Today

- How to solve undecidable problem?
  - ex) program analysis
- Classes of decidable problems
  - P / NP / NP-complete

The Fundamental SW Challenge

The Fundamental SW Challenge

The Fundamental SW Challenge

The Fundamental SW Challenge

'/' 응용 프로그램에 서버 오류가 있습니다.

인덱스가 배열 범위를 벗어났습니다.

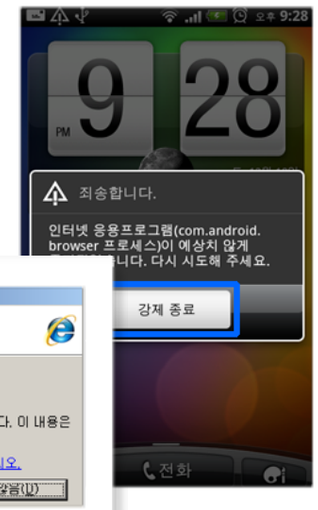
설명: 현재 웹 요청을 실행하는 동안 처리되지 않은 예외가 발생했습니다. 스택 추적을 검
생한 위치에 대한 자세한 정보를 확인하십시오.

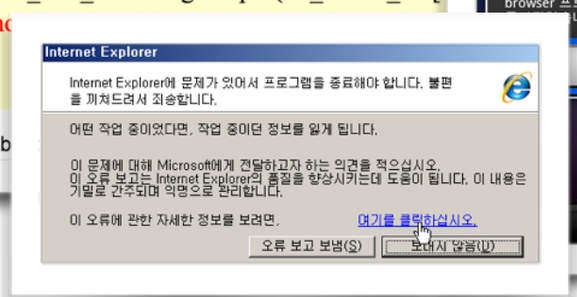
예외 정보: System.IndexOutOfRangeException: 인덱스가 배열 범위를 벗어났습니

소스 오류:

```
줄 192:                {
줄 193:                    new_link_aid = Regex.Split(rel_article_list[i
줄 194:                    mc
줄 195:                }
줄 196:            }
```

소스 파일: d:\WEB\mnews.jtb





사회 "외주업체 지하철 신호관리 프로그램 자주 오류 발생"

과학 과학일반 '나로호 불발' 압력측정 소프트웨어 오류가 원인

오픈SSL 보안 취약점 하트블리드의 해결은 수 개월 이상 소요될 것

최근 밝혀진 OpenSSL 관련 보안 취약점, 일명 하트블리드(Heartbleed)의 해결에는 수 개월이 걸릴 것이라는 전망이 나왔다.

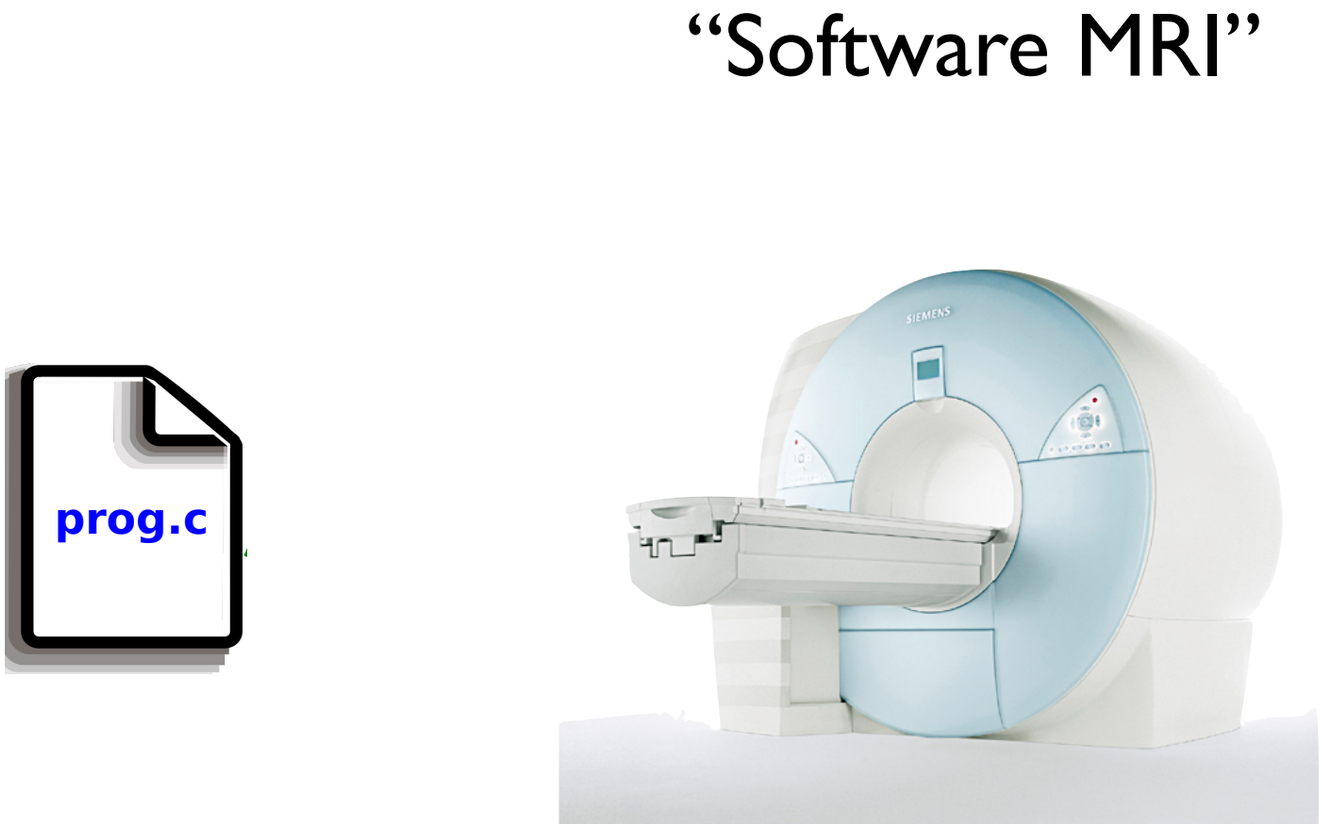
Technology for Safe Softwares

Manual, ad-hoc, postmortem:

code review, testing, simulation, debugging, etc

Static Program Analysis

- Predict sw behavior statically and automatically



Static Program Analysis

- Predict sw behavior statically and automatically

"Software MRI"

Static Program Analysis

- Predict sw behavior statically and automatically

"Software MRI"

Example: Sparrow

- Detects memory errors in C programs
  - e.g., buffer-overrun, memory leak, null-dereference, etc
- Features (vs. testing)
  - Full **automation**
  - Find bugs **early**
  - **All bugs** found (ensured by theory)

```
16      static char *curfinal = "HDACB  FE";
17
18      keysym = read_from_input ();
19
20      if ((((KeySym)(keysym) >= 0xFF91) && ((KeySym)(keysym) <= 0xFF94)))
21      {
22         unparseputc((char)(keysym-0xFF91 +'P'), pty);
23         key = 1;
24      }
25      else if (keysym >= 0)
26      {
27          if (keysym < 16)
28          {
29              if (read_from_input())
30              {
31                 if (keysym >= 10) return;
32                 curfinal[keysym] = 1;
33              }
34              else
35              {
36                  curfinal[keysym] = 2;
37              }
38          }
39          if (keysym < 10)
40          {
41              unparseputc(curfinal[keysym], pty);
42          }
43      }
```

```
16      static char *curfinal = "HDACB  FE";
17
18      keysym = read_from_input ();
19
20      if ((((KeySym)(keysym) >= 0xFF91) && ((KeySym)(keysym) <= 0xFF94)))
21      {
22         unparseputc((char)(keysym-0xFF91 +'P'), pty);
23         key = 1;
24      }
25      else if (keysym >= 0)
26      {
27          if (keysym < 16)
28          {
29              if (read_from_input())
30              {
31                  if (keysym >= 10) return;
32          safe    curfinal[keysym] = 1;
33              }
34              else
35              {
36  buffer-overrun  curfinal[keysym] = 2;
37              }
38          }
39          if (keysym < 10)
40          {
41              unparseputc(curfinal[keysym], pty);
42          }
43      }
                    safe
```

```
16      static char *curfinal = "HDACB  FE";
17
18      keysym = read_from_input ();
19
20      if ((((KeySym)(keysym) >= 0xFF91) && ((KeySym)(keysym) <= 0xFF94)))
21      {
22         unparseputc((char)(keysym-0xFF91 +'P'), pty);
23         key = 1;
24      }
25      else if (keysym >= 0)
26      {
27          if (keysym < 16)
28          {
29              if (read_from_input())
30              {
31                  if (keysym >= 10) return;
32                  curfinal[keysym] = 1;
33              }
34              else
35              {
36                  curfinal[keysym] = 2;
37              }
38          }
39          if (keysym < 10)
40          {
41              unparseputc(curfinal[keysym], pty);
42          }
43      }
```

curfinal: buffer of size 10

safe

buffer-overrun

safe

```
16      static char *curfinal = "HDACB   FE";
17
18      keysym = read_from_input ();
19
20      if ((((KeySym)(keysym) >= 0xFF9                ym) <= 0xFF94)))
21      {
22          unparseputc((char)(keysym-0xFF91 +'P'), pty);
23          key = 1;
24      }
25      else if (keysym >= 0)
26      {
27          if (keysym < 16)
28          {
29              if (read_from_input())
30              {
31                  if (keysym >= 10) return;
32                  curfinal[keysym] = 1;
33              }
34              else
35              {
36                  curfinal[keysym] = 2;
37              }
38          }
39          if (keysym < 10)
40          {
41              unparseputc(curfinal[keysym], pty);
42          }
43      }
```

curfinal: buffer of size 10

keysym: any integer

safe

buffer-overrun

safe

```
16      static char *curfinal = "HDACB   FE";
17
18      keysym = read_from_input ();
19
20      if ((((KeySym)(keysym) >= 0xFF9                ym) <= 0xFF94)))
21      {
22          unparseputc((char)(keysym-0xFF91 +'P'), pty);
23          key = 1;
24      }
25      else if (keysym >= 0)
26      {
27          if (keysym < 16)
28          {
29              if (read_from_input())
30              {
31                  if (keysym >= 10) return;
32                  curfinal[keysym] = 1;
33              }
34              else
35              {
36                  curfinal[keysym] = 2;
37              }
38          }
39          if (keysym < 10)
40          {
41              unparseputc(curfinal[keysym], pty);
42          }
43      }
```

curfinal: buffer of size 10

keysym: any integer

keysym: [0,15]

safe

buffer-overrun

safe

```
16      static char *curfinal = "HDACB   FE";
17
18      keysym = read_from_input ();
19
20      if ((((KeySym)(keysym) >= 0xFF9                ym) <= 0xFF94)))
21      {
22          unparseputc((char)(keysym-0xFF91 +'P'), pty);
23          key = 1;
24      }
25      else if (keysym >= 0)
26      {
27          if (keysym < 16)
28          {
29              if (read_from_input())
30              {
31                  if (keysym >= 10) return;
32                  curfinal[keysym] = 1;
33              }
34              else
35              {
36                  curfinal[keysym] = 2;
37              }
38          }
39          if (keysym < 10)
40          {
41              unparseputc(curfinal[keysym], pty);
42          }
43      }
```

curfinal: buffer of size 10

keysym: any integer

keysym: [0,15]

safe

buffer-overrun

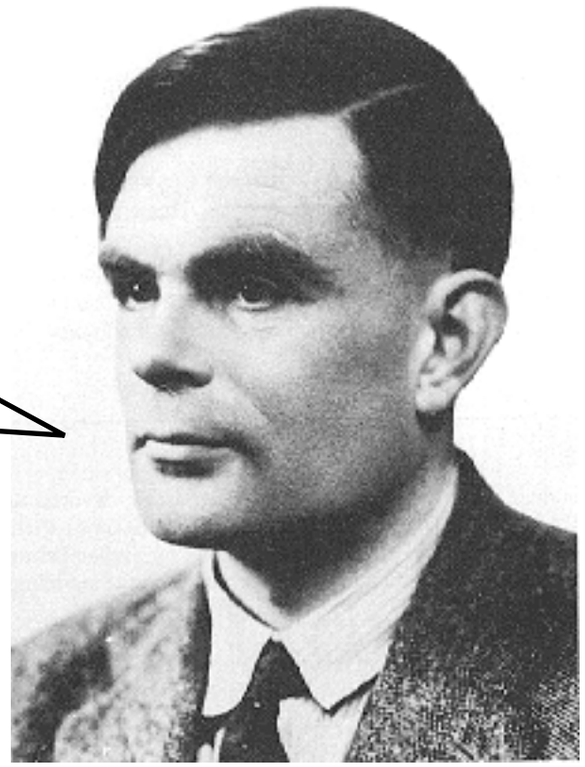
curfinal:[10,10]
keysym:[10,15]

safe

Is it possible?

Is it possible?

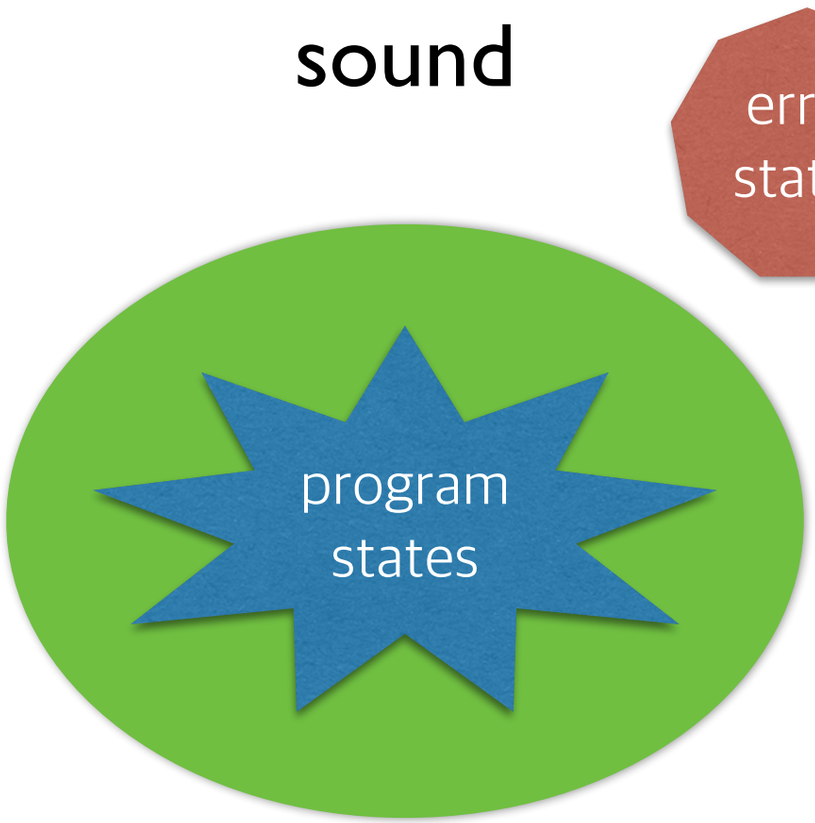
Impossible!



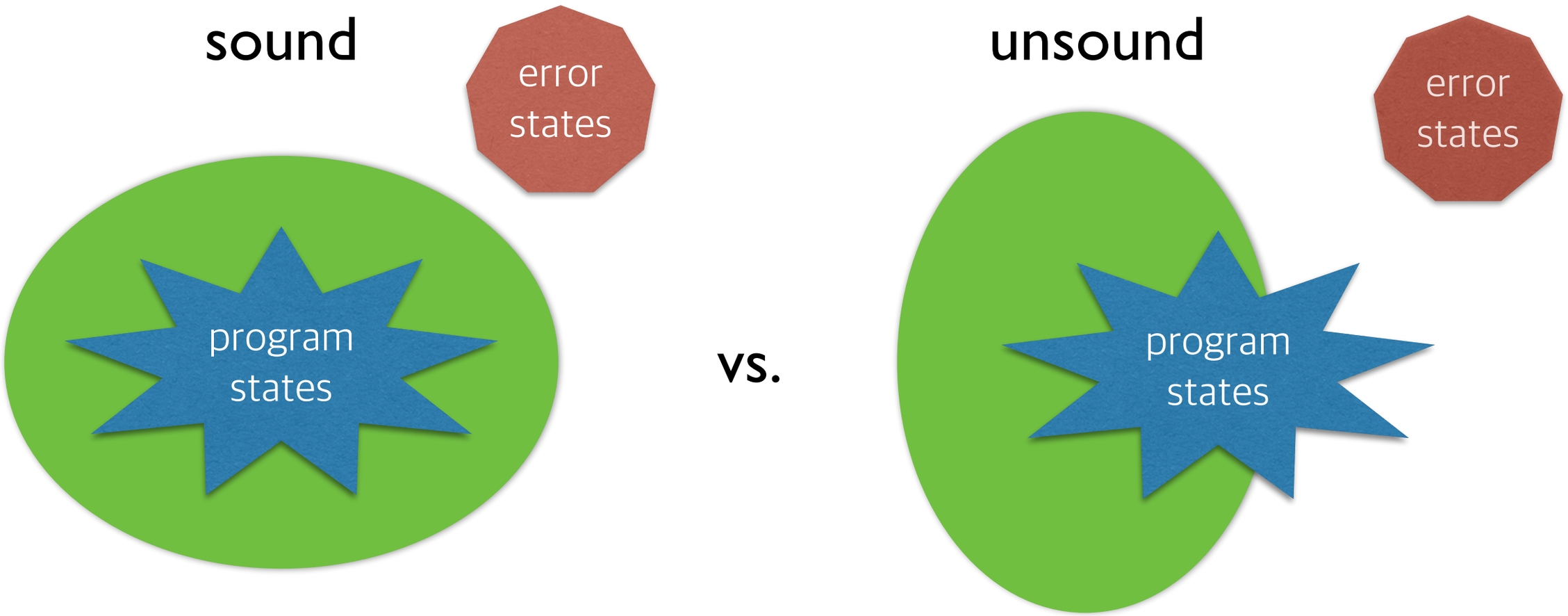
Alan Turing

Key Idea

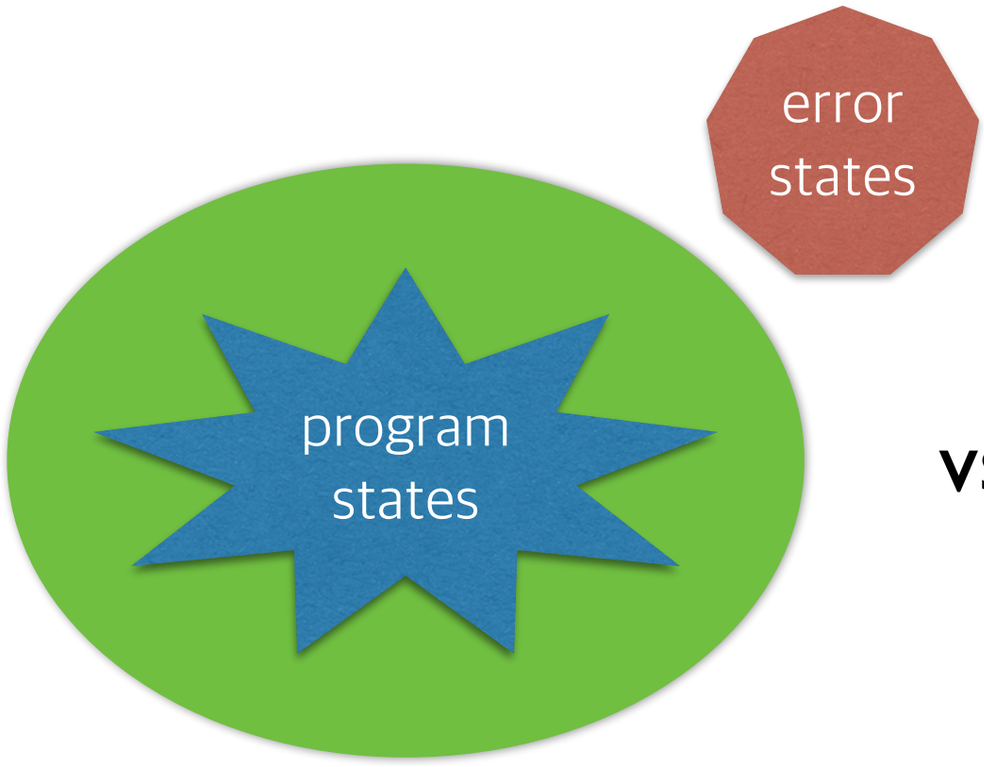
Key Idea



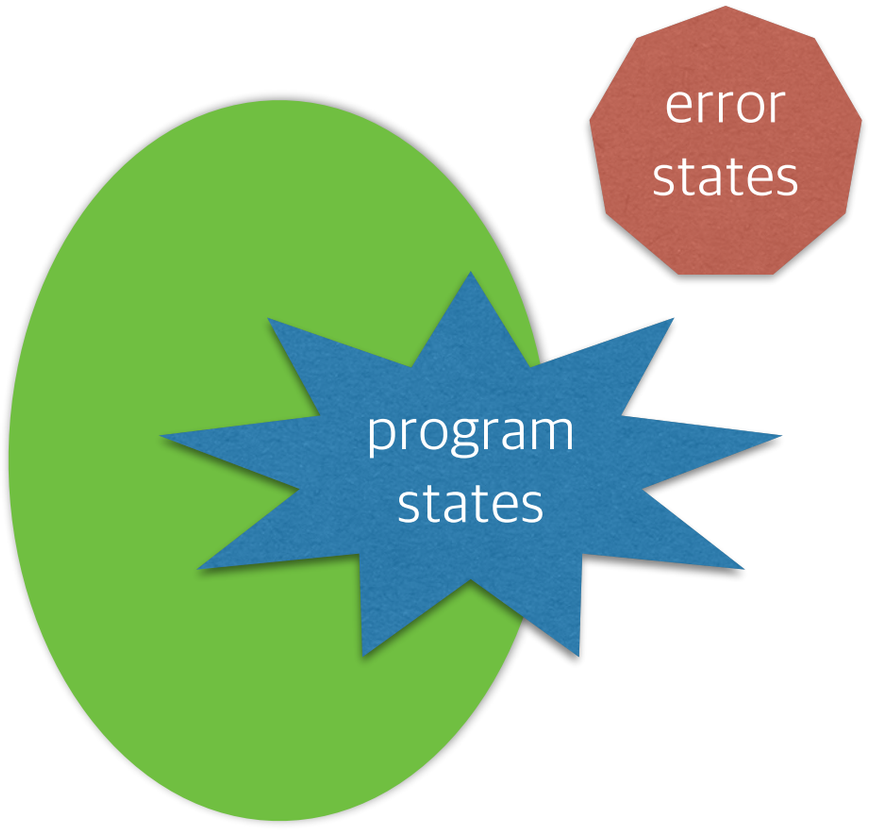
Key Idea


sound
error states
program states
vs.
unsound
error states
program states

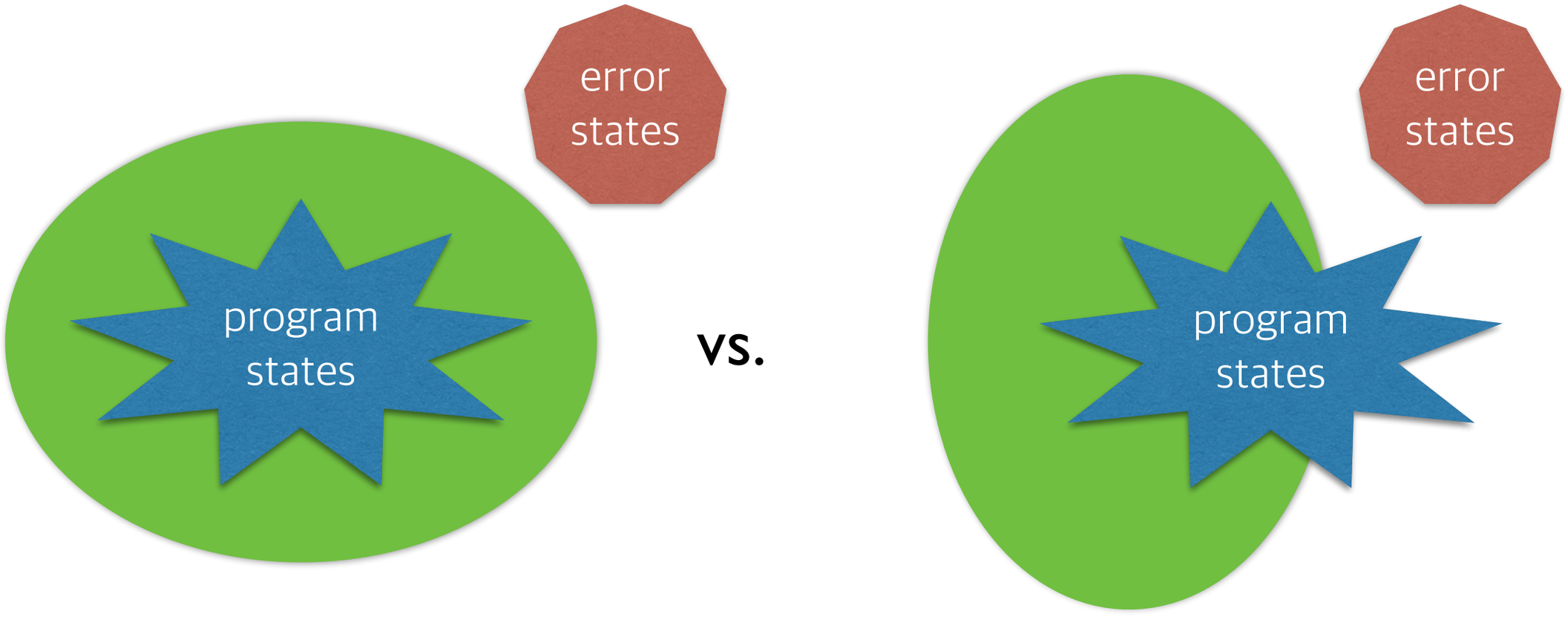
Challenge 1: Soundness Guarantee



vs.

