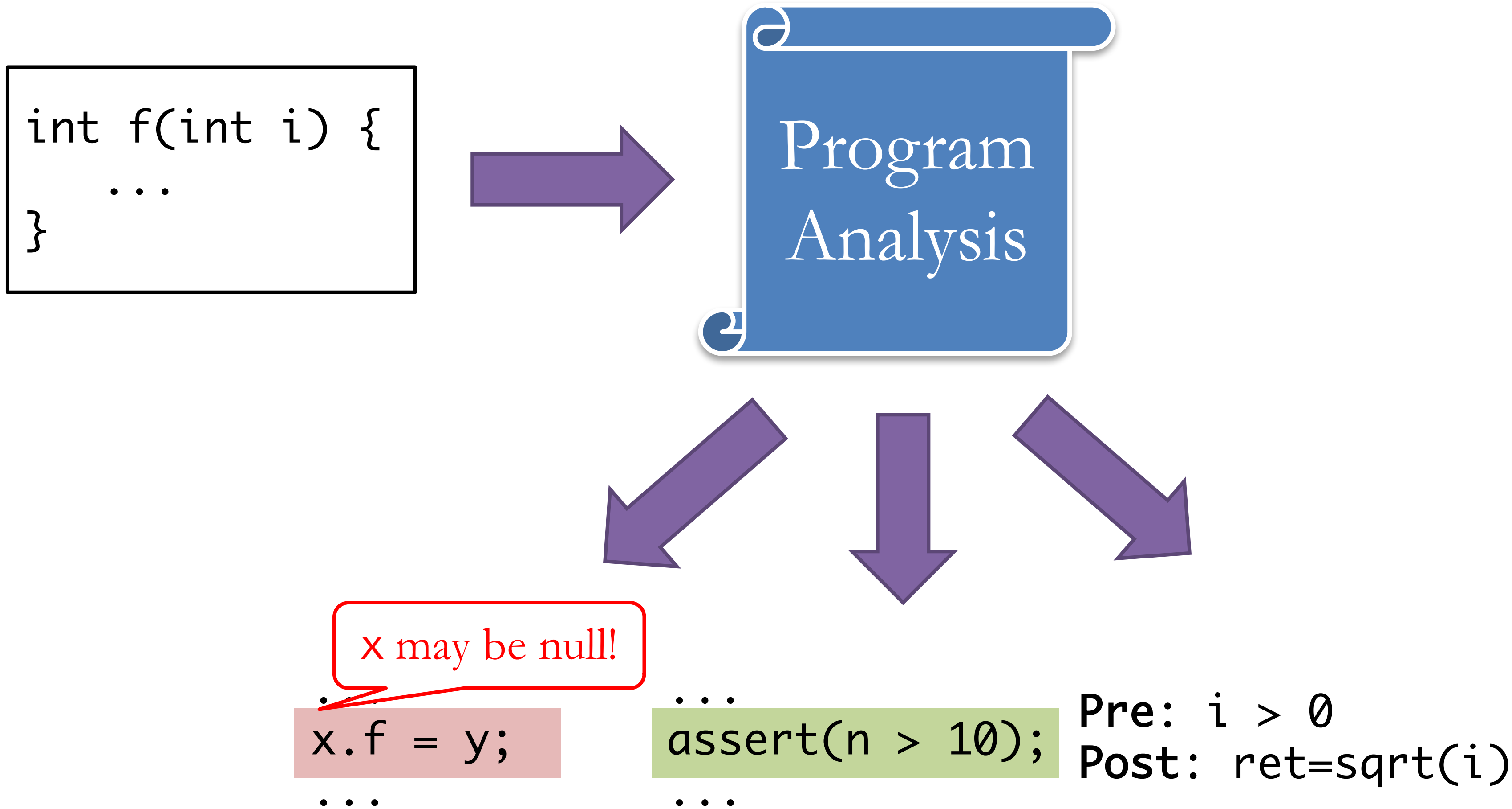
Combining the Logical and the Probabilistic in Program Analysis

Xin Zhang Xujie Si Mayur Naik
University of Pennsylvania

What is Program Analysis?



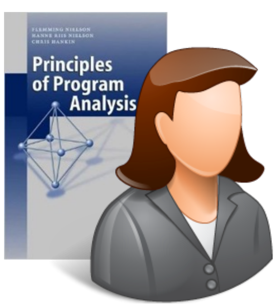
Conventional Logical Approach

+ Easy to specify

```
path(a, a).
path(a, c) :- path(a, b), edge(b, c).
```

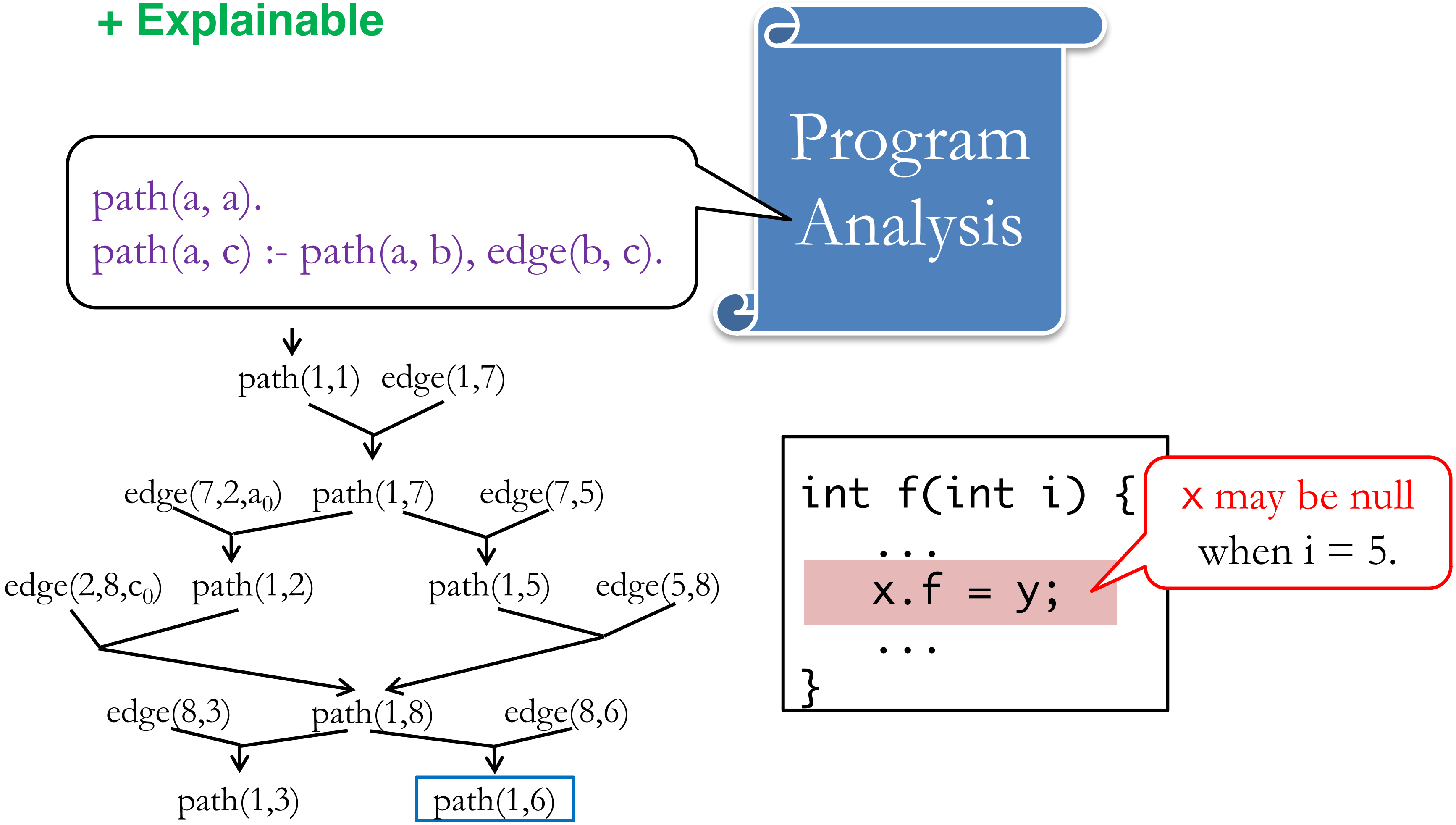
Program Analysis

1. Each node connects to itself.
2. If there is a path from **a** to **b**, and there is an edge from **b** to **c**, then there is a path from **a** to **c**.



Conventional Logical Approach

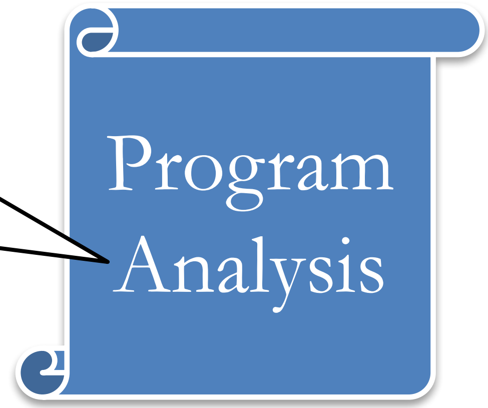
+ Easy to specify
+ Explainable



Conventional Logical Approach

+ Easy to specify
+ Explainable
+ With formal guarantees

```
path(a, a).
path(a, c) :- path(a, b), edge(b, c).
```




```
int f(int i){
    ...
    assert(n > 10);
    ...
}
```

$\forall\, i$

Conventional Logical Approach

+ Easy to specify
+ Explainable
+ With formal guarantees

path(a, a).
path(a, c) :- path(a, b), edge(b, c).

Program Analysis

- Unable to handle uncertainty

Will this buffer overflow lead to a security exploit?

