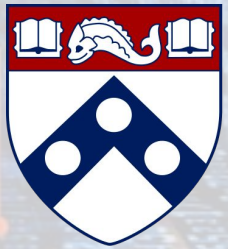
Learning to Reason about Programs

Mayur Naik

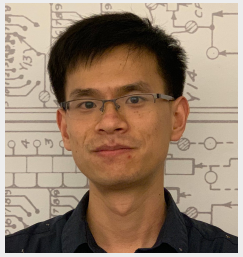
Associate Professor
Computer & Information Science



Penn
University of Pennsylvania

Acknowledgments


Halley Young


Xujie Si


Sulekha Kulkarni

PhD Students


Pardis Pashakhanloo


Jiani Huang


Elizabeth Dinella


Mukund Raghothaman

Postdocs


Kihong Heo


Yuan Yang (PhD Student)


Hanjun Dai (PhD Student)

Collaborators


Osbert Bastani (Faculty)


Le Song (Faculty)

Where is ML for Programming?

DeepMind's AlphaZero AI is the new champion in chess, shogi, and Go

Dec 2018

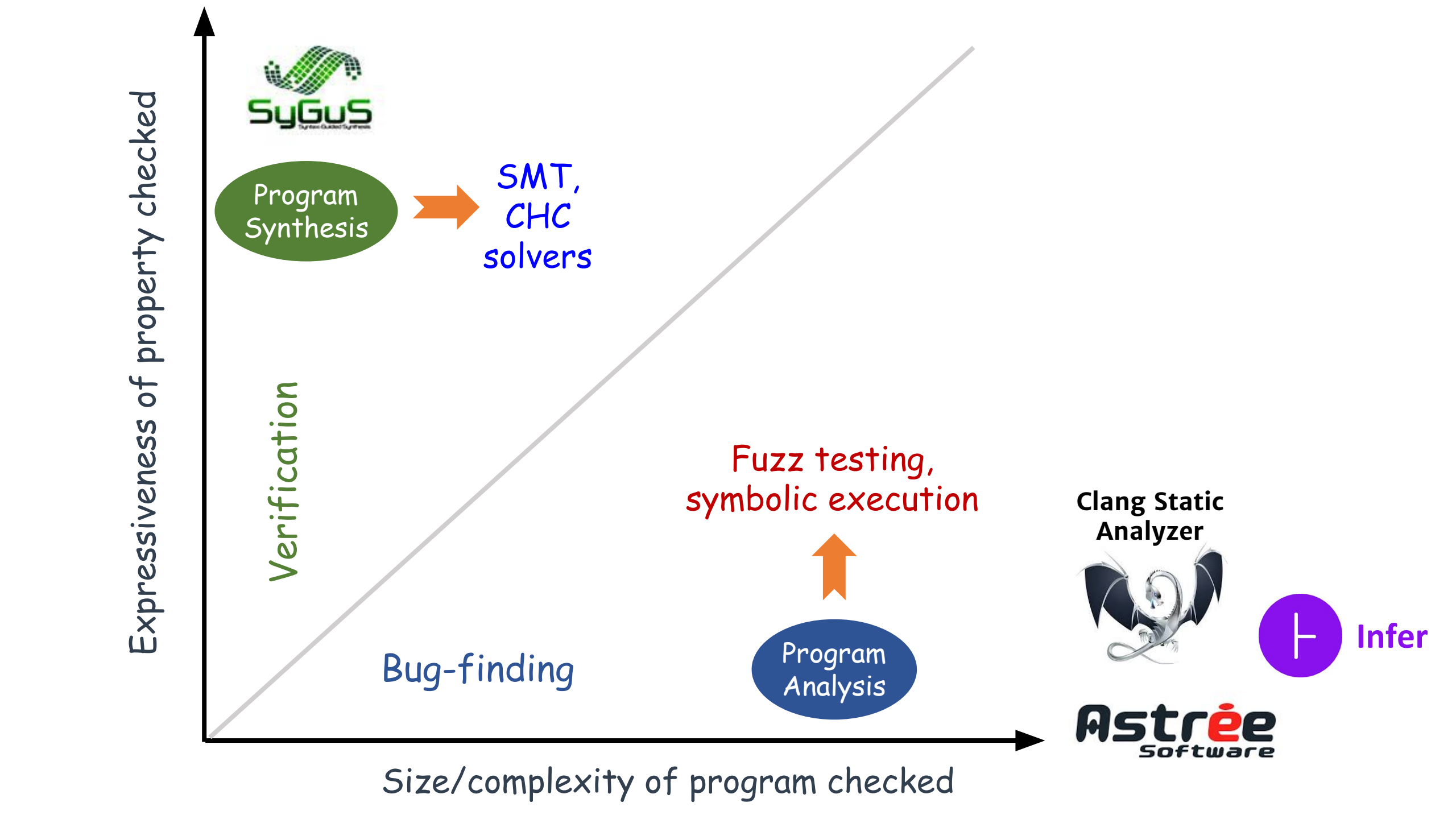
Forget chess and Go - artificial intelligence can now beat humans in StarCraft II

Jan 2019

A.I. Is Flying Drones (Very, Very Slowly)

Artificial intelligence has bested top players in chess, Go and even StarCraft. But can it fly a drone faster than a pro racer? More than $1 million is on the line to find out.

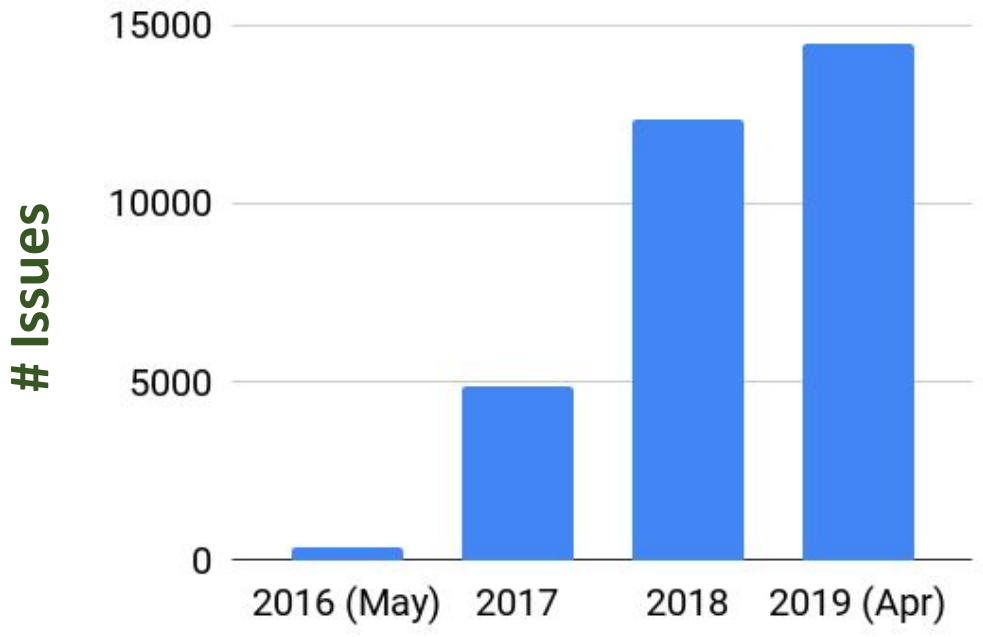
March 2019


Expressiveness of property checked
Size/complexity of program checked
SyGuS
Program Synthesis
SMT, CHC solvers
Verification
Fuzz testing, symbolic execution
Program Analysis
Bug-finding
Clang Static Analyzer
Infer
Astrée Software

State of the Practice: Bug-Finding

OSS-Fuzz: Continuous Fuzzing of Open-Source Software

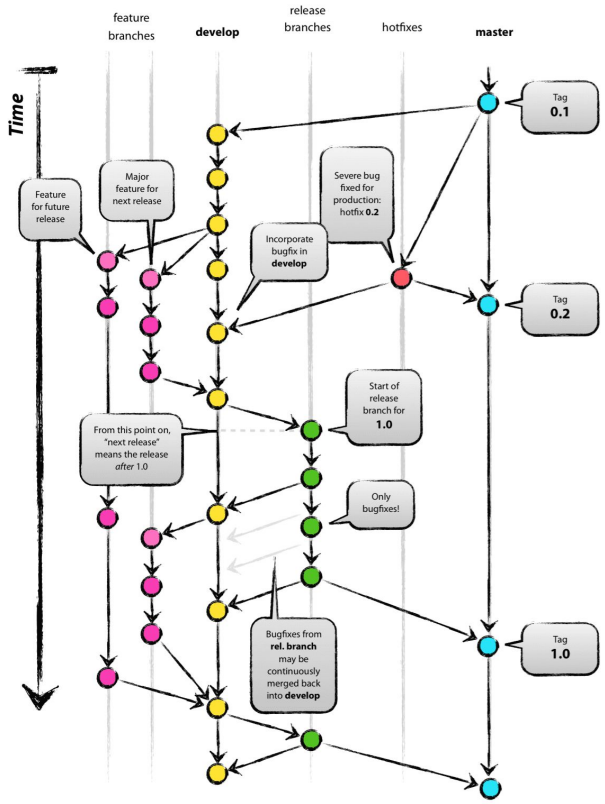
- heap buffer overflows
- global buffer overflows
- stack buffer overflows
- use after frees
- uninitialized memory
- stack overflows
- timeouts
- ooms
- leaks
- ubsan
- unknown crashes
- other (e.g. assertions)



Project	# MLOC
jsc	5.05
gnutls	2.31
llvm	2.18
solidity	2.10
grpc	1.82

(top 5 projects by size)

A Challenge in Bug-Finding



OpenSSL
Cryptography and SSL/TLS Toolkit

- **Heartbleed**
- **Buffer over-read** bug in OpenSSL's Heartbeat implementation
- Introduced in 2011, discovered in 2014
- Estimated to affect nearly 66% of all web servers

The coding mistake that caused Heartbleed can be traced to a single line of code:

```
memcpy(bp, pl, payload);
```

`memcpy()` is the command that copies data. `bp` is the place it's copying it to, `pl` is where it's being copied from, and `payload` is the length of the data being copied. The problem is that there's never any attempt to check if the amount of data in `pl` is equal to the value given of `payload`.