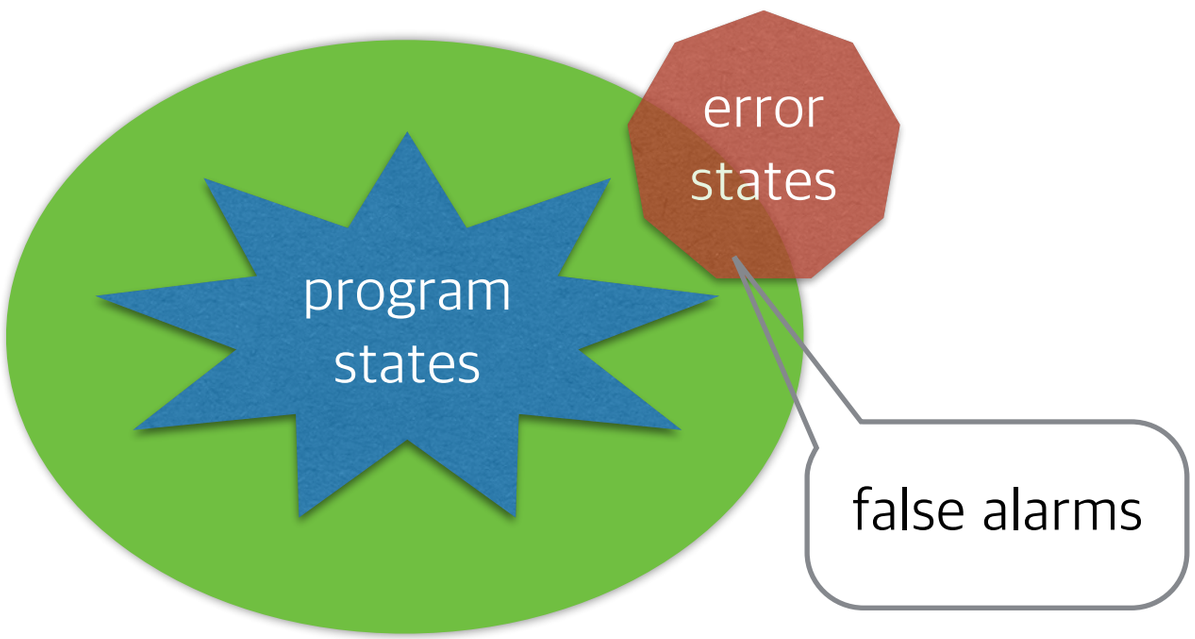


Challenge 1: Soundness Guarantee

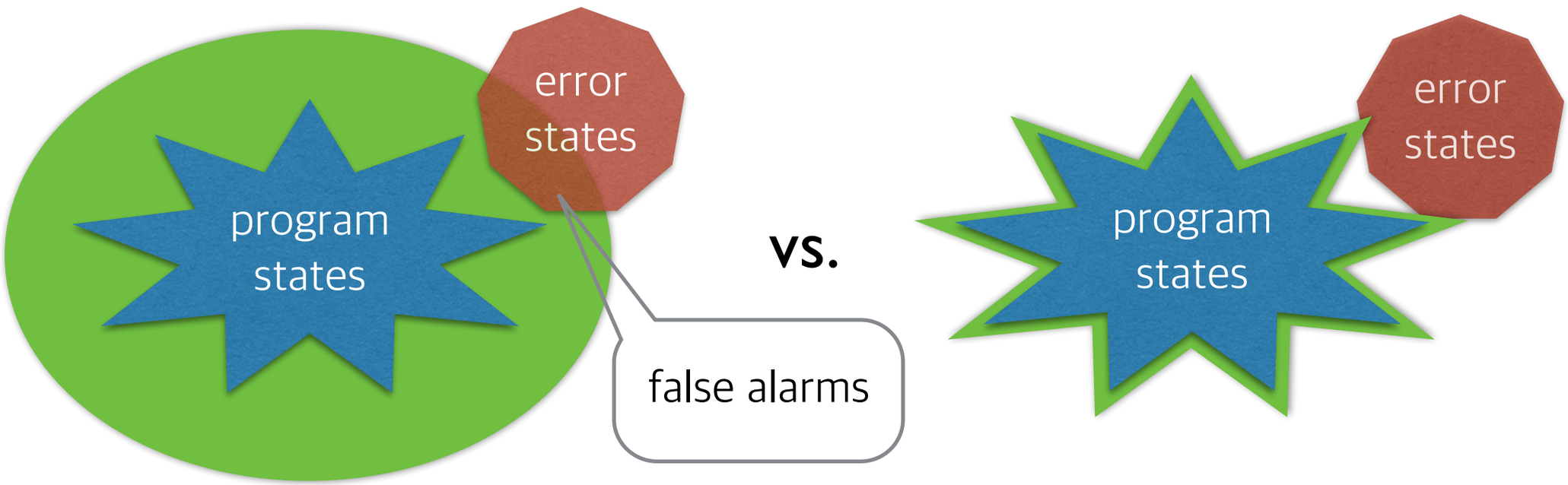


Abstract Interpretation Theory (Cousot&Cousot, 1977)

Challenge 2: False Alarms

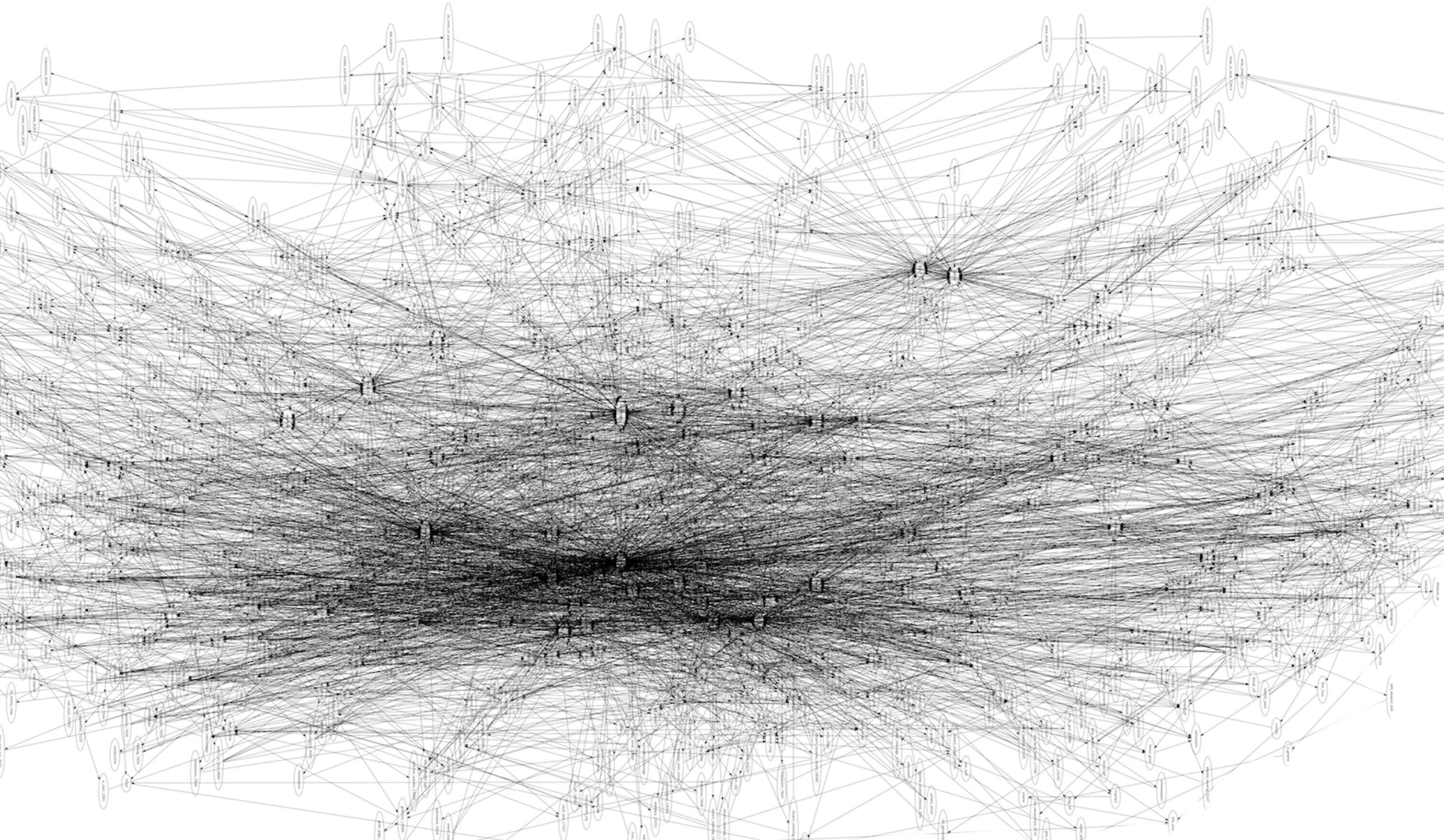


Challenge 2: False Alarms

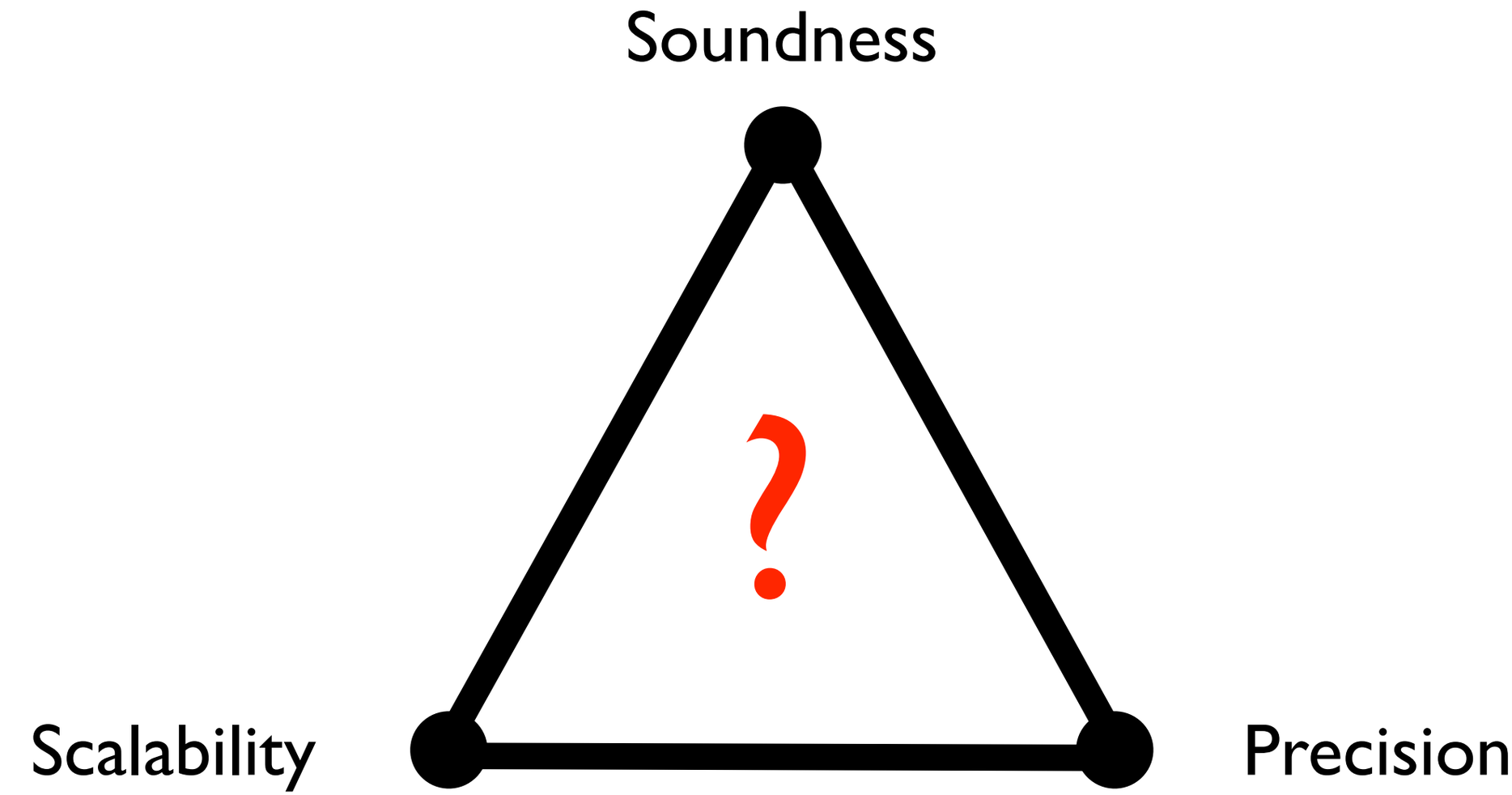


Challenge 3: Scalability

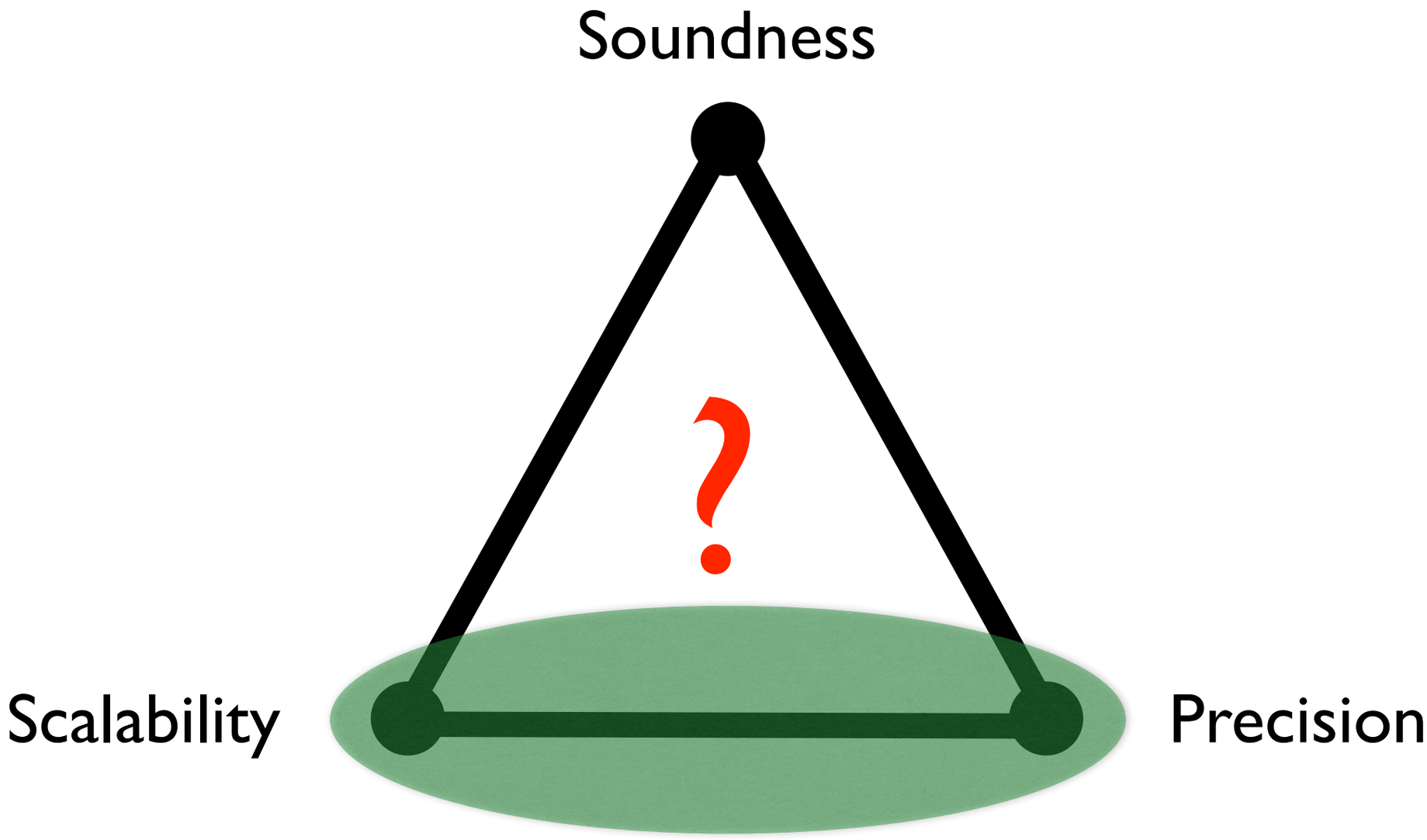
nethack-3.3.0 (211KLoC)



Common Sense: Infeasible

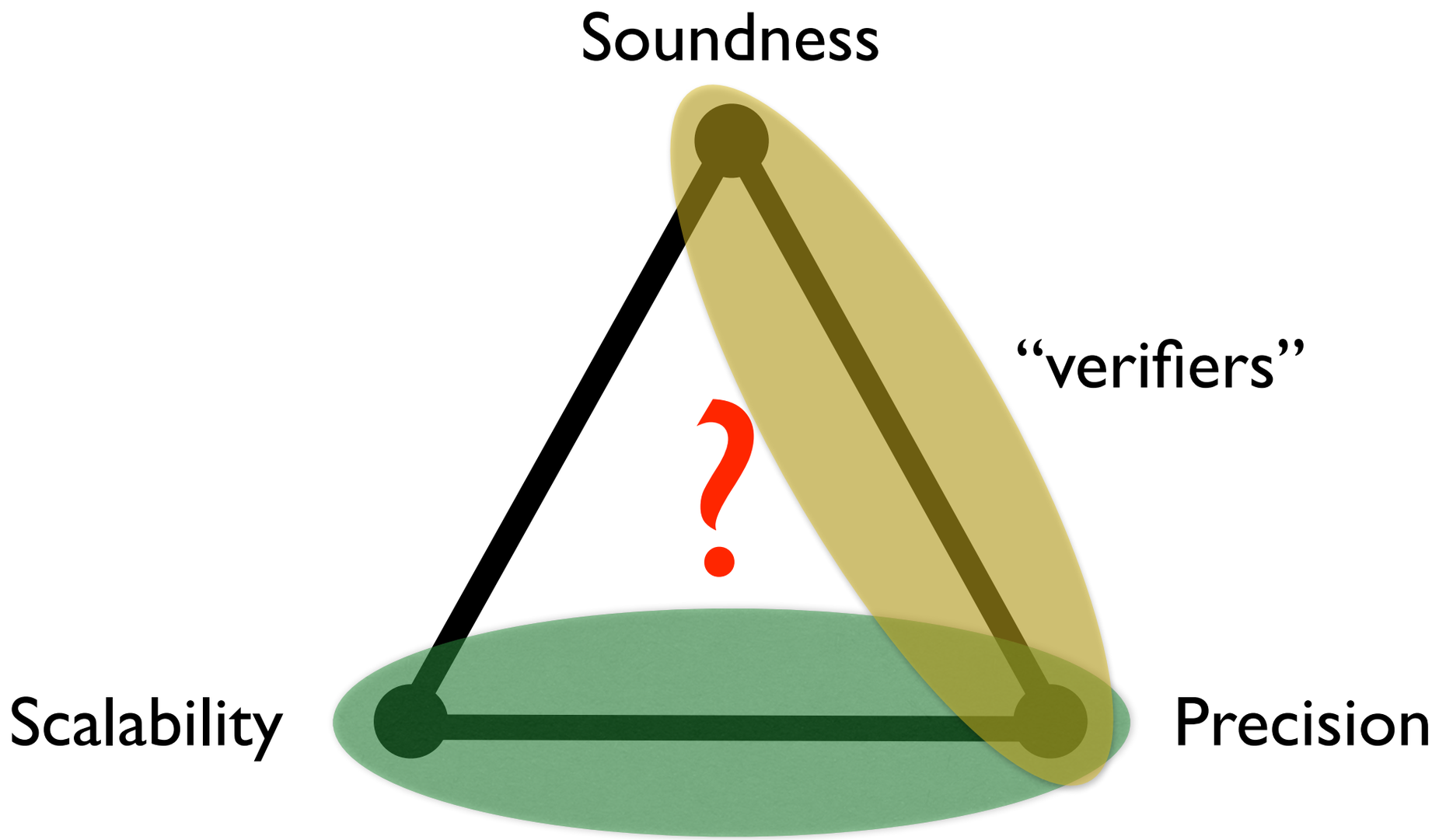


Common Sense: Infeasible

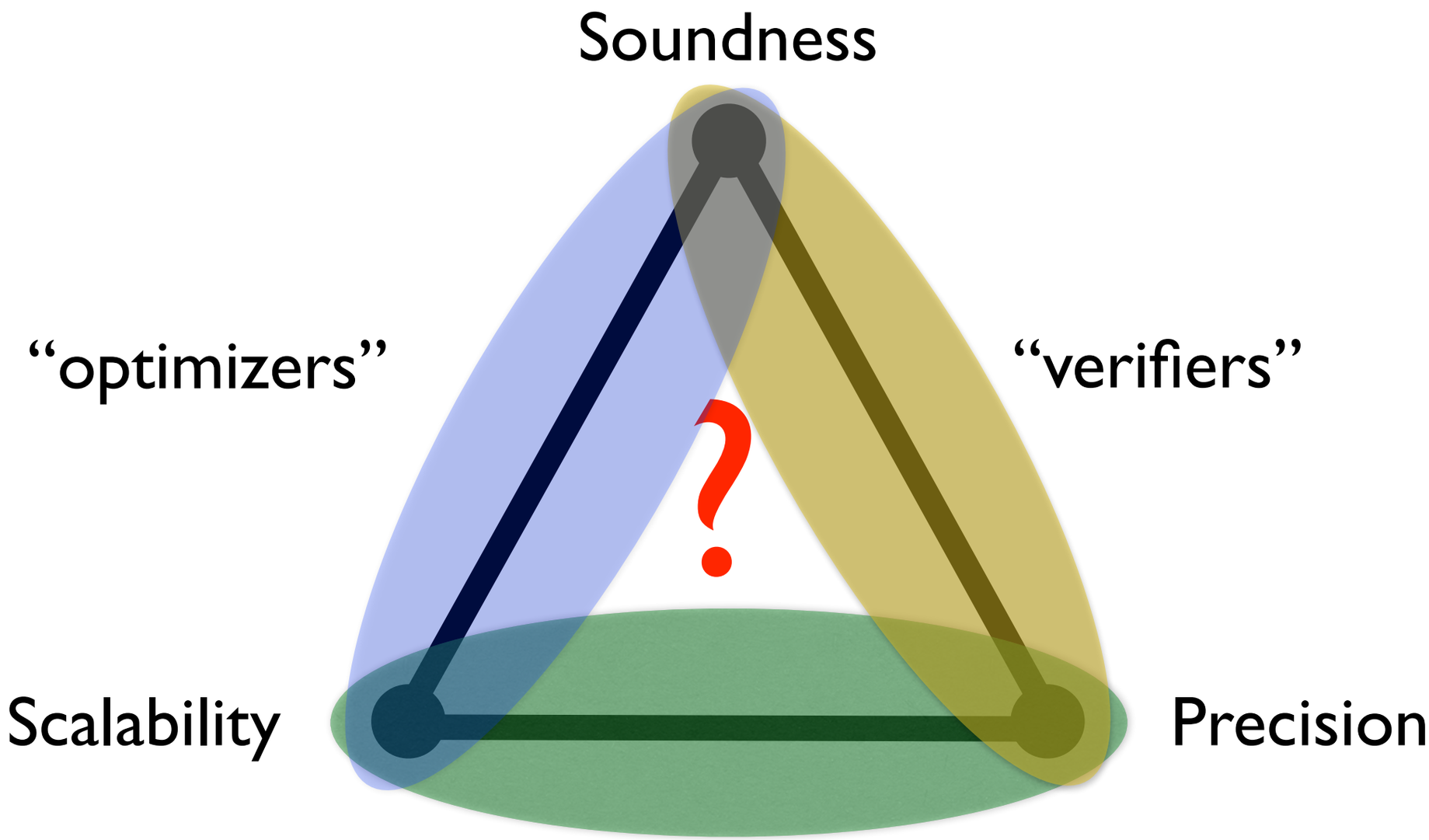


"bug-finders"

