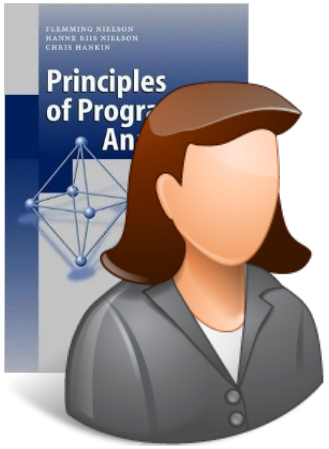
A User-Guided Approach to Program Analysis

Xin Zhang
Georgia Tech & University of Pennsylvania

Joint work with:
Ravi Mangal, Aditya Nori, and Mayur Naik

Motivation

Motivation



Analysis Writer

Approximations

Computing exact solutions impossible

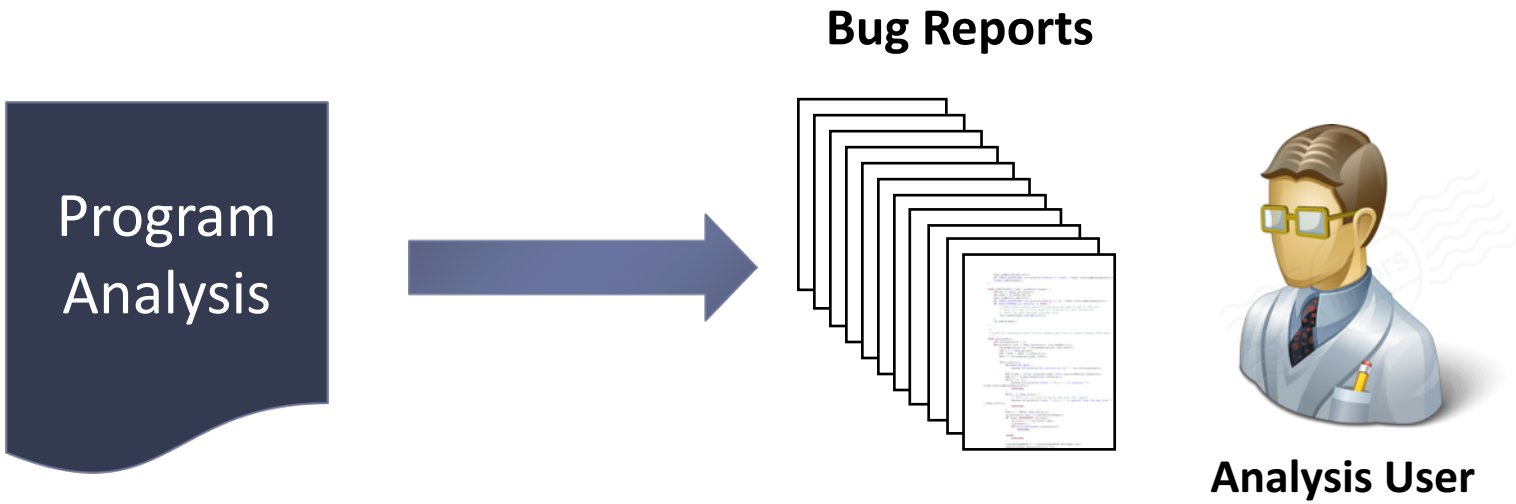
Imprecisely defined properties

Missing program parts

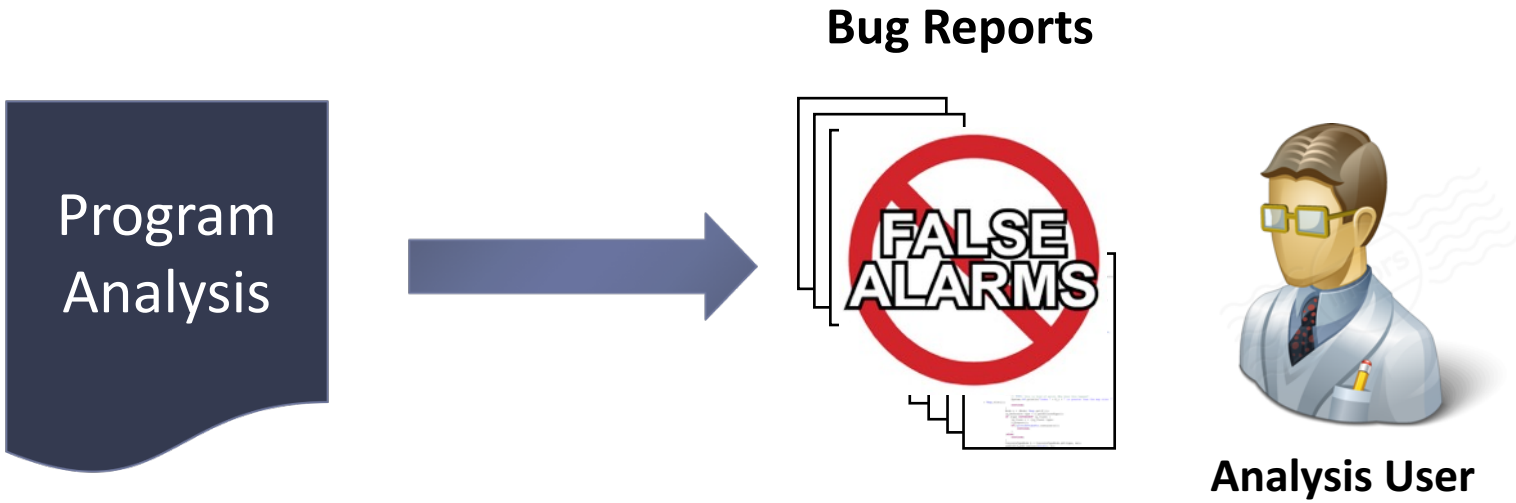
…

Program Analysis

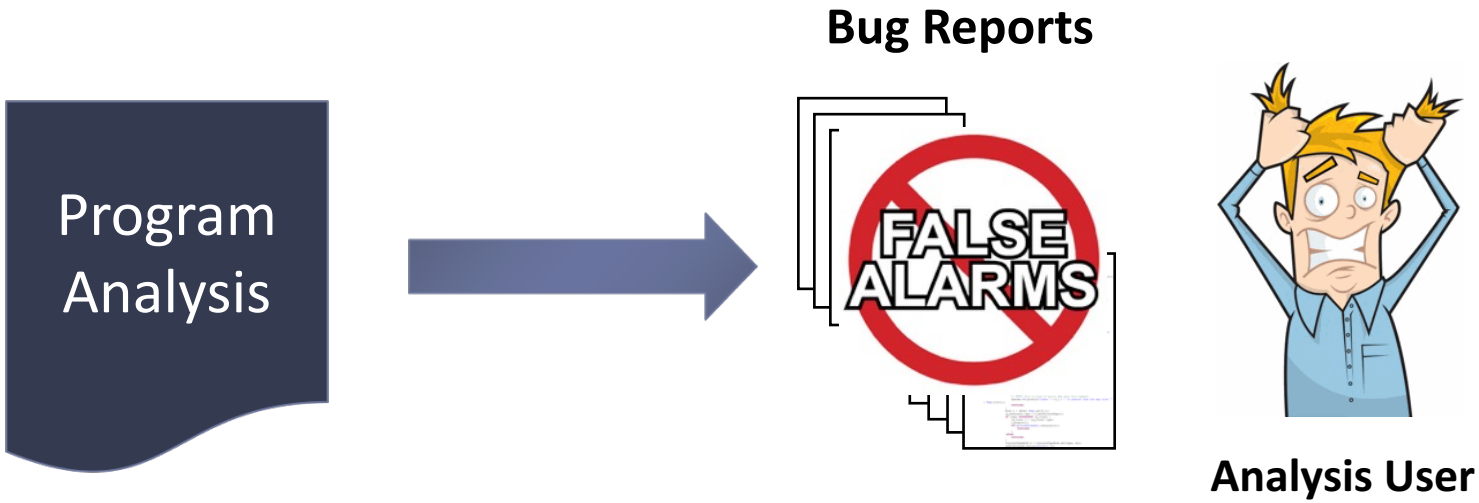
Motivation



Motivation



Motivation



"People ignore the tool if more than 30% false positives are reported …"
[Coverity, CACM'10]

Our Key Idea

Shift decisions about **usefulness** of results from **analysis writers** to **analysis users**

Example: Static Datarace Detection

Code snippet from Apache FTP Server

```
1  public class RequestHandler {
2      FtpRequestImpl request;
3      FtpWriter writer;
4      BufferedReader reader;
5      Socket controlSocket;
6      boolean isConnectionClosed;
7      …

8   public void getRequest( ) {
9          return request;      // x0
10 }
```

```
11  public void close( ) {
12        synchronized (this) {
13            if (isConnectionClosed)
14                return;
15            isConnectionClosed = true;
16        }
17        request.clear();      // x1
18        request = null;       // x2
19        writer.close();       // y1
20        writer = null;        // y2
21        reader.close();
22        reader = null;
23        controlSocket.close();
24        controlSocket = null;
25  }
```

R1

R2

R3

R4

R5

Before User Feedback

Detected Races	
R1: Race on field org.apache.ftpserver.RequestHandler.m_request	
org.apache.ftpserver.RequestHandler: 9	org.apache.ftpserver.RequestHandler: 18
R2: Race on field org.apache.ftpserver.RequestHandler.m_request	
org.apache.ftpserver.RequestHandler: 17	org.apache.ftpserver.RequestHandler: 18
R3: Race on field org.apache.ftpserver.RequestHandler.m_writer	
org.apache.ftpserver.RequestHandler: 19	org.apache.ftpserver.RequestHandler: 20
R4: Race on field org.apache.ftpserver.RequestHandler.m_reader	
org.apache.ftpserver.RequestHandler: 21	org.apache.ftpserver.RequestHandler: 22
R5: Race on field org.apache.ftpserver.RequestHandler.m_controlSocket	
org.apache.ftpserver.RequestHandler: 23	org.apache.ftpserver.RequestHandler: 24

Eliminated Races	
E1: Race on field org.apache.ftpserver.RequestHandler. m_isConnectionClosed	
org.apache.ftpserver.RequestHandler: 13	org.apache.ftpserver.RequestHandler: 15