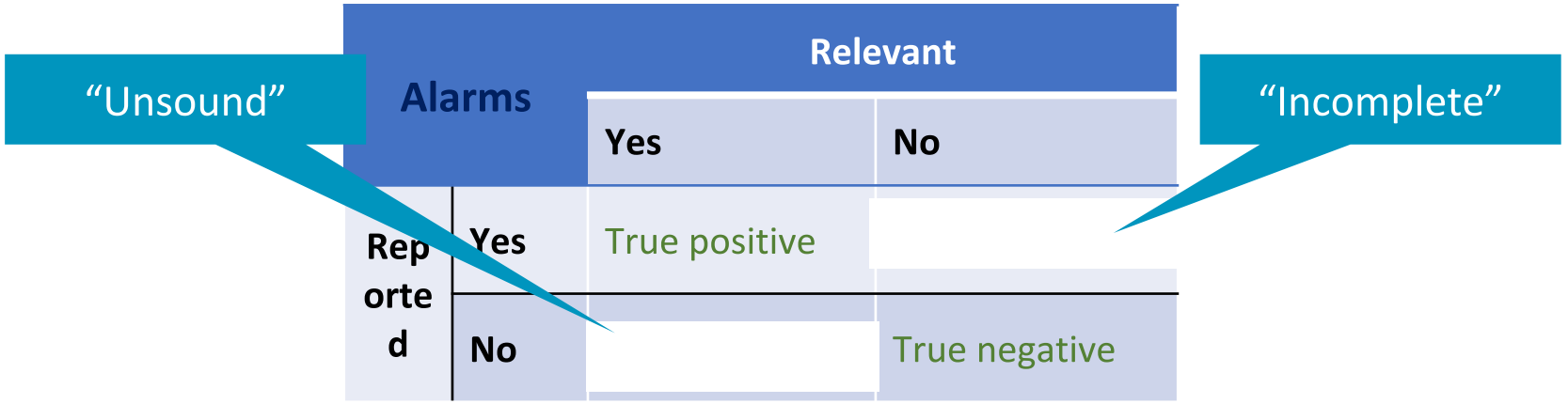
The most ironic thing here is that OpenSSL is open source software. Anyone could look at the code, and presumably hundreds did, but nobody noticed the fairly elementary coding error.

https://gizmodo.com/how-heartbleed-works-the-code-behind-the-internets-se-1561341209

Why did program analysis tools not discover the Heartbleed bug?

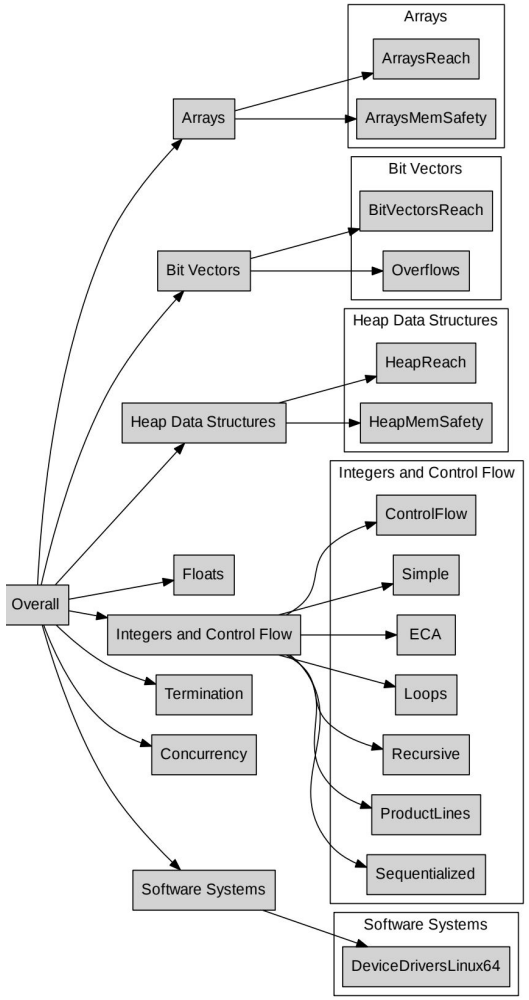
Approximations in Program Analysis

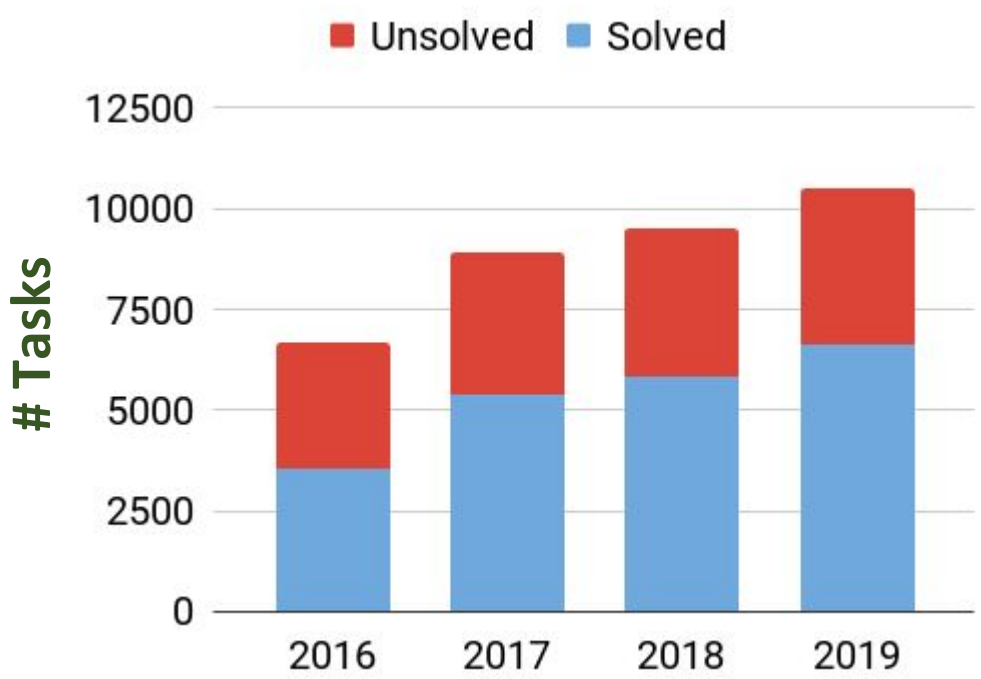
> "… can be difficult to do without introducing large numbers of **false positives**, or scaling **performance** exponentially poorly. In this case, **balancing these** and other factors in the analysis design caused us to miss the defect."
>
> **— Coverity, On Detecting Heartbleed with Static Analysis, 2014**



State of the Practice: Verification

SV-COMP: International Software Verification Competition





Task	# KLOC
linux-4.2-rc1.tar.xz-32_7a-drivers	228
linux-4.2-rc1.tar.xz-08_1a-drivers	205
eca-programs/ Problem103_label59	185
eca-programs/ Problem103_label58	185
eca-programs/ Problem103_label57	185