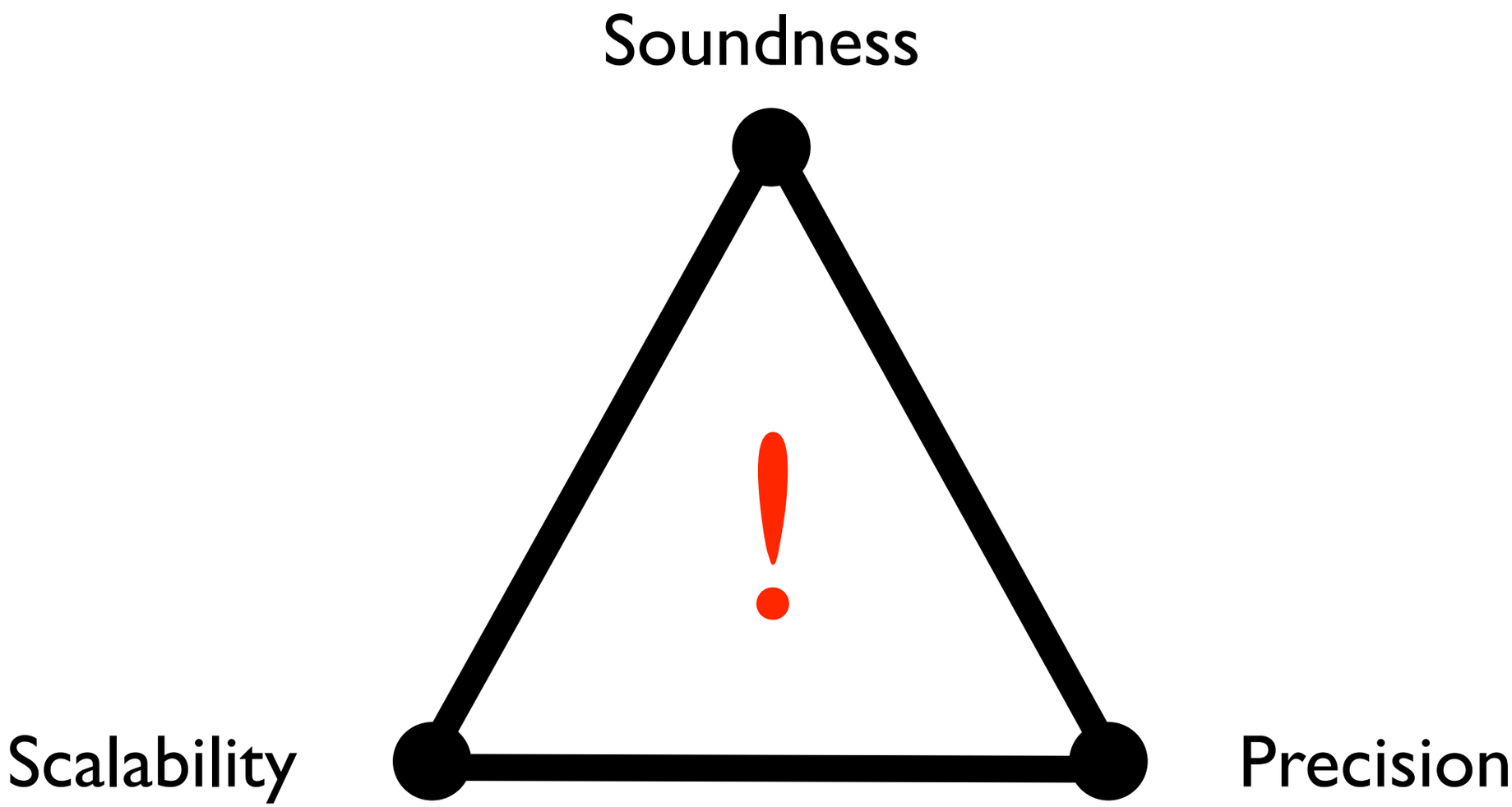
Common Sense: Infeasible



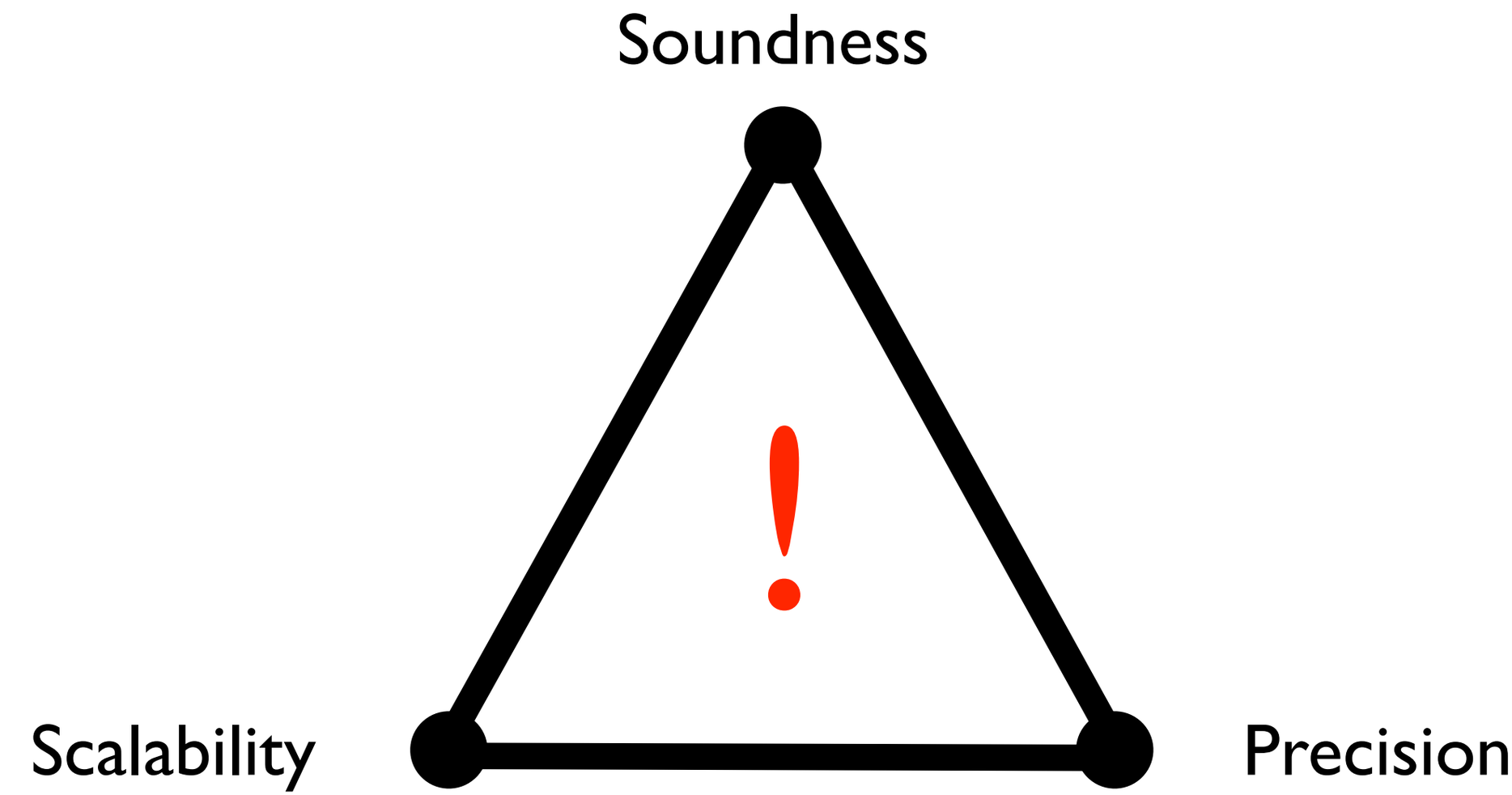
Common Sense: Infeasible



Goal: Not Any More

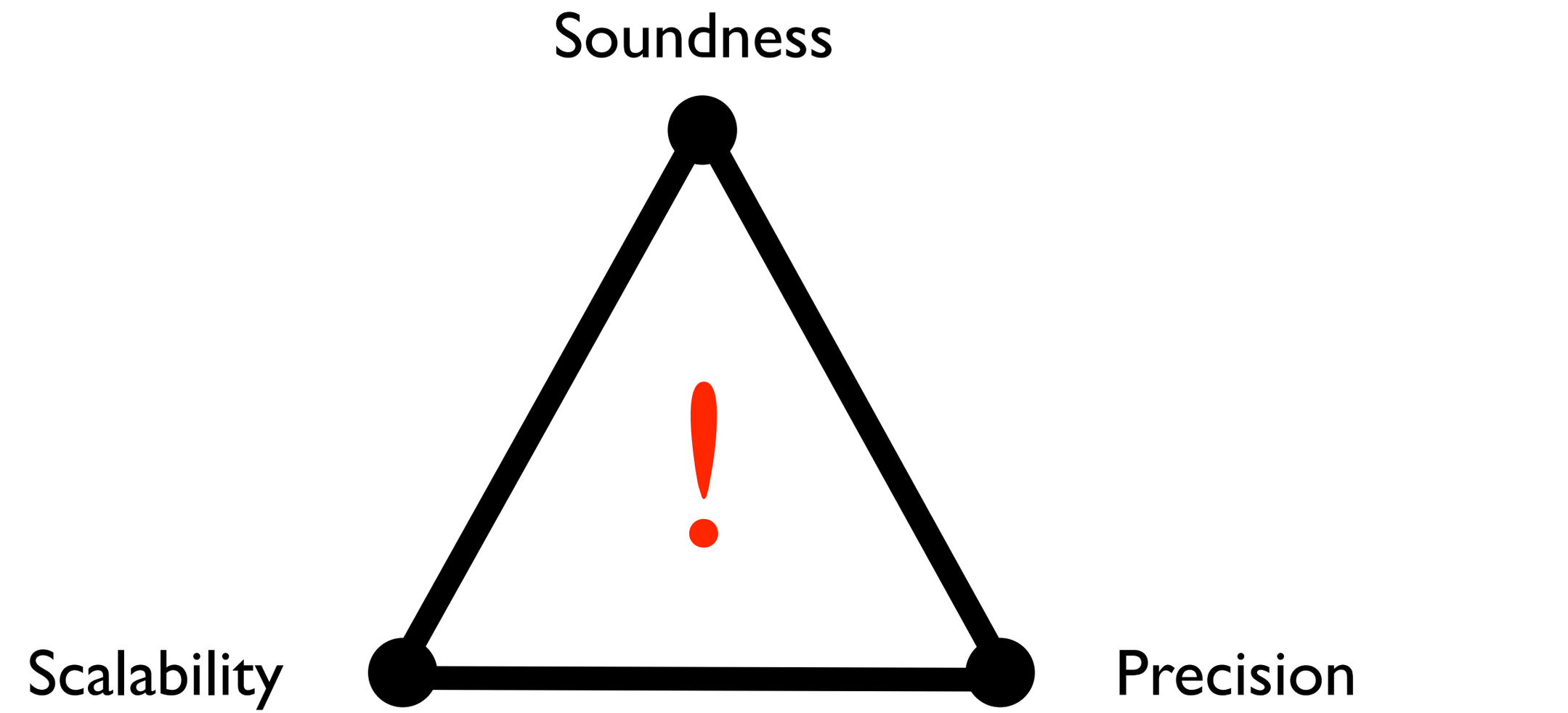


Goal: Not Any More



General Sparse Analysis Framework [PLDI'12]

Goal: Not Any More



General Sparse Analysis Framework [PLDI'12]

Selective X-Sensitivity Framework [PLDI'14]

Significance

- Crack down the common sense
- Publication:
  - General Sparse Analysis Framework
    - **ACM PLDI 2012** (top conference in programming languages)
    - **ACM TOPLAS 2014** (top journal in programming languages)
  - Selective X-Sensitivity Framework
    - **ACM PLDI 2014** (top conference in programming languages)

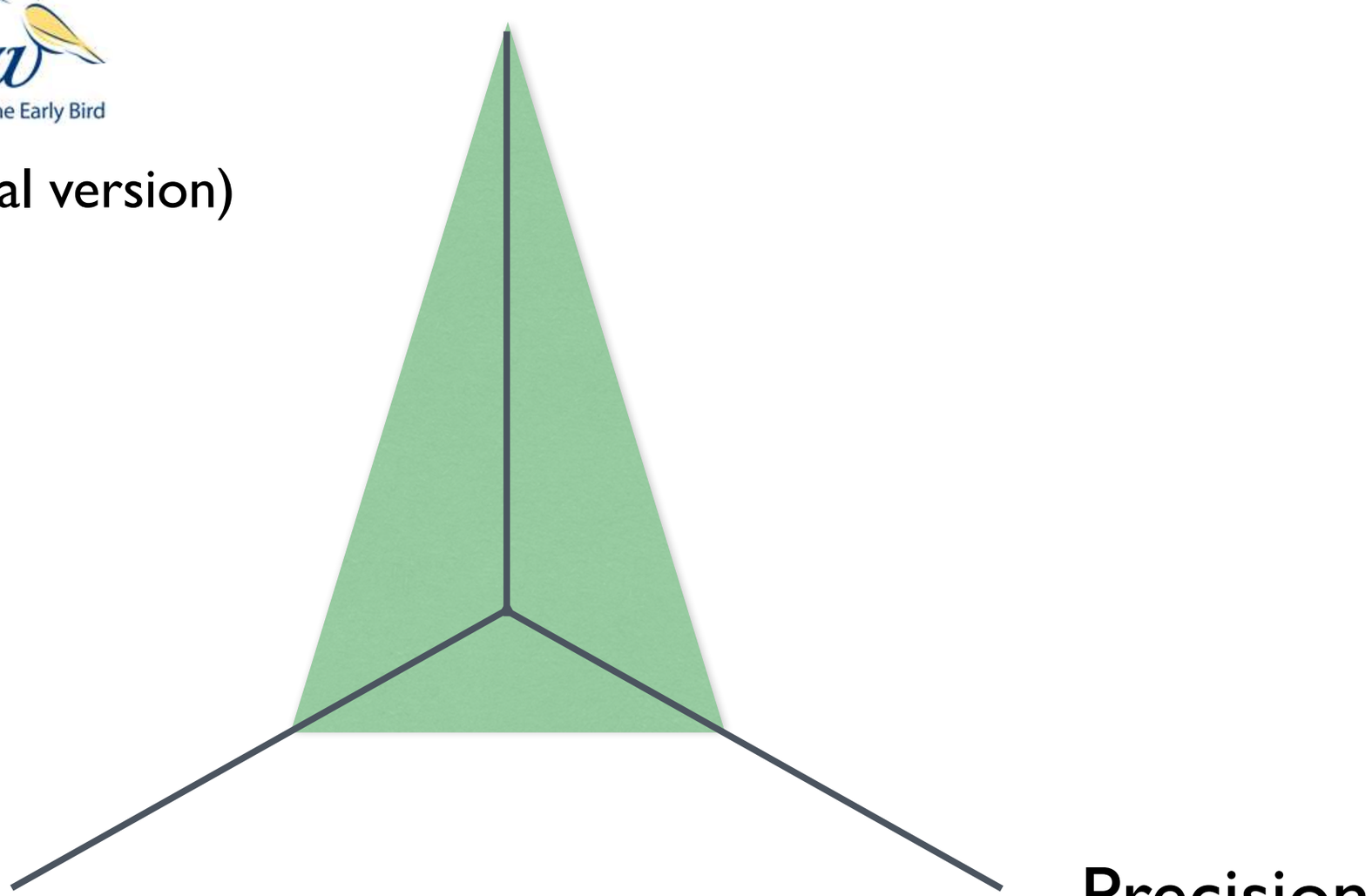
Motivation

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

The Challenge in Reality

Sparrow
The Early Bird

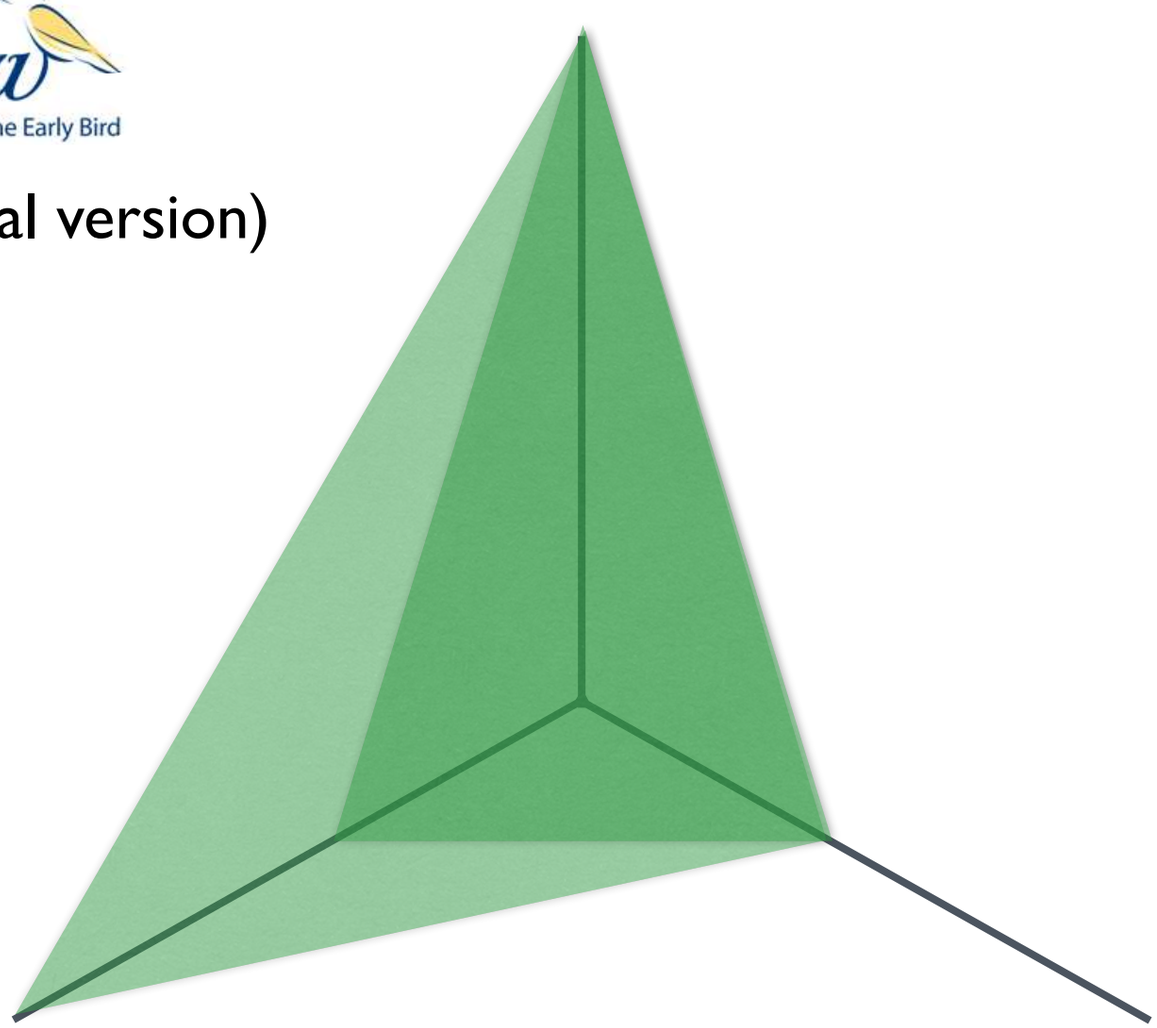
(2007, sound-&-global version)



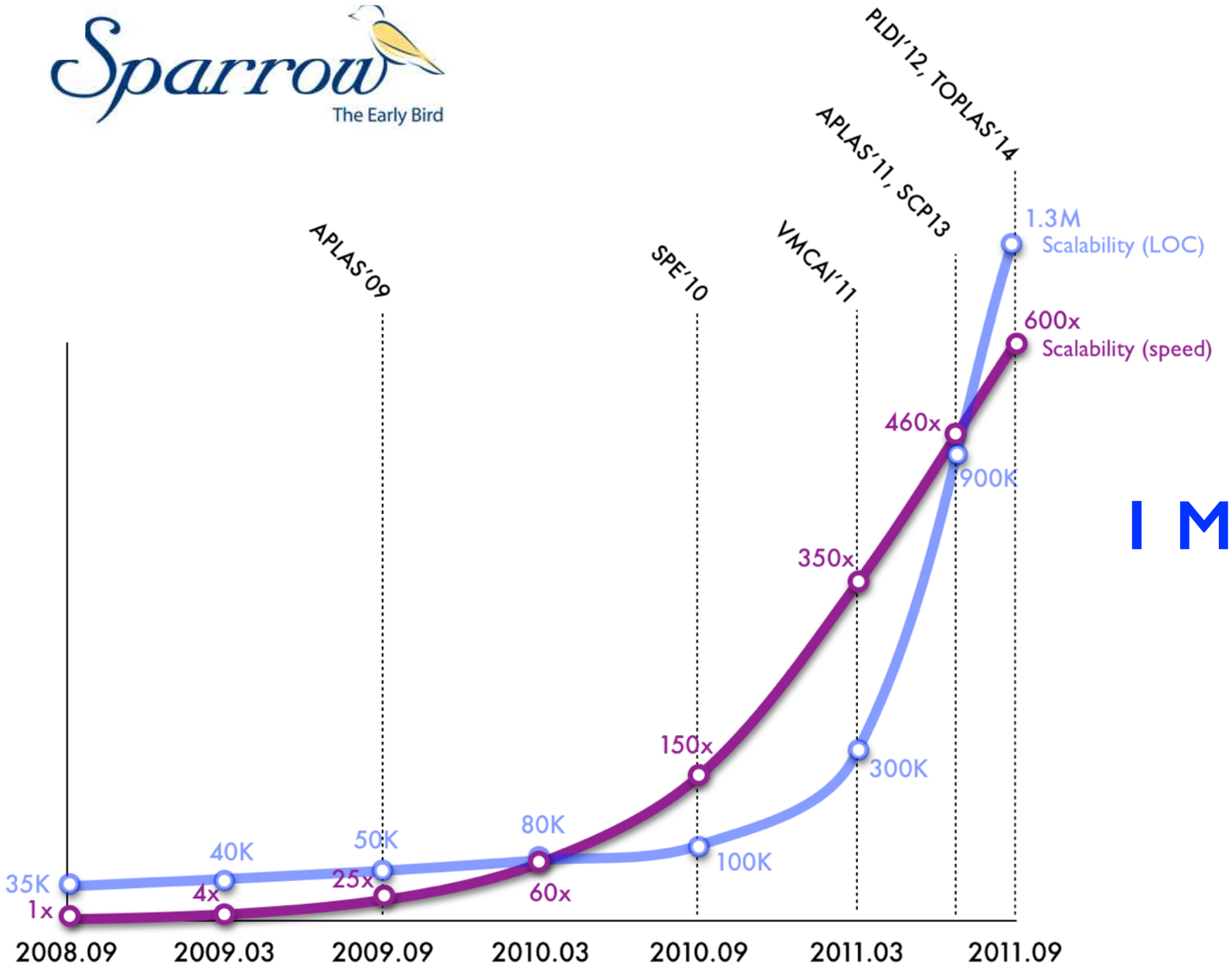
The First Goal: Scalability

Sparrow
The Early Bird

(2012, sound-&-global version)



Scalability Improvement

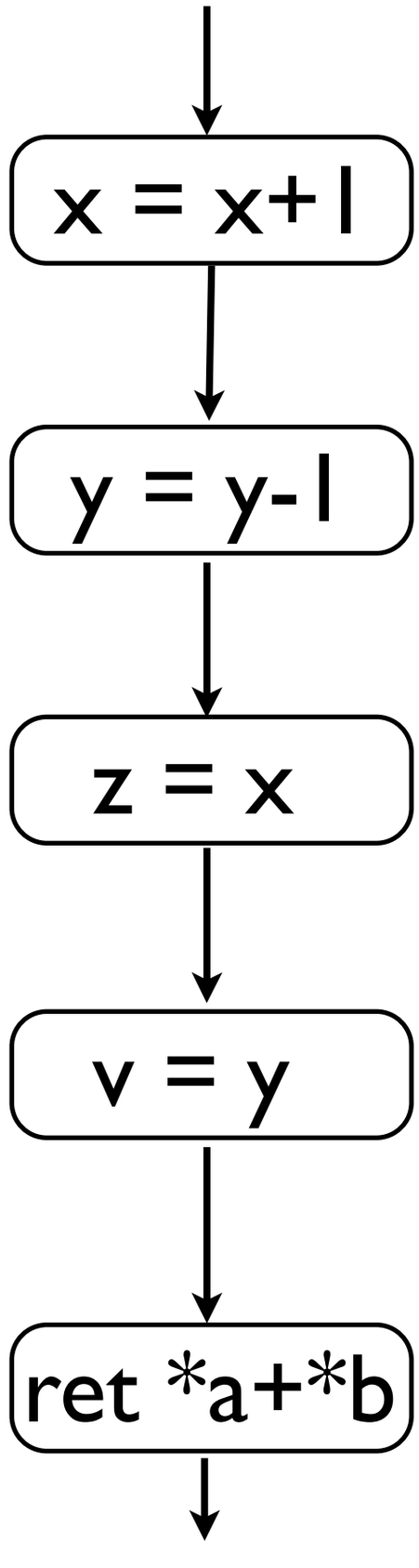


1 Million LoC in 10 hrs

PLDI'12

Key: General Sparse Analysis

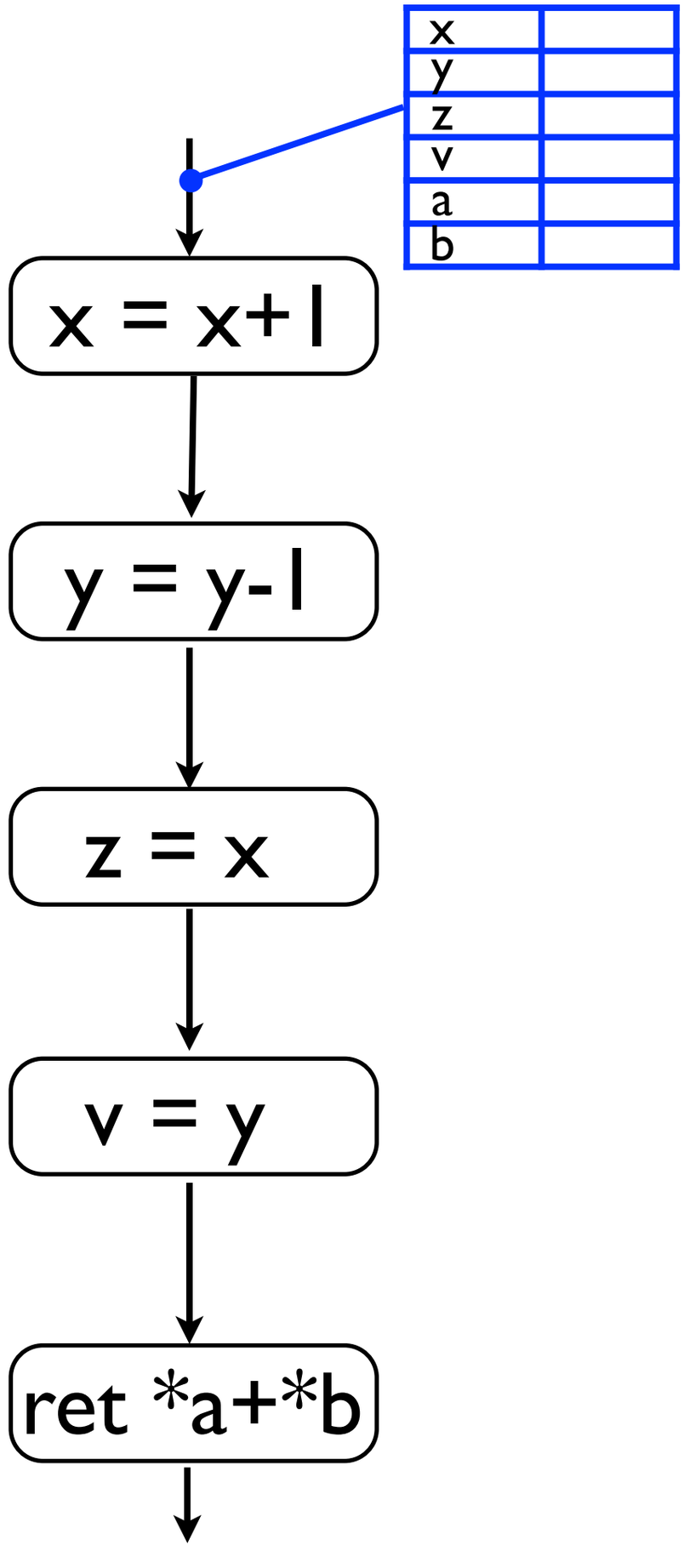
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

