Combining the Logical and the Probabilistic in Program Analysis

Xin Zhang          Xujie Si          Mayur Naik
University of Pennsylvania
What is Program Analysis?

int f(int i) {
...
}

Program Analysis

x may be null!

x.f = y;

... assert(n > 10);
... Pre: i > 0
Post: ret=sqrt(i)
Conventional Logical Approach

+ Easy to specify

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

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

\begin{align*}
\text{path}(a, a). \\
\text{path}(a, c) & :\text{ path}(a, b), \text{edge}(b, c).
\end{align*}

```
int f(int i) {
    ...
    x.f = y;
    ...
}
```

x may be null when i = 5.
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);
  ...
}

∀ i
Conventional Logical Approach

+ Easy to specify
+ Explainable
+ With formal guarantees

- Unable to handle uncertainty

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

Will this buffer overflow lead to a security exploit?

*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
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

Program Analysis

Bug Detection

Verification

Pointer Analysis
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

+ 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

Logic
- Encodes Formal Semantics

Probability
- Handles Uncertainty

Stochastic Logic Programs (SLP)
[Richardson & Domingos, 2006]

Probabilistic Relational Models (PRM)
[Koller, 1999]

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

Markov Logic Network (MLN)
[Koller, 1999]

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

…
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

Solution:
path(1, 1) = T, path(2, 2) = T,
path(1, 2) = T, path(2, 1) = F,
...

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

MaxSAT

Hard constraints:
edge(1, 7) ∧ edge(4, 7) ∧ ...
path(1, 1) ∧ path(2, 2) ∧ ...
(path(1, 1) ∨ ¬path(1, 1) ∨ ¬edge(1, 1)) ∧
(path(1, 2) ∨ ¬path(1, 1) ∨ ¬edge(1, 2)) ∧
(path(1, 2) ∨ ¬path(1, 2) ∨ ¬edge(2, 2)) ∧
...

Soft constraints:
(¬path(1, 1) weight 5) ∧
(¬path(1, 2) weight 5) ∧
...
Our Overall Methodology

Offline Learning → Markov Logic Network → Online Inference

EXE FILEs
Talk Outline

- Motivation
- A General Approach
- Instance Applications
- Solver
- Empirical Results
## Detected Races

<table>
<thead>
<tr>
<th>Race</th>
<th>Field</th>
<th>Line 1</th>
<th>Line 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R1</strong></td>
<td><code>org.apache.ftpserver.RequestHandler.request</code></td>
<td>9</td>
<td>18</td>
</tr>
<tr>
<td><strong>R2</strong></td>
<td><code>org.apache.ftpserver.RequestHandler.request</code></td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td><strong>R3</strong></td>
<td><code>org.apache.ftpserver.RequestHandler.writer</code></td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td><strong>R4</strong></td>
<td><code>org.apache.ftpserver.RequestHandler.reader</code></td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td><strong>R5</strong></td>
<td><code>org.apache.ftpserver.RequestHandler.controlSocket</code></td>
<td>23</td>
<td>24</td>
</tr>
</tbody>
</table>

## Eliminated Races

<table>
<thead>
<tr>
<th>Race</th>
<th>Field</th>
<th>Line 1</th>
<th>Line 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>E1</strong></td>
<td><code>org.apache.ftpserver.RequestHandler.isClosed</code></td>
<td>13</td>
<td>15</td>
</tr>
</tbody>
</table>
### Detected Races

<table>
<thead>
<tr>
<th>Race</th>
<th>Field</th>
<th>Line 1</th>
<th>Line 2</th>
</tr>
</thead>
</table>

### Eliminated Races

<table>
<thead>
<tr>
<th>Race</th>
<th>Field</th>
<th>Line 1</th>
<th>Line 2</th>
</tr>
</thead>
</table>
### Detected Races

<table>
<thead>
<tr>
<th>R1: Race on field org.apache.ftpserver.RequestHandler.request</th>
</tr>
</thead>
</table>

### Eliminated Races

<table>
<thead>
<tr>
<th>E1: Race on field org.apache.ftpserver.RequestHandler. isClosed</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>E2: Race on field org.apache.ftpserver.RequestHandler.request</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.ftpserver.RequestHandler: 17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>E3: Race on field org.apache.ftpserver.RequestHandler.writer</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>E4: Race on field org.apache.ftpserver.RequestHandler.reader</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>E5: Race on field org.apache.ftpserver.RequestHandler.controlSocket</th>
</tr>
</thead>
</table>
## Instance Applications