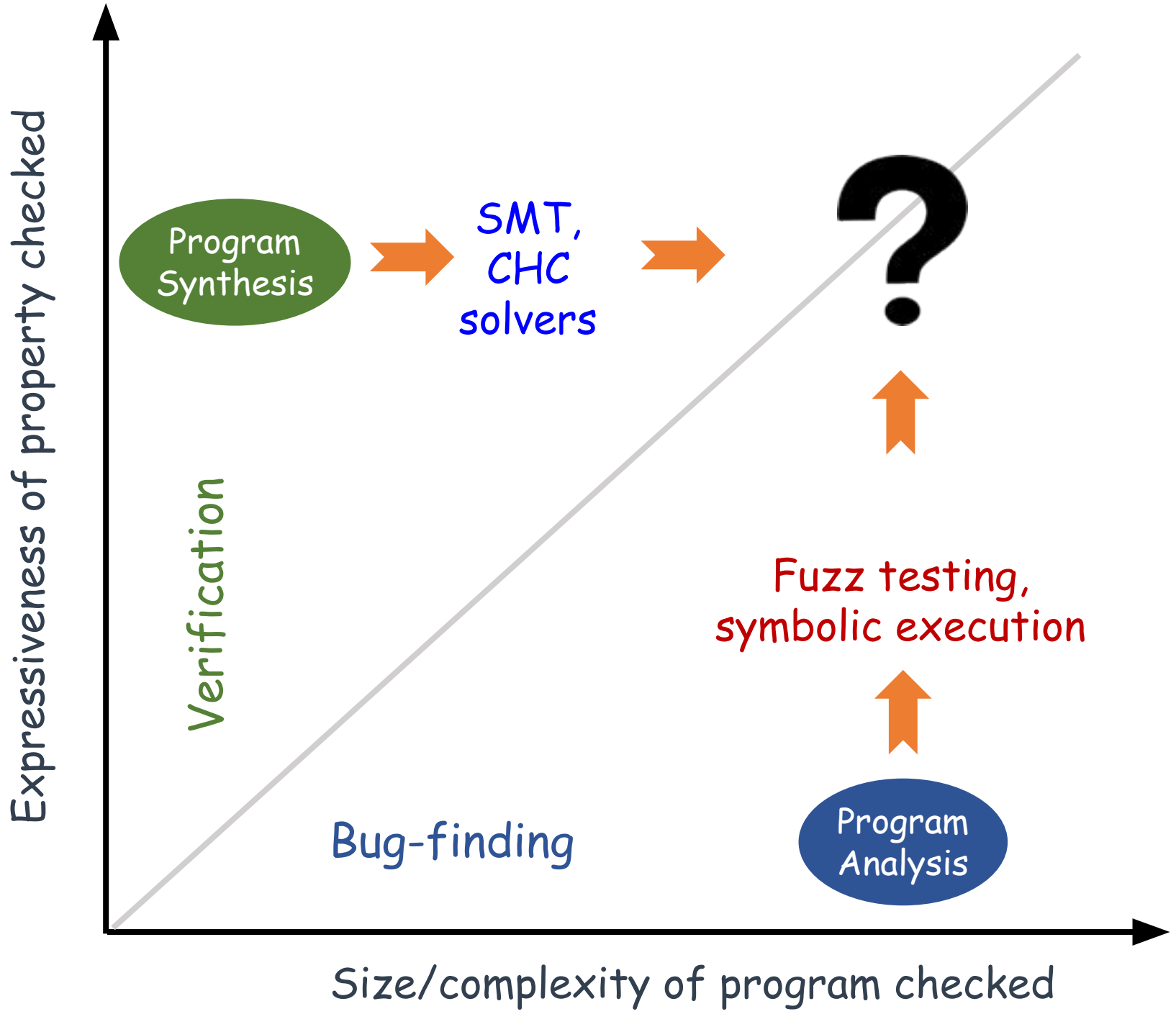
(top 5 solved tasks by size)

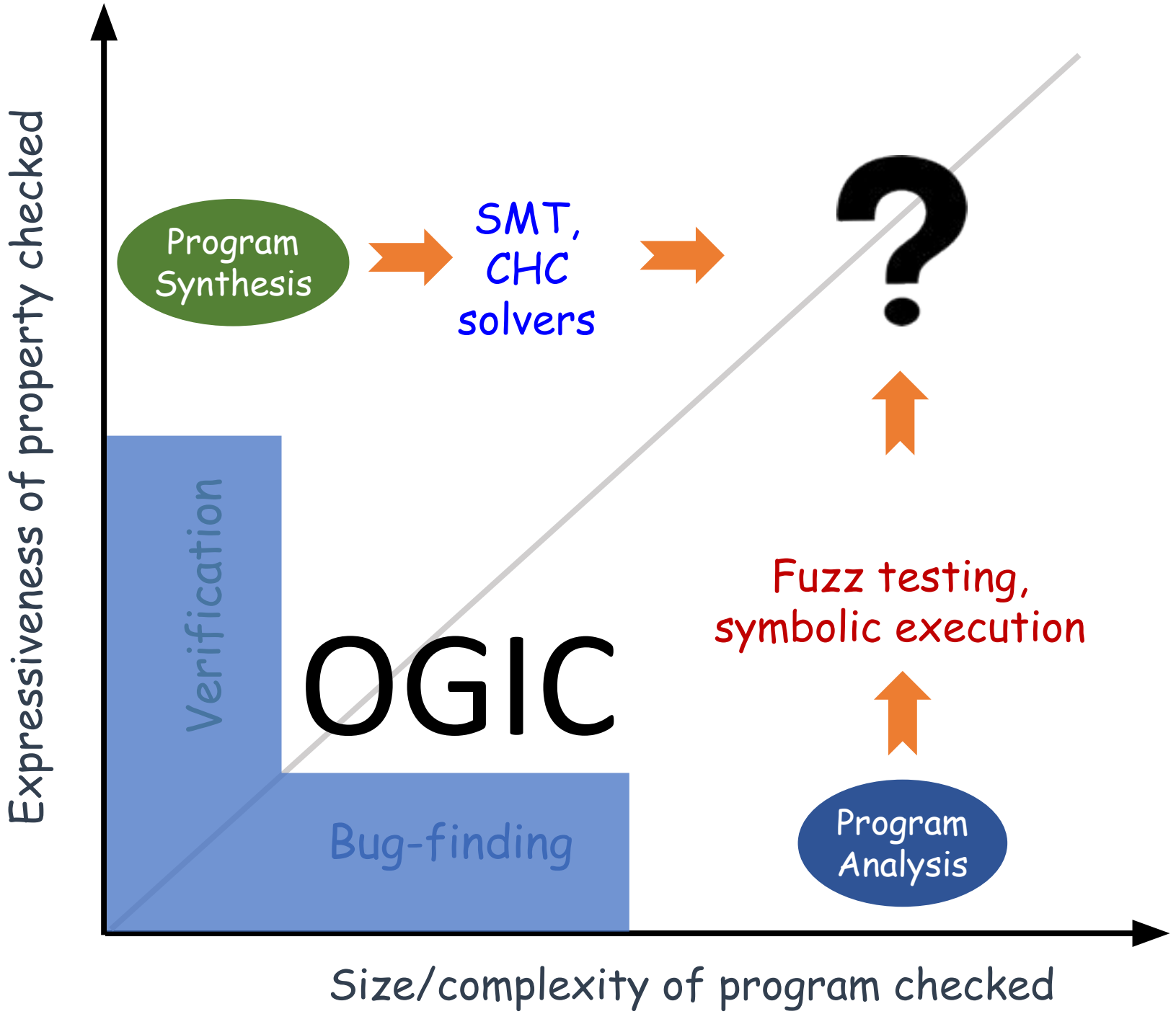
A Challenge in Verification

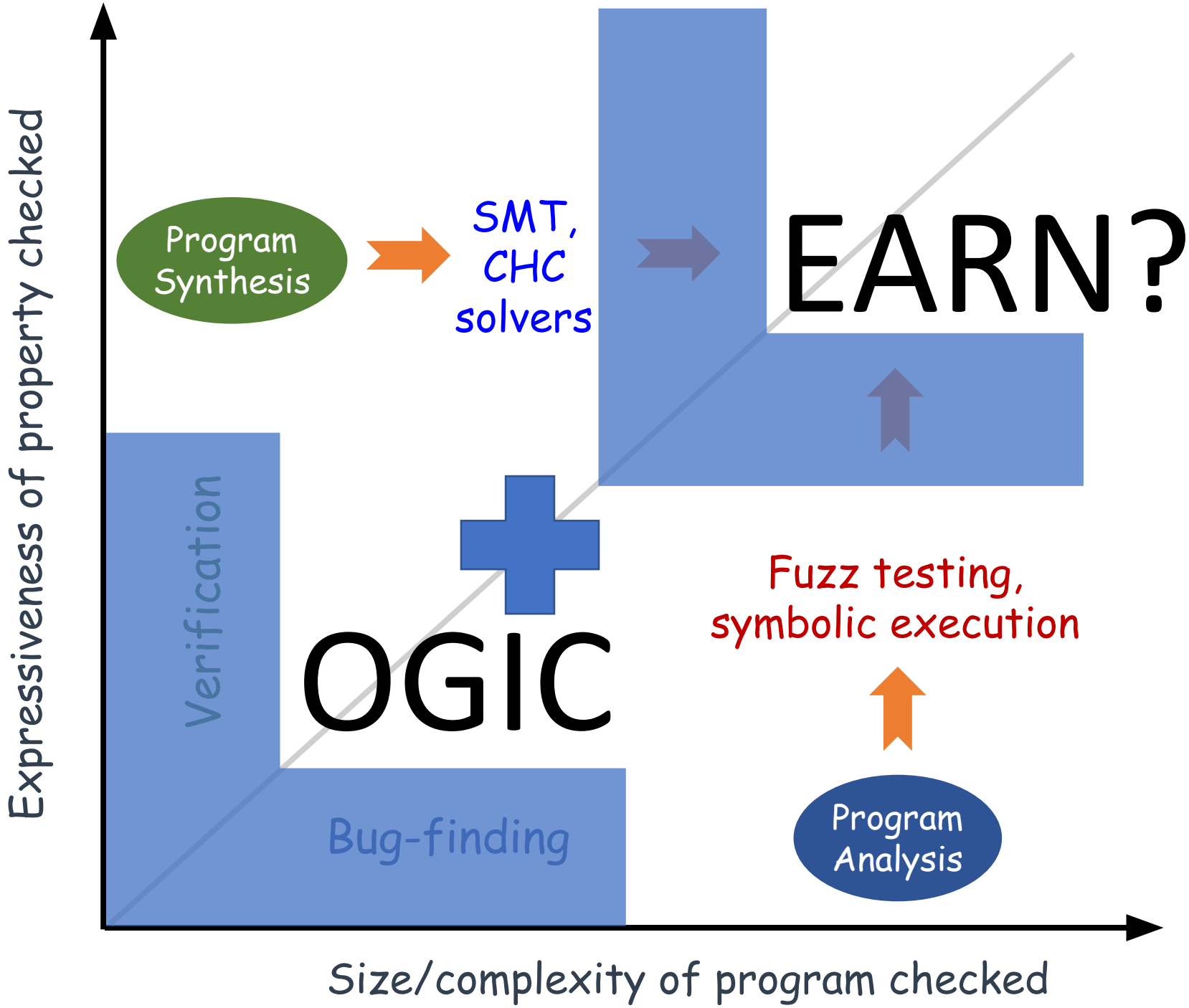
```
1  procedure main(n : int)
2  {
3    var x : int, m : int;
4
5    x := 0;
6    m := 0;
7
8    while (x < n)
9      invariant m <= 0 || (x > 0 && n > m);
10   {
11       if (*) {
12           m := x;
13       }
14       x := x + 1;
15   }
16   if (n > 0) {
17       assert(m < n);
18   }
19 }
```

