Specification Inference for Explicit Information Flow Problems

Merlin

Benjamin Livshits, Aditya V. Nori, Sriram K. Rajamani
Microsoft Research

Anindya Banerjee
IMDEA Software

Mining Security Specifications

- Problem: Can we automatically infer which routines in a program are sources, sinks and sanitizers?

- Technology: Static analysis + Probabilistic inference

- Applications:
  - Lowers false errors from tools
  - Enables more complete flow checking

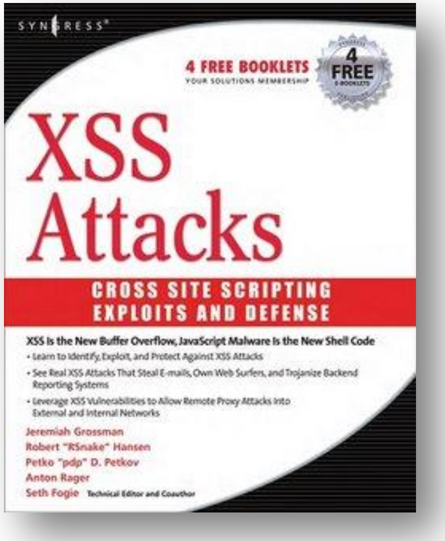
- Results:
  - Over 300 new vulnerabilities discovered in 10 deployed ASP.NET applications

Microsoft Research

Motivation

Static Analysis Tools for Security

- Web application vulnerabilities are a serious threat!



SYNGRESS®

4 FREE BOOKLETS
YOUR SOLUTIONS MEMBERSHIP

4 FREE E-BOOKLETS

XSS Attacks

CROSS SITE SCRIPTING EXPLOITS AND DEFENSE

XSS Is the New Buffer Overflow, JavaScript Malware Is the New Shell Code

- Learn to Identify, Exploit, and Protect Against XSS Attacks
- See Real XSS Attacks That Steal E-mails, Own Web Surfers, and Trojanize Backend Reporting Systems
- Leverage XSS Vulnerabilities to Allow Remote Proxy Attacks into External and Internal Networks

Jeremiah Grossman
Robert "RSnake" Hansen
Petko "pdp" D. Petkov
Anton Rager
Seth Fogie Technical Editor and Coauthor

FORTIFY

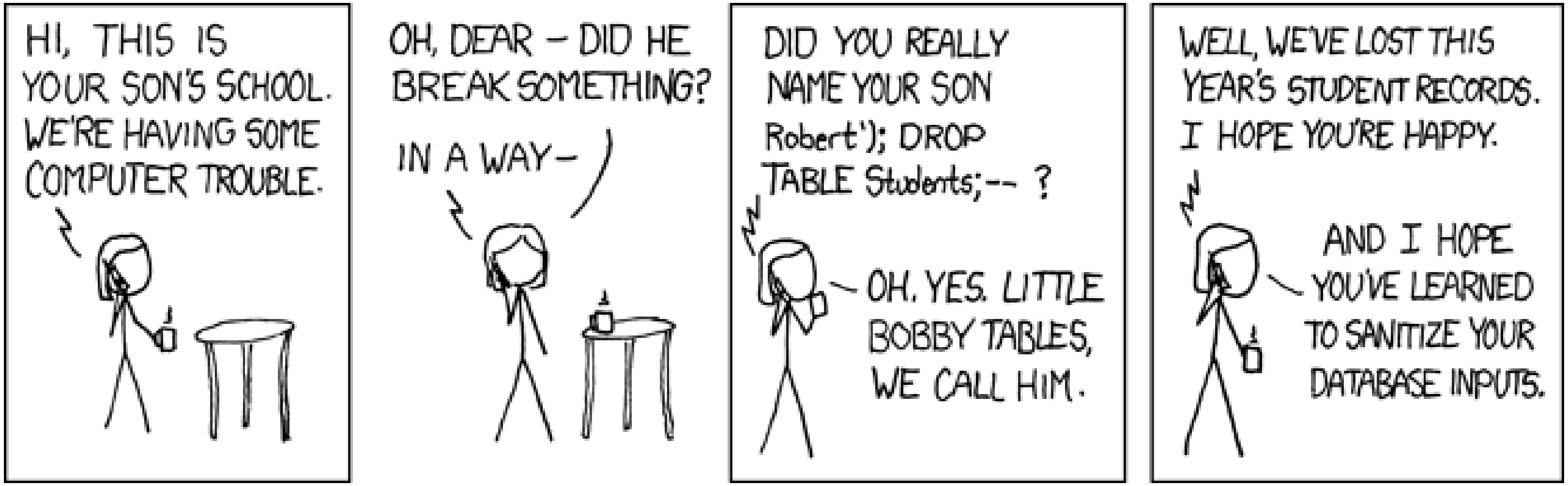
OUNCE LABS

Klocwork

Microsoft Visual Studio CAT.NET

Microsoft Research

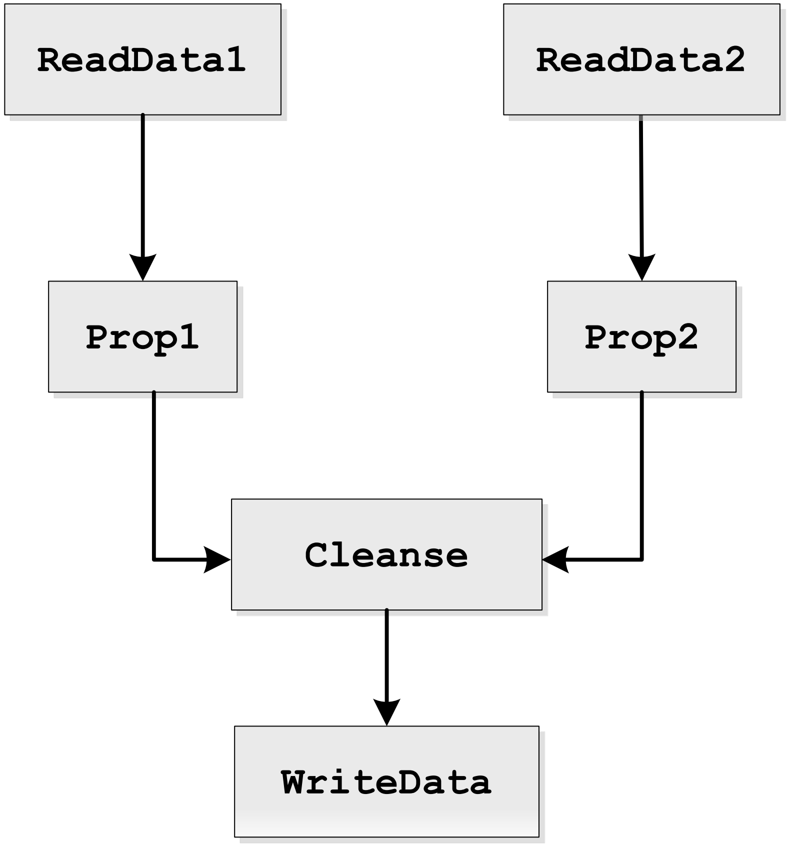
Web Application Vulnerabilities


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY –
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

```
$username = $_REQUEST['username'];
$sql = "SELECT * FROM Students WHERE username = '$username';
```

Propagation graph

```
void ProcessRequest()
 {
    string s1 = ReadData1("name");
    string s2 = ReadData2("encoding");

    string s11 = Prop1(s1);
    string s22 = Prop2(s2);

    string s111 = Cleanse(s11);
    string s222 = Cleanse(s22);

    WriteData("Parameter " + s111);
    WriteData("Header " + s222);
 }
```



Propagation graph

m1→ m2 iff information flows “explicitly” from m1 to m2

Specification

- Source
  - returns tainted data
- Sink
  - error to pass tainted data
- Sanitizer
  - cleanse or untaint the input
- Regular nodes
  - propagate input to output

