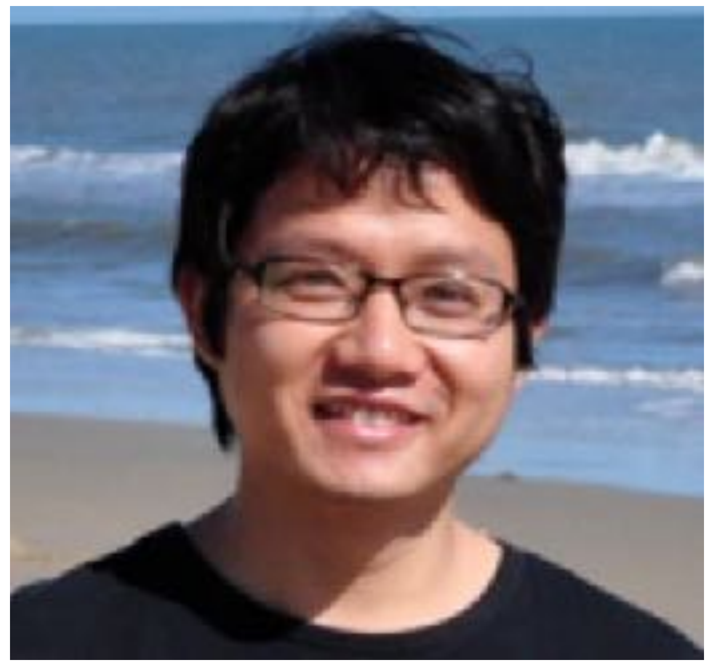
Secure ZK Circuits via Formal Methods

Guest Lecturer: **Yu Feng (UCSB & Veridise)**



Zero Knowledge Proofs

Instructors: Dan Boneh, Shafi Goldwasser, Dawn Song, Justin Thaler, Yupeng Zhang

Stanford University

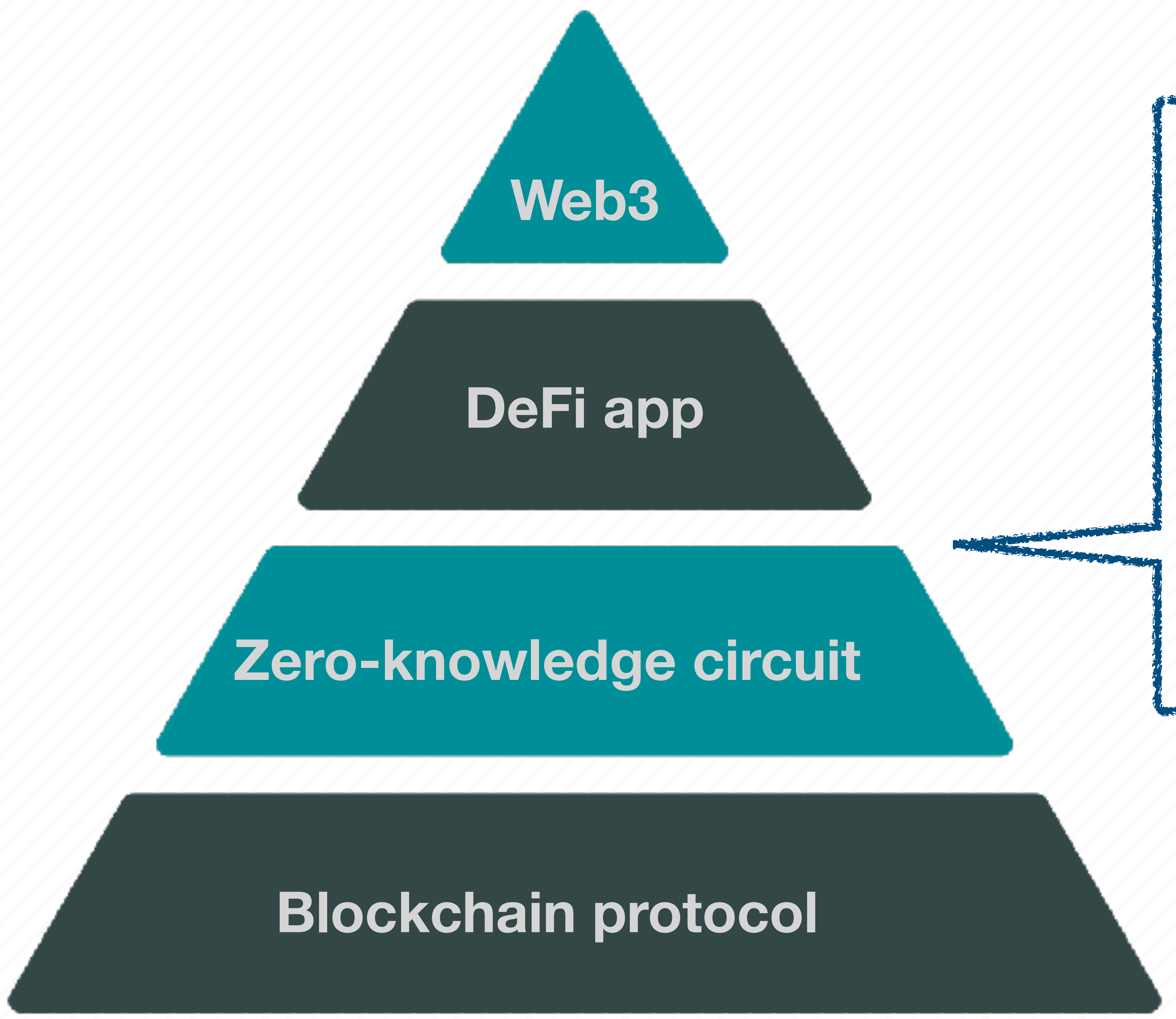
Berkeley UNIVERSITY OF CALIFORNIA

GEORGETOWN UNIVERSITY

TEXAS A&M UNIVERSITY

Motivation

Bugs in blockchain software are **extremely** costly



Bugs in any of these layers can be catastrophic when exploited!