Before User Feedback

Detected Races	
R1: Race on field org.apache.ftpserver.RequestHandler.m_request	
org.apache.ftpserver.RequestHandler: 9	org.apache.ftpserver.RequestHandler: 18
R2: Race on field org.apache.ftpserver.RequestHandler.m_request	
org.apache.ftpserver.RequestHandler: 17	org.apache.ftpserver.RequestHandler: 18
R3: Race on field org.apache.ftpserver.RequestHandler.m_writer	
org.apache.ftpserver.RequestHandler: 19	org.apache.ftpserver.RequestHandler: 20
R4: Race on field org.apache.ftpserver.RequestHandler.m_reader	
org.apache.ftpserver.RequestHandler: 21	org.apache.ftpserver.RequestHandler: 22
R5: Race on field org.apache.ftpserver.RequestHandler.m_controlSocket	
org.apache.ftpserver.RequestHandler: 23	org.apache.ftpserver.RequestHandler: 24

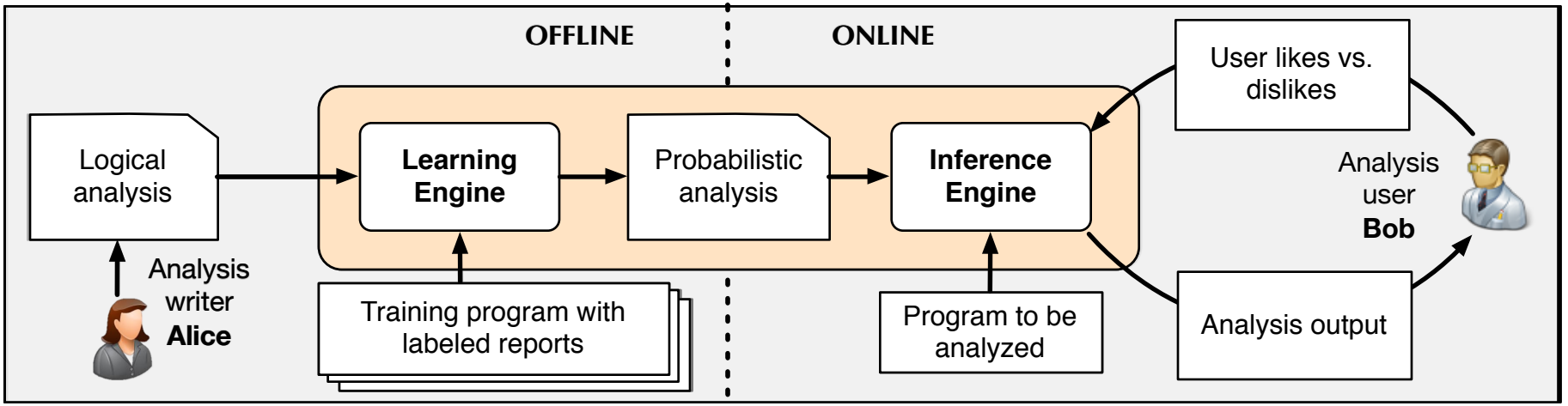
Eliminated Races	
E1: Race on field org.apache.ftpserver.RequestHandler. m_isConnectionClosed	
org.apache.ftpserver.RequestHandler: 13	org.apache.ftpserver.RequestHandler: 15

After User Feedback

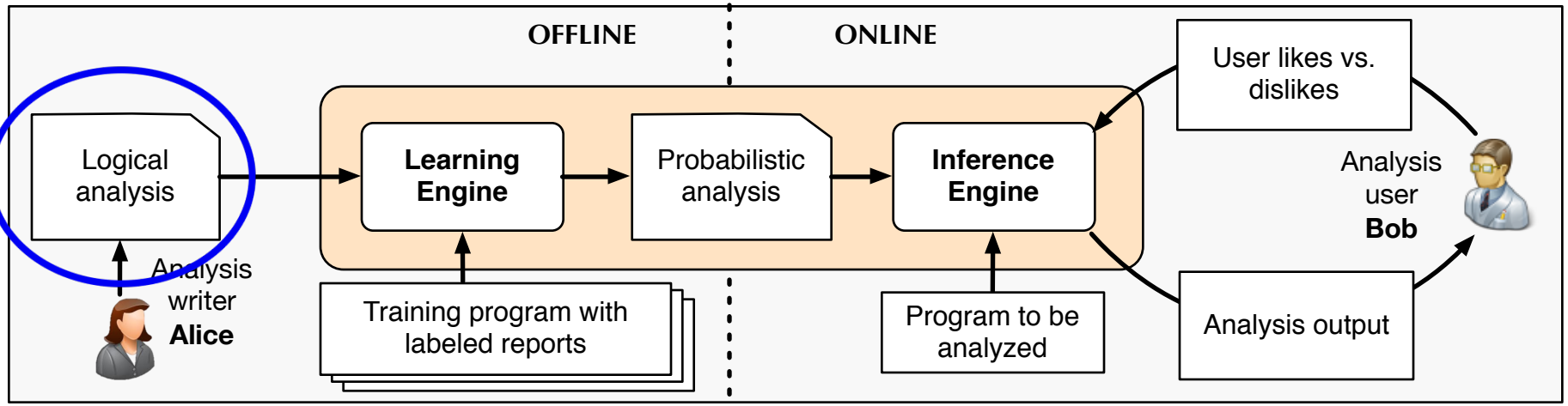
Detected Races	
R1: Race on field org.apache.ftpserver.RequestHandler.m_request 	
org.apache.ftpserver.RequestHandler: 9	org.apache.ftpserver.RequestHandler: 18

Eliminated Races	
E1: Race on field org.apache.ftpserver.RequestHandler. m_isConnectionClosed	
org.apache.ftpserver.RequestHandler: 13	org.apache.ftpserver.RequestHandler: 15
E2: Race on field org.apache.ftpserver.RequestHandler.m_request	
org.apache.ftpserver.RequestHandler: 17	org.apache.ftpserver.RequestHandler: 18
E3: Race on field org.apache.ftpserver.RequestHandler.m_writer	
org.apache.ftpserver.RequestHandler: 19	org.apache.ftpserver.RequestHandler: 20
E4: Race on field org.apache.ftpserver.RequestHandler.m_reader	
org.apache.ftpserver.RequestHandler: 21	org.apache.ftpserver.RequestHandler: 22
E5: Race on field org.apache.ftpserver.RequestHandler.m_controlSocket	
org.apache.ftpserver.RequestHandler: 23	org.apache.ftpserver.RequestHandler: 24

Our System For User-Guided Analysis



Logical Analysis



Logical Datarace Analysis Using Datalog

Input relations:

next(p1, p2), mayAlias(p1, p2), guarded(p1, p2)

Output relations:

parallel(p1, p2), race(p1, p2)

Rules:

parallel(p3, p2) :- parallel(p1, p2), next (p3, p1).

parallel(p1, p2) :- parallel(p2, p1).

race(p1, p2) :- parallel(p1, p2), mayAlias(p1, p2), ¬guarded(p1, p2).

Logical Datarace Analysis Using Datalog

Input relations:

next(p1, p2), mayAlias(p1, p2), guarded(p1, p2)

Output relations:

race(p1, p2)

p1 is immediate successor of p2.

p1 & p2 may access the same memory location.

p1 & p2 are guarded by the same lock.

Rules:

parallel(p3, p2) :- parallel(p1, p2), next (p3, p1).

parallel(p1, p2) :- parallel(p2, p1).

race(p1, p2) :- parallel(p1, p2), mayAlias(p1, p2), ¬guarded(p1, p2).

Logical Datarace Analysis Using Datalog

Input relations:

next(p1, p2), mayAlias(p1, p2), guarded(p1, p2)

Output relations:

parallel(p1, p2), race(p1, p2)

Rules:

para (p1

para (p2

p1 & p2 may happen in parallel.

p1 & p2 may have a datarace.

race(p1, p2) :- parallel(p1, p2), mayAlias(p1, p2), ¬guarded(p1, p2).

Logical Datarace Analysis Using Datalog

Input relations:

next(p1, p2), mayAlias(p1, p2), guarded(p1, p2)

Output relations:

parallel(p1, p2), race

Rules:

parallel(p3, p2) :-

parallel(p1, p2) :- paralle

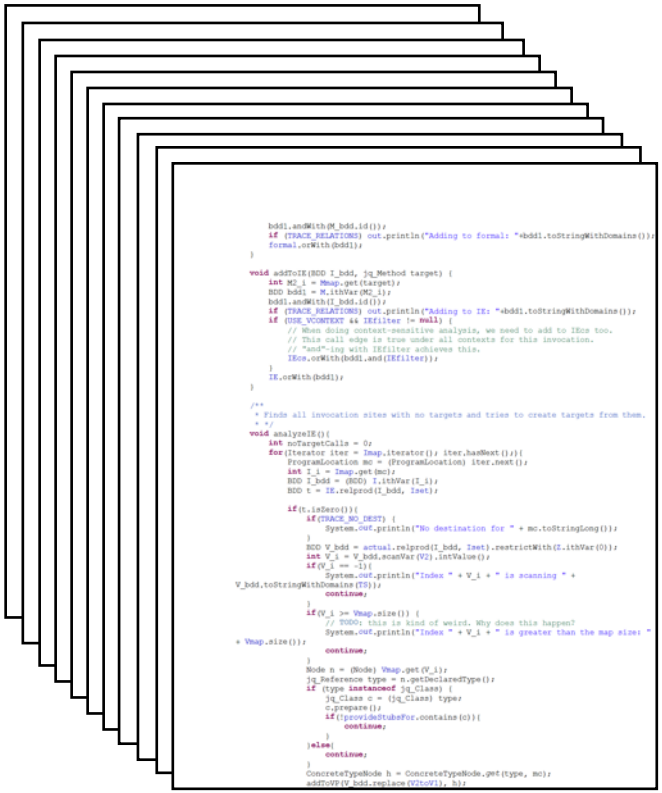
race(p1, p2) :- parallel(p1, p2), mayAlias(p1, p2), ¬guarded(p1, p2).

If p1 & p2 may happen in parallel,
and they may access the same memory location,
and they are not guarded by the same lock,
then p1 & p2 may have a datarace.

Why Datalog?

- Easier to specify

Analysis in Java



VS.

