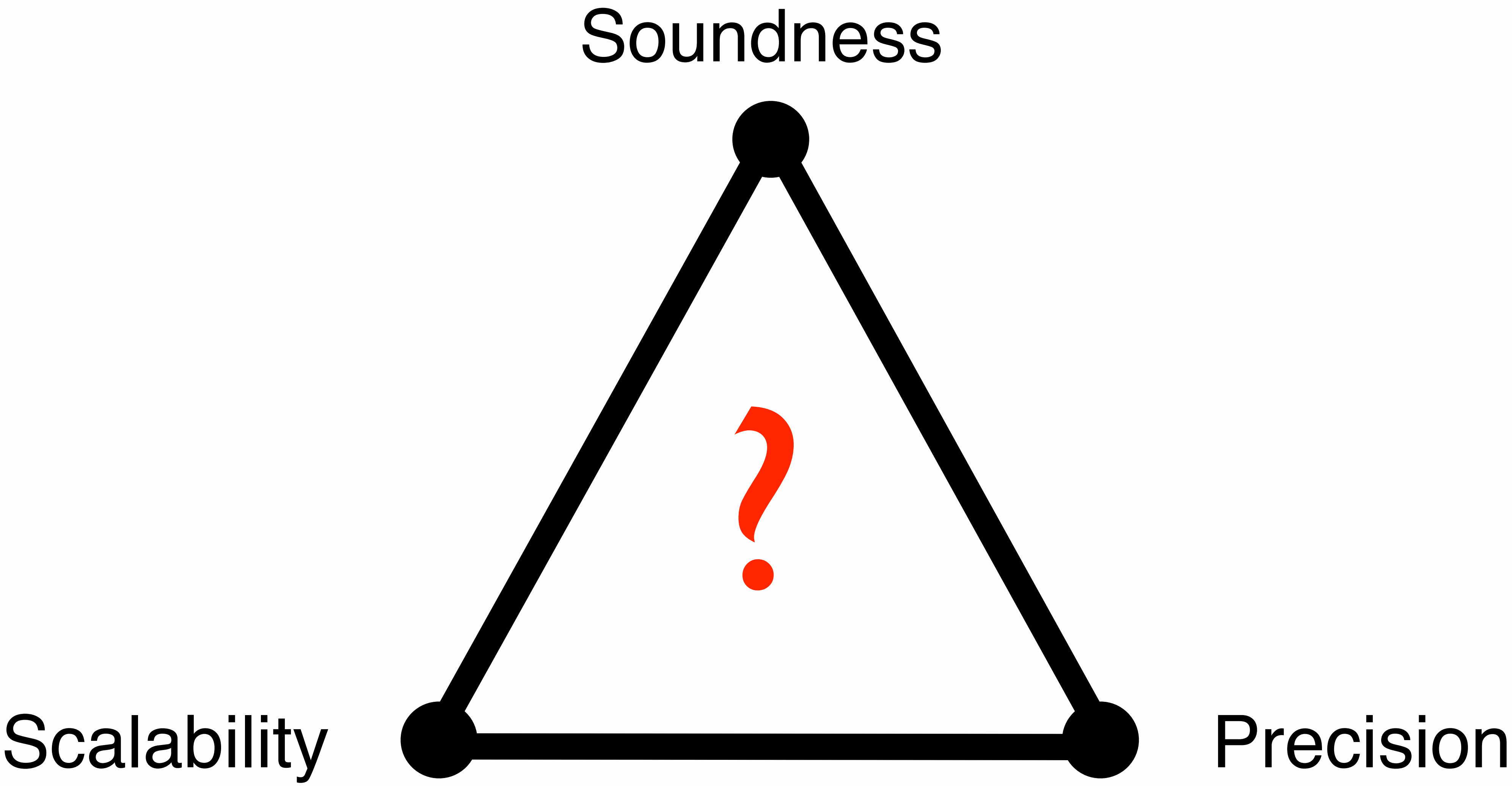
Scalability X Precision X Soundness by Sparsity and Selectivity

Kwangkeun Yi
Seoul National University, Korea

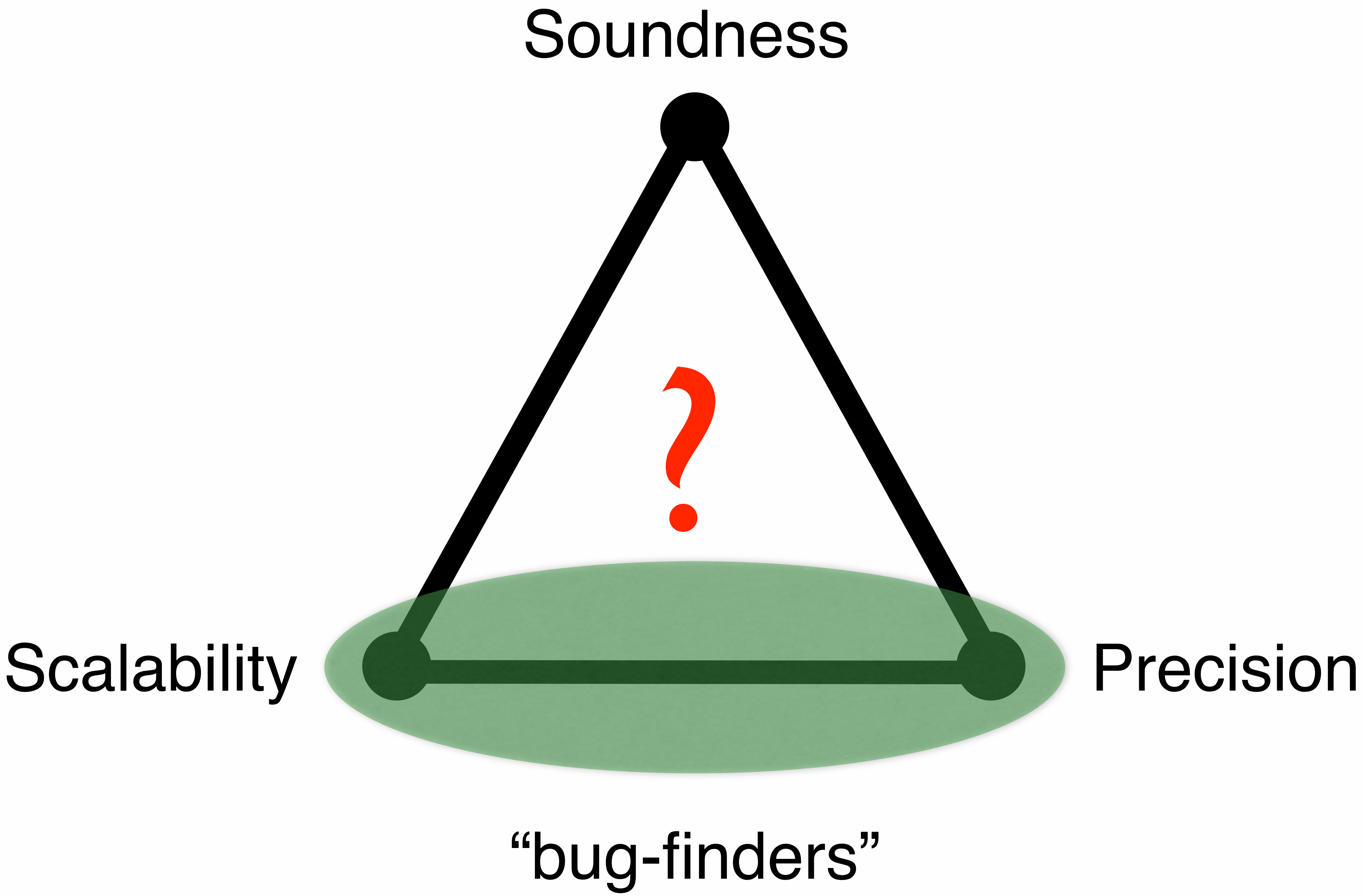
6/27/2014@ENS

co-work with
Hakjoo Oh, Wonchan Lee, Woosuk Lee, Kihong Heo, Hongseok Yang, Jihoon Kang

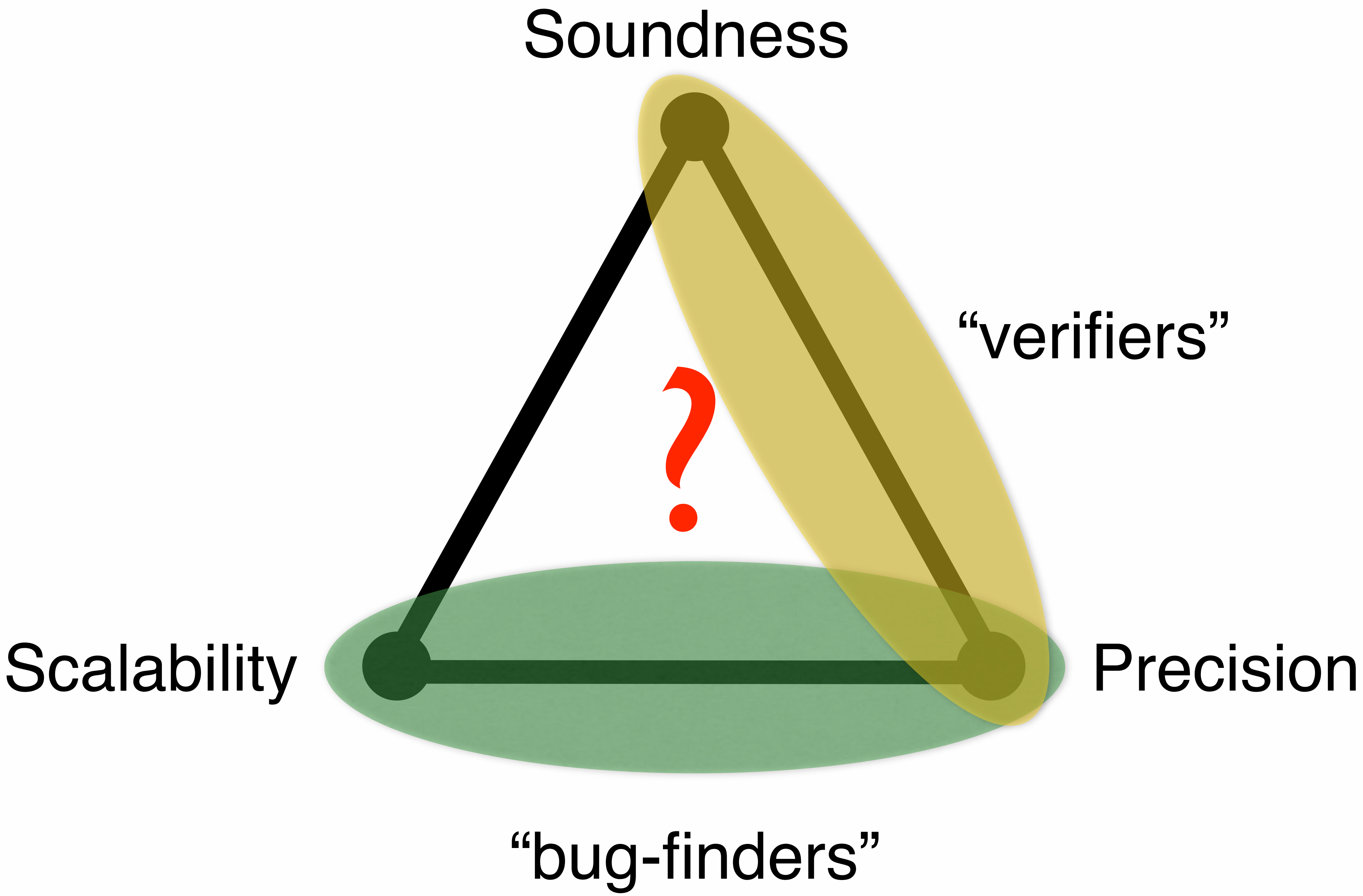
Challenge in Static Analysis



Challenge in Static Analysis



Challenge in Static Analysis



Soundness

"verifiers"

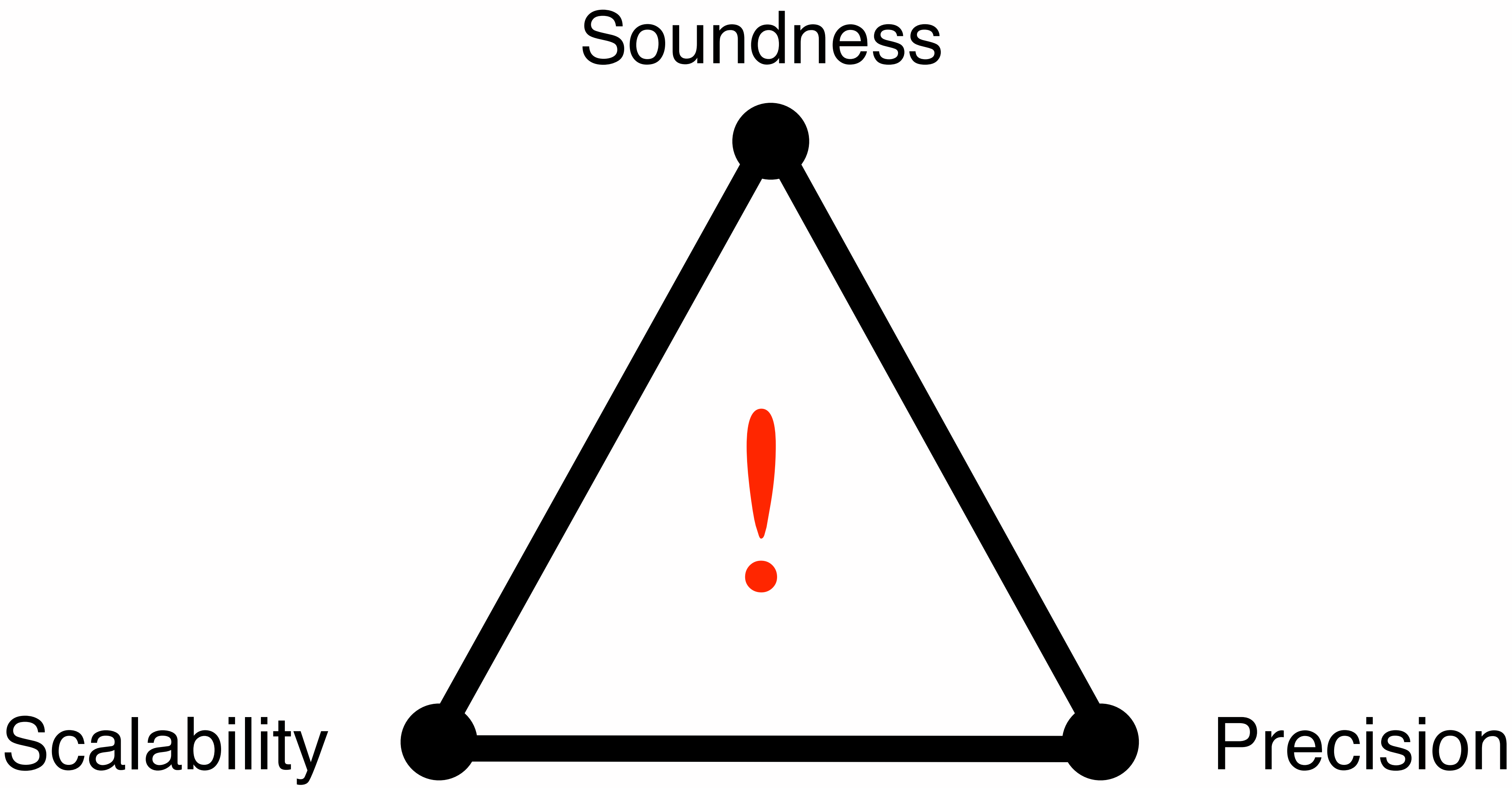
?

Scalability

Precision

"bug-finders"

Our Long-term Goal



Soundness

Scalability

Precision

!

General Sparse Analysis Framework

[PLDI'12]

Selective X-Sensitivity Approach

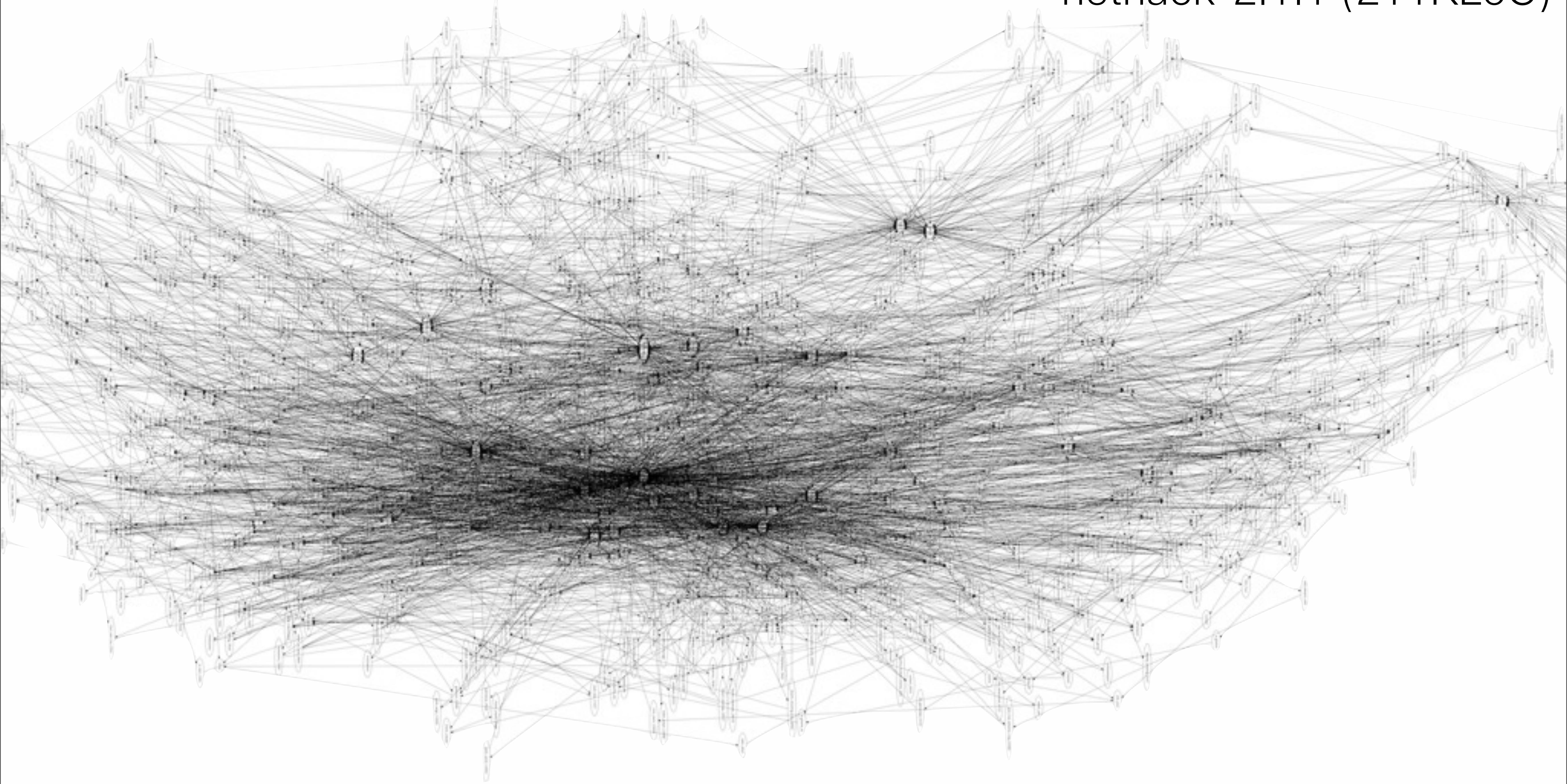
[PLDI'14]

Our story

- In 2007, we commercialized Sparrow The Early Bird
  - memory-bug-finding tool for full C
  - designed in abstract interpretation framework
  - sound in design, unsound yet scalable in reality (non-global)
- Realistic workbench available
  - “let’s try to achieve sound, precise, yet scalable global version”

The Challenge in Reality

nethack-2.1.1 (211KLoC)

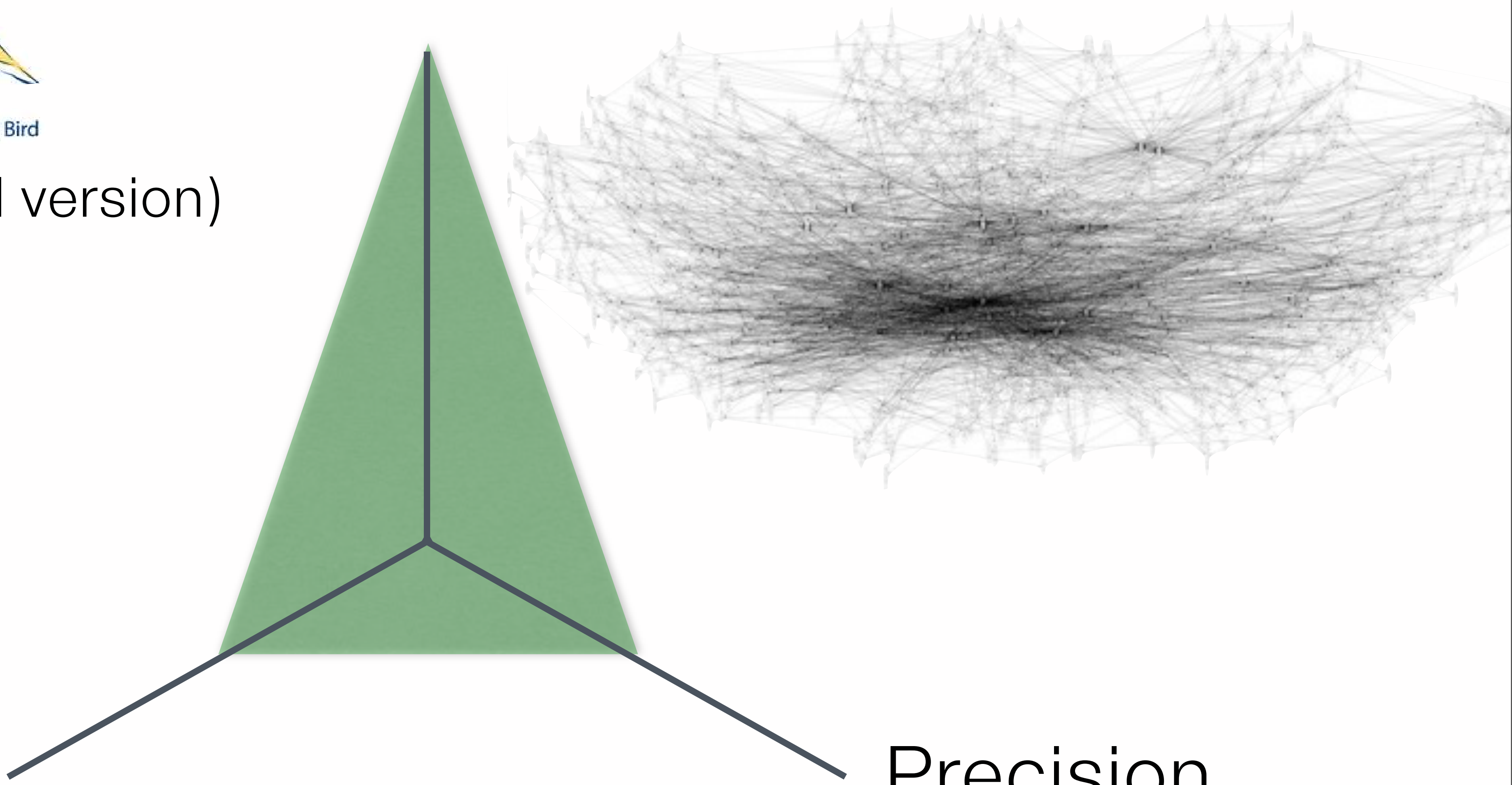


The Challenge in Reality

Soundness

Sparrow
The Early Bird

(2007, sound-&-global version)



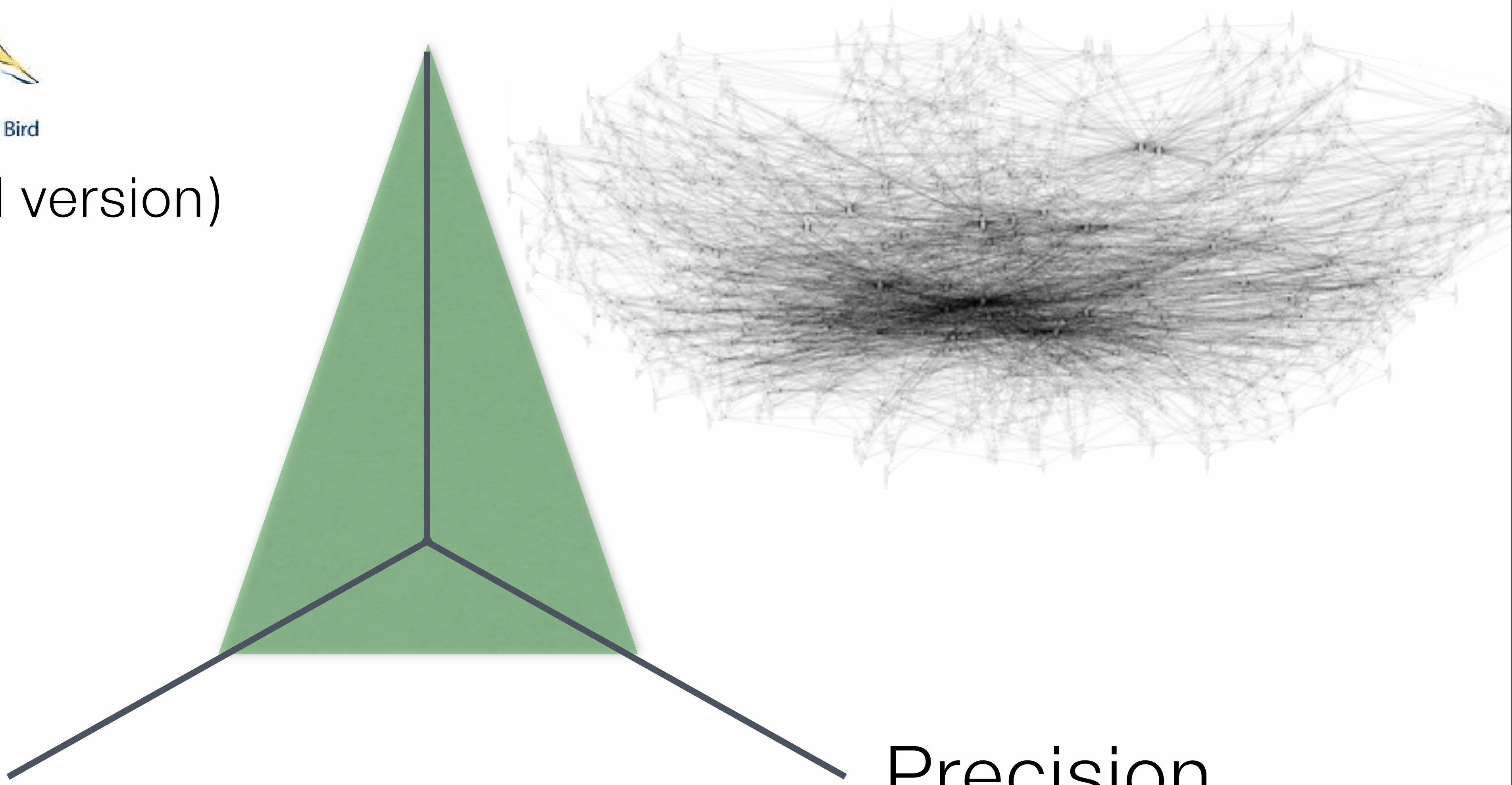
Scalability

Precision

The Challenge in Reality

Sparrow
The Early Bird

(2007, sound-&-global version)



Soundness

35KLoC

Scalability

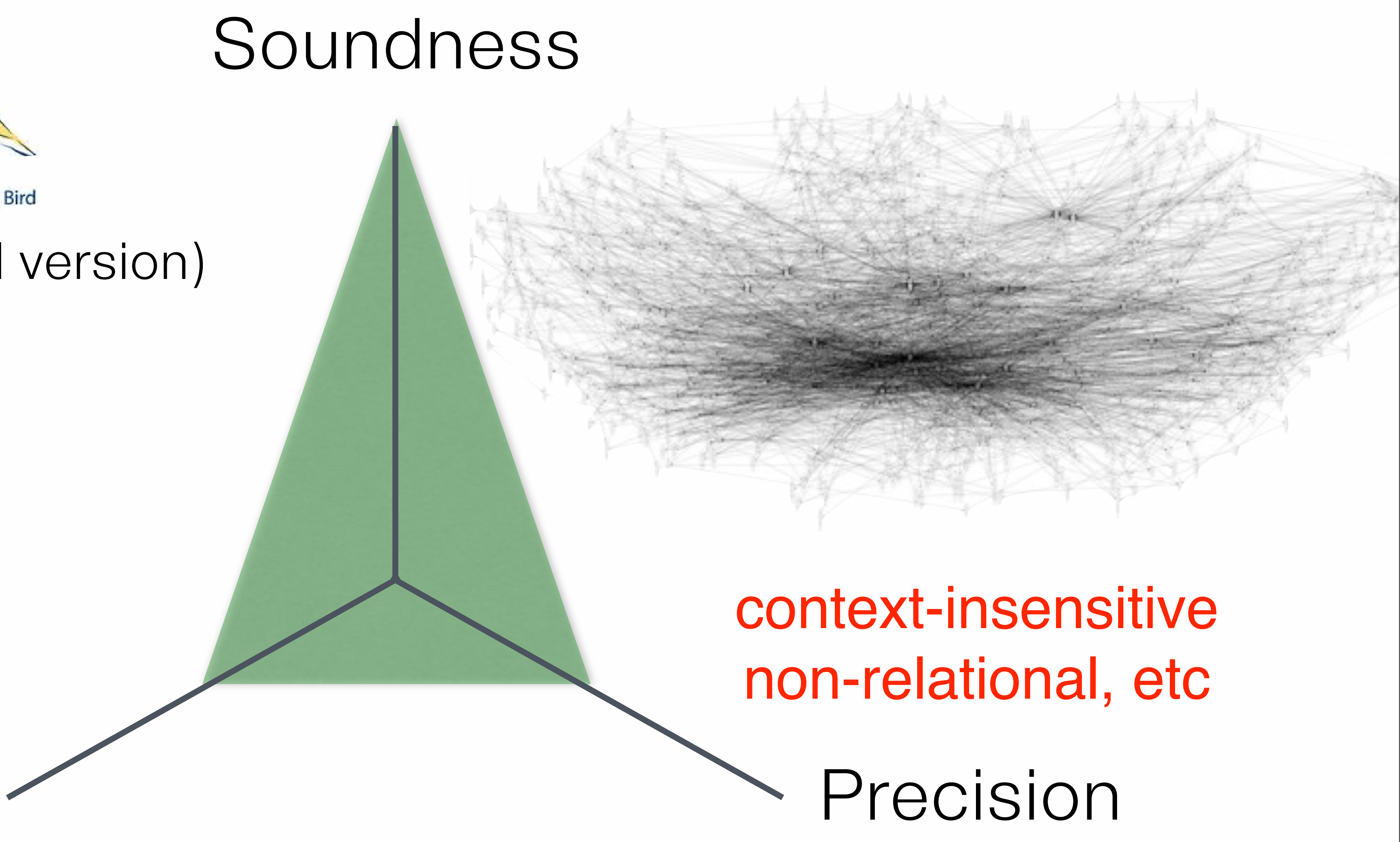
Precision

The Challenge in Reality

Sparrow

The Early Bird

(2007, sound-&-global version)



35KLoC

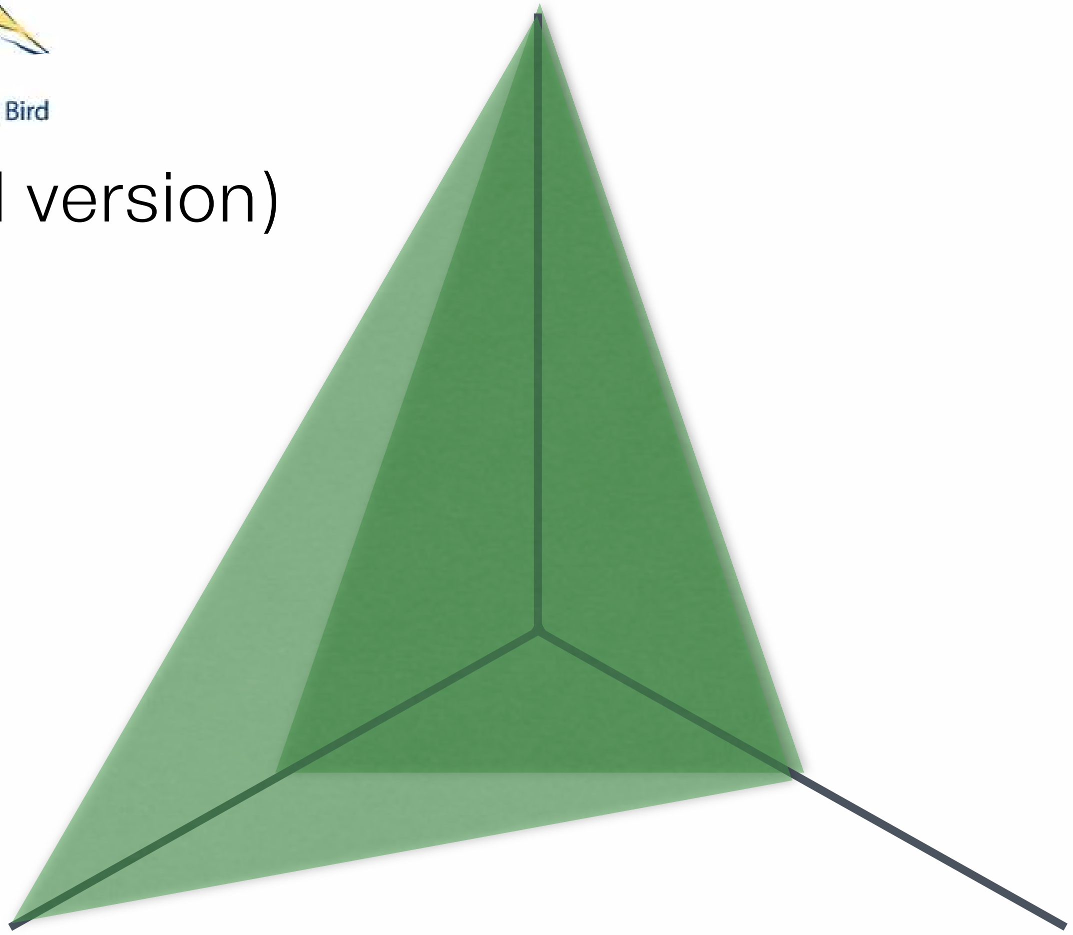
context-insensitive
non-relational, etc

Scalability: time-mem sparsity

Sparrow

The Early Bird

(2012, sound-&-global version)