Task	# LOC
sorting_bubblesort_false-unreach-call_ground	31
array_mul_init_true-unreach-call_true-termination	33
array_shadowinit_true-unreach-call_true-termination	33
nr5_true-unreach-call	44
revcpyswp2_true-unreach-call	52

(smallest 5 unsolved tasks)


Expressiveness of property checked
Size/complexity of program checked
Program Synthesis
SMT, CHC solvers
?
Verification
Fuzz testing, symbolic execution
Bug-finding
Program Analysis


Expressiveness of property checked
Size/complexity of program checked
Program Synthesis
SMT, CHC solvers
?
Verification
OGIC
Bug-finding
Fuzz testing, symbolic execution
Program Analysis


Expressiveness of property checked
Size/complexity of program checked
Program Synthesis
SMT, CHC solvers
EARN?
Verification
OGIC
Bug-finding
Fuzz testing, symbolic execution
Program Analysis

Talk Outline

- Motivation
- Learning for Bug-Finding
- Learning for Verification

- User-Guided Program Reasoning Using Bayesian Inference. **PLDI 2018**.
- Continuously Reasoning about Programs using Differential Bayesian Inference. **PLDI 2019**.

- Learning Loop Invariants for Program Verification. **NeurIPS 2018**.
- Learning a Meta-Solver for Syntax-Guided Program Synthesis. **ICLR 2019**.

Talk Outline

- Motivation
- Learning for Bug-Finding
- Learning for Verification

A Static Analysis in Datalog

Analysis inputs:

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

Analysi

paral

Analysis

paralle

parallel(p1, p2) :- pa

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

…

progr
immedi

p
happ

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?

…50+ pages!

vs.

```
### 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).
```

- Fewer bugs
- Extensible
- Runs faster

Applying the Analysis to a Program

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

8  public Request getRequest() {
9        return request;
10 }
```

R1

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

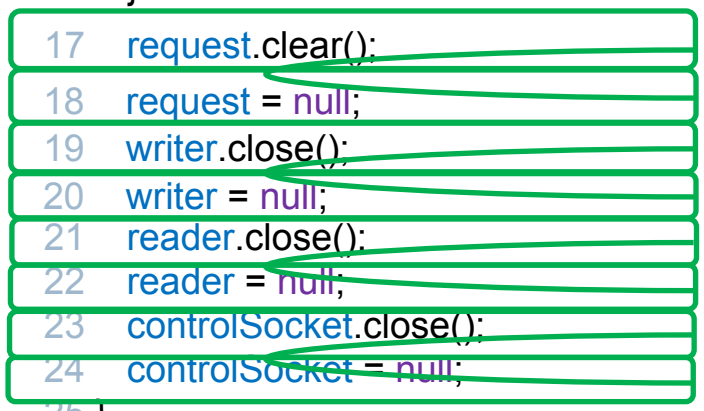
Code snippet of concurrent program Apache FTP Server

Applying the Analysis to a Program

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

8  public Request getRequest() {
9        return request;
10 }
```

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



Code snippet of concurrent program Apache FTP Server

Applying the Analysis to a Program

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

8  public Request getRequest() {
9         return request;
10 }
```

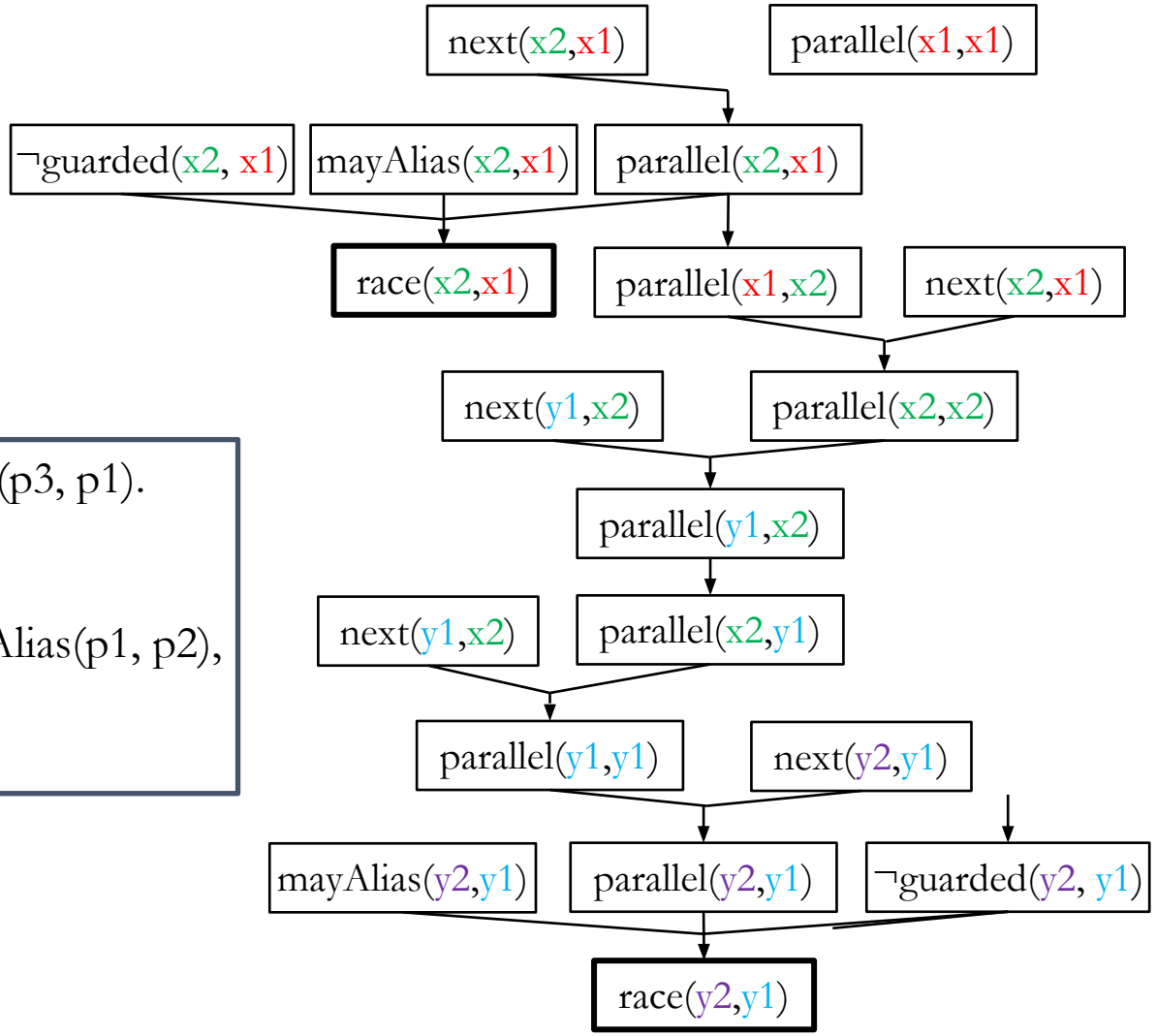
```
11 public void close() {
12    synchronized (this) {
13     if (isClosed)
14         return;
15     isClosed = true;
16    }
17    request.clear();        // x1
18    request = null;         // x2
19    writer.close();         // y1
20    writer = null;          // y2
21    reader.close();
22   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

Code snippet of concurrent program Apache FTP Server

How Does Datalog Work?

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

```
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).
…
```



How To Go From This …

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

… To This?

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

An Idea: Mixed Hard and Soft Rules

Analysis inputs:

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

Analysis outputs:

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

```
if (num_threads == 1) { // p1
    x := x + 1          // p3
}
```

```
x := x + 1   // p2
```

Analysis rules:

parallel(p3, p2) :- parallel(p1, p2), next (p3, p1). **prob. 0.9**

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

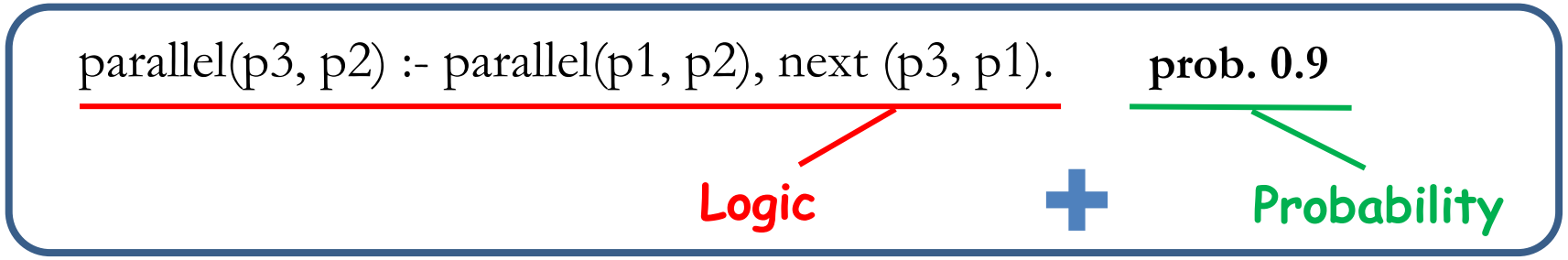
race(p1, p2) :- parallel(p1, p2), mayAlias(p1, p2), ¬guarded(p1, p2).