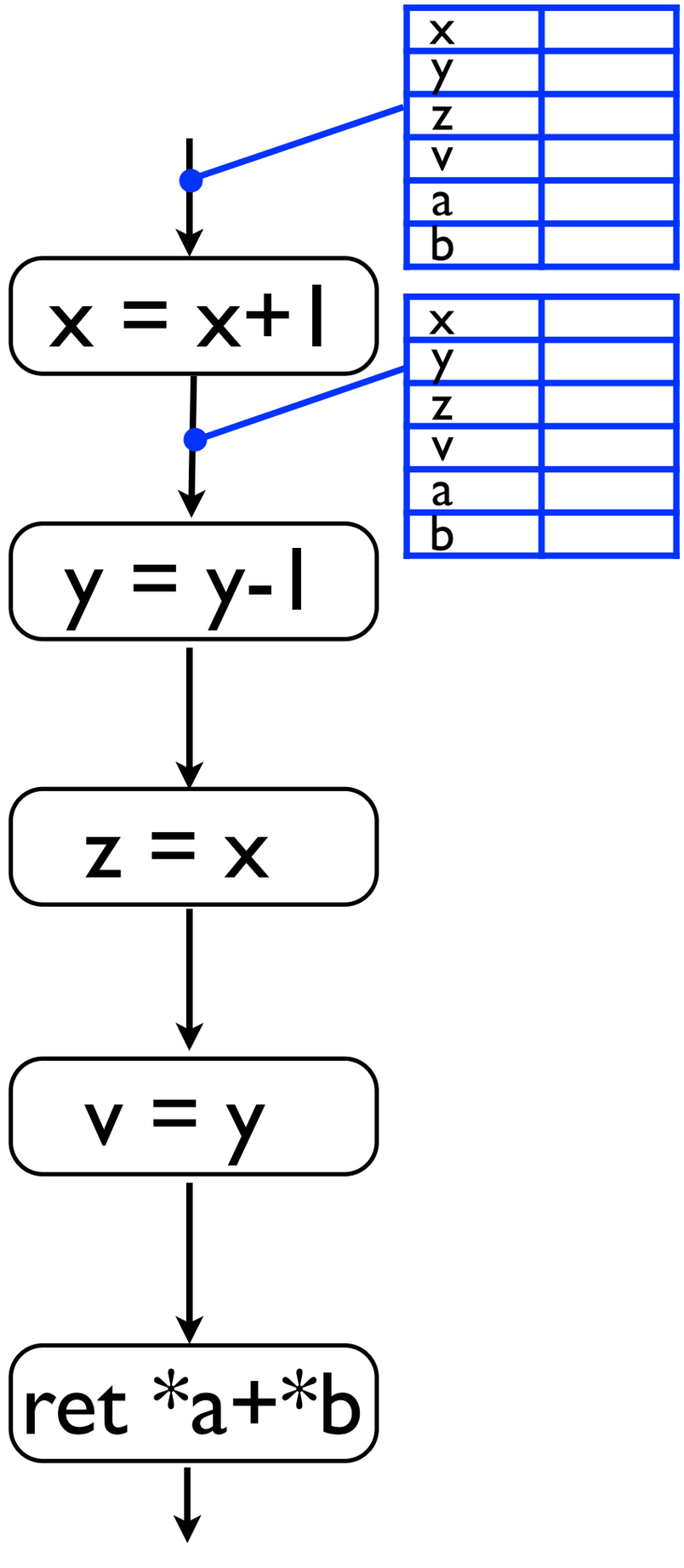
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

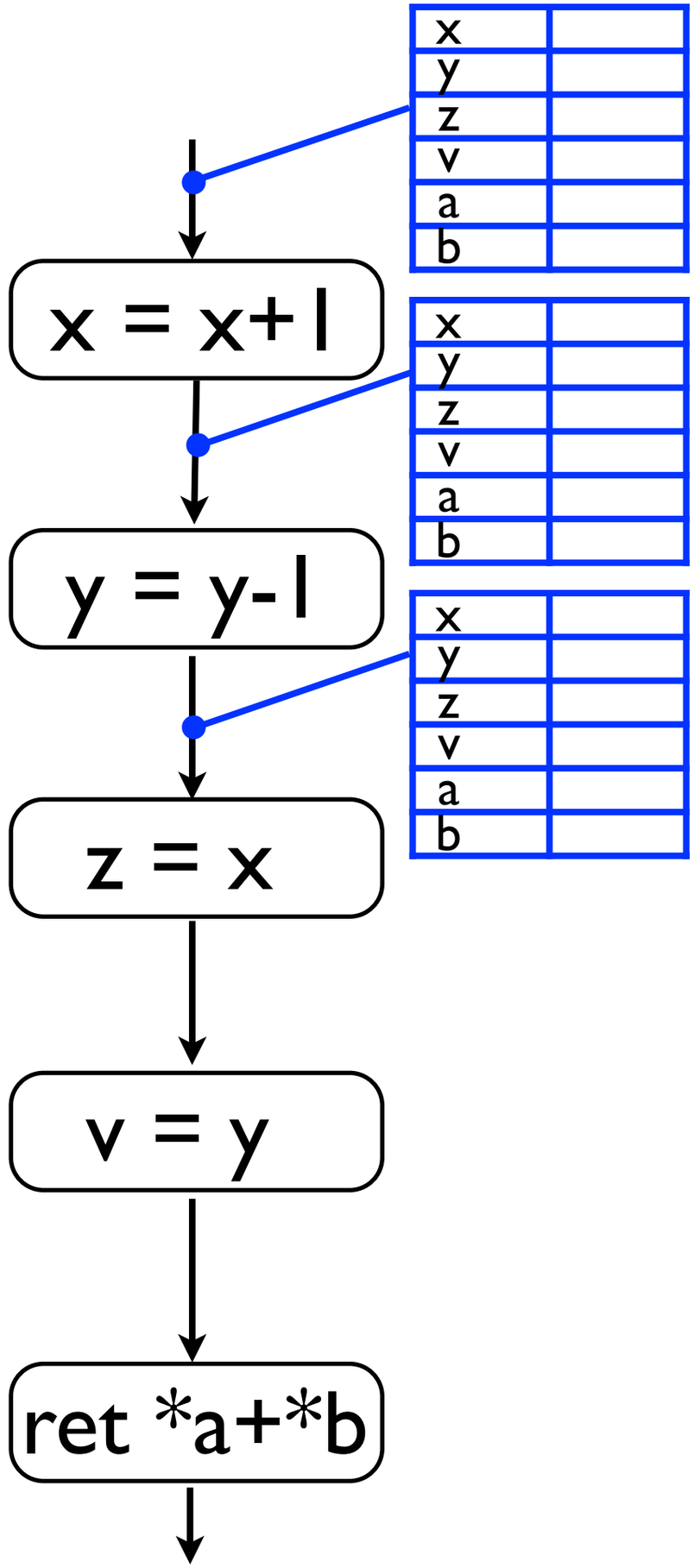
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

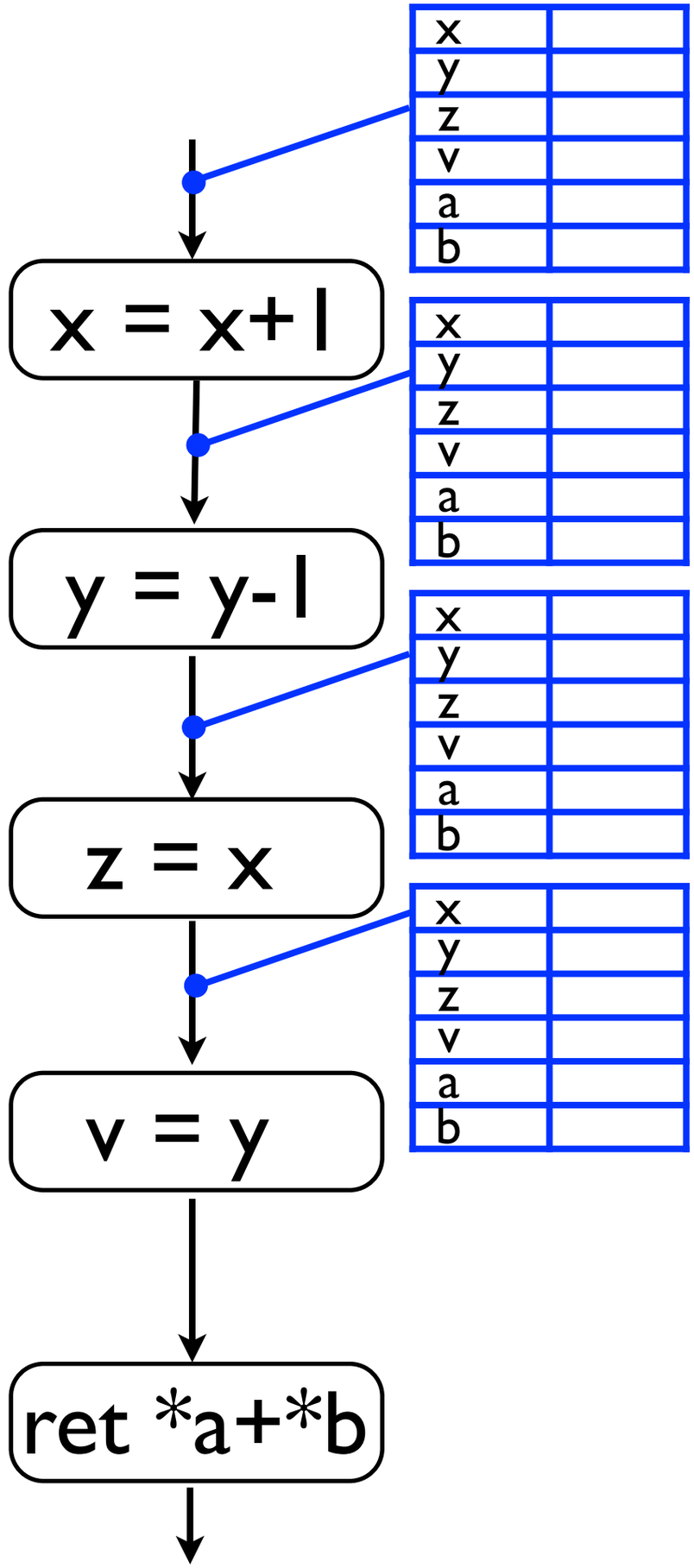
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

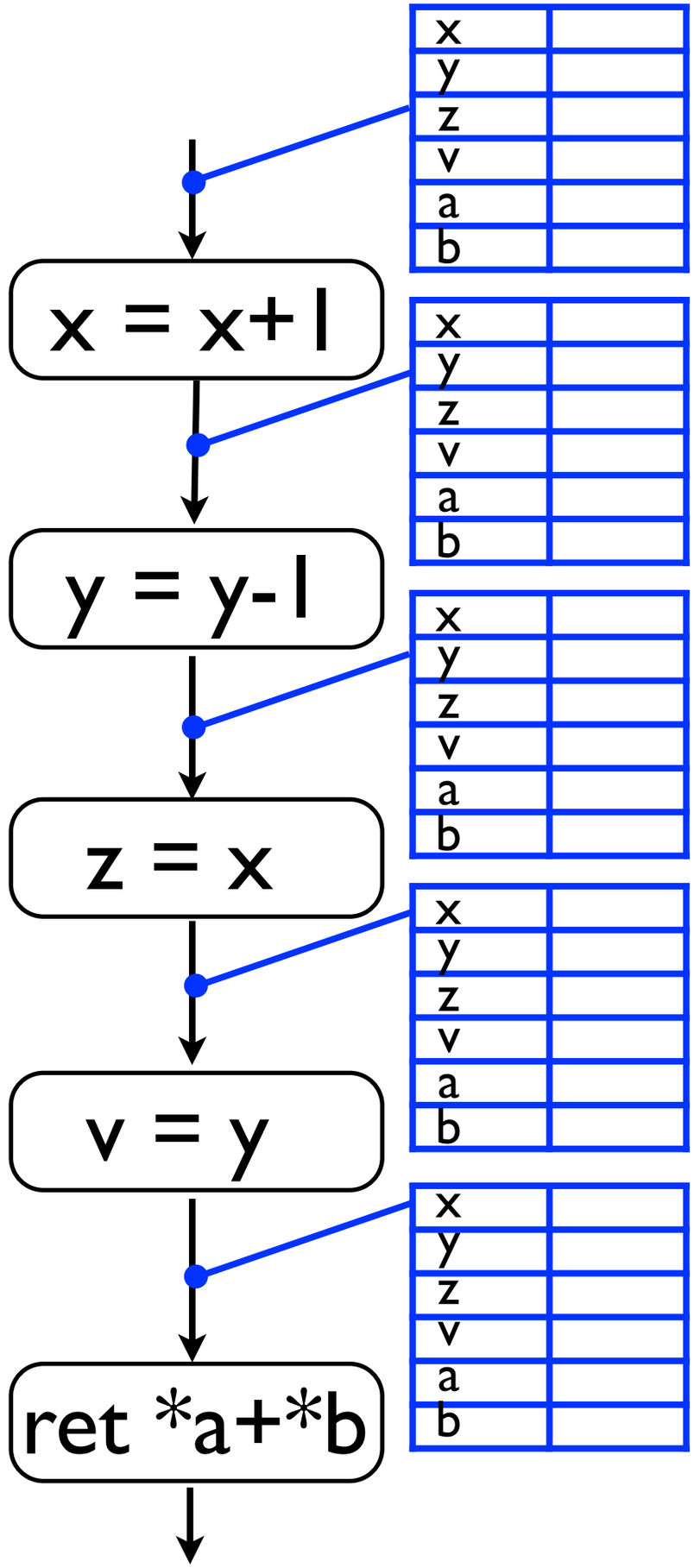
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

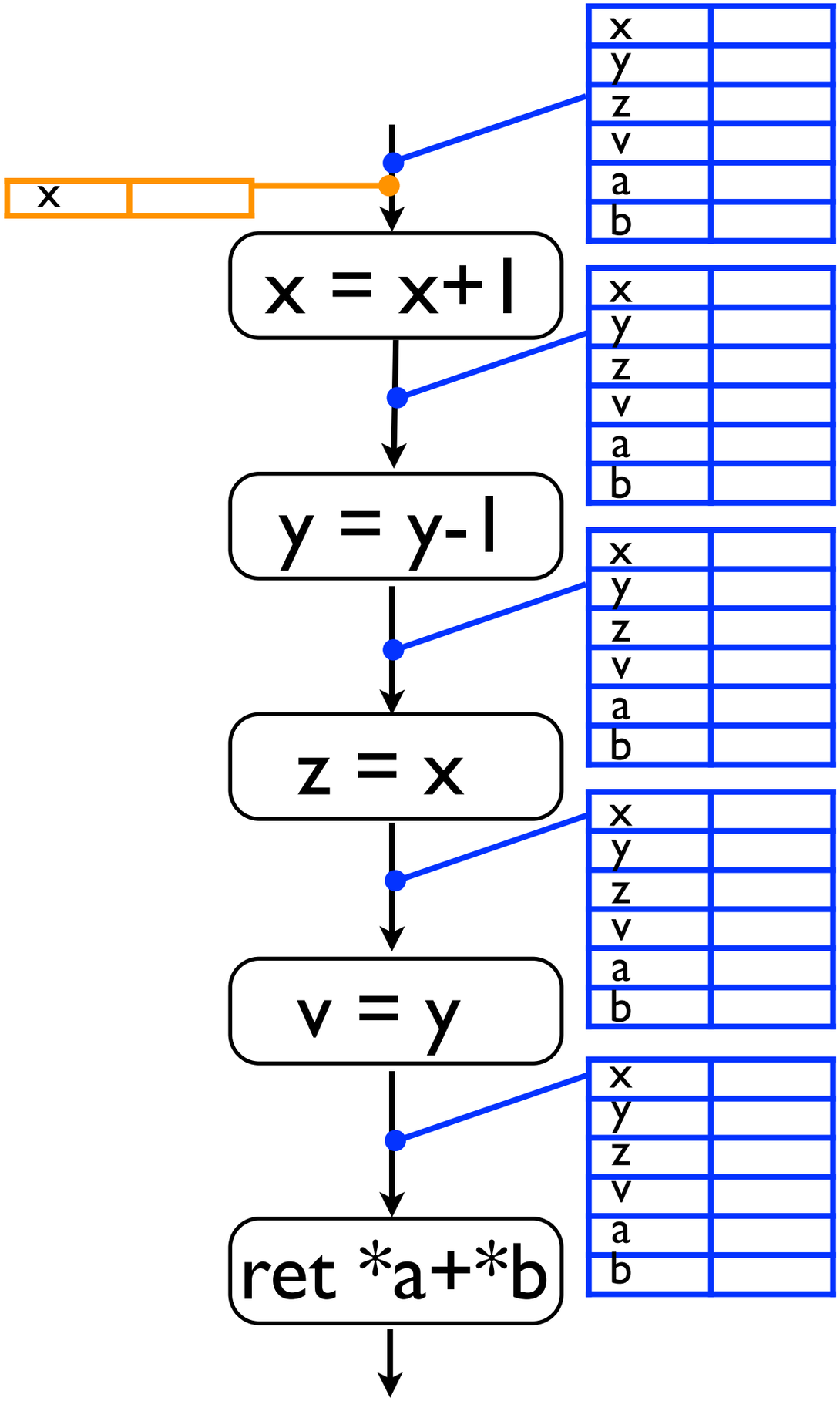
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

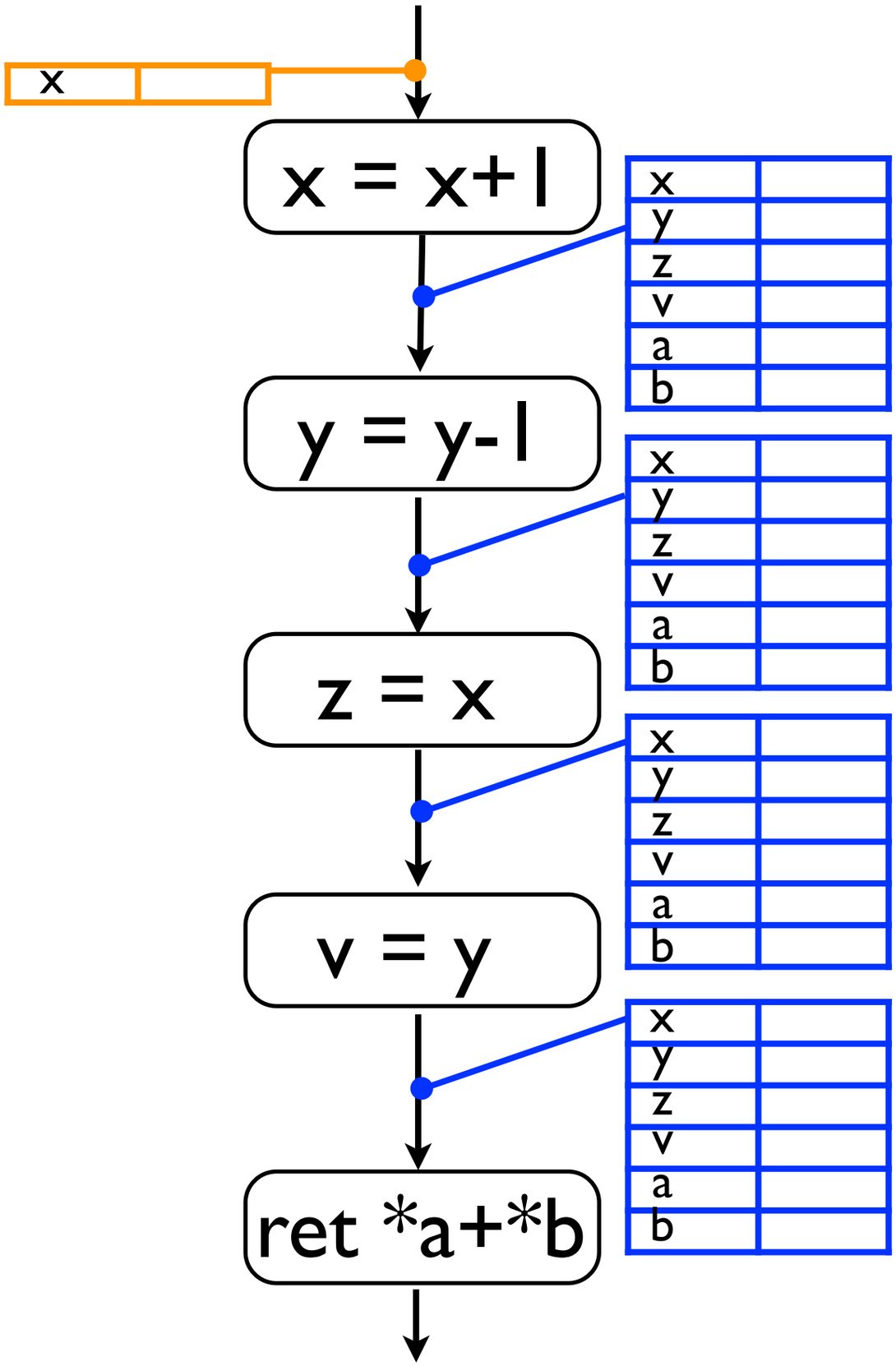
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

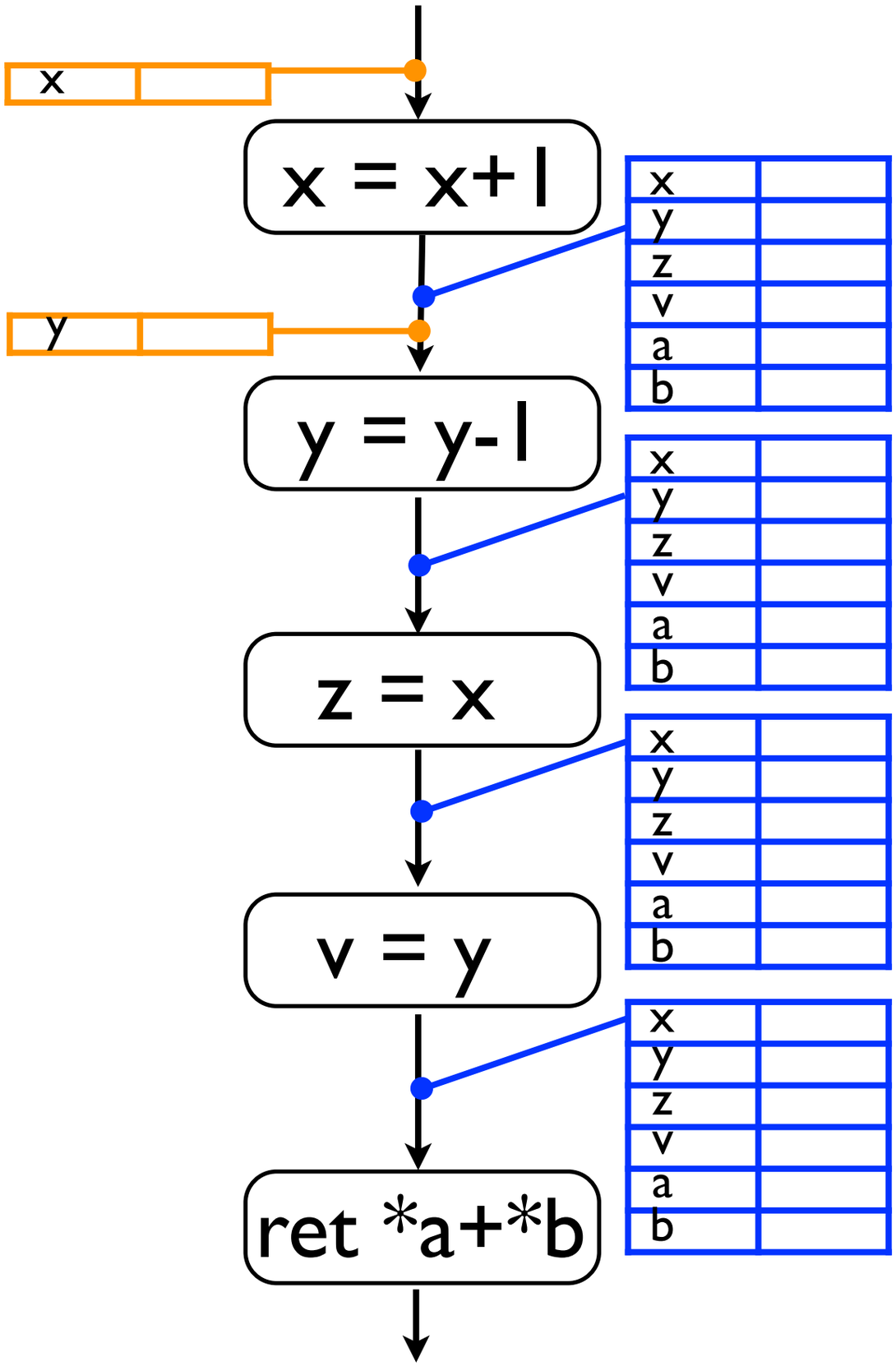
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

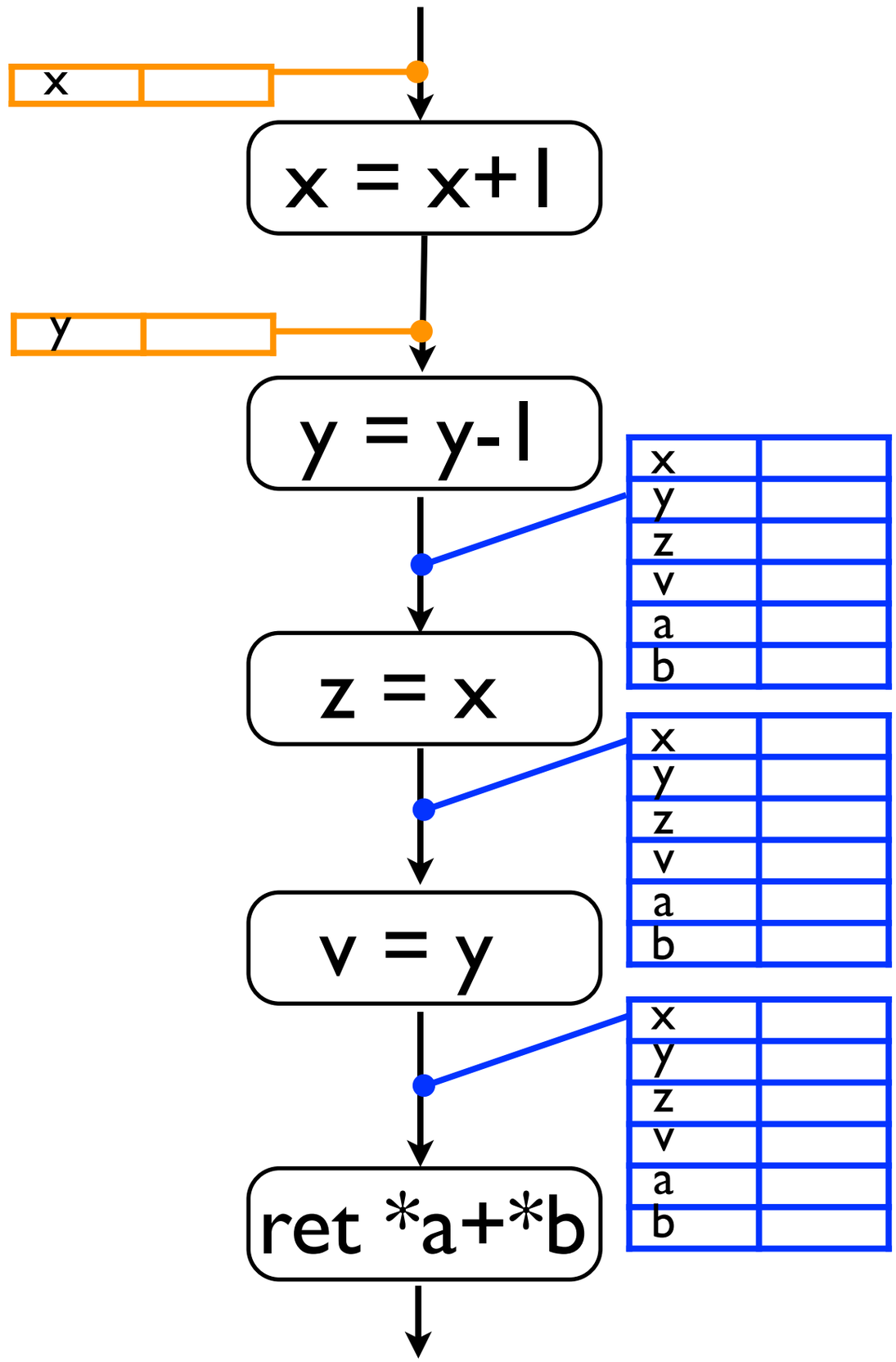
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