Analysis in Datalog

```
### Context-sensitive inclusion-based pointer analysis using cloning
#
# Calculates the numbering based on the call graph relation.
#
# Author: John Whaley

.include "fielddomains.pa"
.bddnodes 10000000
.bddcache 1000000
.bddvarorder N0_F0_I0_M1_M0_V1_V0_VC1xVC0_T0_Z0_T1_H0_H1

mI0(m,i) :- mI(m,i,_).
IEnum(i,m,vc2,vc1) :- roots(m), mI0(m,i), IE0(i,m). number

cvP(_,v,h) :- vP0(v,h).
cA(_,v1,_,v2) :- A(v1,v2).
IEcs(vc2,i,vc1,m) :- IE0(i,m), IEnum(i,m,vc2,vc1).
vPfilter(v,h) :- vT(v,tv), aT(tv,th), hT(h,th).
cA(vc1,v1,vc2,v2) :- formal(m,z,v1), IEcs(vc2,i,vc1,m), actual(i,z,v2).
cA(vc2,v2,vc1,v1) :- Mret(m,v1), IEcs(vc2,i,vc1,m), Iret(i,v2).
cA(vc2,v2,vc1,v1) :- Mthr(m,v1), IEcs(vc2,i,vc1,m), Ithr(i,v2).

cvP(vc1,v1,h) :- cA(vc1,v1,vc2,v2), cvP(vc2,v2,h), vPfilter(v1,h).
hP(h1,f,h2) :- S(v1,f,v2), cvP(vc1,v1,h1), cvP(vc1,v2,h2).
cvP(vc1,v2,h2) :- L(v1,f,v2), cvP(vc1,v1,h1), hP(h1,f,h2), vPfilter(v2,h2). split

IE(i,m) :- IEcs(_,i,_,m).
```

Why Datalog?

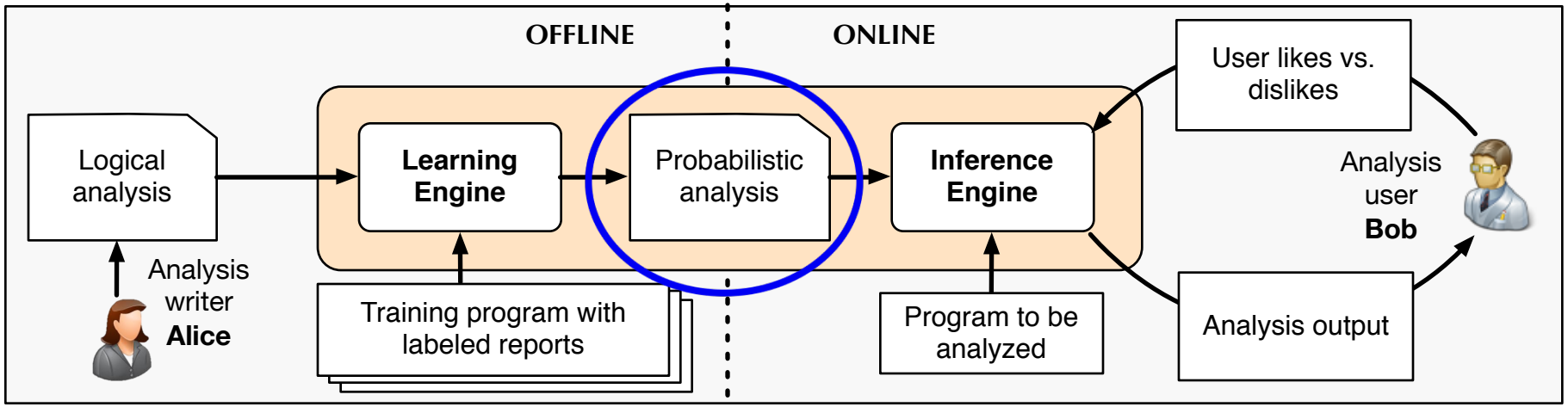
- Easier to specify
- Leverage efficient solvers

Paddle

- Widely adaptable

Soot

Probabilistic Analysis



Datarace Analysis: Logical ➔ Probabilistic

Input relations:

next(p1, p2), mayAlias(p1, p2), guarded(p1, p2)

Output relations:

parallel(p1, p2), race(p1, p2)

Rules:

parallel(p3, p2) :- parallel(p1, p2), next (p3, p1). **weight 5**

parallel(p1, p2) :- parallel(p2, p1).

race(p1, p2) :- parallel(p1, p2), mayAlias(p1, p2), ¬guarded(p1, p2).

¬race(x2, x1). **weight 25**

Datarace Analysis: Logical ➔ Probabilistic

Input relations:

next(p1, p2), mayAlias(p1, p2), guarded(p1, p2)

Output relations:

parallel(p1, p2), race(p1, p2)

Rules:

parallel(p3, p2) :- p ext (p3, p1). **weight 5**

"Soft" Rule

"Hard" Rule

parallel(p1, p2) :- parallel(p2, p1).

race(p1, p2) :- parallel(p1, p2), mayAlias(p1, p2), ¬guarded(p1, p2).

¬race(x2, x1). **weight 25**

A Semantics for Probabilistic Analysis

- Probabilistic Analysis => Markov Logic Network (MLN)

  [Richardson & Domingos, Machine Learning'06]

- MLN defines a probability distribution over all possible analysis outputs

- Probability of an output *x*:

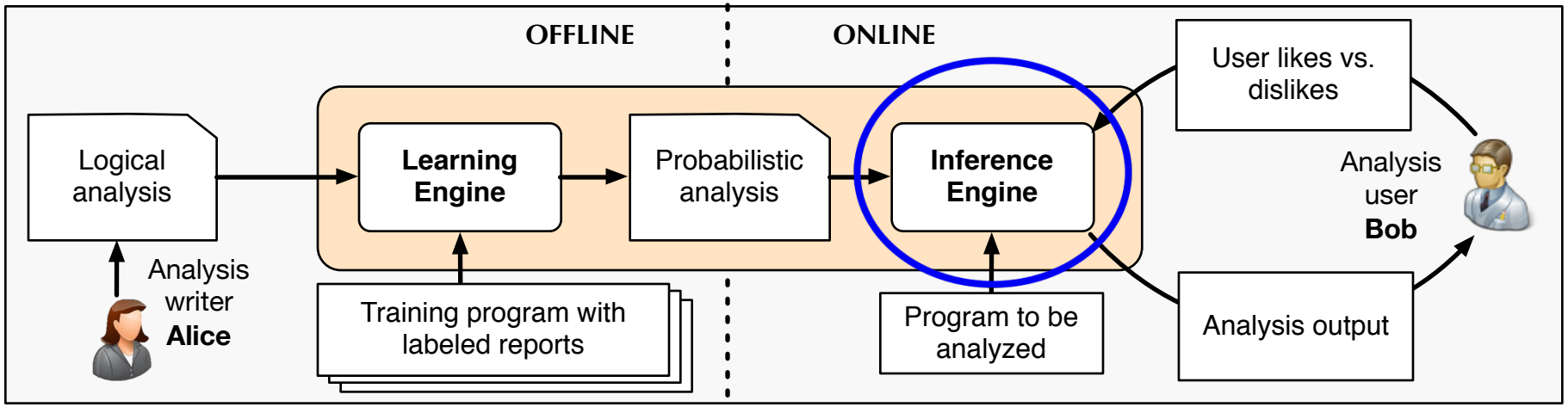
$$P(X=x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right)$$

Normalization factor

Weight of rule *i*

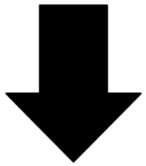
Number of true instances of rule *i* in *x*

Inference Engine



Probabilistic Inference

Find the most likely output given the input program



$$\arg\max_x P(x) = \arg\max_x \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right)$$
$$= \arg\max_x \sum_i w_i n_i(x)$$

What is MaxSAT?

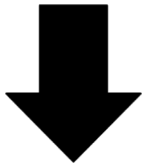
```
¬ b1 ∨ ¬ b2 ∨ b3   weight 5   ∧
  b3 ∨   b4        weight 10  ∧
¬ b4 ∨ ¬ b2        weight 7   ∧
...
```

Find a boolean assignment such that the sum of the weights of the satisfied clauses is maximized

$$\arg\max_{x} \sum_{i} w_i n_i(x)$$

Probabilistic Inference ➔ MaxSAT

Find the most likely output given the input program



$$\arg\max_x P(x) = \arg\max_x \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right)$$
$$= \arg\max_x \sum_i w_i n_i(x)$$

Solve the MaxSAT instance entailed by the MLN

Example: Static Datarace Detection

Code snippet from Apache FTP Server

```
1  public class RequestHandler {
2      FtpRequestImpl request;
3      FtpWriter writer;
4      BufferedReader reader;
5      Socket controlSocket;
6      boolean isConnectionClosed;
7      …

8   public void getRequest( ) {
9          return request;      // x0
10 }
```

```
11  public void close( ) {
12         synchronized (this) {
13             if (isConnectionClosed)
14                 return;
15             isConnectionClosed = true;
16         }
17         request.clear();     // x1
18         request = null;      // x2
19         writer.close();      // y1
20         writer = null;       // y2
21         reader.close();
22         reader = null;
23         controlSocket.close();
24         controlSocket = null;
25  }
```

R1

R2

R3

R4

R5

How Does Online Phase Work?

Input facts:

mayAlias(x2, x1), ¬guarded(x2, x1), next(x2, x1),
mayAlias(y2, y1), ¬guarded(y2, y1), next(y1, x2),
next(y2, x1)

Output facts:

parallel(x2, x0), race(x2, x0),
parallel(x2, x1), race(x2, x1),
parallel(y2, y1), race(y2, y1)

MaxSAT formula:

(parallel(x1, x1) ∧ next(x2, x1) => parallel(x2, x1)) **weight 5** ∧

(parallel(x2, x1) => parallel(x1, x2)) ∧

(parallel(x1, x2) ∧ next(x2, x1) => parallel(x2, x2)) **weight 5** ∧

(parallel(x2, x2) ∧ next(y1, x2) => parallel(y1, x2)) **weight 5** ∧

(parallel(y1, x2) => parallel(x2, y1)) ∧

(parallel(x2, y1) ∧ next(y1, x2) => parallel(y1, y1)) **weight 5** ∧

(parallel(y1, y1) ∧ next(y2, y1) => parallel(y2, y1)) **weight 5** ∧

(parallel(y2, y1) ∧ mayAlias(y2, y1) ∧ ¬guarded(y2, y1) => race(y2, y1)) ∧

(parallel(x2, x1) ∧ mayAlias(x2, x1) ∧ ¬guarded(x2, x1) => race(x2, x1)) ∧

¬race(x2, x1) **weight 25**

How Does Online Phase Work?