Smart Contract Bugs

Ethereum DeFi Protocol Beanstalk Hacked for $182 Million—What You Need to Know

Beanstalk got jacked by a giant flash attack.

By Jeff Benson

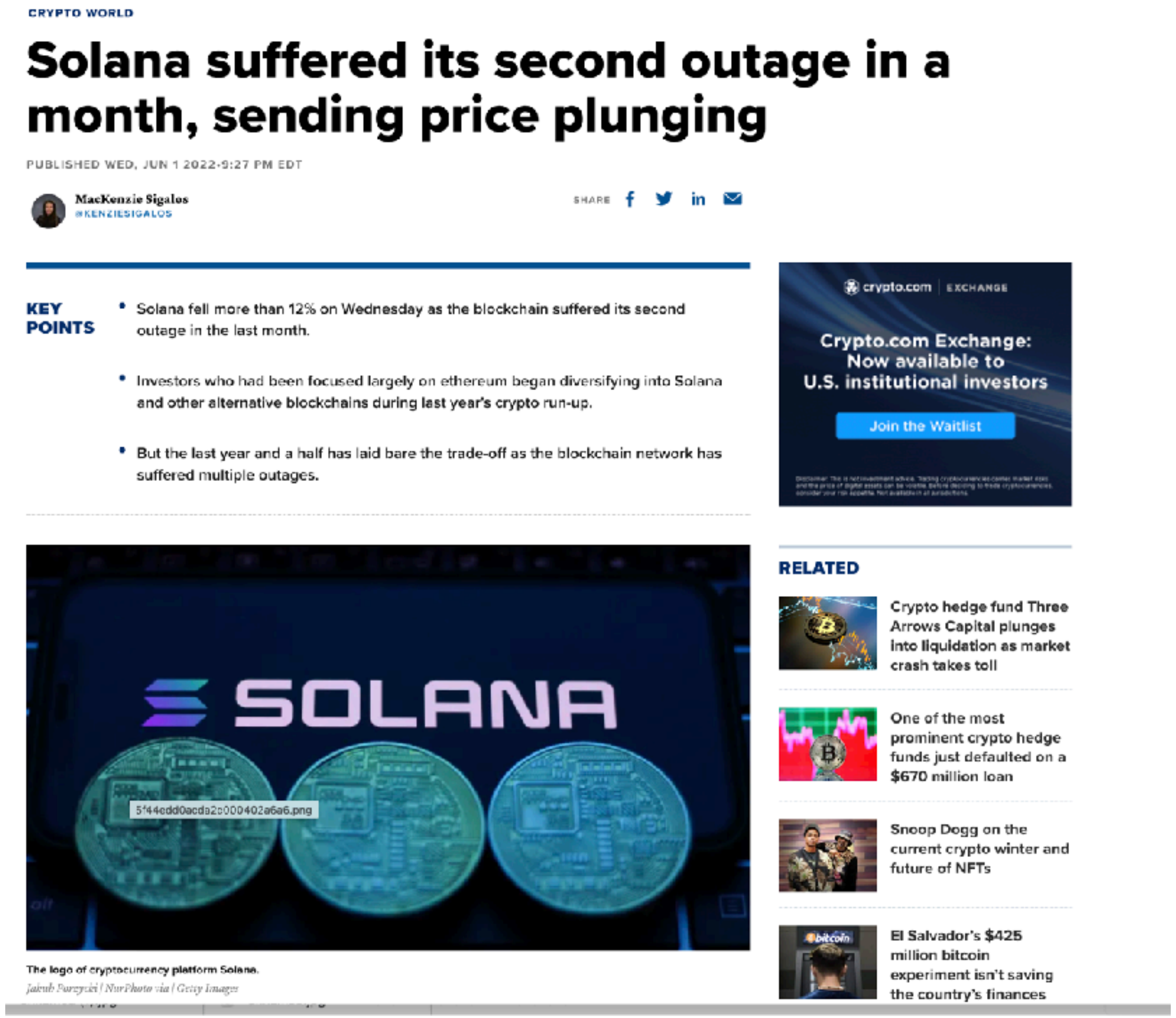
Apr 18, 2022

2 min read

Beanstalk. Image: Shutterstock

Flash loan vulnerability in smart contract

Blockchain Protocol Bugs



CRYPTO WORLD

Solana suffered its second outage in a month, sending price plunging

PUBLISHED WED, JUN 1 2022-9:27 PM EDT

MacKenzie Sigalos
@KENZIESIGALOS

SHARE

KEY POINTS

- Solana fell more than 12% on Wednesday as the blockchain suffered its second outage in the last month.
- Investors who had been focused largely on ethereum began diversifying into Solana and other alternative blockchains during last year's crypto run-up.
- But the last year and a half has laid bare the trade-off as the blockchain network has suffered multiple outages.