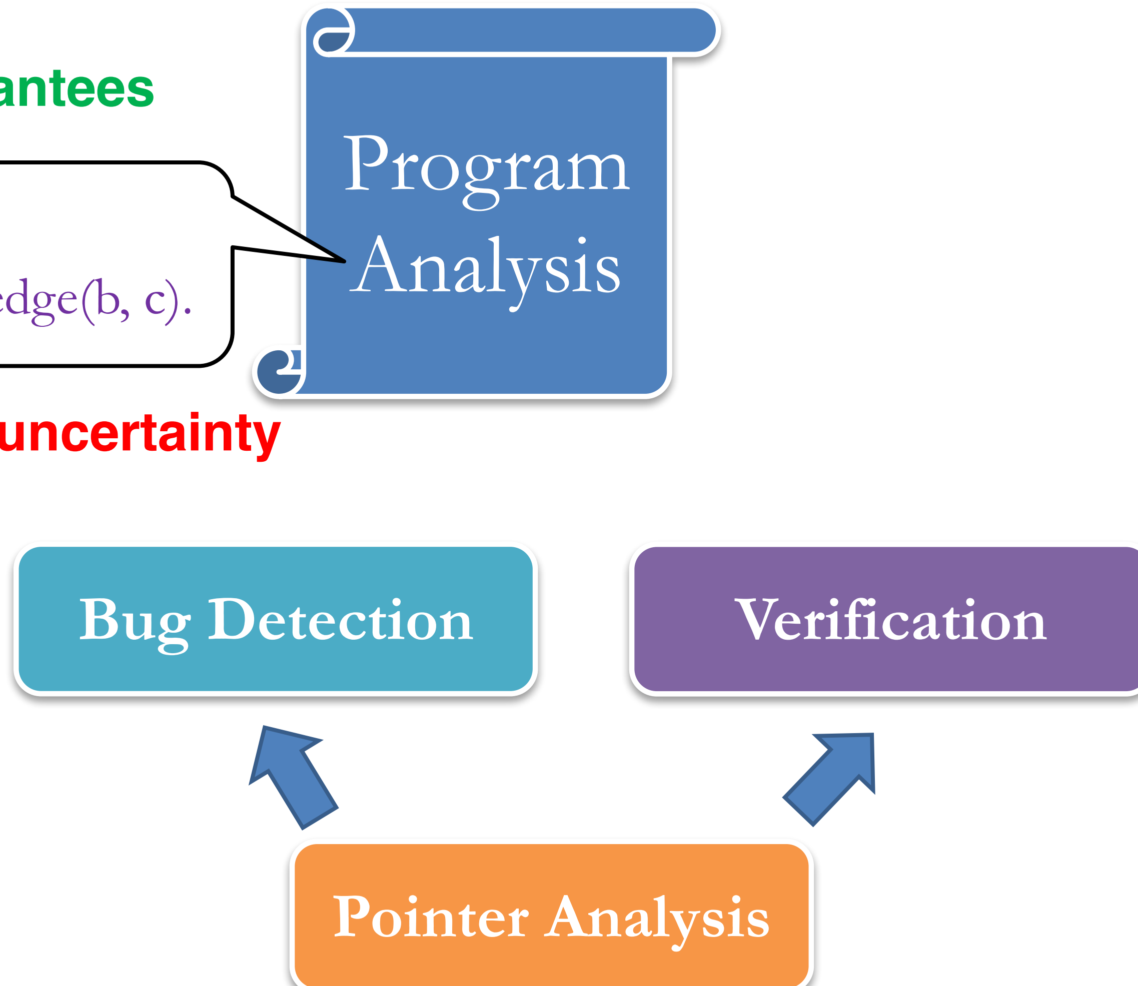
```
*bp++ = TLS1_   SPONSE;
s2n(payload,   p);
memcpy(bp, pl, payload);
```

CVE-2014-0160 (Heartbleed)

Conventional Logical Approach

+ Easy to specify
+ Explainable
+ With formal guarantees

```
path(a, a).
path(a, c) :- path(a, b), edge(b, c).
```



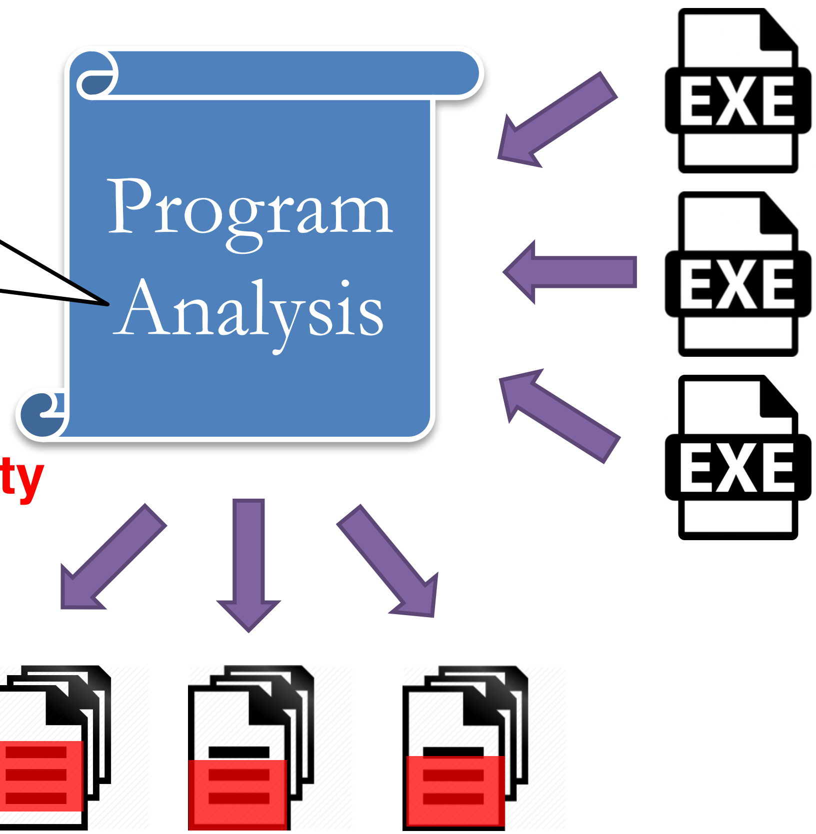
- Unable to handle uncertainty
- Unable to adapt

Conventional Logical Approach

+ Easy to specify
+ Explainable
+ With formal guarantees

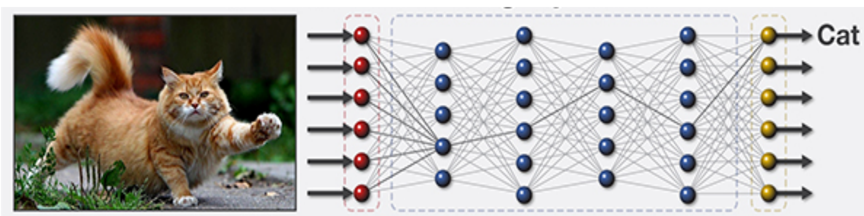
```
path(a, a).
path(a, c) :- path(a, b), edge(b, c).
```

- Unable to handle uncertainty
- Unable to adapt
- Unable to learn



Emerging Probabilistic Approach

- Hard to specify
- Hard to interpret
- Without formal guarantees

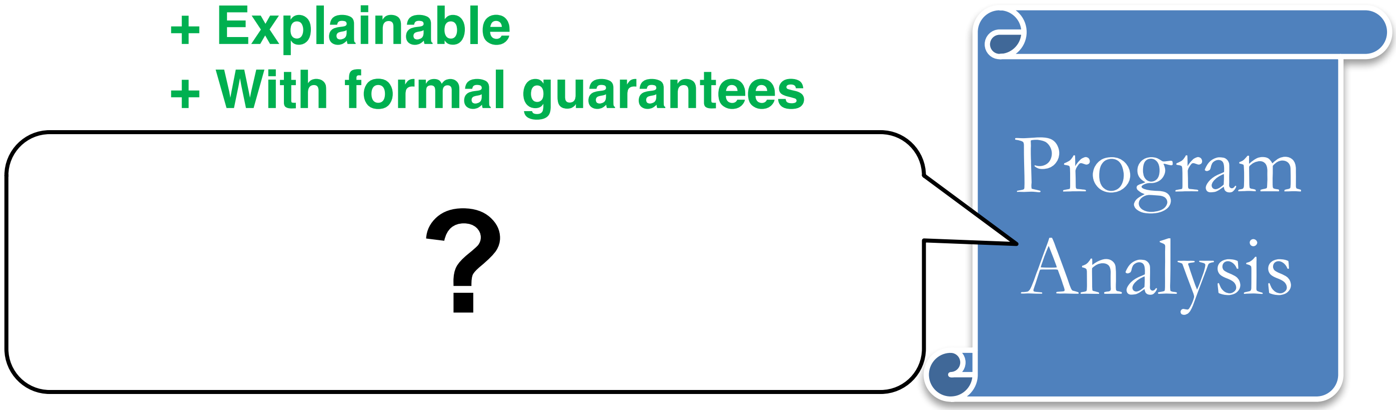


ML System

+ Able to handle uncertainty
+ Able to adapt
+ Able to learn

An Overarching Question

+ Easy to specify
+ Explainable
+ With formal guarantees



+ Able to handle uncertainty
+ Able to adapt
+ Able to learn

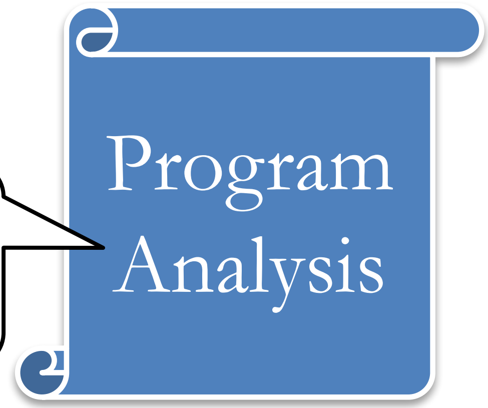
Talk Outline

- Motivation
- A General Approach
- Instance Applications
- Solver
- Empirical Results

Our Approach: Mixed Hard and Soft Constraints

+ Easy to specify
+ Explainable
+ With formal guarantees

```
path(a, a).
path(a, c) :- path(a, b), edge(b, c)   2.0
```



+ Able to handle uncertainty
+ Able to adapt
+ Able to learn

Background: Mixed Hard and Soft Constraints

p(p3, p2) :- p(p1, p2), n(p3, p1)   **5**

Logic

Encodes
Formal Semantics

Probability

Handles Uncertainty

Stochastic Logic Programs (SLP)
[Muggleton, 1996]

Probabilistic Relational Models (PRM)
[Koller, 1999]

Bayesian Logic (BLOG)
[Milch et al., 2005]

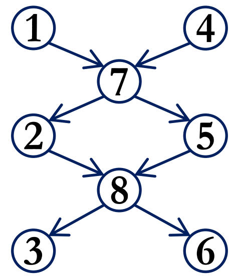
Markov Logic Network (MLN)
[Richardson & Domingos, 2006]

Probabilistic Soft Logic (PSL)
[Kimmig et al., 2012]

...

