On-Demand Machine Reasoning for The New Cyber Security

Vesal Vojdani
"The current trajectories for benefit and risk are unsustainable"

(2016 Federal Cybersecurity Research and Development Strategic Plan)
Ten Years Ago

51% are just “repeated basic mistakes”
Ten Years of Tremendous Progress…
And Now...

2/3 of that 73% are “basic input validation and buffer errors”
“The proportion of implementation vulnerabilities for 2008 to 2016 is close to the 64% reported for 1998 to 2003… This suggests that little progress has been made in reducing these vulnerabilities that result from simple mistakes which should be easy to prevent.”

–NIST 2016, An Analysis of Vulnerability Trends
WHY?

Not rhetorical!

Please answer if you know…
Status Quo

• “Great strides have been made in defining software vulnerabilities, cataloging them and understanding them. [As well as...] in educating the software community about vulnerabilities, attendant patches and underlying weaknesses.

• This work, however, is insufficient... [CNN]

• Clearly a different approach—one that relies on improving software—is needed.”
The New Cyber Security

NISTIR 8151: Dramatically Reducing Software Vulnerability
“Heuristic analysis is faster than sound analysis, but lacks assurance that comes from a chain of logical reasoning.”

–NISTIR 8151: Dramatically Reducing Software Vulnerability
Machine reasoning for these “simple bugs”

• Nothing is really simple about C or even Java.

• The heartbleed bug: bit operations (byte swapping) and pointers:

  unsigned char *p = &s->s3->rrec.data[0]
In General: Satisfiability

\[ \mathcal{M} \models \varphi \]

- \( \mathcal{M} \): a model of the system
- \( \varphi \): a specification of what is expected of \( \mathcal{M} \)

Machine Reasoning

All about satisfiability
Deductive Methods

- Describe the system as a set of logical formulas $\Gamma$.
- The specification is another formula $\varphi$.
- Verify by finding a proof of

$$\Gamma \vdash \varphi$$

Deductive Methods

Programming Languages like Dafny, F*, etc.
Example: Avoid division by Zero

\[ x := y \]

\[ x := x - 1 \]

\[ z := y / x \]
Example: Avoid division by Zero

\[
x := y
\]

\[
x := x - 1
\]
\[\text{if } x \neq 0 \]

\[
z := \frac{y}{x}
\]
Example: Avoid division by Zero

\[ x := y \]
\[ (x - 1 \neq 0) \]
\[ x := x - 1 \]
\[ (x \neq 0) \]
\[ z := y / x \]
Example: Avoid division by Zero

\[
\begin{align*}
\text{(y - 1) \neq 0} \\
x &:= y \\
\text{(x - 1) \neq 0} \\
x &:= x - 1 \\
\text{(x \neq 0)} \\
z &:= y/x
\end{align*}
\]

Answer: “y \neq 1”

This is fed to MS Brain (Z3)
Definition

A set $\mathbb{D}$ together with a relation $\sqsubseteq$ is a **partial order** if for all $a, b, c \in \mathbb{D}$,

- $a \sqsubseteq a$ \quad \text{reflexivity}
- $a \sqsubseteq b \land b \sqsubseteq a \implies a = b$ \quad \text{anti-symmetry}
- $a \sqsubseteq b \land b \sqsubseteq c \implies a \sqsubseteq c$ \quad \text{transitivity}

Not everything is decidable…

Need to approximate safely!
Our domain elements represent propositions about the program.

Let $p \models x$ denote “$x$ holds whenever execution reaches program point $p$.”

We order these propositions such that

\[ x \sqsubseteq y \text{ whenever } (p \models x) \implies (p \models y) \]

**The Implication Ordering**

“age is 37” $\sqsubseteq$ “age is over 30” $\sqsubseteq$ “can’t really say”
Assume there are two paths to reach \( p \) (true-branch and false-branch).

If we have \( x \) along one path and \( y \) along the other, how can we combine this information?

We want something that is true of both paths, and as precise as possible.

Combining Information

What is this core piece of AI technology?
For the interval domain

\[ 0 \leq x \leq 5 \]

\[ x := 0 \]

\[ x \geq 5 \] ?

\[ x = 5 ! \]

\[ x = x + 1 \]

\[ x < 5 \] ?
For the interval domain

0 > [0, 0] x := 0

1 x ≥ 5?

2 x = x + 1

3 x = 5!