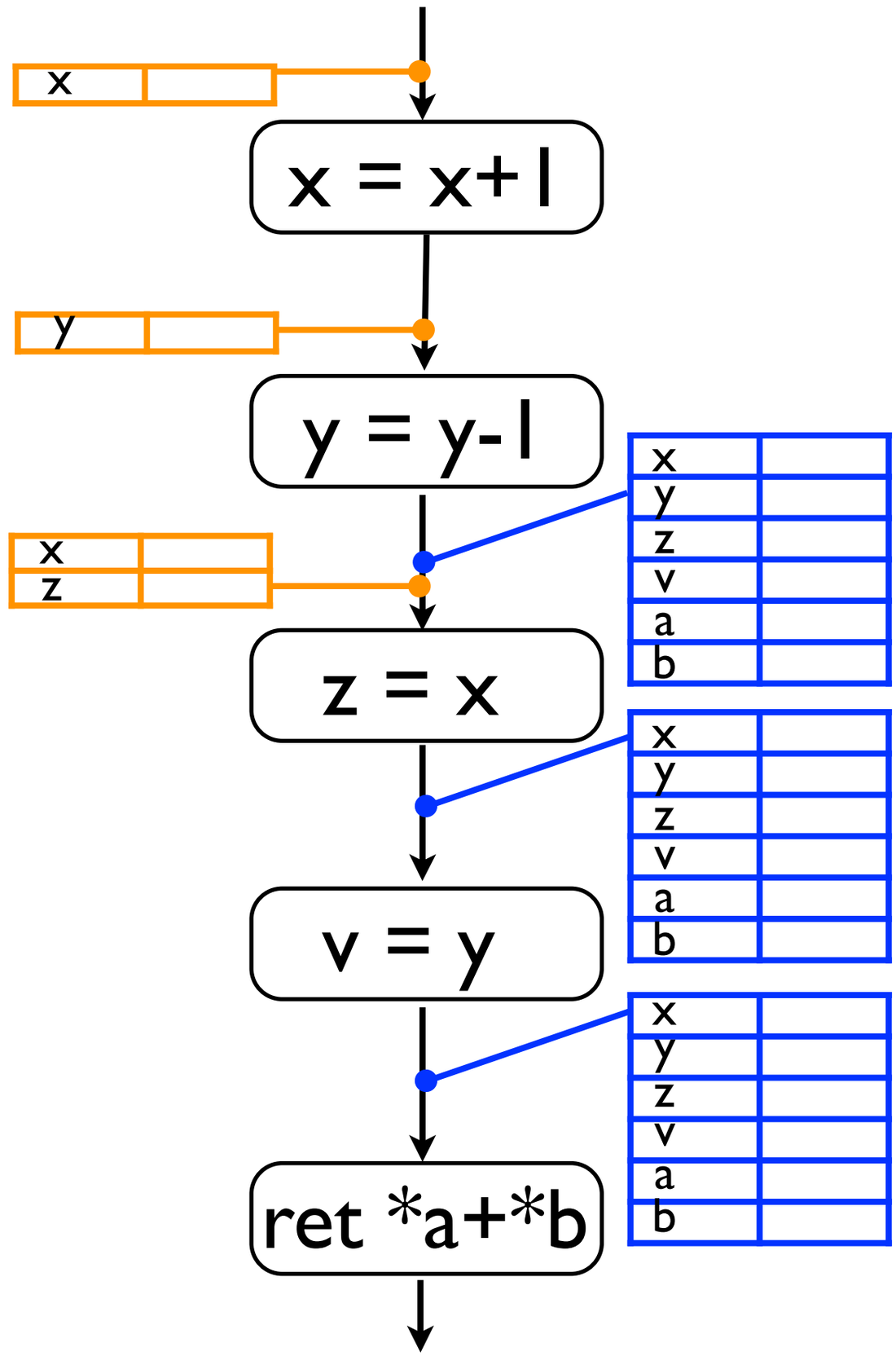
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

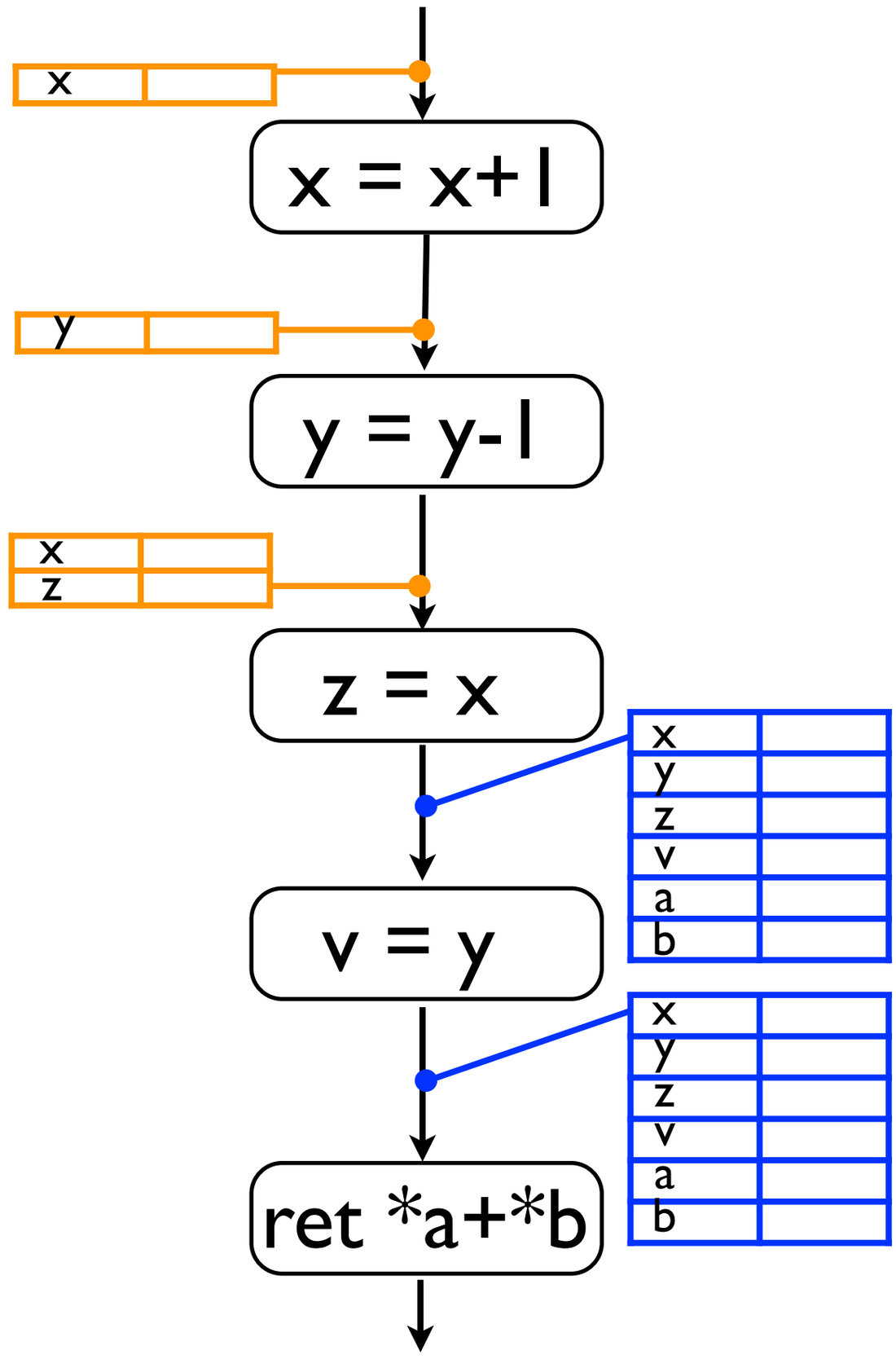
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

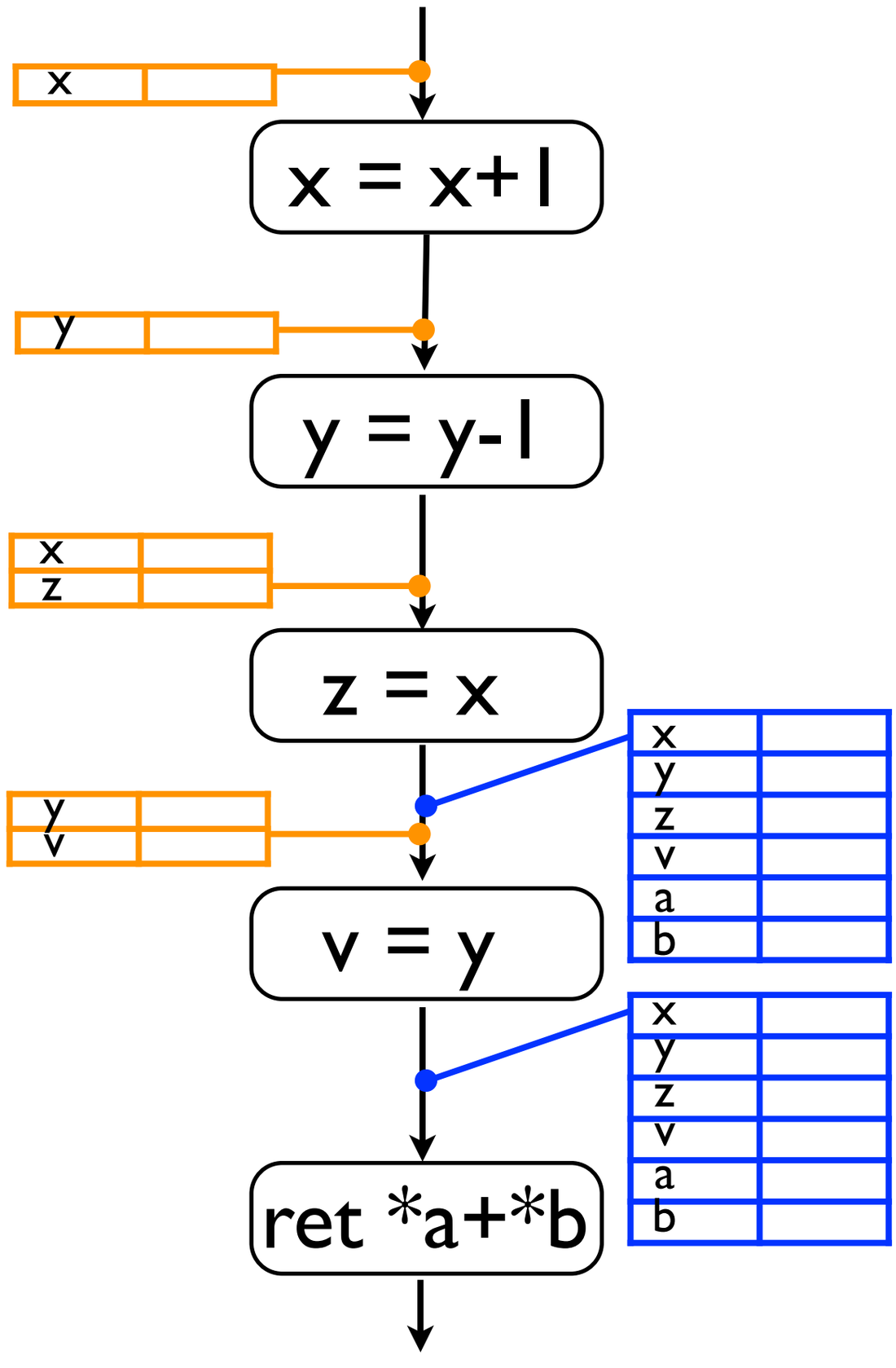
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

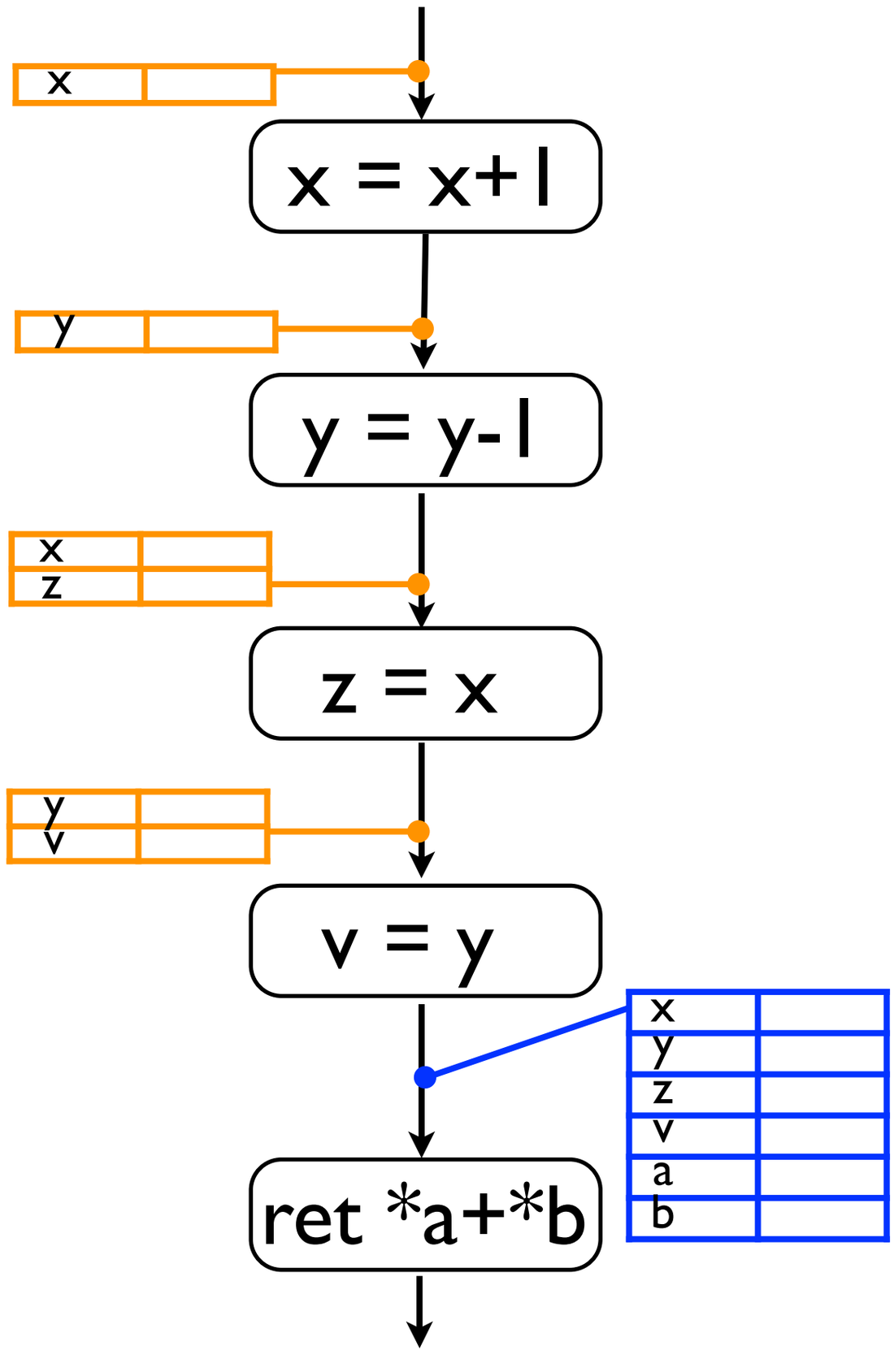
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

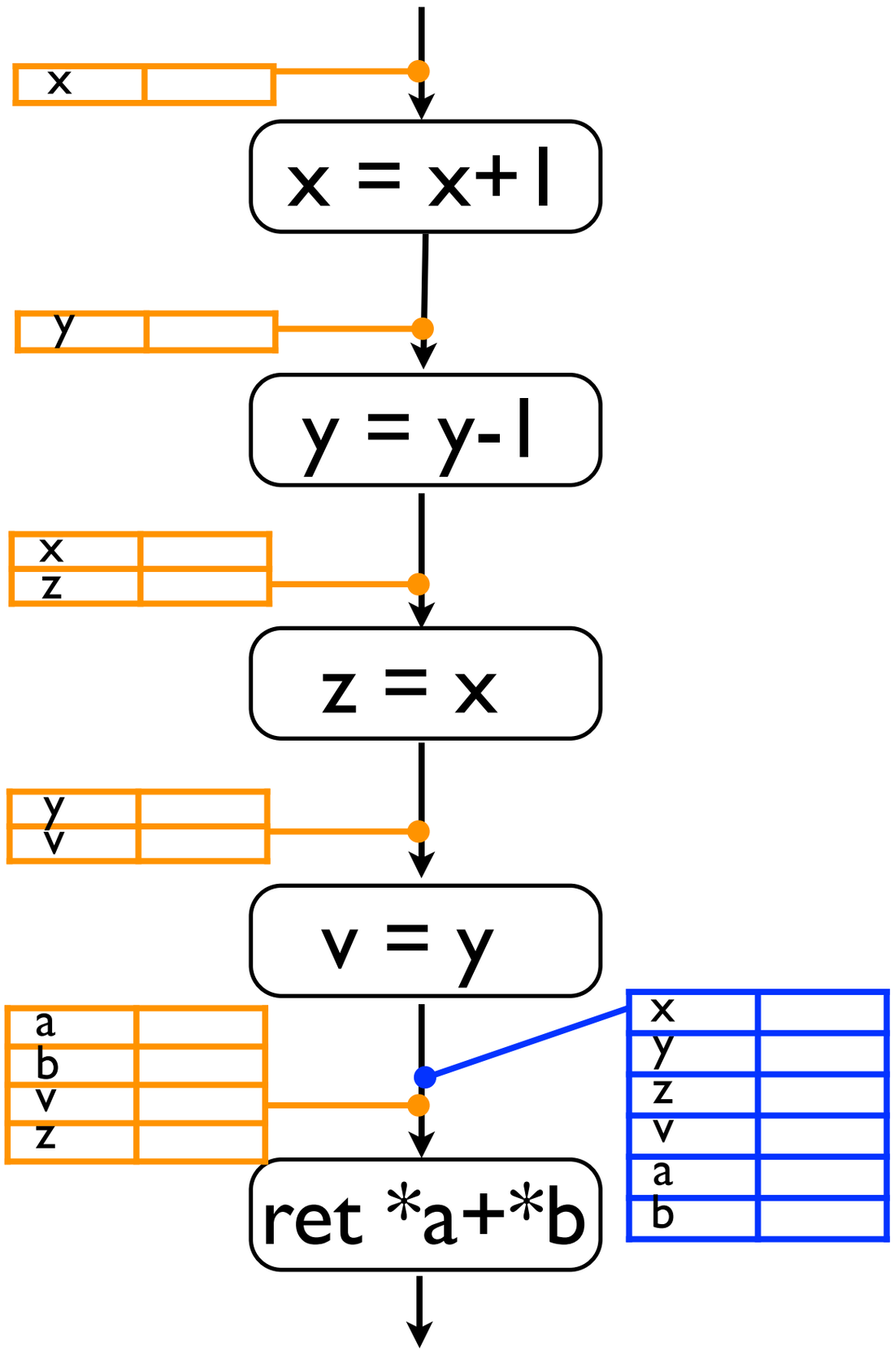
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

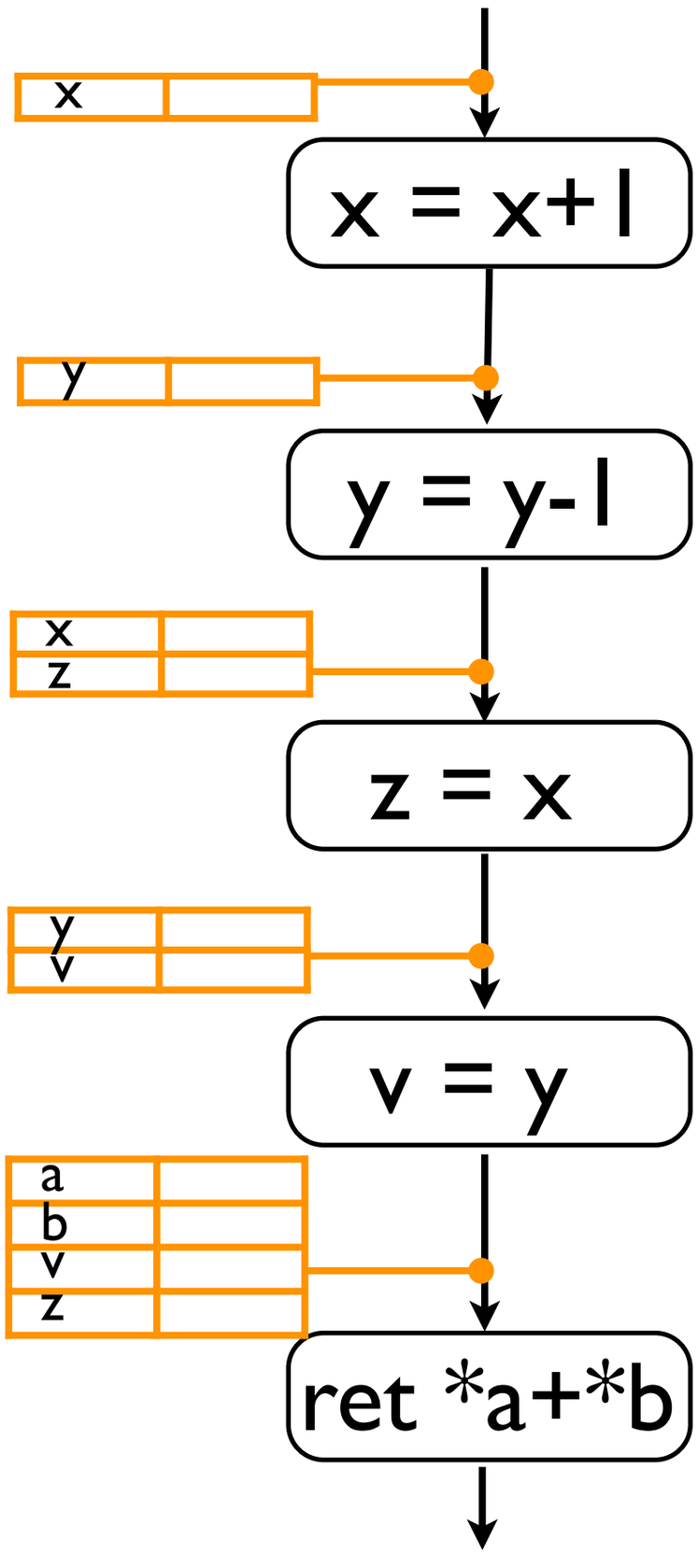
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

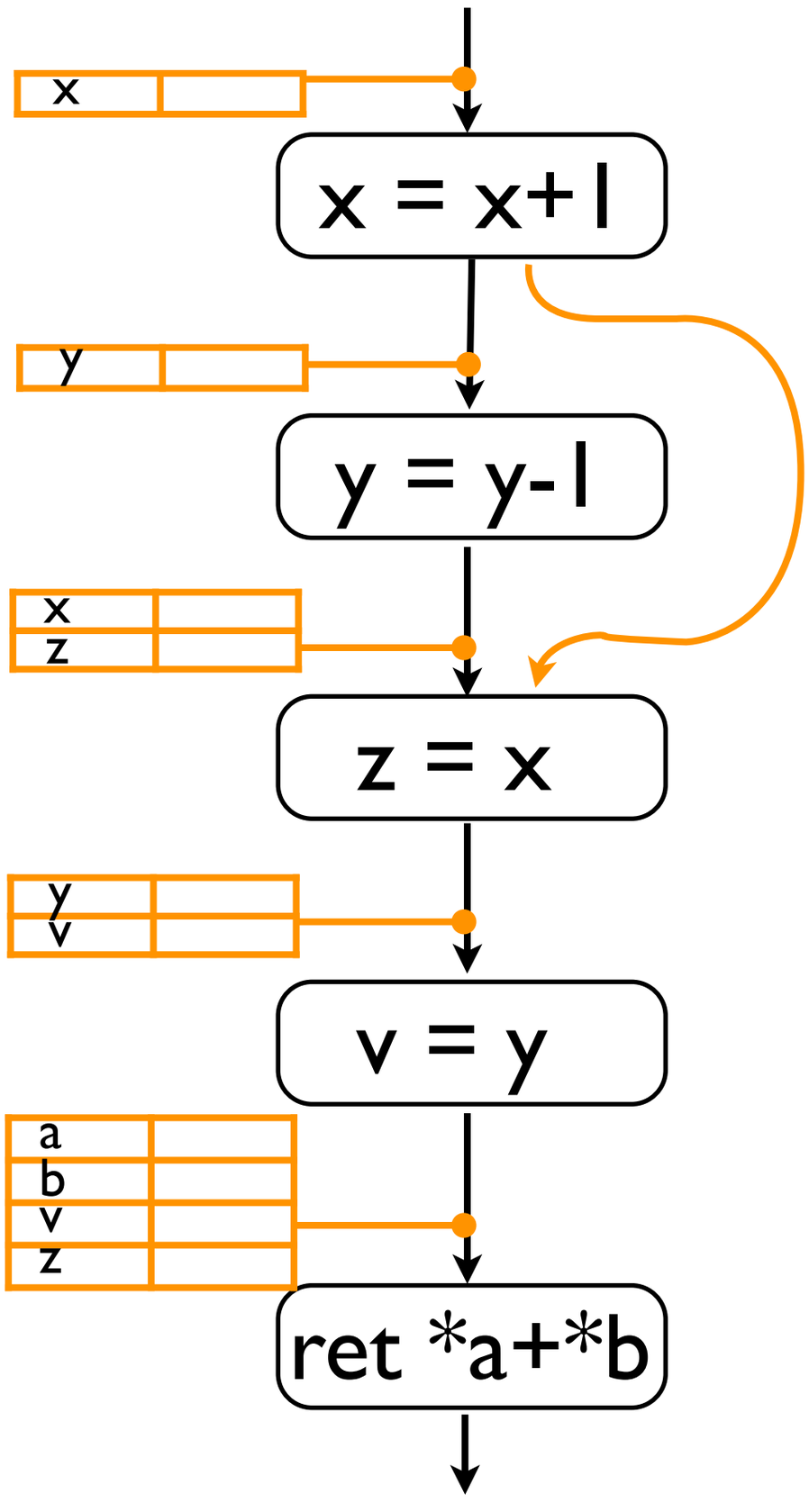
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