Soundness

Scalability

General Sparse Analysis Framework

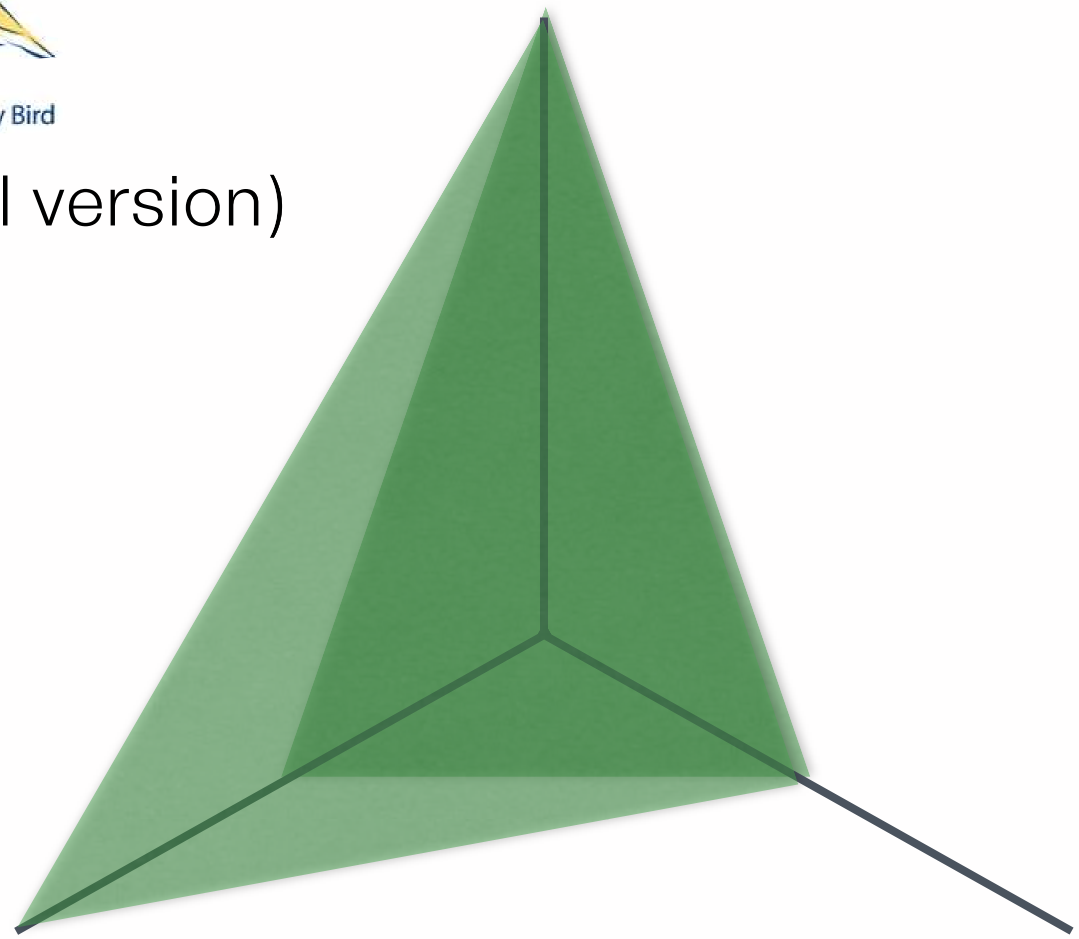
[PLDI’12]

Precision

Scalability: time-mem sparsity

Sparrow
The Early Bird

(2012, sound-&-global version)



1 Million LoC

General Sparse Analysis Framework

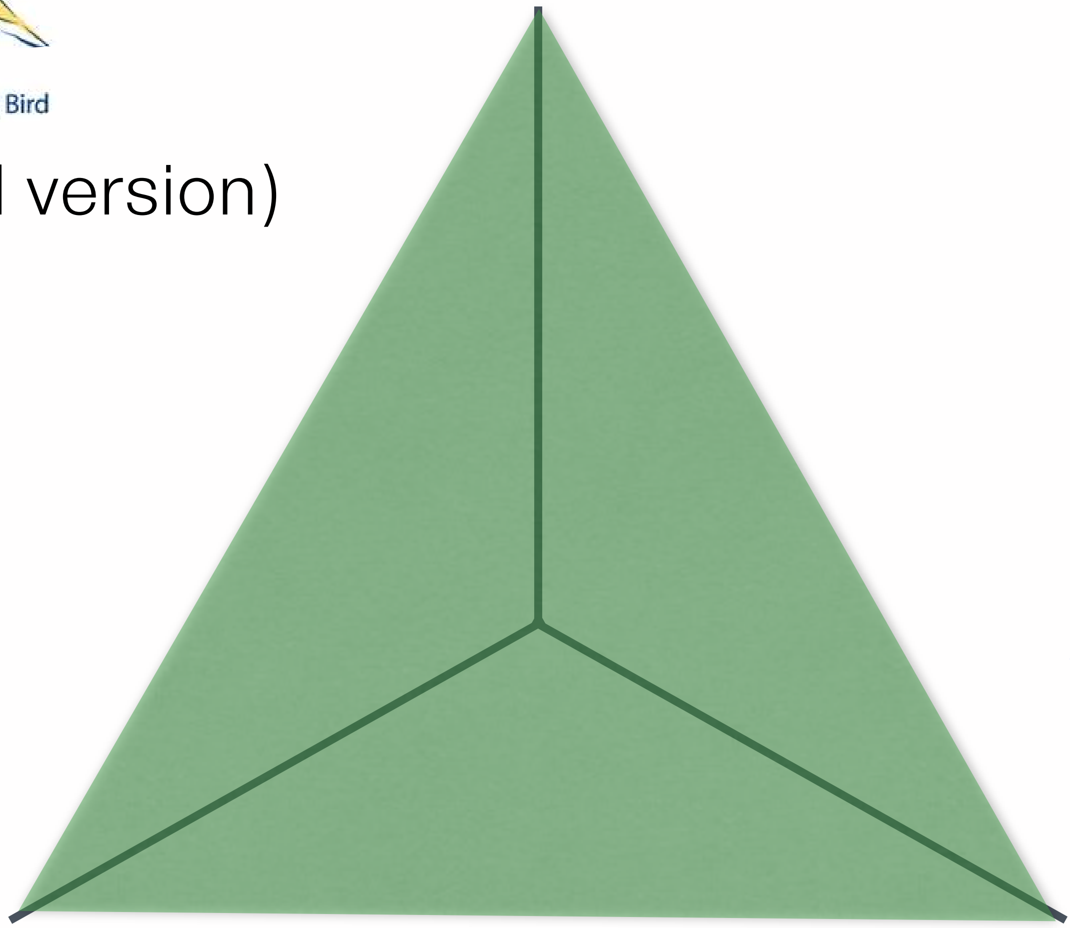
[PLDI'12]

Precision: selective sensitivity

Soundness

Sparrow
The Early Bird

(2014, sound-&-global version)



1 Million LoC

context-sensitivity

Scalability

Precision

General Sparse Analysis Framework
[PLDI'12]

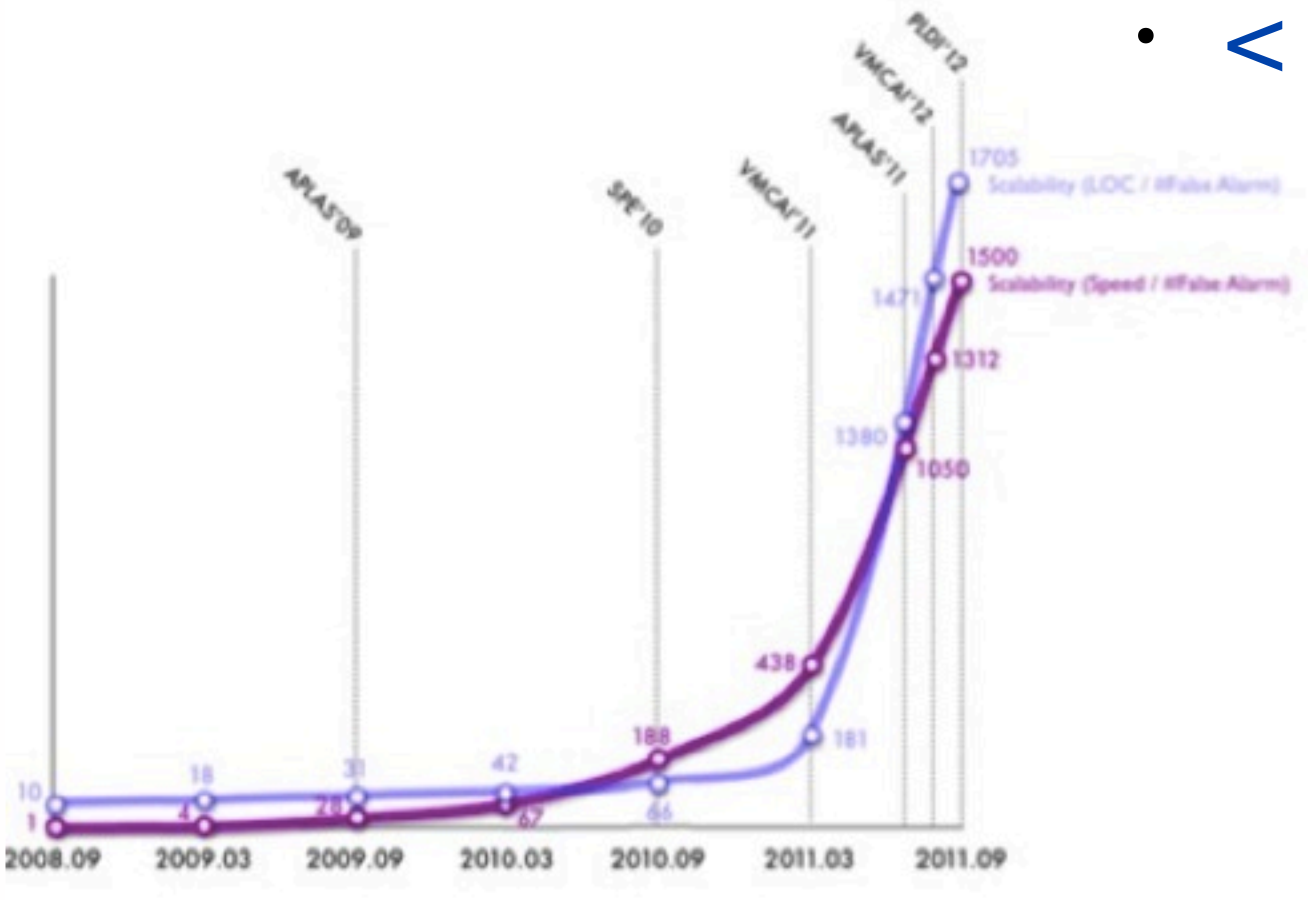
Selective X-Sensitivity Approach
[PLDI'14]

Our Scalability Improvement

Sparrow
The Early Bird

sound-&-global version

- < 1.4M in 10hrs (intrvls)
- < 0.14M in 20hrs (octgns)



Our Precision Improvement

Sparrow
The Early Bird

sound-&-global version

24% / 28%

reduction of false alarms   increase of analysis time

vs. context-insensitivity

Contents

- Sparrow System
- Scalability by Sparsity
- Precision by Selectivity

Sparrow

The Early Bird

- Designed in the *abstract interpretation* framework
- To find memory safety violations in C
  - buffer-overrun, memory leak, null deref., etc.
  - flow-sensitive values analysis for int & ptrs (static + dynamic)
  - for the full set of C

Abstract Semantics

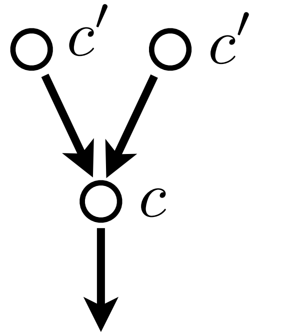
- One abstract state $\in \hat{\mathbb{S}}$ that subsumes all reachable states at each program point

$$\begin{aligned} [\![\hat{P}]\!] \in \mathbb{C} \to \hat{\mathbb{S}} &= \mathit{fix} \hat{F} \\ \hat{\mathbb{S}} &= \hat{\mathbb{L}} \to \hat{\mathbb{V}} \end{aligned}$$

$$\begin{aligned} \hat{\mathbb{L}} &= Var + AllocSite + AllocSite \times FieldName \\ \hat{\mathbb{V}} &= \hat{\mathbb{Z}} \times 2^{\hat{\mathbb{L}}} \times 2^{AllocSite \times \hat{\mathbb{Z}} \times \hat{\mathbb{Z}}} \times 2^{AllocSite \times 2^{FieldName}} \\ \hat{\mathbb{Z}} &= \{[l,u] \mid l,u \in \mathbb{Z} \cup \{-\infty,+\infty\} \wedge l \le u\} \cup \{\bot\} \end{aligned}$$

- Abstract semantic function

$$\begin{aligned} \hat{F} &\in (\mathbb{C} \to \hat{\mathbb{S}}) \to (\mathbb{C} \to \hat{\mathbb{S}}) \\ \hat{F}(\hat{X}) &= \lambda c \in \mathbb{C}. \hat{f}_c\Big(\bigsqcup_{c' \hookrightarrow c} \hat{X}(c')\Big) \end{aligned}$$



$\hat{f}_c \in \hat{\mathbb{S}} \to \hat{\mathbb{S}}$: abstract semantics at point c

Computing $\mathit{fix}\hat{F} = \bigsqcup_{i\in\mathbb{N}} \hat{F}^i(\hat{\bot})$

$$\hat{F}(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c(\bigsqcup_{c' \hookrightarrow c} \hat{X}(c')).$$

```
$\hat{X}, \hat{X}' \in \mathbb{C} \to \hat{\mathbb{S}}$
$\hat{f}_c \in \hat{\mathbb{S}} \to \hat{\mathbb{S}}$

$\hat{X} := \hat{X}' := \lambda c.\bot$
**repeat**
    $\hat{X}' := \hat{X}$
    **for all** $c \in \mathbb{C}$ **do**
        $\hat{X}(c) := \hat{f}_c(\bigsqcup_{c' \hookrightarrow c} X(c'))$
**until** $\hat{X} \sqsubseteq \hat{X}'$
```

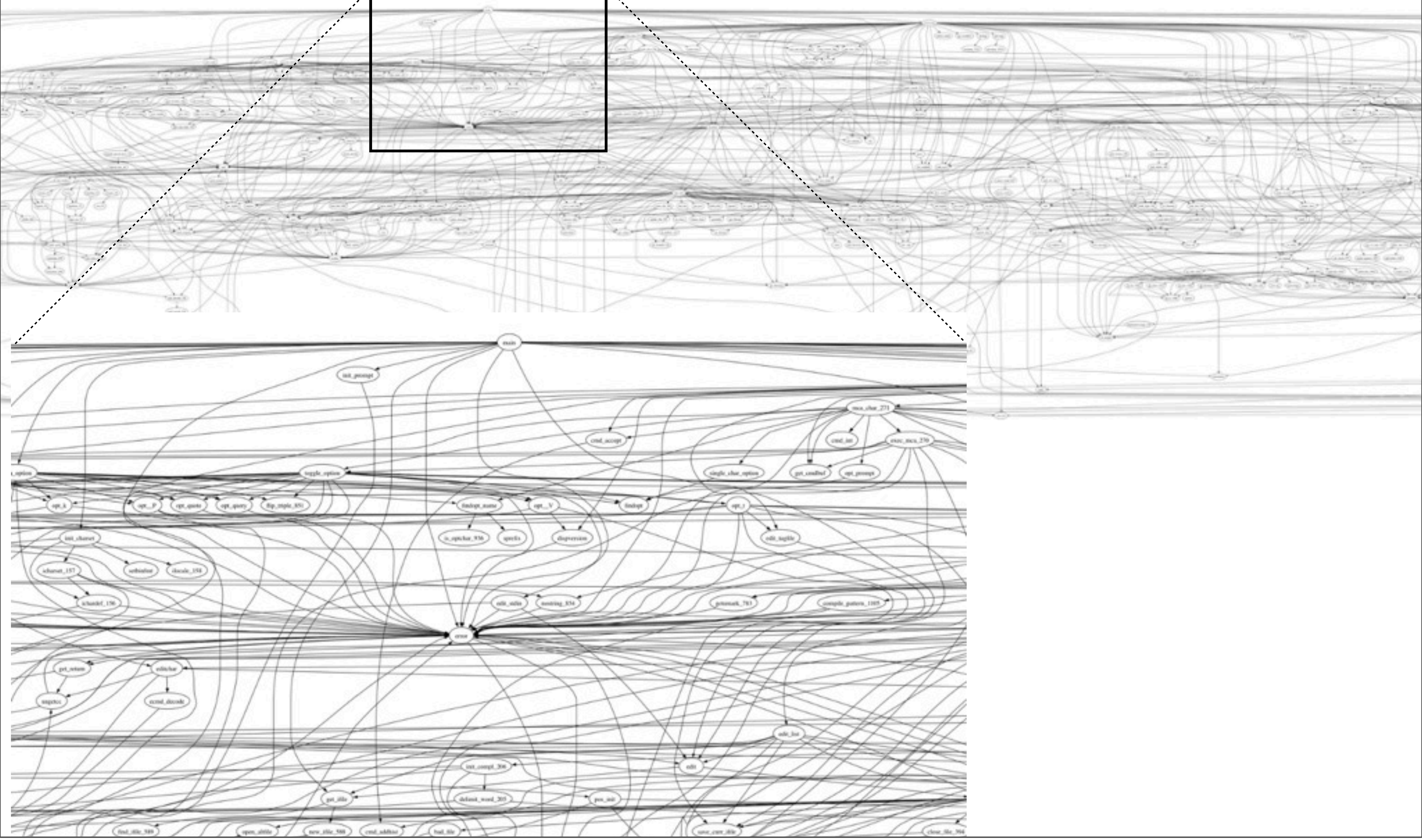
Naive fixpoint algorithm

```
$W \in \mathit{Worklist} = 2^{\mathbb{C}}$
$\hat{X} \in \mathbb{C} \to \hat{\mathbb{S}}$
$\hat{f}_c \in \hat{\mathbb{S}} \to \hat{\mathbb{S}}$

$W := \mathbb{C}$
$\hat{X} := \lambda c.\bot$
**repeat**
    $c :=$ choose$(W)$
    $\hat{s} := \hat{f}_c(\bigsqcup_{c' \hookrightarrow c} X(c'))$
    **if** $\hat{s} \not\sqsubseteq \hat{X}(c)$
        $W := W \cup \{c' \in \mathbb{C} \mid c \hookrightarrow c'\}$
        $\hat{X}(c) := \hat{X}(c) \sqcup \hat{s}$
**until** $W = \emptyset$
```

Worklist algorithm

The Algorithms Too Weak To Scale

less-382 (23,822 LoC)

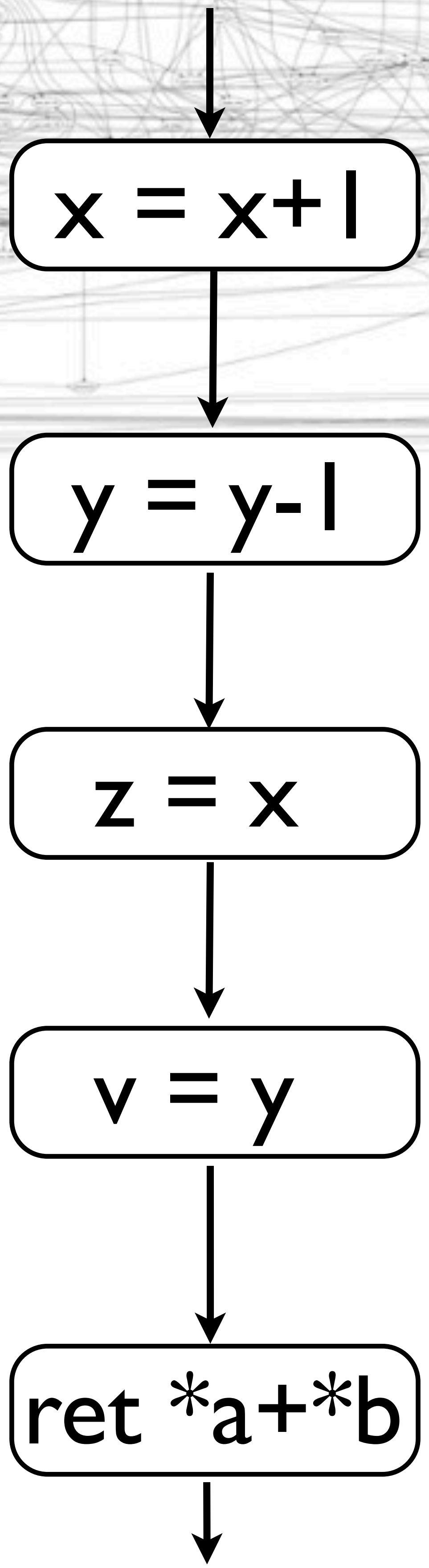


Improving Scalability

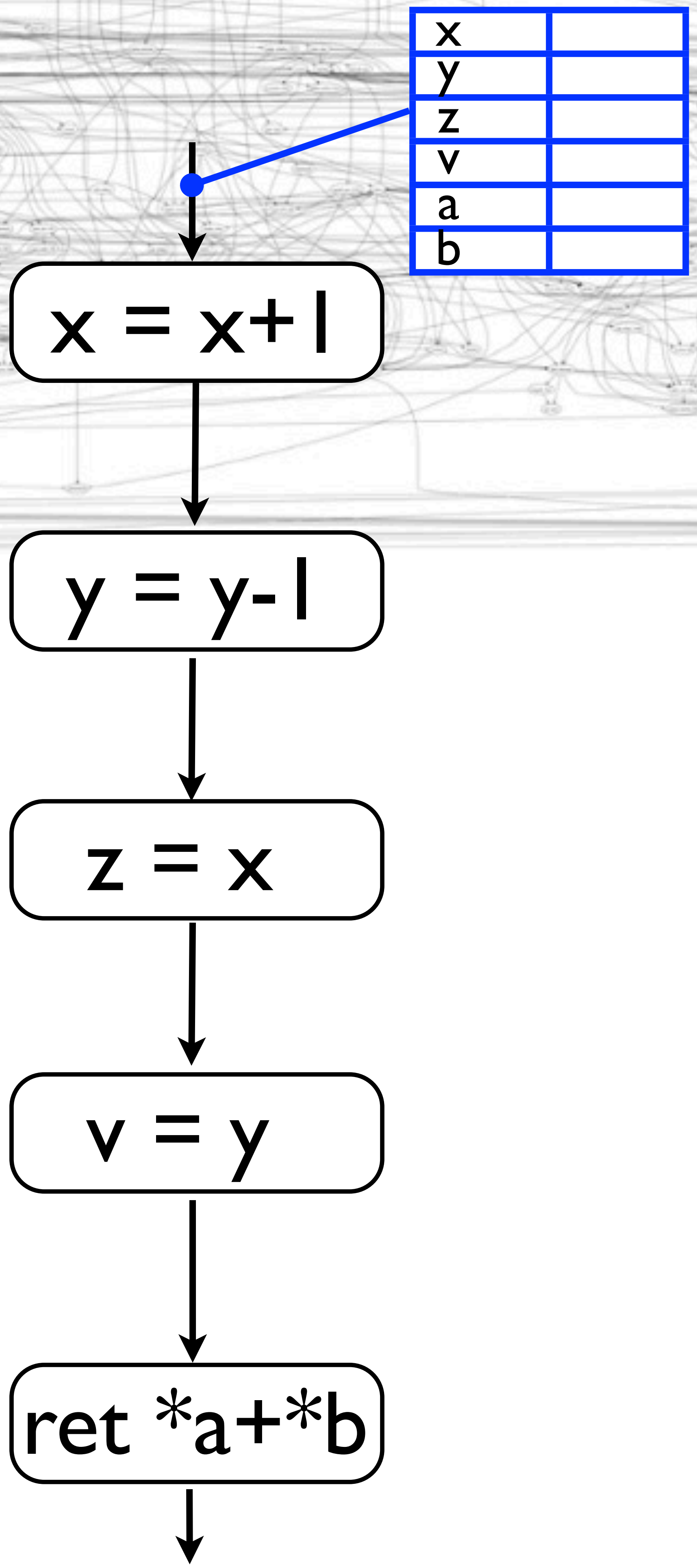
Key Idea: Localization

“Right Part at Right Moment”

