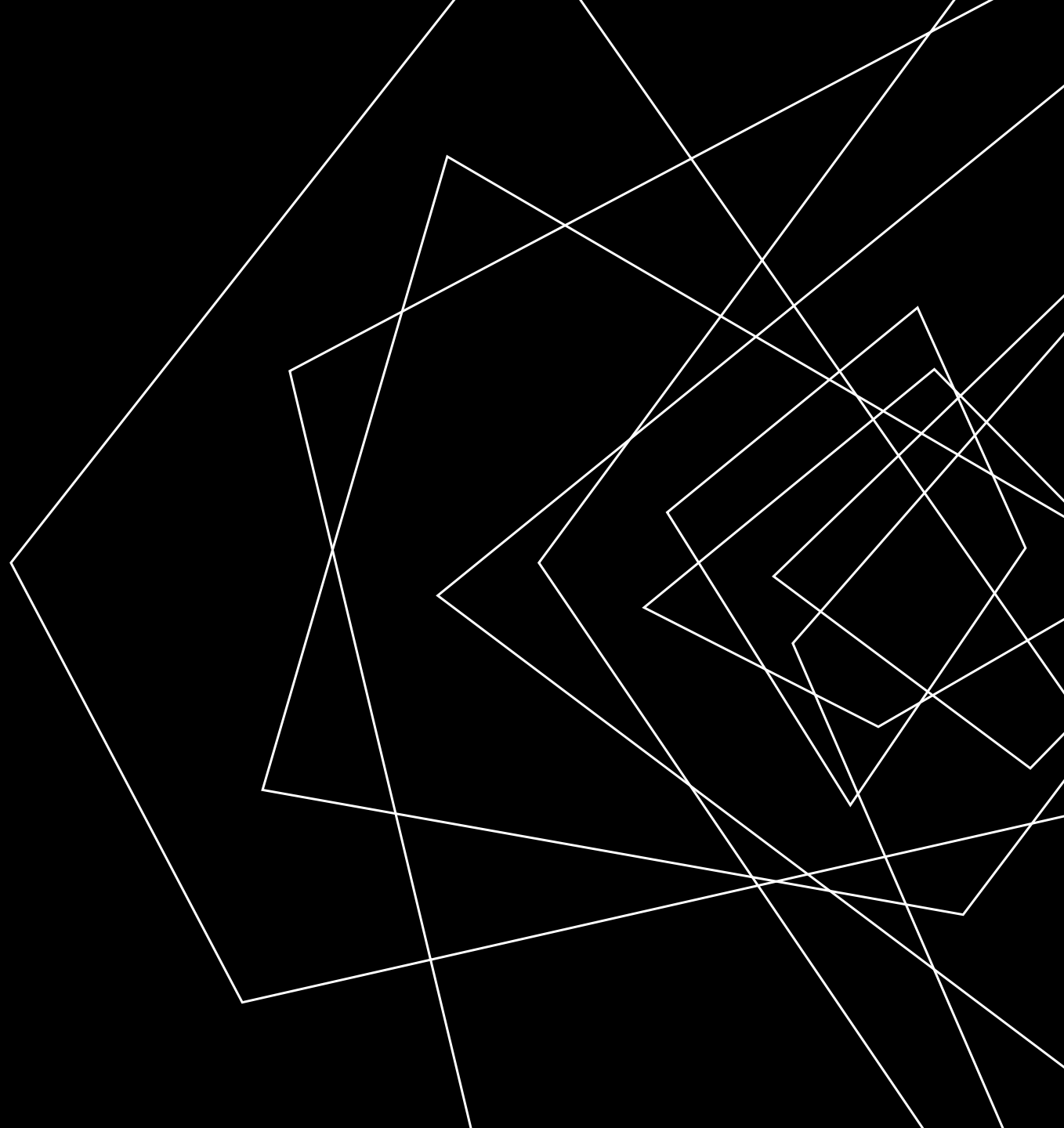
MITIGATION

A flow-insensitive, context-insensitive analysis



LECTURE OUTLINE

- Aliasing Relationships
- Representing Points-To
- Analyses



TWO ALGORITHMS

POINTS-TO ANALYSIS

ANDERSEN'S ANALYSIS AND STEENSGARD'S ANALYSIS

Both flow-insensitive

Both saturation algorithms

Both translate operations into constraints, solve system of constraints

Steensgard's is a coarser approximation

(less granularity means faster saturation)