The logo of cryptocurrency platform Solana.
Jakub Porzycki | NurPhoto via | Getty Images

crypto.com | EXCHANGE

Crypto.com Exchange: Now available to U.S. institutional investors

Join the Waitlist

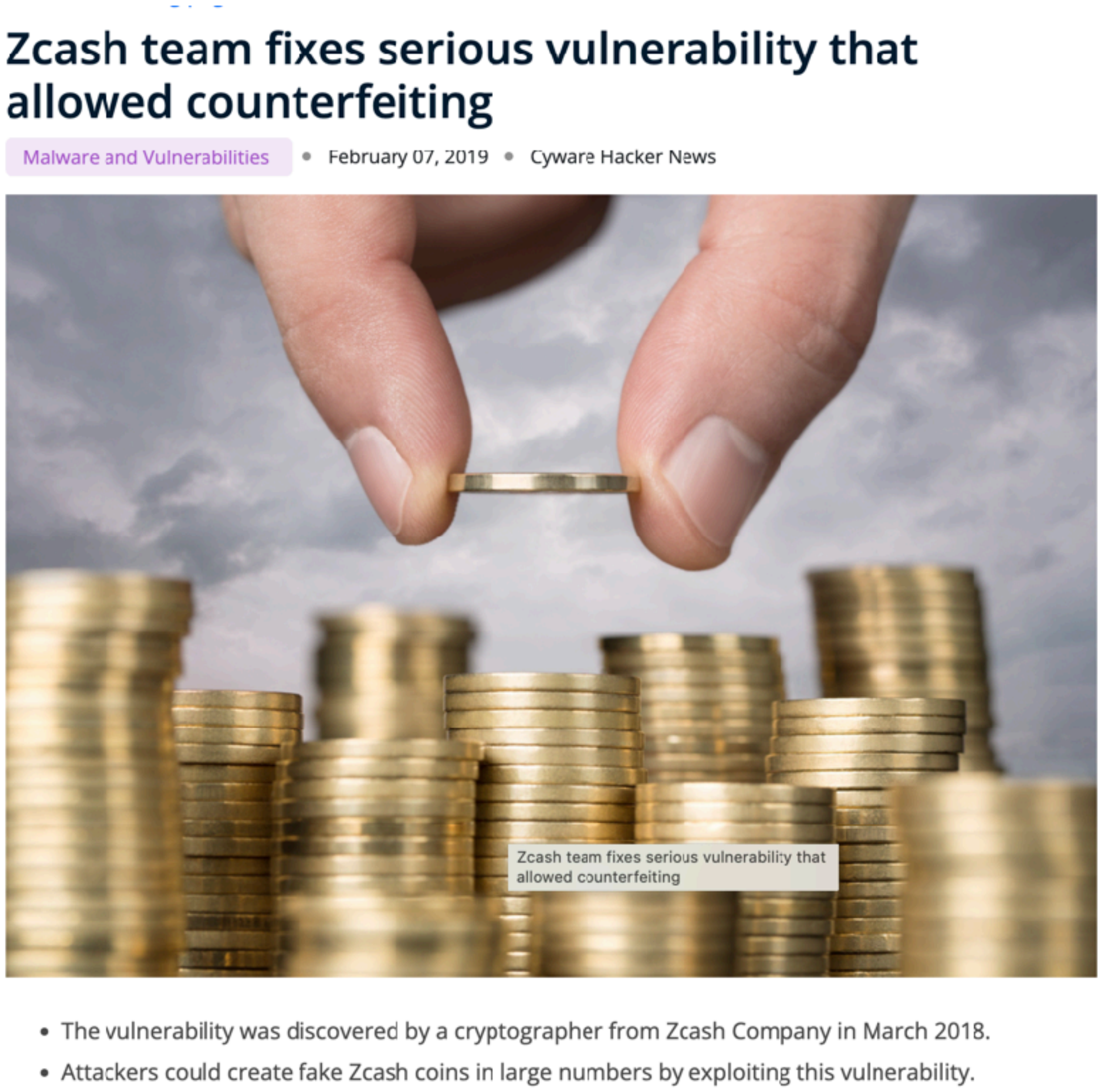
Disclaimer: This is not investment advice. Trading cryptocurrencies carries market risk and the price of digital assets can be volatile. Before deciding to trade cryptocurrencies, consider your risk appetite. Not available in all jurisdictions.

RELATED

- Crypto hedge fund Three Arrows Capital plunges into liquidation as market crash takes toll
- One of the most prominent crypto hedge funds just defaulted on a $670 million loan
- Snoop Dogg on the current crypto winter and future of NFTs
- El Salvador's $425 million bitcoin experiment isn't saving the country's finances