Background: Mixed Hard and Soft Constraints

Input relations:
edge(a, b)
Output relations:
path(a, b)
Hard Rules:
path(a, a).
path(a, c) :- path(a, b), edge(b, c).
Soft Rules:
¬ path(a, b). **weight 5**



Grounding

Input tuples:
edge(1, 7), edge(4, 7), …

MaxSAT

Hard constraints:

$$edge(1,7) \wedge edge(4,7) \wedge \ldots \wedge$$
$$path(1,1) \wedge path(2,2) \wedge \ldots \wedge$$
$$(path(1,1) \vee \neg path(1,1) \vee \neg edge(1,1)) \wedge$$
$$(path(1,2) \vee \neg path(1,1) \vee \neg edge(1,2)) \wedge$$
$$(path(1,2) \vee \neg path(1,2) \vee \neg edge(2,2)) \wedge$$
$$\ldots$$

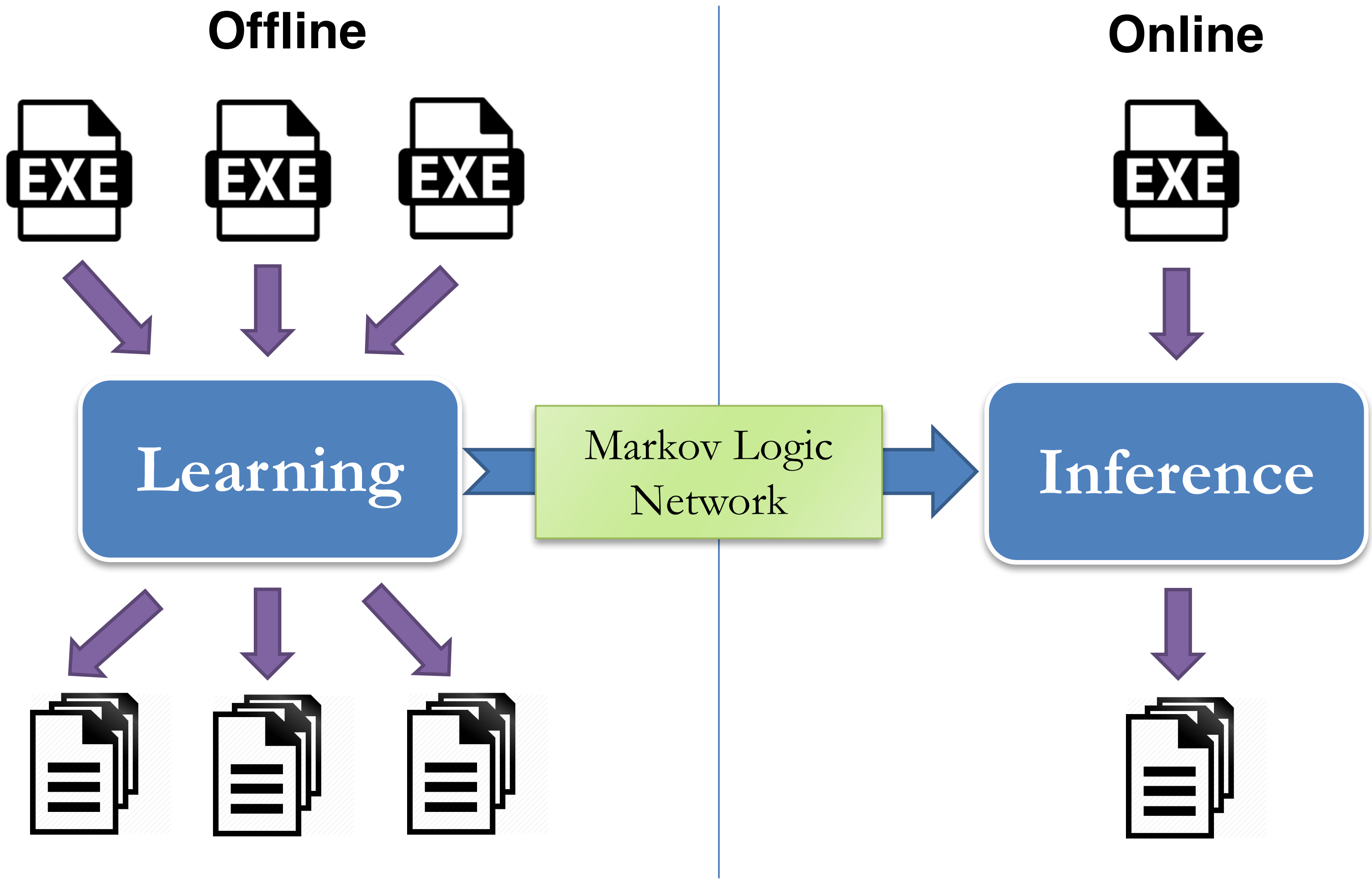
Soft constraints:

$$(\neg path(1,1)\ \textbf{weight 5}) \wedge$$
$$(\neg path(1,2)\ \textbf{weight 5}) \wedge$$
$$\ldots$$

Solving

Solution:
$path(1,1) = \text{T},\ path(2,2) = \text{T},$
$path(1,2) = \text{T},\ path(2,1) = \text{F},$
…

Our Overall Methodology



Talk Outline

- Motivation
- A General Approach
- Instance Applications
- Solver
- Empirical Results

Static Bug Detection: Prevalent Approach

Detected Races	
R1: Race on field org.apache.ftpserver.RequestHandler.request	
org.apache.ftpserver.RequestHandler: 9	org.apache.ftpserver.RequestHandler: 18
R2: Race on field org.apache.ftpserver.RequestHandler.request	
org.apache.ftpserver.RequestHandler: 17	org.apache.ftpserver.RequestHandler: 18
R3: Race on field org.apache.ftpserver.RequestHandler.writer	
org.apache.ftpserver.RequestHandler: 19	org.apache.ftpserver.RequestHandler: 20
R4: Race on field org.apache.ftpserver.RequestHandler.reader	
org.apache.ftpserver.RequestHandler: 21	org.apache.ftpserver.RequestHandler: 22
R5: Race on field org.apache.ftpserver.RequestHandler.controlSocket	
org.apache.ftpserver.RequestHandler: 23	org.apache.ftpserver.RequestHandler: 24

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

Static Bug Detection: Our Approach

Detected Races	
R1: Race on field org.apache.ftpserver.RequestHandler.request	
org.apache.ftpserver.RequestHandler: 9	org.apache.ftpserver.RequestHandler: 18
R2: Race on field org.apache.ftpserver.RequestHandler.request	
org.apache.ftpserver.RequestHandler: 17	org.apache.ftpserver.RequestHandler: 18
R3: Race on field org.apache.ftpserver.RequestHandler.writer	
org.apache.ftpserver.RequestHandler: 19	org.apache.ftpserver.RequestHandler: 20
R4: Race on field org.apache.ftpserver.RequestHandler.reader	
org.apache.ftpserver.RequestHandler: 21	org.apache.ftpserver.RequestHandler: 22
R5: Race on field org.apache.ftpserver.RequestHandler.controlSocket	
org.apache.ftpserver.RequestHandler: 23	org.apache.ftpserver.RequestHandler: 24

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

Static Bug Detection: Our Approach

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

Eliminated Races	
E1: Race on field org.apache.ftpserver.RequestHandler. isClosed	
org.apache.ftpserver.RequestHandler: 13	org.apache.ftpserver.RequestHandler: 15
E2: Race on field org.apache.ftpserver.RequestHandler.request	
org.apache.ftpserver.RequestHandler: 17	org.apache.ftpserver.RequestHandler: 18
E3: Race on field org.apache.ftpserver.RequestHandler.writer	
org.apache.ftpserver.RequestHandler: 19	org.apache.ftpserver.RequestHandler: 20
E4: Race on field org.apache.ftpserver.RequestHandler.reader	
org.apache.ftpserver.RequestHandler: 21	org.apache.ftpserver.RequestHandler: 22
E5: Race on field org.apache.ftpserver.RequestHandler.controlSocket	
org.apache.ftpserver.RequestHandler: 23	org.apache.ftpserver.RequestHandler: 24

Instance Applications [CAV'17 Invited Tutorial]

	Hard Rules	Soft Rules
Static Bug Detection [FSE'15]	analysis rules	analysis rule_i **weight w_i** **confidence of writer** ¬ result_j **weight w_j** **confidence of user**
Automated Verification [PLDI'14, POPL'16]	analysis rules $\text{abstraction}_1 \oplus \ldots \oplus \text{abstraction}_n$	¬ result_i **weight w_i** **query resolution award** abstraction_j **weight w_j** **abstraction cost**
Interactive Verification [OOPSLA'17]	analysis rules	¬ cause_i **weight w_i** **cost of inspection** result_j **weight w_j** **reward of resolution**

An Example: Datarace Analysis

Input relations:

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

Output

paral

progra
immed

If p1 & p2 may happen in parallel,

a

n,

allel.

If p2 & p1 may happen in parallel,
then p1 & p2 may happen in parallel.

& p2 may have a datarace.