Input facts:

mayAlias(x2, x1), ¬guarded(x2, x1), next(x2, x1),
mayAlias(y2, y1), ¬guarded(y2, y1), next(y1, x2),
next(y2, x1)

Output facts:

parallel(x2, x0), race(x2, x0),
parallel(x2, x1), race(x2, x1),
parallel(y2, y1), race(y2, y1)

MaxSAT formula:

(parallel(x1, x1) ∧ next(x2, x1) => parallel(x2, x1)) **weight 5** ∧ 

(parallel(x2, x1) => parallel(x1, x2)) ∧

(parallel(x1, x2) ∧ next(x2, x1) => parallel(x2, x2)) **weight 5** ∧

(parallel(x2, x2) ∧ next(y1, x2) => parallel(y1, x2)) **weight 5** ∧

(parallel(y1, x2) => parallel(x2, y1)) ∧

(parallel(x2, y1) ∧ next(y1, x2) => parallel(y1, y1)) **weight 5** ∧

(parallel(y1, y1) ∧ next(y2, y1) => parallel(y2, y1)) **weight 5** ∧

(parallel(y2, y1) ∧ mayAlias(y2, y1) ∧ ¬guarded(y2, y1) => race(y2, y1)) ∧

(parallel(x2, x1) ∧ mayAlias(x2, x1) ∧ ¬guarded(x2, x1) => race(x2, x1)) ∧

¬race(x2, x1) **weight 25**

How Does Online Phase Work?

Input facts:

mayAlias(x2, x1), ¬guarded(x2, x1), next(x2, x1),
mayAlias(y2, y1), ¬guarded(y2, y1), next(y1, x2),
next(y2, x1)

Output facts:

parallel(x2, x0), race(x2, x0),
~~parallel(x2, x1),~~ ~~race(x2, x1),~~
parallel(y2, y1), race(y2, y1)

MaxSAT formula:

(parallel(x1, x1) ∧ next(x2, x1) => ~~parallel(x2, x1)~~) **weight 5** ∧

(parallel(x2, x1) => parallel(x1, x2)) ∧

(parallel(x1, x2) ∧ next(x2, x1) => parallel(x2, x2)) **weight 5** ∧

(parallel(x2, x2) ∧ next(y1, x2) => parallel(y1, x2)) **weight 5** ∧

(parallel(y1, x2) => parallel(x2, y1)) ∧

(parallel(x2, y1) ∧ next(y1, x2) => parallel(y1, y1)) **weight 5** ∧

(parallel(y1, y1) ∧ next(y2, y1) => parallel(y2, y1)) **weight 5** ∧

(parallel(y2, y1) ∧ mayAlias(y2, y1) ∧ ¬guarded(y2, y1) => race(y2, y1)) ∧

~~(parallel(x2, x1)~~ ∧ mayAlias(x2, x1) ∧ ¬guarded(x2, x1) => ~~race(x2, x1))~~ ∧

¬race(x2, x1) **weight 25**

How Does Online Phase Work?

Input facts:

mayAlias(x2, x1), ¬guarded(x2, x1), next(x2, x1),
mayAlias(y2, y1), ¬guarded(y2, y1), next(y1, x2),
next(y2, x1)

Output facts:

parallel(x2, x0), race(x2, x0),
~~parallel(x2, x1),~~ ~~race(x2, x1),~~
~~parallel(y2, y1),~~ ~~race(y2, y1)~~

MaxSAT formula:

(parallel(x1, x1) ∧ next(x2, x1) => ~~parallel(x2, x1)~~) **weight 5** ∧

~~(parallel(x2, x1) => parallel(x1, x2))~~ ∧

~~(parallel(x1, x2)~~ ∧ next(x2, x1) => ~~parallel(x2, x2))~~ **weight 5** ∧

~~(parallel(x2, x2)~~ ∧ next(y1, x2) => ~~parallel(y1, x2))~~ **weight 5** ∧

~~(parallel(y1, x2)~~ => ~~parallel(x2, y1))~~ ∧

~~(parallel(x2, y1)~~ ∧ next(y1, x2) => ~~parallel(y1, y1))~~ **weight 5** ∧

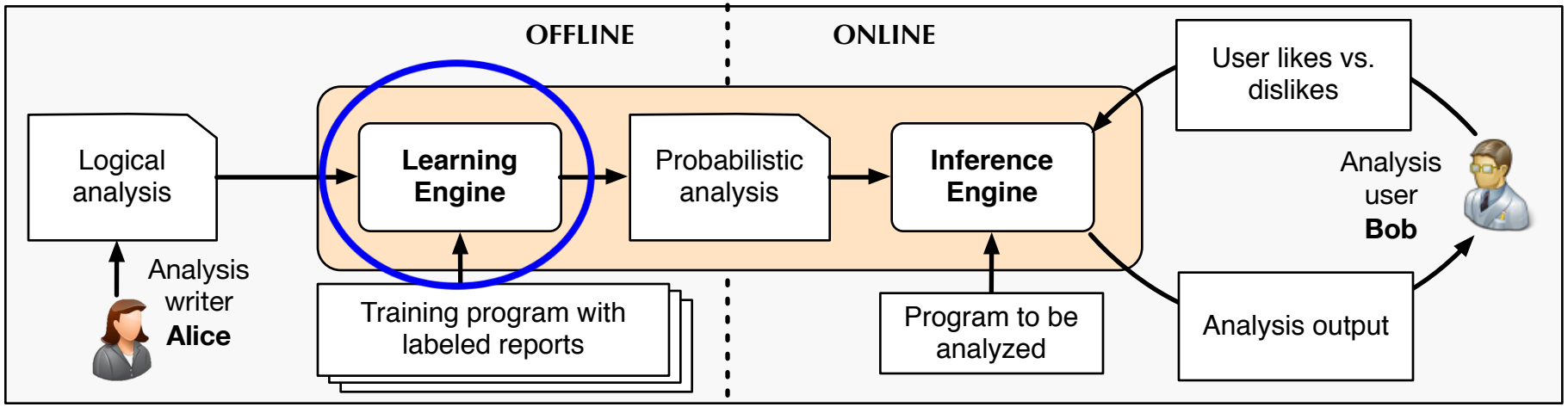
~~(parallel(y1, y1)~~ ∧ next(y2, y1) => ~~parallel(y2, y1))~~ **weight 5** ∧

~~(parallel(y2, y1)~~ ∧ mayAlias(y2, y1) ∧ ¬guarded(y2, y1) => ~~race(y2, y1))~~ ∧

~~(parallel(x2, x1)~~ ∧ mayAlias(x2, x1) ∧ ¬guarded(x2, x1) => ~~race(x2, x1))~~ ∧

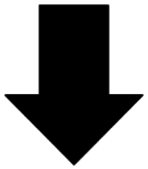
¬race(x2, x1) **weight 25**

Learning Engine



Weight Learning

Learn rule weights such that the probability of the training data is maximized



Perform gradient descent
[Singla & Domingos, AAAI'05]

Empirical Evaluation Questions

- Does user feedback help in improving analysis precision?
- How much feedback is needed and does the amount of feedback affect the precision?
- How feasible is it for users to inspect analysis output and provide useful feedback?

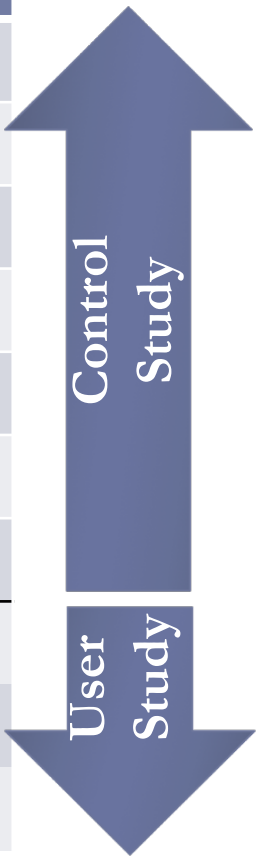
Empirical Evaluation Setup

- Control Study:
  - **Analyses:** (1) Pointer analysis, (2) Datarace analysis
  - **Benchmarks:** 7 Java programs (130-200 KLOC each)
  - **Feedback:** Automated [Zhang et.al, PLDI'14]

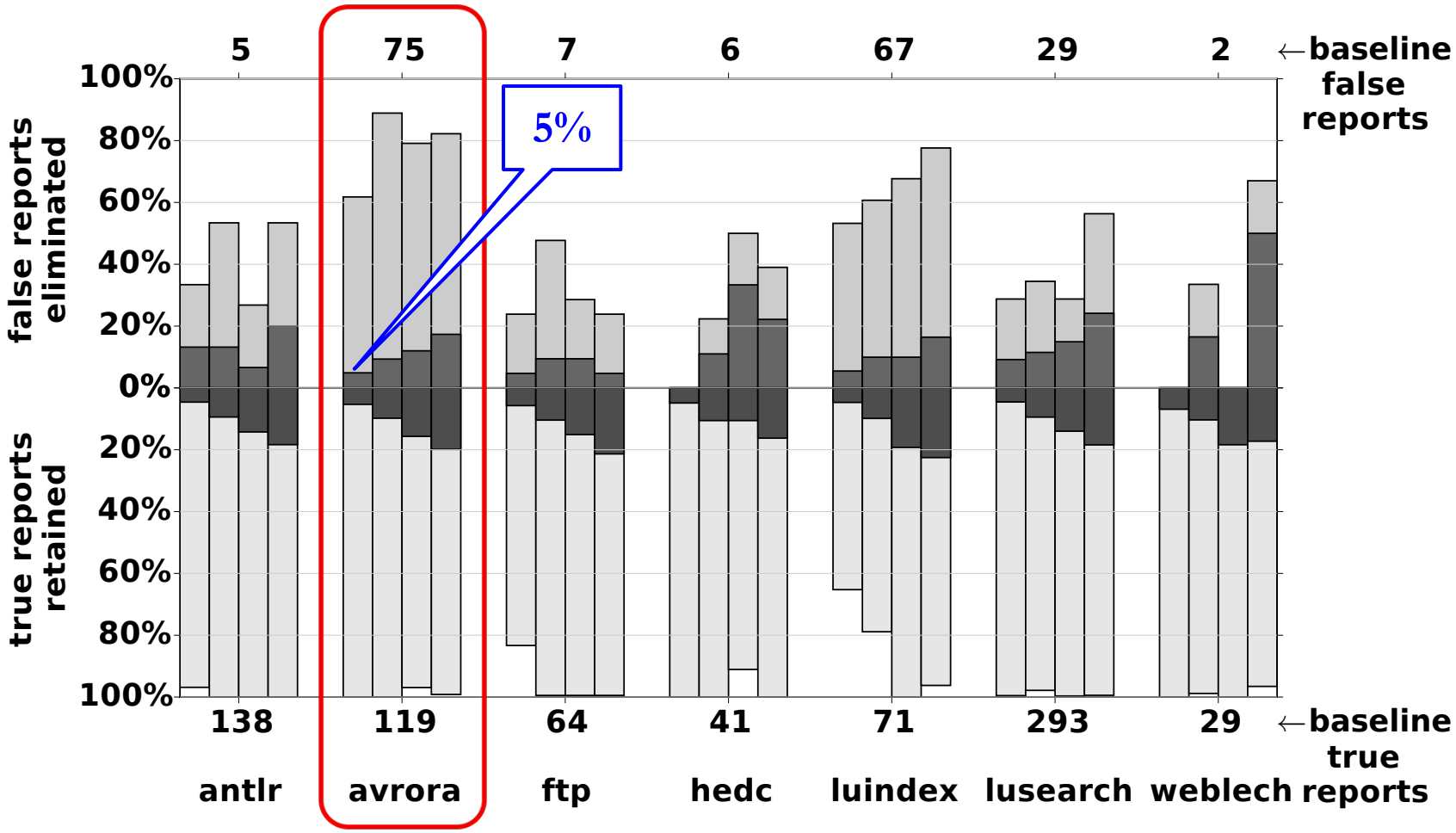
- User Study:
  - **Analyses:** Information flow analysis
  - **Benchmarks:** 3 security micro-benchmarks
  - **Feedback:** 9 users