4 x < 5?
For the interval domain

\[
0 \leq x < 5
\]

\[
x = x + 1
\]

\[
x \geq 5?
\]

\[
x = 5!
\]

\[
x < 5?
\]
For the interval domain

\[
\begin{align*}
x &= x + 1 \\
\text{if } x \geq 5 \text{ then } x &= 5! \\
\text{if } x < 5 \text{ then } x &= x + 1
\end{align*}
\]
For the interval domain

\[
\begin{align*}
\text{0} & \quad \text{\( \chi \equiv 0 \)} \\
[0, 1] & \quad \text{\( \chi \geq 5 \) ?} \\
\text{1} & \quad \text{\( \chi = \chi + 1 \)} \\
\text{2} & \quad \text{\( \chi < 5 \) ?} \\
\text{3} & \quad \text{\( \chi = 5! \)} \\
\text{4} & \\
\end{align*}
\]
For the interval domain

\[\begin{align*}
0 & \xrightarrow{x := 0} [0, 2] \\
\top & \xrightarrow{} 1 \\
1 & \xrightarrow{x \geq 5 ?} 3 \\
3 & \xrightarrow{} 4 \\
2 & \xrightarrow{x < 5 ?} 1 \\
2 & \xrightarrow{} [0, 1] \\
3 & \xrightarrow{x = 5 !} 4 \\
4 & \xrightarrow{} 3
\end{align*}\]
For the interval domain

\[ 0 \leq x < 5 \]

\[ x = x + 1 \]

\[ x \geq 5 \]

\[ x < 5 \]

\[ x \leq 0 \]
For the interval domain

\[ 0 \leq x < 5 \]

\[ x = x + 1 \]

\[ x = 5! \]

\[ x \geq 5 \]

\[ x < 5 \]
For the interval domain
For the interval domain

\[
\begin{align*}
0 & \quad [0, 4] \\
1 & \quad x \geq 5? \\
2 & \quad x = x + 1 \\
3 & \quad x < 5? \\
4 & \quad x = 5! \\
\end{align*}
\]
For the interval domain

\[
\begin{align*}
\text{For the interval domain} & \\
0 & \\
\text{\quad \quad } x := 0 & \\
\text{\quad [0, 4]} & \\
\text{\quad \quad } x \geq 5 ? & \\
\text{\quad \quad } x = 5 ! & \\
\text{\quad \quad } x \leq 5 ? & \\
\text{\quad \quad } x = x + 1 & \\
\text{\quad [0, 4]} & \\
\end{align*}
\]
For the interval domain

\[ 0 \leq x < 5 \]
For the interval domain

\[ x := 0 \]

\[ x = x + 1 \]

\[ x < 5 \text{?} \]

\[ x \geq 5 \text{?} \]

\[ x = 5! \]

\[ T \]

\[ [0, 4] \]

\[ [0, 5] \]

\[ [5, 5] \]
For the interval domain

\[ 0, 4 \]  
\[ 0, 5 \]  
\[ 5, 5 \]  
Success?  
\[ 5, 5 \]  
\[ 5, 5 \]  
\[ 5, 5 \]  
\[ 5, 5 \]  
\[ 5, 5 \]
What an amazing analyzer...

• This was not analysis...
• I was just executing the program.
• Termination not guaranteed.
\[ \nabla : \mathbb{D} \times \mathbb{D} \to \mathbb{D} \text{ is a widening operator if} \]

1. \( \forall x, y \in \mathbb{D} : (x \sqsubseteq x \nabla y) \land (y \sqsubseteq x \nabla y) \)
2. for every chain \( x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq \cdots \),

\[
\begin{align*}
y_0 &= x_0 \\
y_1 &= y_0 \nabla x_1 \\
y_2 &= y_1 \nabla x_2 \\
\cdots
\end{align*}
\]

is not strictly increasing.

More AI: Widening

(Actually, here people are trying to apply learning!)
Replay with Widening/Narrowing

\[
\begin{align*}
\text{0: } & \quad \mathbf{T} \\
\text{1: } & \quad x := 0 \\
\text{2: } & \quad x = x + 1 \\
\text{3: } & \quad x \geq 5 \text{?} \\
\text{4: } & \quad x = 5! \\
\text{5: } & \quad x < 5 \text{?}
\end{align*}
\]
Replay with Widening/Narrowing

0

1

2

3

4

\( x \equiv 0 \)

\([0, 0]\)

\( x \geq 5 \) ?