DoS vulnerability in consensus protocol

ZK Bugs are Coming



Zcash team fixes serious vulnerability that allowed counterfeiting

Malware and Vulnerabilities • February 07, 2019 • Cyware Hacker News

Zcash team fixes serious vulnerability that allowed counterfeiting

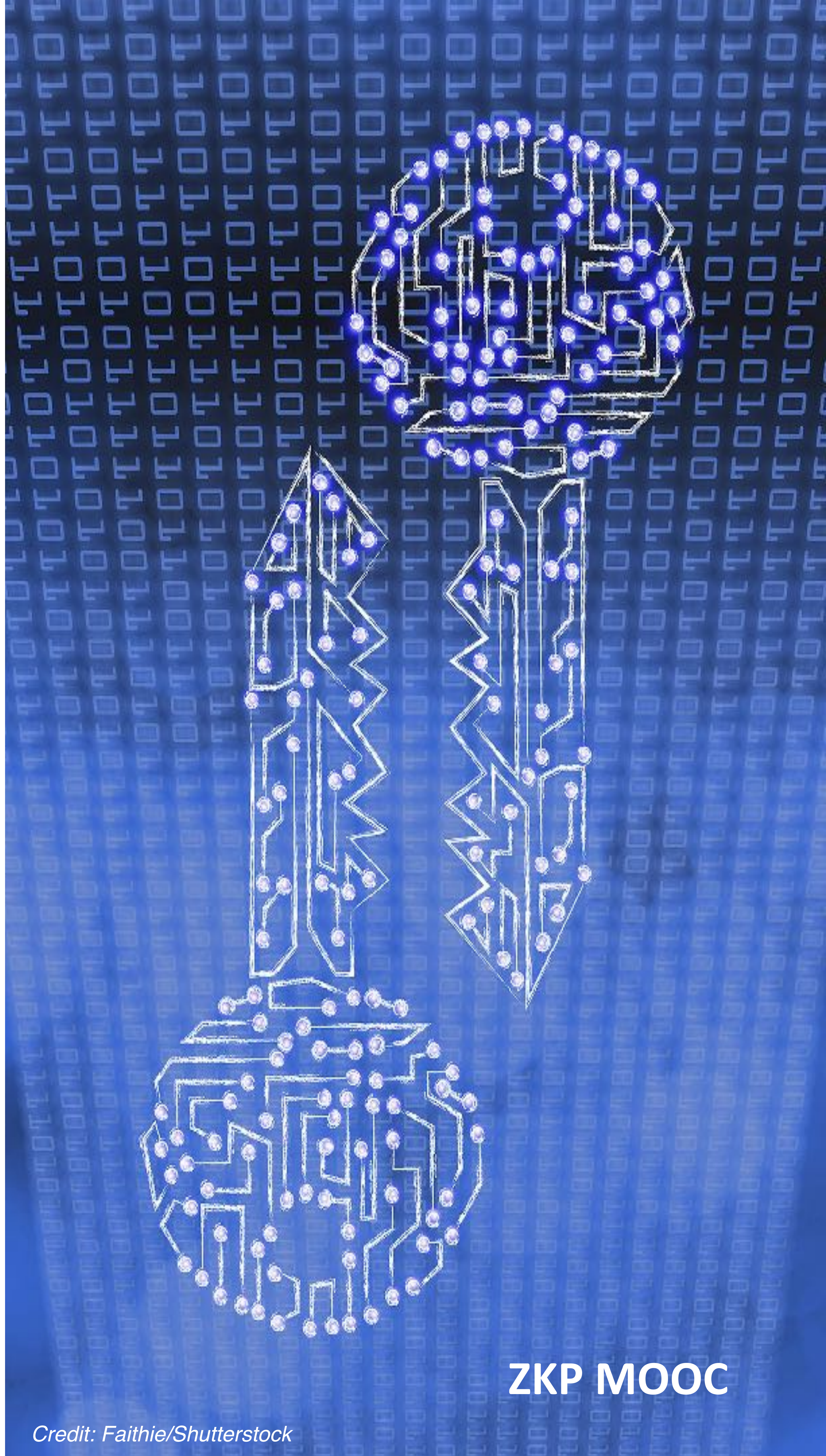
- The vulnerability was discovered by a cryptographer from Zcash Company in March 2018.
- Attackers could create fake Zcash coins in large numbers by exploiting this vulnerability.

Bug in arithmetic circuit implementing zkSNARK!