**[CAV’17 Invited Tutorial]**

<table>
<thead>
<tr>
<th></th>
<th><strong>Hard Rules</strong></th>
<th><strong>Soft Rules</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Static Bug Detection</strong> [FSE’15]</td>
<td>analysis rules</td>
<td>analysis rule_i weight w_i confidence of writer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>¬ result_j weight w_j confidence of user</td>
</tr>
<tr>
<td><strong>Automated Verification</strong> [PLDI’14, POPL’16]</td>
<td>analysis rules</td>
<td>¬ result_i weight w_i query resolution award</td>
</tr>
<tr>
<td></td>
<td>abstraction_1 ⊕ … ⊕ abstraction_n</td>
<td>abstraction_j weight w_j abstraction cost</td>
</tr>
<tr>
<td><strong>Interactive Verification</strong> [OOPSLA’17]</td>
<td>analysis rules</td>
<td>¬ cause_i weight w_i cost of inspection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>result_j weight w_j reward of resolution</td>
</tr>
</tbody>
</table>
### An Example: Datarace Analysis

**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)`.
- ...

*Program points p1 & p2 may have a datarace.*

*Program points p1 & p2 may happen in parallel.*

*If p1 & p2 may happen in parallel, and they may access the same memory location, then they may have a datarace.*

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

*If p1 & p2 may happen in parallel, and p3 is successor of p1, then p3 & p2 may happen in parallel.*
An Example: Datarace Analysis

**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*

```
a = 1;
if (a > 2) { // p1
    ... // p2
}
```
An Example: Datarace Analysis

```java
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;
        }
        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;
    }

    public void close() {
        synchronized (this) {
            if (isClosed)
                return;
            isClosed = true;
        }

        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;
    }

datarrace:

    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
```
How Does Generalization Work?

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

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?

```
17  request.clear();  // x1
18  request = null;   // x2
19  writer.close();  // y1
20  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).
...
```
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?

17  request.clear();  // x1
18  request = null;  // x2
19  writer.close();  // y1
20  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).

…

¬race(x2, x1) weight 25
next(x2,x1)
parallel(x1,x1)

¬guarded(x2, x1) mayAlias(x2,x1) parallel(x2,x1)

race(x2,x1) parallel(x1,x2)
next(x2,x1)

next(y1,x2) parallel(x2,x2)

parallel(y1,x2)
next(y1,x2)
parallel(x2,y1)

parallel(y1,y1) next(y2,y1)

mayAlias(y2,y1) parallel(y2,y1) ¬guarded(y2, y1)

race(y2,y1)
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

<table>
<thead>
<tr>
<th>Existing Solvers</th>
<th>Soundness</th>
<th>Optimality</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuffy [VLDB’11]</td>
<td>✘</td>
<td>✘</td>
<td>✓</td>
</tr>
<tr>
<td>Alchemy [ML’06]</td>
<td>✘</td>
<td>✘</td>
<td>✓</td>
</tr>
<tr>
<td>CPI [UAI’08]</td>
<td>✓</td>
<td>✓</td>
<td>✘</td>
</tr>
<tr>
<td>RockIt [AAAI’13]</td>
<td>✓</td>
<td>✓</td>
<td>✘</td>
</tr>
<tr>
<td>Z3 [TACAS’08]</td>
<td>✓</td>
<td>✓</td>
<td>✘</td>
</tr>
</tbody>
</table>

Markov Logic Network

Solver

The Inference Problem
The Nichrome Solver

MLN instance → work set → candidate solution → checker

MaxSAT Solver → expanded work set → checker

Yes → sound & optimal solution
No → expanded work set
The Nichrome Solver

MLN instance

work set
candidate solution

MaxSAT Solver

Yes

Checker

No

expanded work set

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

Datalog
SQL
MaxSAT
ILP
SMT
...

sound & optimal solution
On Abstraction Refinement for Program Analyses in Datalog [PLDI 2014]
Scaling Relational Inference Using Proofs and Refutations [AAAI 2016]
The Nichrome Solver

MaxSAT instance

work set

candidate solution

Yes

MaxSAT Solver

Checker

No

expanded work set

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

Program Reasoning:

Query-Guided Maximum Satisfiability
[POPL 2016]

Information Retrieval:

Is Dijkstra most likely author of Structured Programming?

Sound & optimal solution

Datalog
SQL
MaxSAT
ILP
SMT
...
The Nichrome Solver

MaxSAT instance

candidate solution

MaxSAT Solver

checker

overestimates the effect of unexplored clauses

expanded work set

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
  - **Benchmarks:** 7 Java programs (130-200 KLOC each)
  - **Feedback:** Automated [Zhang et al., PLDI’14]

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

- **Details:**
  - 30 rules
  - 18 input relations
  - 18 output relations

  - 76 rules
  - 50 input relations
  - 42 output relations

  - 76 rules
  - 52 input relations
  - 42 output relations
## Benchmarks Characteristics