…

"Soft" Rule

"Hard" Rule

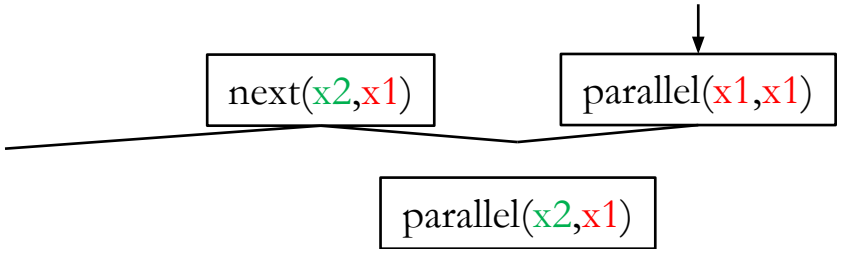
A Long History



- 1988: Bayesian Networks [Pearl]
- 1996: Stochastic Logic Programs (SLP) [Muggleton]
- 1999: Probabilistic Relational Models (PRM) [Koller]
- 2005: Bayesian Logic (BLOG) [Milch et al.]
- 2006: Markov Logic Network (MLN) [Richardson & Domingos]
- 2007: Probabilistic Prolog (ProbLog) [De Raedt et al.]

…

From Derivation Trees to Bayesian Networks

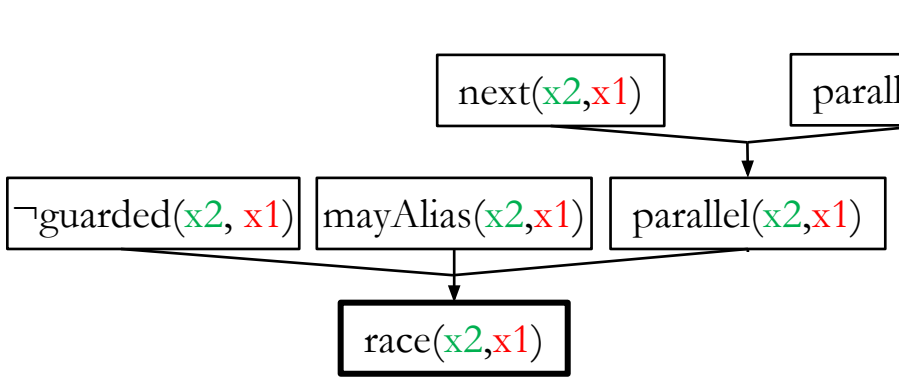


parallel(p3,p2) :- parallel(p1,p2), next(p3,p1). **prob. 0.9**

parallel(x1,x1)	next(x2,x1)	P(parallel(x2,x1) | parallel(x1,x1), next(x2,x1))
True	True	0.9
True	False	0
False	True	0
False	False	0

parallel(x2,x1) may only hold if parallel(x1,x1) and next(x2,x1) are true.

Marginal Inference in Bayesian Networks



Marginal inference performed using off-the-shelf solvers (LibDAI, Dlib, Infer.Net, etc.)

P(race(x2,x1)) = P(race(x2,x1), ¬guarded(x2,x1), mayAlias(x2,x1), parallel(x2,x1))

= P(race(x2,x1) | ¬guarded(x2,x1), mayAlias(x2,x1), parallel(x2,x1)) * P(¬guarded(x2,x1)) * P(mayAlias(x2,x1)) * P(parallel(x2,x1))

+ P(race(x2,x1), ¬guarded(x2,x1), mayAlias(x2,x1), ¬parallel(x2,x1))

+ P(race(x2,x1), ¬guarded(x2,x1), ¬mayAlias(x2,x1), parallel(x2,x1))

= P(race(x2,x1) | ¬guarded(x2,x1), mayAlias(x2,x1), parallel(x2,x1)) * P(¬guarded(x2,x1)) * P(mayAlias(x2,x1)) * P(parallel(x2,x1))

+ ...

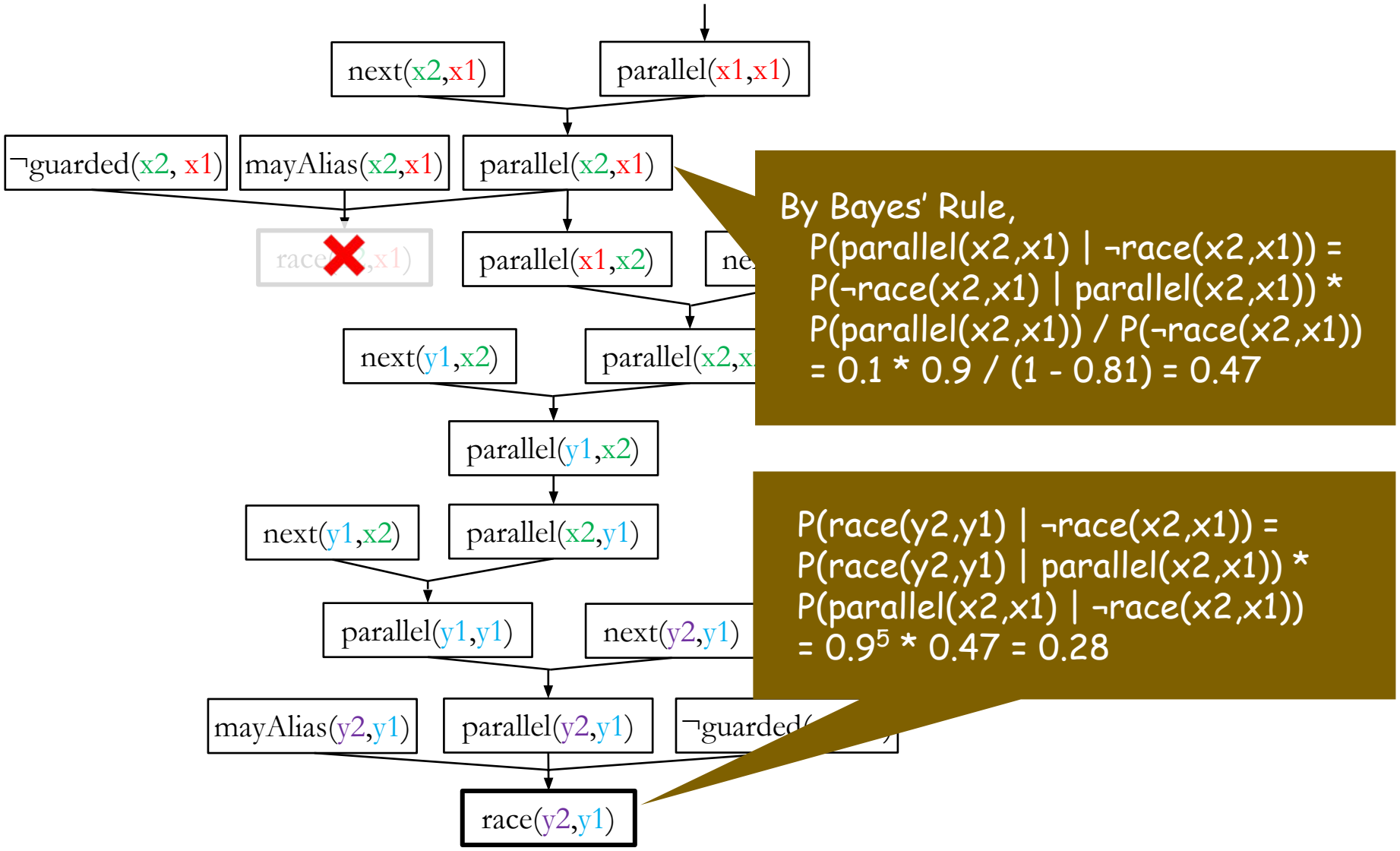
+ P(race(x2,x1), guarded(x2,x1), ¬mayAlias(x2,x1), ¬parallel(x2,x1))

0. If any of the antecedents fail, then the race cannot happen.

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







next(x2,x1)
parallel(x1,x1)
¬guarded(x2, x1)
mayAlias(x2,x1)
parallel(x2,x1)
parallel(x1,x2)
next(y1,x2)
parallel(y1,x2)
next(y1,x2)
parallel(x2,y1)
parallel(y1,y1)
next(y2,y1)
mayAlias(y2,y1)
parallel(y2,y1)
race(y2,y1)
By Bayes' Rule,
P(parallel(x2,x1) | ¬race(x2,x1)) =
P(¬race(x2,x1) | parallel(x2,x1)) *
P(parallel(x2,x1)) / P(¬race(x2,x1))
= 0.1 * 0.9 / (1 - 0.81) = 0.47
P(race(y2,y1) | ¬race(x2,x1)) =
P(race(y2,y1) | parallel(x2,x1)) *
P(parallel(x2,x1) | ¬race(x2,x1))
= 0.9^5 * 0.47 = 0.28

Confidenc	Detected Races	
0.81	**R2**: Race on field org.apache.ftpserver.RequestHandler.request	
0	org.apache.ftpserver.RequestHandler:17	org.apache.ftpserver.RequestHandler:18
0.53	**R3**: Race on field org.apache.ftpserver.RequestHandler.writer	
0.28	org.apache.ftpserver.RequestHandler:19	org.apache.ftpserver.RequestHandler:20
0.35	**R4**: Race on field org.apache.ftpserver.RequestHandler.reader	
0.18	org.apache.ftpserver.RequestHandler:21	org.apache.ftpserver.RequestHandler:22
0.30	**R1**: Race on field org.apache.ftpserver.RequestHandler.request	
0.30	org.apache.ftpserver.RequestHandler:9	org.apache.ftpserver.RequestHandler:18
0.23	**R5**: Race on field org.apache.ftpserver.RequestHandler.controlSo	
0.12	org.apache.ftpserver.RequestHandler:23	org.apache.ftpserver.RequestHandler:24