Formal Methods to Rescue

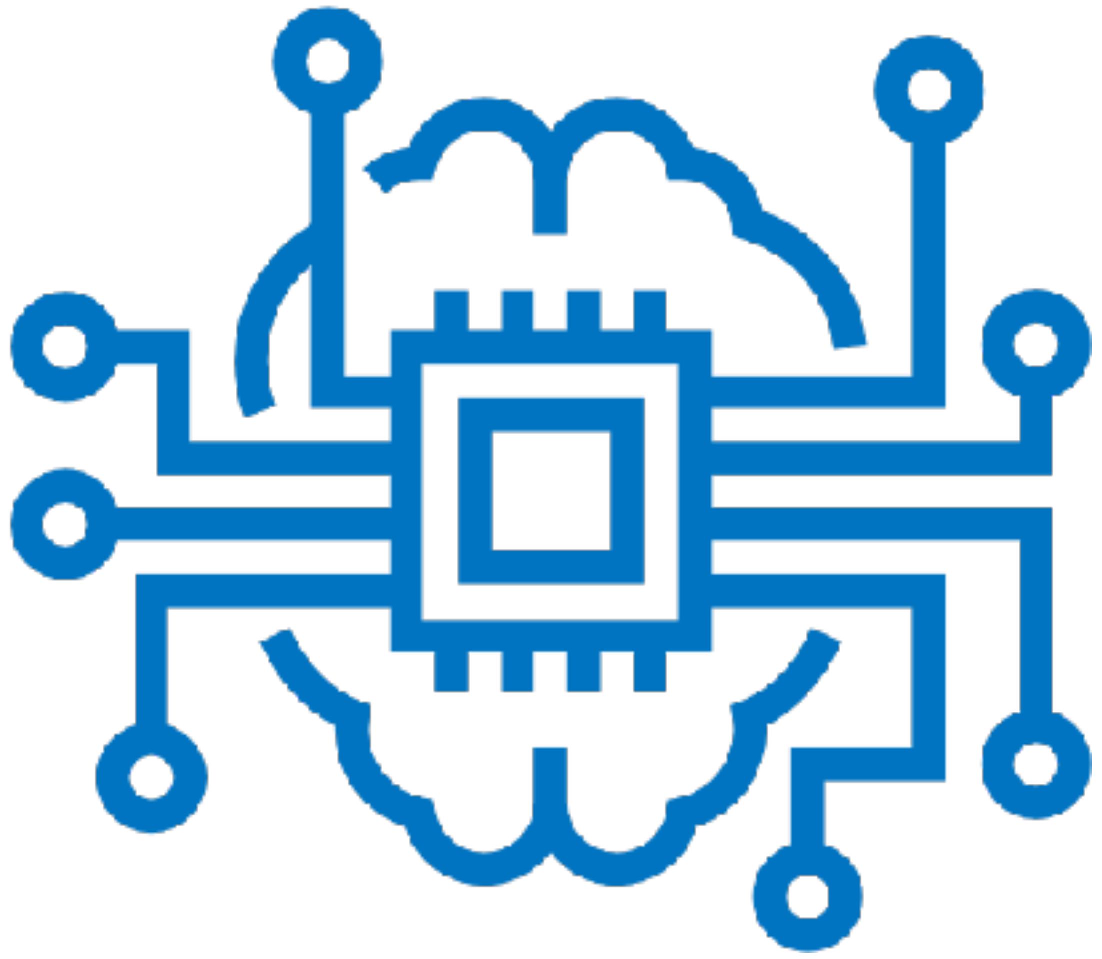
Section 1

Formal Methods in a Nutshell



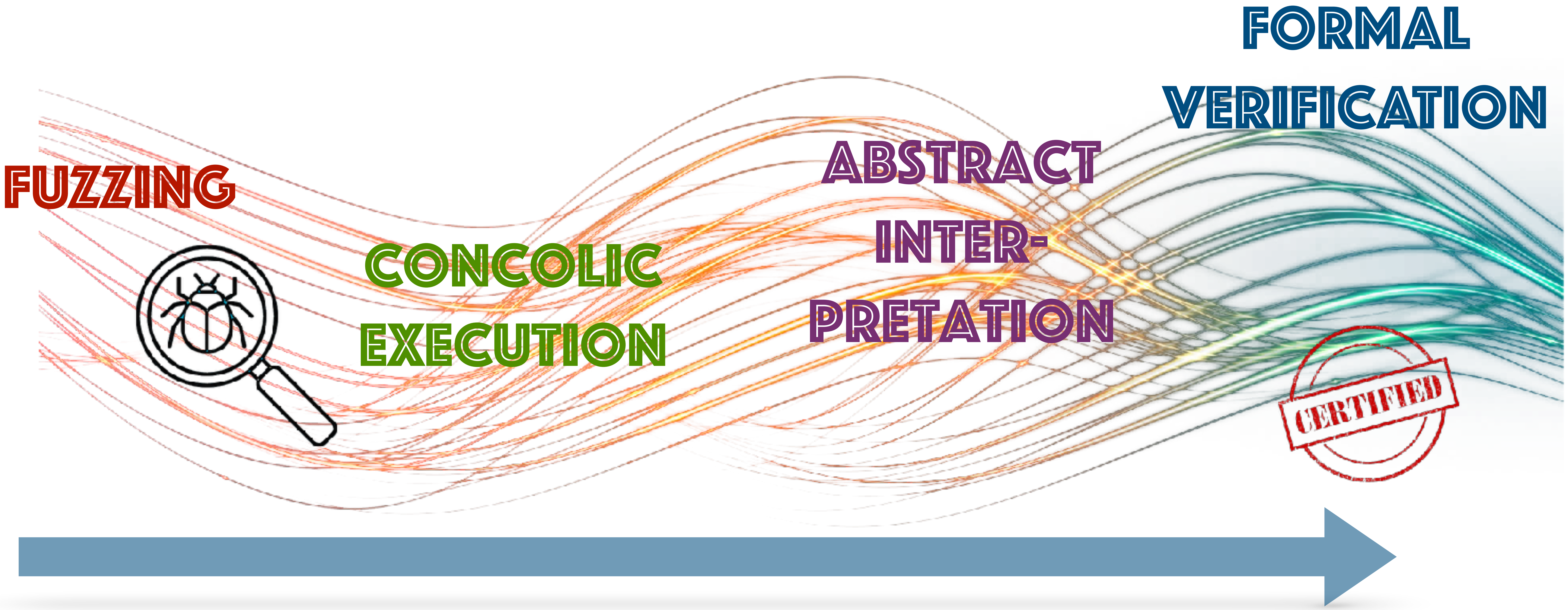
Credit: Faithie/Shutterstock

What is Formal Methods



Set of mathematically rigorous techniques for finding bugs and constructing proofs about software