Benchmarks Characteristics

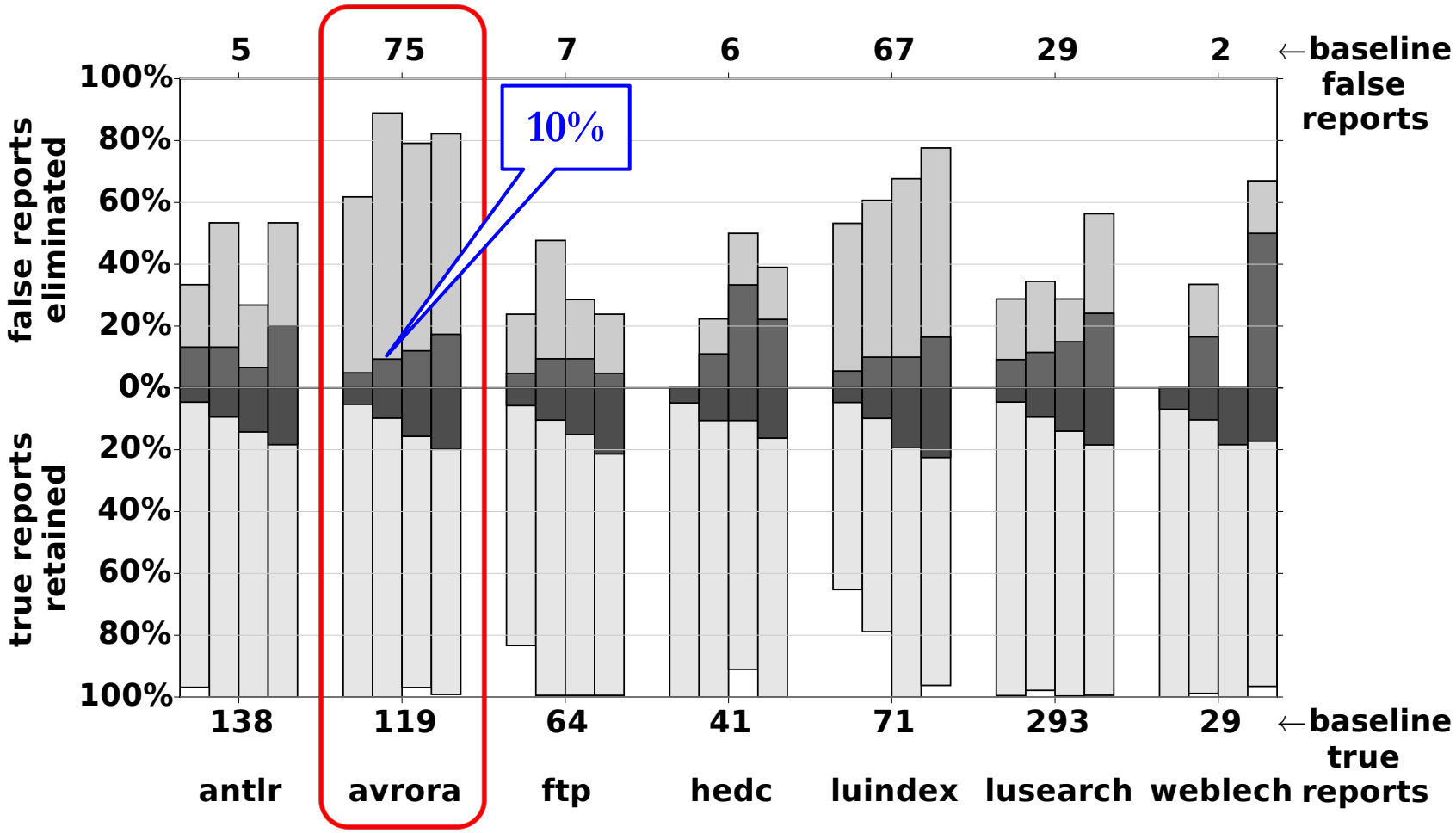
	classes	methods	bytecode(KB)	KLOC
antlr	350	2.3K	186	131
avrora	1,544	6.2K	325	193
ftp	414	2.2K	118	130
hedc	353	2.1K	140	153
luindex	619	3.7K	235	190
lusearch	640	3.9K	250	198
weblech	576	3.3K	208	194
secbench1	5	13	0.3	0.6
secbench2	4	12	0.2	0.6
secbench3	17	46	1.3	4.2