Spatial & Temporal Localizations



Spatial & Temporal Localizations



x	
y	
z	
v	
a	
b	

x = x+1

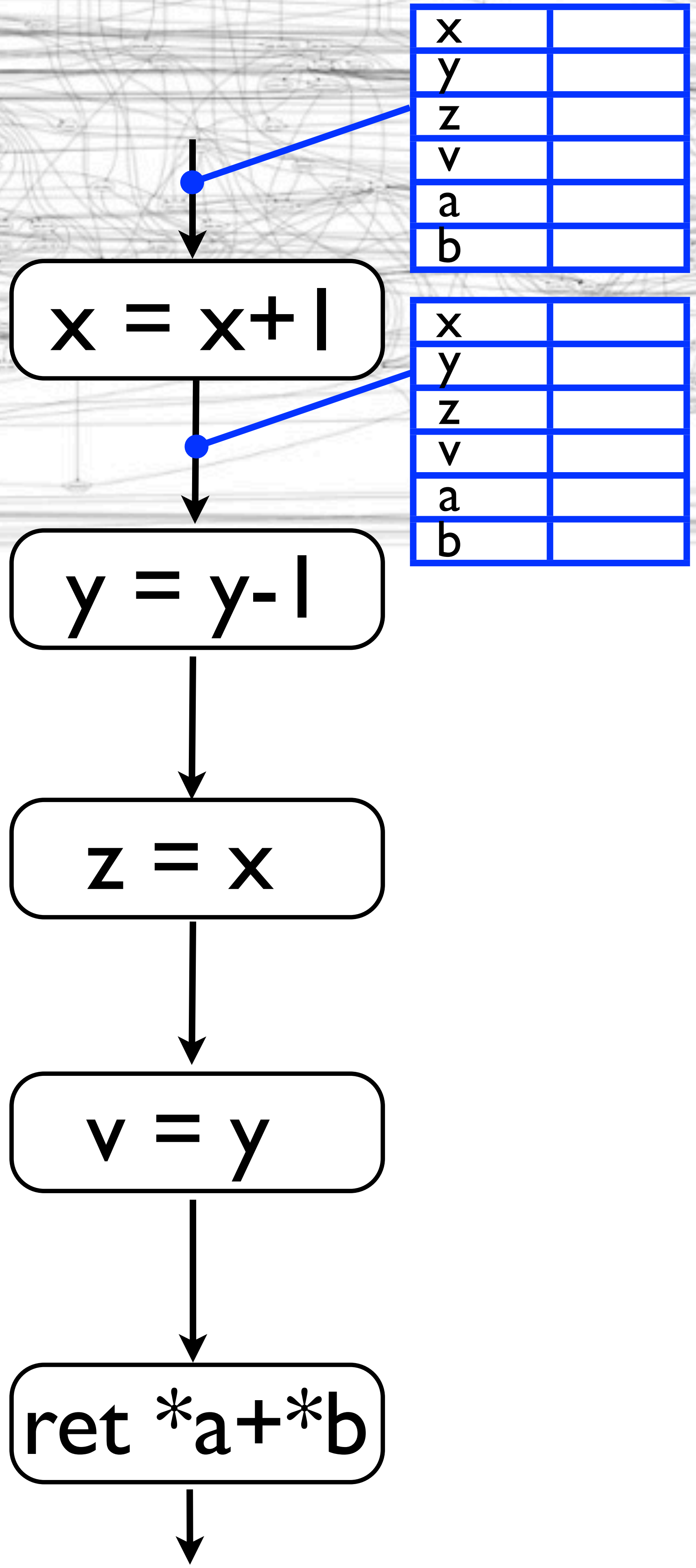
y = y-1

z = x

v = y

ret *a+*b

Spatial & Temporal Localizations



x	
y	
z	
v	
a	
b	

x = x+1

x	
y	
z	
v	
a	
b	

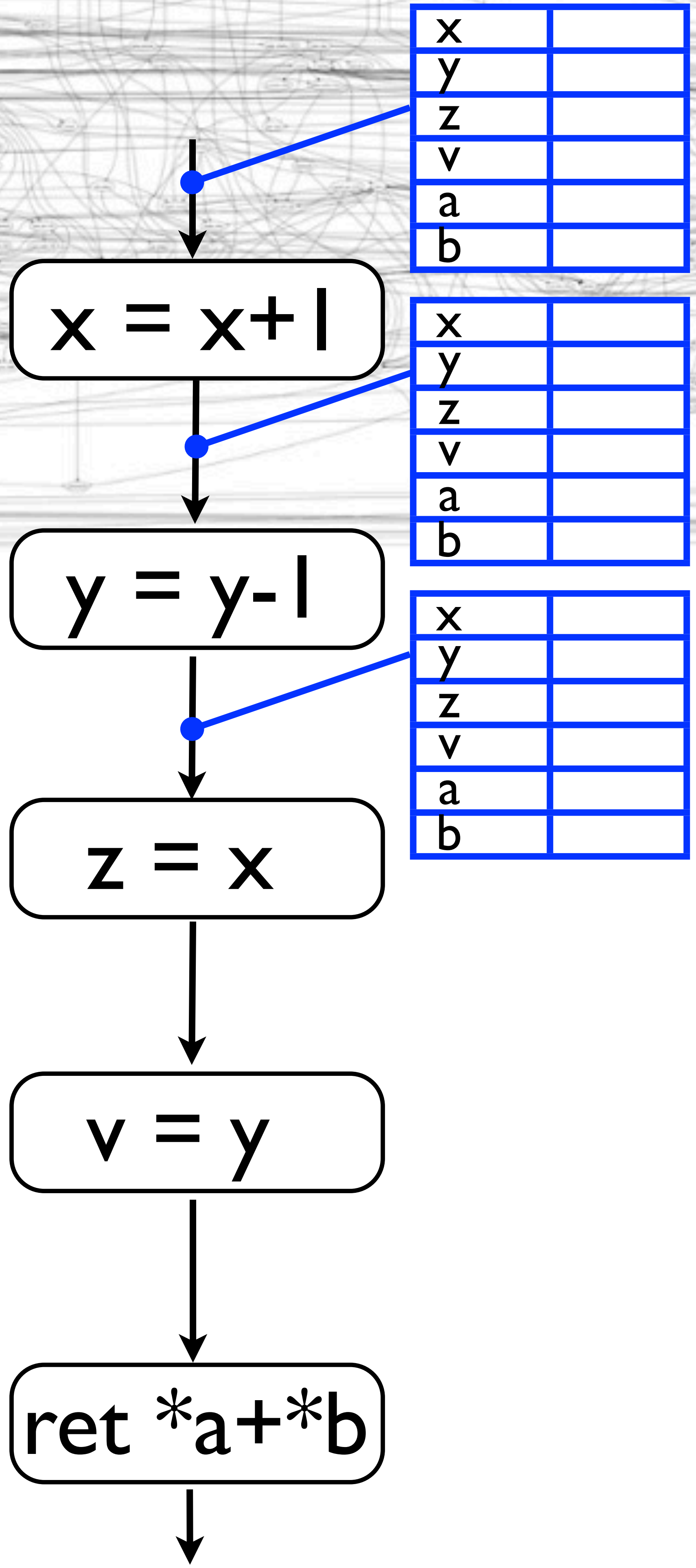
y = y-1

z = x

v = y

ret *a+*b

Spatial & Temporal Localizations



x = x+1

y = y-1

z = x

v = y

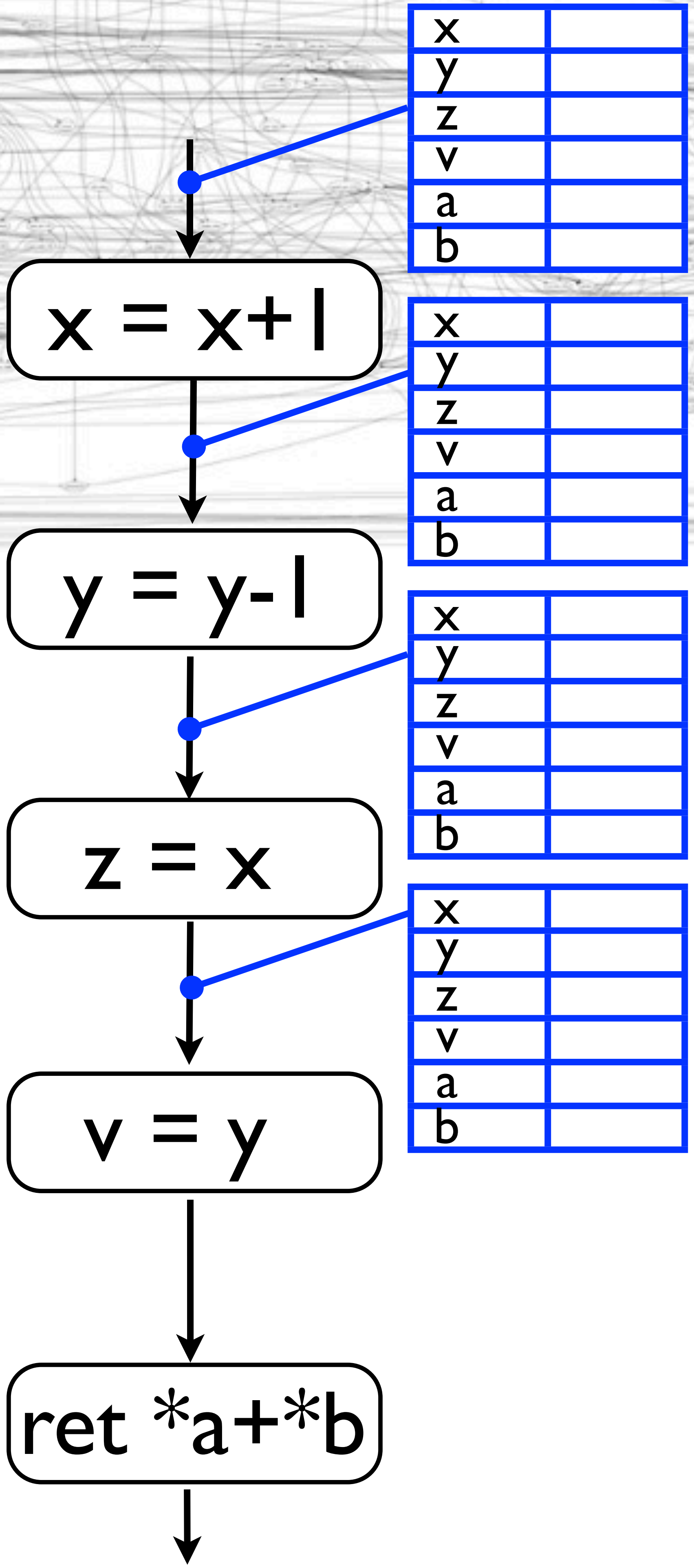
ret *a+*b

x	
y	
z	
v	
a	
b	

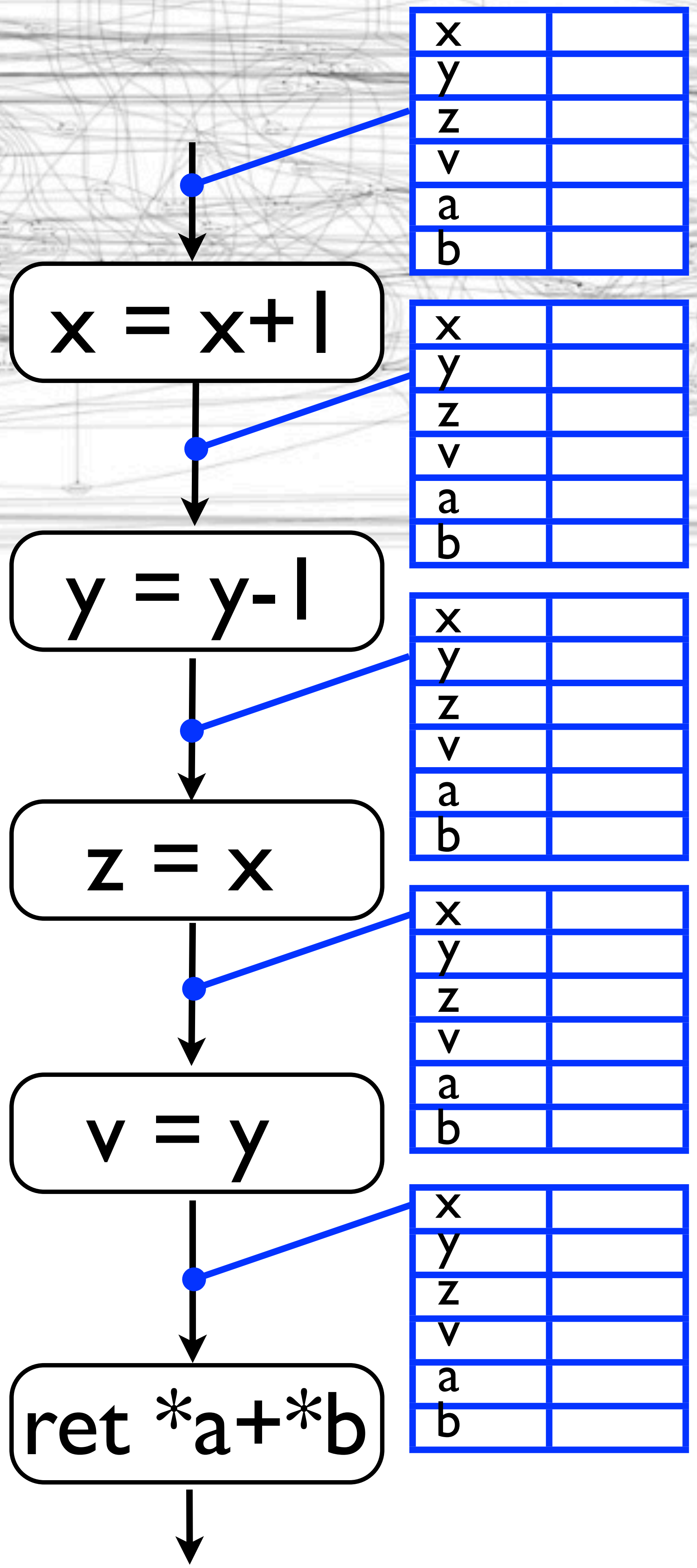
x	
y	
z	
v	
a	
b	

x	
y	
z	
v	
a	
b	

Spatial & Temporal Localizations



Spatial & Temporal Localizations



x = x+1

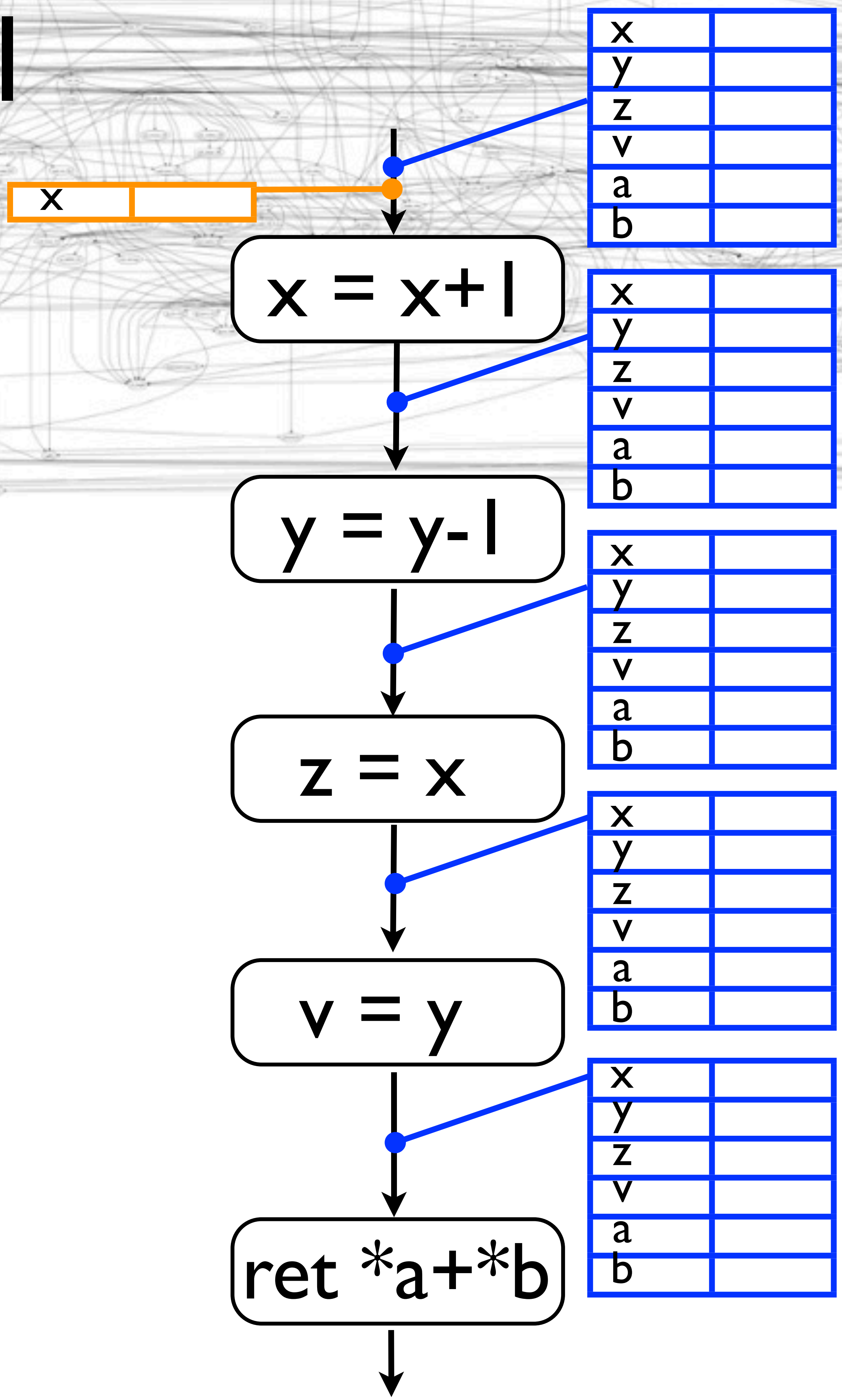
y = y-1

z = x

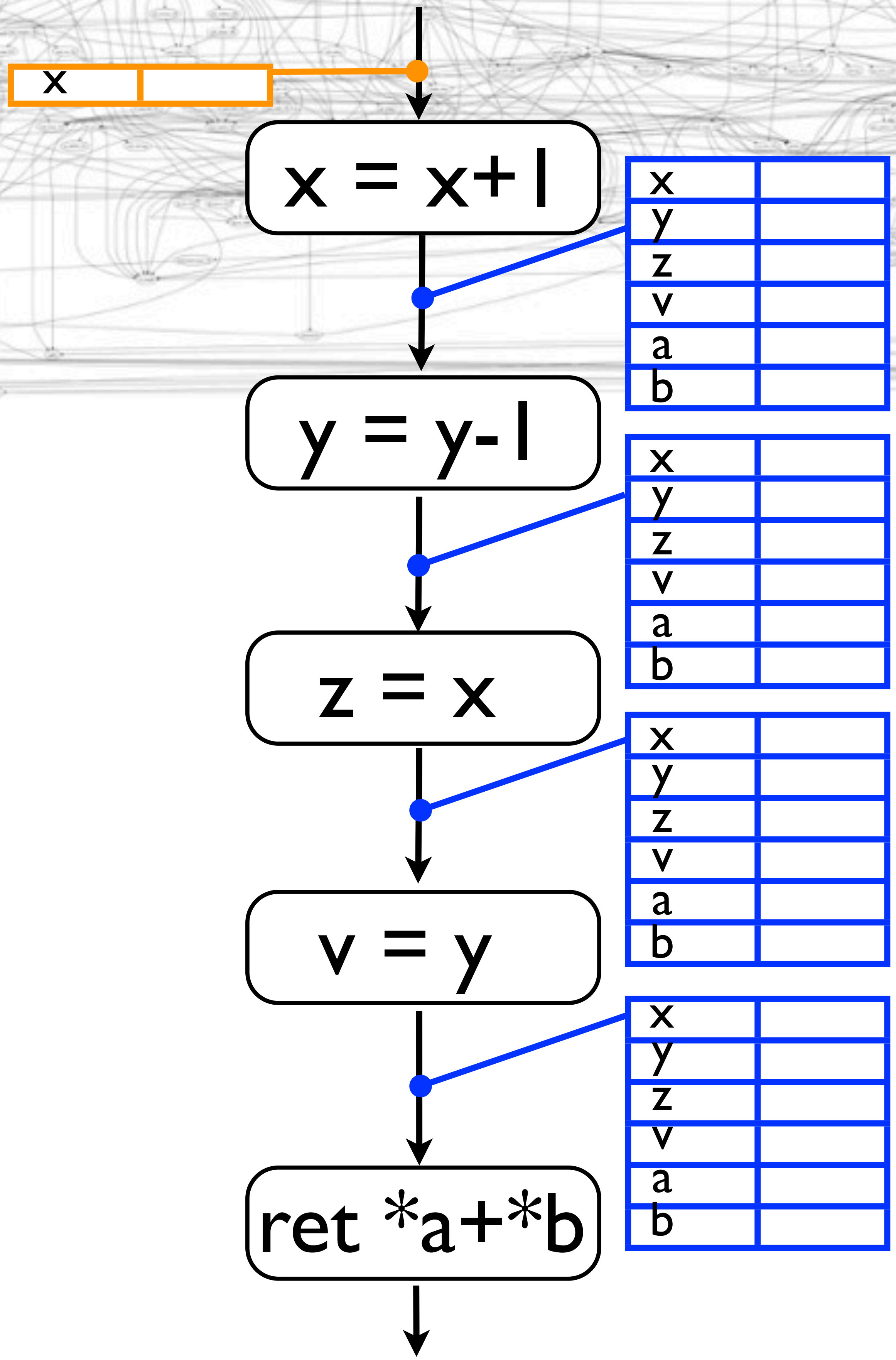
v = y

ret *a+*b

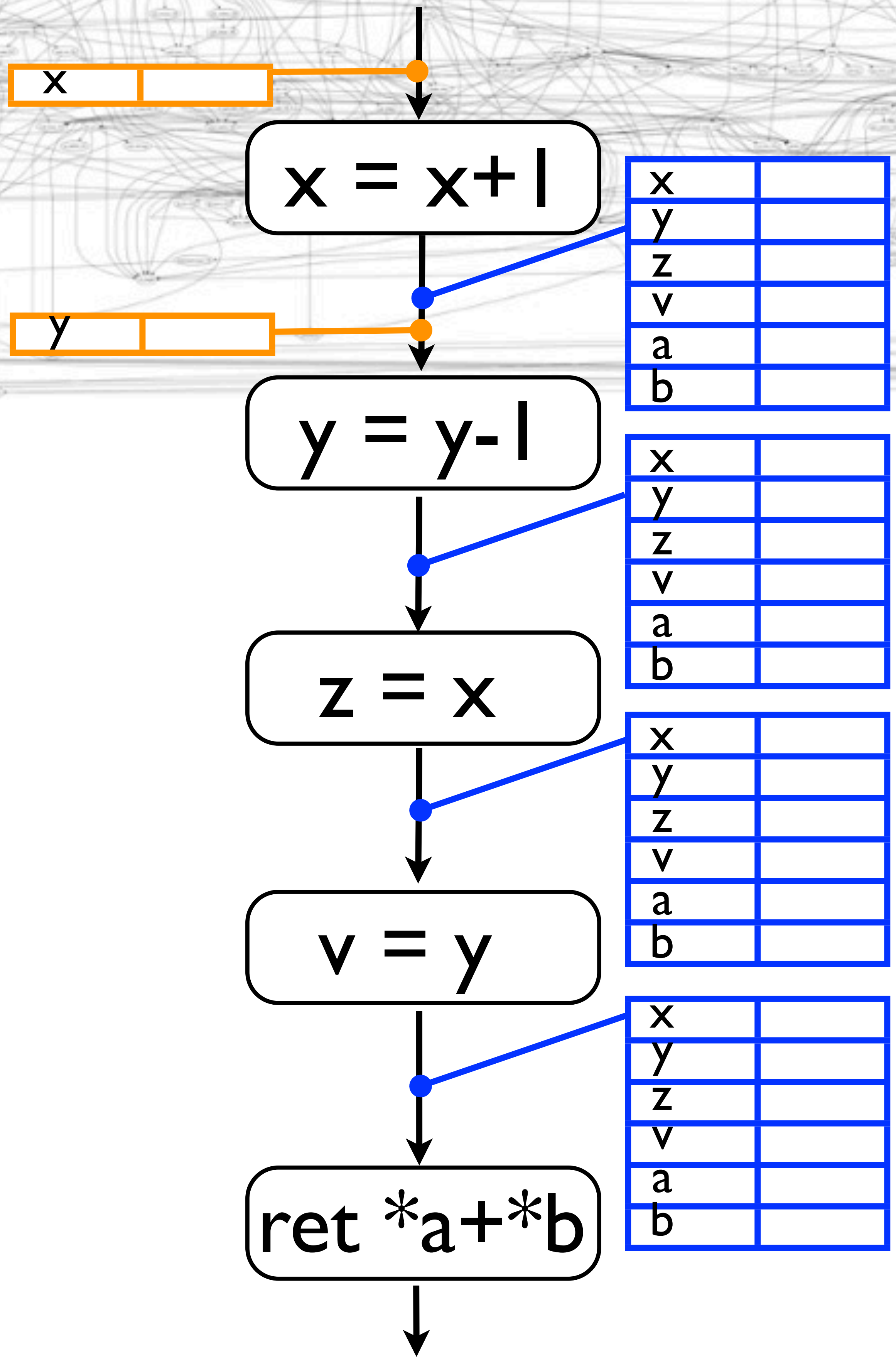
Spatial & Temporal Localizations



Spatial & Temporal Localizations



Spatial & Temporal Localizations



x	

x = x+1

x	
y	
z	
v	
a	
b	

y	

y = y-1

x	
y	
z	
v	
a	
b	

z = x

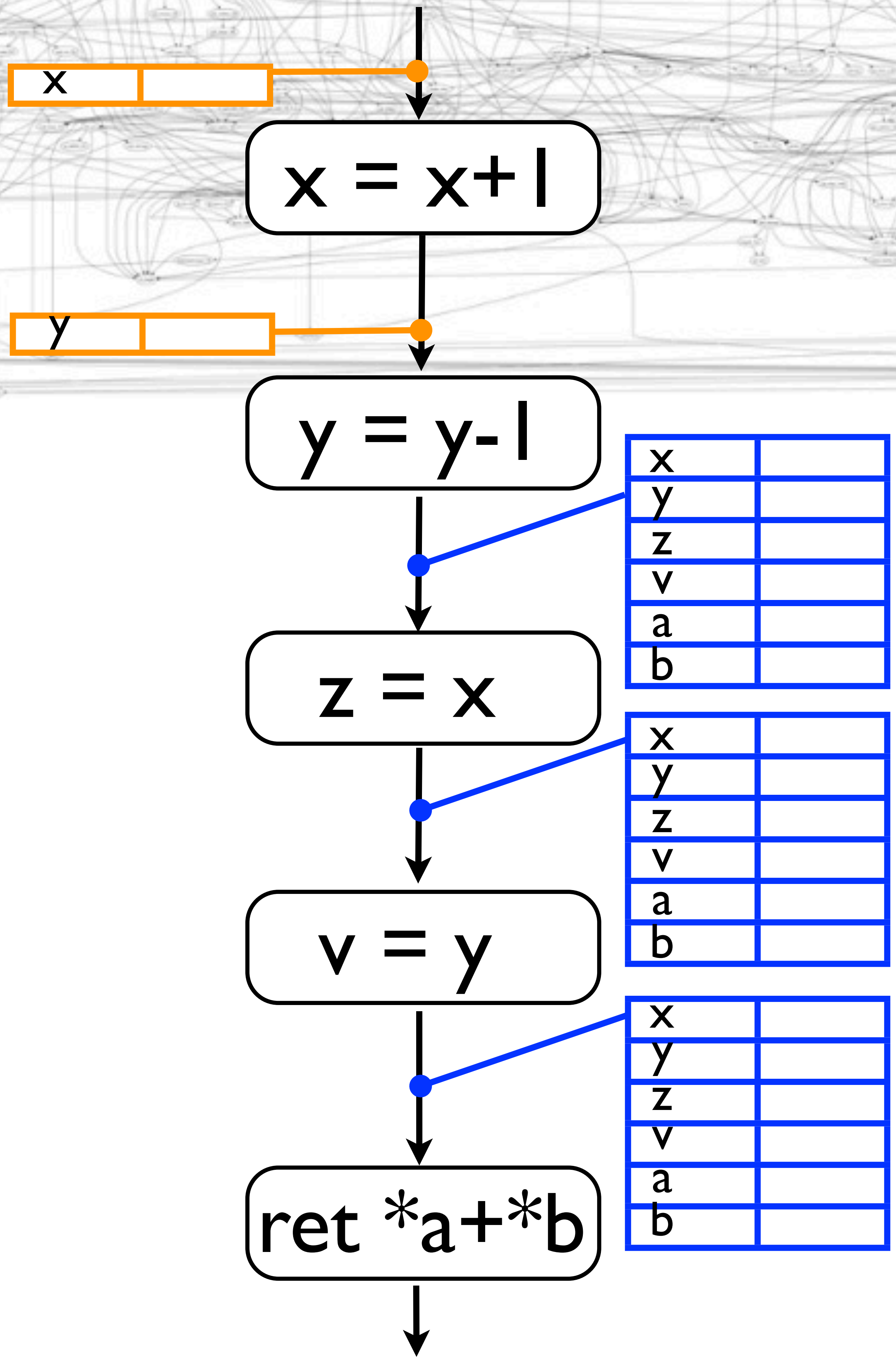
x	
y	
z	
v	
a	
b	

v = y

x	
y	
z	
v	
a	
b	

ret *a+*b

Spatial & Temporal Localizations



x

y

x = x+1

y = y-1

z = x

v = y

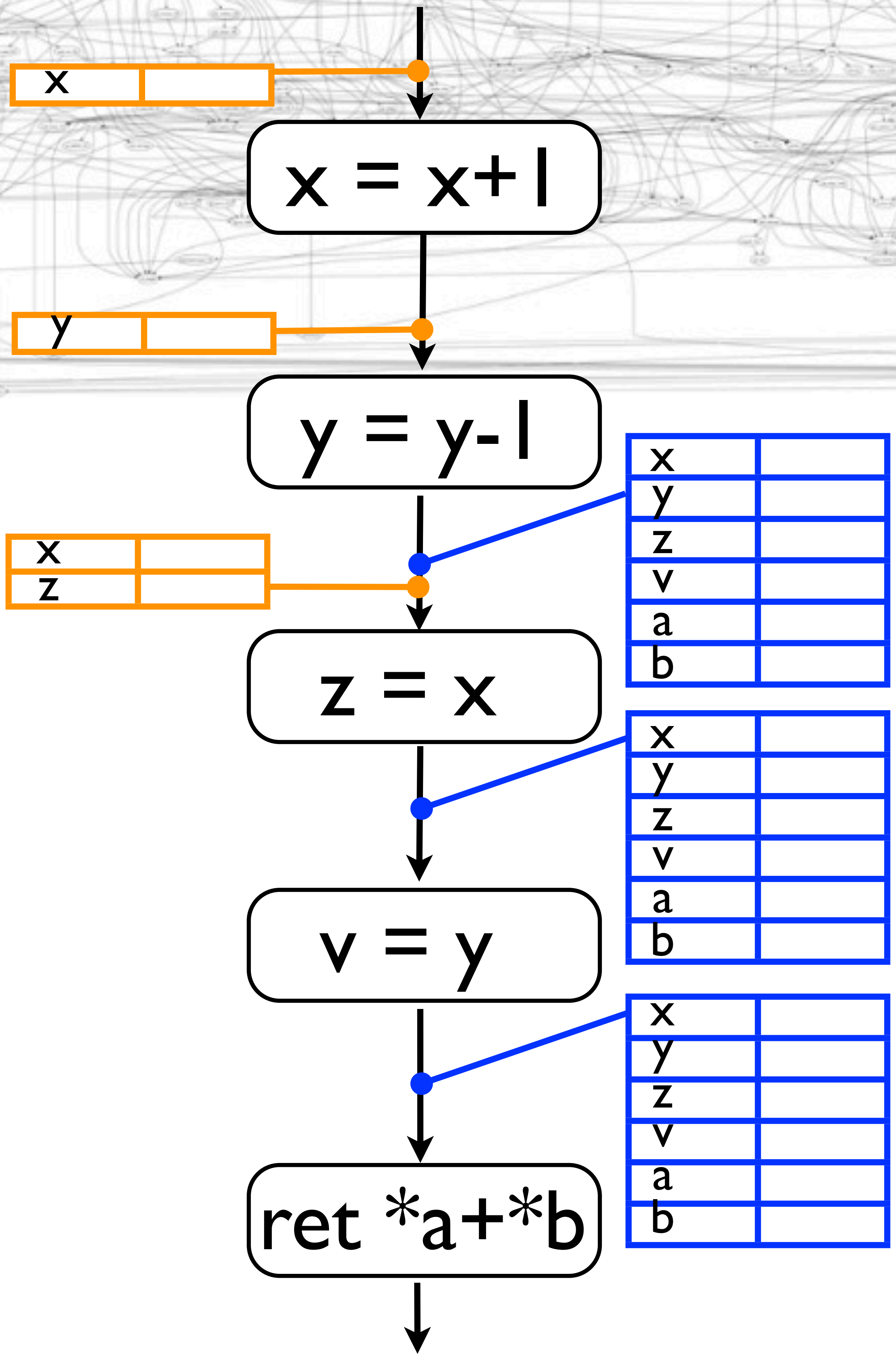
ret *a+*b

x
y
z
v
a
b

x
y
z
v
a
b

x
y
z
v
a
b

Spatial & Temporal Localizations



x = x+1

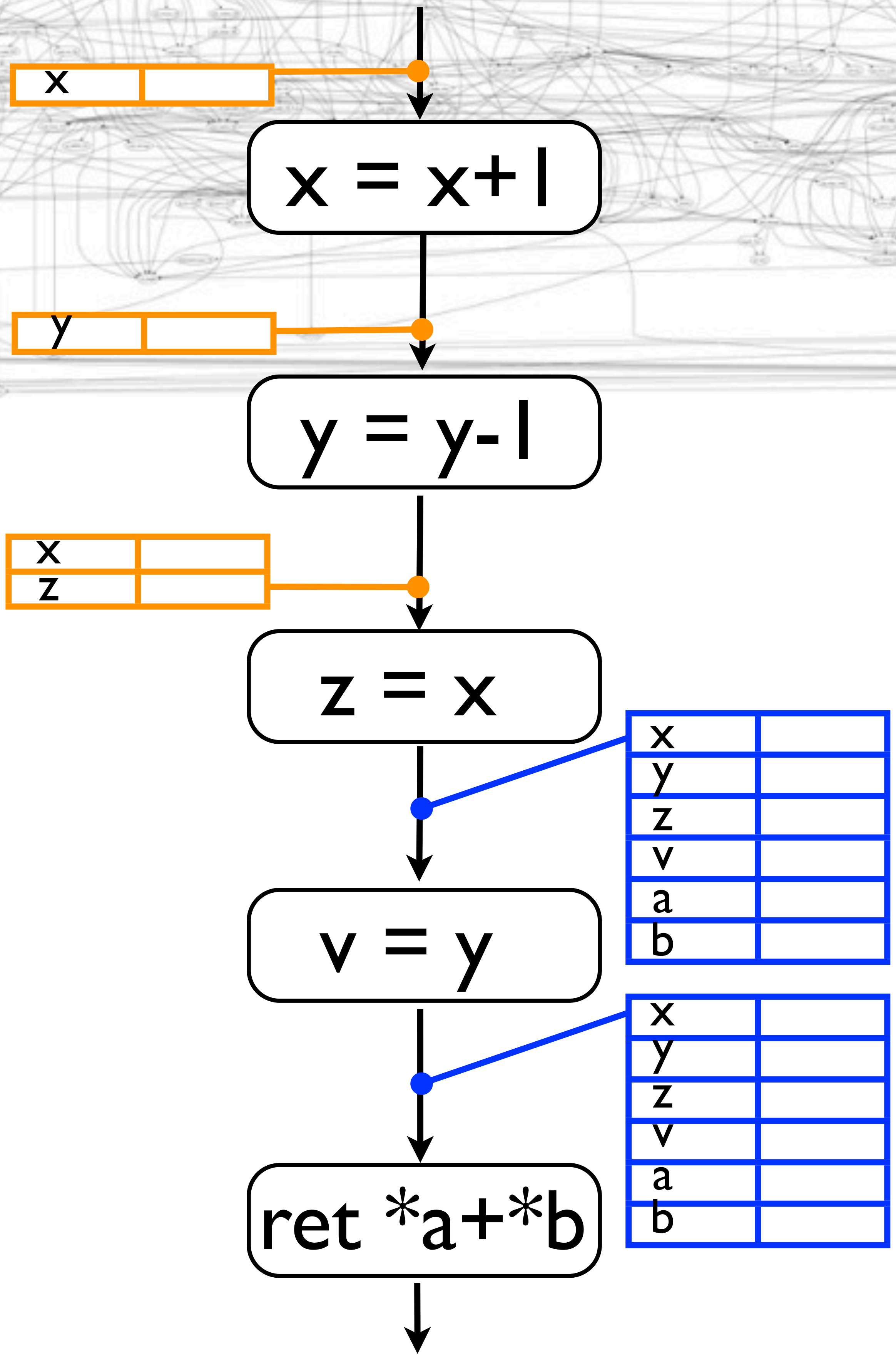
y = y-1

z = x

v = y

ret *a+*b

Spatial & Temporal Localizations



x = x+1

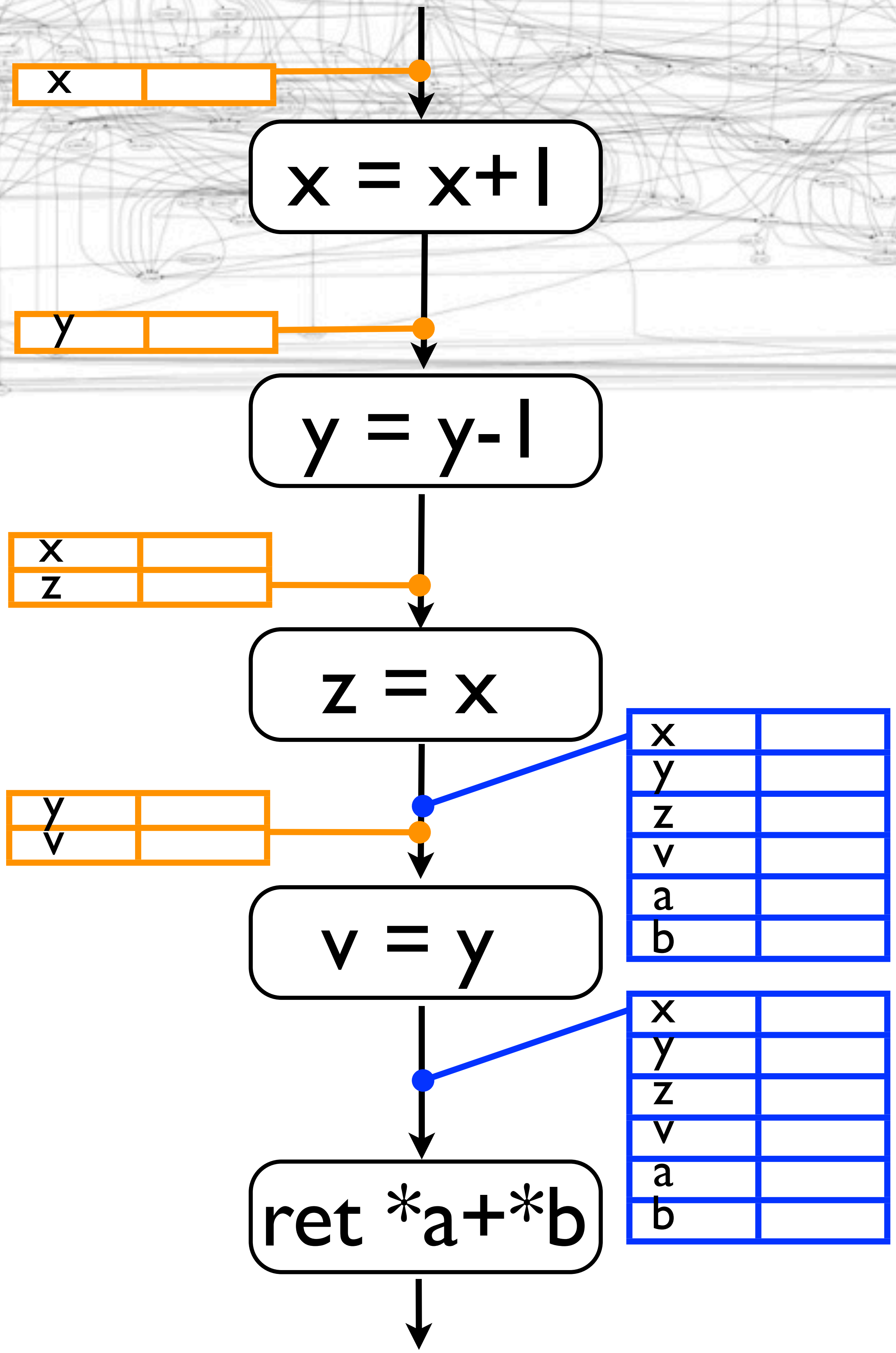
y = y-1

z = x

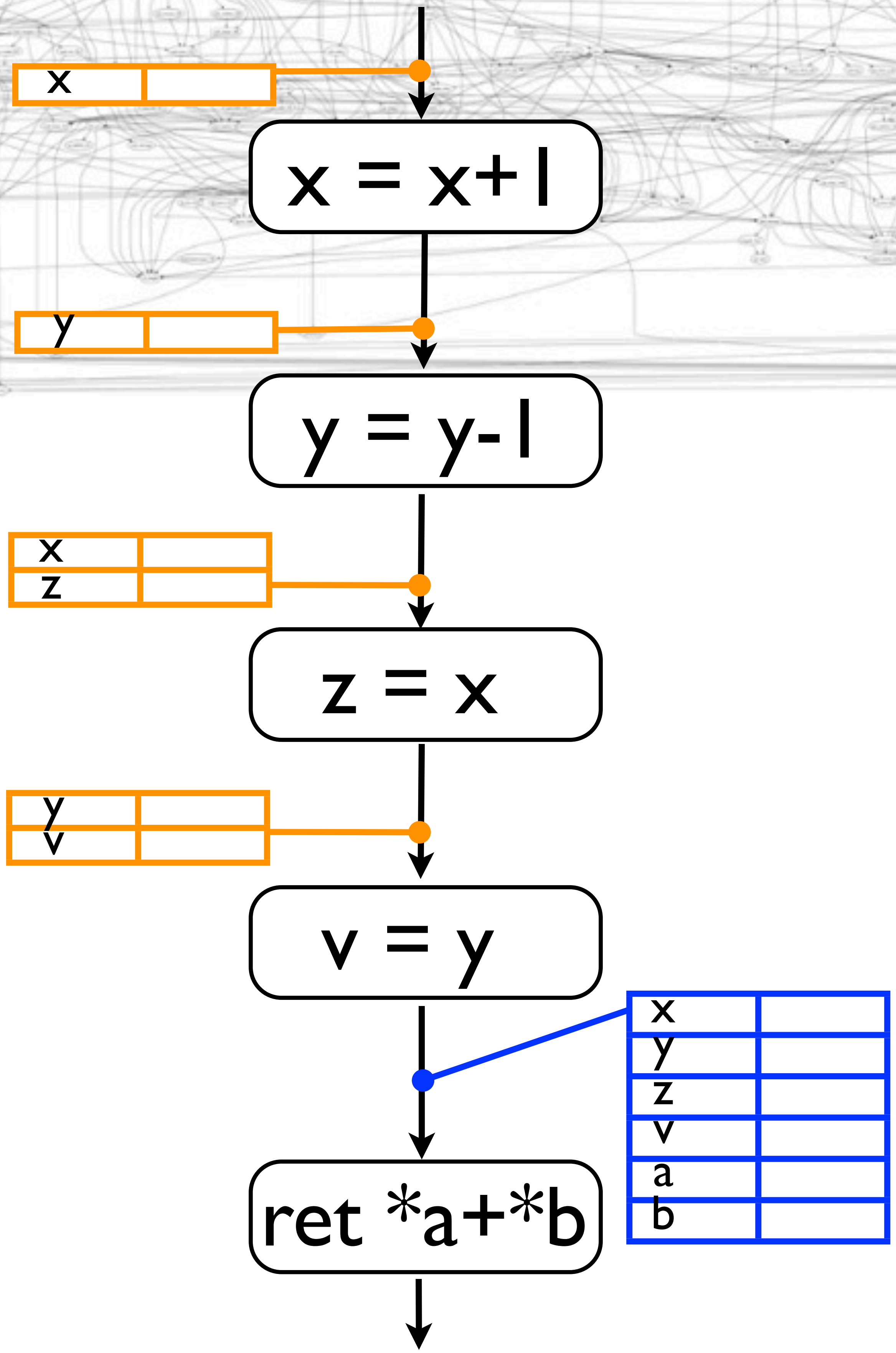
v = y

ret *a+*b

Spatial & Temporal Localizations



Spatial & Temporal Localizations



x = x+1

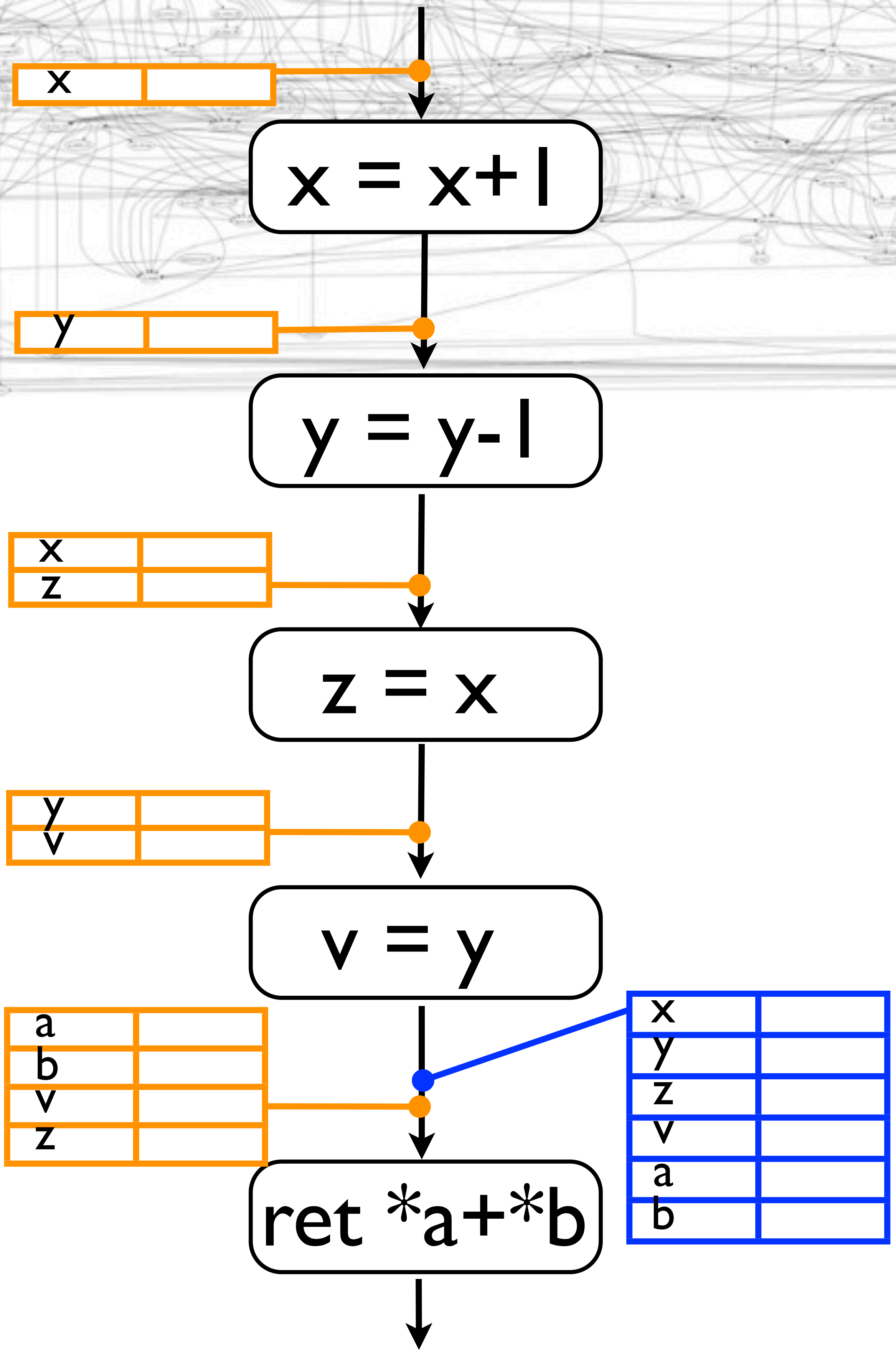
y = y-1

z = x

v = y

ret *a+*b

Spatial & Temporal Localizations



x = x+1

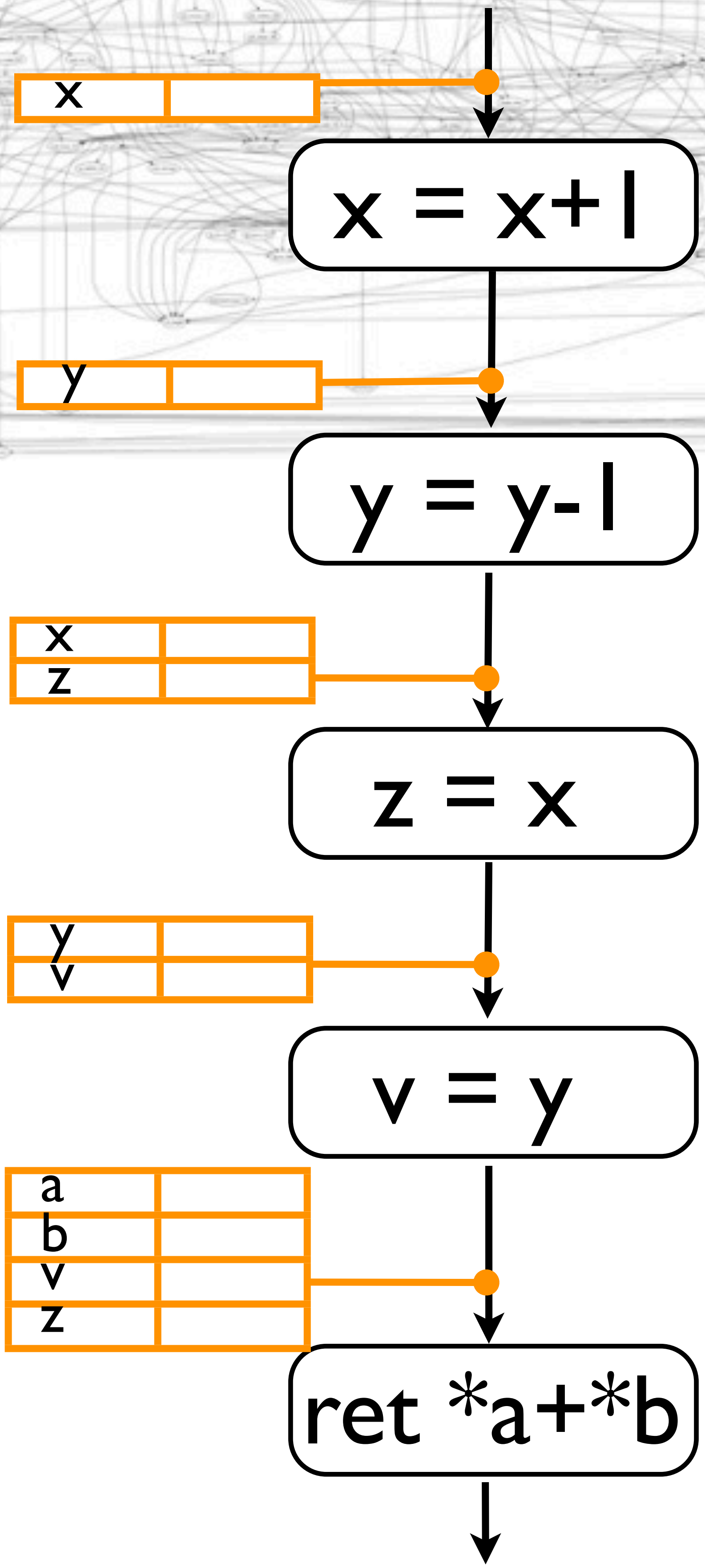
y = y-1

z = x

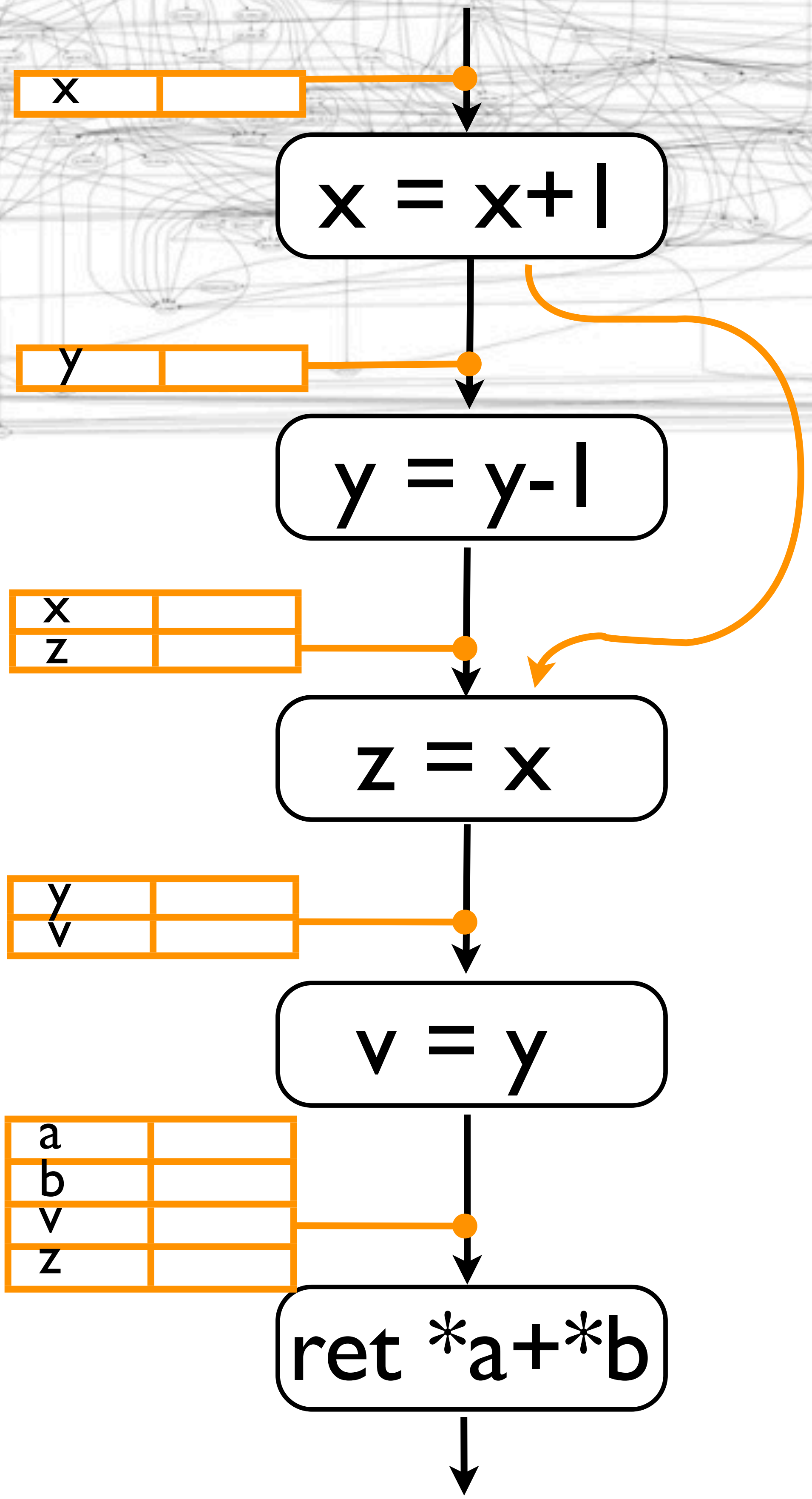
v = y

ret *a+*b

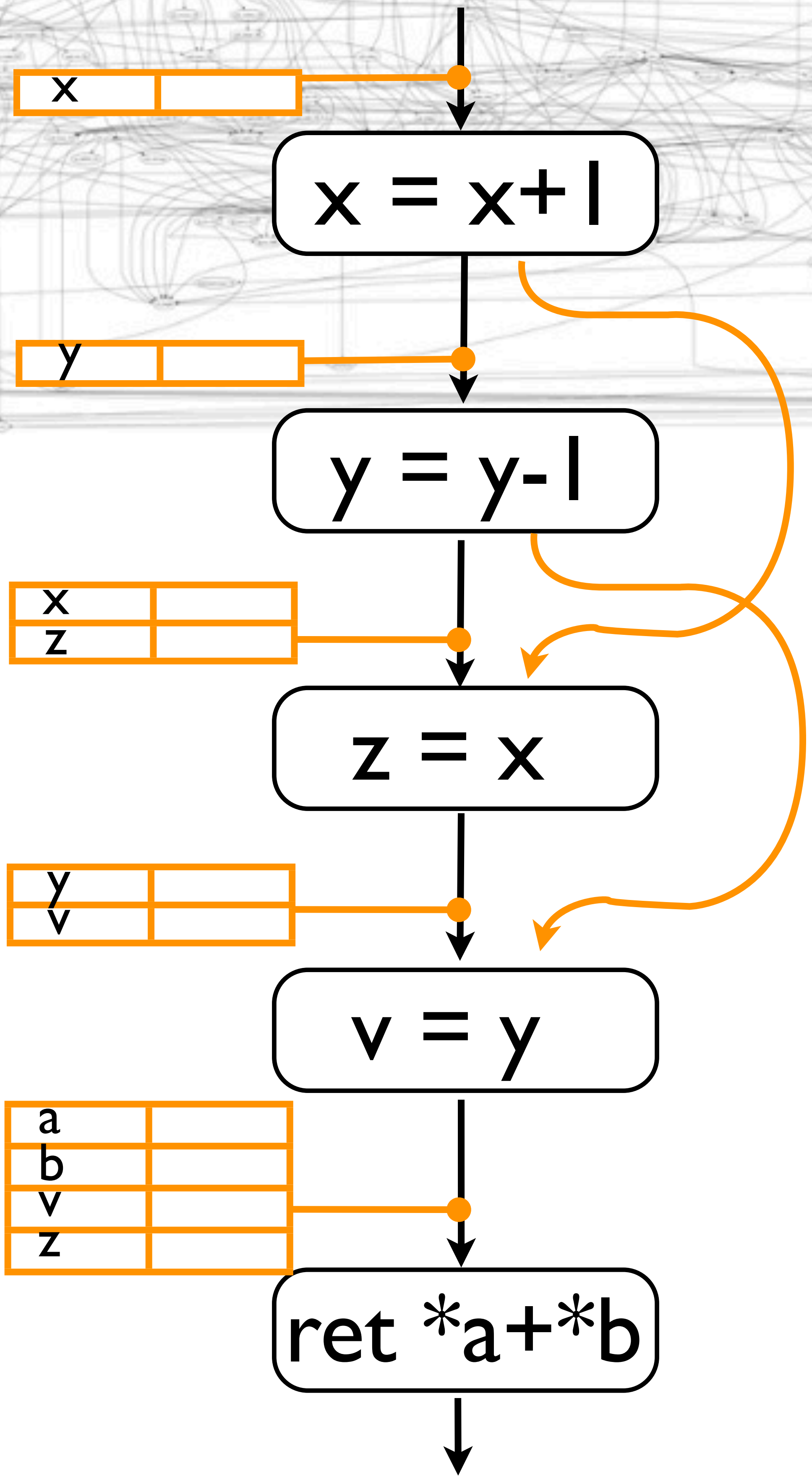
Spatial & Temporal Localizations



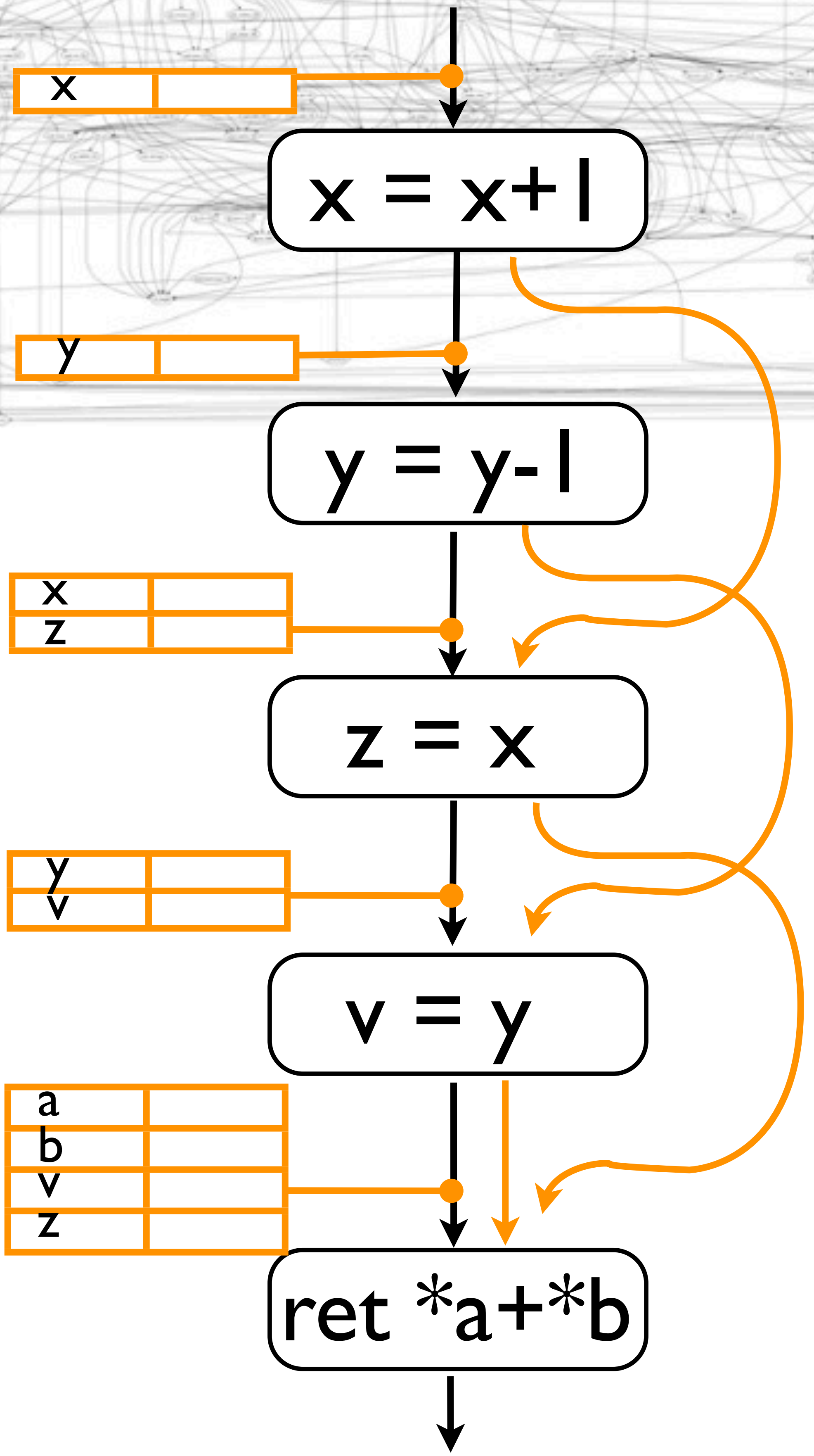
Spatial & Temporal Localizations



Spatial & Temporal Localizations



Spatial & Temporal Localizations



x

x = x+1

y

y = y-1

x
z

z = x

y
v

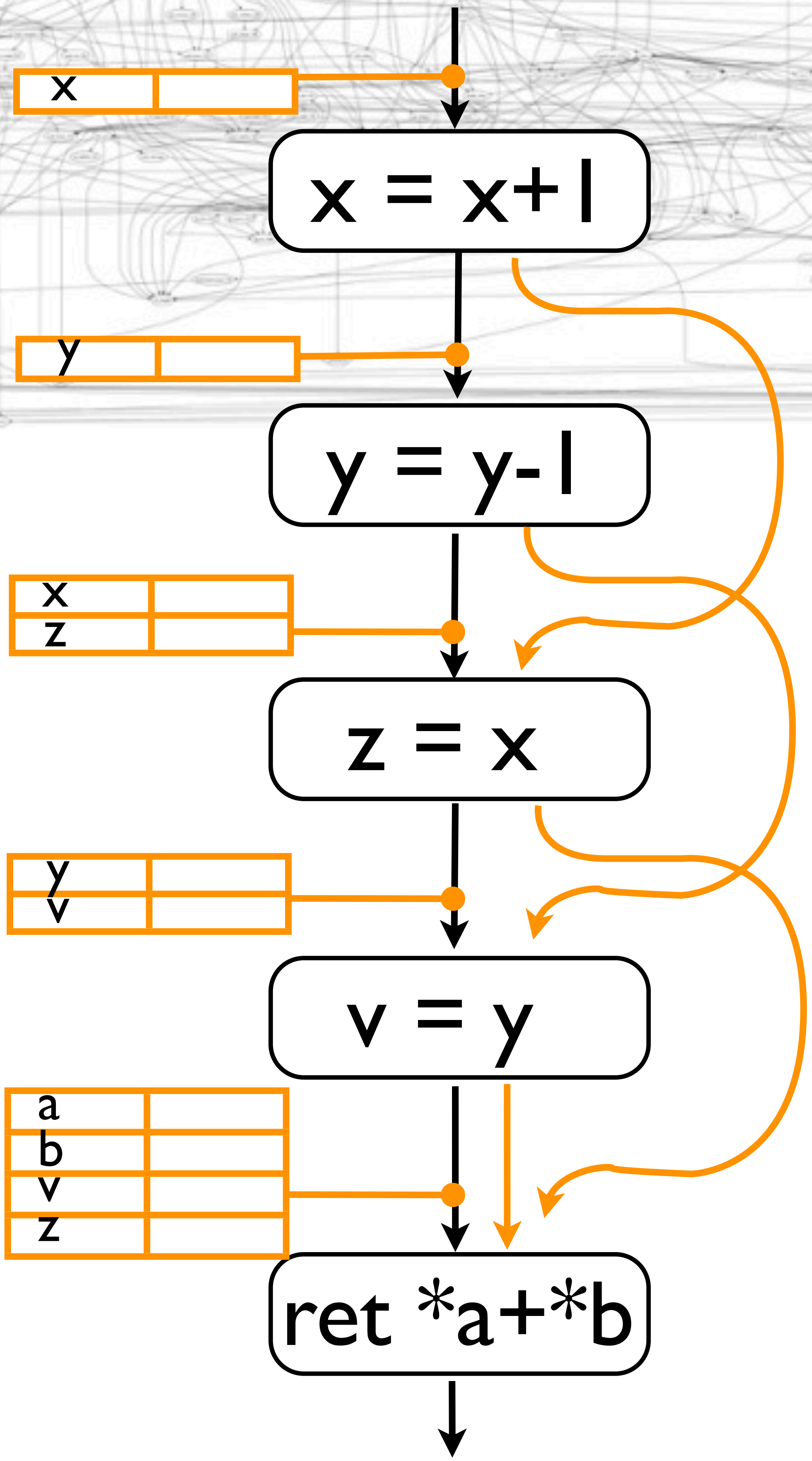
v = y

a
b
v
z

ret *a+*b

Spatial & Temporal Localizations

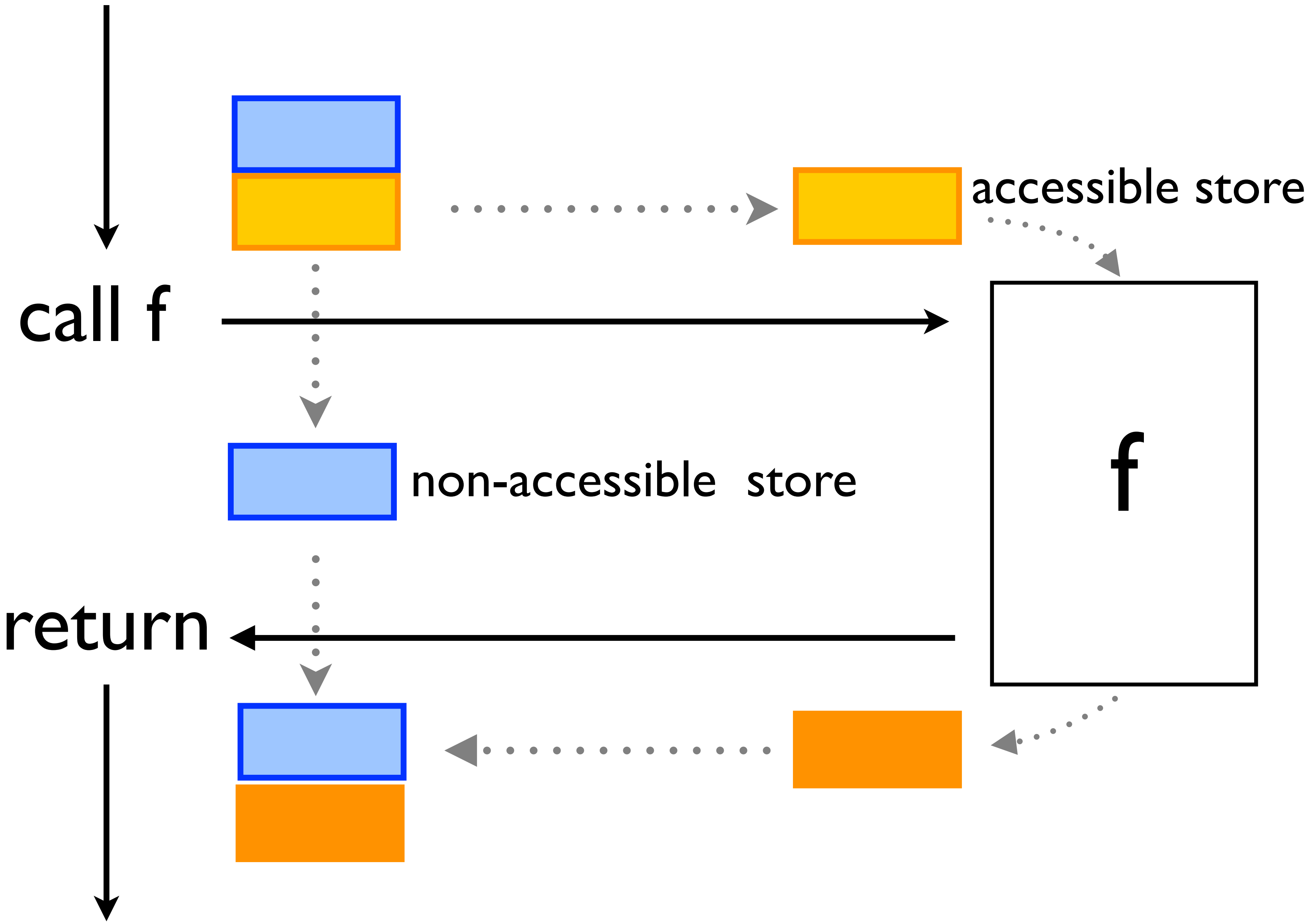
```
$\hat{X}, \hat{X}' \in \mathbb{C} \to \hat{\mathbb{S}}$
$\hat{f}_c \in \hat{\mathbb{S}} \to \hat{\mathbb{S}}$

$\hat{X} := \hat{X}' := \lambda c.\bot$
repeat
    $\hat{X}' := \hat{X}$
    for all $c \in \mathbb{C}$ do
        $\hat{X}(c) := \hat{f}_c(\bigsqcup_{\boxed{c' \hookrightarrow c}} \boxed{\hat{X}(c')})$
until $\hat{X} \sqsubseteq \hat{X}'$
```



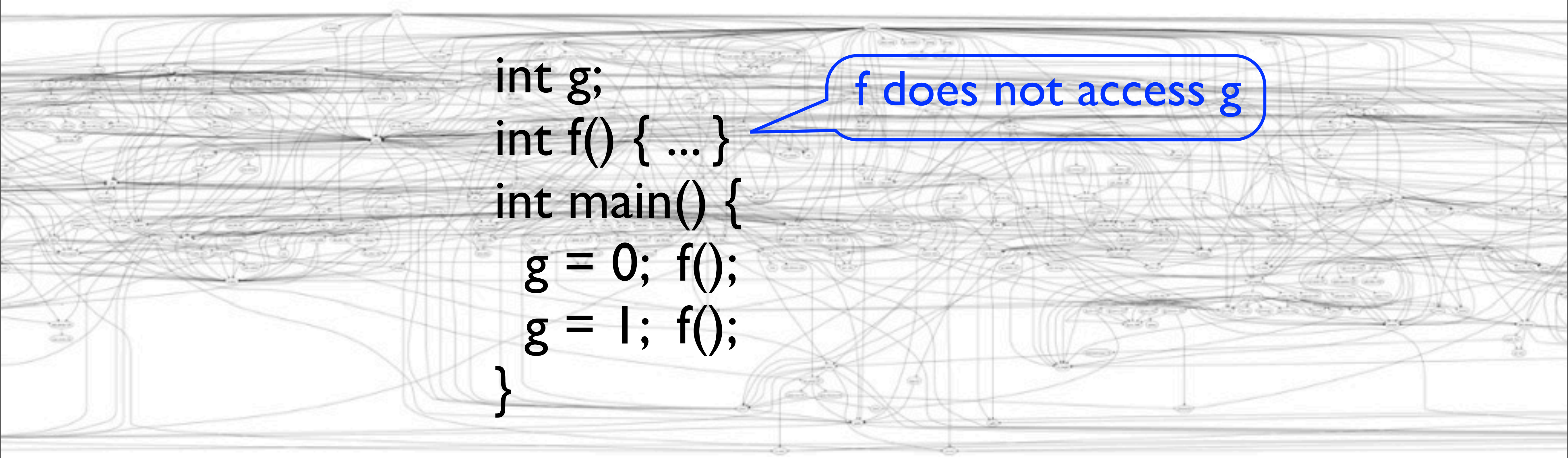
Spatial Localization

Spatial Localization

(“framing”, “abstract gc”)



Vital in Analysis Practice



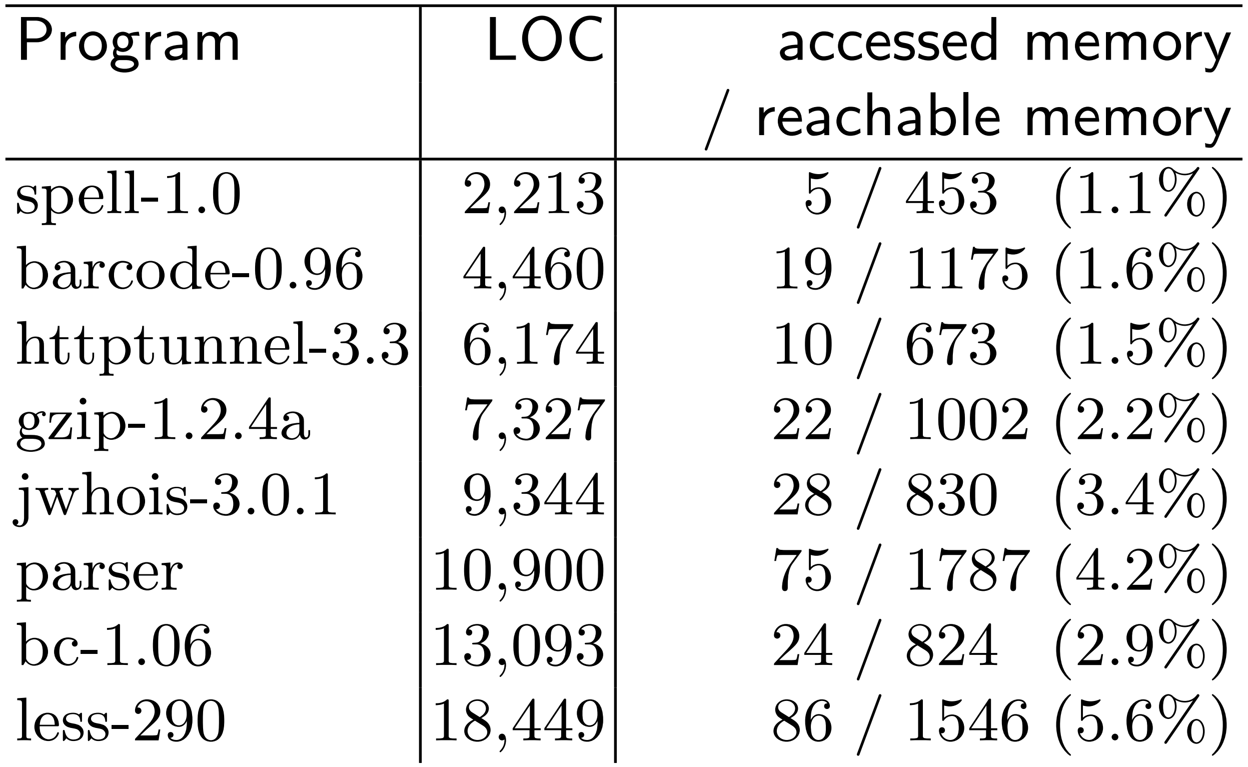
```
int g;
int f() { ... }
int main() {
  g = 0;  f();
  g = 1;  f();
}
```

f does not access g