$P(R_i \mid \neg R2)$

Confidenc	Detected Races	
0	**R2**: Race on field org.apache.ftpserver.RequestHandler.request	
	org.apache.ftpserver.RequestHandler:17	org.apache.ftpserver.RequestHandler:18
0.28	**R3**: Race on field org.apache.ftpserver.RequestHandler.writer	
	org.apache.ftpserver.RequestHandler:19	org.apache.ftpserver.RequestHandler:20
0.18	**R4**: Race on field org.apache.ftpserver.RequestHandler.reader	
	org.apache.ftpserver.RequestHandler:21	org.apache.ftpserver.RequestHandler:22
0.30	**R1**: Race on field org.apache.ftpserver.RequestHandler.request	
	org.apache.ftpserver.RequestHandler:9	org.apache.ftpserver.RequestHandler:18
0.12	**R5**: Race on field org.apache.ftpserver.RequestHandler.controlS	
	org.apache.ftpserver.RequestHandler:23	org.apache.ftpserver.RequestHandler:24

$P(R_i \mid \neg R2)$

Experimental Setup

- **Analyses:**

58 input relations
44 output relations
102 rules

Race conditions checker

52 input relations
25 output relations
62 rules

Information flow checker

- **Programs:**

Concurrent Java programs

(~ 50-550 KB in size)

Symentec Android apps

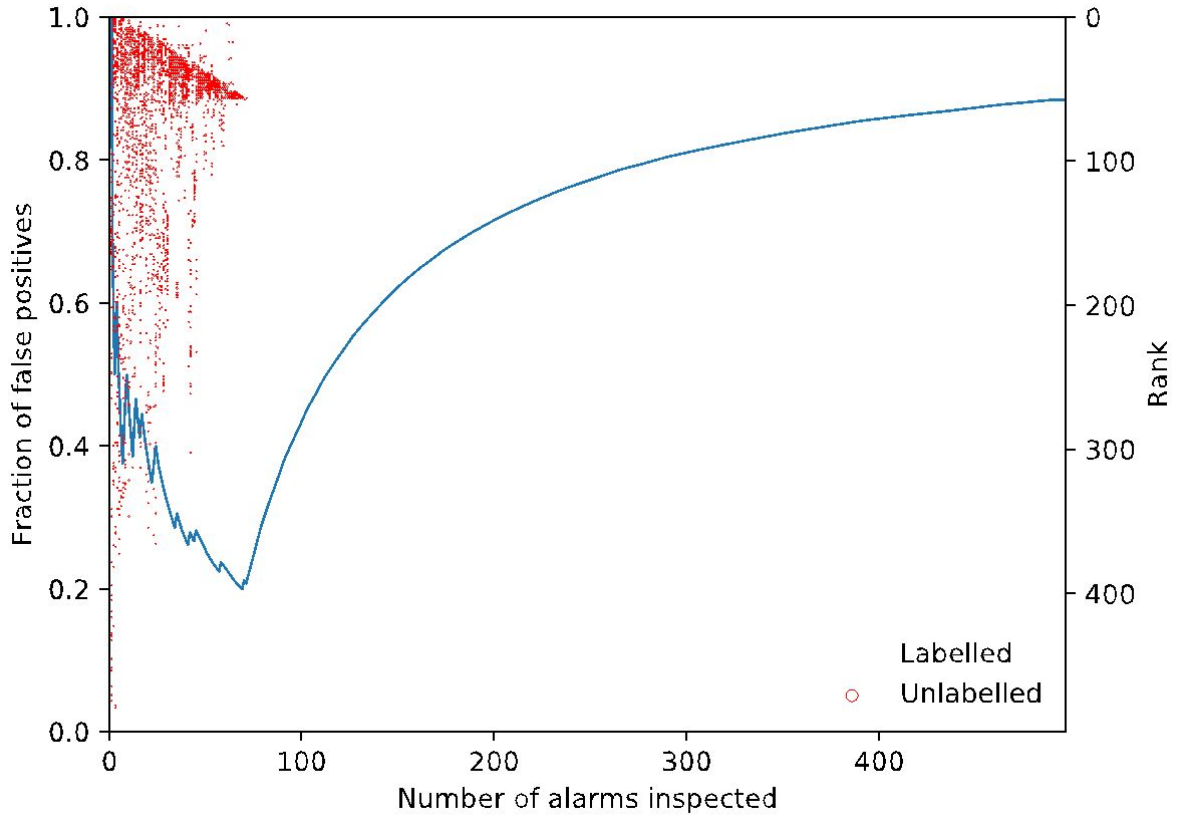
(~ 68-81 KB in size)

Empirical Results

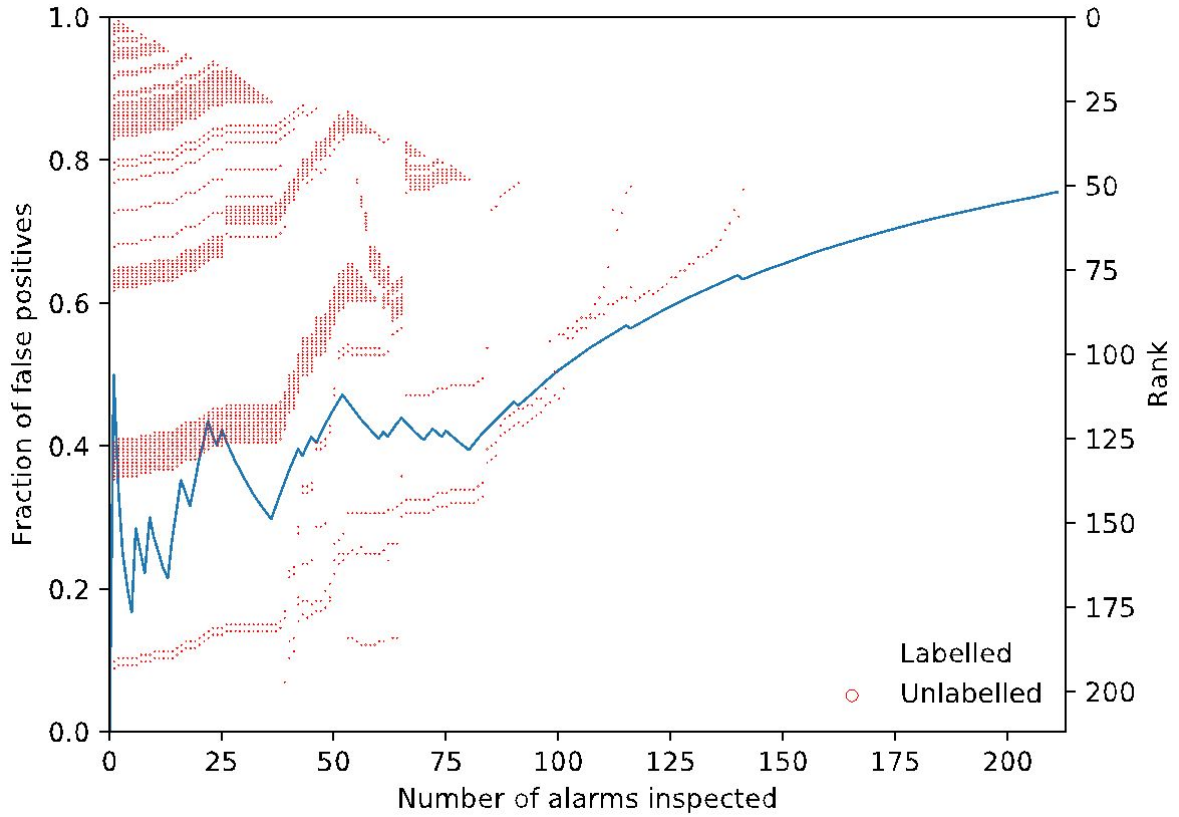
.90 - 1	=	**excellent**	**(A)**
.80 - .90	=	**good**	**(B)**
.70 - .80	=	**fair**	**(C)**
.60 - .70	=	**poor**	**(D)**
.50 - .60	=	**fail**	**(F)**

	Graph size		Alarms		FP rate	AUC
	Tuples	Clauses	Total	Bugs		
Race conditions checker						
weblech	2.5K	1.5K	188	55	71%	0.88
hedc	12K	10K	152	9	94%	0.71
jspider	45K	45K	257	7	97%	0.87
ftpserver	110K	112K	522	75	86%	0.97
Information flow checker						
AndorsTrail	2.7K	3.2K	156	7	96%	0.99
kQm-LO	12K	18K	817	160	81%	0.94
gingermaster	15K	20K	437	87	80%	0.88
iNJ-Cw	17K	24K	1,012	248	76%	0.91

Ranking Quality: Race Conditions Checker



Ranking Quality: Information Flow Checker



Taxonomy of Research Directions

Balancing Analysis Tradeoffs

- Analysis Accuracy vs. Soundness
- Analysis Accuracy vs. Cost

Tailoring Analysis Results

- Unguided vs. Interactive
- Batch vs. Continuous Reasoning
- Alarm Clustering vs. Ranking