Formal Methods Techniques on Spectrum



FUZZING

CONCOLIC EXECUTION

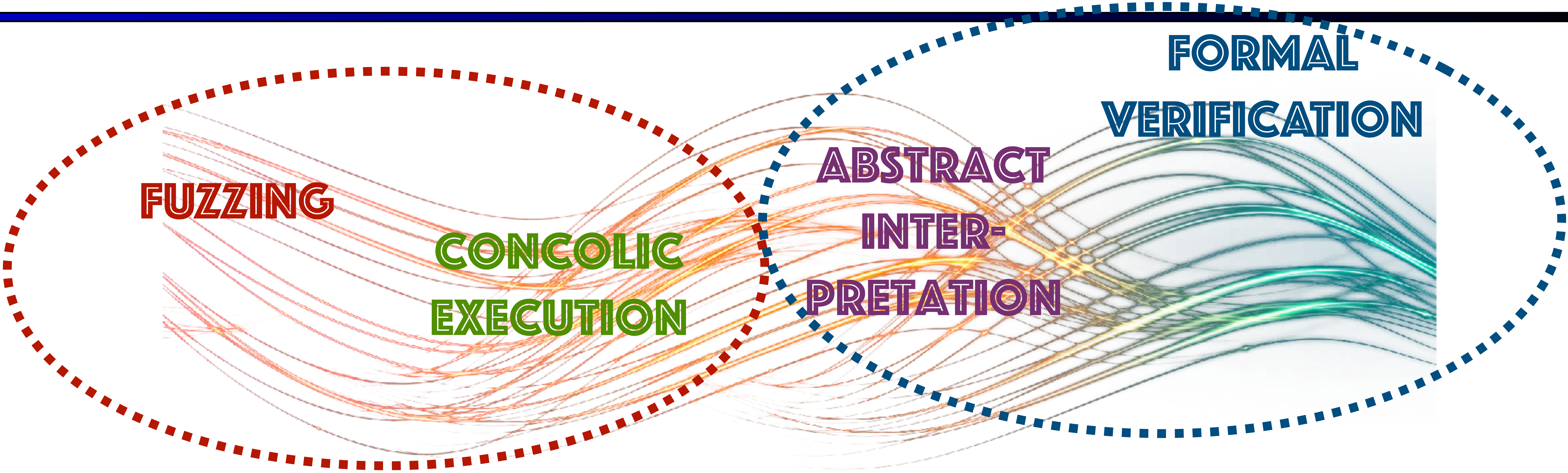
ABSTRACT INTERPRETATION

FORMAL VERIFICATION

Stronger guarantees

More human effort

Classification of FM Techniques



DYNAMIC

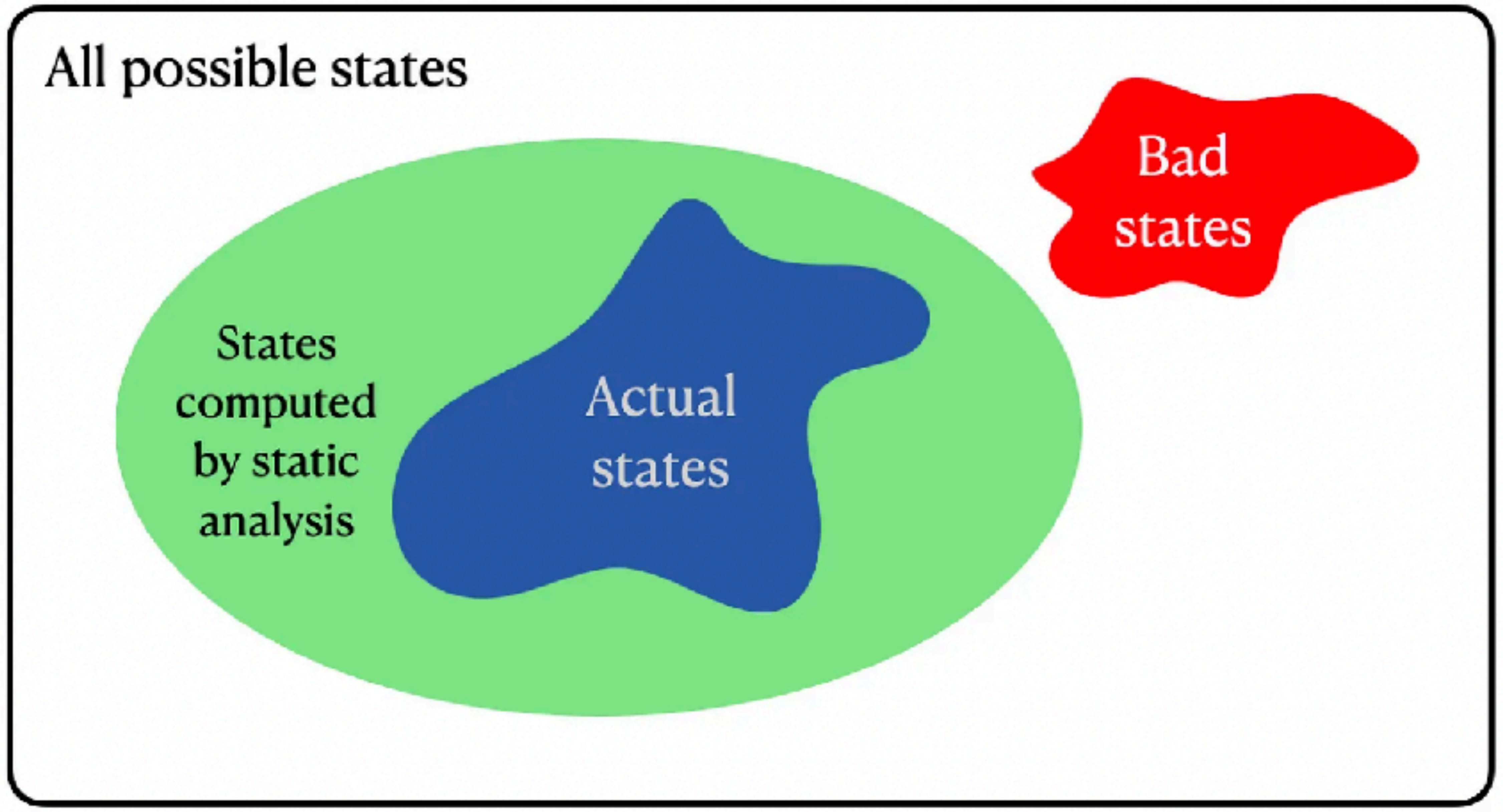
Execute the program on interesting inputs & monitor what happens