On average, 755 re-analysis per procedure

But Existing Approach is Too Conservative

huge room for localizations than reachability-based technique



Program	LOC	accessed memory / reachable memory
spell-1.0	2,213	5 / 453 (1.1%)
barcode-0.96	4,460	19 / 1175 (1.6%)
httptunnel-3.3	6,174	10 / 673 (1.5%)
gzip-1.2.4a	7,327	22 / 1002 (2.2%)
jwhois-3.0.1	9,344	28 / 830 (3.4%)
parser	10,900	75 / 1787 (4.2%)
bc-1.06	13,093	24 / 824 (2.9%)
less-290	18,449	86 / 1546 (5.6%)

average : only 4%

Hurdle: Accessed Locations Before Analysis?

- Yes, by yet another analysis
- The pre-analysis must be quick
- The pre-analysis must be safe
  - over-estimating the accessed abstract locs

Our Pre-analysis

For Safely Estimating the Accessed Abstract Locations

- one further abstraction
- correct design

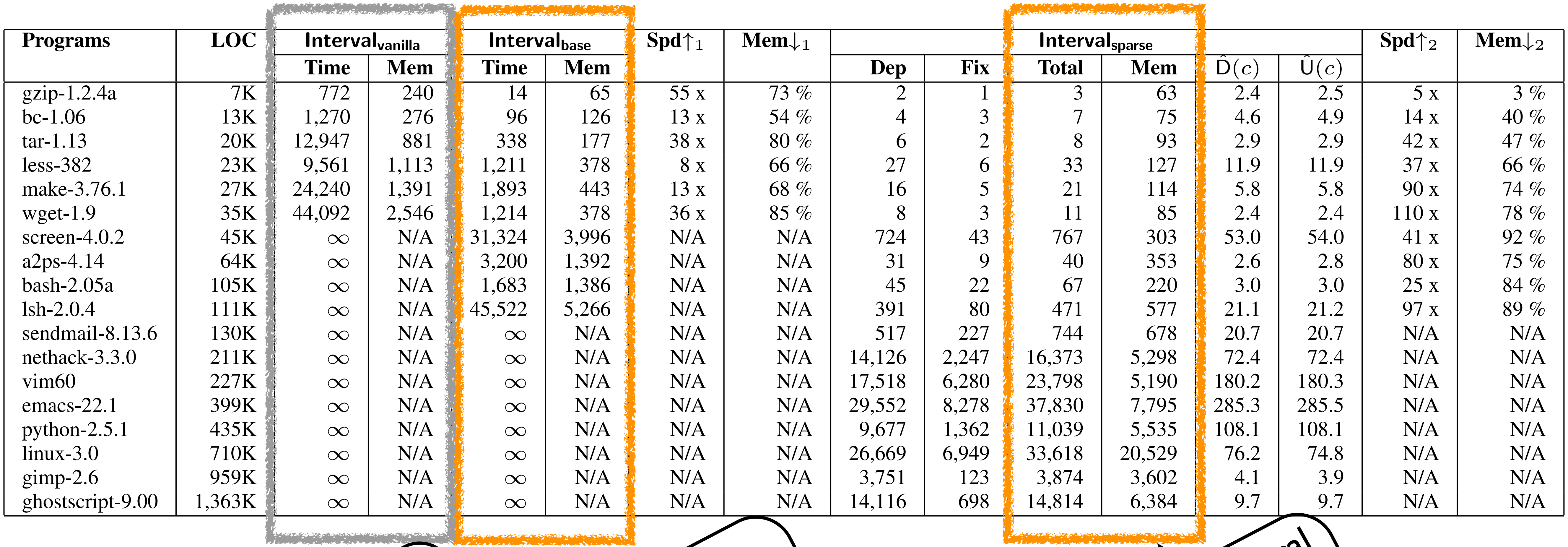
$$\mathbb{C} \to \hat{\mathbb{S}} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \hat{\mathbb{S}}$$

- abstract semantic function: flow-insensitive

$$\hat{F}_p = \lambda \hat{s}.(\bigsqcup_{c \in \mathbb{C}} \hat{f}_c(\hat{s}))$$

Performance of sound & global Sparrow

The Early Bird



Programs	LOC	Interval$_{\text{vanilla}}$		Interval$_{\text{base}}$		Spd$\uparrow_1$	Mem$\downarrow_1$	Interval$_{\text{sparse}}$						Spd$\uparrow_2$	Mem$\downarrow_2$
		Time	Mem	Time	Mem			Dep	Fix	Total	Mem	$\hat{D}(c)$	$\hat{U}(c)$		
gzip-1.2.4a	7K	772	240	14	65	55 x	73 %	2	1	3	63	2.4	2.5	5 x	3 %
bc-1.06	13K	1,270	276	96	126	13 x	54 %	4	3	7	75	4.6	4.9	14 x	40 %
tar-1.13	20K	12,947	881	338	177	38 x	80 %	6	2	8	93	2.9	2.9	42 x	47 %
less-382	23K	9,561	1,113	1,211	378	8 x	66 %	27	6	33	127	11.9	11.9	37 x	66 %
make-3.76.1	27K	24,240	1,391	1,893	443	13 x	68 %	16	5	21	114	5.8	5.8	90 x	74 %
wget-1.9	35K	44,092	2,546	1,214	378	36 x	85 %	8	3	11	85	2.4	2.4	110 x	78 %
screen-4.0.2	45K	∞	N/A	31,324	3,996	N/A	N/A	724	43	767	303	53.0	54.0	41 x	92 %
a2ps-4.14	64K	∞	N/A	3,200	1,392	N/A	N/A	31	9	40	353	2.6	2.8	80 x	75 %
bash-2.05a	105K	∞	N/A	1,683	1,386	N/A	N/A	45	22	67	220	3.0	3.0	25 x	84 %
lsh-2.0.4	111K	∞	N/A	45,522	5,266	N/A	N/A	391	80	471	577	21.1	21.2	97 x	89 %
sendmail-8.13.6	130K	∞	N/A	∞	N/A	N/A	N/A	517	227	744	678	20.7	20.7	N/A	N/A
nethack-3.3.0	211K	∞	N/A	∞	N/A	N/A	N/A	14,126	2,247	16,373	5,298	72.4	72.4	N/A	N/A
vim60	227K	∞	N/A	∞	N/A	N/A	N/A	17,518	6,280	23,798	5,190	180.2	180.3	N/A	N/A
emacs-22.1	399K	∞	N/A	∞	N/A	N/A	N/A	29,552	8,278	37,830	7,795	285.3	285.5	N/A	N/A
python-2.5.1	435K	∞	N/A	∞	N/A	N/A	N/A	9,677	1,362	11,039	5,535	108.1	108.1	N/A	N/A
linux-3.0	710K	∞	N/A	∞	N/A	N/A	N/A	26,669	6,949	33,618	20,529	76.2	74.8	N/A	N/A
gimp-2.6	959K	∞	N/A	∞	N/A	N/A	N/A	3,751	123	3,874	3,602	4.1	3.9	N/A	N/A
ghostscript-9.00	1,363K	∞	N/A	∞	N/A	N/A	N/A	14,116	698	14,814	6,384	9.7	9.7	N/A	N/A

none

spatial localization

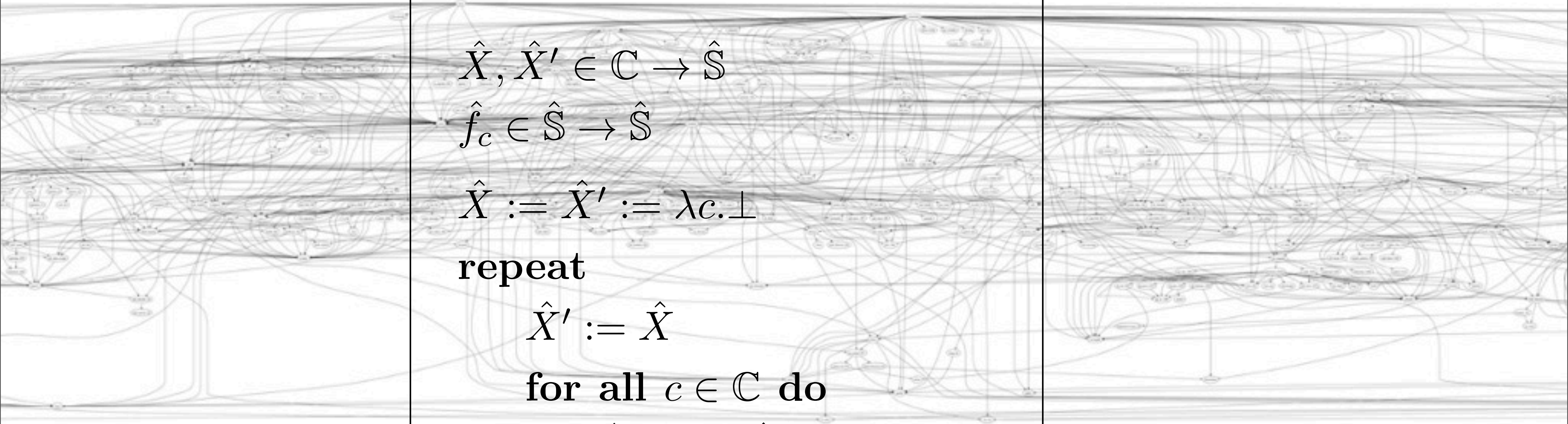
spatial+temporal localization

Temporal Localization

(and spatial localization automatically follows)