Vulnerability

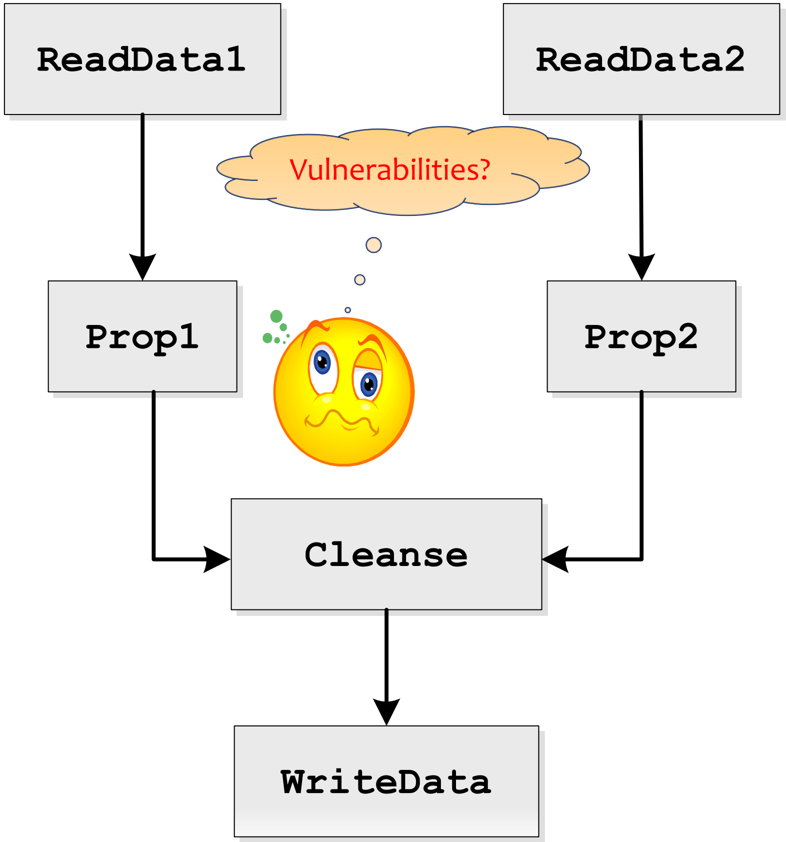
- *Every path from a source to a sink should go through a sanitizer*

- *Any source to sink path without a sanitizer is an information flow vulnerability*

Microsoft Research

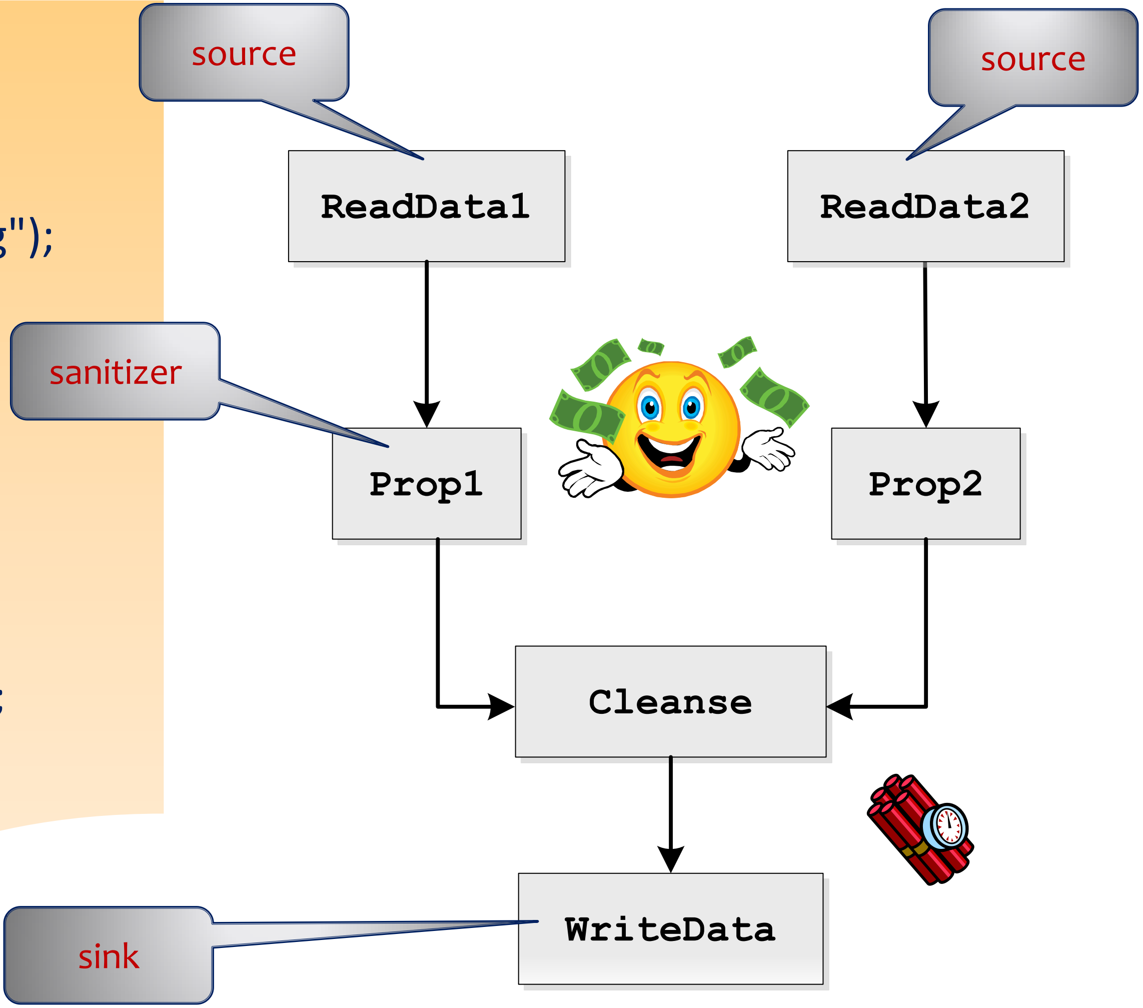
Information flow vulnerabilities

```
void ProcessRequest()
 {
    string s1 = ReadData1("name");
    string s2 = ReadData2("encoding");

    string s11 = Prop1(s1);
    string s22 = Prop2(s2);

    string s111 = Cleanse(s11);
    string s222 = Cleanse(s22);

    WriteData("Parameter " + s111);
    WriteData("Header " + s222);
 }
```



Information flow vulnerabilities

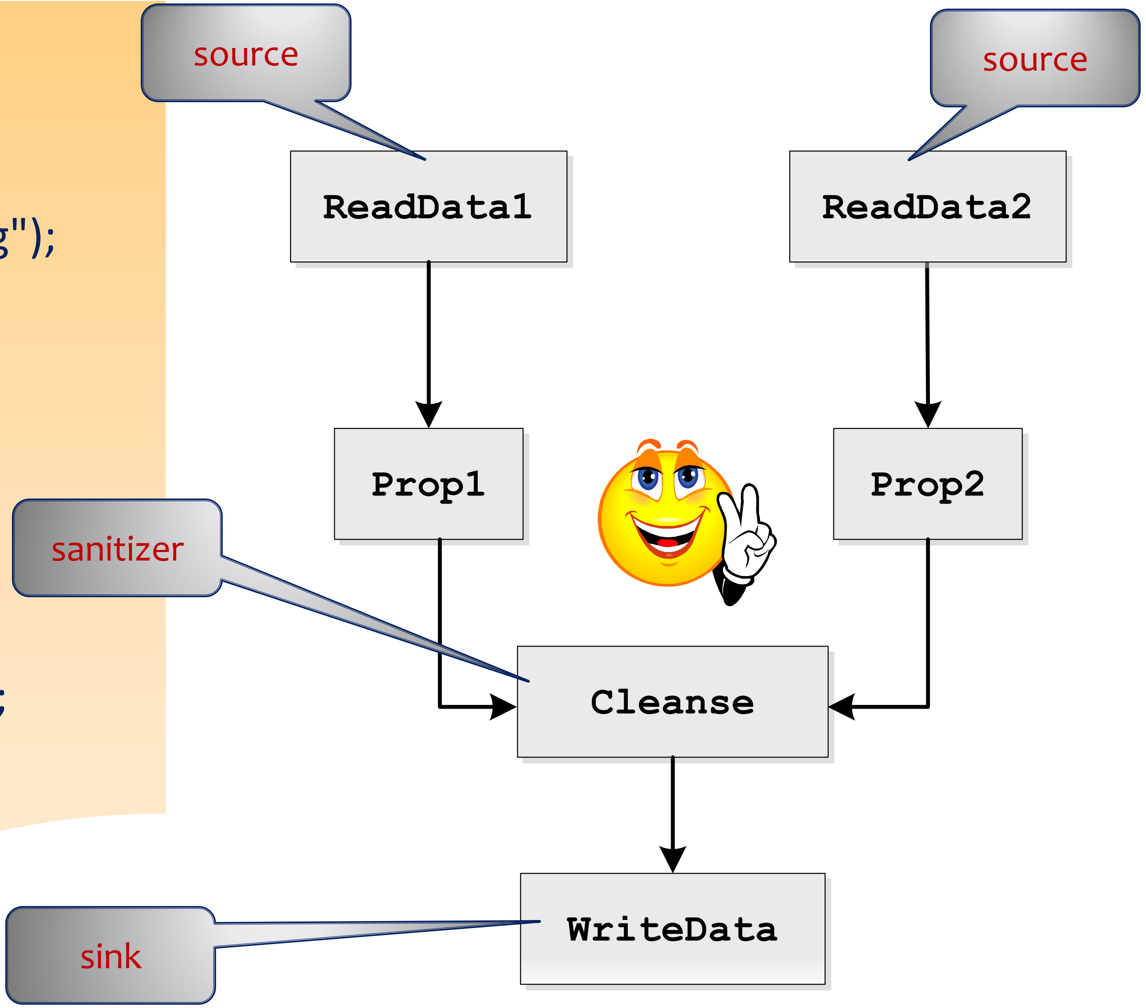
```
void ProcessRequest()
{
    string s1 = ReadData1("name");
    string s2 = ReadData2("encoding");

    string s11 = Prop1(s1);
    string s22 = Prop2(s2);

    string s111 = Cleanse(s11);
    string s222 = Cleanse(s22);

    WriteData("Parameter " + s111);
    WriteData("Header " + s222);
}
```