STATIC

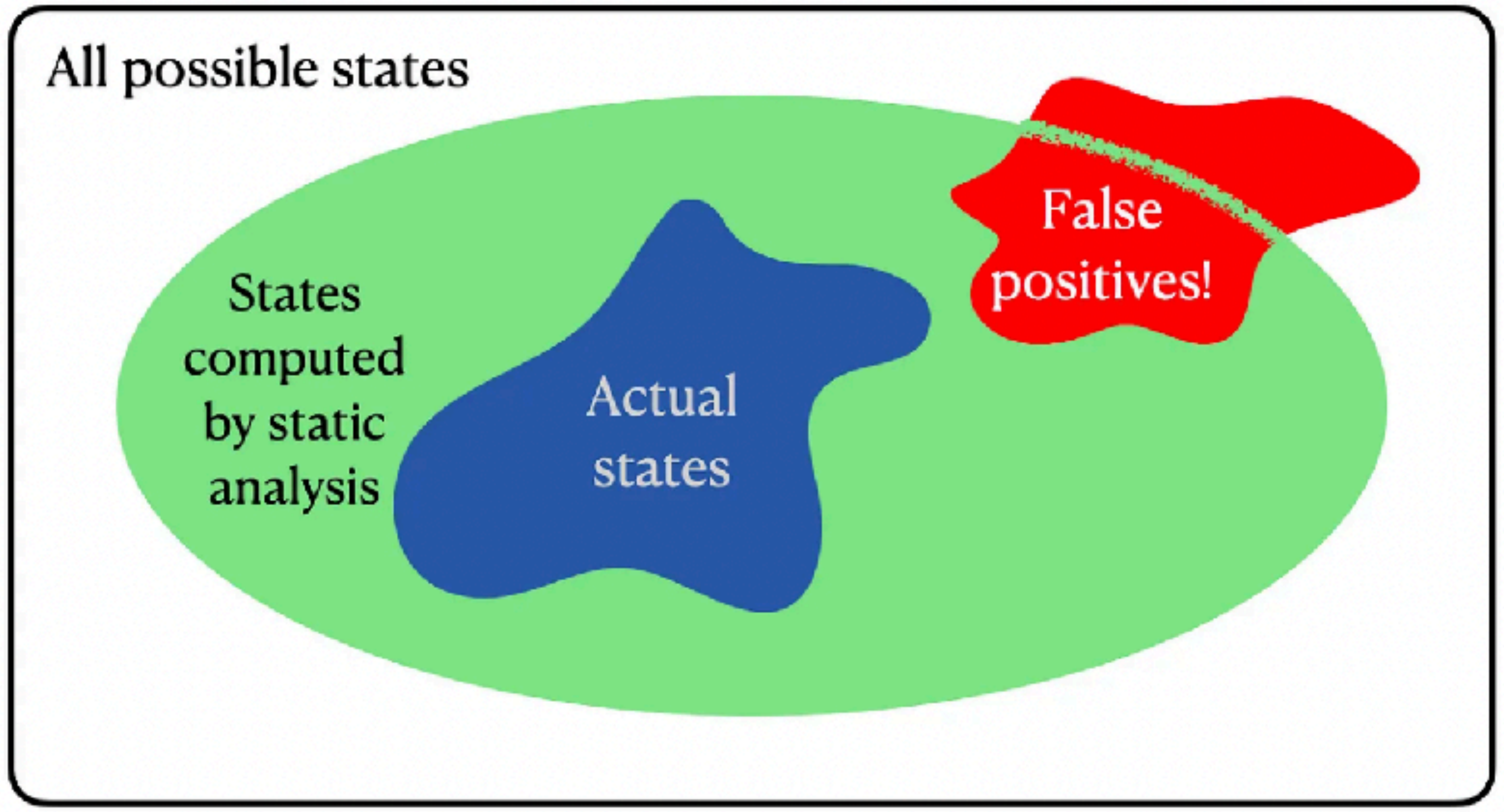
Analyze source code and reason about all executions

Fundamentals of Static Analysis

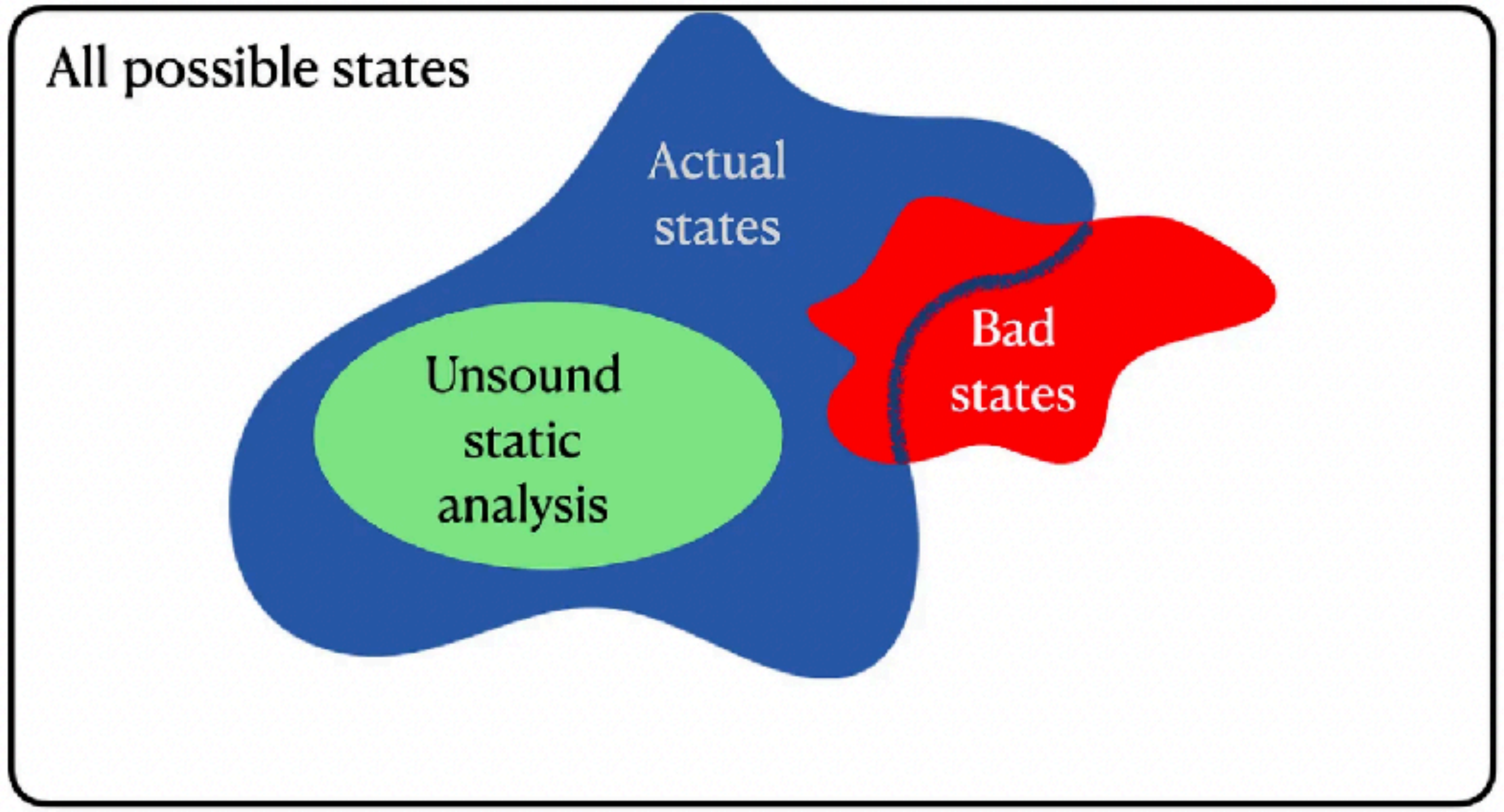
- Blue irregular shape is the actual states
- Red region corresponds to “bad states”
- Due to undecidability, we can never determine exactly what the blue region is
- Over-approximate blue region with the regular green region above