Analysis Specification and Implementation

- Synthesizing Analyses from Data
- Expressiveness of Analysis Language
- Capabilities of Analysis Solvers

More details in SAS 2019 paper “Rethinking Static Analysis by Combining Discrete and Continuous Reasoning”

Talk Outline

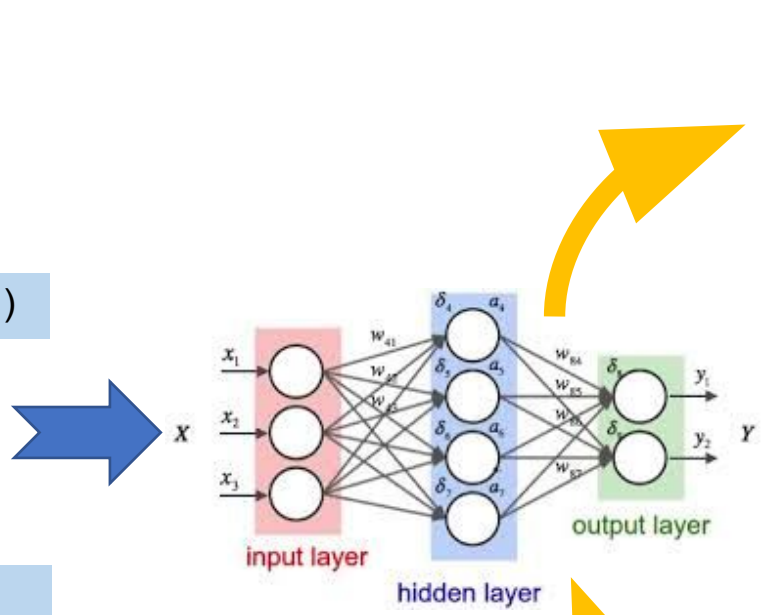
- Motivation
- Learning for Bug-Finding
- Learning for Verification

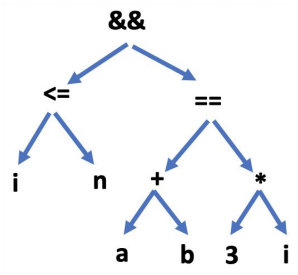
Example: Loop Invariants

```
void main(int n) {
    int x = 0; int m = 0;
    human expert:
    (m == 0 || m < x) && (n <= 0 || x <= n)
    generated:
    (m <= 0 || x > 0) && (m <= 0 || n > m)
    while (x < n) {
        if (*) { m = x; }
        x = x + 1;
    }
    if (n > 0) assert(m < n);
}
```

Architecture of code2inv

```
assume (a == 1 /\ b == 1)
while (b < 1000) {
    a = a + b;
    b = b + 1;
}
assert (a >= 1600)
```





Z3
Theorem Prover

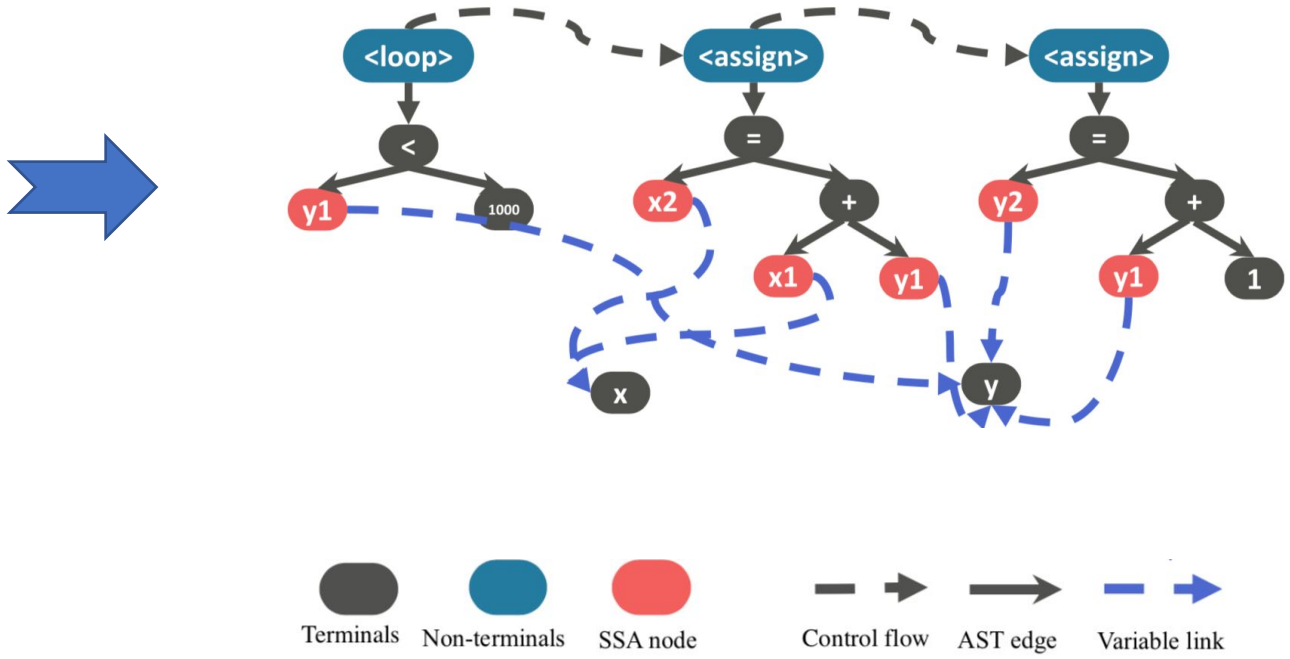
PROOF

Counter-
Examples

Step 1: Representing Program as Graph

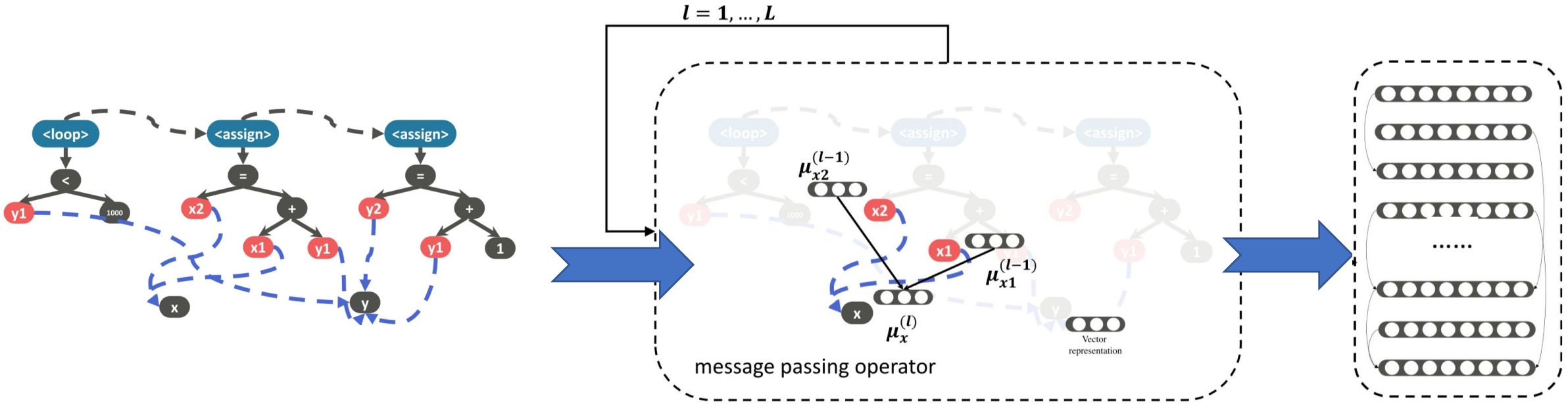
- Encode the program as a graph that captures its rich structure

```
while (y < 1000) {
    x = x + y;
    y = y + 1;
}
```