Rules:

paralle

p
happ

parallel(p1, p2) :- p

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

…

An Example: Datarace Analysis

Input relations:

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

Output relations:

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

```
a = 1;
if (a > 2) { // p1
   ... // p2
}
```

Rules:

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

“Soft” Rule

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

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

“Hard” Rule

…

¬race(x2, x1). **weight 25**

An Example: Datarace Analysis

```
public class RequestHandler {
    Request request;
    FtpWriter writer;
    BufferedReader reader;
    Socket controlSocket;
    boolean isConnectionClosed;
    …

 public Request getRequest() {
        return request;
 }
```

```
public void close() {
    synchronized (this) {
       if (isClosed)
           return;
       isClosed = true;
    }
    request.clear();
    request = null;
    writer.close();
    writer = null;
    reader.close();
    reader = null;
    controlSocket.close();
    controlSocket = null;
}
```

Source code snippet from Apache FTP Server

An Example: Datarace Analysis

```
public class RequestHandler {
    Request request;
    FtpWriter writer;
    BufferedReader reader;
    Socket controlSocket;
    boolean isConnectionClosed;
    …

public Request getRequest() {
        return request;
}
```

R1

```
public void close() {
    synchronized (this) {
        if (isClosed)
            return;
        isClosed = true;
    }
    request.clear();
    request = null;
    writer.close();
    writer = null;
    reader.close();
    reader = null;
    controlSocket.close();
    controlSocket = null;
}
```

Source code snippet from Apache FTP Server

An Example: Datarace Analysis

```java
public class RequestHandler {
    Request request;
    FtpWriter writer;
    BufferedReader reader;
    Socket controlSocket;
    boolean isConnectionClosed;
    …

 public Request getRequest() {
       return request;
 }
```

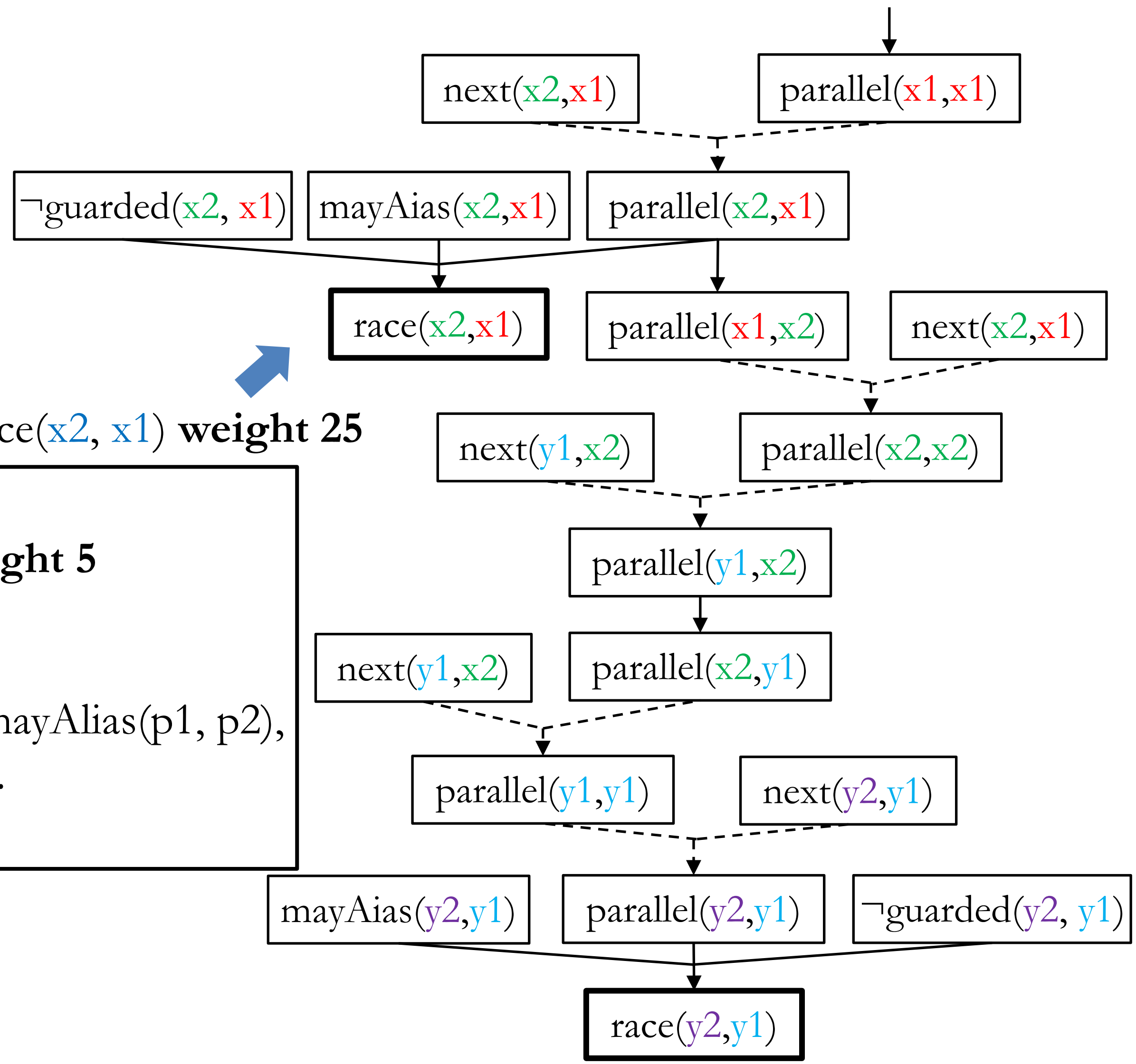
```java
public void close() {
    synchronized (this) {
       if (isClosed)
           return;
       isClosed = true;
    }
    request.clear();
    request = null;
    writer.close();
    writer = null;
    reader.close();
    reader = null;
    controlSocket.close();
    controlSocket = null;
}
```

R2 R3 R4 R5

Source code snippet from Apache FTP Server

How Does Generalization Work?

```
            ...
    request.clear();// x1
    request = null; // x2
    writer.close(); // y1
    writer = null;  // y2
            ...
```



¬race(x2, x1) **weight 25**

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).

...

How Does Generalization Work?

```
...
request.clear();// x1
request = null; // x2
writer.close(); // y1
writer = null;  // y2
...
```

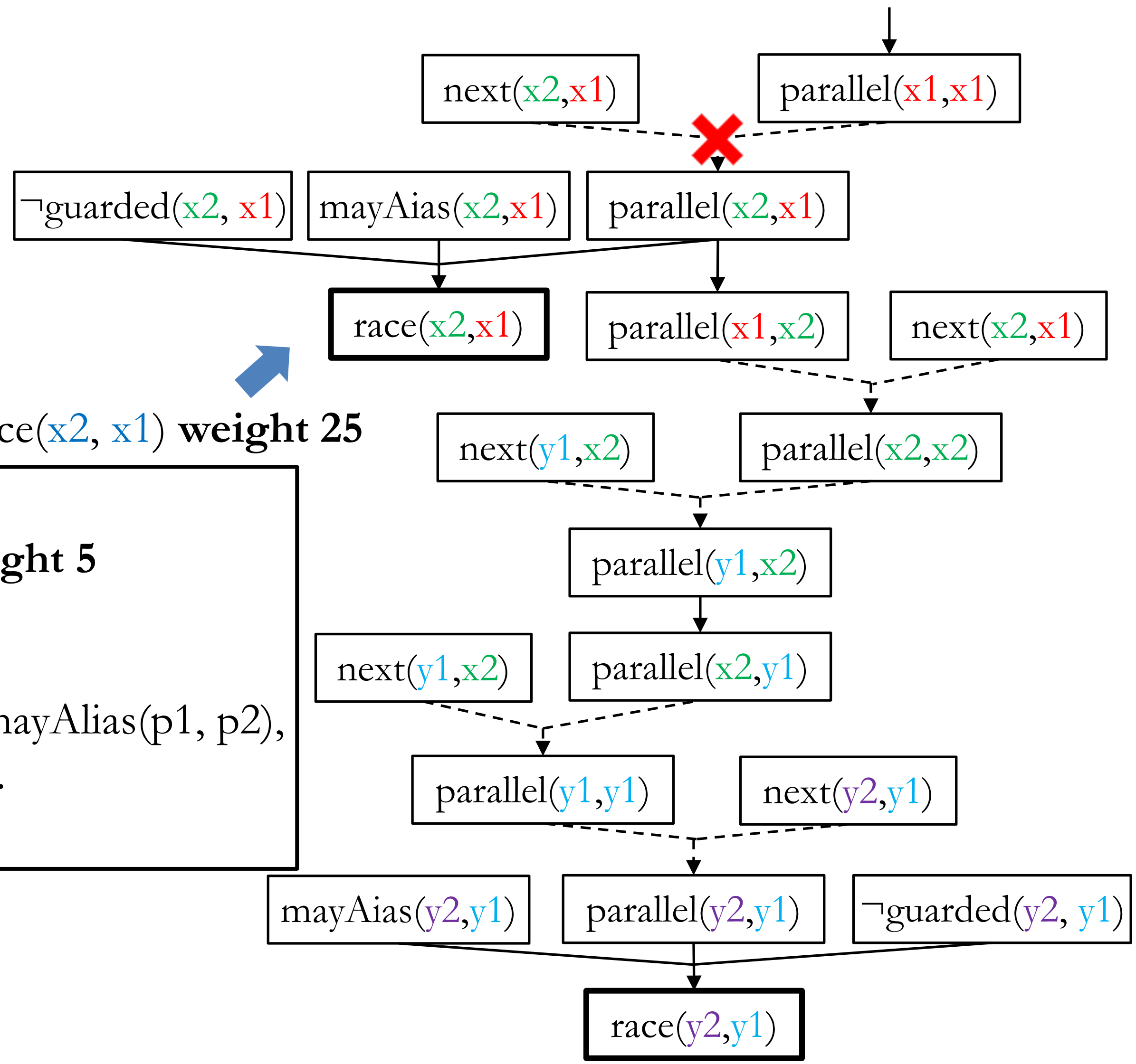
¬race(x2, x1) **weight 25**

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).