SUBSET CONSTRAINTS

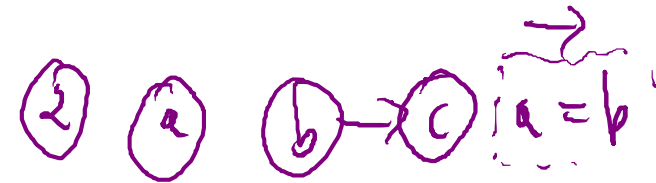
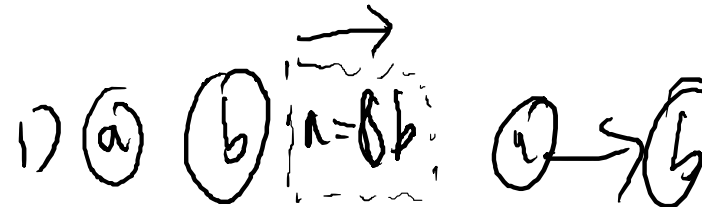
ANDERSEN'S ANALYSIS

$\{a = \&k$
 $\{b = \&m$
 $\}$
 $a = b$

A FLOW-INSENSITIVE ALGORITHM

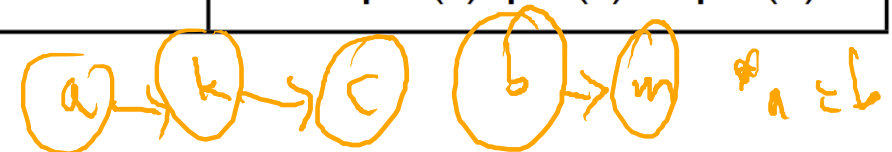
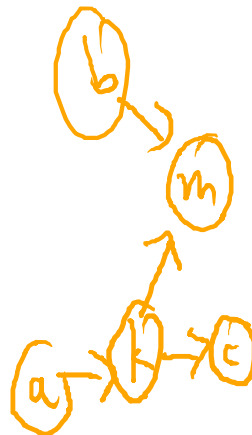
Each statement adds a constraint over the points-to sets

End up with a (solvable) system of constraints



1
 2
 3

Constraint type	Assignment	Constraint	Meaning
Base	$a = \&b$	$a \supseteq \{b\}$	$\text{loc}(b) \in \text{pts}(a)$
Simple	$a = b$	$a \supseteq b$	$\text{pts}(a) \supseteq \text{pts}(b)$
Complex	$a = *b$	$a \supseteq *b$	$\forall v \in \text{pts}(b). \text{pts}(a) \supseteq \text{pts}(v)$
Complex	$*a = b$	$*a \supseteq b$	$\forall v \in \text{pts}(a). \text{pts}(v) \supseteq \text{pts}(b)$



SUBSET CONSTRAINTS

ANDERSEN'S ANALYSIS

A FLOW-INSENSITIVE ALGORITHM

Each statement adds a constraint over the points-to sets

End up with a (solvable) system of constraints

Constraint type	Assignment	Constraint	Meaning
Base	$a = \&b$	$a \supseteq \{b\}$	$\text{loc}(b) \in \text{pts}(a)$
Simple	$a = b$	$a \supseteq b$	$\text{pts}(a) \supseteq \text{pts}(b)$
Complex	$a = *b$	$a \supseteq *b$	$\forall v \in \text{pts}(b). \text{pts}(a) \supseteq \text{pts}(v)$
Complex	$*a = b$	$*a \supseteq b$	$\forall v \in \text{pts}(a). \text{pts}(v) \supseteq \text{pts}(b)$

Program

$p = \&a;$
 $q = p;$
 $p = \&b;$
 $r = p;$

Constraints

$p \supseteq \{a\}$
 $q \supseteq p$
 $p \supseteq \{b\}$
 $r \supseteq p$

Initial

$\text{pts}(p) = \{\}$
 $\text{pts}(q) = \{\}$
 $\text{pts}(r) = \{\}$
 $\text{pts}(a) = \{\}$
 $\text{pts}(b) = \{\}$

Pass One

$\text{pts}(p) = \{a, b\}$
 $\text{pts}(q) = \{a\}$
 $\text{pts}(r) = \{a, b\}$
 $\text{pts}(a) = \{\}$
 $\text{pts}(b) = \{\}$