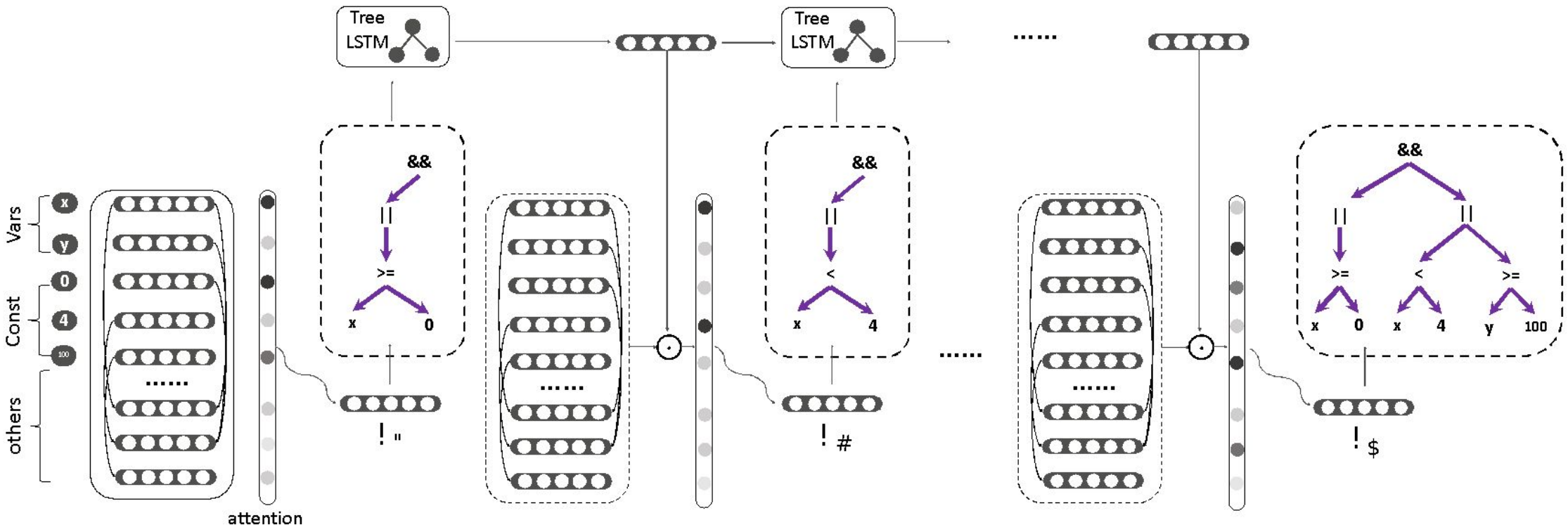
Step 2: Converting Graph to Vector

- Convert the graph to a vector representation using a graph neural network



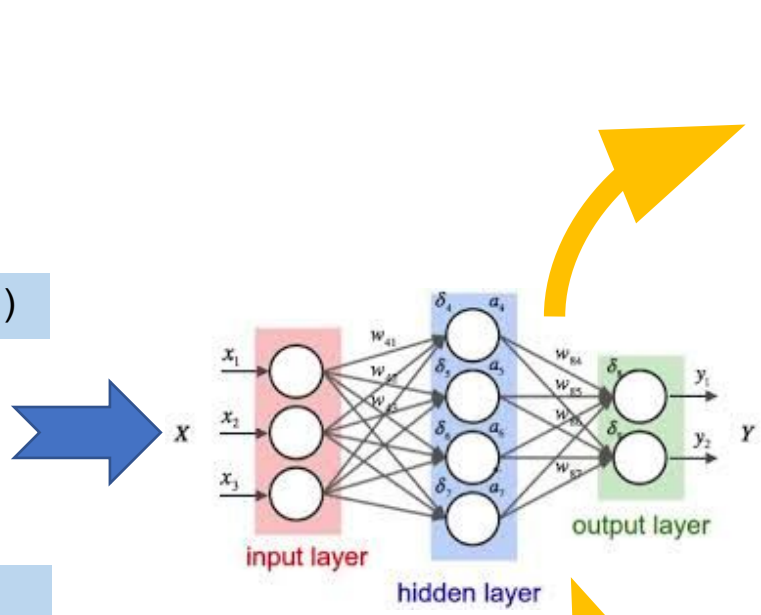
Step 3: Predicting Loop Invariant

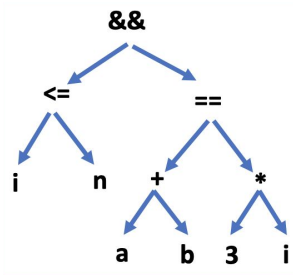
- Model loop invariant generation as a multi-step decision making process



Architecture of code2inv

```
assume (a == 1 /\ b == 1)
while (b < 1000) {
    a = a + b;
    b = b + 1;
}
assert (a >= 1600)
```



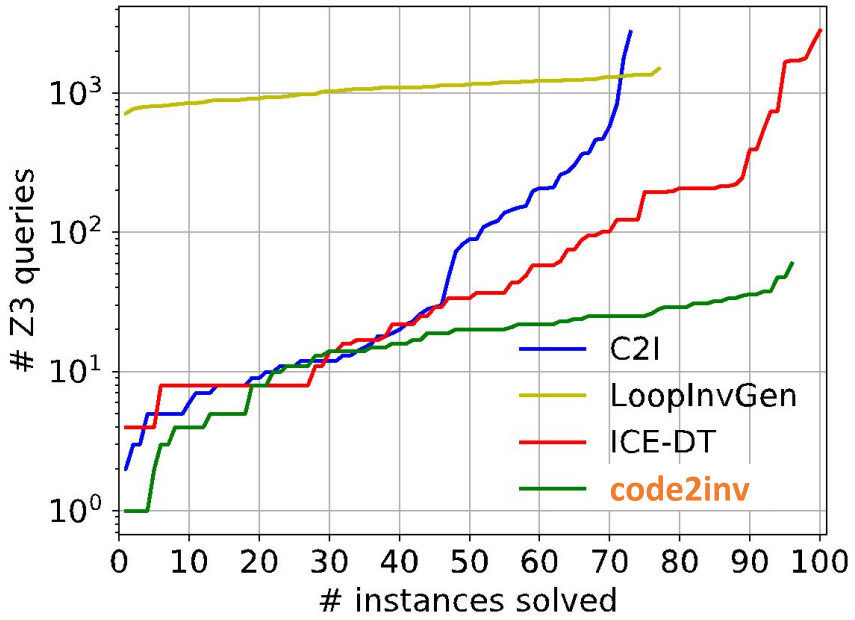


Z3
Theorem Prover

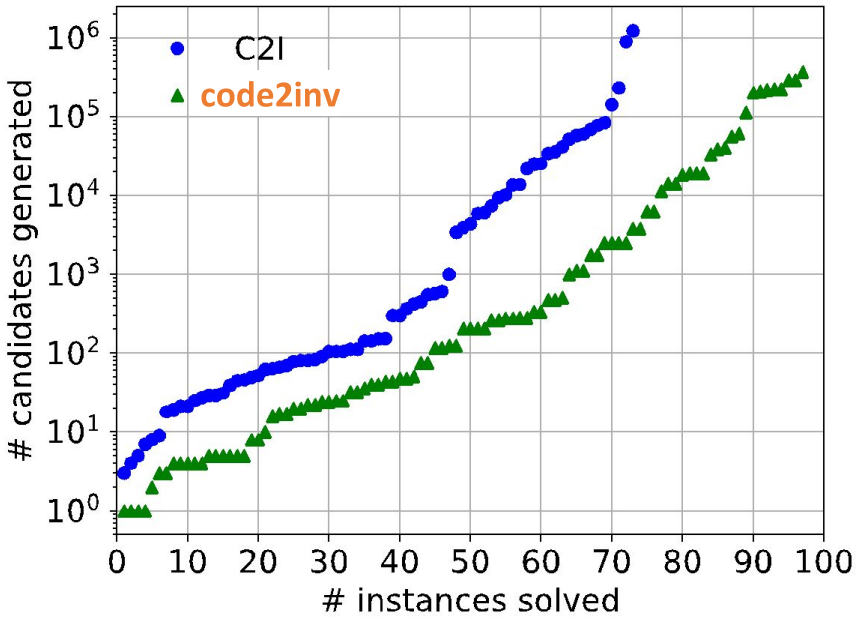
PROOF

Counter-
Examples

Comparison to State-of-the-Art



Verification Cost



Sample Complexity

Other Applications

- Verification: Program -> Invariant
- Bug-Finding: Program -> Counterexample
- Repair: Program -> Edit Sequence

Example: Bug Detection and Repair for JS

2 helpers.js

```
@@ -5,7 +5,7 @@ exports.parseTitle = title => {
        if (matches) {
                return {
                        episode: Number(matches.groups.episode),
-                       hosts: matches.groups.hosts.split(/([,&]+|\sand\s)/).map(el => S(el).trim().s)
                };
        }
        return false;
```

```
        if (matches) {
                return {
                        episode: Number(matches.groups.episode),
+                       hosts: matches.groups.hosts.split(/[,&]+|\sand\s/).map(el => S(el).trim().s)
                };
        }
        return false;
```

Intended Goal:
Split a string based on delimiter that matches regex: `[,&]+|\sand\s`

Input: `" and "`

Output of buggy code:
`[ '', ' and ', '' ]`

Output of fixed code:
`[ '', '' ]`

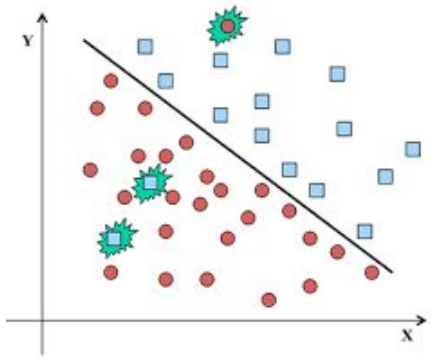
```
const S = require ('string');

exports.parseTitle = title => {
        const matches = title.match (/ Joe Rogan Experience # (? <episode> \ d *) (\ s? [-] {0,} \ s?) (? <hosts>. *?) (? <part ?> \ s \ (share \ s \ d {1} \)) $ / i);
        if (matches) {
                return {
                        episode: Number (matches.groups.episode),
                        hosts: matches.groups.hosts.split (/ [, &] + | \ sand \ s /). map (el => S (el) .trim (). s)
                };
        }
        return false;
};
```

Limits of Training Data

Stats of Data Crawled from Github per Week:

Downloaded JS files:	9,425,472
Valid AST files and diffs:	4,712,736
ASTs with a single diff:	585,984
Valid data points:	47,040



Sampling 50 data points in test set reveals
21 real bugs and 29 non-bugs

Conclusions

- Logical Reasoning: Discrete -> Continuous
- Which machine learning models worked?
  - Bug-finding: Bayesian networks, MLNs, SLPs, …
    - Relies on good human-engineered features
  - Verification: graph neural network
    - Suitable program representation is critical
- Challenge: How to obtain training data?
  - Bug-finding: supervised learning
    - Leverage continuously growing open-source datasets: OSS-Fuzz, GHArchive, …
  - Verification: reinforcement learning
    - Leverage formal methods tools evolved over decades: SMT/CHC solvers, Coq, …