…



How Does Generalization Work?

```
    …
    request.clear();// x1
    request = null; // x2
    writer.close(); // y1
    writer = null;  // y2
    …
```

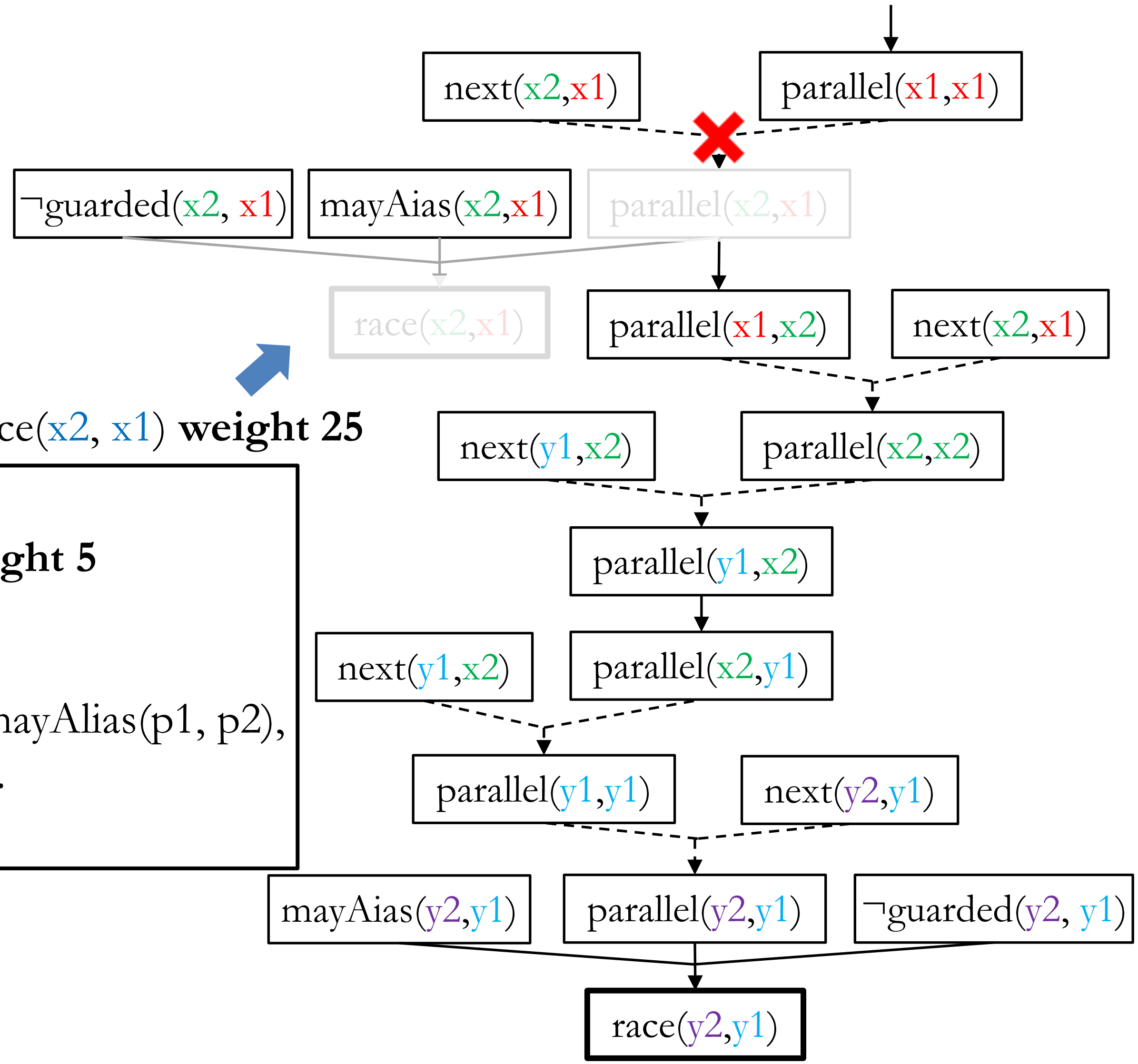
¬race(x2, x1) **weight 25**

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

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

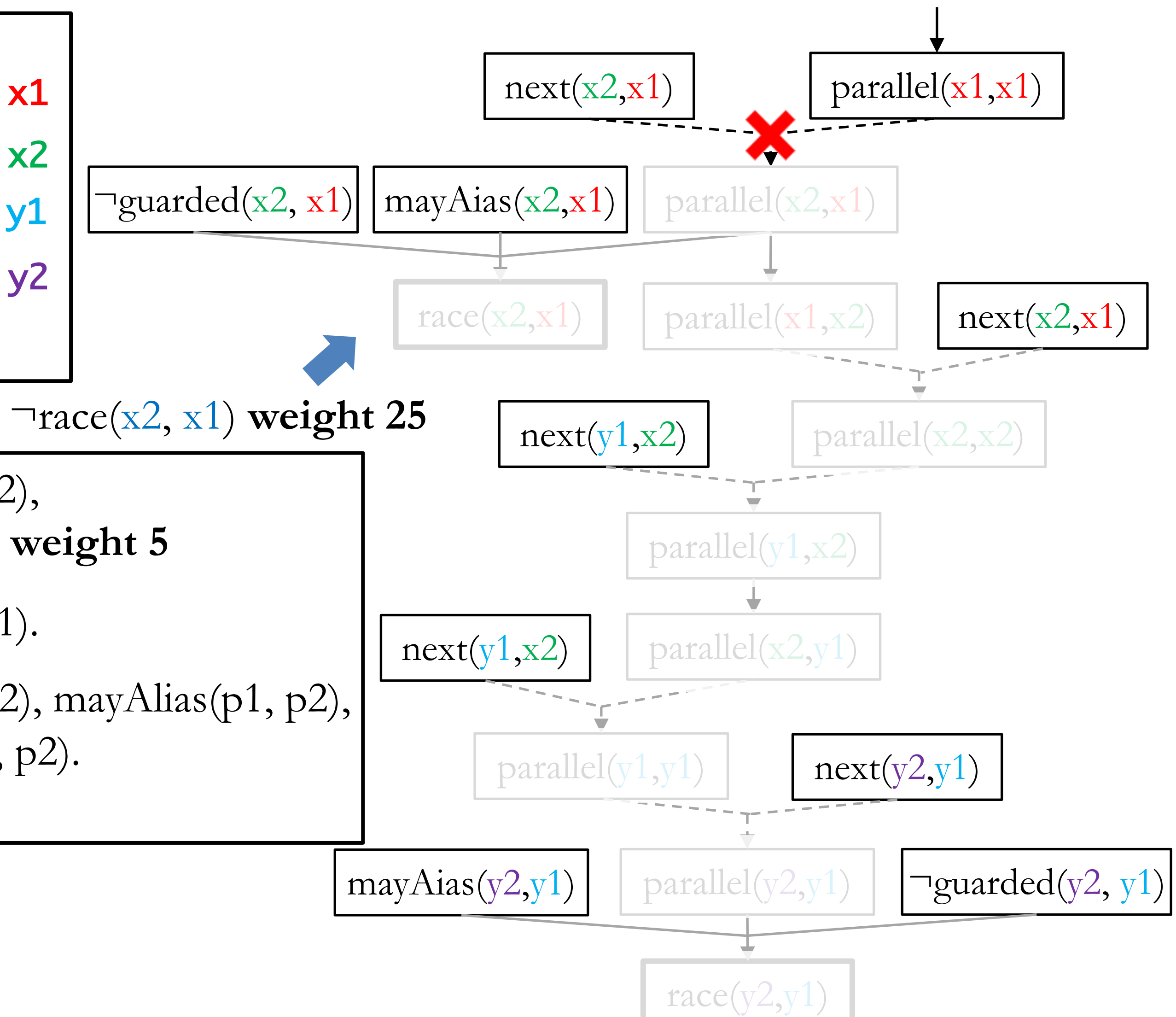
race(p1, p2) :- parallel(p1, p2), mayAlias(p1, p2),
¬guarded(p1, p2).

…



How Does Generalization Work?

```
                ...
      request.clear();// x1
      request = null; // x2
      writer.close(); // y1
      writer = null;  // y2
                ...
```



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).

…

Talk Outline

- Motivation
- A General Approach
- Instance Applications
- Solver
- Empirical Results