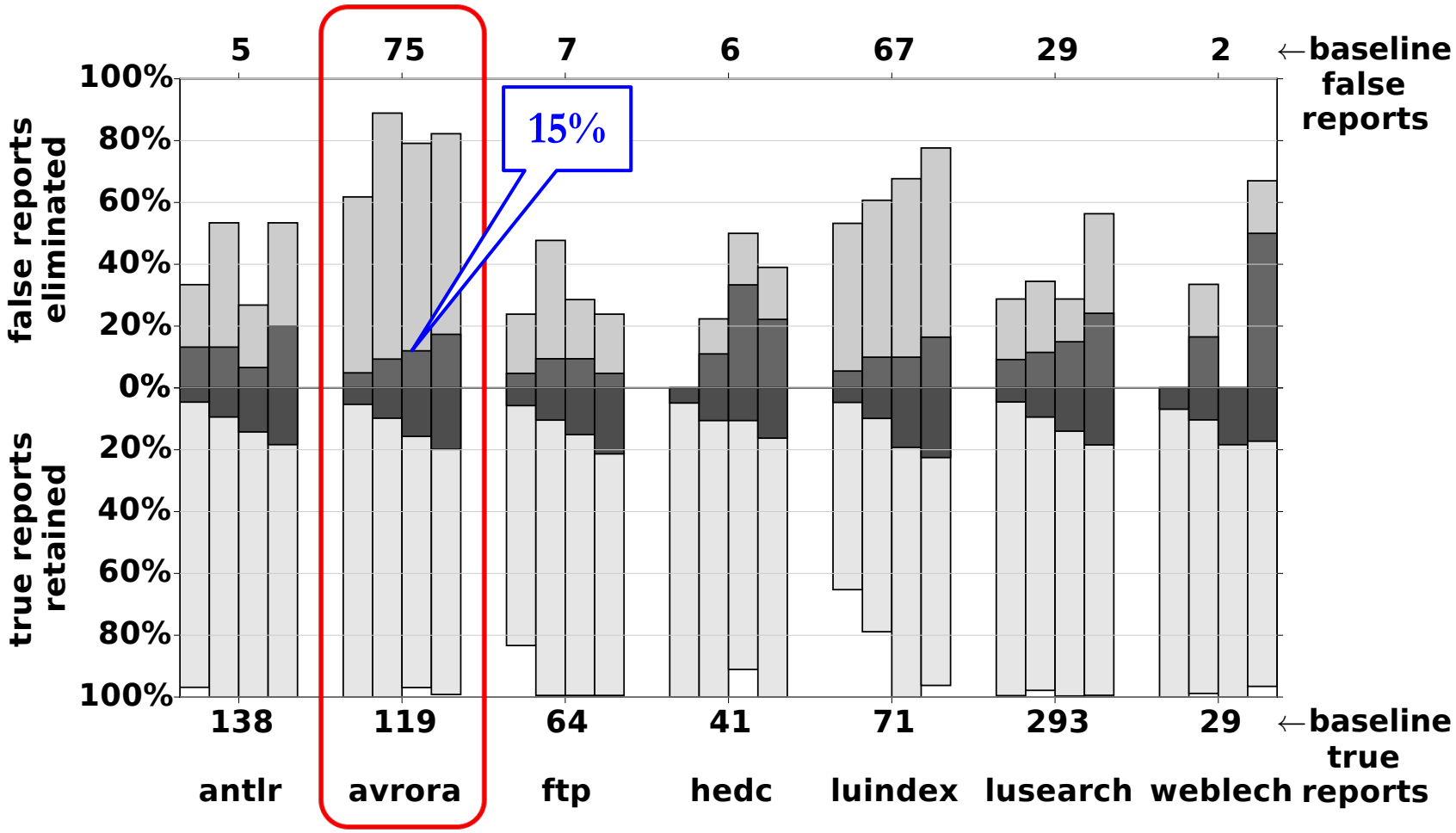
Precision Results: Pointer Analysis



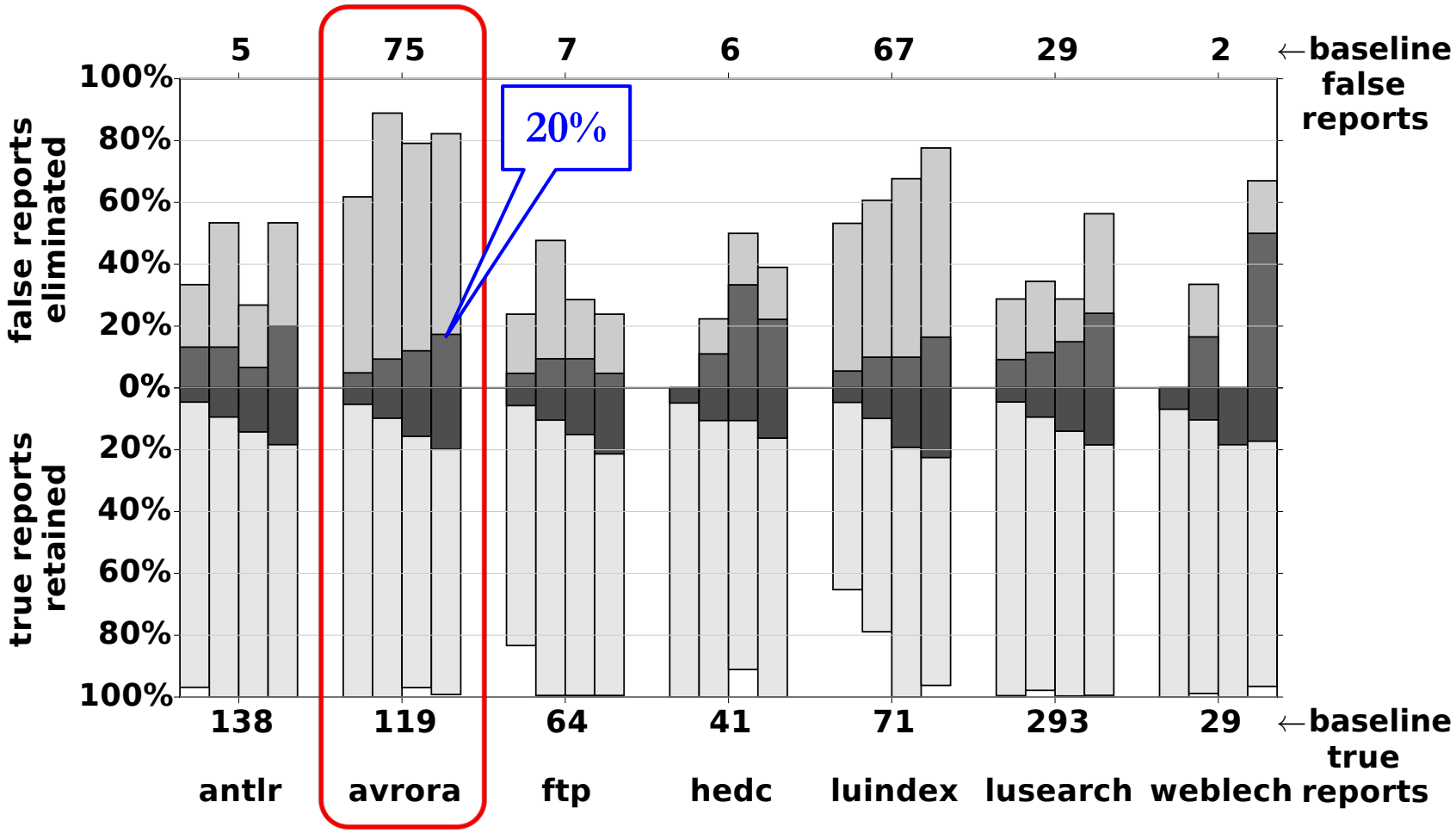
Precision Results: Pointer Analysis



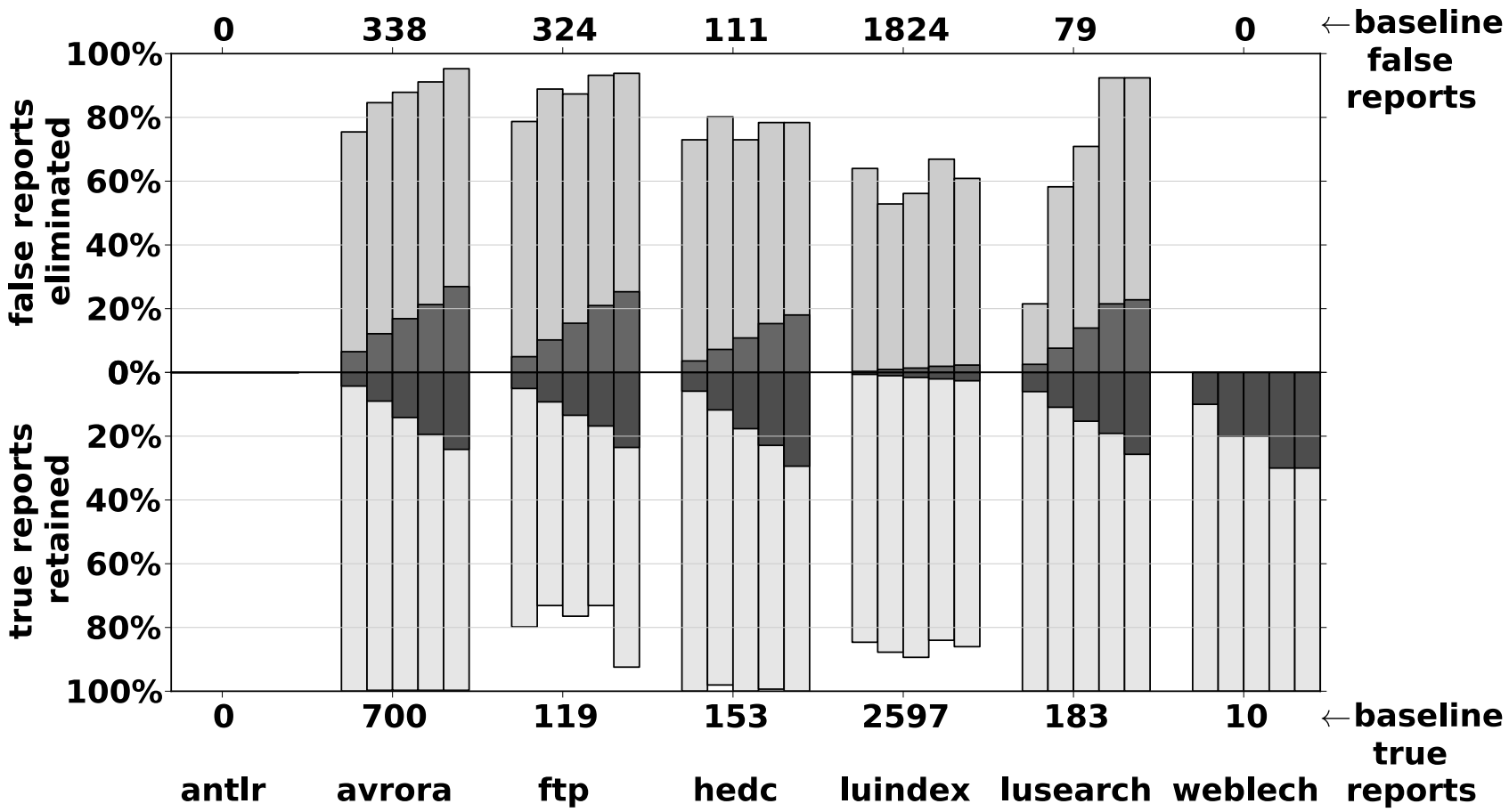
Precision Results: Pointer Analysis



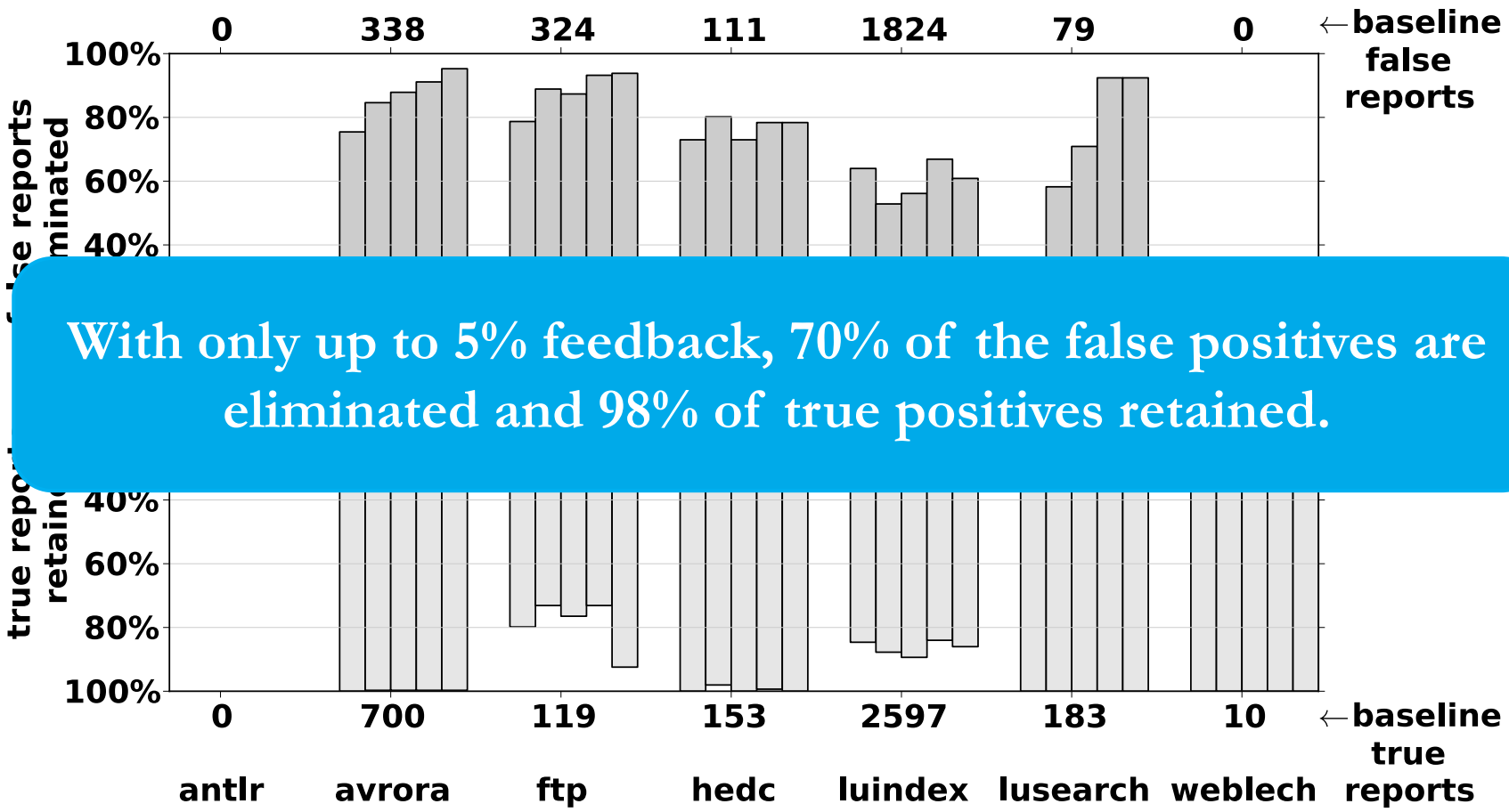
Precision Results: Pointer Analysis



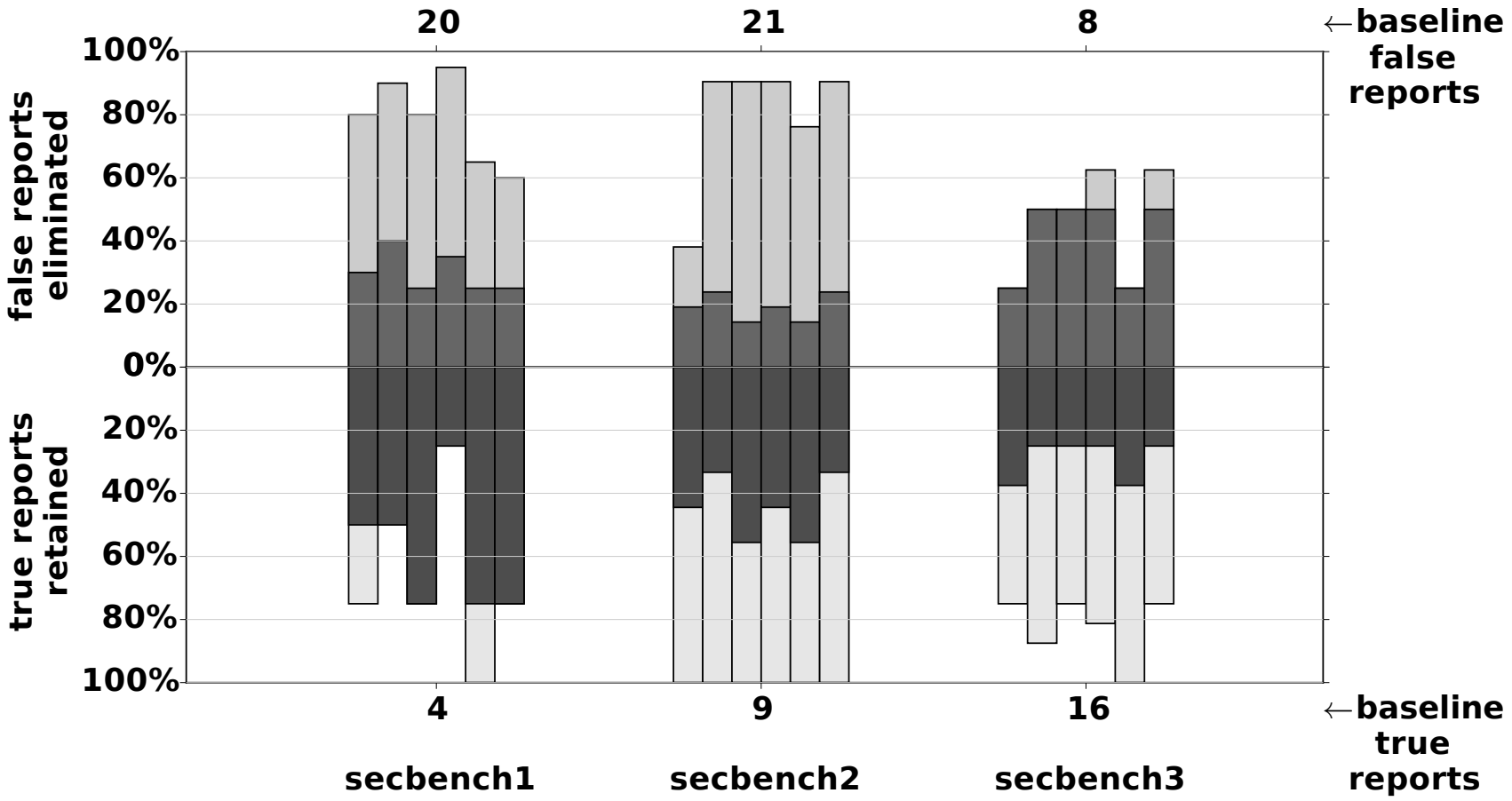
Precision Results: Datarace Analysis



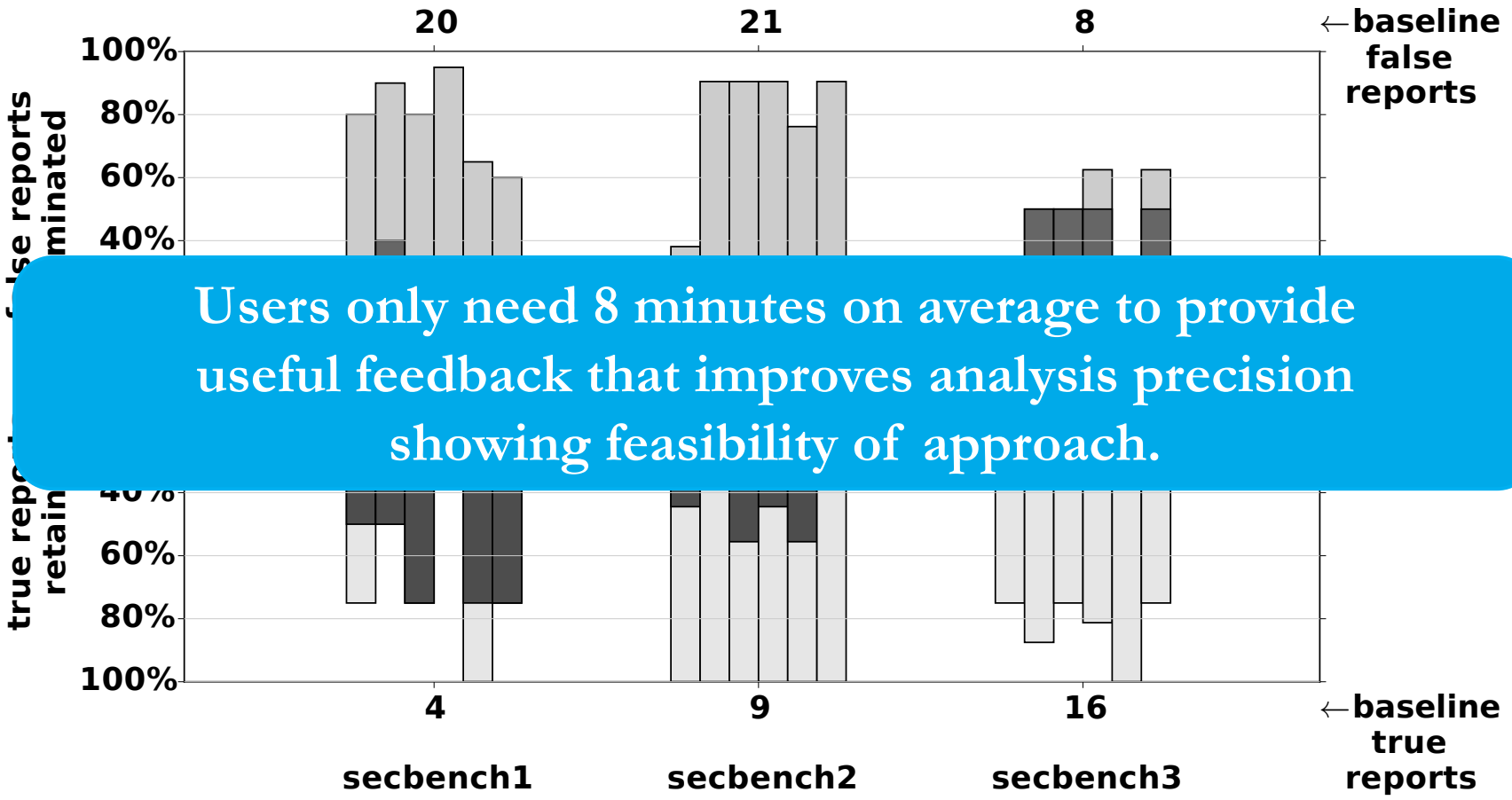
Precision Results: Datarace Analysis



Precision Results: User Study

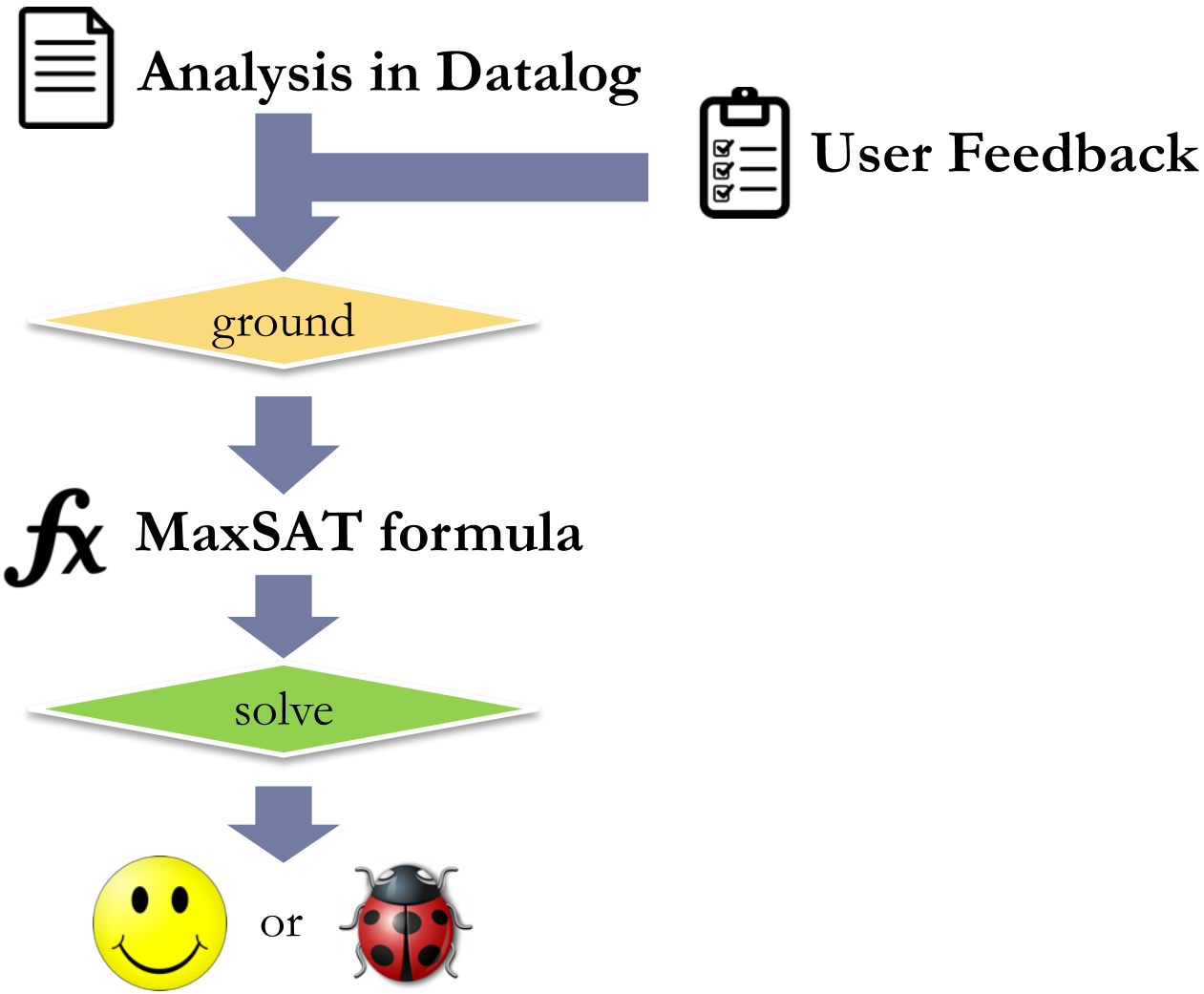


Precision Results: User Study