Microsoft Research

Information flow vulnerabilities

```
void ProcessRequest()
{
    string s1 = ReadData1("name");
    string s2 = ReadData2("encoding");

    string s11 = Prop1(s1);
    string s22 = Prop2(s2);

    string s111 = Cleanse(s11);
    string s222 = Cleanse(s22);

    WriteData("Parameter " + s111);
    WriteData("Header " + s222);
}
```



Goal

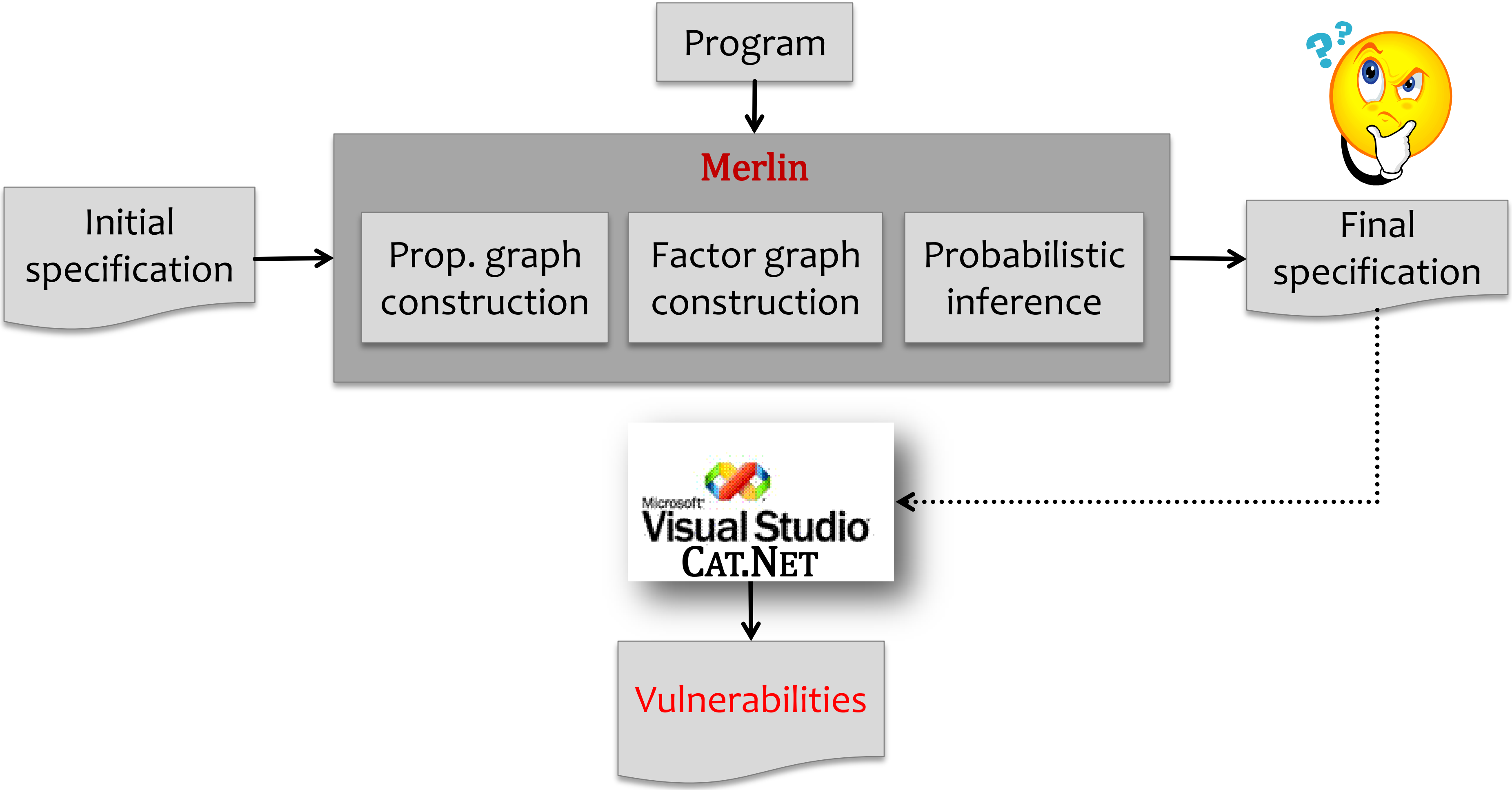
Given a propagation graph, can we infer a specification or 'complete' a partial specification?

Assumption

Most flow paths in the propagation graph are secure

Algorithms

Merlin Architecture



Program

Merlin

Initial specification

Prop. graph construction

Factor graph construction

Probabilistic inference

Final specification

Microsoft Visual Studio CAT.NET

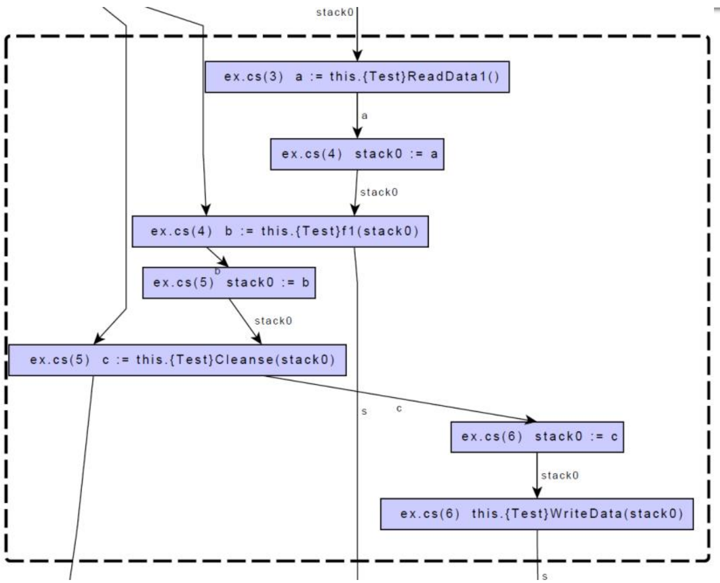
Vulnerabilities

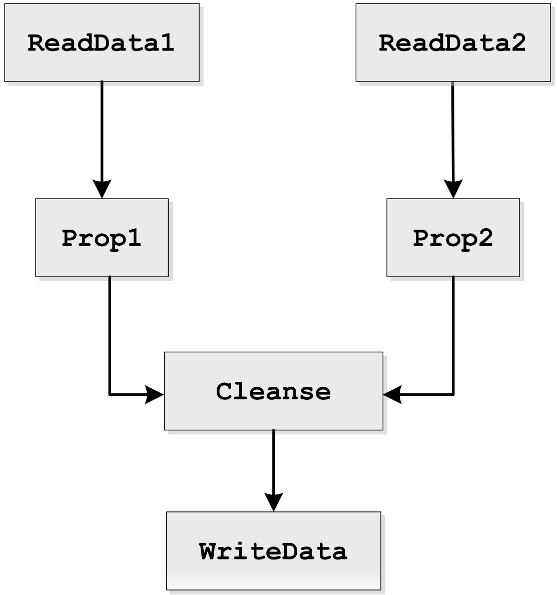
Microsoft Research

Propagation Graph Construction

```
void ProcessRequest()
 {
    string s1 = ReadData1("name");
    string s2 =
ReadData2("encoding");

    string s11 = Prop1(s1);
    string s22 = Prop2(s2);

    string s111 = Cleanse(s11);
    string s222 = Cleanse(s22);

    WriteData("Parameter " + s111);
    WriteData("Header " + s222);
 }
```