Pass Two

$\text{pts}(p) = \{a, b\}$
 $\text{pts}(q) = \{a, b\}$
 $\text{pts}(r) = \{a, b\}$
 $\text{pts}(a) = \{\}$
 $\text{pts}(b) = \{\}$

ANOTHER EXAMPLE

ANDERSEN'S ANALYSIS

A FLOW-INSENSITIVE ALGORITHM

Each statement adds a constraint over the points-to sets

End up with a (solvable) system of constraints

Constraint type	Assignment	Constraint	Meaning
Base	$a = \&b$	$a \supseteq \{b\}$	$\text{loc}(b) \in \text{pts}(a)$
Simple	$a = b$	$a \supseteq b$	$\text{pts}(a) \supseteq \text{pts}(b)$
Complex	$a = *b$	$a \supseteq *b$	$\forall v \in \text{pts}(b). \text{pts}(a) \supseteq \text{pts}(v)$
Complex	$*a = b$	$*a \supseteq b$	$\forall v \in \text{pts}(a). \text{pts}(v) \supseteq \text{pts}(b)$

Program

$p = \&a$

$q = \&b$

$*p = q;$

$r = \&c;$

$s = p;$

$t = *p;$

$*s = r;$

Constraints

$p \supseteq \{a\}$

$q \supseteq \{b\}$

$*p \supseteq q$

$r \supseteq \{c\}$

$s \supseteq p$

$t \supseteq *p$

$*s \supseteq r$

Pass one

$\text{pts}(p) = \{a\}$

$\text{pts}(q) = \{b\}$

$\text{pts}(r) = \{c\}$

$\text{pts}(s) = \{a\}$

$\text{pts}(t) = \{b\}$

$\text{pts}(a) = \{b, c\}$

$\text{pts}(b) = \{ \}$

$\text{pts}(c) = \{ \}$

ANOTHER EXAMPLE

ANDERSEN'S ANALYSIS

A FLOW-INSENSITIVE ALGORITHM

Each statement adds a constraint over the points-to sets

End up with a (solvable) system of constraints

Constraint type	Assignment	Constraint	Meaning
Base	$a = \&b$	$a \supseteq \{b\}$	$\text{loc}(b) \in \text{pts}(a)$
Simple	$a = b$	$a \supseteq b$	$\text{pts}(a) \supseteq \text{pts}(b)$
Complex	$a = *b$	$a \supseteq *b$	$\forall v \in \text{pts}(b). \text{pts}(a) \supseteq \text{pts}(v)$
Complex	$*a = b$	$*a \supseteq b$	$\forall v \in \text{pts}(a). \text{pts}(v) \supseteq \text{pts}(b)$

Program	Constraints	Pass one	Pass two
$p = \&a$	$p \supseteq \{a\}$	$\text{pts}(p) = \{a\}$	$\text{pts}(p) = \{a\}$
$q = \&b$	$q \supseteq \{b\}$	$\text{pts}(q) = \{b\}$	$\text{pts}(q) = \{b\}$
$*p = q;$	$*p \supseteq q$	$\text{pts}(r) = \{c\}$	$\text{pts}(r) = \{c\}$
$r = \&c;$	$r \supseteq \{c\}$	$\text{pts}(s) = \{a\}$	$\text{pts}(s) = \{a\}$
$s = p;$	$s \supseteq p$	$\text{pts}(t) = \{b\}$	$\text{pts}(t) = \{b, c\}$
$t = *p;$	$t \supseteq *p$	$\text{pts}(a) = \{b, c\}$	$\text{pts}(a) = \{b, c\}$
$*s = r;$	$*s \supseteq r$	$\text{pts}(b) = \{ \}$	$\text{pts}(b) = \{ \}$
		$\text{pts}(c) = \{ \}$	$\text{pts}(c) = \{ \}$

SOLVING CONSTRAINTS

ANDERSEN'S ANALYSIS

Graph closure on the subset relation

Assgmt.	Constraint	Meaning	Edge
$a = \&b$	$a \supseteq \{b\}$	$b \in \text{pts}(a)$	no edge
$a = b$	$a \supseteq b$	$\text{pts}(a) \supseteq \text{pts}(b)$	$b \rightarrow a$
$a = *b$	$a \supseteq *b$	$\forall v \in \text{pts}(b). \text{pts}(a) \supseteq \text{pts}(v)$	no edge
$*a = b$	$*a \supseteq b$	$\forall v \in \text{pts}(a). \text{pts}(v) \supseteq \text{pts}(b)$	no edge

OVERHEAD

ANDERSEN'S ANALYSIS

WORST CASE: CUBIC TIME

That's not great!