The Inference Problem

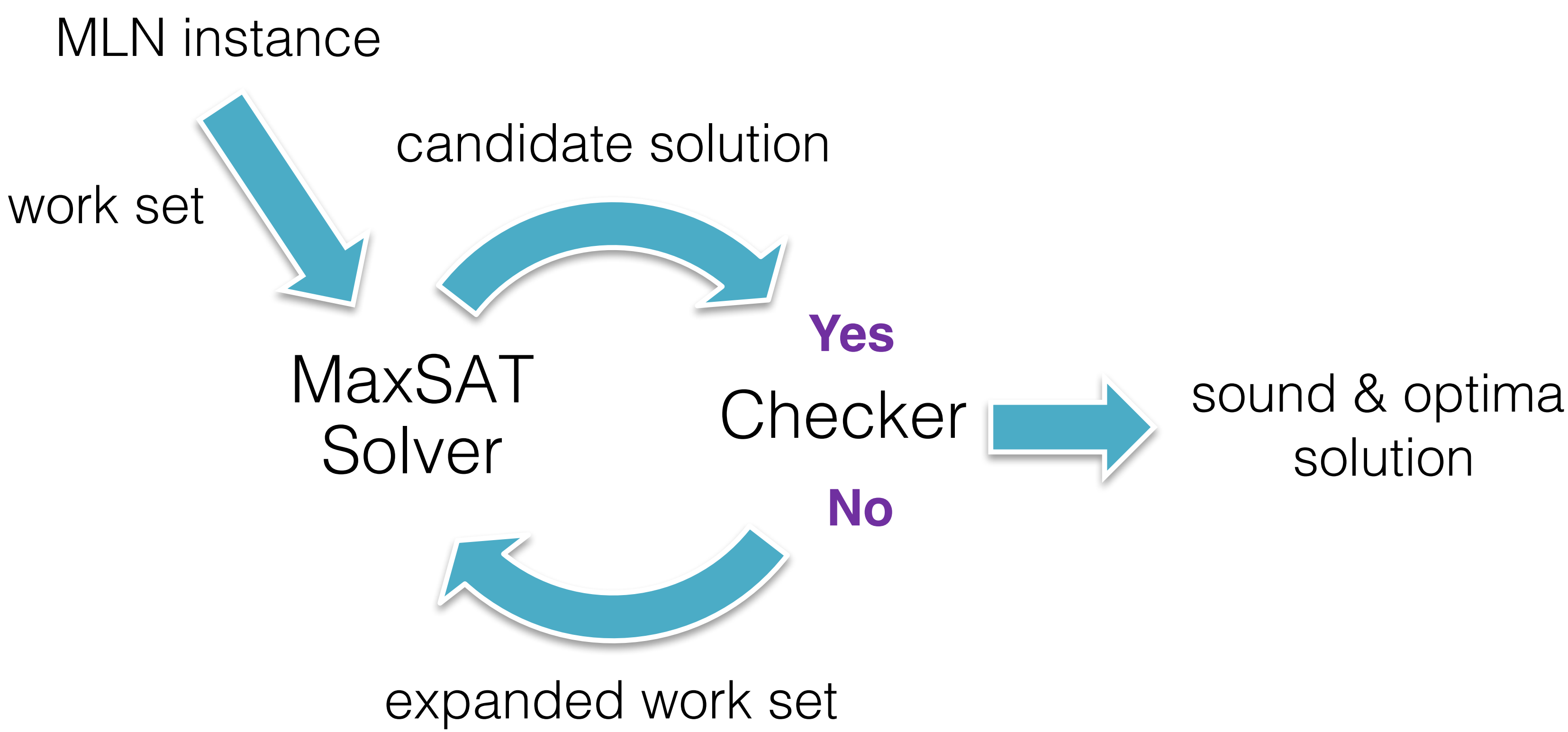
Input relations:
edge(a, b)
Output relations:
path(a, b)
Hard Rules:
path(a, a).
path(a, c) :- path(a, b), edge(b, c).
Soft Rules:
¬ path(a, b). **weight 5**

Markov Logic Network

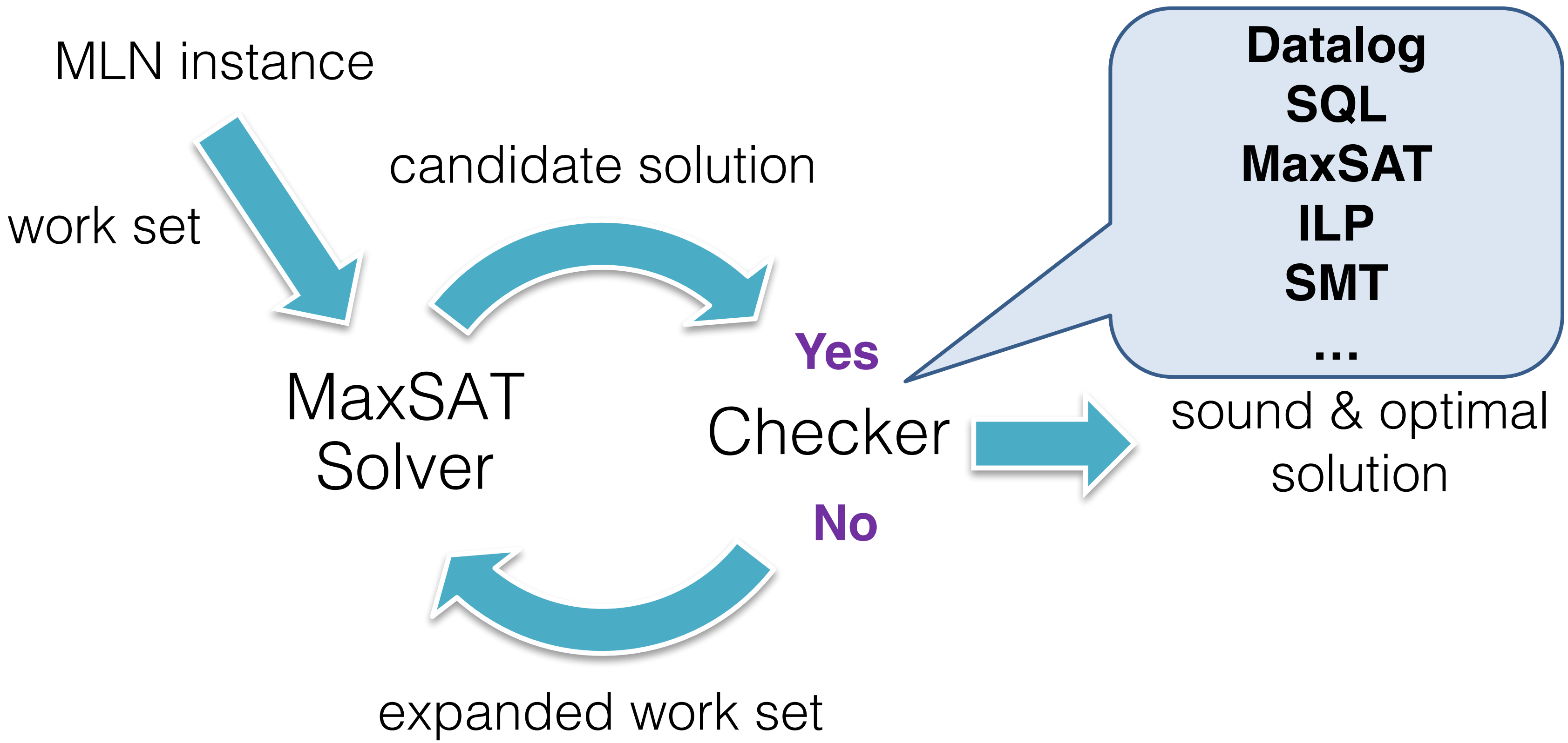
Solver

Existing Solvers	Soundness	Optimality	Scalability
Tuffy [VLDB'11]	✗	✗	✓
Alchemy [ML'06]	✗	✗	✓
CPI [UAI'08]	✓	✓	✗
RockIt [AAAI'13]	✓	✓	✗
Z3 [TACAS'08]	✓	✓	✗