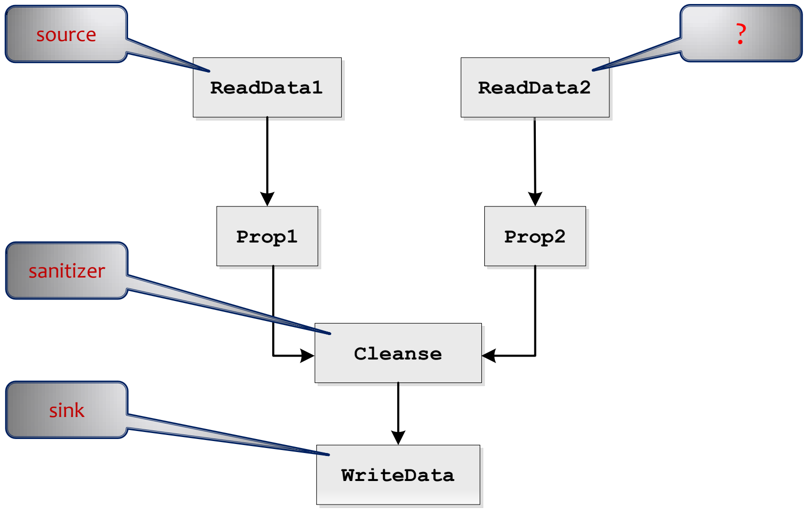
Microsoft Visual Studio CAT.NET

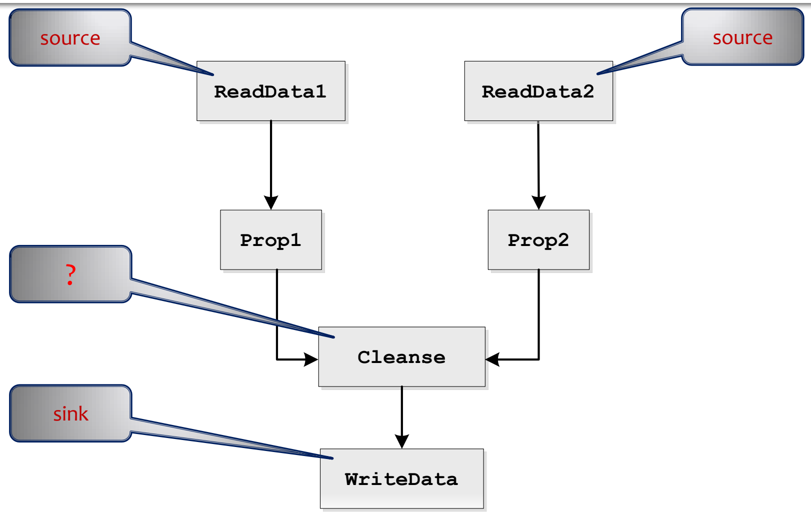

stack0
ex.cs(3) a := this.{Test}ReadData1()
a
ex.cs(4) stack0 := a
stack0
ex.cs(4) b := this.{Test}f1(stack0)
b
ex.cs(5) stack0 := b
stack0
ex.cs(5) c := this.{Test}Cleanse(stack0)
s
c
ex.cs(6) stack0 := c
stack0
ex.cs(6) this.{Test}WriteData(stack0)
s

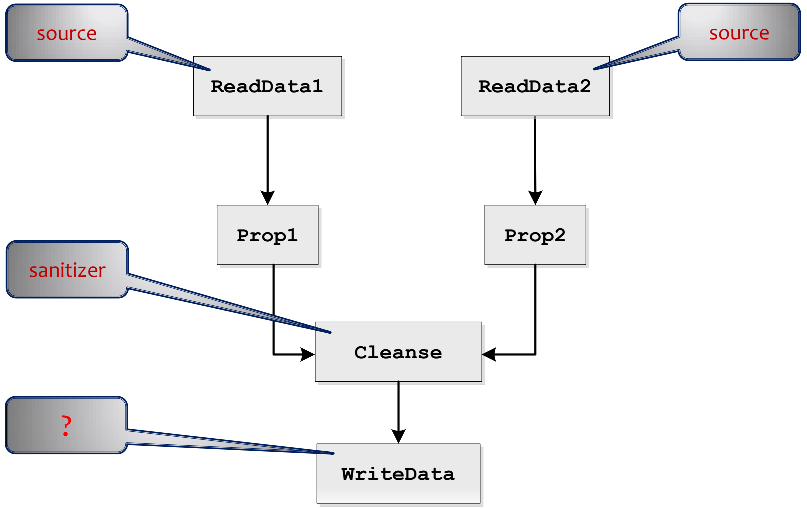


Microsoft Research

Inference?

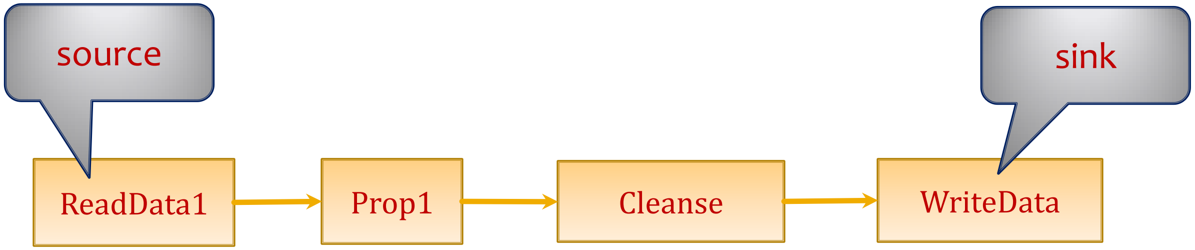

source
?
ReadData1
ReadData2
Prop1
Prop2
sanitizer
Cleanse
sink
WriteData

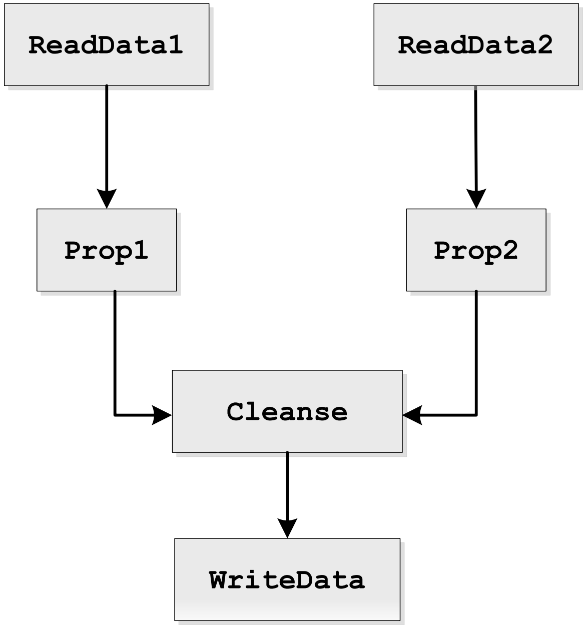

source
source
ReadData1
ReadData2
Prop1
Prop2
?
Cleanse
sink
WriteData


source
source
ReadData1
ReadData2
Prop1
Prop2
sanitizer
Cleanse
?
WriteData

Path constraints

- For every acyclic path $m_1\ m_2 \dots m_n$ the probability that m_1 is a source, m_n is a sink, and $m_2, \dots, m_{n-1}$ are not sanitizers is very low