OPTIMIZATION: CYCLE ELIMINATION

Detect and collapse SCCs in the points-to relation

AN ALTERNATIVE APPROACH

STEENSGARD'S ANALYSIS

AIM FOR NEAR-LINEAR-TIME POINTS-TO ANALYSIS

Going to require us to reduce our search-space somewhat

INTUITION: EQUALITY CONSTRAINTS

Do away with the notion of subsets

EQUALITY CONSTRAINTS

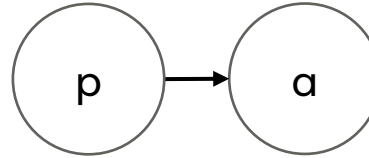
STEENSGARD'S ANALYSIS

Constraint type	Assignment	Constraint	Meaning
Base	$a = \&b$	$a \supseteq \{b\}$	$\text{loc}(b) \in \text{pts}(a)$
Simple	$a = b$	$a = b$	$\text{pts}(a) = \text{pts}(b)$
Complex	$a = *b$	$a = *b$	$\forall v \in \text{pts}(b). \text{pts}(a) = \text{pts}(v)$
Complex	$*a = b$	$*a = b$	$\forall v \in \text{pts}(a). \text{pts}(v) = \text{pts}(b)$

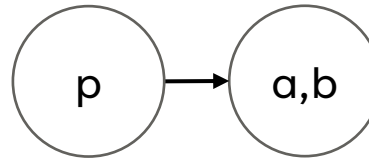
EQUALITY CONSTRAINTS

STEENSGARD'S ANALYSIS

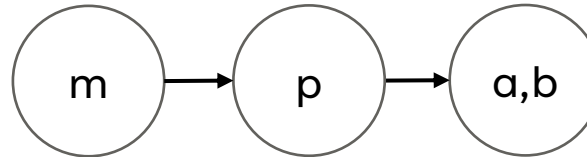
$p = \&a$



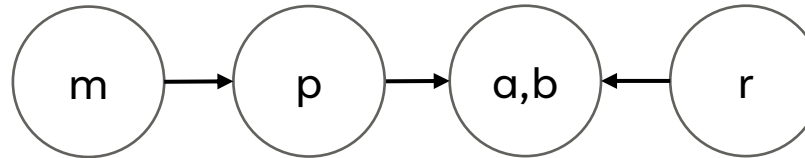
$p = \&b$



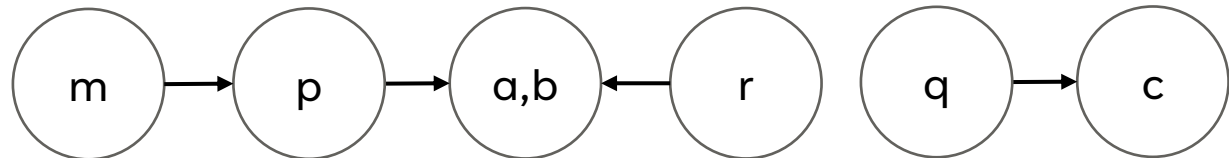
$m = \&p$



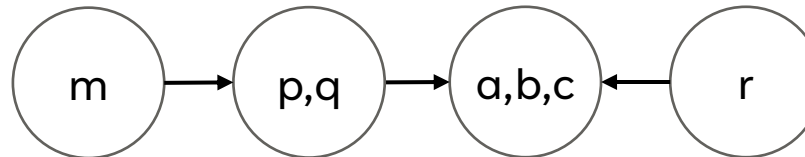
$r = *m$



$q = \&c$



$m = \&q$

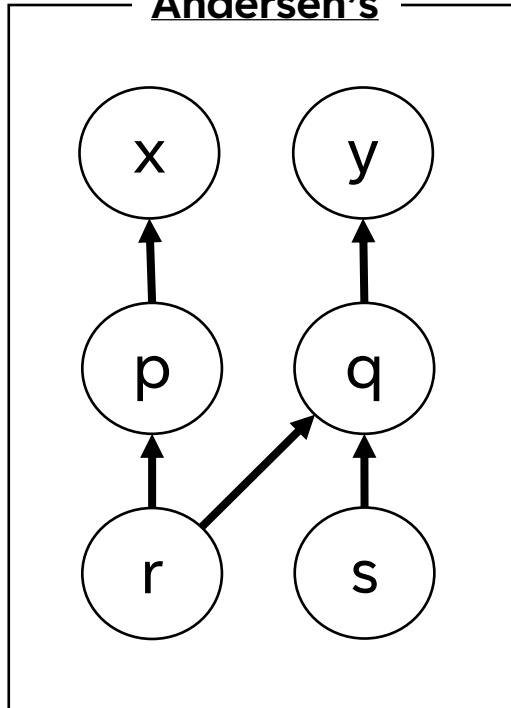


COMPARING ANALYSIS RESULTS

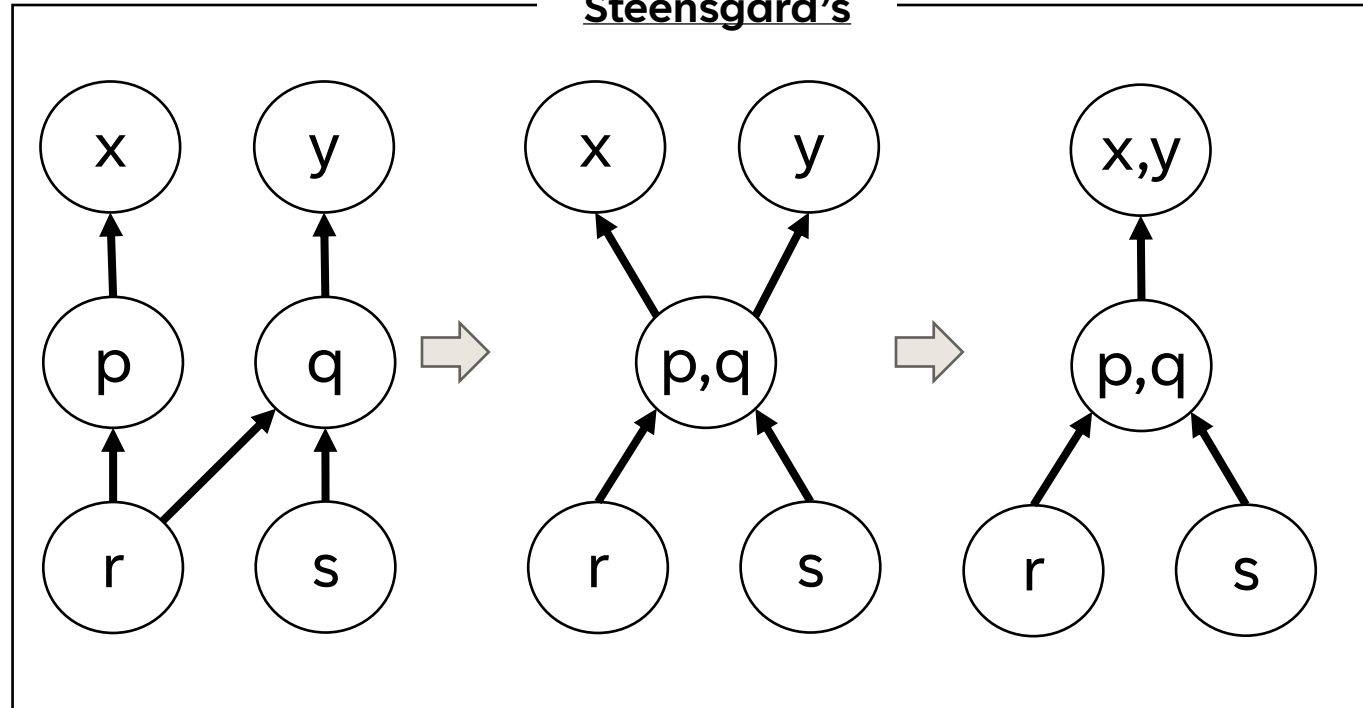
STEENSGARD'S ANALYSIS

1 : $p := \&x$
2 : $r := \&p$
3 : $q := \&y$
4 : $s := \&q$
5 : $r := s$

Andersen's



Steensgard's



WRAP-UP