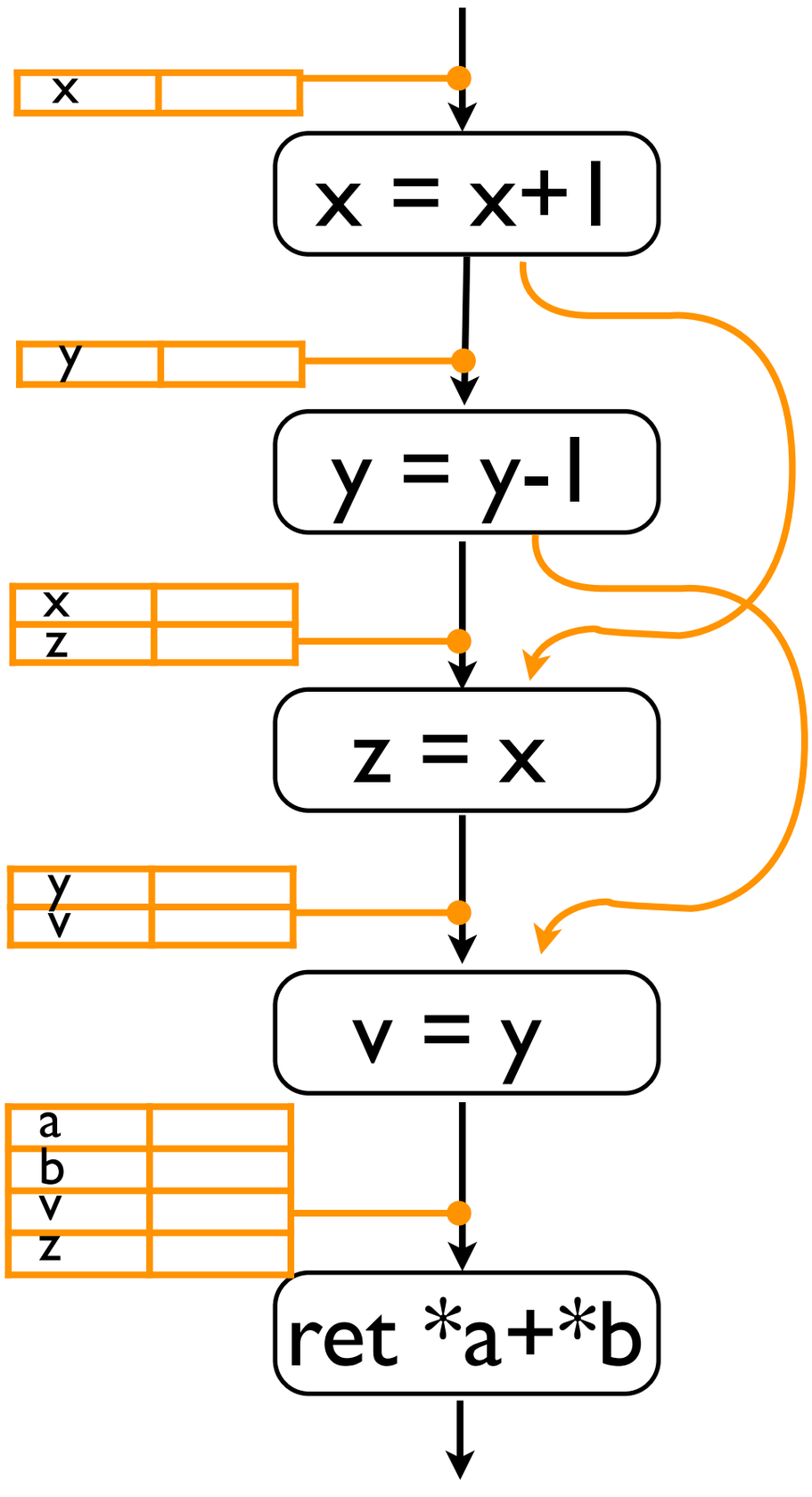
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

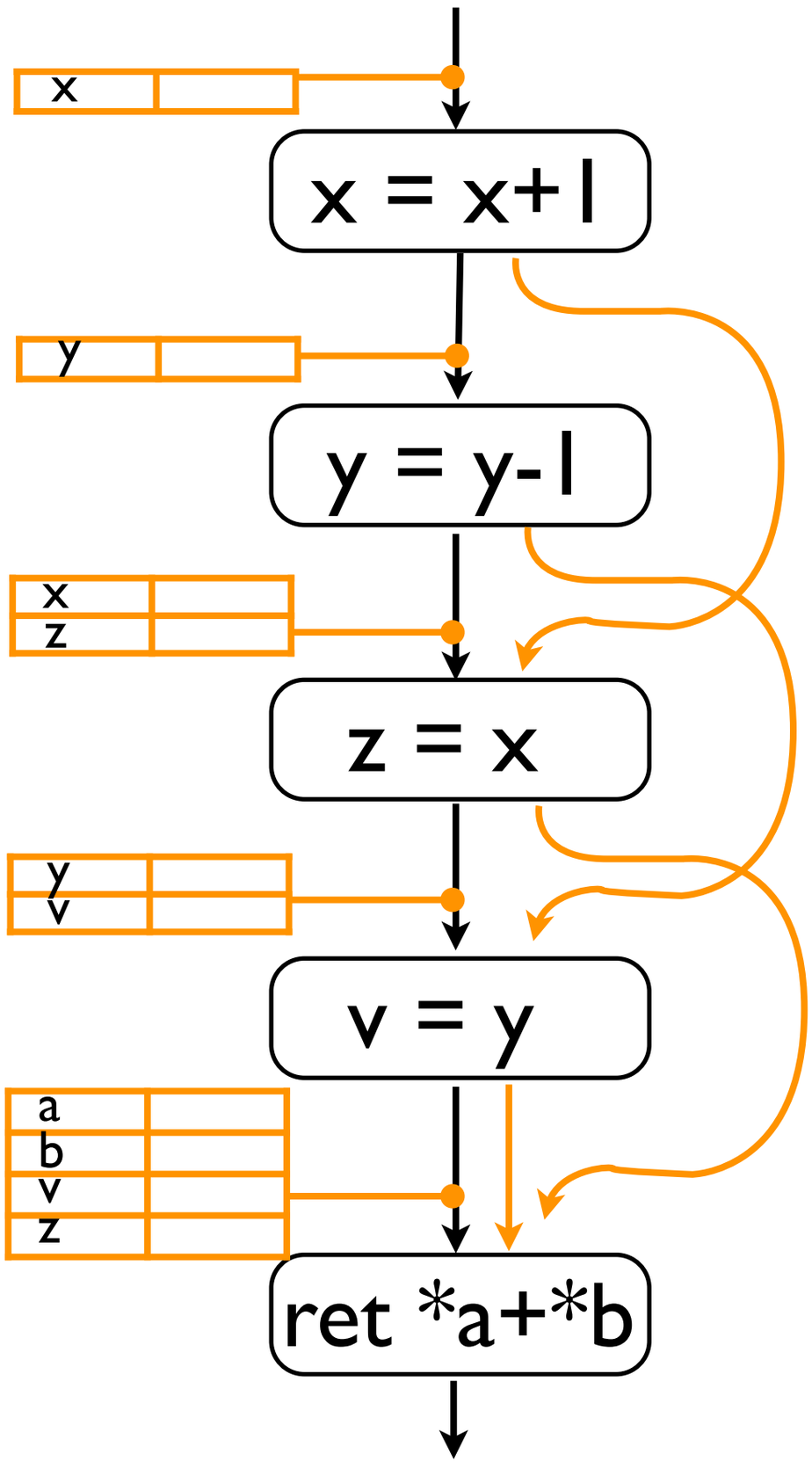
"Right Part at Right Moment"



PLDI'12

Key: General Sparse Analysis

"Right Part at Right Moment"



PLDI'12

General Sparse Analysis Framework

Thm. (preservation of soundness and precision)

$$\hat{F} : \hat{D} \to \hat{D} \quad \xRightarrow{\text{sparsify}} \quad \hat{F}_s : \hat{D} \to \hat{D}$$

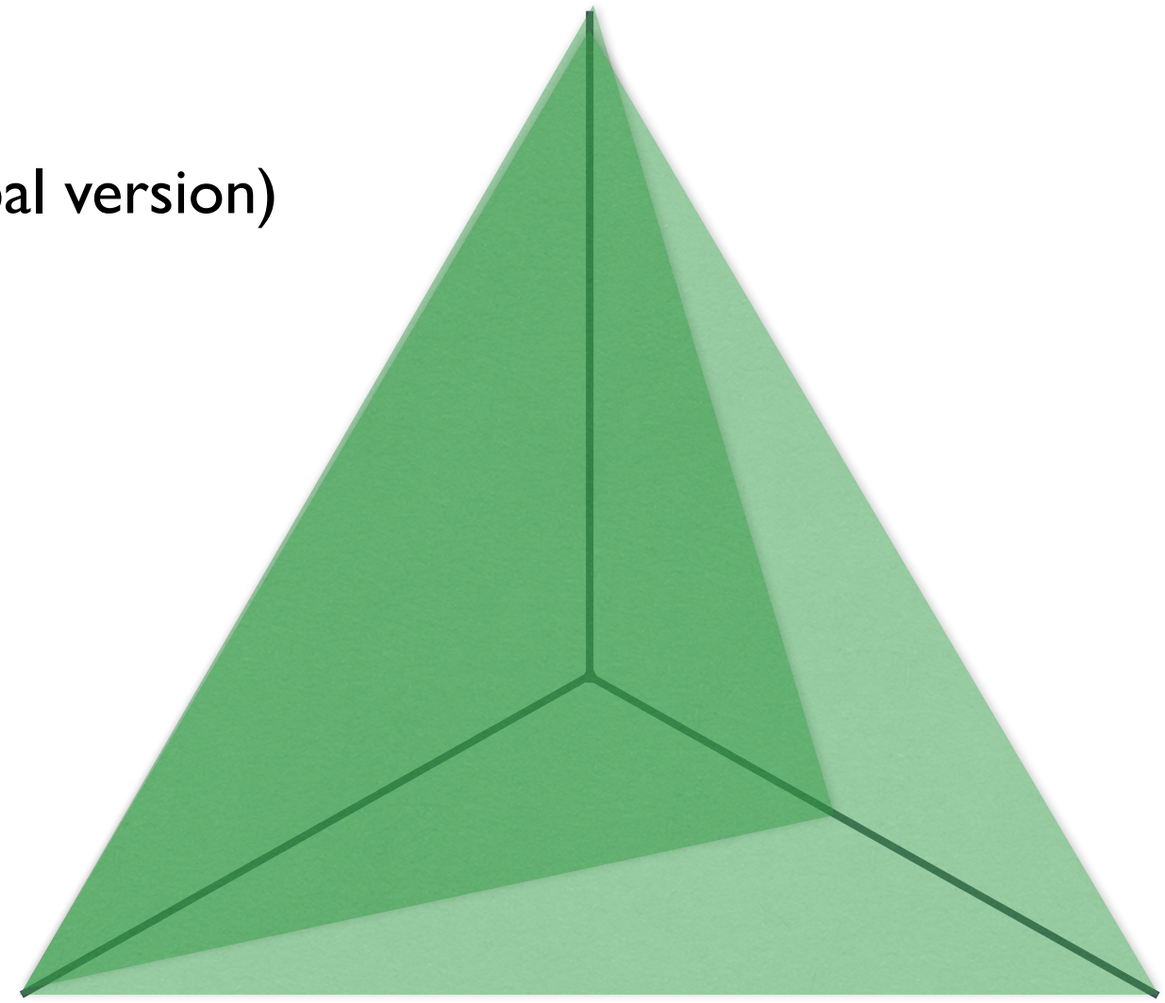
$$fix\,\hat{F} \quad = \quad fix\,\hat{F}_s$$

*"An important strength is that the **theoretical result** is **very general** ... The result should be **highly influential** on future work in sparse analysis."* *(from PLDI reviews)*

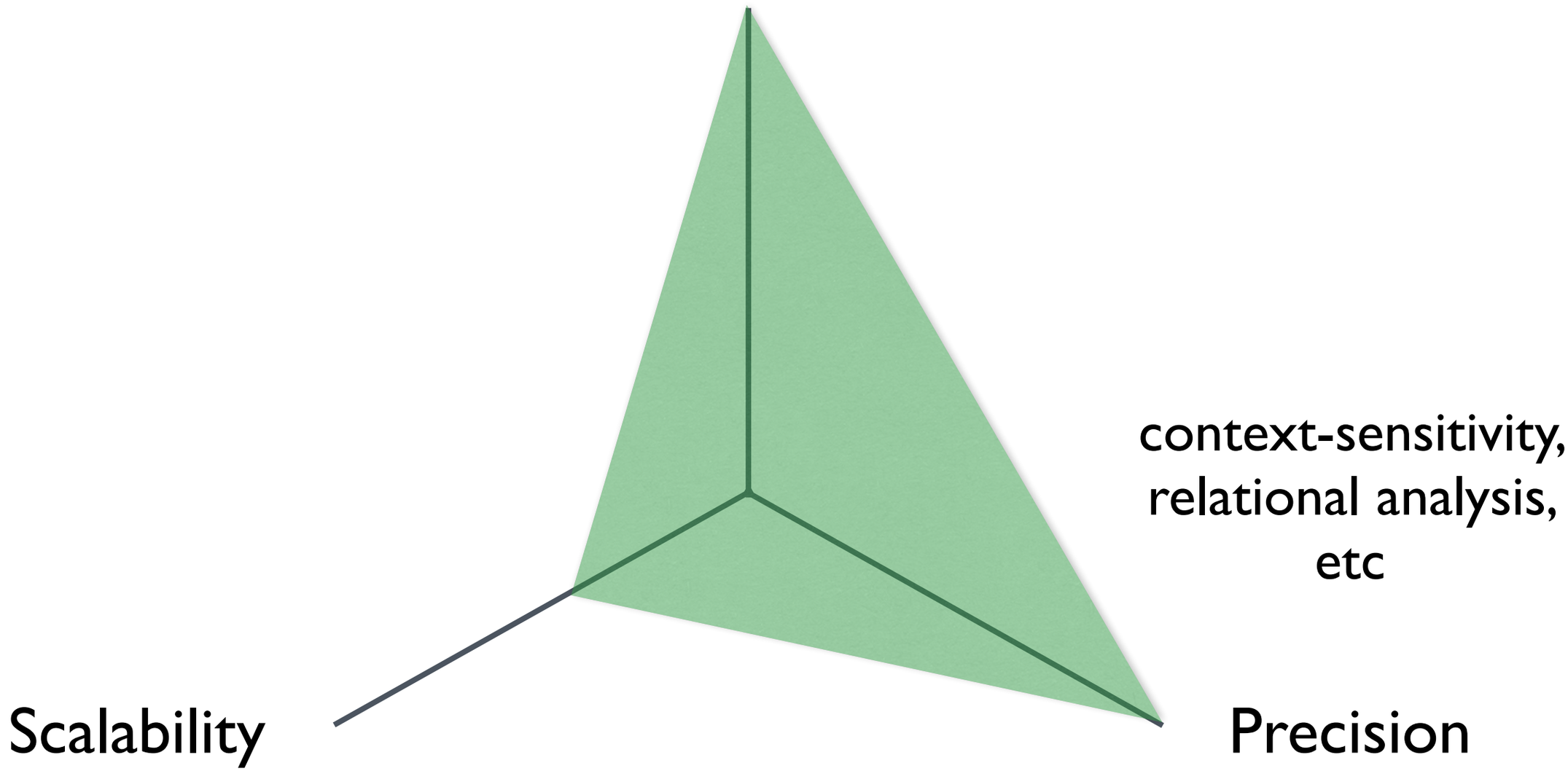
The Second Goal: Precision

Sparrow
The Early Bird

(2012, sound-&-global version)



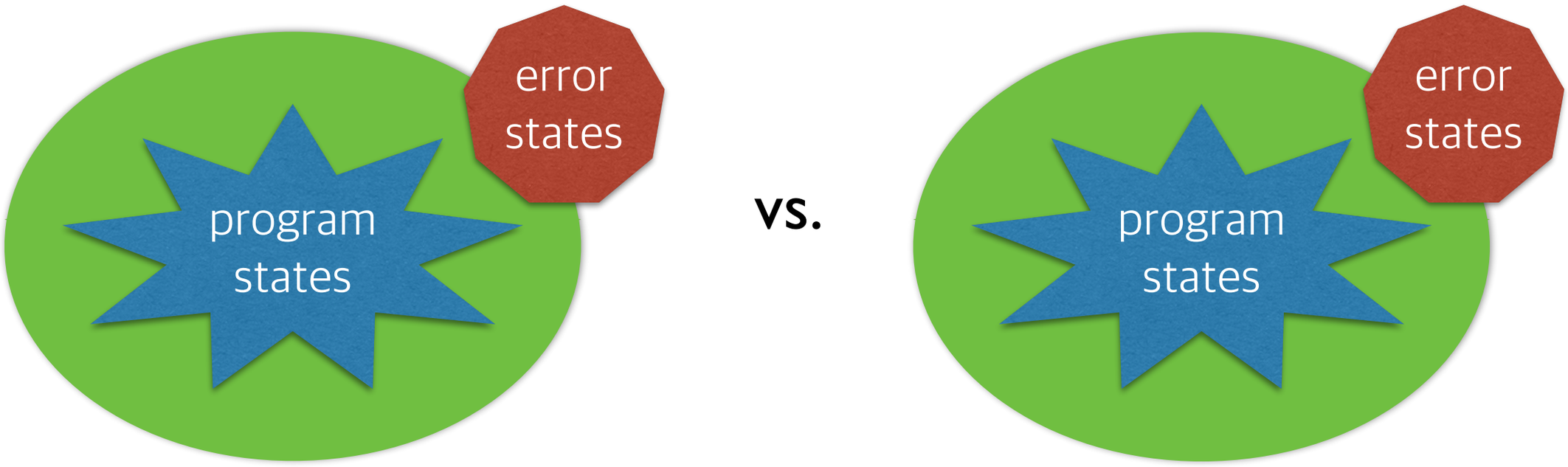
cf) Existing Techniques



PLDI'14

Selective X-Sensitivity Framework

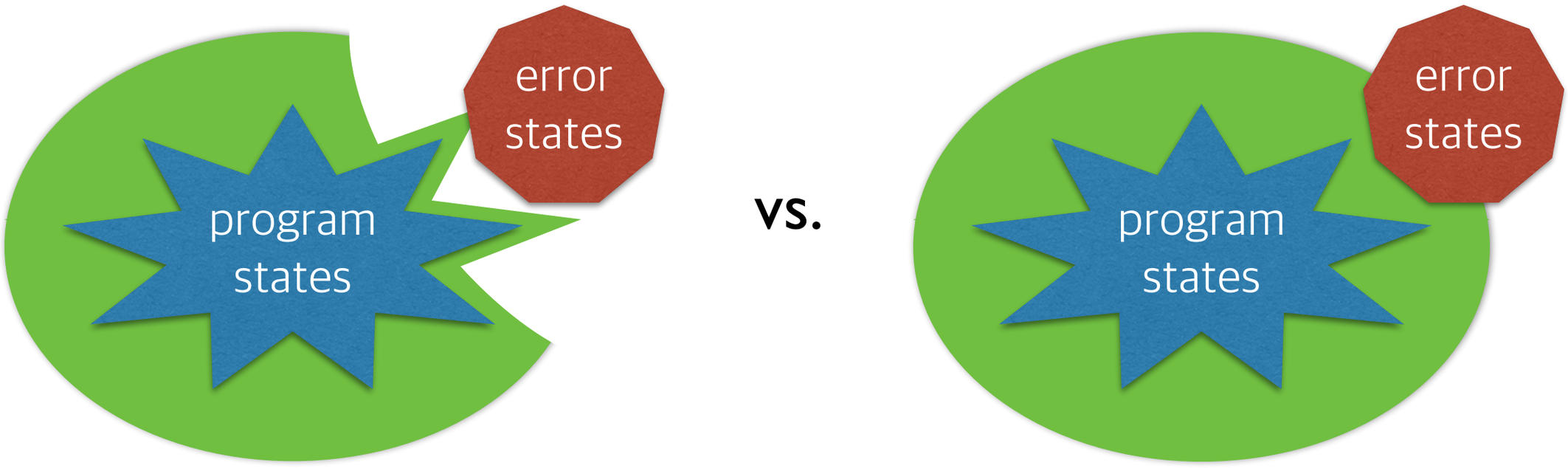
- **Key Idea**: Improve precision only when it matters



PLDI'14

Selective X-Sensitivity Framework

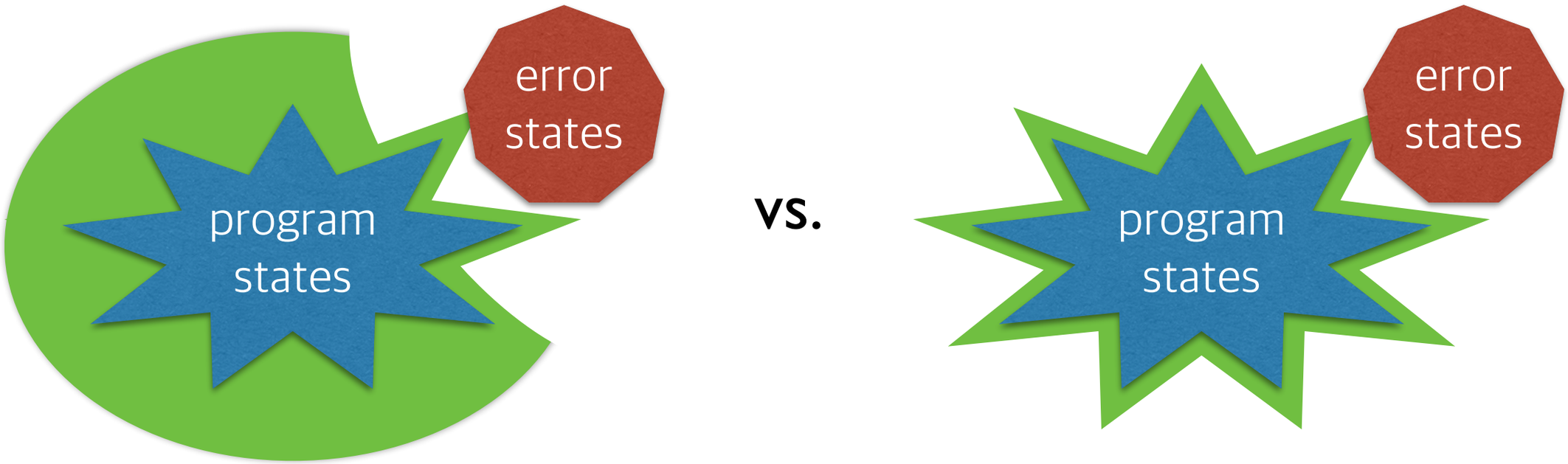
- **Key Idea**: Improve precision only when it matters



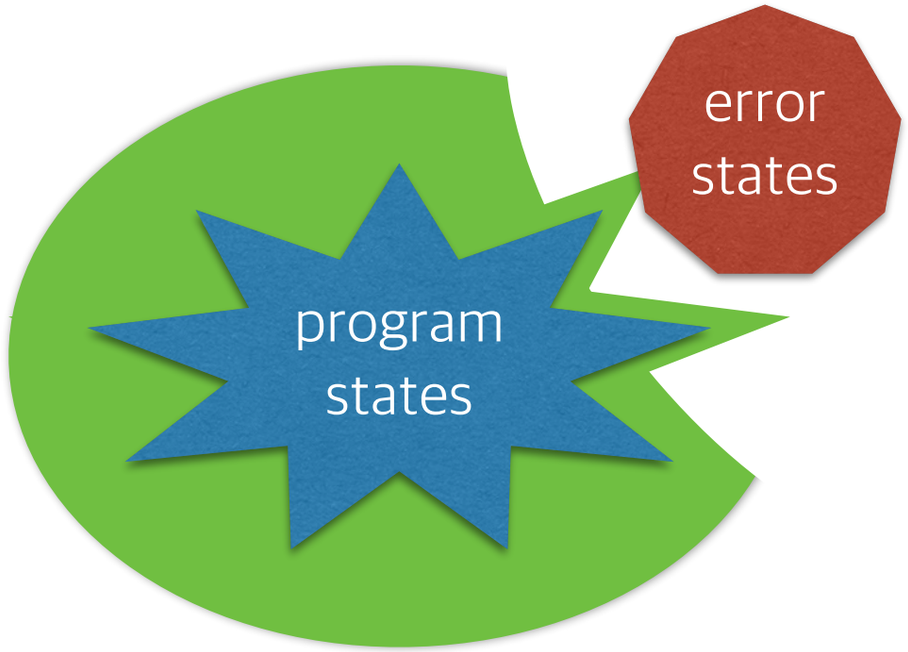
PLDI'14

Selective X-Sensitivity Framework

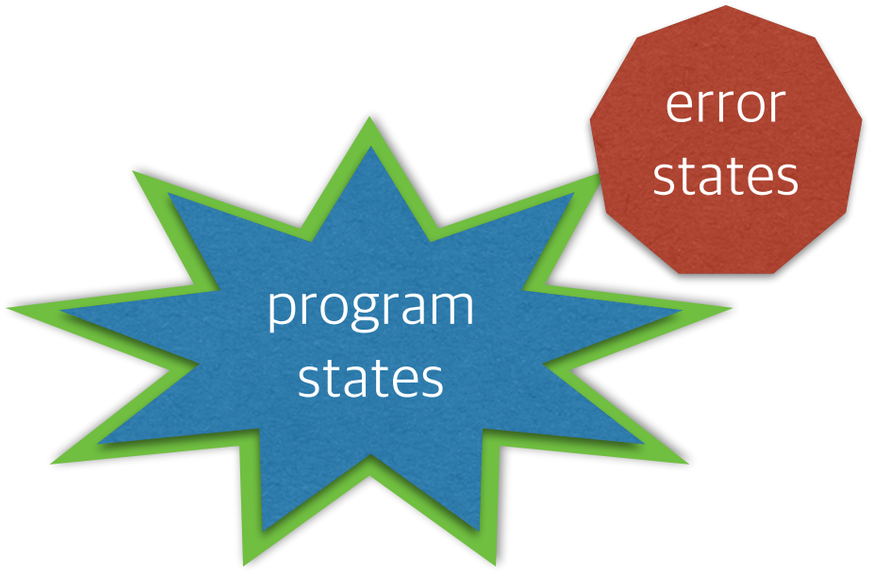
- **Key Idea**: Improve precision only when it matters



Effectiveness



vs.