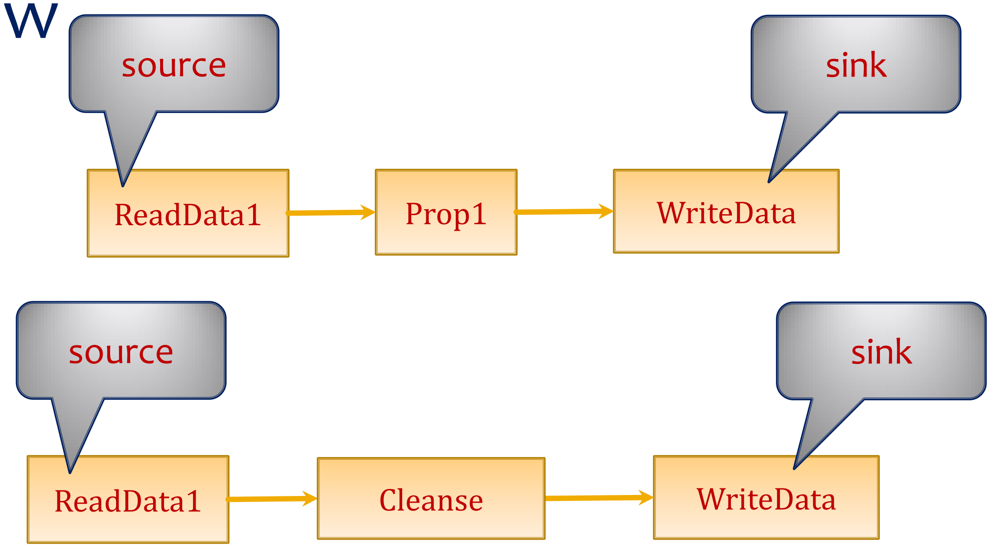


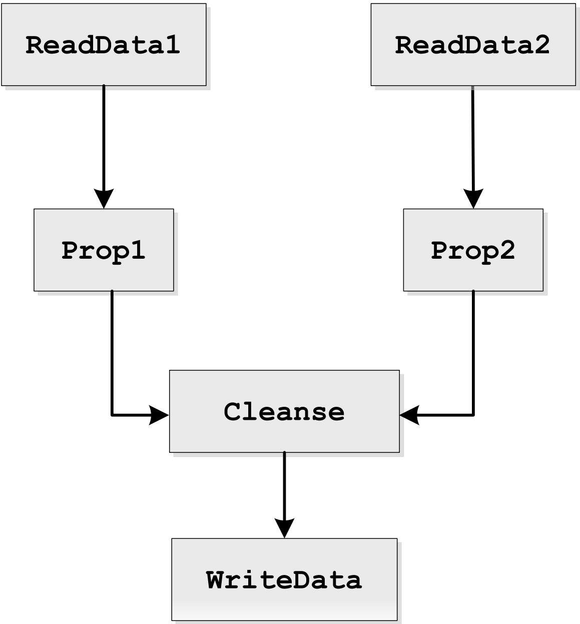


Exponential number of path constraints: $O(2^{|V|})$!

Triple constraints

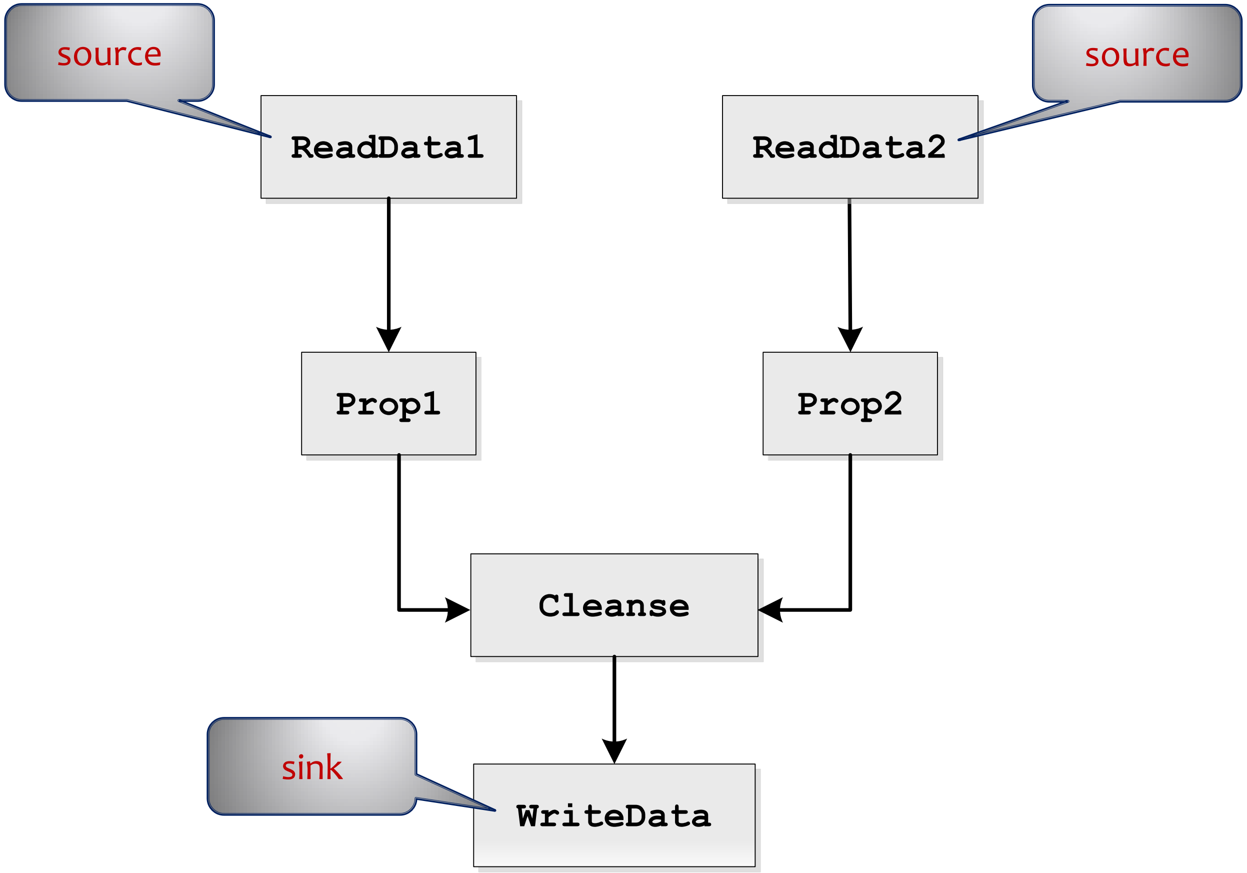
- For every triple $<m_1, m_i, m_n>$ such that m_i is on a path from m_1 to m_n, the probability that m_1 is a source, m_n is a sink, and m_i is not a sanitizer is very low





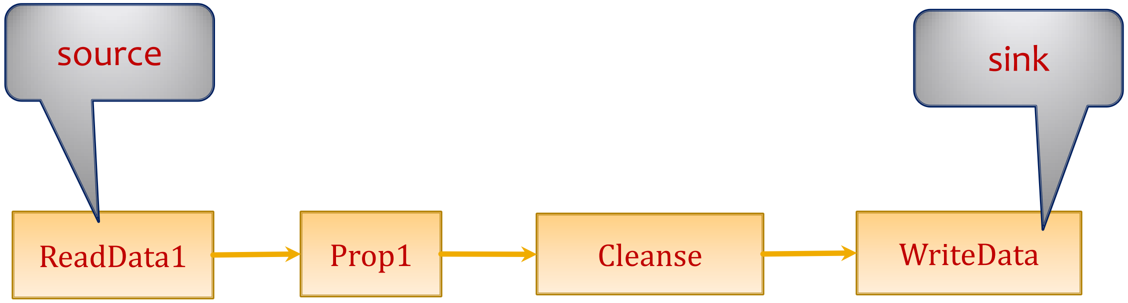
Cubic number of triple constraints: $O(|V|^3)$!