The Nichrome Solver

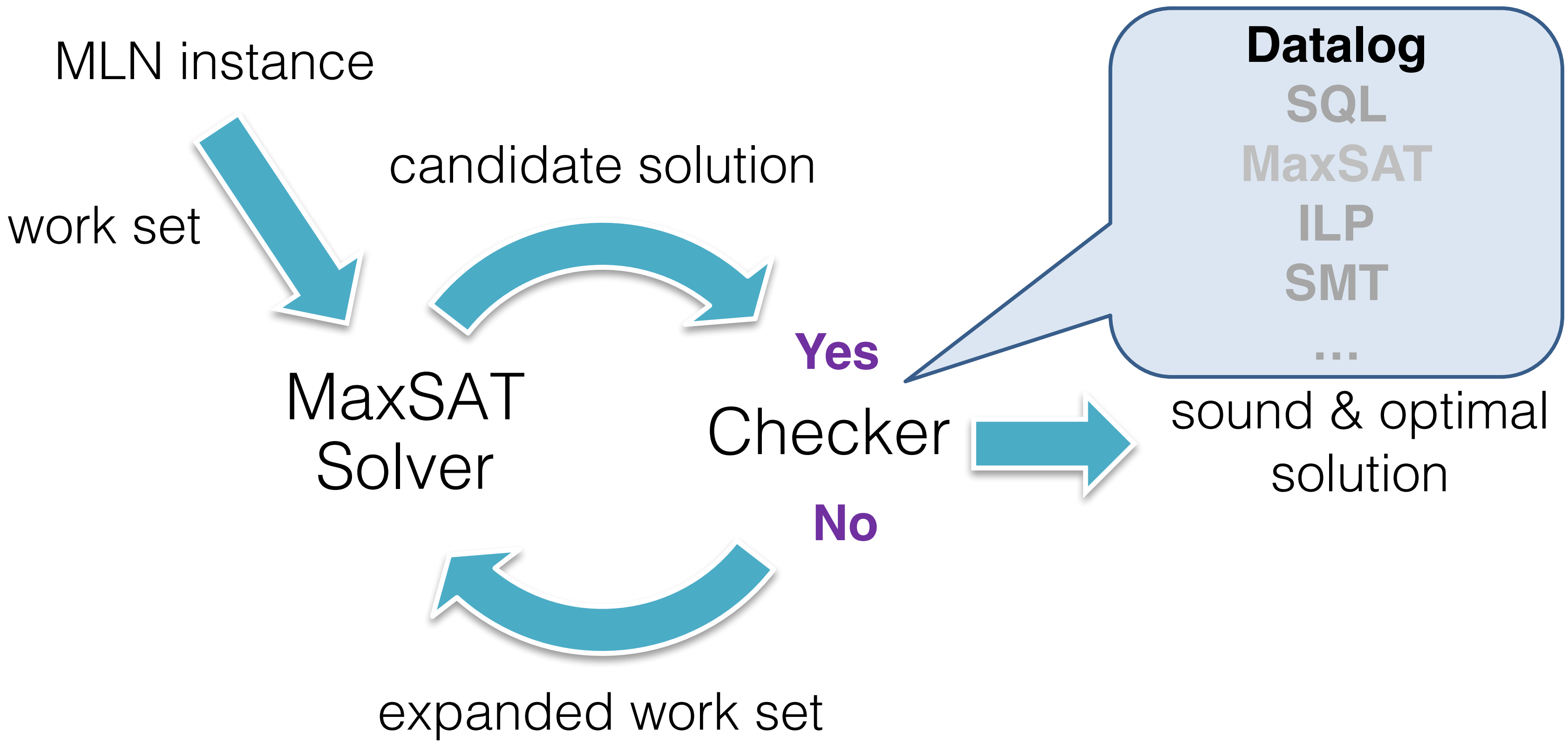


The Nichrome Solver



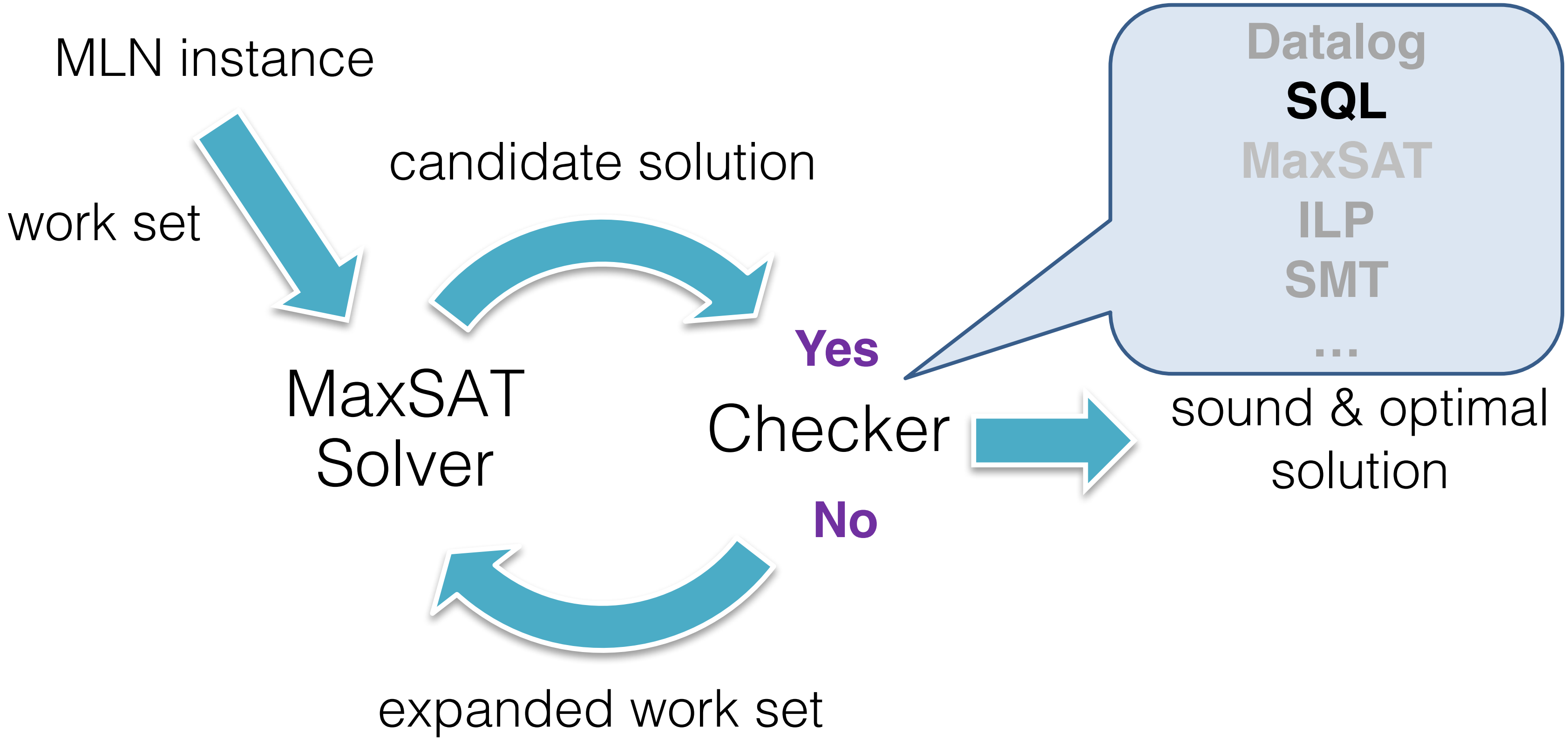
Volt: A Lazy Grounding Framework for Solving Very Large MaxSAT Instances [SAT 2015]

The Nichrome Solver



On Abstraction Refinement for Program Analyses in Datalog [PLDI 2014]

The Nichrome Solver



Scaling Relational Inference Using Proofs and Refutations
[AAAI 2016]

The Nichrome Solver

MaxSAT instance

work set

candidate solution

MaxSAT Solver

Yes

Checker

No

Datalog
SQL
MaxSAT
ILP
SMT
...

sound & optimal solution

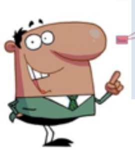
expanded work set

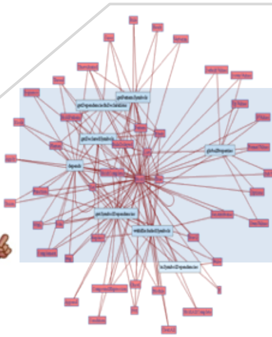
Program Reasoning:

Does **head** alias **tail** on line 50 in Complex.java?

Information Retrieval:

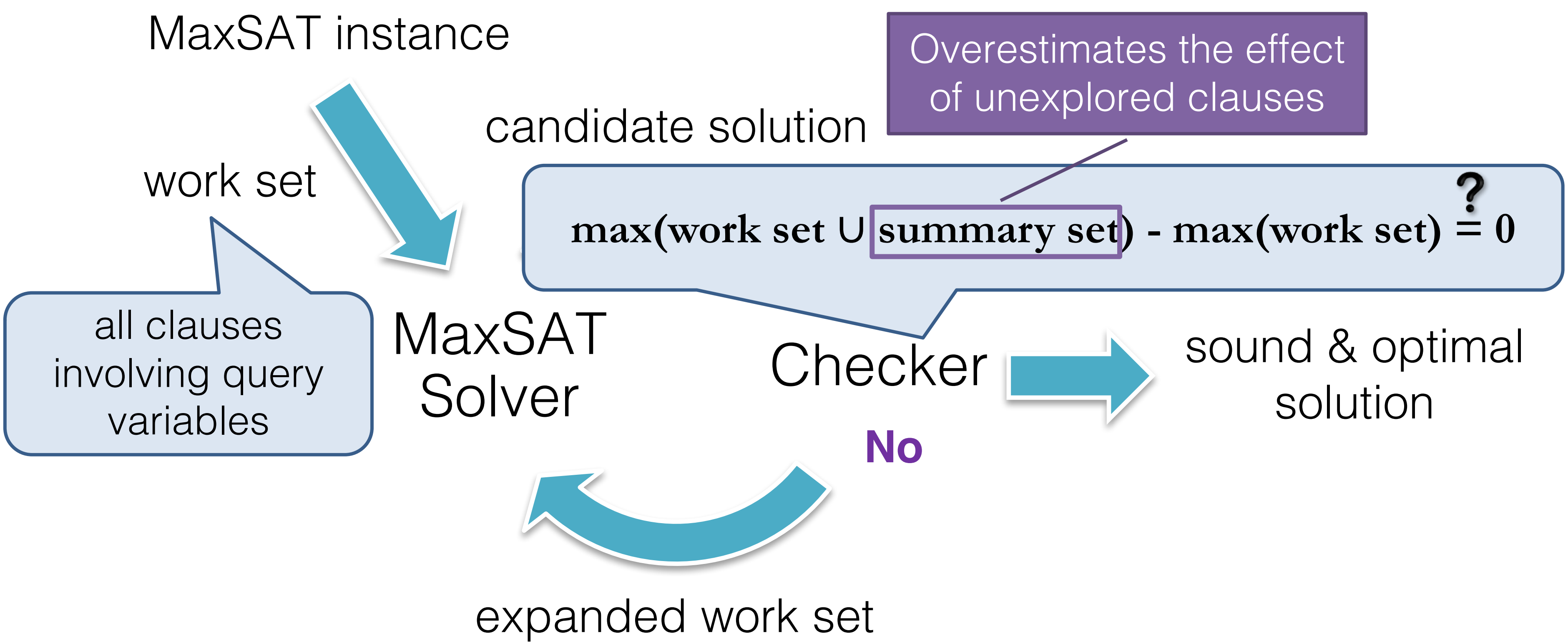
Is **Dijkstra** most likely author of **Structured Programming**?





Query-Guided Maximum Satisfiability
[POPL 2016]

The Nichrome Solver



Query-Guided Maximum Satisfiability
[POPL 2016]

Talk Outline

- Motivation
- A General Approach
- Instance Applications
- Solver
- Empirical Results

Experimental Setup

- Control Study:
  - **Analyses:** (1) Datarace analysis, (2) Pointer analysis

    30 rules
    18 input relations
    18 output relations

    76 rules
    50 input relations
    42 output relations

  - **Benchmarks:** 7 Java programs (130-200 KLOC each)
  - **Feedback:** Automat … PLDI'14]

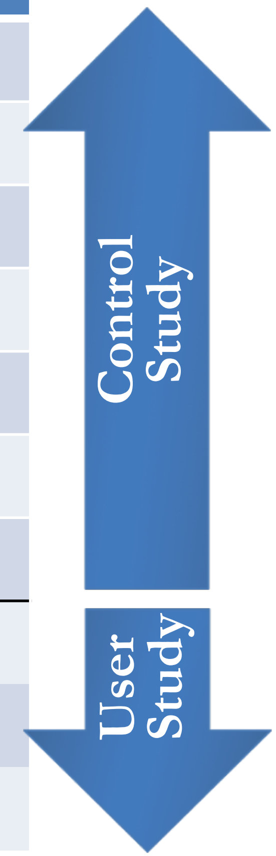
- User Study:

    76 rules
    52 input relations
    42 output relations

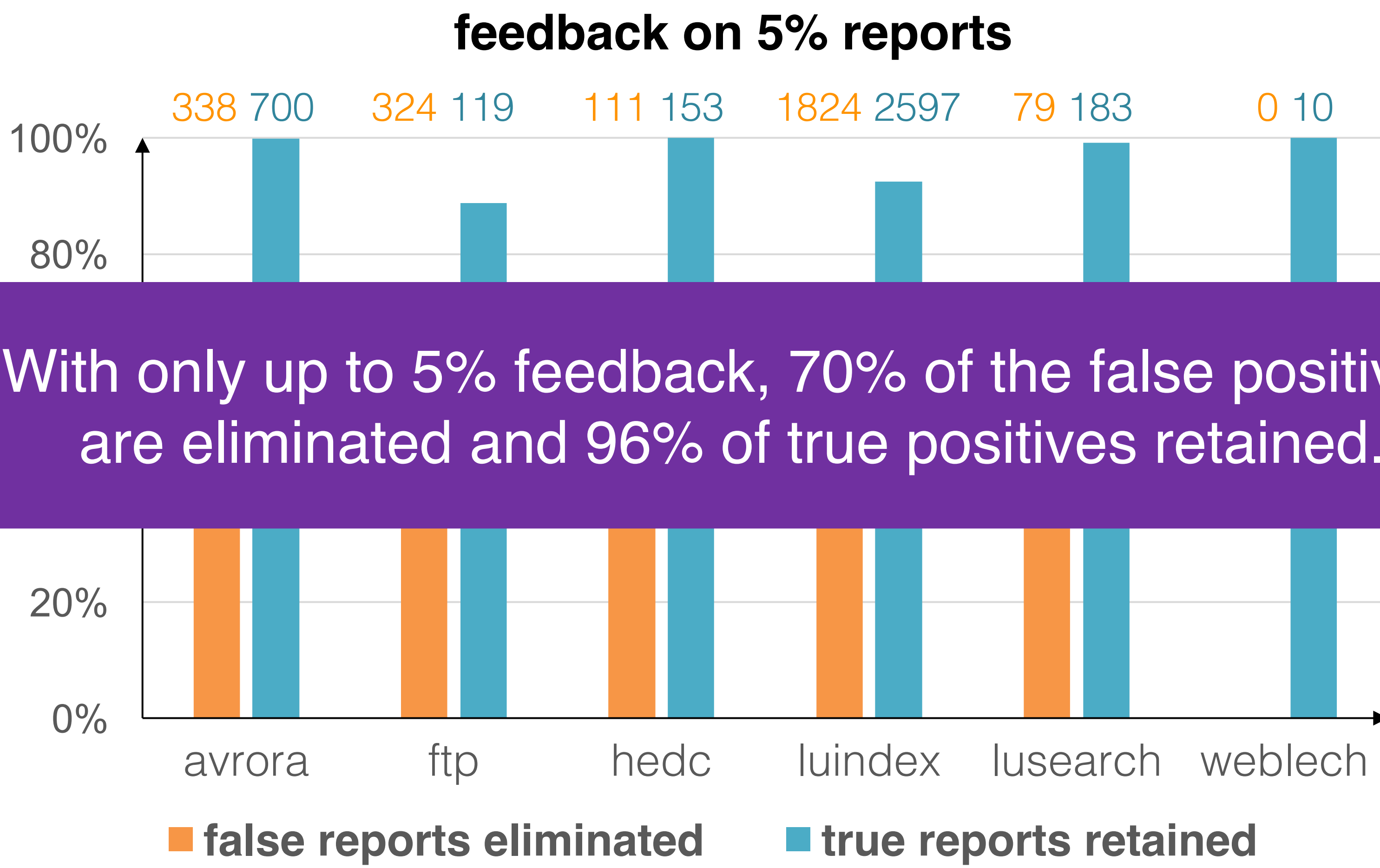
  - **Analyses:** Information flow analysis
  - **Benchmarks:** 3 Android Apps
  - **Feedback:** 9 users

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
App 1	5	13	0.3	0.6
App 2	4	12	0.2	0.6
App 3	17	46	1.3	4.2

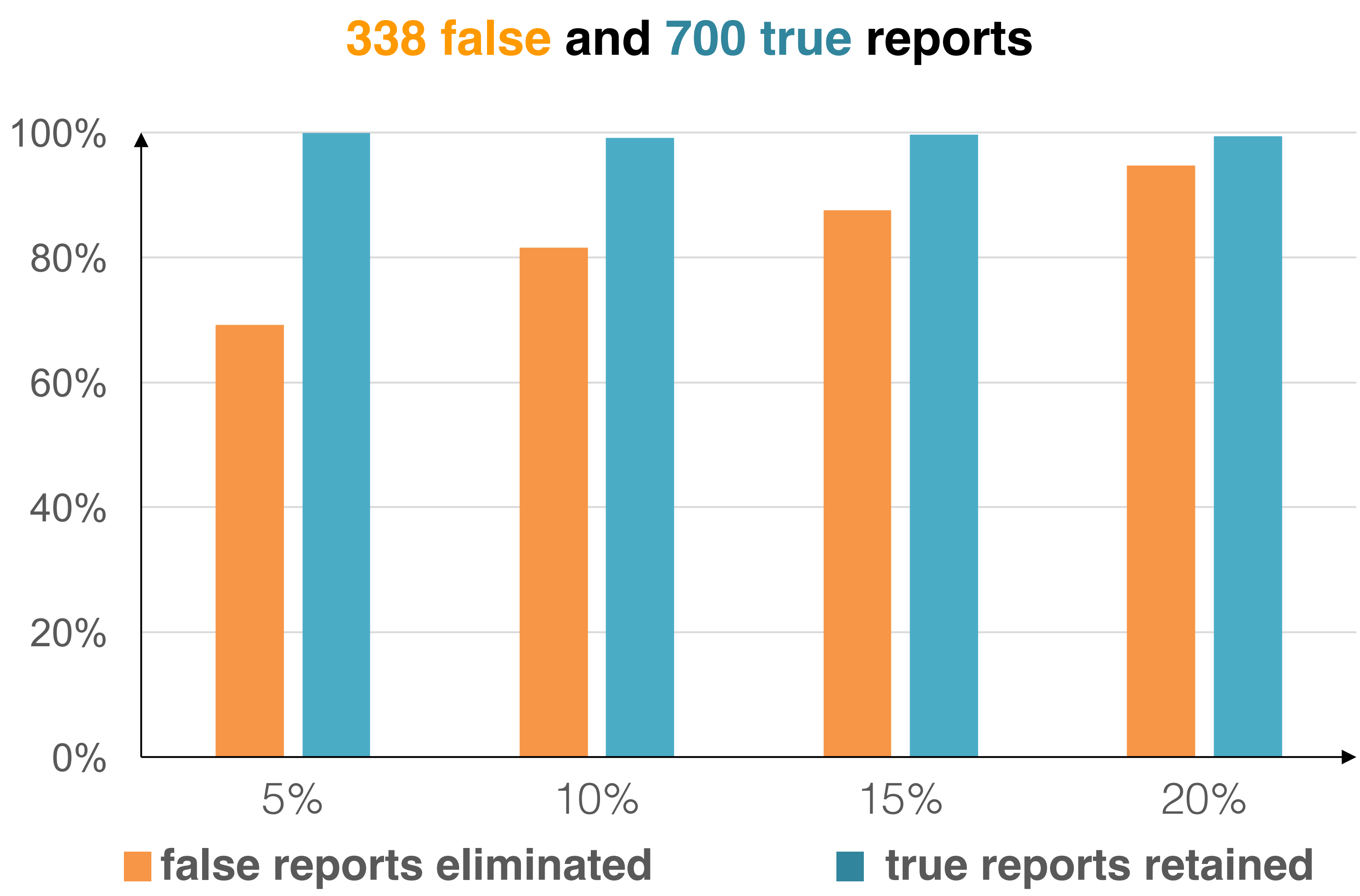


Precision Results: Datarace Analysis



With only up to 5% feedback, 70% of the false positives are eliminated and 96% of true positives retained.

Precision Results: Varying Amount of Feedback



Solver Performance Results

Winner of the 2015 Annual MaxSAT Competition

	runni (sec				# clauses (M = million)	
	Nichrome	Eva500	Nichrome	Eva500	Nichrome	Eva500
ftp	**16**	11	**16**	1,262	**29K**	3M
hedc	**23**	21	**181**	1,918	**400K**	4.8M
weblech	**4**	timeout	**363**	timeout	**922K**	8.4M
antlr	**190**	timeout	**1,405**	timeout	**3.3M**	13M
avrora	**178**	timeout	**1,095**	timeout	**2.6M**	16.3M
chart	**253**	timeout	**721**	timeout	**1.8M**	22.3M
luindex	**169**	timeout	**944**	timeout	**2.2M**	11.9M

Solver Performance Results

	running time (seconds)		peak memory (MB)		# clauses (M = million)	
	Nichrome	**Eva500**	**Nichrome**	**Eva500**	**Nichrome**	**Eva500**
ftp	**16**	11	**16**	1,262	**29K**	3M
hedc	**23**	21	**181**	1,918	**400K**	4.8M
weblech	**4**	timeout	**363**	timeout	**922K**	8.4M
antlr	**190**	timeout	**1,405**	timeout	**3.3M**	13M
avrora	**178**	timeout	**1,095**	timeout	**2.6M**	16.3M
chart	**253**	timeout	**721**	timeout	**1.8M**	22.3M
luindex	[illegible]	[illegible]	**944**	timeout	**2.2M**	11.9M
AR	[illegible]	[illegible]	**4**	timeout	**2K**	7.9M
ER	[illegible]	[illegible]	**6**	44	**9K**	4.8M
IE	[illegible]	[illegible]	**13**	335	**27K**	0.9M

AR — Advisor Recommendation

ER — Entity Resolution

IE — Information Extraction

Conclusions

- Combining logical and probabilistic reasoning in program analysis provides best of both worlds
- Our approach: extend conventional program analyses by augmenting logical rules with weights
  => Adopt semantics of MLN from the AI community
- New solver that achieves accuracy and scalability by leveraging program analysis domain insights