Temporal Localization

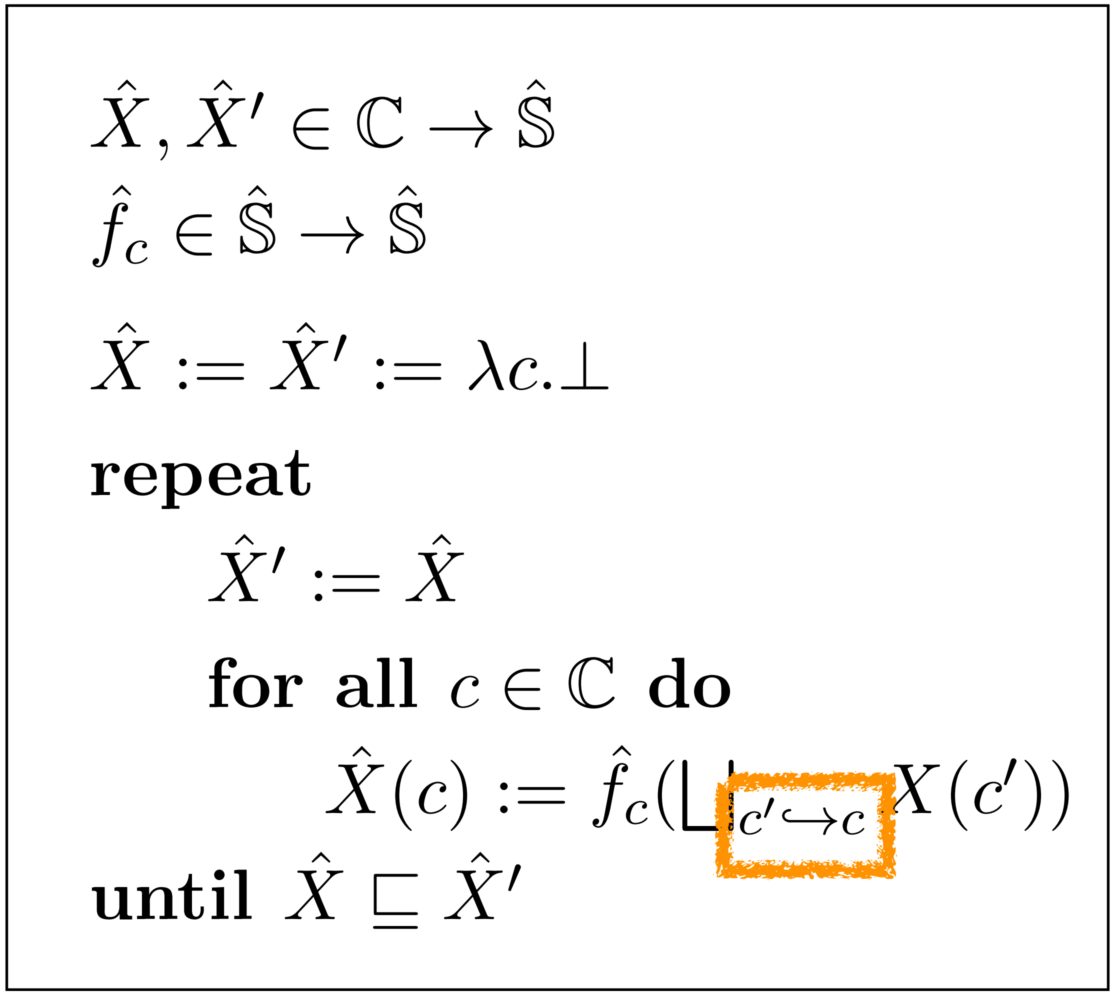
- Don’t blindly follow the control flow of pgm text
- Follow the dependency of statement semantics
  - from definition points directly to their use points



```
$\hat{X}, \hat{X}' \in \mathbb{C} \to \hat{\mathbb{S}}$
$\hat{f}_c \in \hat{\mathbb{S}} \to \hat{\mathbb{S}}$

$\hat{X} := \hat{X}' := \lambda c.\bot$
repeat
    $\hat{X}' := \hat{X}$
    for all $c \in \mathbb{C}$ do
        $\hat{X}(c) := \hat{f}_c(\bigsqcup_{\boxed{c' \hookrightarrow c}} X(c'))$
until $\hat{X} \sqsubseteq \hat{X}'$
```

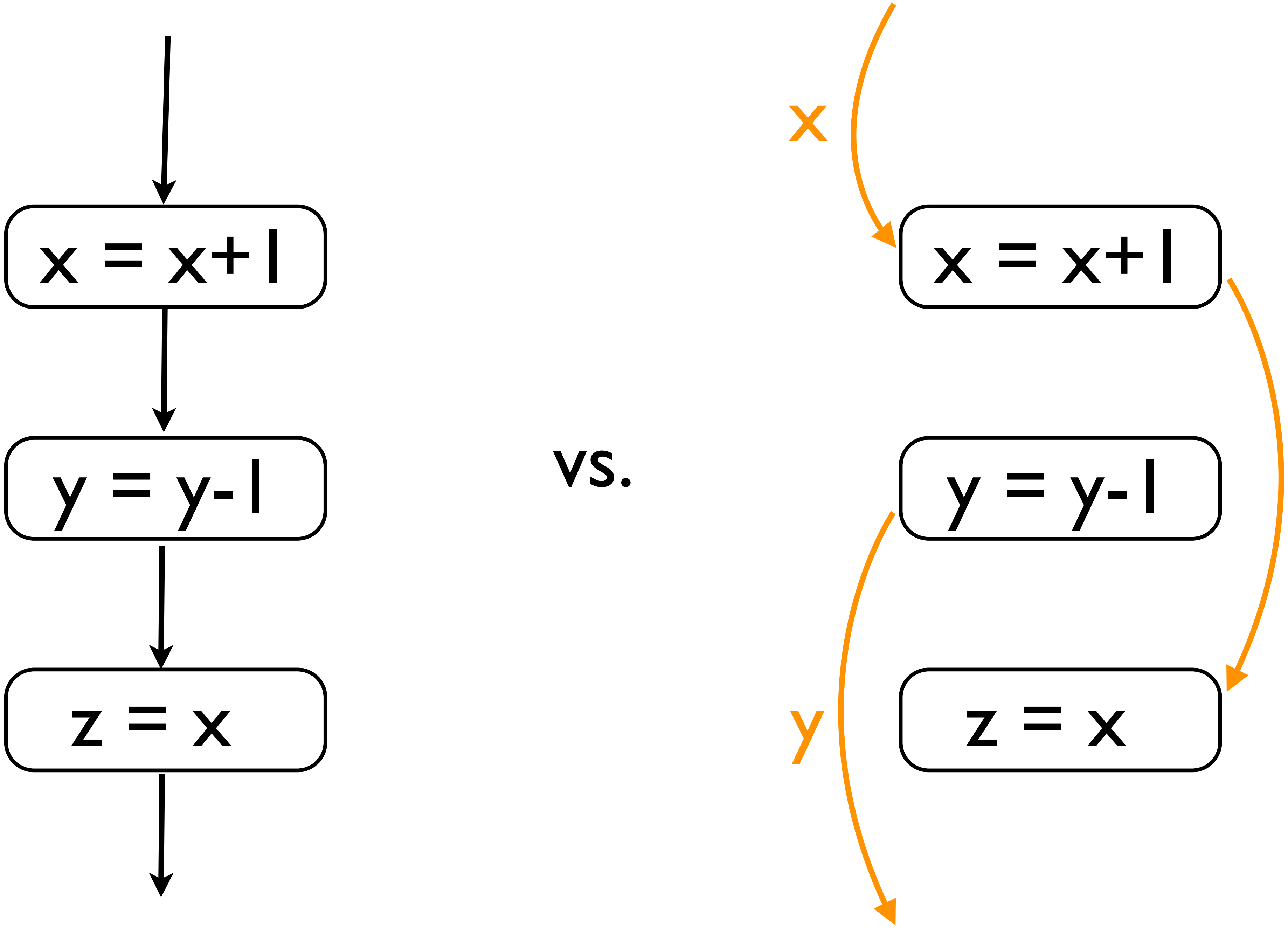
Temporal Localization

- Don’t blindly follow the control flow of pgm text
- Follow the dependency of statement semantics
  - from definition points directly to their use points



```
$\hat{X}, \hat{X}' \in \mathbb{C} \to \hat{\mathbb{S}}$
$\hat{f}_c \in \hat{\mathbb{S}} \to \hat{\mathbb{S}}$

$\hat{X} := \hat{X}' := \lambda c.\bot$
**repeat**
    $\hat{X}' := \hat{X}$
    **for all** $c \in \mathbb{C}$ **do**
        $\hat{X}(c) := \hat{f}_c(\bigsqcup_{\boxed{c' \hookrightarrow c}} X(c'))$
**until** $\hat{X} \sqsubseteq \hat{X}'$
```

Temporal Localization



Precision Preserving Sparse Analysis Framework

$$\hat{F} : \hat{D} \to \hat{D} \qquad \overset{\text{sparsify}}{\Longrightarrow} \qquad \hat{F}_s : \hat{D} \to \hat{D}$$

$$\mathit{fix}\,\hat{F} \qquad \overset{\text{still}}{=} \qquad \mathit{fix}\,\hat{F}_s$$

Towards Sparse Version

Analyzer computes the fixpoint of $\hat{F}_{-} \in (\mathbb{C} \to \hat{\mathbb{S}}) \to (\mathbb{C} \to \hat{\mathbb{S}})$

- baseline non-sparse one

$$\hat{F}(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c(\bigsqcup_{\boxed{c' \hookrightarrow c}} \hat{X}(c')).$$

- unrealizable sparse version

$$\hat{F}_s(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c(\bigsqcup_{\boxed{c' \overset{l}{\rightsquigarrow} c}} \hat{X}(c')|_l).$$

- realizable sparse version

$$\hat{F}_a(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c(\bigsqcup_{\boxed{c' \overset{l}{\rightsquigarrow}_a c}} \hat{X}(c')|_l).$$

Unrealizable Sparse One

$$\hat{F}_s(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c(\bigsqcup_{\boxed{c' \overset{l}{\leadsto} c}} \hat{X}(c')|_l).$$

Data Dependency

$$\begin{aligned} c_0 \overset{l}{\leadsto} c_n \quad \triangleq \quad & \exists c_0 \ldots c_n \in \mathsf{Paths}, l \in \hat{\mathbb{L}}. \\ & l \in \mathsf{D}(c_0) \cap \mathsf{U}(c_n) \wedge \forall i \in (0, n). l \notin \mathsf{D}(c_i) \end{aligned}$$

Unrealizable Sparse One

$$\hat{F}_s(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c(\bigsqcup_{\boxed{c' \overset{l}{\rightsquigarrow} c}} \hat{X}(c')|_l).$$

Data Dependency

$$\begin{aligned} c_0 \overset{l}{\rightsquigarrow} c_n \quad \triangleq \quad & \exists c_0 \ldots c_n \in \mathsf{Paths}, l \in \hat{\mathbb{L}}. \\ & l \in \mathsf{D}(c_0) \cap \mathsf{U}(c_n) \wedge \forall i \in (0, n).l \notin \mathsf{D}(c_i) \end{aligned}$$

Def-Use Sets

$$\mathsf{D}(c) \triangleq \{l \in \hat{\mathbb{L}} \mid \exists \hat{s} \sqsubseteq \bigsqcup_{c' \hookrightarrow c} \mathcal{S}(c').\hat{f}_c(\hat{s})(l) \neq \hat{s}(l)\}$$

$$\mathsf{U}(c) \triangleq \{l \in \hat{\mathbb{L}} \mid \exists \hat{s} \sqsubseteq \bigsqcup_{c' \hookrightarrow c} \mathcal{S}(c').\hat{f}_c(\hat{s})|_{\mathsf{D}(c)} \neq \hat{f}_c(\hat{s}\backslash_l)|_{\mathsf{D}(c)}\}$$

Unrealizable Sparse One

$$\hat{F}_s(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c\Big(\bigsqcup_{\boxed{c' \overset{l}{\rightsquigarrow} c}} \hat{X}(c')|_l\Big).$$

Data Dependency

$$\begin{aligned} c_0 \overset{l}{\rightsquigarrow} c_n \quad \triangleq \quad & \exists c_0 \ldots c_n \in \mathsf{Paths}, l \in \hat{\mathbb{L}}. \\ & l \in \mathsf{D}(c_0) \cap \mathsf{U}(c_n) \wedge \forall i \in (0,n).l \notin \mathsf{D}(c_i) \end{aligned}$$

Def-Use Sets

$$\mathsf{D}(c) \triangleq \{l \in \hat{\mathbb{L}} \mid \exists \hat{s} \sqsubseteq \bigsqcup_{c' \hookrightarrow c} \mathcal{S}(c').\hat{f}_c(\hat{s})(l) \neq \hat{s}(l)\}$$

$$\mathsf{U}(c) \triangleq \{l \in \hat{\mathbb{L}} \mid \exists \hat{s} \sqsubseteq \bigsqcup_{c' \hookrightarrow c} \mathcal{S}(c').\hat{f}_c(\hat{s})|_{\mathsf{D}(c)} \neq \hat{f}_c(\hat{s}\backslash_l)|_{\mathsf{D}(c)}\}$$

Precision Preserving

$$\mathit{fix}\hat{F} = \mathit{fix}\hat{F}_s \quad \text{modulo D}$$

Realizable Sparse One

$$\hat{F}_a(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c\Big(\bigsqcup_{\boxed{c' \overset{l}{\rightsquigarrow}_a c}} \hat{X}(c')|_l\Big).$$

Realizable Data Dependency

$$\begin{aligned} c_0 \overset{l}{\rightsquigarrow}_a c_n \quad \triangleq \quad & \exists c_0 \ldots c_n \in \mathsf{Paths}, l \in \hat{\mathbb{L}}. \\ & l \in \underline{\hat{\mathsf{D}}}(c_0) \cap \underline{\hat{\mathsf{U}}}(c_n) \wedge \forall i \in (0, n).l \notin \hat{\mathsf{D}}(c_i) \end{aligned}$$

Realizable Sparse One

$$\hat{F}_a(\hat{X}) = \lambda c \in \mathbb{C}.\hat{f}_c\Big(\bigsqcup_{\boxed{c' \overset{l}{\leadsto}_a c}} \hat{X}(c')|_l\Big).$$

Realizable Data Dependency

$$\begin{aligned} c_0 \overset{l}{\leadsto}_a c_n \quad \triangleq \quad & \exists c_0 \ldots c_n \in \mathsf{Paths}, l \in \hat{\mathbb{L}}. \\ & l \in \underline{\hat{\mathsf{D}}}(c_0) \cap \underline{\hat{\mathsf{U}}}(c_n) \wedge \forall i \in (0,n). l \notin \hat{\mathsf{D}}(c_i) \end{aligned}$$

Precision Preserving

$$\mathit{fix}\hat{F} = \mathit{fix}\hat{F}_a \quad \text{modulo } \hat{\mathsf{D}}$$

If the following conditions hold

Conditions on $\hat{\mathsf{D}}$ & $\hat{\mathsf{U}}$

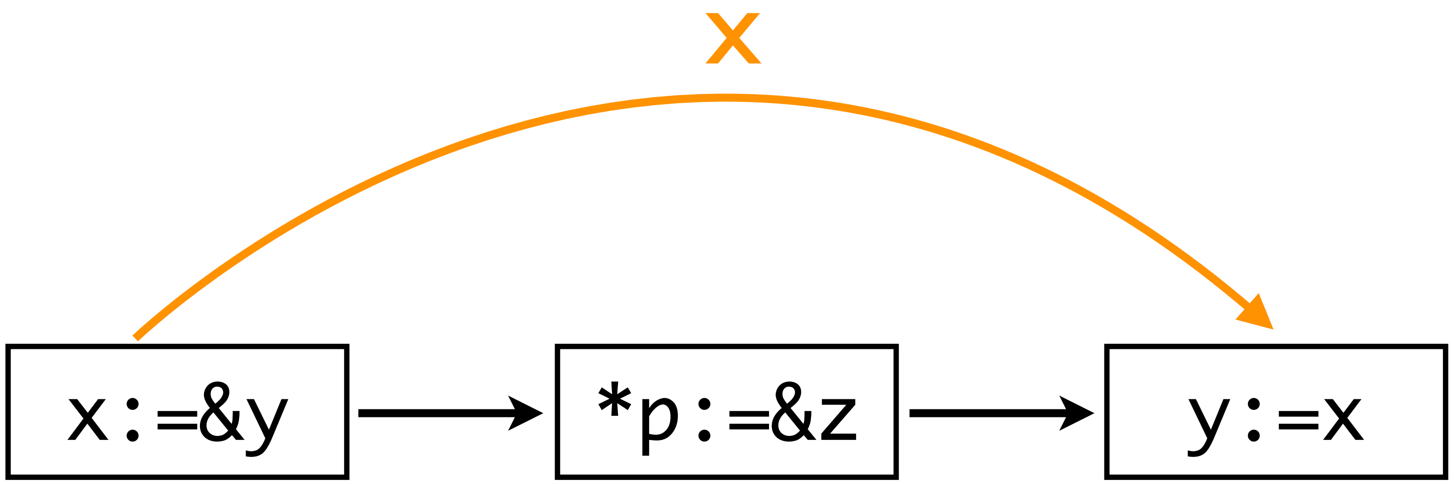
- over-approximation

$$\hat{\mathsf{D}}(c) \supseteq \mathsf{D}(c) \wedge \hat{\mathsf{U}}(c) \supseteq \mathsf{U}(c)$$

- spurious definitions should be also included in uses

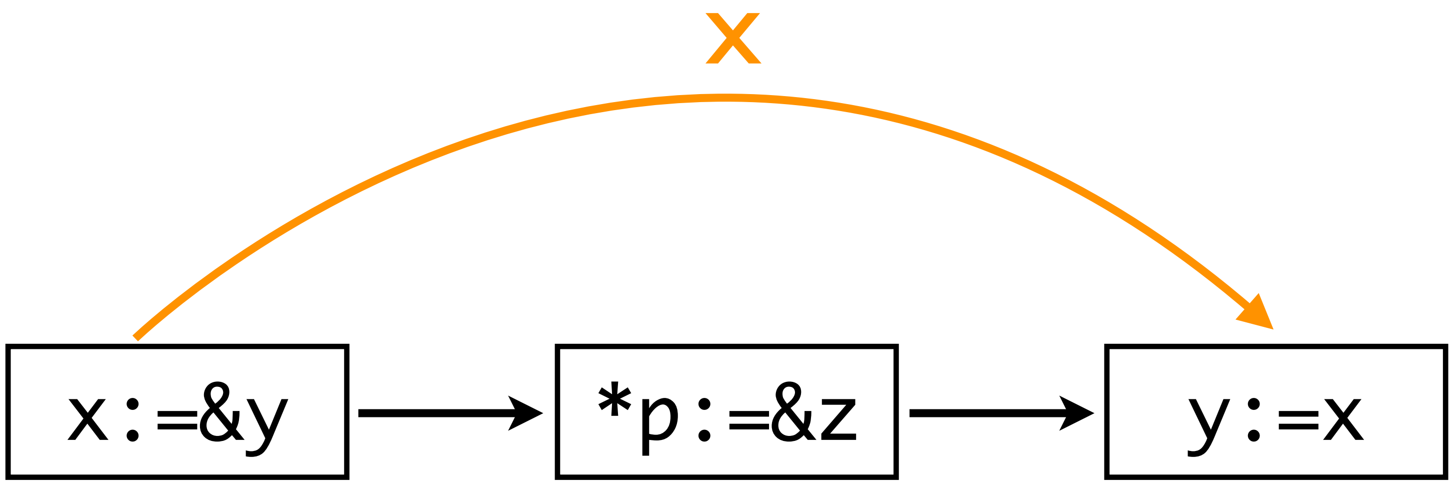
$$\hat{\mathsf{D}}(c) - \mathsf{D}(c) \subseteq \hat{\mathsf{U}}(c)$$

Why the Conditions of $\hat{D}$ & $\hat{U}$



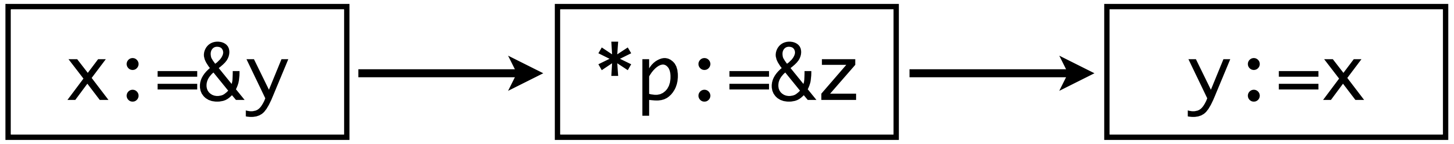
Def	{x}	{v, w}	{y}
Use	ϕ	{p, v, w}	{x}

Why the Conditions of $\hat{D}$ & $\hat{U}$



	x:=&y	*p:=&z	y:=x
Def	{x}	{v, w, x}	{y}
Use	ϕ	{p, v, w}	{x}

Why the Conditions of $\hat{D}$ & $\hat{U}$



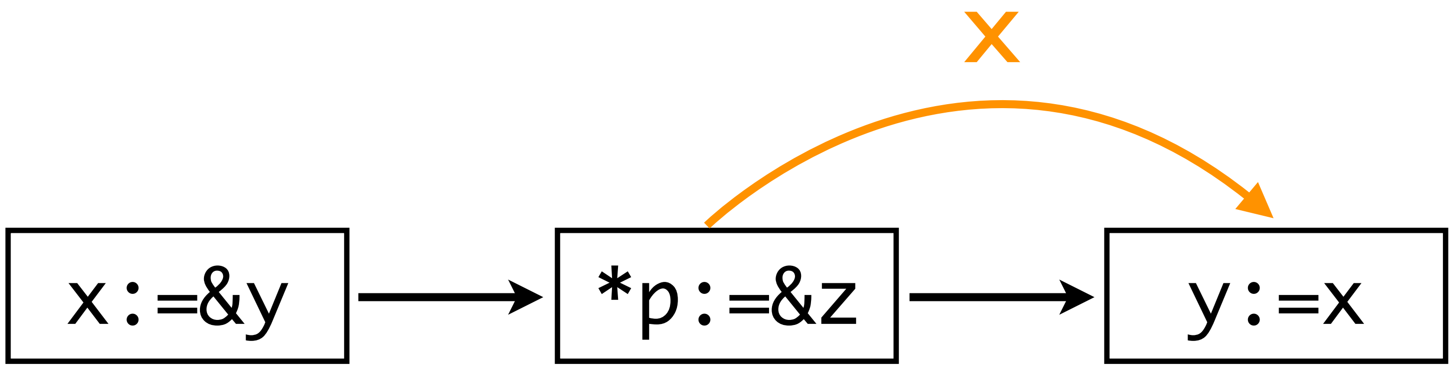
x:=&y

*p:=&z

y:=x

Def	{x}	{v, w, x}	{y}
Use	ϕ	{p, v, w}	{x}

Why the Conditions of $\hat{D}$ & $\hat{U}$



x

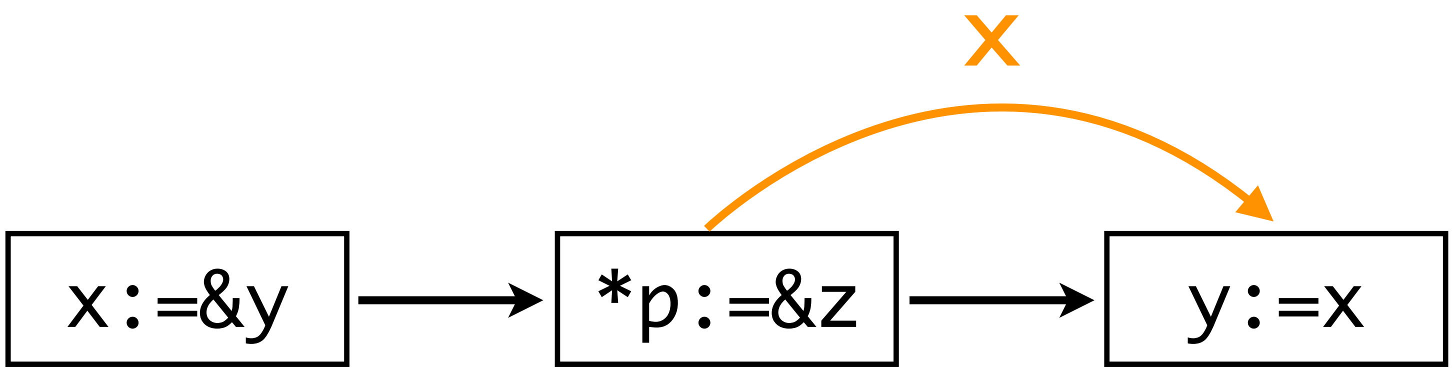
x:=&y

*p:=&z

y:=x

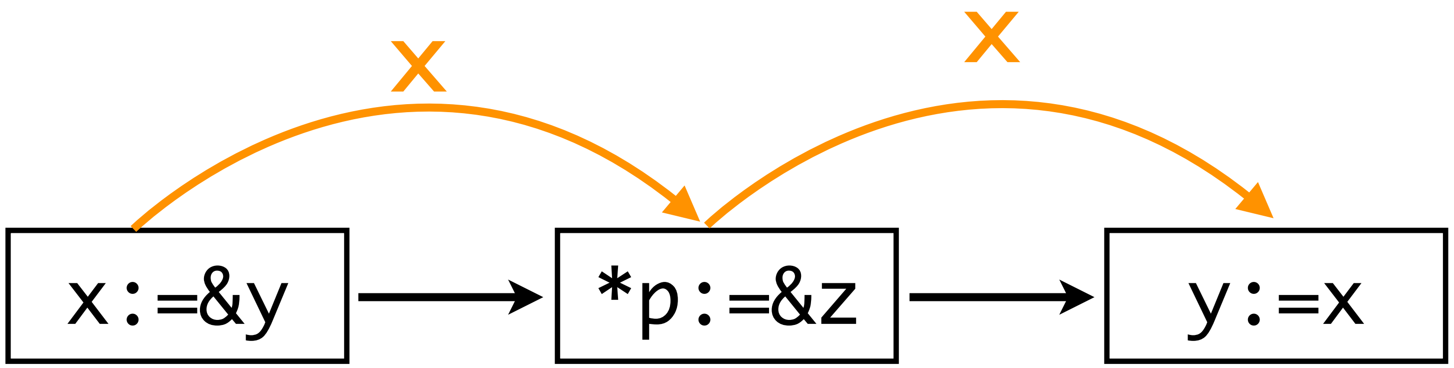
Def	{x}	{v, w, x}	{y}
Use	ϕ	{p, v, w}	{x}

Why the Conditions of $\hat{D}$ & $\hat{U}$



Def	{x}	{v, w, x}	{y}
Use	ϕ	{p, v, w, x}	{x}

Why the Conditions of $\hat{D}$ & $\hat{U}$



Def	{x}	{v, w, x}	{y}
Use	ϕ	{p, v, w, x}	{x}

Hurdle: $\hat{\mathrm{D}}$ & $\hat{\mathrm{U}}$ Before Analysis?

- Yes, by yet another analysis with further abstraction
- correct design

$$\begin{tikzcd}
\mathbb{C} \arrow[r] & \hat{\mathbb{S}} \arrow[r, shift right, "\alpha"'] & \hat{\mathbb{S}} \arrow[l, shift right, "\gamma"']
\end{tikzcd}$$

- abstract semantic function: flow-insensitive

$$\hat{F}_p = \lambda \hat{s}.(\bigsqcup_{c \in \mathbb{C}} \hat{f}_c(\hat{s}))$$

Performance of sound & global Sparrow

The Early Bird

Programs	LOC	Interval$_{vanilla}$		Interval$_{base}$		Spd↑$_1$	Mem↓$_1$	Interval$_{sparse}$						Spd↑$_2$	Mem↓$_2$
		Time	Mem	Time	Mem			Dep	Fix	Total	Mem	$\hat{D}(c)$	$\hat{U}(c)$		
gzip-1.2.4a	7K	772	240	14	65	55 x	73 %	2	1	3	63	2.4	2.5	5 x	3 %
bc-1.06	13K	1,270	276	96	126	13 x	54 %	4	3	7	75	4.6	4.9	14 x	40 %
tar-1.13	20K	12,947	881	338	177	38 x	80 %	6	2	8	93	2.9	2.9	42 x	47 %
less-382	23K	9,561	1,113	1,211	378	8 x	66 %	27	6	33	127	11.9	11.9	37 x	66 %
make-3.76.1	27K	24,240	1,391	1,893	443	13 x	68 %	16	5	21	114	5.8	5.8	90 x	74 %
wget-1.9	35K	44,092	2,546	1,214	378	36 x	85 %	8	3	11	85	2.4	2.4	110 x	78 %
screen-4.0.2	45K	∞	N/A	31,324	3,996	N/A	N/A	724	43	767	303	53.0	54.0	41 x	92 %
a2ps-4.14	64K	∞	N/A	3,200	1,392	N/A	N/A	31	9	40	353	2.6	2.8	80 x	75 %
bash-2.05a	105K	∞	N/A	1,683	1,386	N/A	N/A	45	22	67	220	3.0	3.0	25 x	84 %
lsh-2.0.4	111K	∞	N/A	45,522	5,266	N/A	N/A	391	80	471	577	21.1	21.2	97 x	89 %
sendmail-8.13.6	130K	∞	N/A	∞	N/A	N/A	N/A	517	227	744	678	20.7	20.7	N/A	N/A
nethack-3.3.0	211K	∞	N/A	∞	N/A	N/A	N/A	14,126	2,247	16,373	5,298	72.4	72.4	N/A	N/A
vim60	227K	∞	N/A	∞	N/A	N/A	N/A	17,518	6,280	23,798	5,190	180.2	180.3	N/A	N/A
emacs-22.1	399K	∞	N/A	∞	N/A	N/A	N/A	29,552	8,278	37,830	7,795	285.3	285.5	N/A	N/A
python-2.5.1	435K	∞	N/A	∞	N/A	N/A	N/A	9,677	1,362	11,039	5,535	108.1	108.1	N/A	N/A
linux-3.0	710K	∞	N/A	∞	N/A	N/A	N/A	26,669	6,949	33,618	20,529	76.2	74.8	N/A	N/A
gimp-2.6	959K	∞	N/A	∞	N/A	N/A	N/A	3,751	123	3,874	3,602	4.1	3.9	N/A	N/A
ghostscript-9.00	1,363K	∞	N/A	∞	N/A	N/A	N/A	14,116	698	14,814	6,384	9.7	9.7	N/A	N/A

none

spatial localization

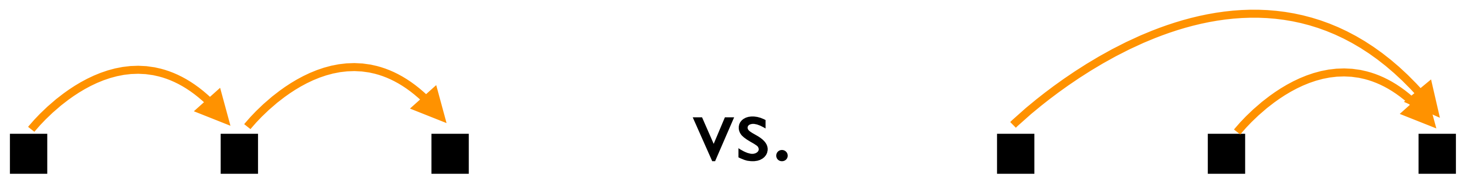
spatial+temporal localization

Existing Sparse Techniques
(developed mostly in dfa community)

- Different notion of data dependency

$$c_0 \overset{l}{\rightsquigarrow}_{\mathsf{du}} c_n \;\triangleq\; \begin{array}{l}\exists c_0 \ldots c_n \in \mathsf{Paths}, l \in \hat{\mathbb{L}}.\\ l \in \mathsf{D}(c_0) \cap \mathsf{U}(c_n) \wedge \forall i \in (0, n). l \notin \underline{\mathsf{D}_{\mathsf{always}}}(c_i)\end{array}$$

  - fail to preserve the original accuracy



- Not general for arbitrary analysis for full C
  - tightly coupled with particular analysis (e.g. pointer analysis for "simple" subsets of C)

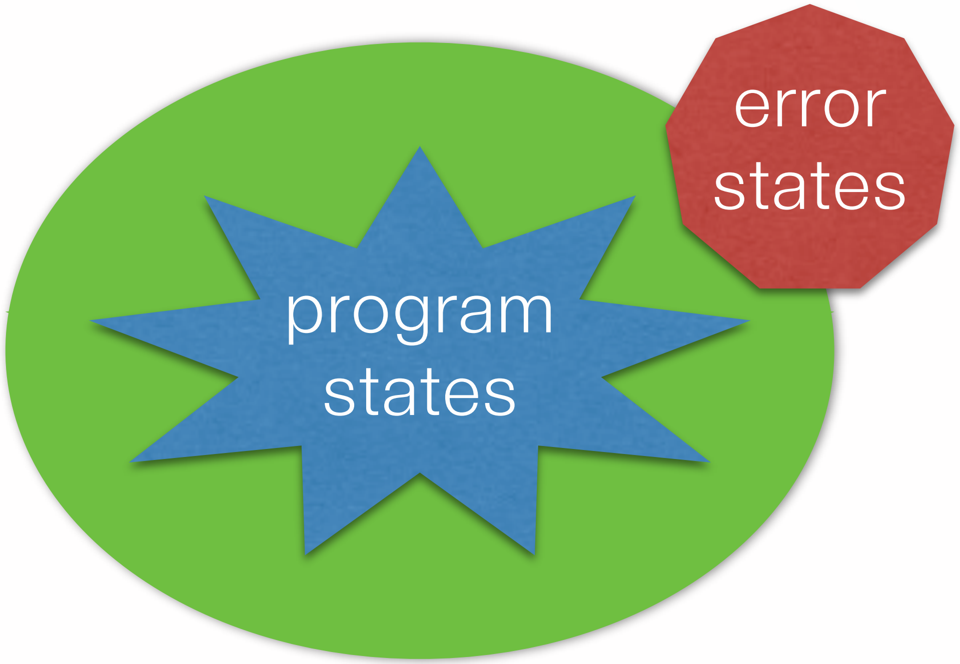
Improving Precision

Key Idea: Selectivity

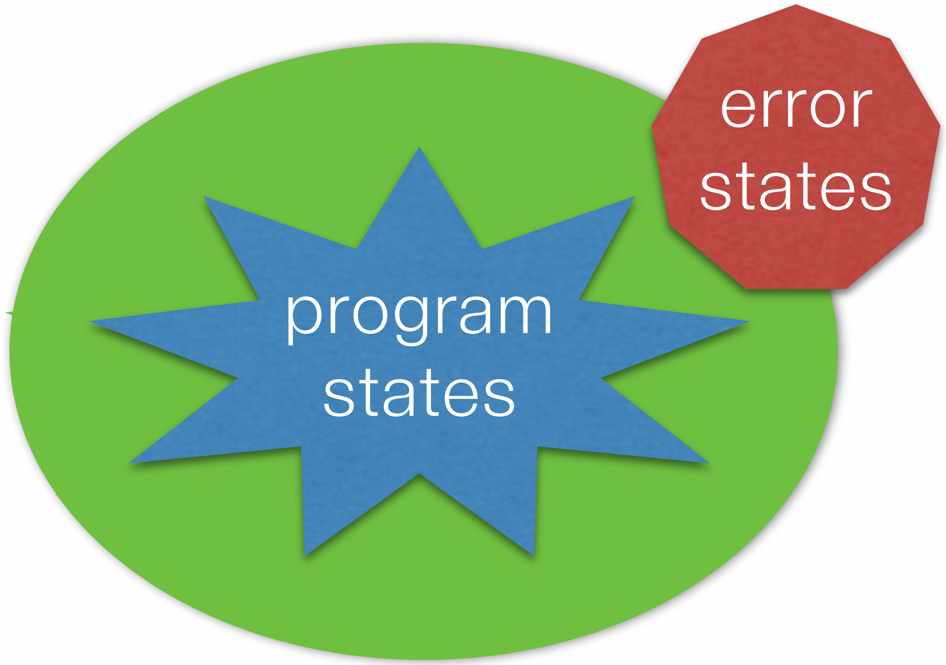
“X-sensitivity at Right Moment”

Selective Context-Sensitivity

- context-sensitivity only when/where it matters



vs.