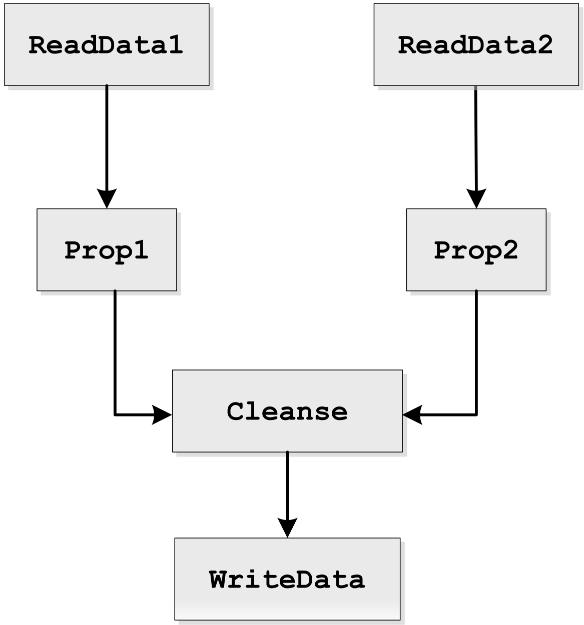
Minimizing Sanitizers



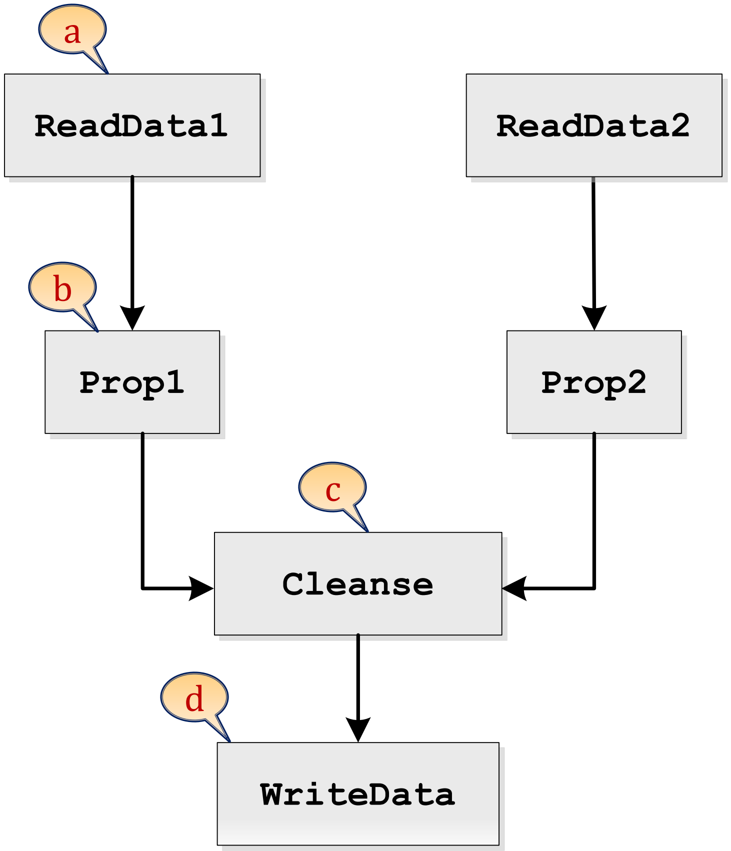
Minimizing Sanitizers

For every pair of nodes m_1, m_2 such that m_1 and m_2 lie on the same path from a potential source to a potential sink, the probability that both m_1 and m_2 are sanitizers is low





Need for probabilistic constraints



Triple constraints

- $\neg(a \land \neg b \land d)$
- $\neg(a \land \neg c \land d)$

Avoid double sanitizers

- $\neg(b \land c)$

- $a \land d \Rightarrow b$
- $a \land d \Rightarrow c$
- $\neg(b \land c)$

Boolean formulas as probabilistic constraints

$$\underbrace{(x_1 \vee x_2)}_{C_1} \wedge \underbrace{(x_1 \vee \neg x_3)}_{C_2}$$

$$f(x_1, x_2, x_3) = f_{C1}(x_1, x_2) \wedge f_{C2}(x_1, x_3)$$

$$f_{C1}(x_1, x_2) = \begin{cases} 1 \text{ if } x_1 \vee x_2 = \text{true} \\ 0 \text{ otherwise} \end{cases}$$

$$f_{C2}(x_1, x_3) = \begin{cases} 1 \text{ if } x_1 \vee \neg x_3 = \text{true} \\ 0 \text{ otherwise} \end{cases}$$

Boolean formulas as probabilistic constraints

$$\underbrace{(x_1 \vee x_2)}_{C_1} \wedge \underbrace{(x_1 \vee \neg x_3)}_{C_2}$$

$$f(x_1, x_2, x_3) = f_{C1}(x_1, x_2) \wedge f_{C2}(x_1, x_3)$$

0.9

$$f_{C1}(x_1, x_2) = \begin{cases} 1 \text{ if } x_1 \vee x_2 = \text{true} \\ 0 \text{ otherwise} \end{cases}$$

0.1

0.9

$$f_{C2}(x_1, x_3) = \begin{cases} 1 \text{ if } x_1 \vee \neg x_3 = \text{true} \\ 0 \text{ otherwise} \end{cases}$$

0.1

$$p(x_1, x_2, x_3) = f_{C1}(x_1, x_2) \times f_{C2}(x_1, x_3)/Z$$

$$Z = \sum_{x1,\, x2,\, x3} (f_{C1}(x_1, x_2) \times f_{C2}(x_1, x_3))$$

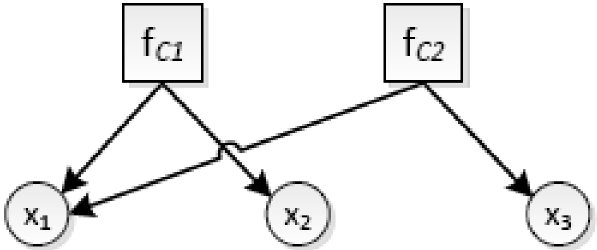
Solution = Marginalization

marginal

$$p_i(x_i) = \sum_{x1} \dots \sum_{x(i-1)} \sum_{x(i+1)} \dots \sum_{xN} p(x_1, \dots, x_N)$$

- Step 1: choose x_i with highest $p_i(x_i)$ and set x_i = true if $p_i(x_i)$ is greater than a threshold, false otherwise
- Step 2: recompute marginals and repeat Step 1 until all variables have been assigned

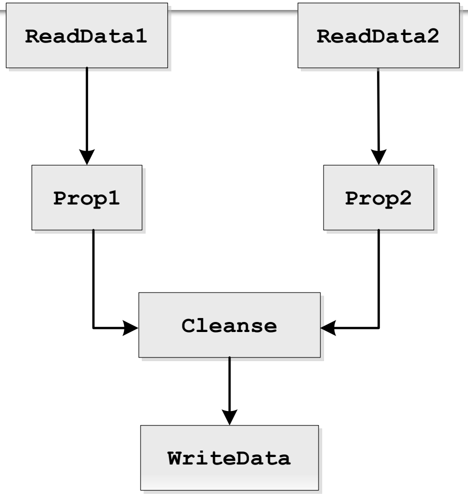
Factor graphs: efficient computation of marginals

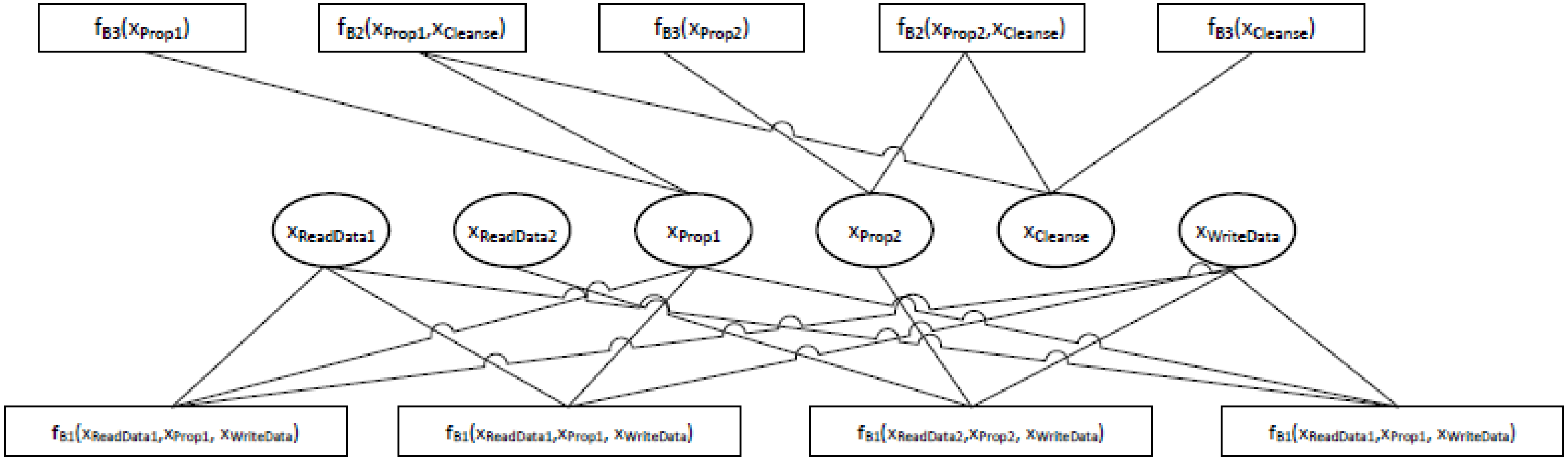


$$f_{C1}(x_1, x_2) = \begin{cases} 1 \text{ if } x_1 \vee x_2 = \text{true} \\ 0 \text{ otherwise} \end{cases}$$

$$f_{C2}(x_1, x_3) = \begin{cases} 1 \text{ if } x_1 \vee \neg x_3 = \text{true} \\ 0 \text{ otherwise} \end{cases}$$