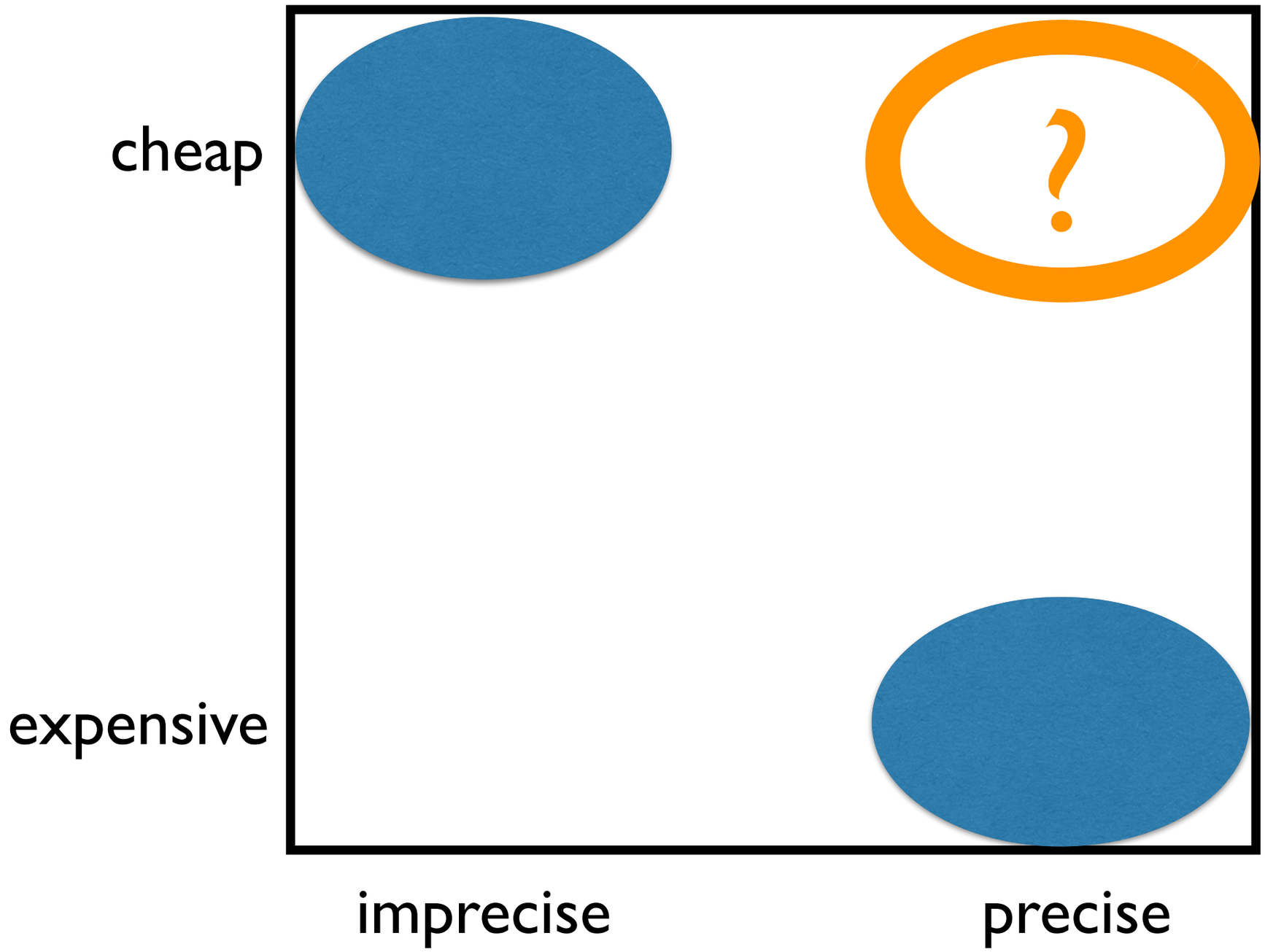
+25% / -25%

+25% / -1300%

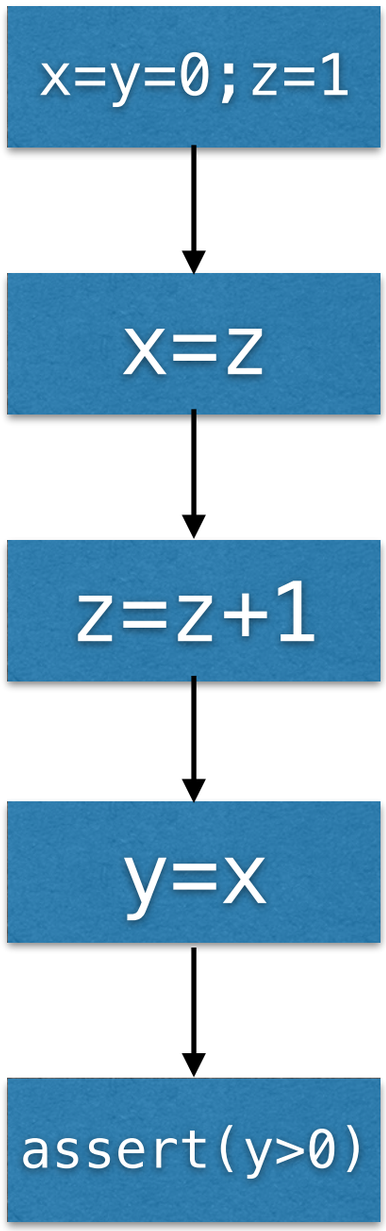
Recruiting Research Interns

- Programming system design (PL)
- High performance static analysis (PL + Algorithm)
- Data-driven program analysis (PL + ML)
- Software security (PL + Security)

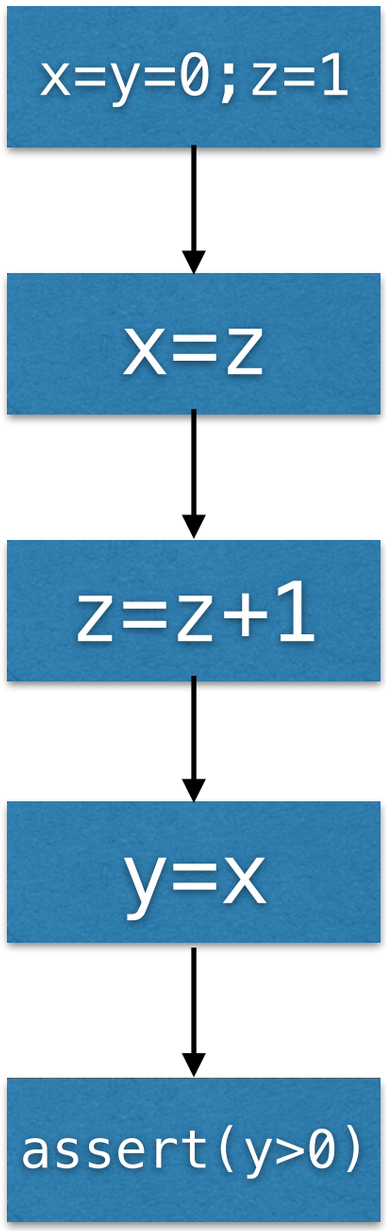
Challenge in Static Analysis



ex) Flow-Sensitivity

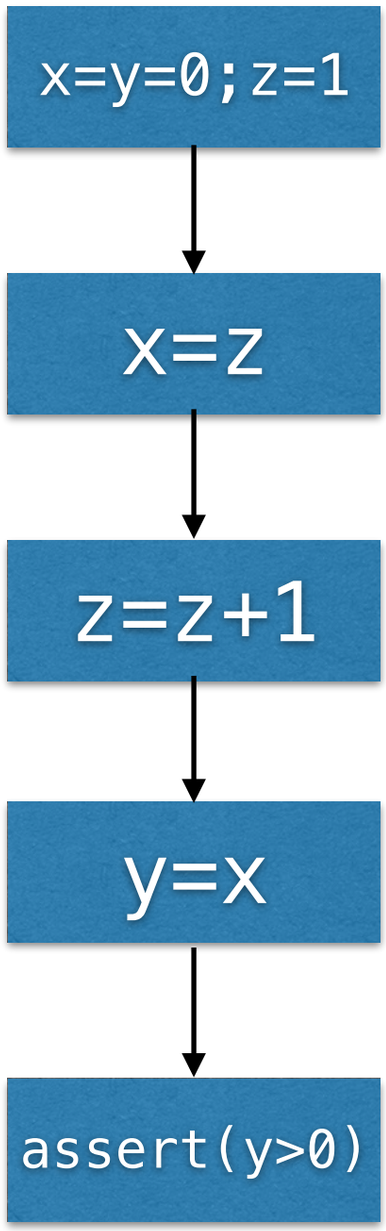


ex) Flow-Sensitivity



x	[0,0]
y	[0,0]
z	[1,1]

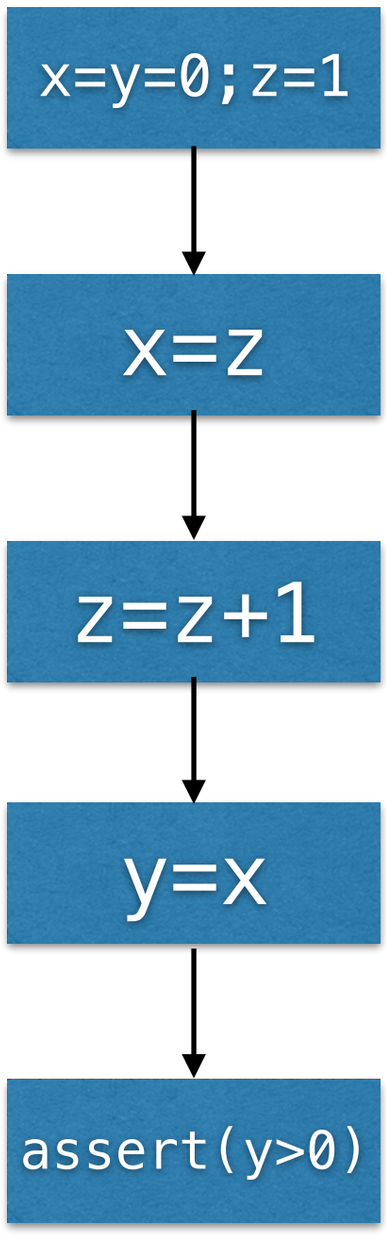
ex) Flow-Sensitivity



x	[0,0]
y	[0,0]
z	[1,1]

x	[1,1]
y	[0,0]
z	[1,1]

ex) Flow-Sensitivity

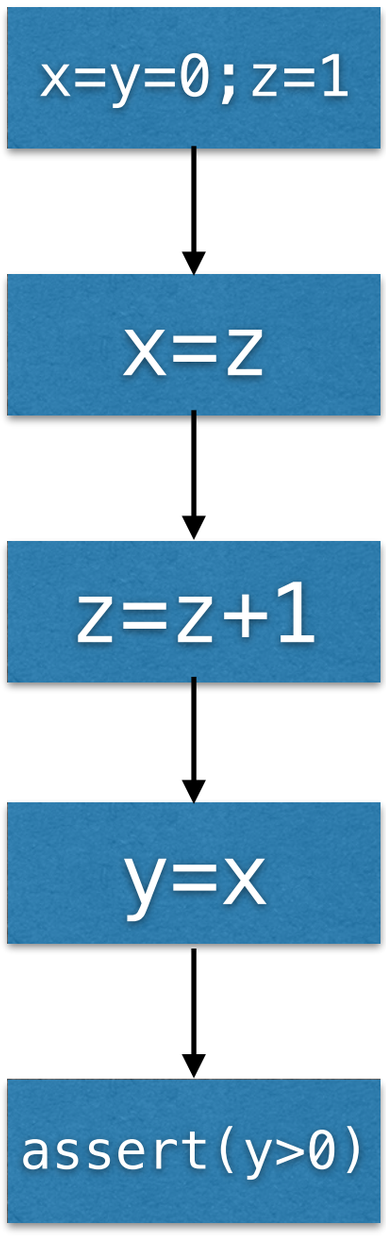


x	[0,0]
y	[0,0]
z	[1,1]

x	[1,1]
y	[0,0]
z	[1,1]

x	[1,1]
y	[0,0]
z	[2,2]

ex) Flow-Sensitivity



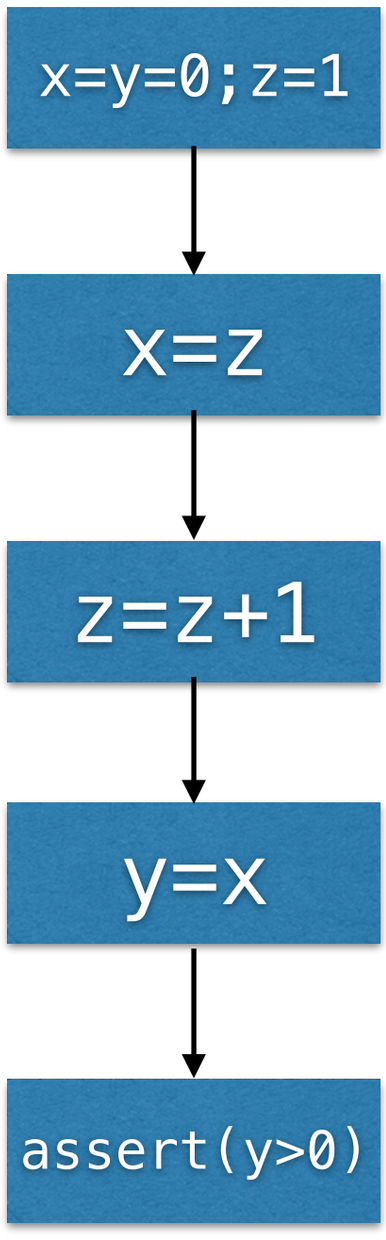
x	[0,0]
y	[0,0]
z	[1,1]

x	[1,1]
y	[0,0]
z	[1,1]

x	[1,1]
y	[0,0]
z	[2,2]

x	[1,1]
y	[1,1]
z	[2,2]

ex) Flow-Sensitivity



x	[0,0]
y	[0,0]
z	[1,1]

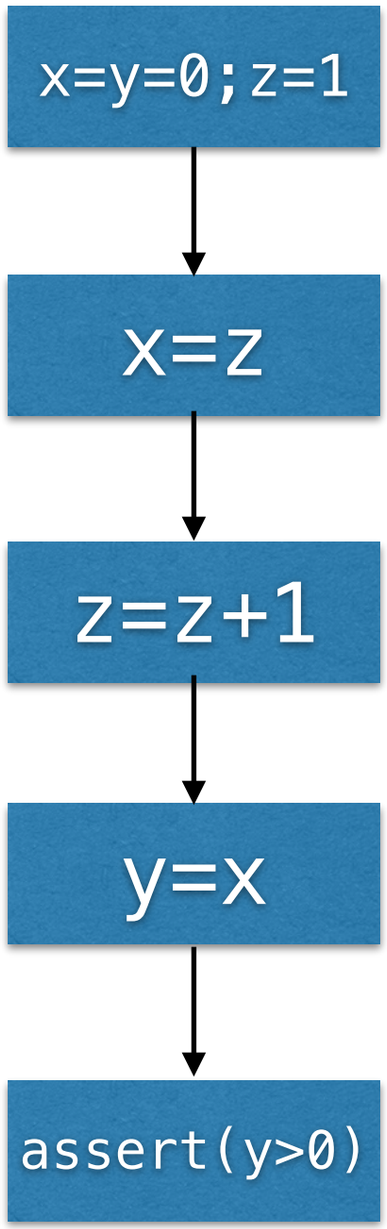
x	[1,1]
y	[0,0]
z	[1,1]

x	[1,1]
y	[0,0]
z	[2,2]

x	[1,1]
y	[1,1]
z	[2,2]

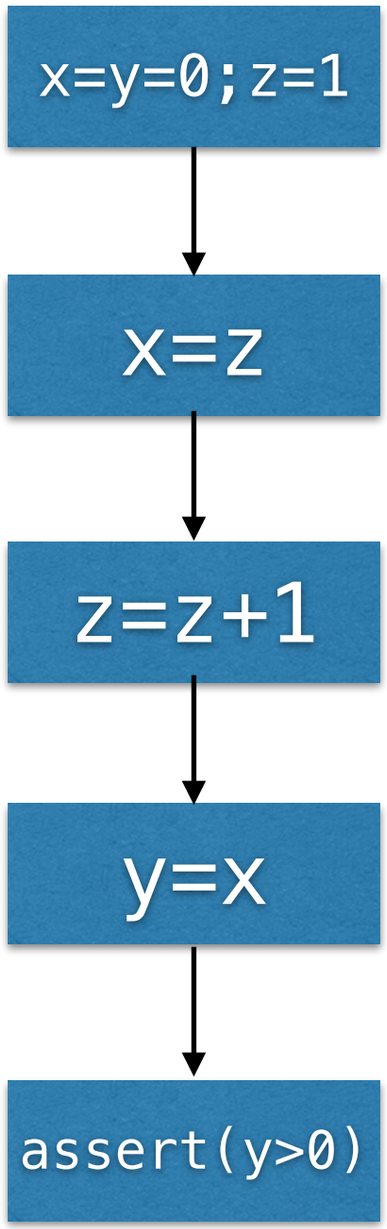
precise but costly

Flow-Insensitivity



x	[0,+∞]
y	[0,+∞]
z	[1,+∞]

Flow-Insensitivity

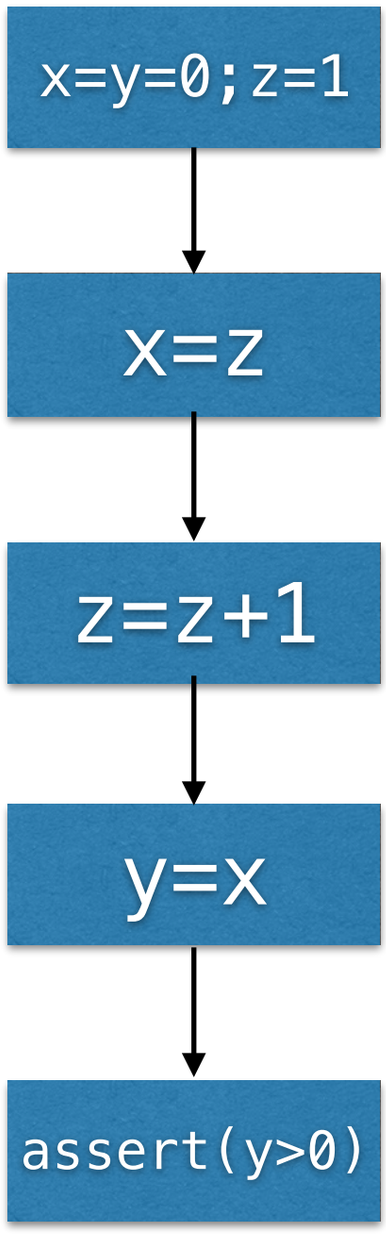


x	[0,+∞]
y	[0,+∞]
z	[1,+∞]

cheap but imprecise

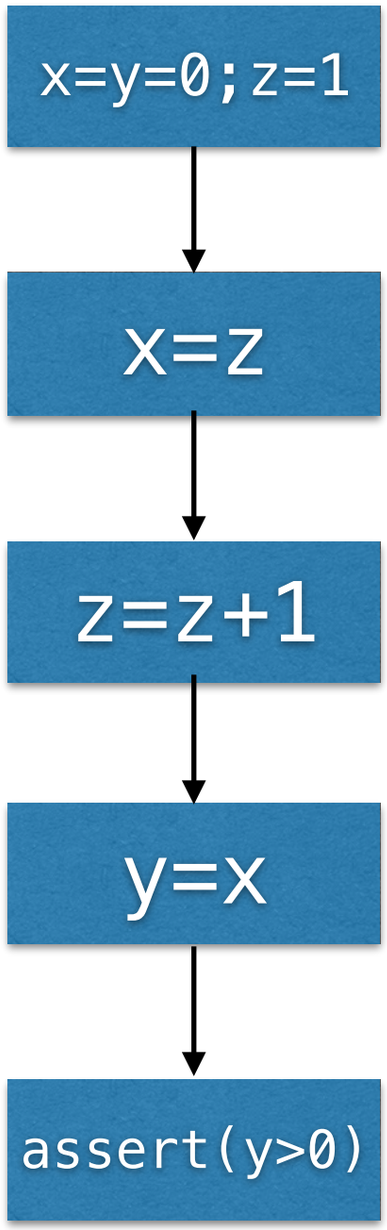
FS : {x}

FI : {y,z}



FS : {x}

FI : {y,z}



x	[0,0]

x	[1,+∞]

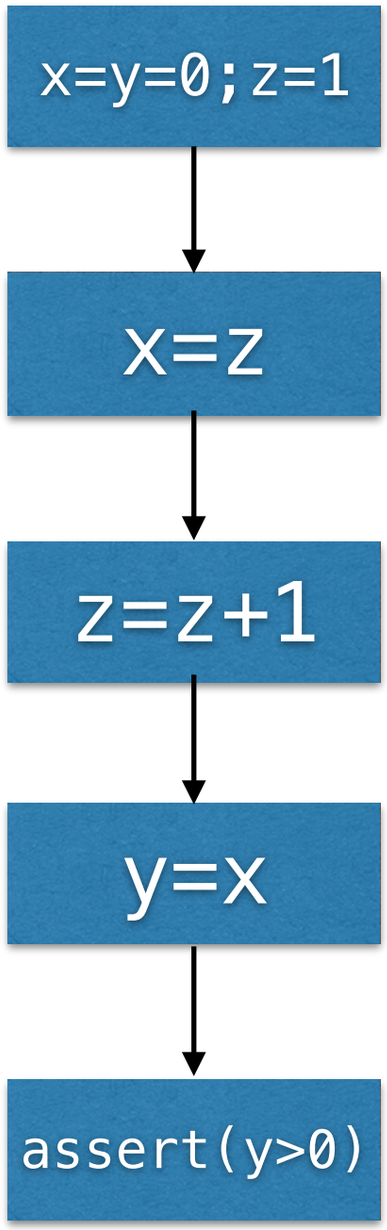
x	[1,+∞]

x	[1,+∞]

y	[0,+∞]
z	[1,+∞]

FS : {x}

FI : {y,z}



x	[0,0]

x	[1,+∞]

x	[1,+∞]

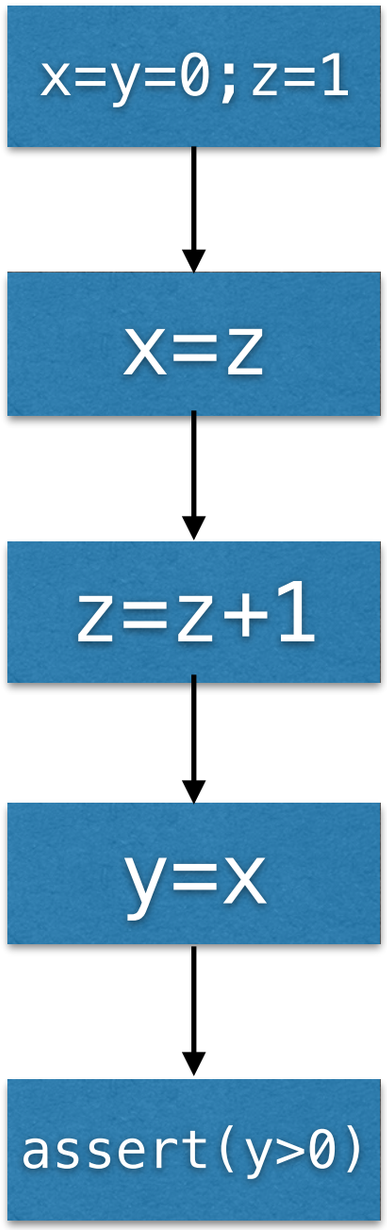
x	[1,+∞]

y	[0,+∞]
z	[1,+∞]

fail to prove

FS : {y}

FI : {x,z}



y	[0,0]

y	[0,0]

y	[0,0]

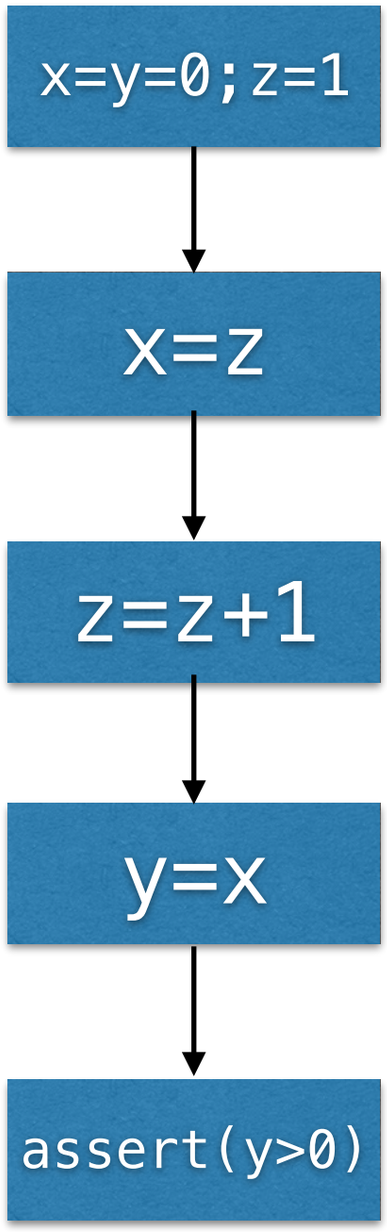
y	[0,+∞]

x	[0,+∞]
z	[1,+∞]

fail to prove

FS : {z}

FI : {x,y}



z	[1,1]

z	[1,1]

z	[2,2]

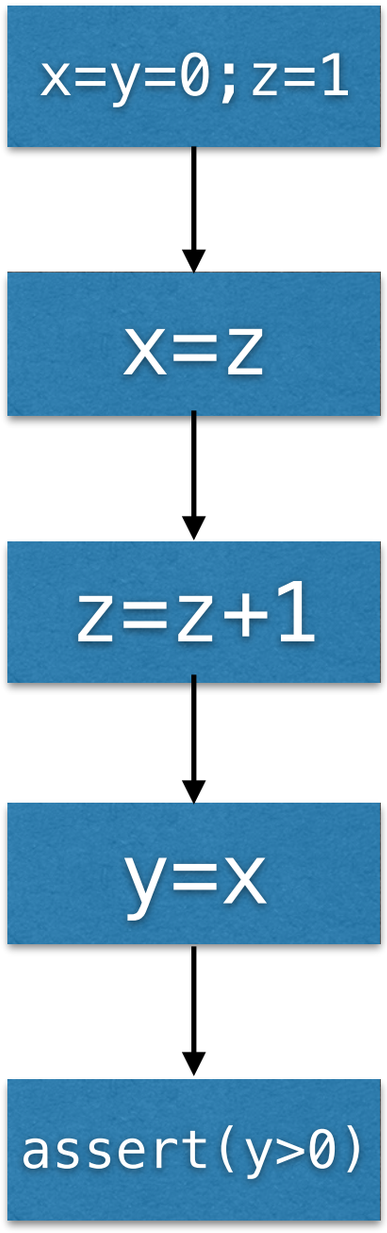
z	[2,2]

x	[0,+∞]
y	[0,+∞]

fail to prove

FS : {y,z}

FI : {x}



y	[0,0]
z	[1,1]

y	[0,0]
z	[1,1]

y	[0,0]
z	[2,2]

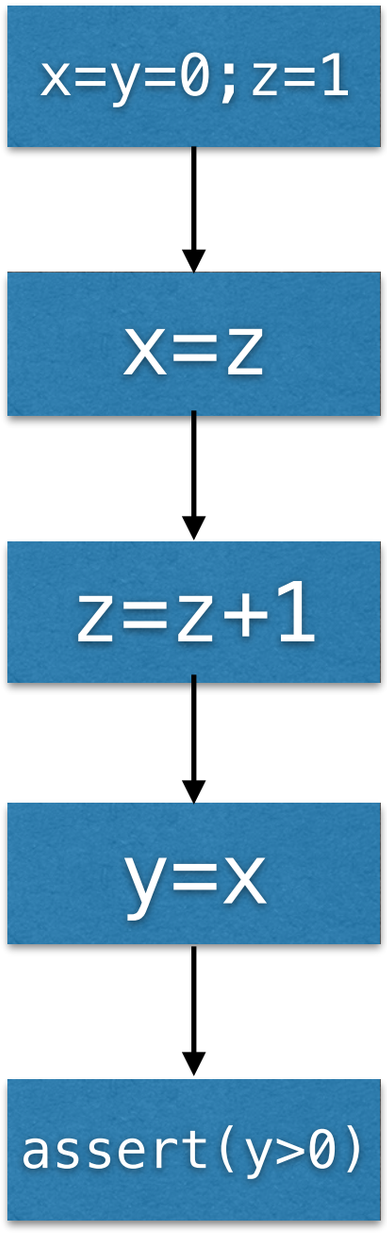
y	[0,+∞]
z	[0,0]

x	[0,+∞]

fail to prove

FS : {x,y}

FI : {z}



x	[0,0]
y	[0,0]

x	[1,+∞]
y	[0,0]

x	[1,+∞]
y	[0,0]

x	[1,+∞]
y	[1,+∞]

z	[1,+∞]

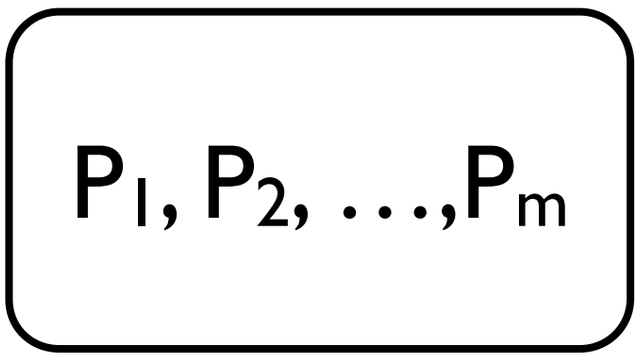
Success!