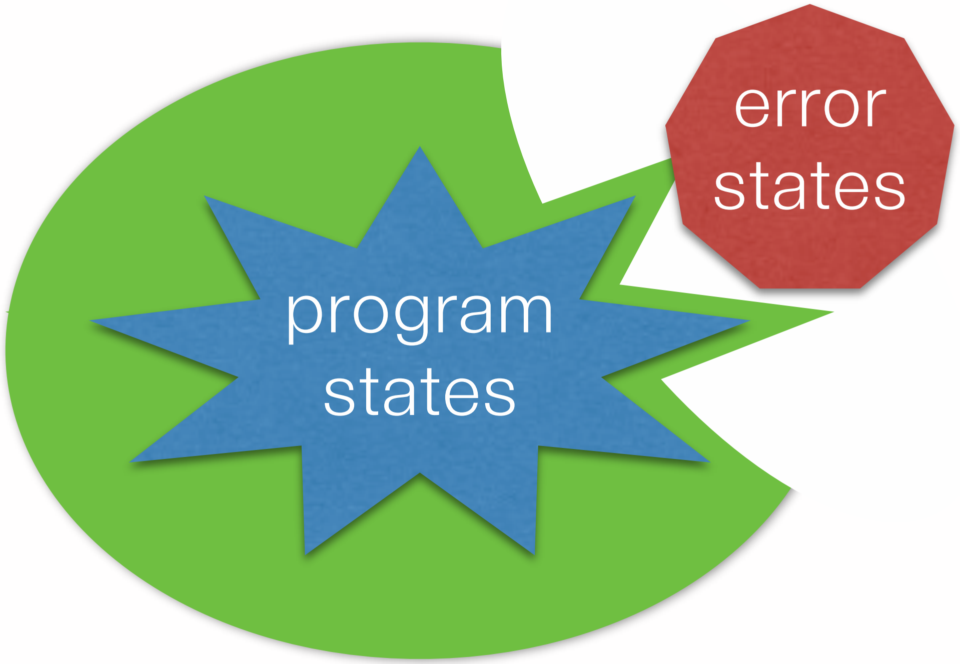


our method: **24%** / **28%**

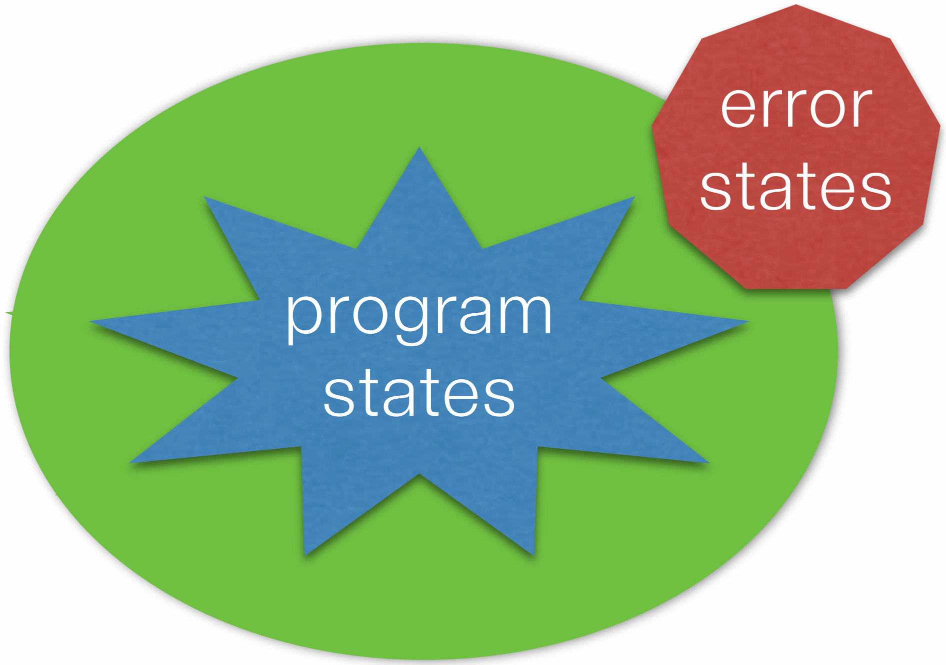
3-CFA: **24%** / **1300%**

Selective Context-Sensitivity

- context-sensitivity only when/where it matters



vs.

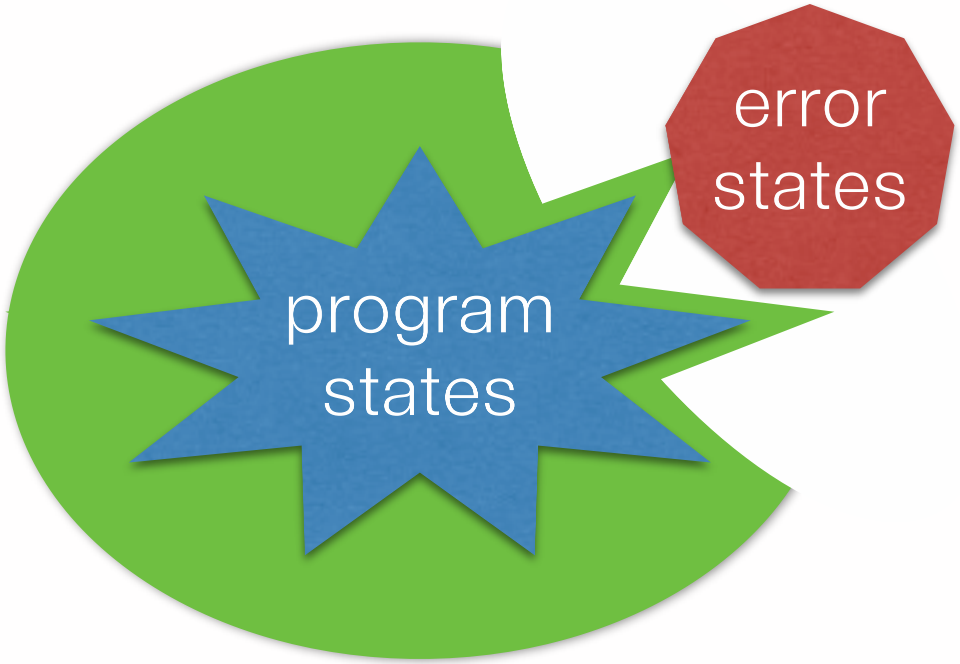


our method: **24%** / **28%**

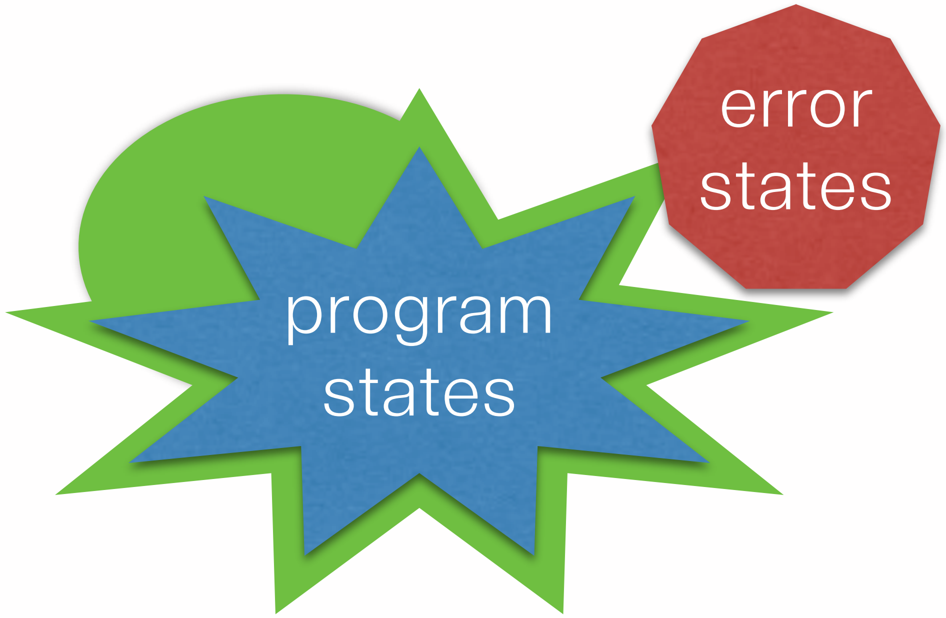
3-CFA: **24%** / **1300%**

Selective Context-Sensitivity

- context-sensitivity only when/where it matters



vs.



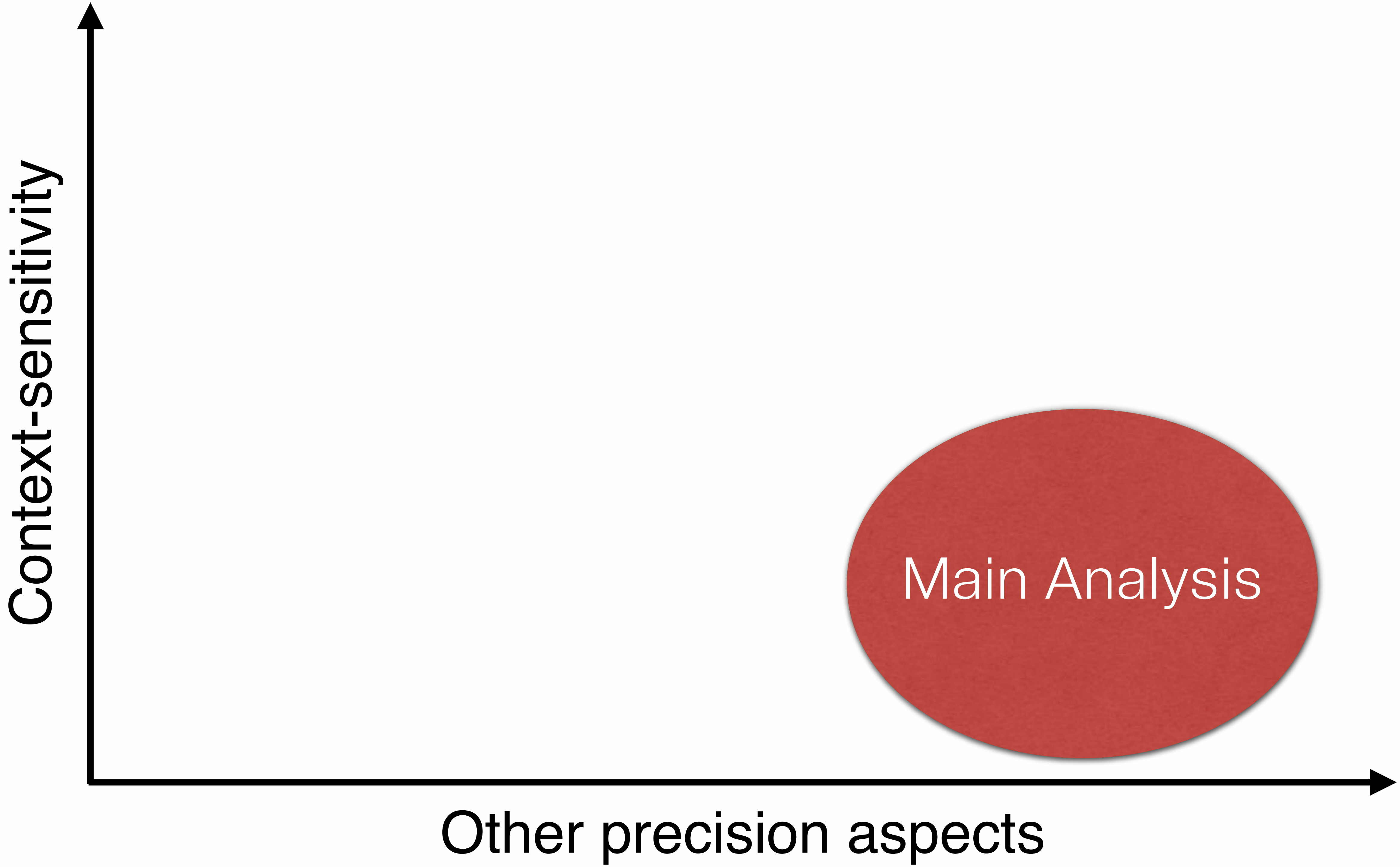
our method: **24%** / **28%**

3-CFA: **24%** / **1300%**

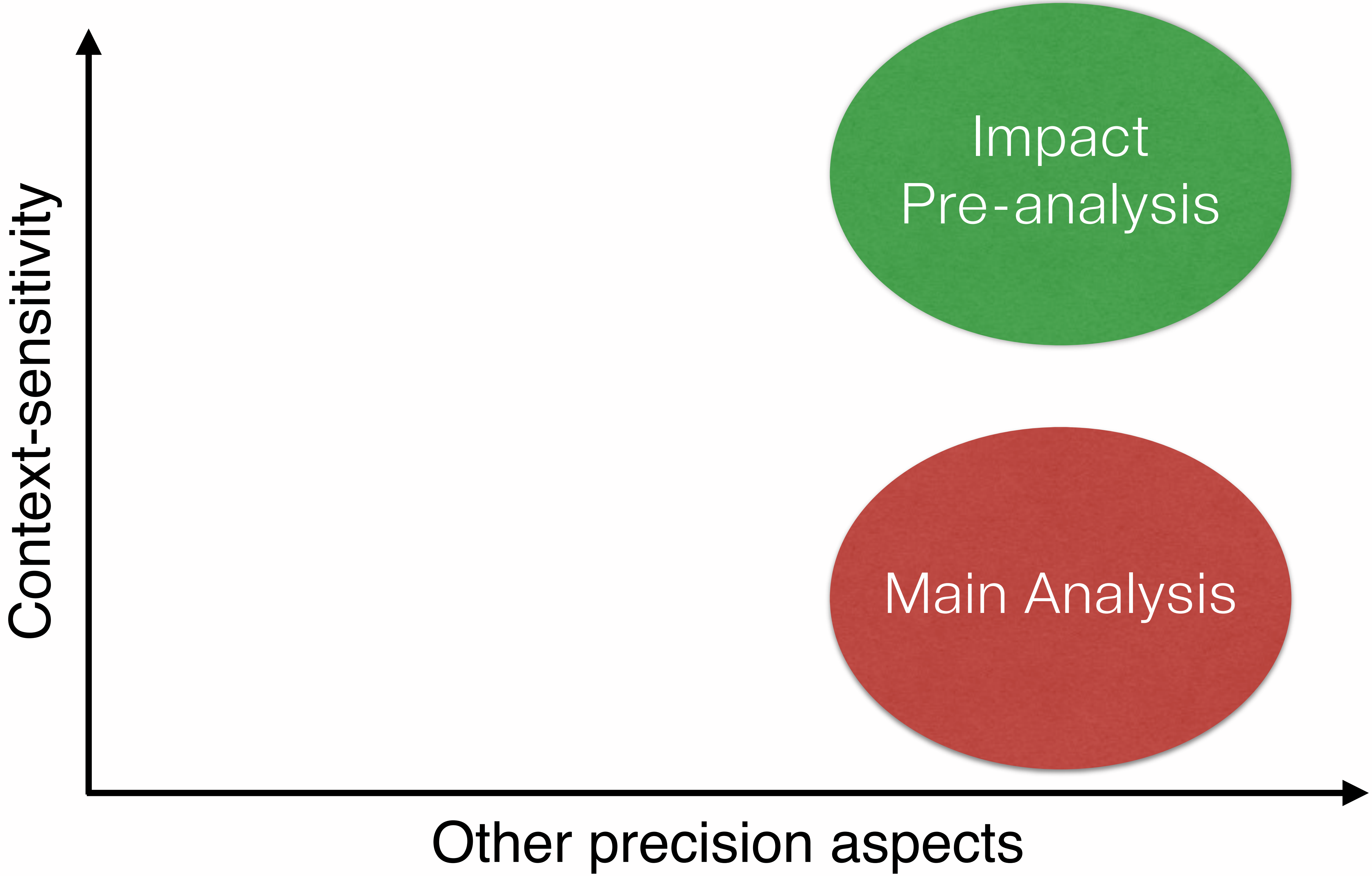
Key Idea: Impact Pre-Analysis

- Estimate the impact of X-sensitivity on main analysis
  - fully X-sensitive
  - but, approximated in other precision aspects

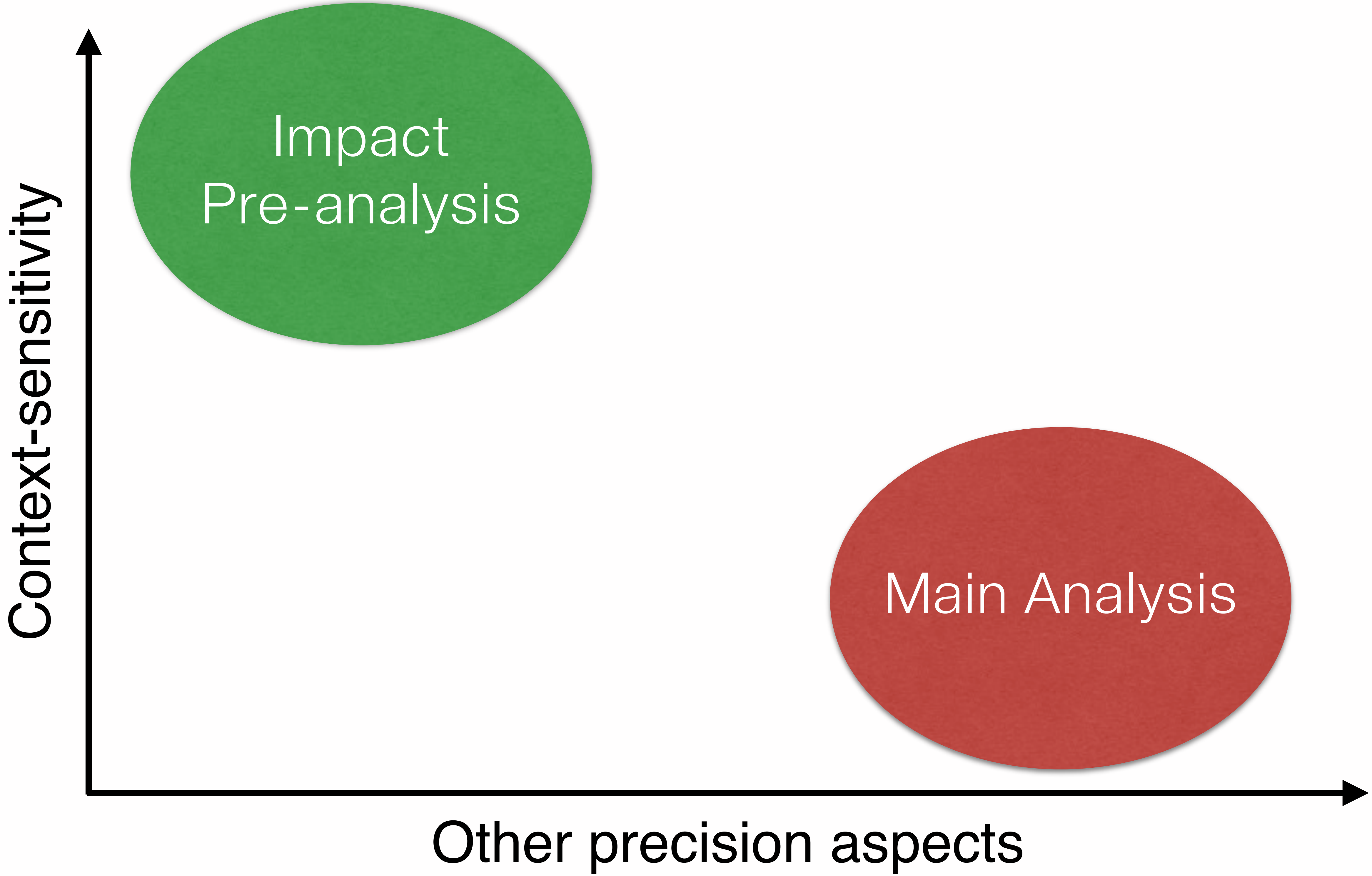
Key Idea: Impact Pre-Analysis



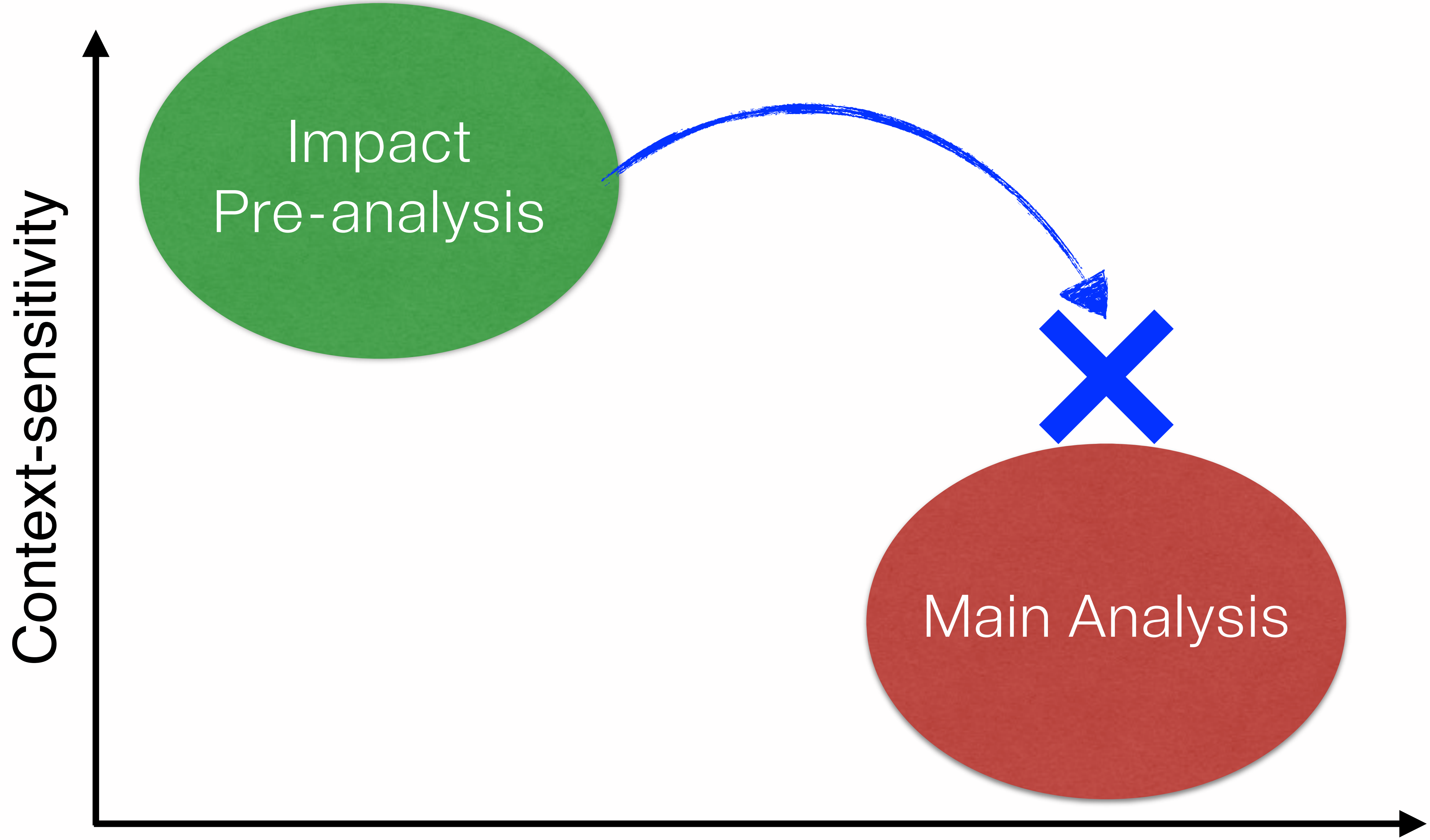
Key Idea: Impact Pre-Analysis



Key Idea: Impact Pre-Analysis

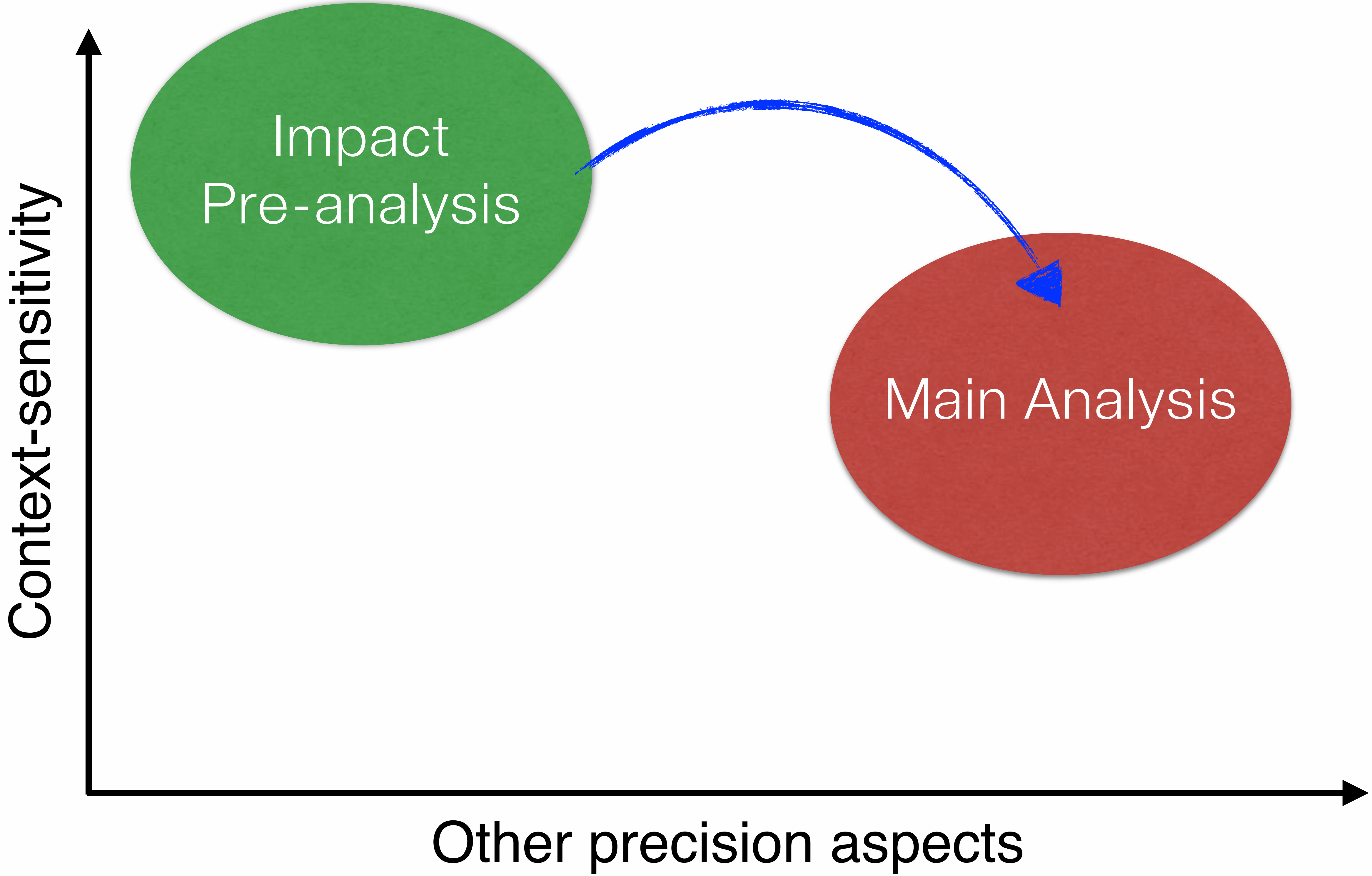


Key Idea: Impact Pre-Analysis

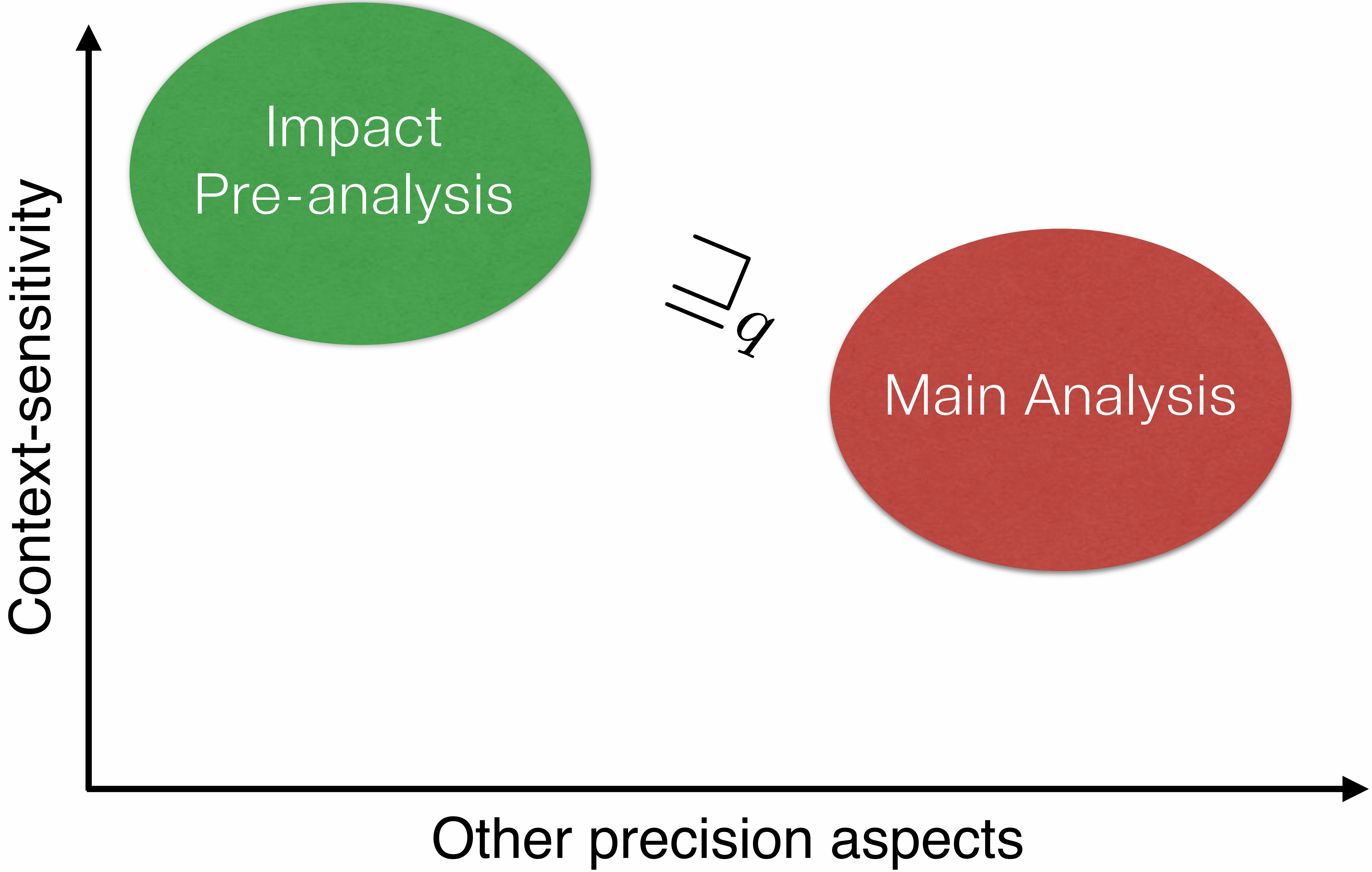


Other precision aspects

Key Idea: Impact Pre-Analysis



Impact Realization



Two Instance Analyses

- Selective context-sensitivity
- Selective relational analysis

Selective Context-Sensitivity

Example Program

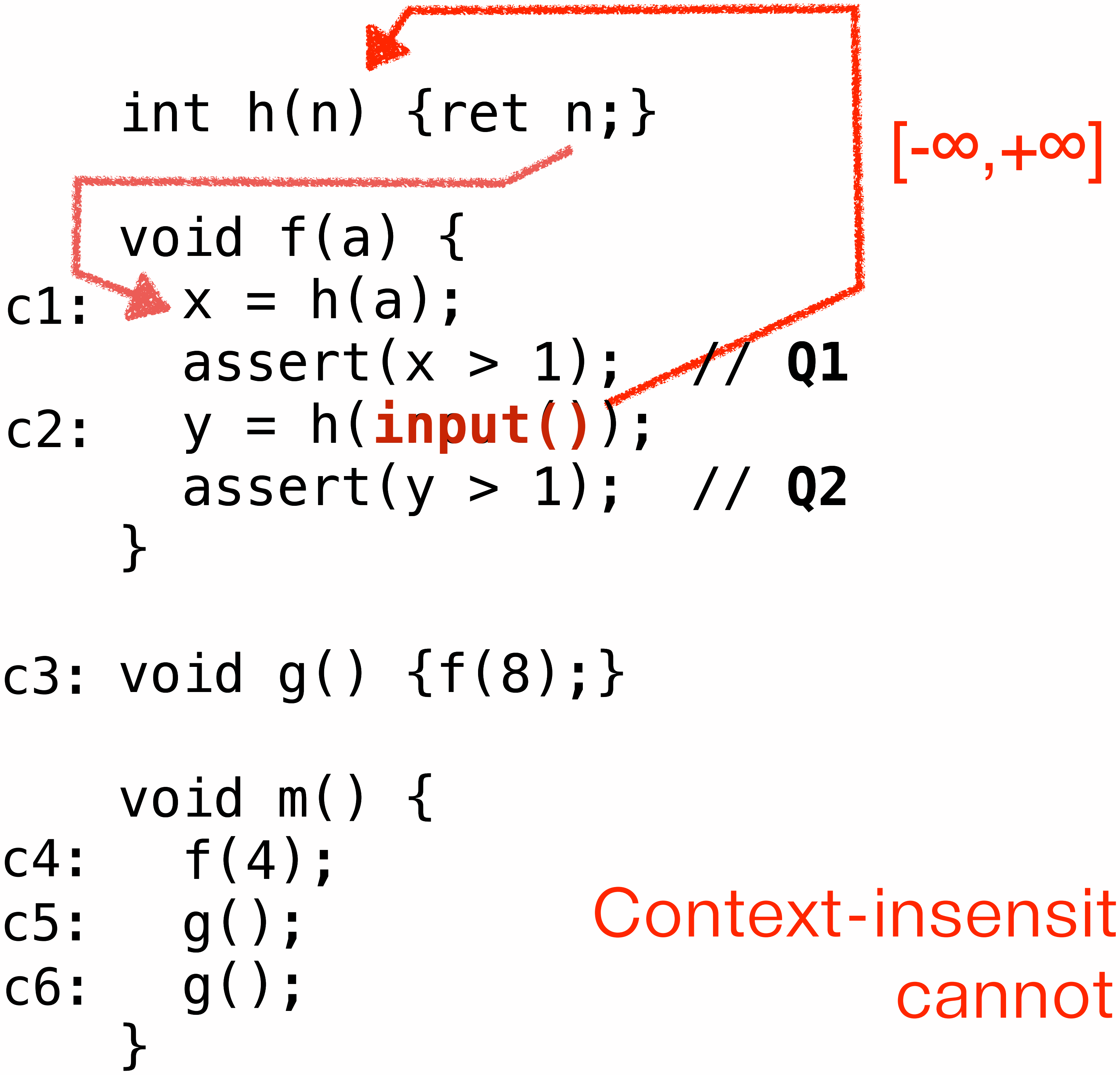
```
    int h(n) {ret n;}

    void f(a) {
c1:   x = h(a);
      assert(x > 1);  // Q1  ← always holds
c2:   y = h(input());
      assert(y > 1);  // Q2  ← does not always hold
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```

Context-Insensitivity