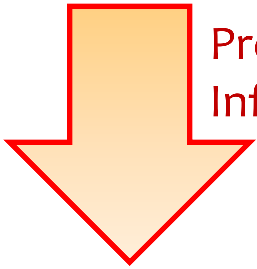
Factor Graphs




$f_{B3}(x_{Prop1})$
$f_{B2}(x_{Prop1},x_{Cleanse})$
$f_{B3}(x_{Prop2})$
$f_{B2}(x_{Prop2},x_{Cleanse})$
$f_{B3}(x_{Cleanse})$
$x_{ReadData1}$
$x_{ReadData2}$
x_{Prop1}
x_{Prop2}
$x_{Cleanse}$
$x_{WriteData}$
$f_{B1}(x_{ReadData1},x_{Prop1}, x_{WriteData})$
$f_{B1}(x_{ReadData1},x_{Prop1}, x_{WriteData})$
$f_{B1}(x_{ReadData2},x_{Prop2}, x_{WriteData})$
$f_{B1}(x_{ReadData1},x_{Prop1}, x_{WriteData})$

Probabilistic Inference

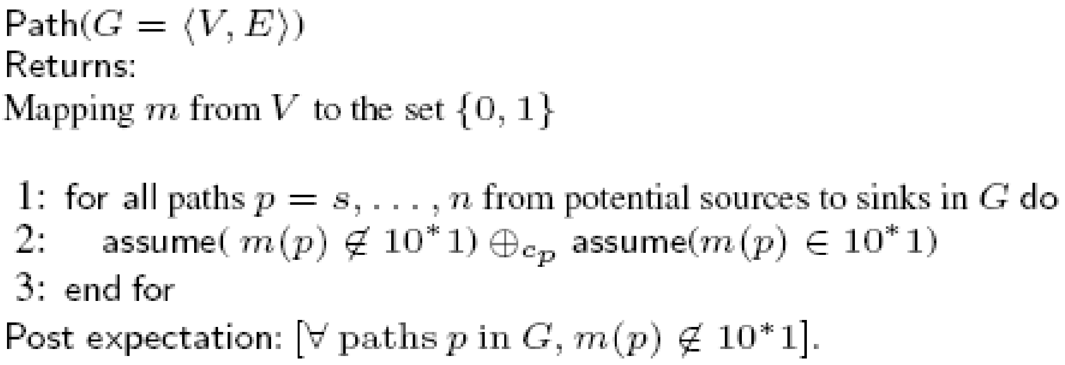
	Source	Sanitizer	Sink
ReadData1	.95	.001	.001
ReadData2	.5	.5	.5
Cleanse	.5	.5	.5
WriteData	.5	.5	.85
…			



Probabilistic Inference

	Source	Sanitizer	Sink
ReadData1	.95	.001	.001
ReadData2	.5	.5	.5
Cleanse	.01	.997	.03
WriteData	.5	.5	.85
…			

Paths vs. Triples



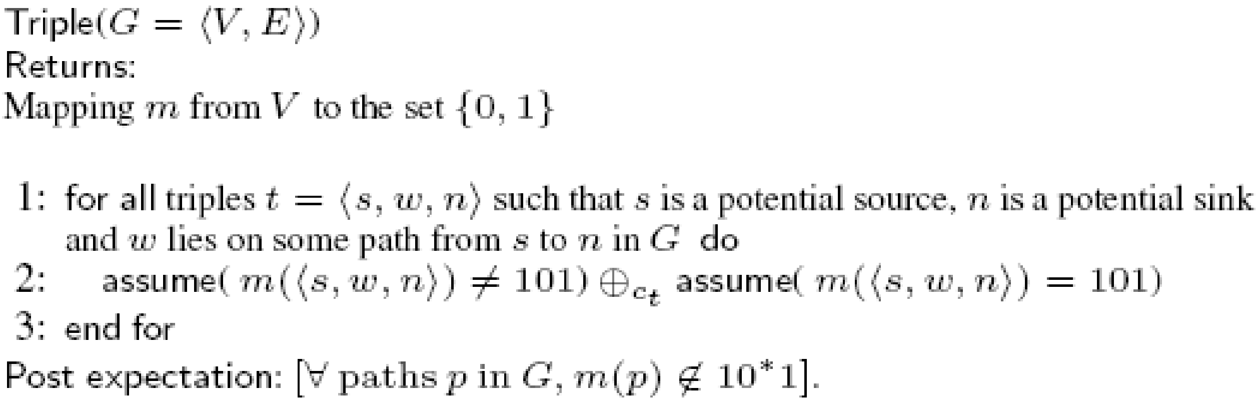
Path($G = \langle V, E \rangle$)
Returns:
Mapping m from V to the set $\{0, 1\}$

```
for all paths $p = s, \ldots, n$ from potential sources to sinks in $G$ do
    assume( $m(p) \notin 10^*1$) $\oplus_{c_p}$ assume($m(p) \in 10^*1$)
end for
```

Post expectation: [$\forall$ paths p in G, $m(p) \notin 10^*1$].

Theorem

Path refines Triple



Triple($G = \langle V, E \rangle$)
Returns:
Mapping m from V to the set $\{0, 1\}$

```
for all triples $t = \langle s, w, n \rangle$ such that $s$ is a potential source, $n$ is a potential sink
  and $w$ lies on some path from $s$ to $n$ in $G$ do
    assume( $m(\langle s, w, n \rangle) \neq 101$) $\oplus_{c_t}$ assume( $m(\langle s, w, n \rangle) = 101$)
end for
```

Post expectation: [$\forall$ paths p in G, $m(p) \notin 10^*1$].