<table>
<thead>
<tr>
<th></th>
<th>classes</th>
<th>methods</th>
<th>bytecode (KB)</th>
<th>KLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>antlr</td>
<td>350</td>
<td>2.3K</td>
<td>186</td>
<td>131</td>
</tr>
<tr>
<td>avrora</td>
<td>1,544</td>
<td>6.2K</td>
<td>325</td>
<td>193</td>
</tr>
<tr>
<td>ftp</td>
<td>414</td>
<td>2.2K</td>
<td>118</td>
<td>130</td>
</tr>
<tr>
<td>hdec</td>
<td>353</td>
<td>2.1K</td>
<td>140</td>
<td>153</td>
</tr>
<tr>
<td>luindex</td>
<td>619</td>
<td>3.7K</td>
<td>235</td>
<td>190</td>
</tr>
<tr>
<td>lusearch</td>
<td>640</td>
<td>3.9K</td>
<td>250</td>
<td>198</td>
</tr>
<tr>
<td>weblech</td>
<td>576</td>
<td>3.3K</td>
<td>208</td>
<td>194</td>
</tr>
<tr>
<td>App 1</td>
<td>5</td>
<td>13</td>
<td>0.3</td>
<td>0.6</td>
</tr>
<tr>
<td>App 2</td>
<td>4</td>
<td>12</td>
<td>0.2</td>
<td>0.6</td>
</tr>
<tr>
<td>App 3</td>
<td>17</td>
<td>46</td>
<td>1.3</td>
<td>4.2</td>
</tr>
</tbody>
</table>

*Control Study*

*User Study*

*MAPL’17 8/5/17*
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

338 false and 700 true reports

- False reports eliminated
- True reports retained

5%: 60%
10%: 80%
15%: 80%
20%: 80%
## Solver Performance Results

<table>
<thead>
<tr>
<th></th>
<th>Nichrome</th>
<th>Eva500</th>
<th>Nichrome</th>
<th>Eva500</th>
<th>Nichrome</th>
<th>Eva500</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftp</td>
<td>16</td>
<td>11</td>
<td>16</td>
<td>1,262</td>
<td>29K</td>
<td>3M</td>
</tr>
<tr>
<td>hedc</td>
<td>23</td>
<td>21</td>
<td>181</td>
<td>1,918</td>
<td>400K</td>
<td>4.8M</td>
</tr>
<tr>
<td>weblech</td>
<td>4</td>
<td>timeout</td>
<td>363</td>
<td>timeout</td>
<td>922K</td>
<td>8.4M</td>
</tr>
<tr>
<td>antlr</td>
<td>190</td>
<td>timeout</td>
<td>1,405</td>
<td>timeout</td>
<td>3.3M</td>
<td>13M</td>
</tr>
<tr>
<td>avrora</td>
<td>178</td>
<td>timeout</td>
<td>1,095</td>
<td>timeout</td>
<td>2.6M</td>
<td>16.3M</td>
</tr>
<tr>
<td>chart</td>
<td>253</td>
<td>timeout</td>
<td>721</td>
<td>timeout</td>
<td>1.8M</td>
<td>22.3M</td>
</tr>
<tr>
<td>luindex</td>
<td>169</td>
<td>timeout</td>
<td>944</td>
<td>timeout</td>
<td>2.2M</td>
<td>11.9M</td>
</tr>
</tbody>
</table>

*Winner of the 2015 Annual MaxSAT Competition*
## Solver Performance Results

<table>
<thead>
<tr>
<th></th>
<th>running time (seconds)</th>
<th>peak memory (MB)</th>
<th># clauses (M = million)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nichrome</td>
<td>Eva500</td>
<td>Nichrome</td>
</tr>
<tr>
<td>ftp</td>
<td>16</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>hedd</td>
<td>23</td>
<td>21</td>
<td>181</td>
</tr>
<tr>
<td>webblech</td>
<td>4</td>
<td>timeout</td>
<td>363</td>
</tr>
<tr>
<td>antlr</td>
<td>190</td>
<td>timeout</td>
<td>1,405</td>
</tr>
<tr>
<td>avrora</td>
<td>178</td>
<td>timeout</td>
<td>1,095</td>
</tr>
<tr>
<td>chart</td>
<td>253</td>
<td>timeout</td>
<td>721</td>
</tr>
<tr>
<td>luindex</td>
<td>469</td>
<td>timeout</td>
<td>944</td>
</tr>
<tr>
<td>Advisor Recommendation</td>
<td>4</td>
<td>timeout</td>
<td></td>
</tr>
<tr>
<td>Entity Resolution</td>
<td>6</td>
<td>44</td>
<td>9K</td>
</tr>
<tr>
<td>Information Extraction</td>
<td>13</td>
<td>335</td>
<td>27K</td>
</tr>
</tbody>
</table>
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