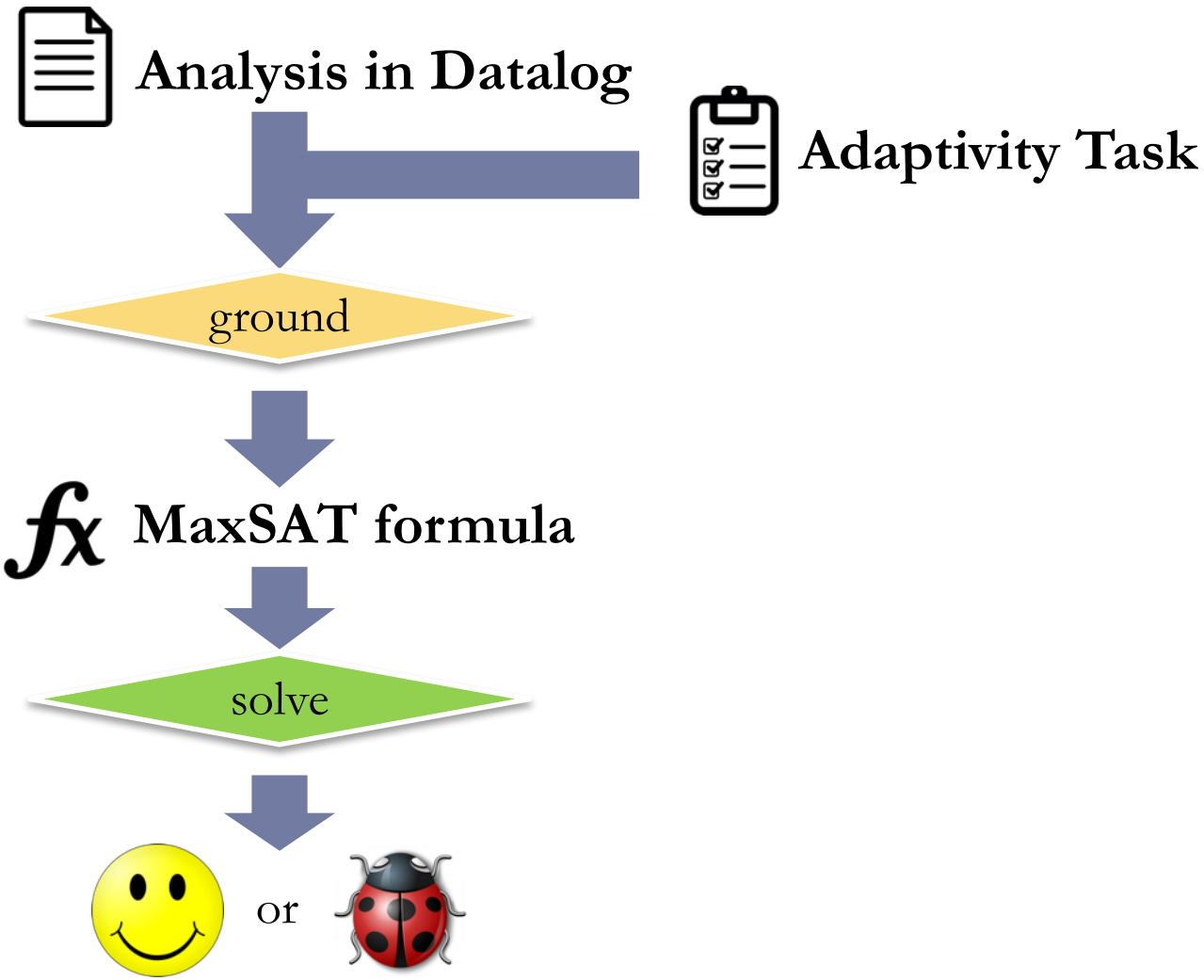


Users only need 8 minutes on average to provide useful feedback that improves analysis precision showing feasibility of approach.

The Story So Far



A General Adaptivity Framework



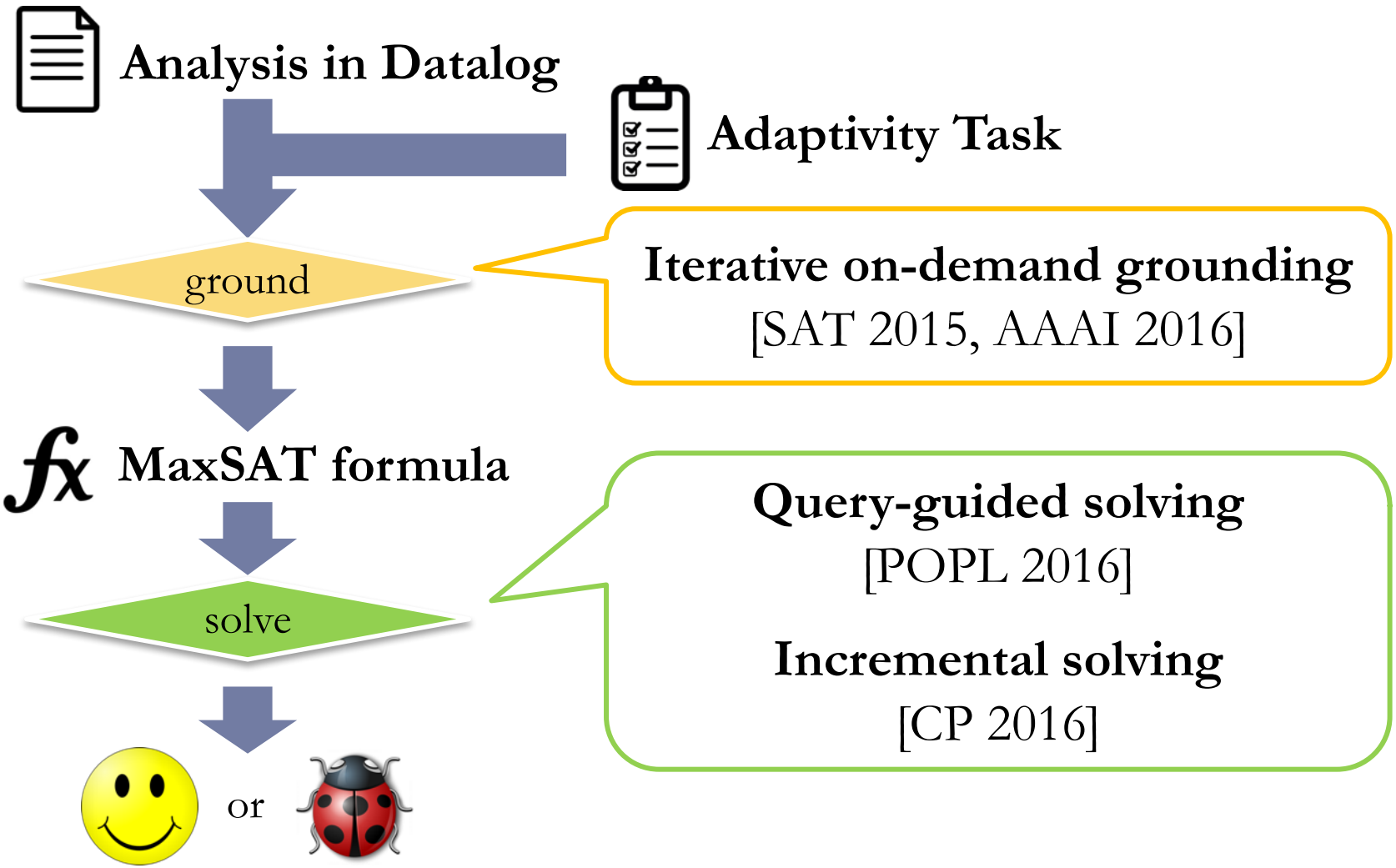
A General Adaptivity Framework

Analysis in Datalog

Adaptivity Task

What to adapt to	Technique	Focus	Tradeoff	Publications
User feedback	User-guided program analysis	Precision (bug-finding)	Soundness vs. completeness	FSE 2015
Assertions of interest	Abstraction refinement	Precision (verification)	Precision vs. scalability	PLDI 2013 PLDI 2014a
Procedure reuse within a program	Hybrid top-down, bottom-up analysis	Scalability (single-program)	Specialization vs. generalization	PLDI 2014b
Procedure reuse across programs	Transfer learning of analysis results	Scalability (across-program)		OOPSLA 2016

A General Adaptivity Framework



Conclusion

- A new paradigm that incorporates user feedback to guide program approximations
- Generalize user feedback on single bug report to other similar bug reports
- A unified framework for adapting program analyses in Datalog using MaxSAT