Microsoft Research

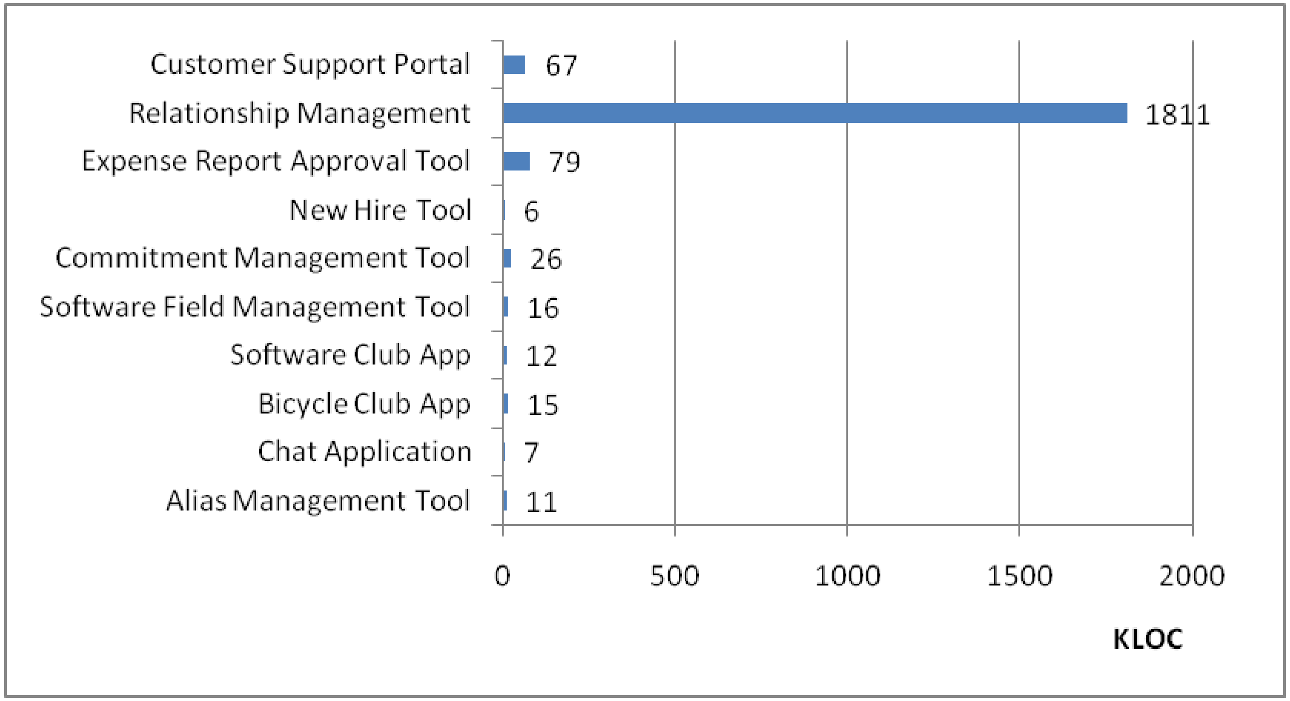
Experiments

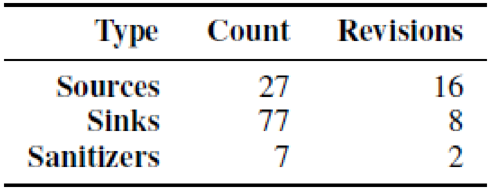
Implementation

- Merlin is implemented in C#
  - Uses CAT.NET for building the propagation graph
  - Uses Infer.NET for probabilistic inference
    - http://research.microsoft.com/infernet

Experiments

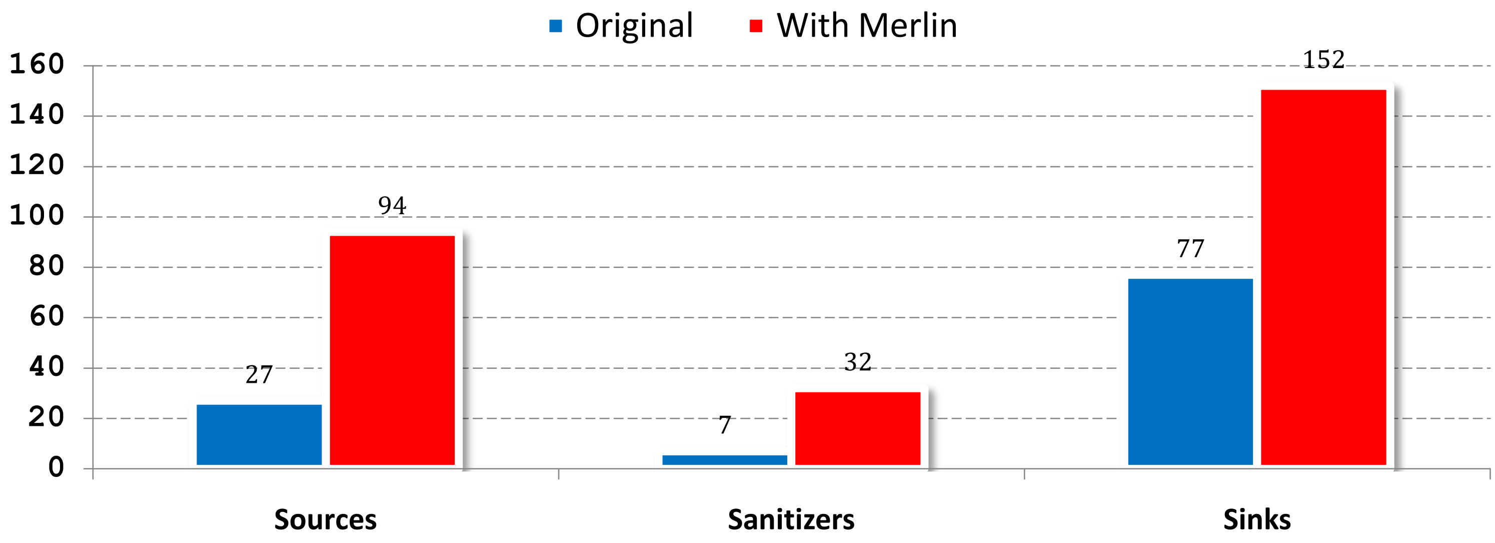
10 line-of-business applications written in C# using ASP.NET





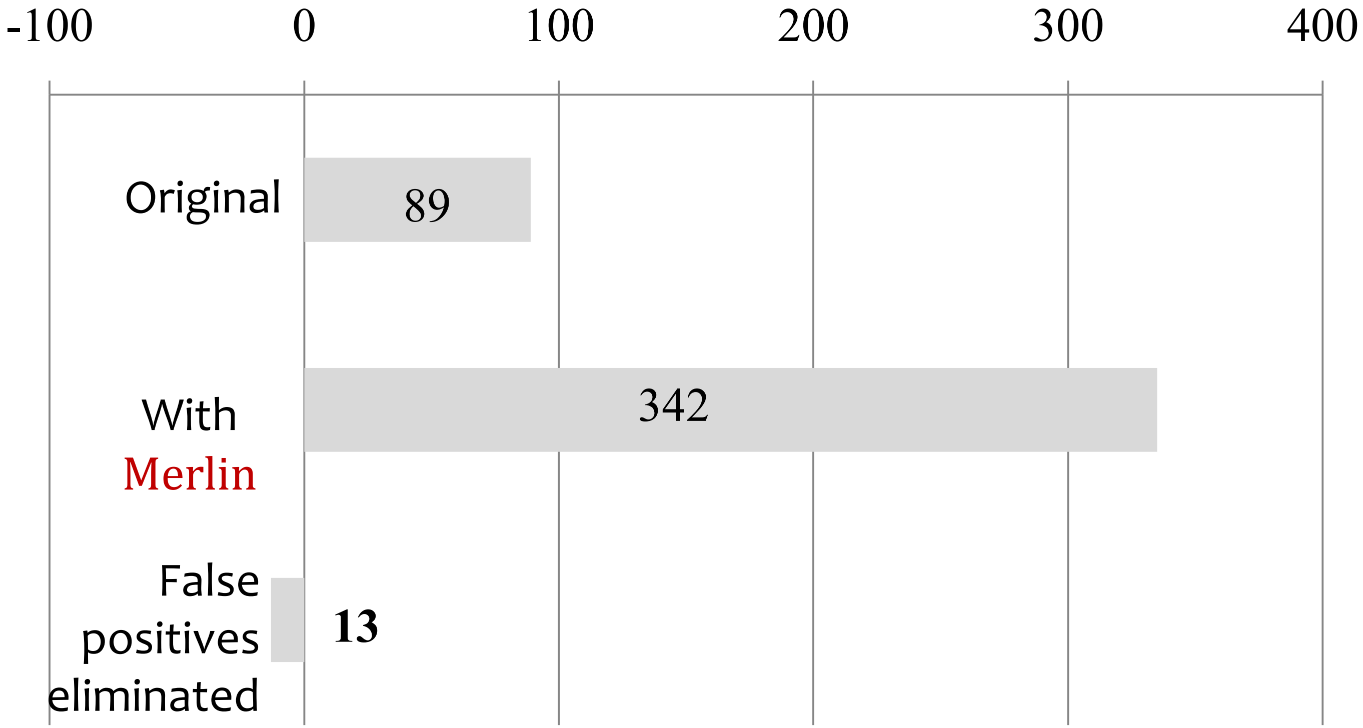
Type	Count	Revisions
Sources	27	16
Sinks	77	8
Sanitizers	7	2

Summary of Discovered Specifications



Microsoft Research

Summary of Discovered Vulnerabilities



Experiments - summary

- 10 large Web apps in .NET
- Time taken per app < 4 minutes
- New specs: 167
- New vulnerabilities: 322
- False positives removed: 13
- Final false positive rate for CAT.NET after Merlin < 1%

Summary

- Merlin is first practical approach to infer explicit information flow specifications
- Design based on a formal characterization of an approximate probabilistic constraint system
- Able to successfully and efficiently infer explicit information flow specifications in large applications which result in detection of new vulnerabilities

http://research.microsoft.com/merlin

Microsoft Research