```
    int h(n) {ret n;}

    void f(a) {
c1:   x = h(a);
      assert(x > 1);  // Q1
c2:   y = h(input());
      assert(y > 1);  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```

Context-insensitive interval analysis cannot prove Q1

Context-Insensitivity



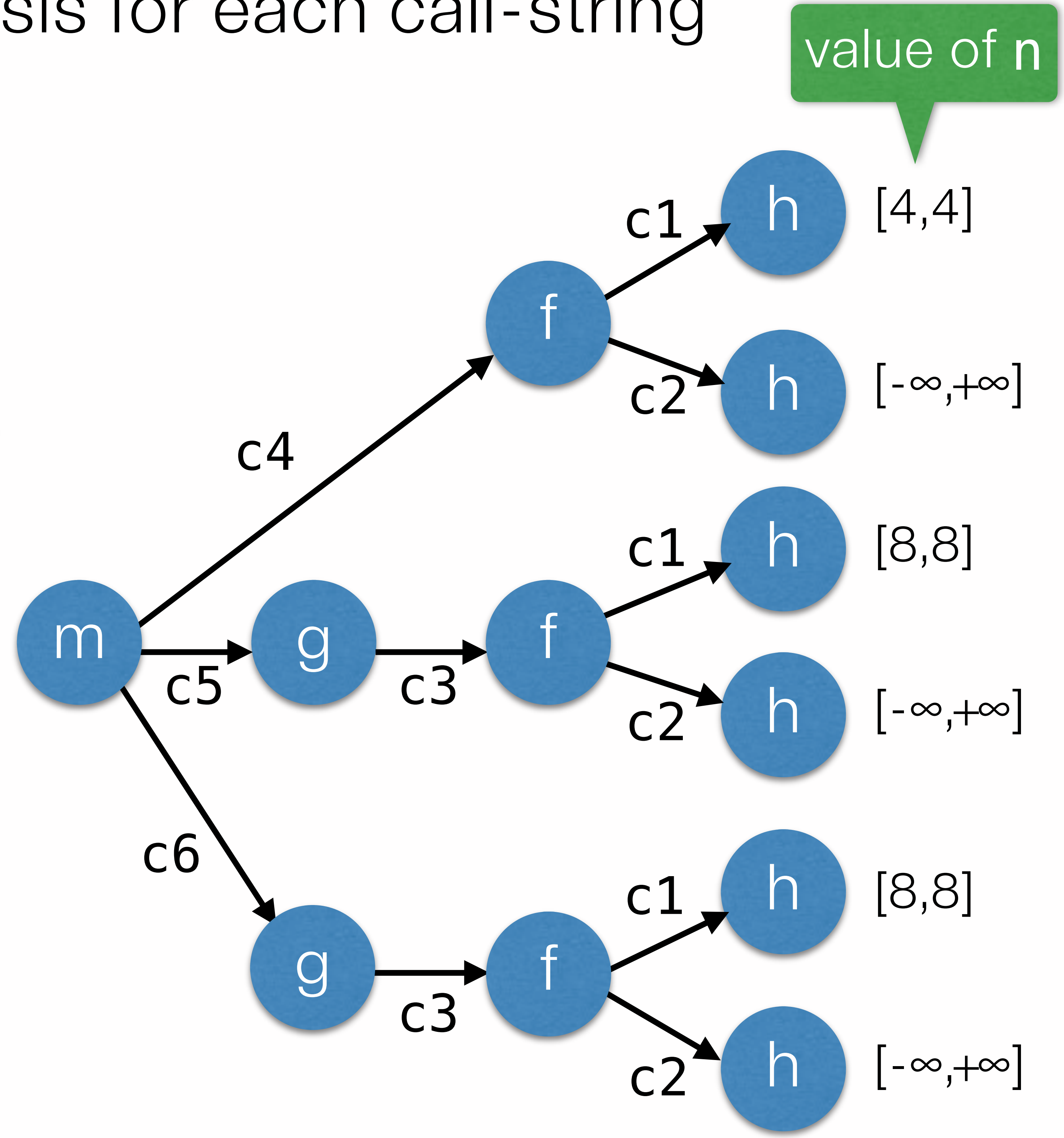
```
    int h(n) {ret n;}

    void f(a) {
c1:   x = h(a);
      assert(x > 1);  // Q1
c2:   y = h(input());
      assert(y > 1);  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```

Context-insensitive interval analysis cannot prove Q1

Context-Sensitivity: 3-CFA

Separate analysis for each call-string

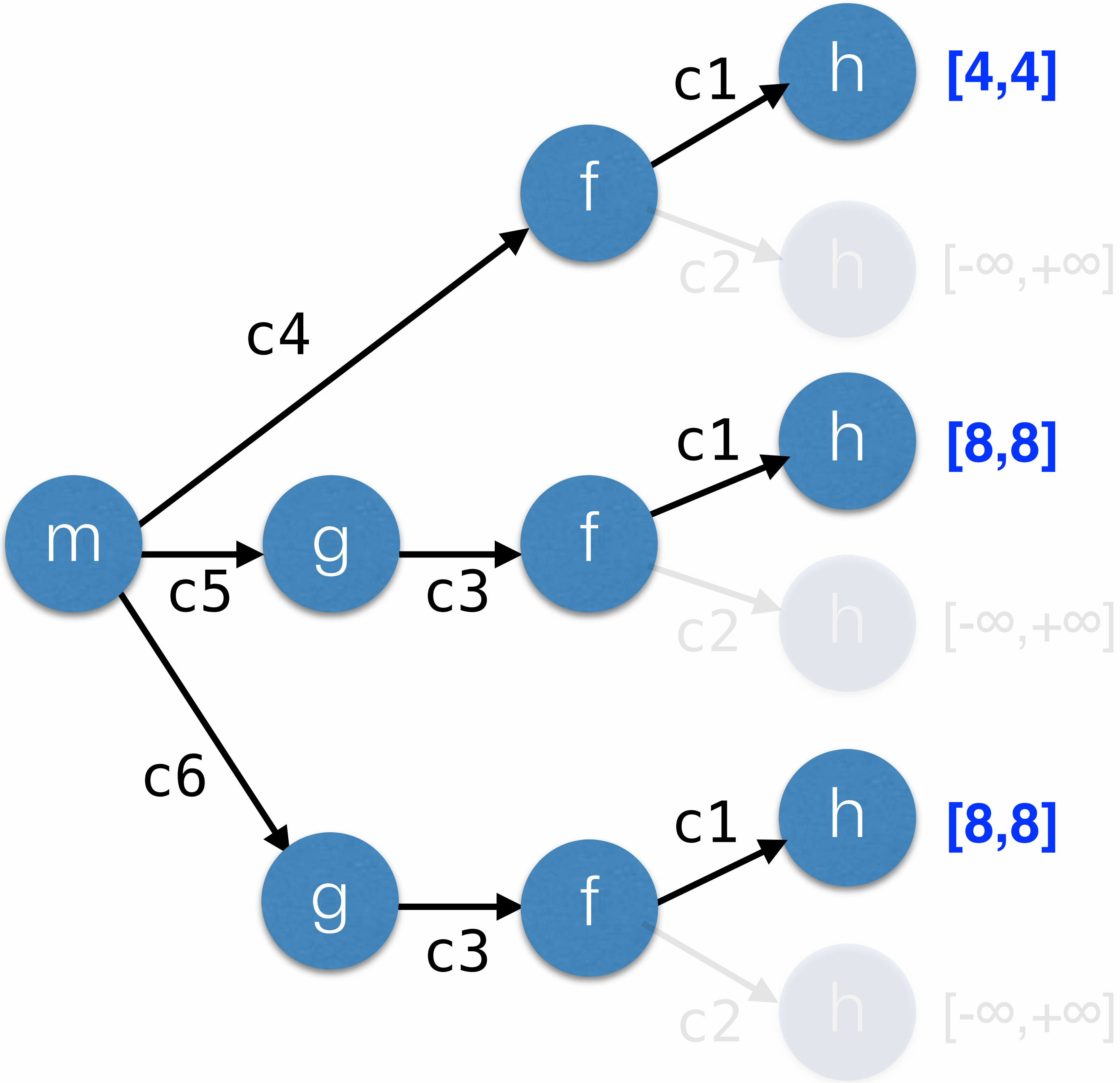
```
    int h(n) {ret n;}

    void f(a) {
c1:   x = h(a);
      assert(x > 1);  // Q1
c2:   y = h(input());
      assert(y > 1);  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```



Context-Sensitivity: 3-CFA

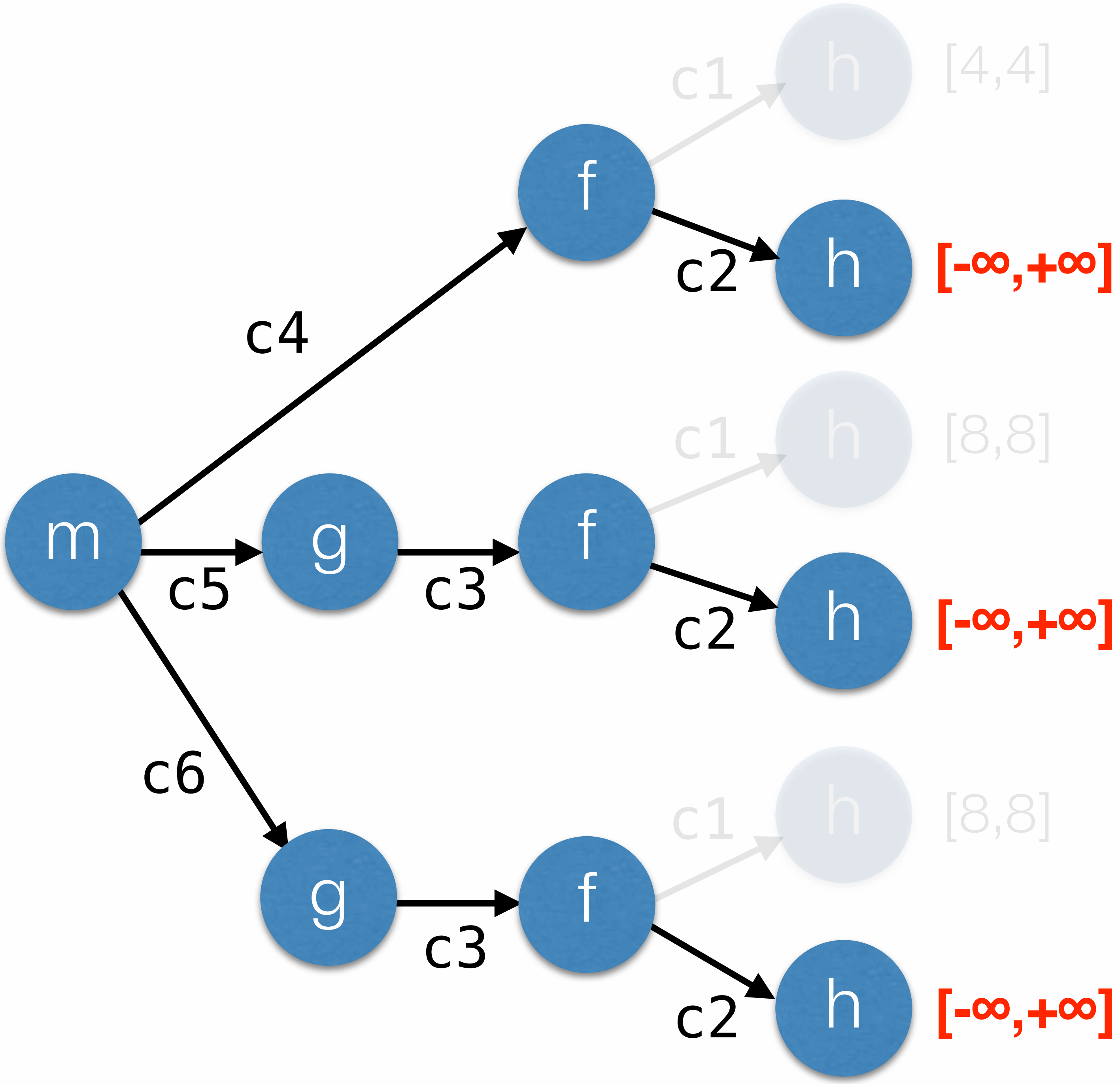
Separate analysis for each call-string

```
    int h(n) {ret n;}

    void f(a) {
c1:   x = h(a);
      assert(x > 1);  // Q1
c2:   y = h(input());
      assert(y > 1);  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```



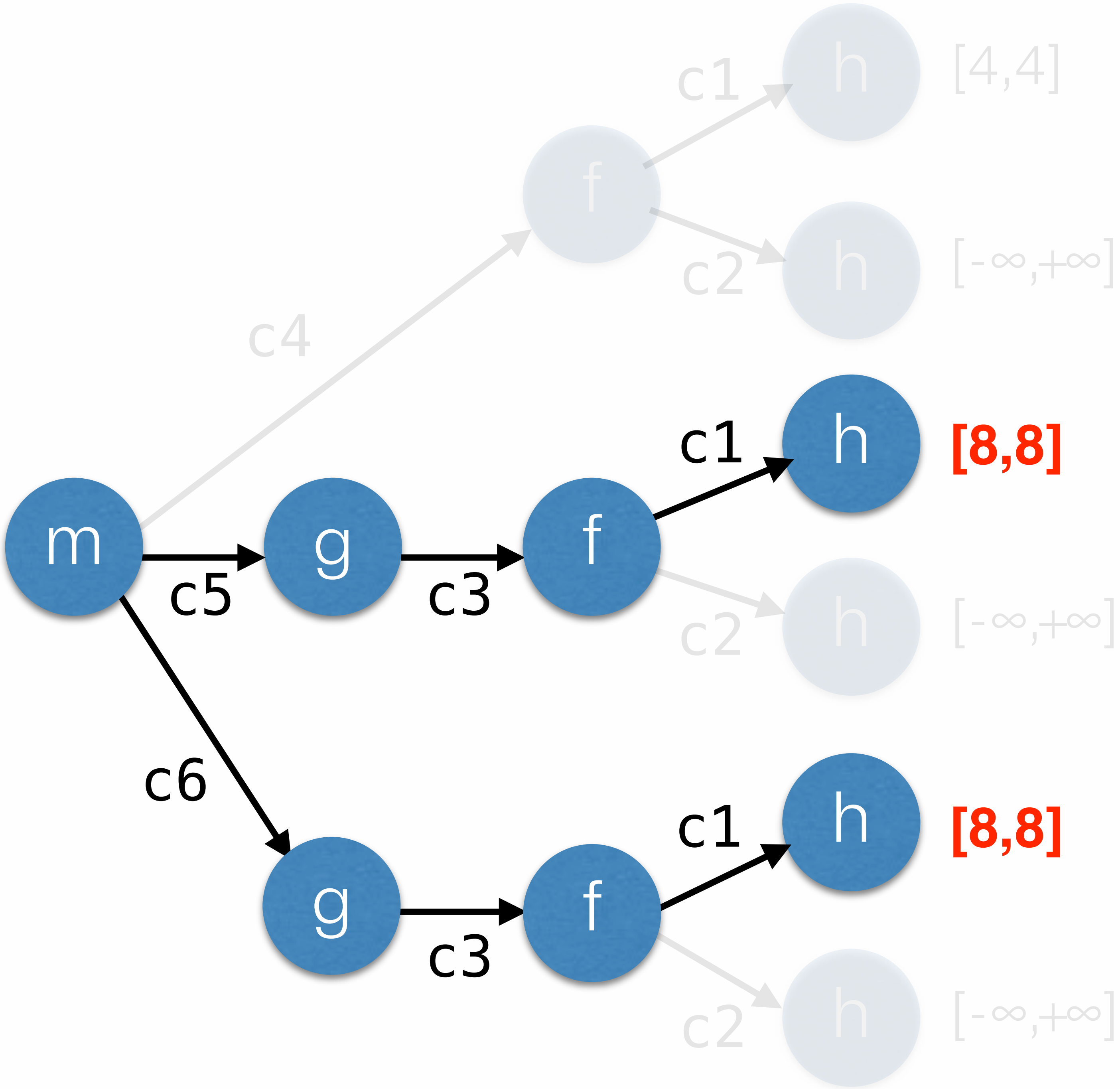
Problems of k-CFA

```
    int h(n) {ret n;}

    void f(a) {
c1:   x = h(a);
      assert(x > 1);  // Q1
c2:   y = h(input());
      assert(y > 1);  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```



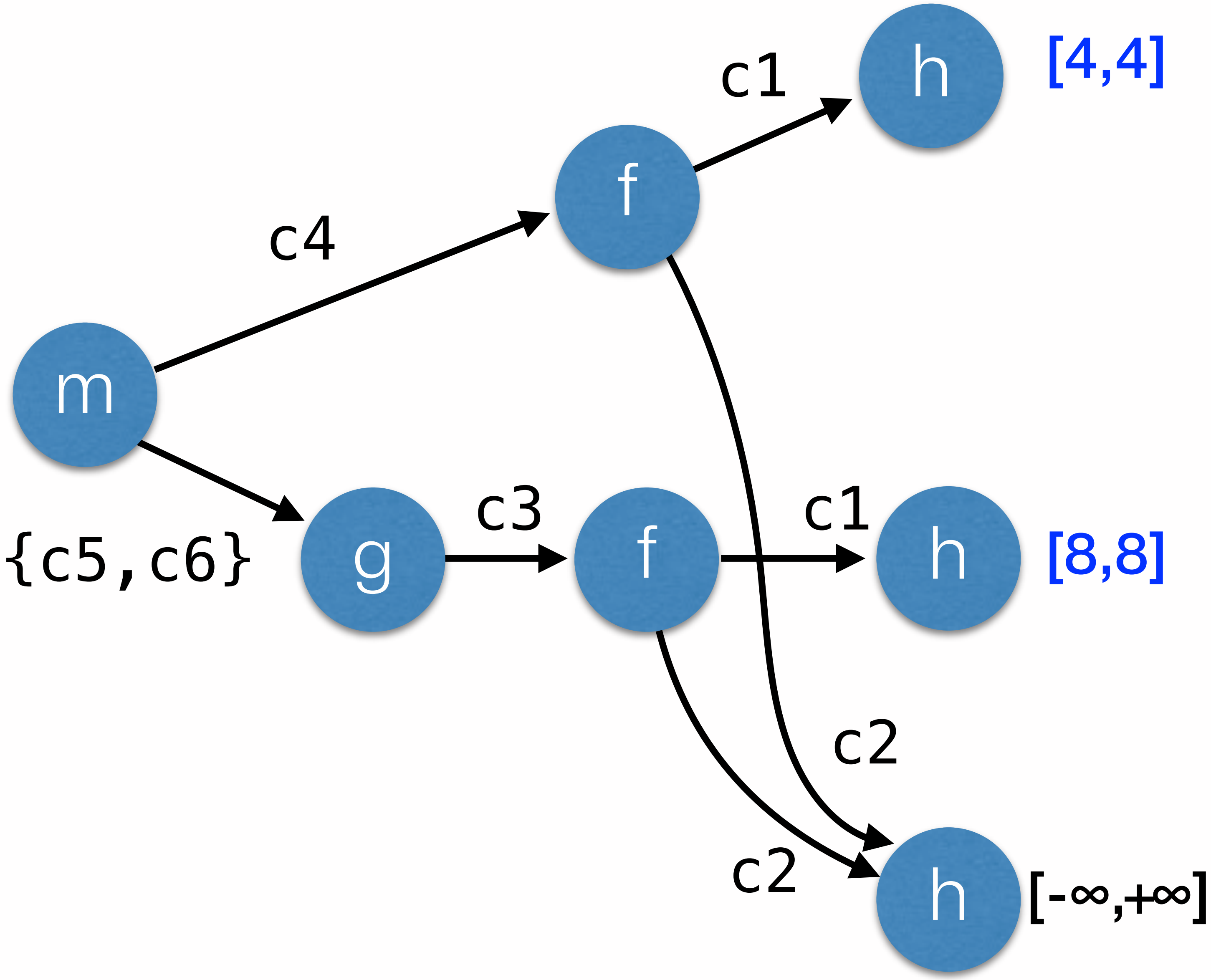
Problems of k-CFA

```
    int h(n) {ret n;}

    void f(a) {
c1:   x = h(a);
      assert(x > 1);  // Q1
c2:   y = h(input());
      assert(y > 1);  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```



Our Selective Context-Sensitivity

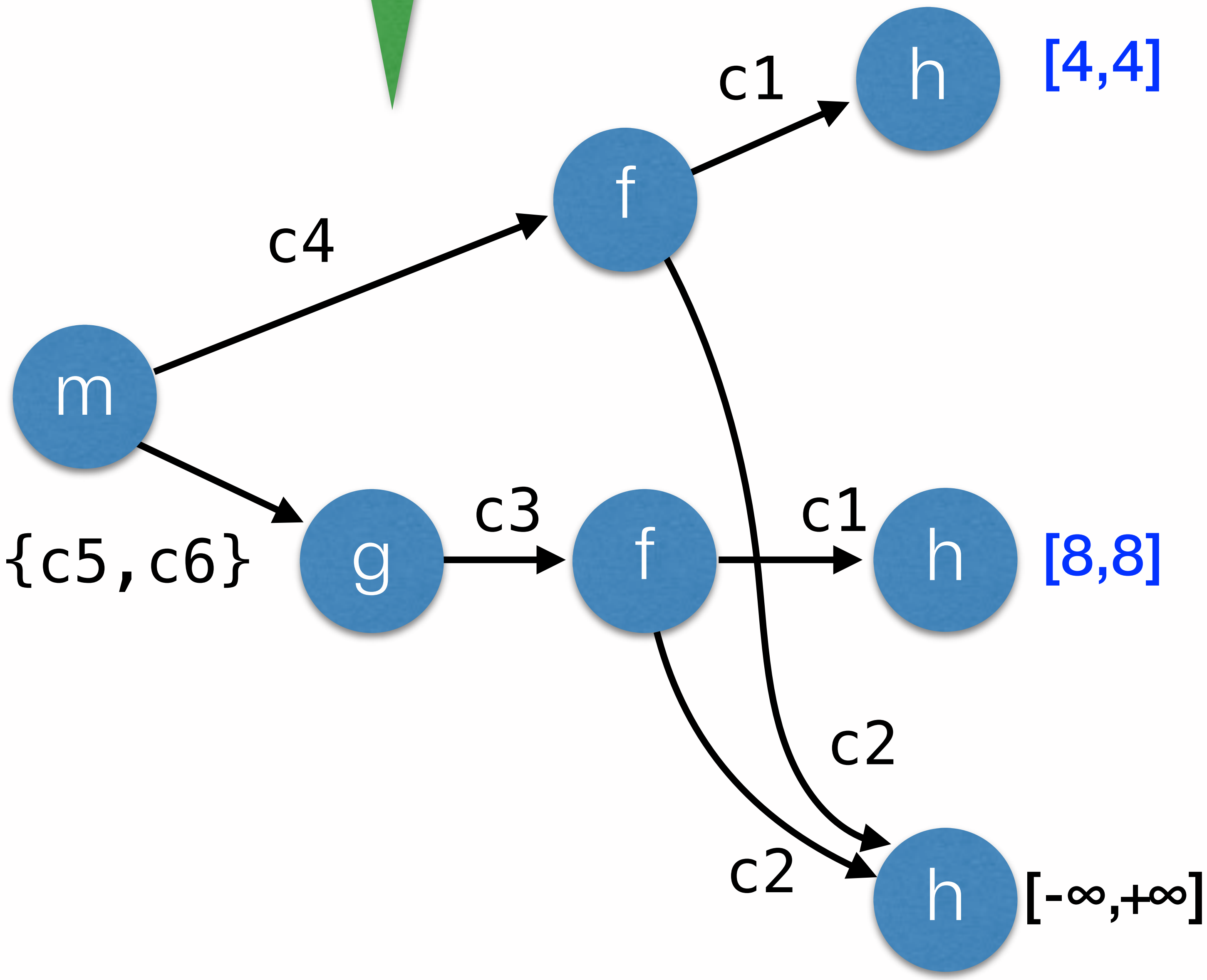
```
    int h(n) {ret n;}

    void f(a) {
c1:   x = h(a);
      assert(x > 1);  // Q1
c2:   y = h(input());
      assert(y > 1);  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```



Our Selective Context-Sensitivity

```
     int h(n) {ret n;}

     void f(a) {
c1:    x = h(a);
       assert(x > 1);  // Q1
c2:    y = h(input());
       assert(y > 1);  // Q2
     }

c3: void g() {f(8);}

     void m() {
c4:    f(4);
c5:    g();
c6:    g();
     }
```

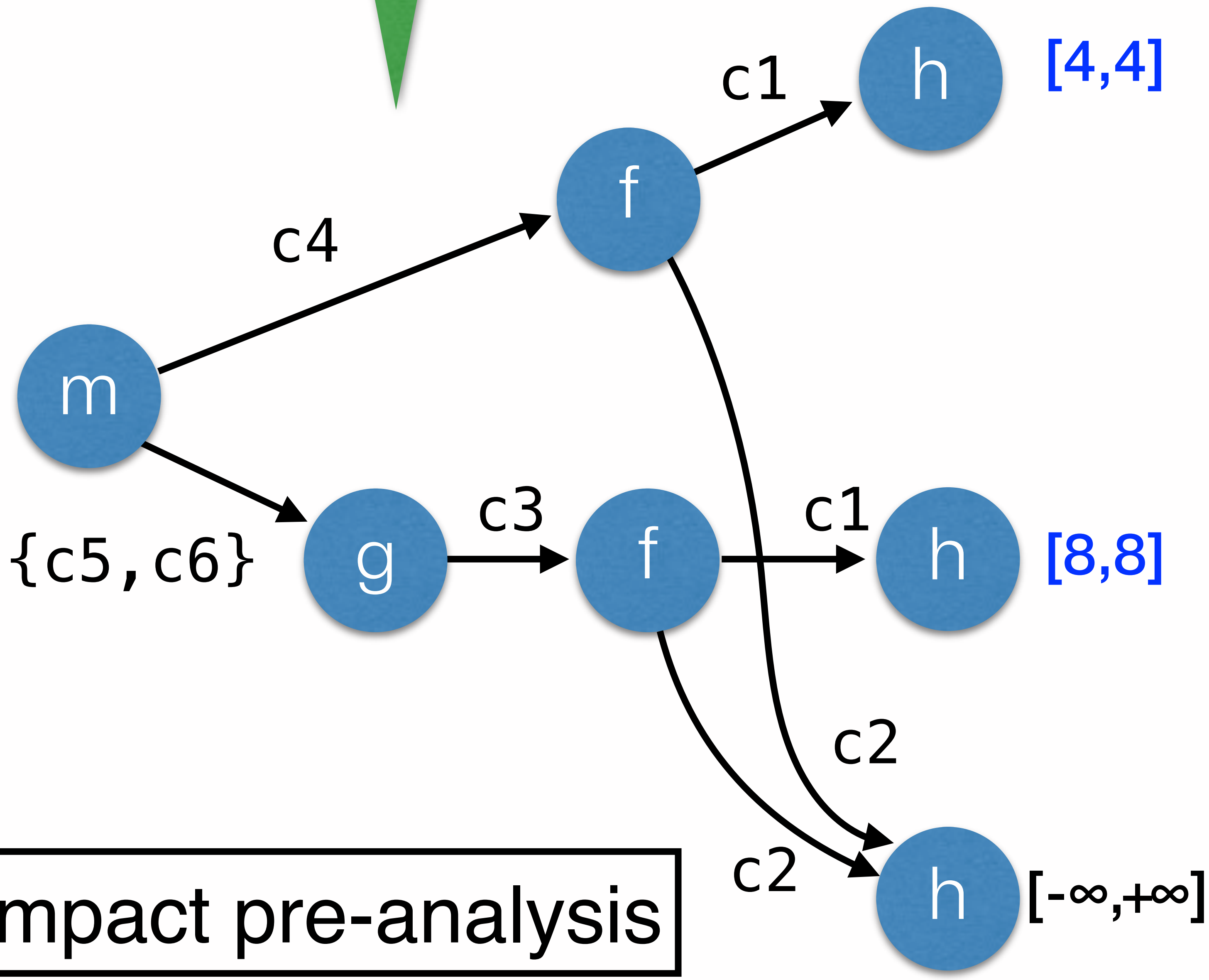
Challenge: How to infer this selective context-sensitivity?



Our Selective Context-Sensitivity

```
    int h(n) {ret n;}

    void f(a) {
c1:   x = h(a);
      assert(x > 1);  // Q1
c2:   y = h(input());
      assert(y > 1);  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```

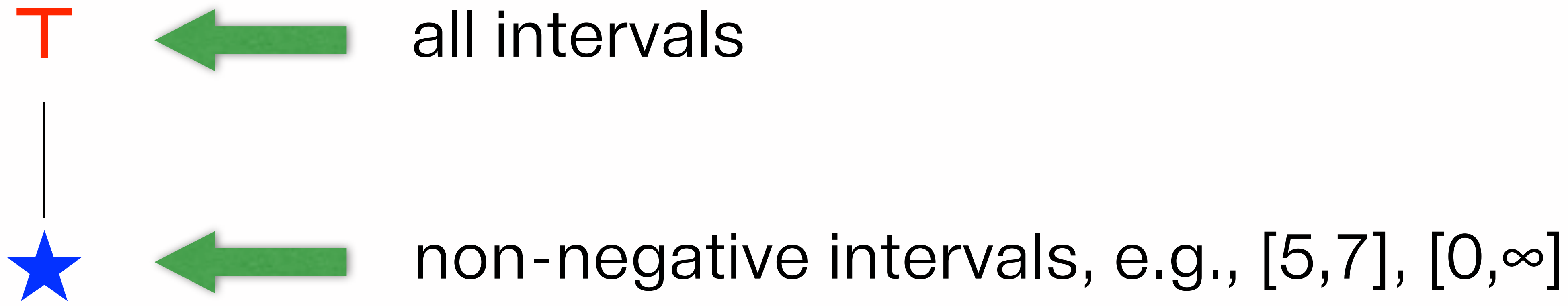
Challenge: How to infer this selective context-sensitivity?



Our solution: Impact pre-analysis

Impact Pre-Analysis

- Full context-sensitivity
- Approximate the interval domain

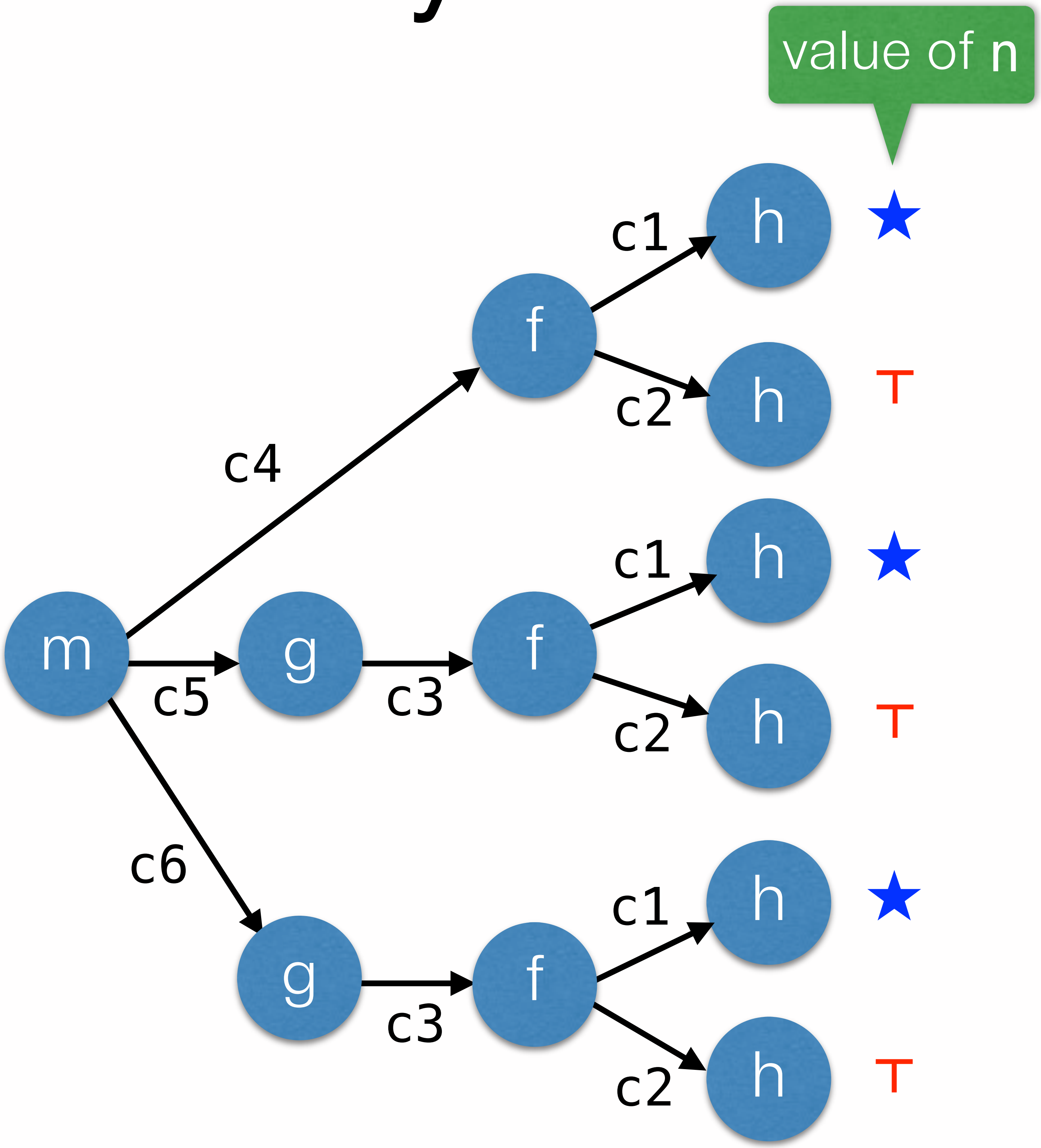


$\top$ all intervals

★ non-negative intervals, e.g., [5,7], [0,∞]

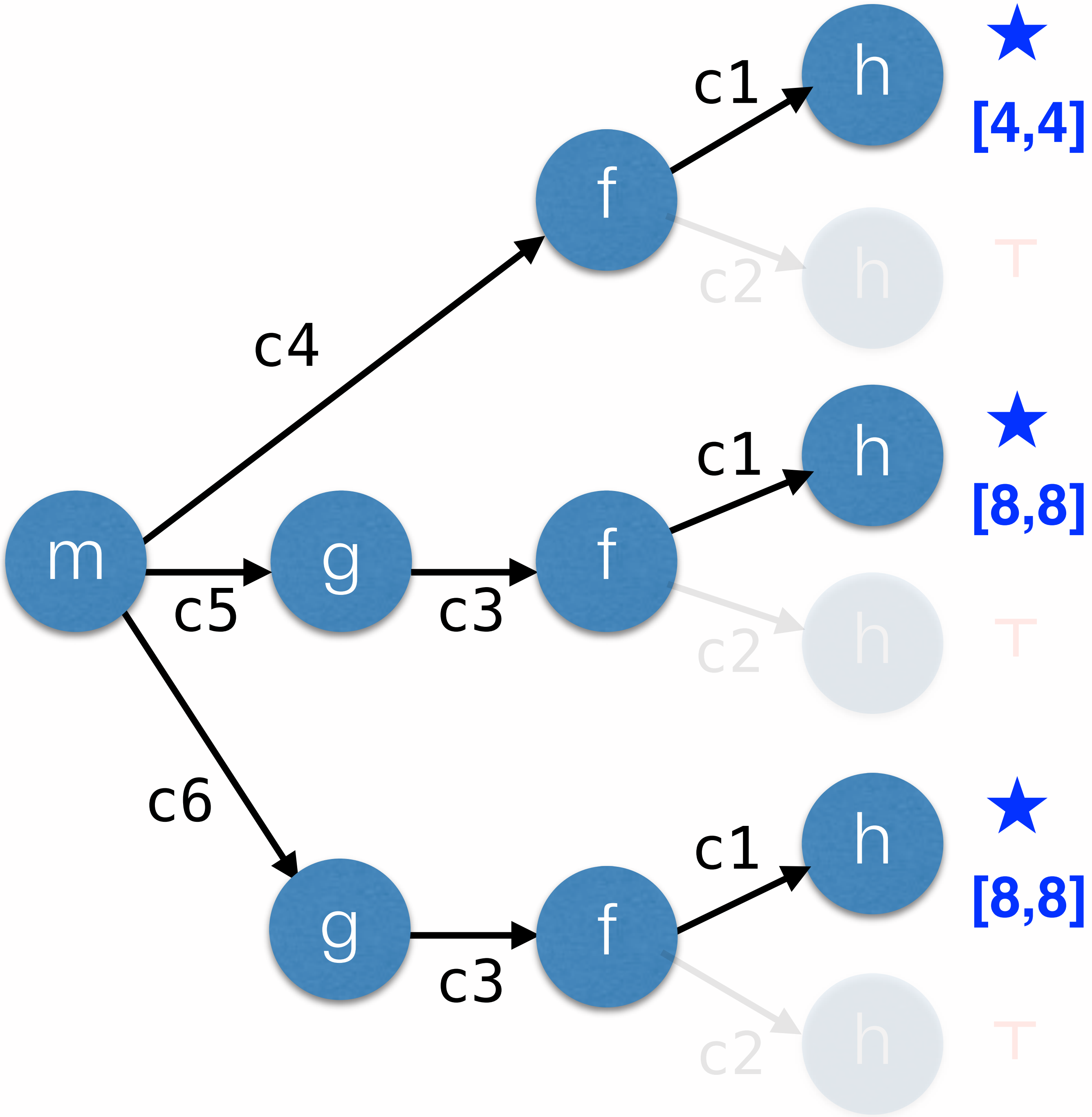
Impact Pre-Analysis