Hard Search Problem

- Intractably large space, if not infinite
  - 2^N different parameters in FS
- Most of them are too imprecise or costly
  - P({x,y,z}) = {~~∅~~,~~{x}~~,~~{y}~~,~~{z}~~,{x,y},~~{y,z}~~,~~{x,z}~~,{x,y,z}}

Our Learning-based Approach

- Learn model *M* from dataset



$\Longrightarrow \quad M : \text{pgm} \rightarrow 2^{\text{Var}}$

- For new program P, run static analysis with *M*(P)

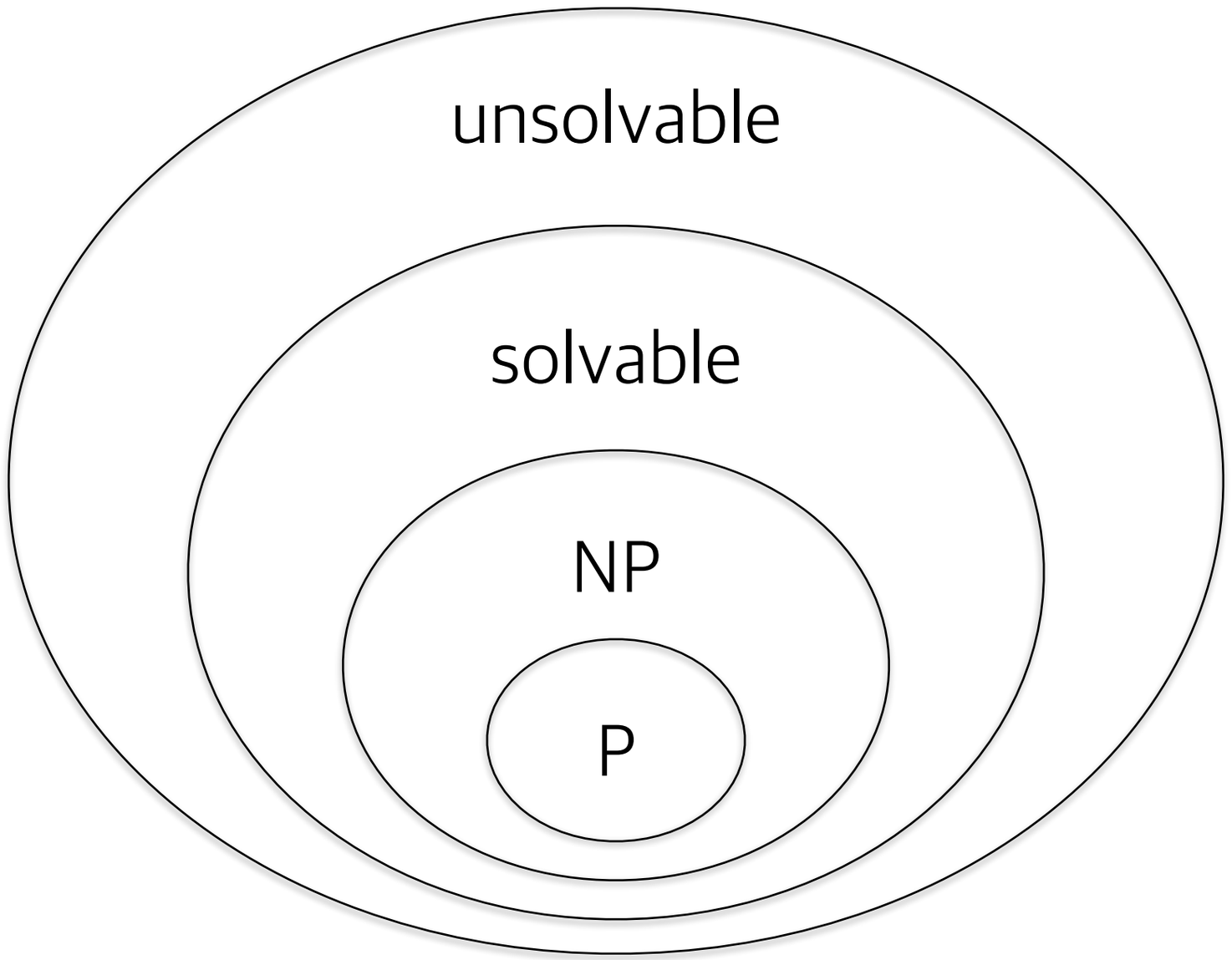
Effectiveness
(flow-sensitivity)

trials	FI		FS			selective FS			
	prove	time	prove	time	cost	prove	time	quality	cost
1	3312	75.8	4602	372.8	4.9 x	4509	100.6	92.8 %	1.3 x
2	4558	56.3	5442	1014.2	18.0 x	5029	107.8	53.3 %	1.9 x
3	5261	109.4	6006	1586.5	14.5 x	5771	192.2	68.5 %	1.8 x
4	2415	35.1	2803	430.8	12.3 x	2725	65.2	79.9 %	1.9 x
5	4050	85.8	5313	1588.2	18.5 x	5124	134.7	85.0 %	1.6 x
total	19596	362.4	24166	4992.5	**13.8 x**	23158	600.5	**77.9 %**	**1.7 x**

Recruiting Research Interns

- Programming system design (PL)
- High performance static analysis (PL + Algorithm)
- Data-driven program analysis (PL + ML)
- Software security (PL + Security)

Classes of Problems

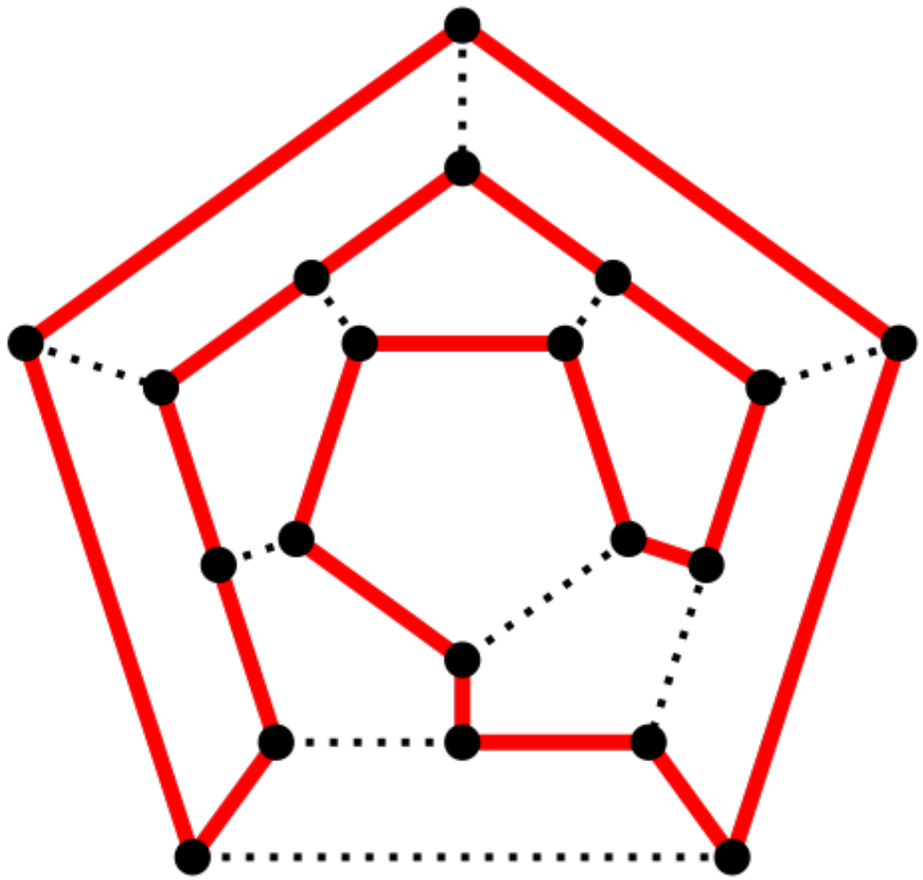


P and NP

- P: solvable in polynomial time.
  - 현실적으로 풀 수 있는 문제
- NP: nondeterministic polynomial time.
  - 운이 좋아야 풀 수 있는 문제.

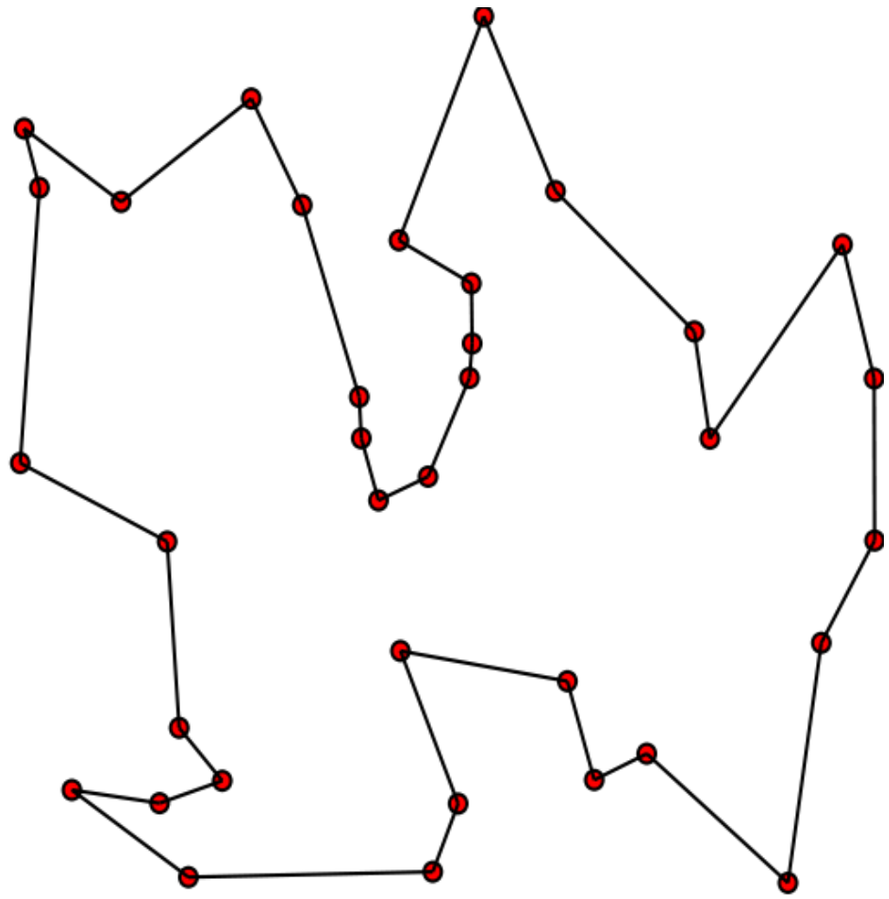
Examples of NP Problems

- finding hamiltonian path



Examples of NP Problems

- traveling salesman problem (TSP)



Examples of NP Problems

- boolean satisfiability problem

$$(x \text{ OR } y \text{ OR } z) \text{ AND } (x \text{ OR } \bar{y} \text{ OR } z) \text{ AND}$$

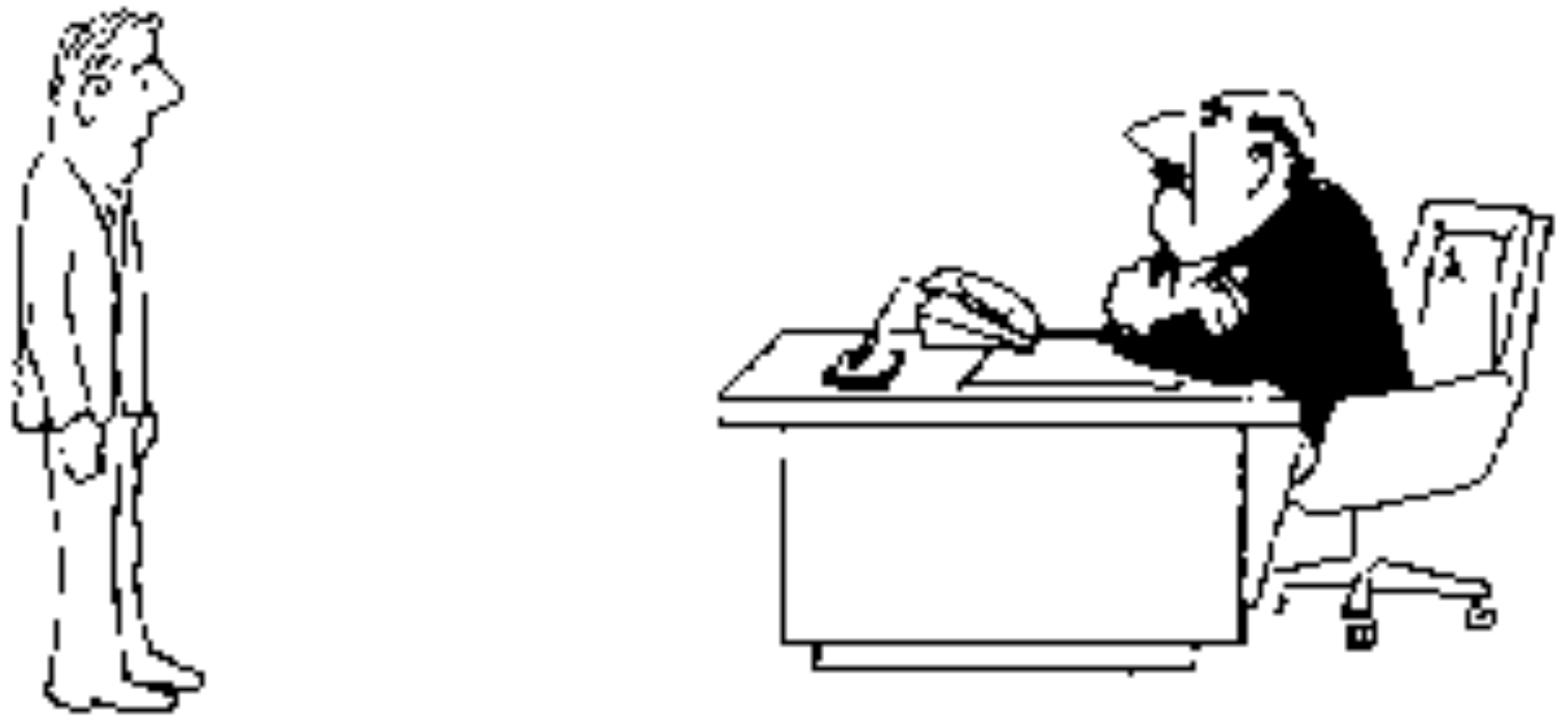
$$(x \text{ OR } y \text{ OR } \bar{z}) \text{ AND } (x \text{ OR } \bar{y} \text{ OR } \bar{z}) \text{ AND}$$

$$(\bar{x} \text{ OR } y \text{ OR } z) \text{ AND } (\bar{x} \text{ OR } \bar{y} \text{ OR } \bar{z})$$

NP-complete Problems

- Most difficult problems in the NP class
- NP-complete problems are logically related
  - If one NP-complete problem is solvable in polynomial time, all the others are solvable in polynomial

Usefulness of NP-completeness

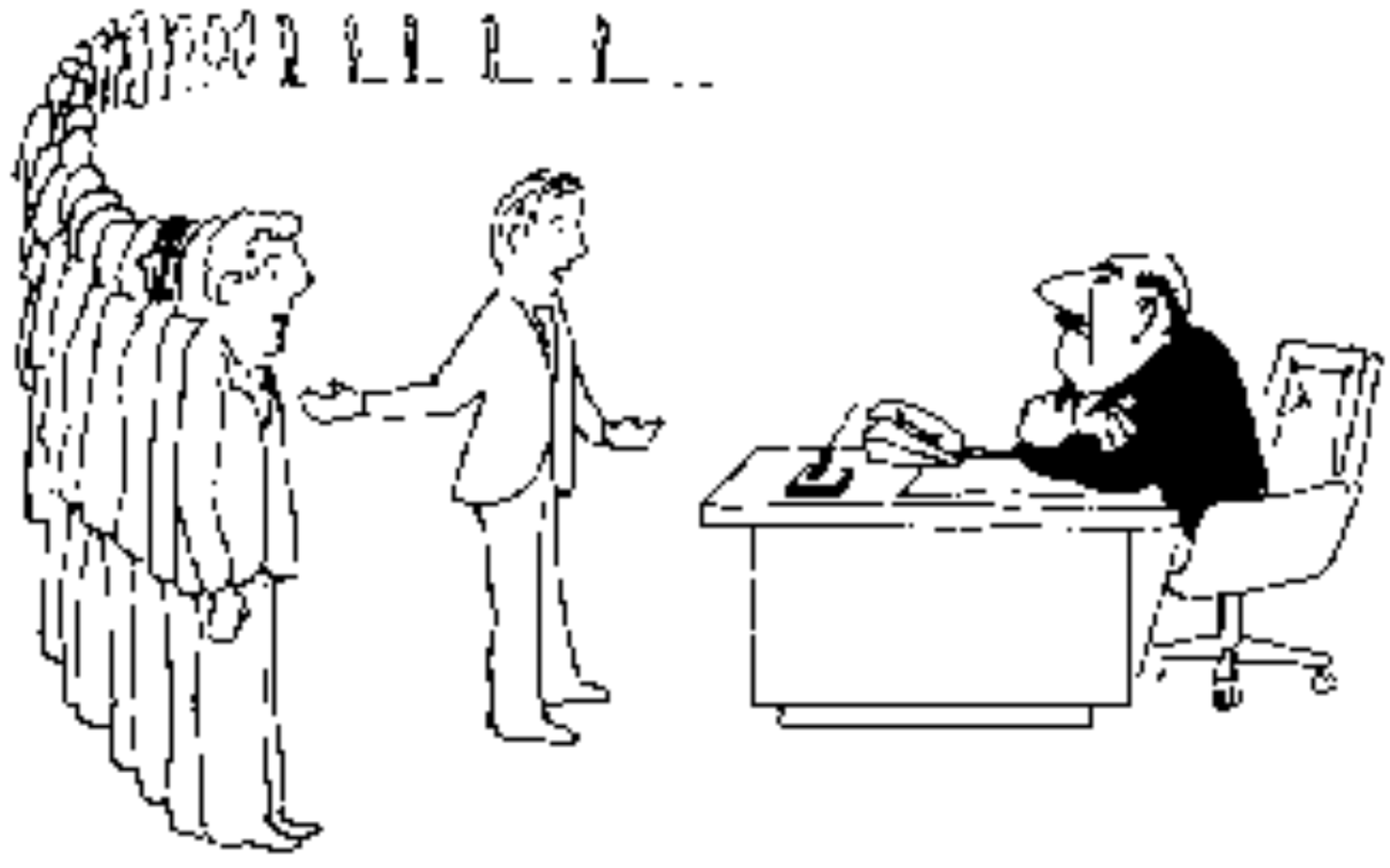


I can't find an efficient algorithm, I guess I'm just too dumb.

Usefulness of NP-completeness

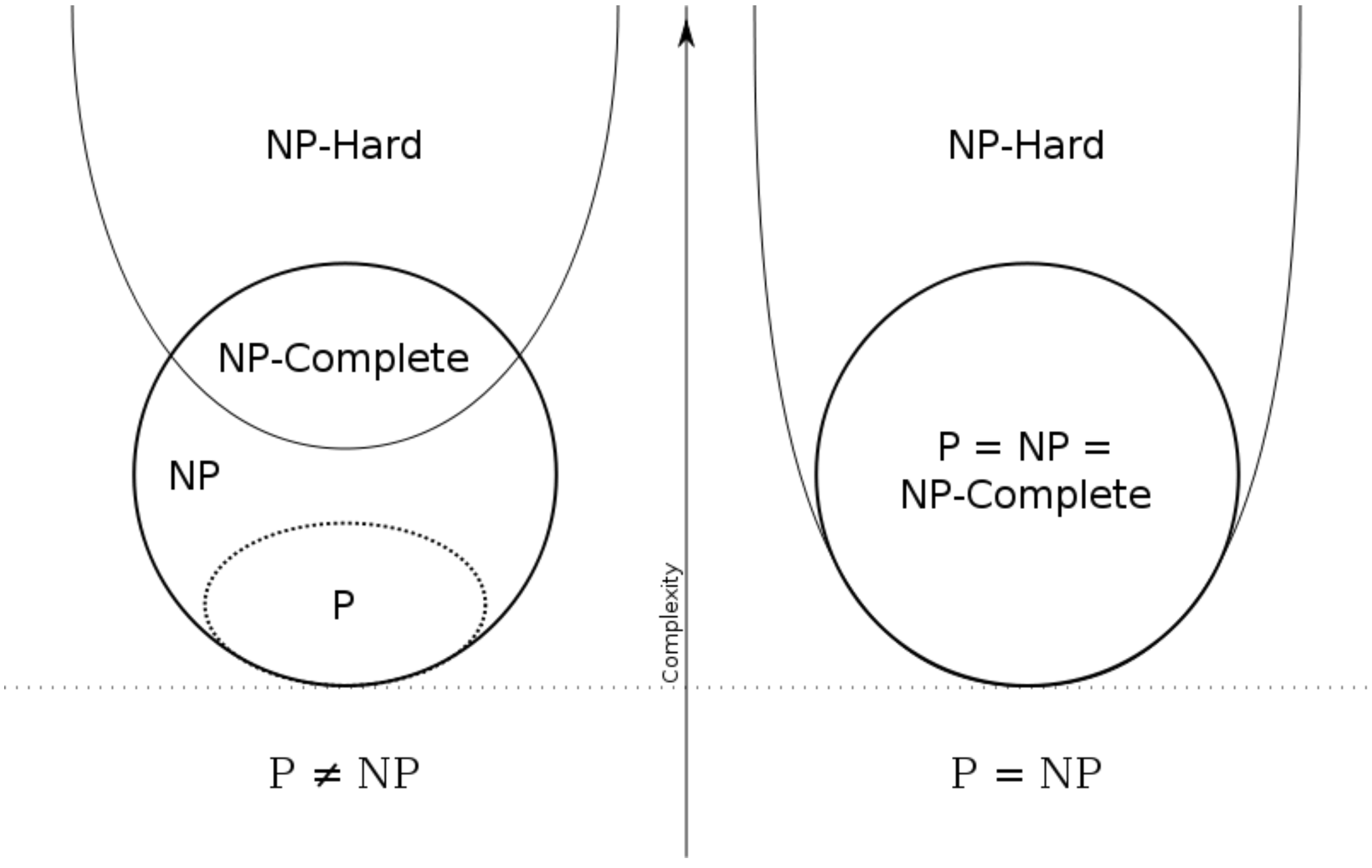
I can't find an efficient algorithm, because no such algorithm is possible

Usefulness of NP-completeness



I can't find an efficient algorithm, but neither can all these famous people.

P=NP?



$P \neq NP$

$P = NP$

Summary

- Undecidable problems can be solved in a useful way via approximation

- P / NP / NP-complete