\( x = 5 ! \)

\( x < 5 ? \)

\( x = x + 1 \)

\( T \)
Replay with Widening/Narrowing

0

$x := 0$

$[0, 0]$

$x \geq 5$?

$x = 5!$

$x = x + 1$

$x < 5$?

$[0, 0]$
Replay with Widening/Narrowing

Ouch: Previous value [0,0], new value [0,1]

$x := 0$

$x = x + 1$

$x \geq 5$?

$x < 5$?

$x = 5!$

$x = x + 1$
Replay with Widening/Narrowing

\[ x := 0 \]

\[ [0, \infty] \]

\[ x = x + 1 \]

\[ x \geq 5 ? \]

\[ x < 5 ? \]

\[ x = 5 ! \]

\[ [0, 0] \]
Replay with Widening/Narrowing

0

\[ x := 0 \]

\([0, \infty]\)

1

\[ x \geq 5 \]?

2

\[ x = x + 1 \]

3

\[ x < 5 \]?

4

\[ x = 5 ! \]

\([0, 4]\)
Replay with Widening/Narrowing

\[ x := 0 \]

\[ [0, \infty] \]

\[ x = x + 1 \]

\[ x \geq 5 \]

\[ x < 5 \]

\[ x = 5 ! \]

\[ [0, \infty] \]
Replay with Widening/Narrowing

This constraint really wants \([0, 4]\)

Done!
Now Narrowing...
Replay with Widening/Narrowing

0
\[0, \infty\] \(x := 0\)

1
\([5, \infty]\) \(x \geq 5\) ?

2
\([0, 4]\) \(x = x + 1\)

3
\(x = 5\) !

4
\(x < 5\) ?
Replay with Widening/Narrowing

\[ x = 0 \]

\[ [0, 5] \]

\[ x = x + 1 \]

\[ [0, 4] \]

\[ x \geq 5 ? \]

\[ x < 5 ? \]

\[ x = 5! \]

\[ [5, \infty] \]
Replay with Widening/Narrowing

\begin{align*}
T & \rightarrow 0 \quad x := 0 \\
[0, 5] & \rightarrow 1 \quad x \geq 5 \quad ? \quad [5, 5] \\
1 & \rightarrow 2 \quad x = x + 1 \quad x < 5 \quad ? \quad [0, 4] \\
2 & \rightarrow 3 \quad x = 5 \quad ! \\
3 & \rightarrow 4 \\
4 & \rightarrow \end{align*}
Replay with Widening/Narrowing

\[ x := 0 \]

\[ x \geq 5 \ ? \]

\[ x = 5 ! \]

\[ x = x + 1 \]

\[ x < 5 ? \]

Success!
Touchpoint #1: Coding

Interactive Static Analysis

Increase developers' awareness of security vulnerabilities, and their secure programming knowledge and behavior.

Vulnerabilities related to lack of input validation, output encoding, or SQL injection

Access control vulnerabilities

ASIDE: Application Security in the IDE

On-Demand Analysis

We want sound machine reasoning in the IDE!
\[ a \sqsupseteq b = \begin{cases} a \sqcap b, & \text{if } b \sqsubseteq a \\ a \sqsupset b, & \text{otherwise} \end{cases} \]

**Intertwined “warrowing”**

Can be used in demand-driven solvers
computing post solutions by using the combined widening and narrowing of the system of equations in contradiction to our assumption! Accordingly, but then we have:

**Proof.**

Consider a widening operator $\\mathcal{W}$ that is a post solution, i.e., for all $x \in X$, we define a new binary operator $\mathcal{E}$, and then every $x \in X$.

Thus, any generic solver can be applied to improve a post solution by means of a narrowing iteration—given that all right-hand sides of equations are monotonic.

At least not far away from the classical iterator.

The worst case complexity for the new round-robin solver turns out to be even by a factor of 2 faster than ordinary round-robin ordering which naturally occurs when solving the system of equations rowing, which intertwined application of widening and narrowing operator. The intertwined application of widening and narrowing is idempotent the following holds:

For the new operator $\mathcal{E}$, has the additional advantage that values may otherwise. Round-robin iteration, enhanced with the natural ordering $\mathcal{D}$, has a significant impact on performance. Hence, the linear ordering should be chosen in a way that innermost loops would be

Moreover, no restriction is imposed any longer concerning monotonicity of right-hand sides.
Our Safety and Security depends on getting sound machine reasoning into the hands of developers!