```
    int h(n) {ret n;}

    void f(a) {
c1:   x = h(a);
      assert(x > 1);  // Q1
c2:   y = h(input());
      assert(y > 1);  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```



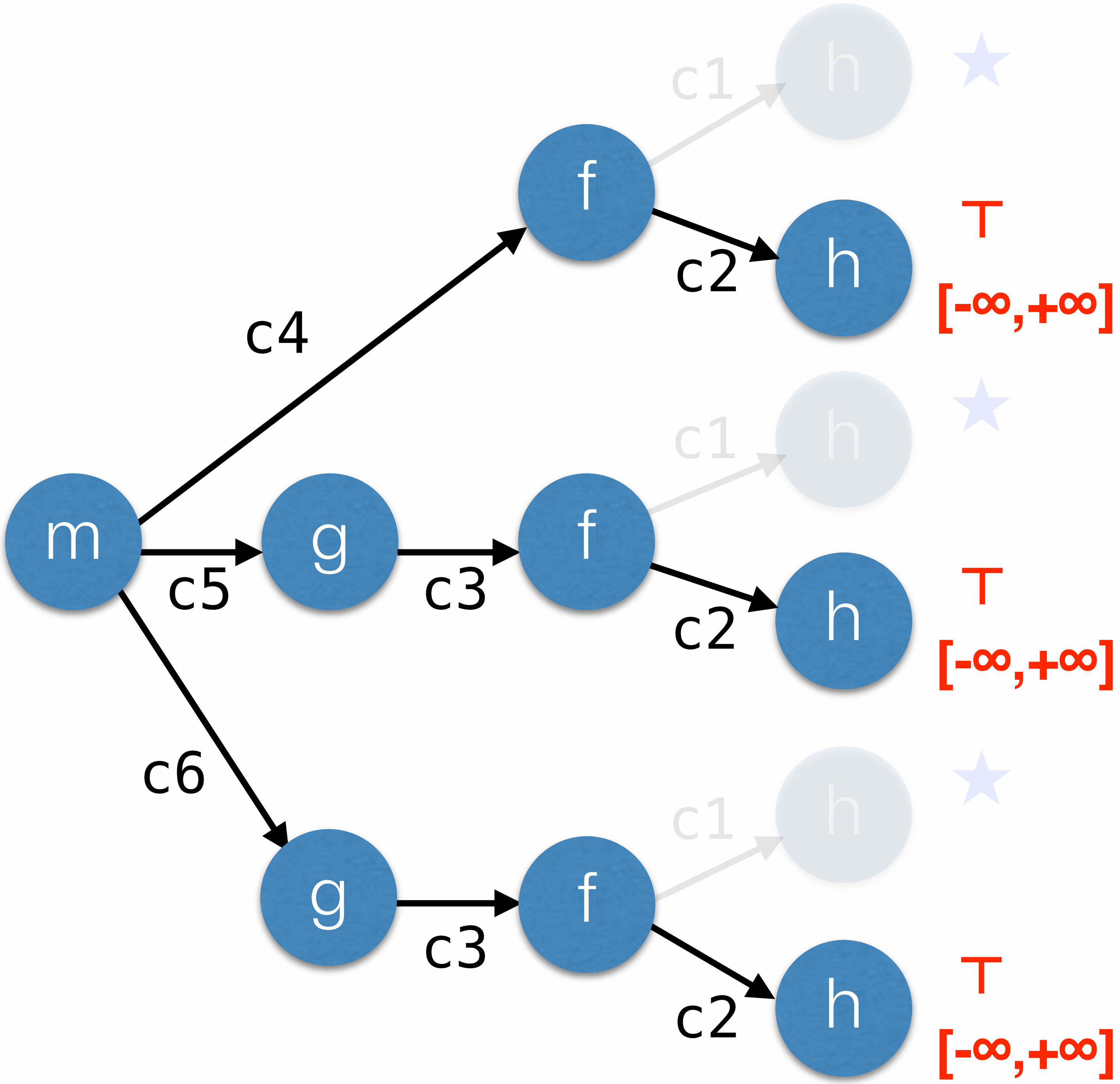
Impact Pre-Analysis

```
    int h(n) {ret n;}

    void f(a) {
c1:   x = h(a);
      assert(x > 1);  // Q1
c2:   y = h(input());
      assert(y > 1);  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```



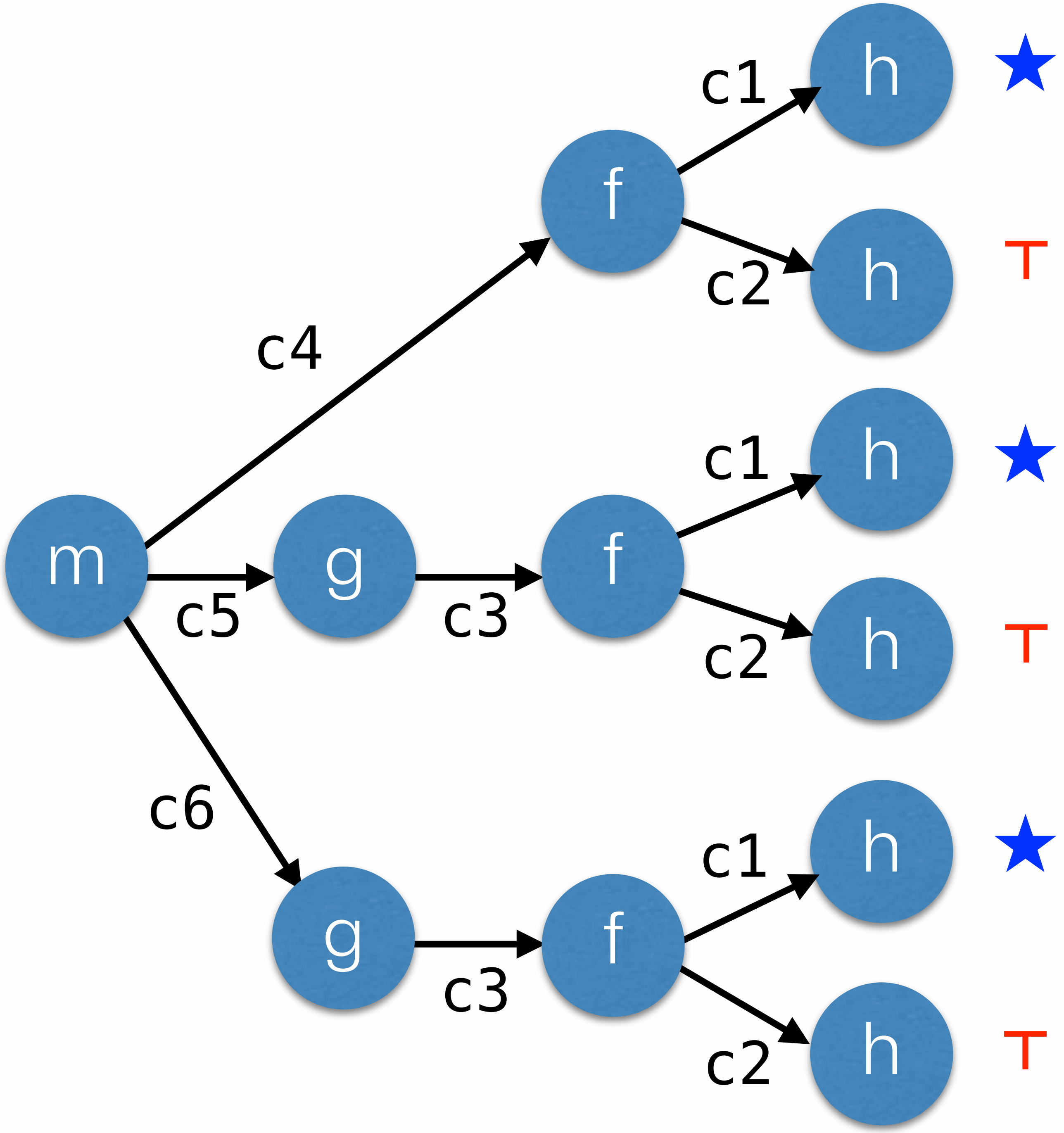
Impact Pre-Analysis

```
    int h(n) {ret n;}

    void f(a) {
c1:   x = h(a);
      assert(x > 1);  // Q1
c2:   y = h(input());
      assert(y > 1);  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```



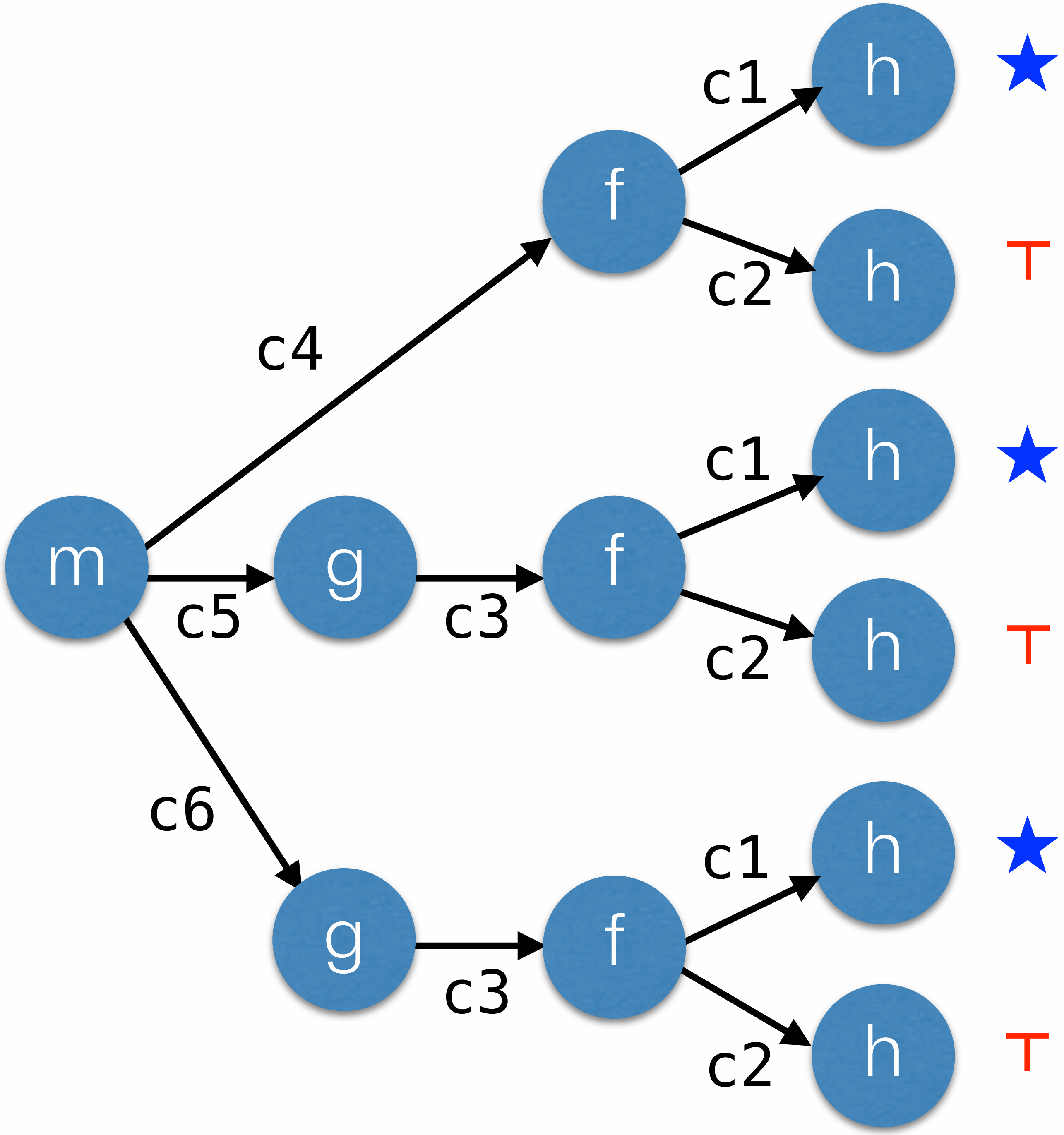
1. Collect queries whose expressions are assigned with ★

```
    int h(n) {ret n;}

    void f(a) {
c1:   x = h(a);
      assert(x > 1);  // Q1
c2:   y = h(input());
      assert(y > 1);  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```



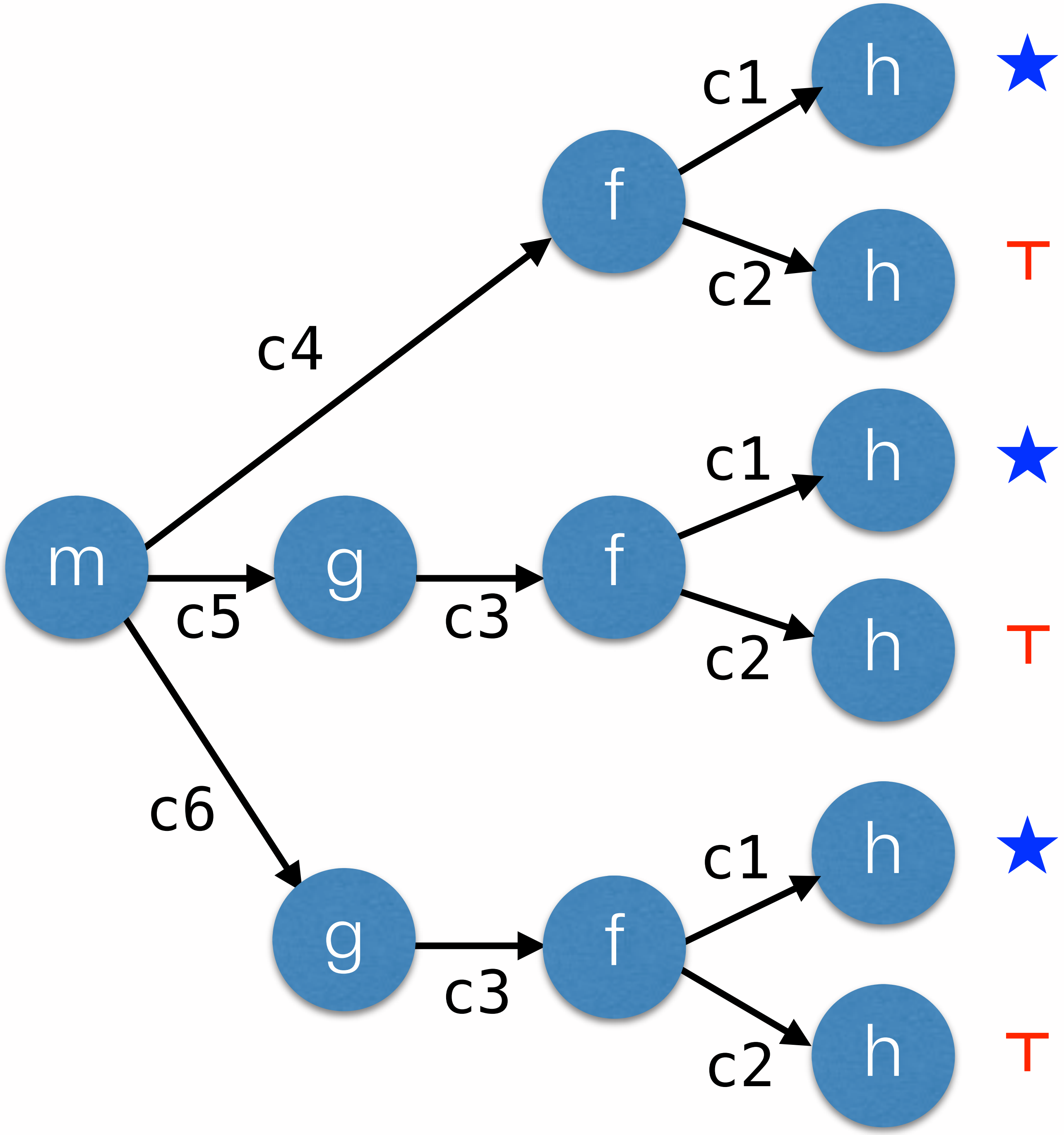
1. Collect queries whose expressions are assigned with ★

```
    int h(n) {ret n;}

    void f(a) {
c1: ★ x = h(a);
      assert(x > 1);  // Q1
c2:   y = h(input());
      assert(y > 1);  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```



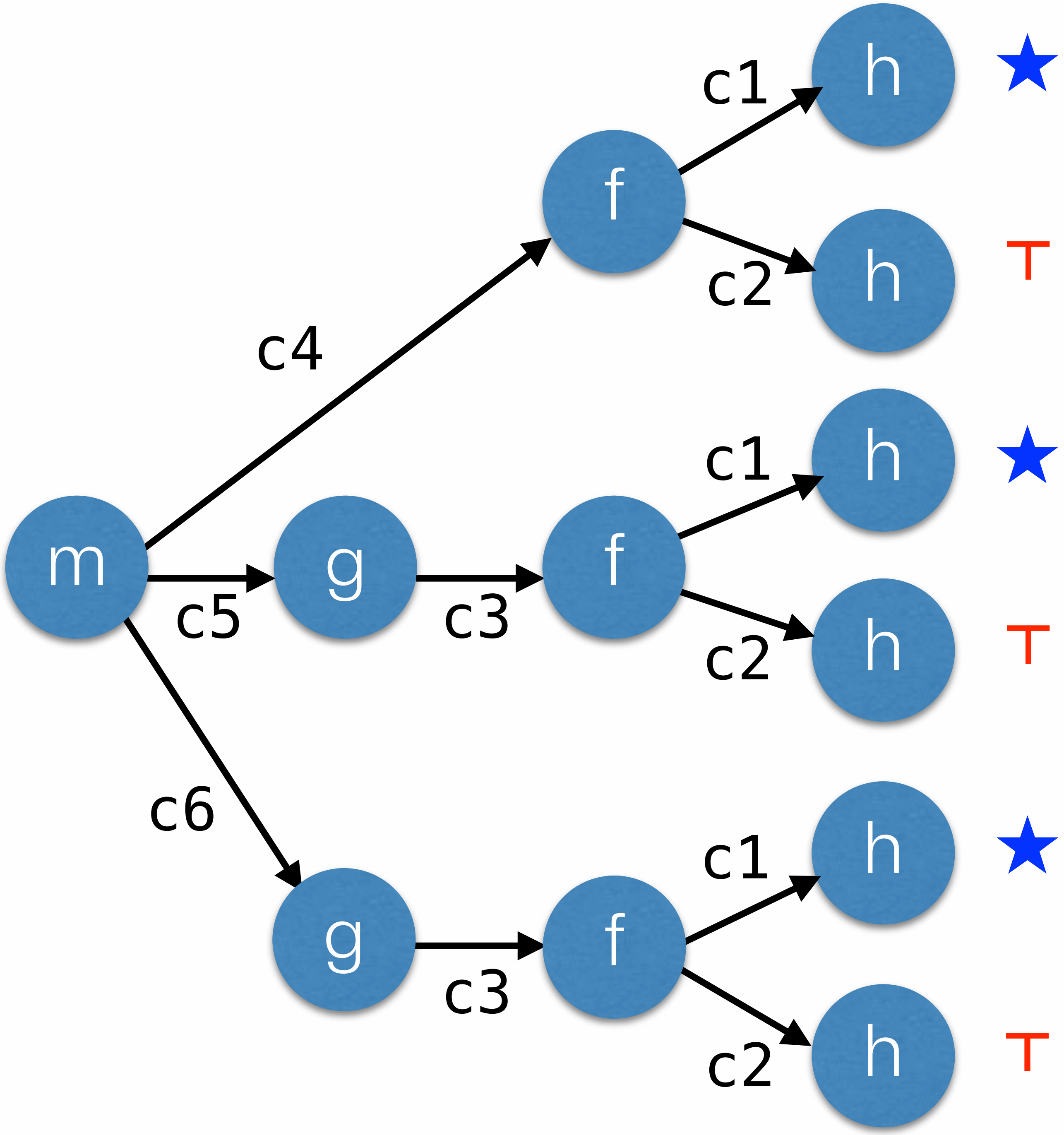
1. Collect queries whose expressions are assigned with ★

```
    int h(n) {ret n;}

    void f(a) {
c1: ★ x = h(a);
      assert(x > 1);  // Q1
c2: ⊤ y = h(input());
      assert(y > 1);  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```



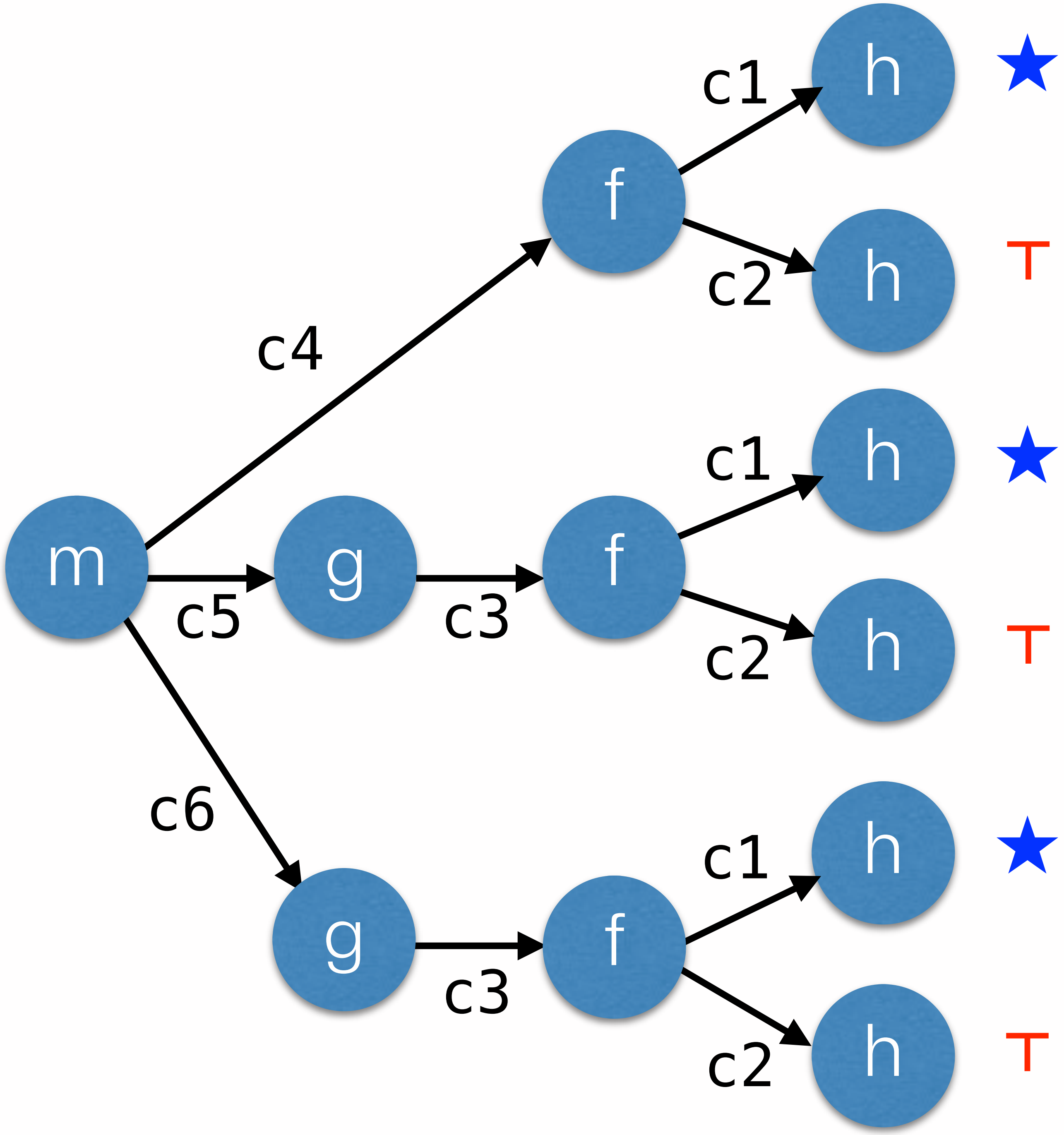
1. Collect queries whose expressions are assigned with ★

```
    int h(n) {ret n;}

    void f(a) {
c1: ★ x = h(a);
      assert(x > 1);  // Q1
c2: ⊤ y = h(input());
      ~~assert(y > 1)~~;  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```



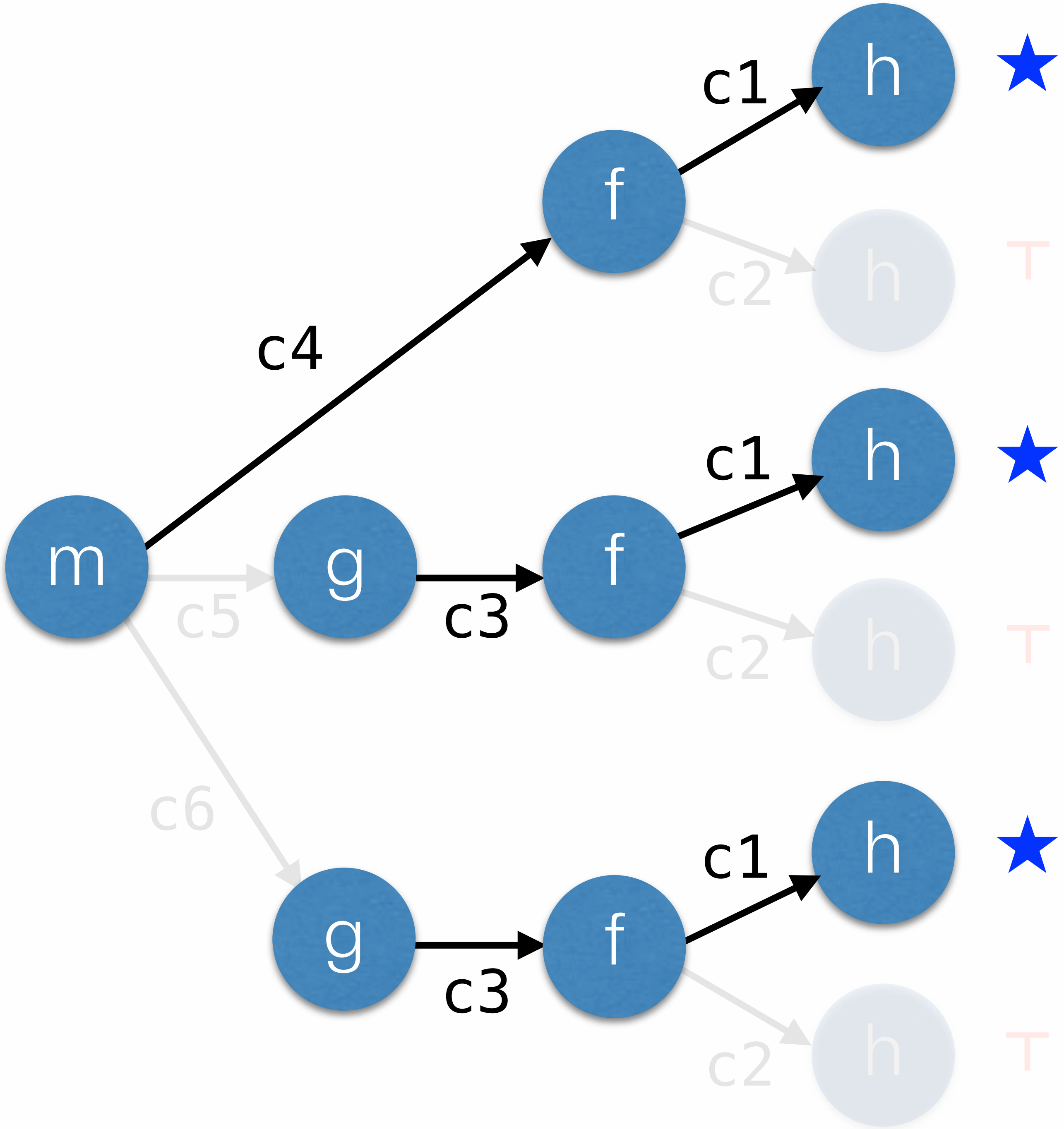
2. Find the program slice that contributes to the selected query

```
    int h(n) {ret n;}

    void f(a) {
c1:   x = h(a);
      assert(x > 1);  // Q1
c2:   y = h(input());
      ~~assert(y > 1)~~;  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```



3. Collect contexts in the slice

```
    int h(n) {ret n;}

    void f(a) {
c1:   x = h(a);
      assert(x > 1);  // Q1
c2:   y = h(input());
      ~~assert(y > 1)~~;  // Q2
    }

c3: void g() {f(8);}

    void m() {
c4:   f(4);
c5:   g();
c6:   g();
    }
```



=> Contexts for **h**: {c3 · c1, c4 · c1}

Selective Context-Sensitivity

		Context-Insensitve		Ours	
Pgm	**LOC**	**#alarms**	**time(s)**	**#alarms**	**time(s)**
spell	2K	58	1	30	1
bc	13K	606	14.0	483	16
tar	20K	940	42	799	47
less	23K	654	123.0	562	166
sed	27K	1,325	108	1,238	118
make	27K	1,500	88	1,028	106
grep	32K	735	12	653	16
wget	35K	1,307	69.0	942	82
a2ps	65K	3,682	118	2,121	178
bison	102K	1,894	136	1,742	173
TOTAL	**346K**	**12,701**	**707.1**	**9,598**	**903.6**

24.4%

Selective Context-Sensitivity

		Context-Insensitve		Ours	
Pgm	LOC	#alarms	time(s)	#alarms	time(s)
spell	2K	58	1	30	1
bc	13K	606	14.0	483	16
tar	20K	940	42	799	47
less	23K	654	123.0	562	166
sed	27K	1,325	108	1,238	118
make	27K	1,500	88	1,028	106
grep	32K	735	12	653	16
wget	35K	1,307	69.0	942	82
a2ps	65K	3,682	118	2,121	178
bison	102K	1,894	136	1,742	173
TOTAL	346K	12,701	707.1	9,598	903.6

27.8%

Selective Context-Sensitivity

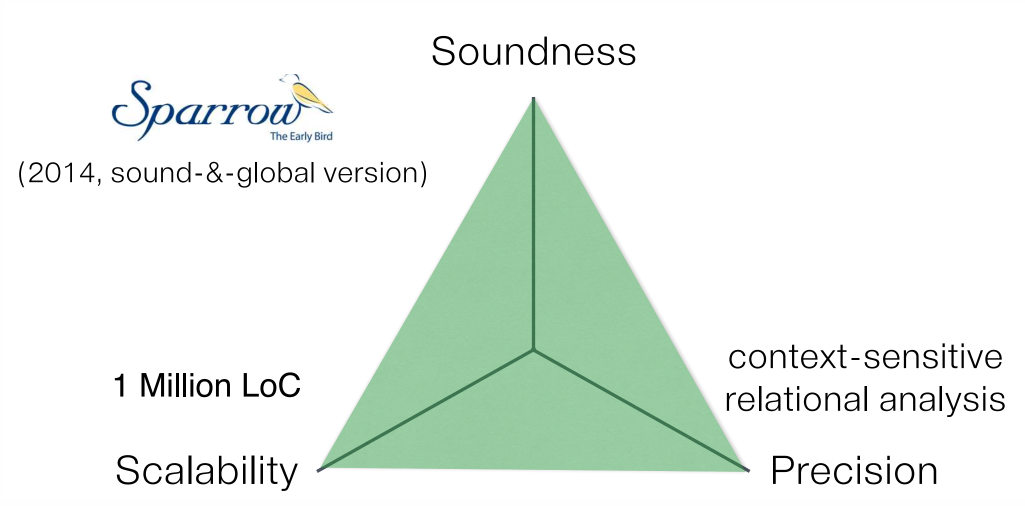
		Context-Insensitve		Ours	
Pgm	LOC	#alarms	time(s)	#alarms	time(s)
spell	2K	58	1	30	1
bc	13K	606	14.0	483	16
tar	20K	940	42	799	47
less	23K	654	123.0	562	166
sed	27K	1,325	108	1,238	118
make	27K	1,500	88	1,028	106
grep	32K	735	12	653	16
wget	35K	1,307	69.0	942	82
a2ps	65K	3,682	118	2,121	178
bison	102K	1,894	136	1,742	173
TOTAL	346K	12,701	707.1	9,598	903.6

pre-analysis : 14.7%
main analysis: 13.1%

27.8%

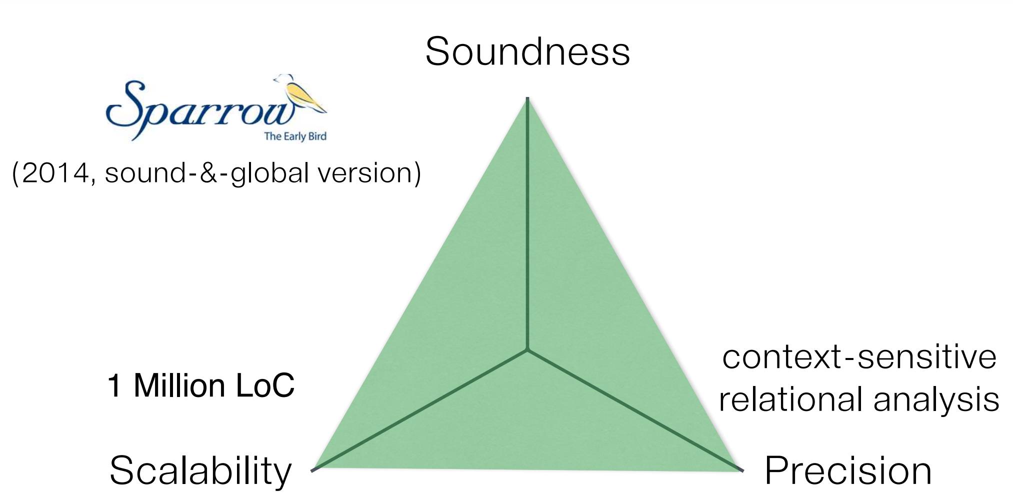
Summary

- Towards Sound, Precise, Scalable Analysis
  - Access Pre-analysis + Sparse Analysis
  - Impact Pre-analysis + Selective X-Sensitive Analysis
- Frameworks
  - Precision-preserving Sparse Analyses
  - Effective X-Sensitive Analyses



Summary

- Towards Sound, Precise, Scalable Analysis
  - Access Pre-analysis + Sparse Analysis
  - Impact Pre-analysis + Selective X-Sensitive Analysis
- Frameworks
  - Precision-preserving Sparse Analyses
  - Effective X-Sensitive Analyses


Soundness
Sparrow
The Early Bird
(2014, sound-&-global version)
1 Million LoC
Scalability
context-sensitive
relational analysis
Precision