Fundamentals of Static Analysis

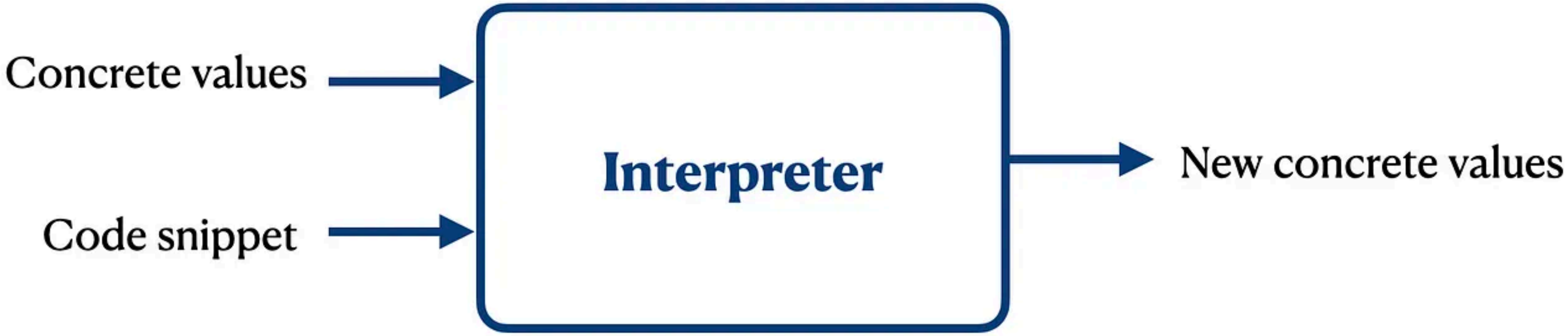


False Positives



False Negatives

Concrete Interpretation is Easy



Concrete values

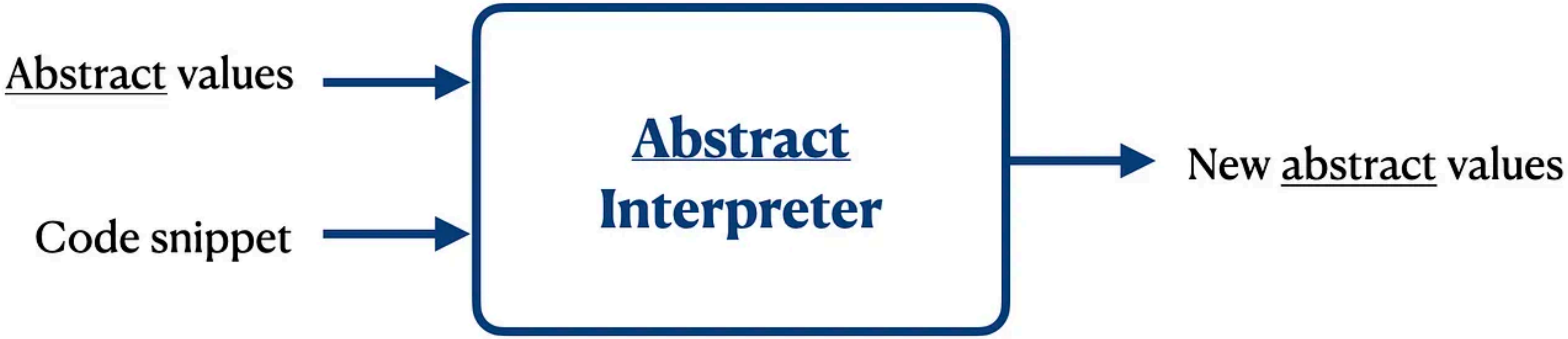
Code snippet

Interpreter

New concrete values

If f(x) = x+2, then f(1) = 3

Static Analysis via Abstract Interpretation



Static Analysis via Abstract Interpretation

Idea: Emulate all possible program paths

```
if(flag)
  x = 1;
else
  x = -1;
```

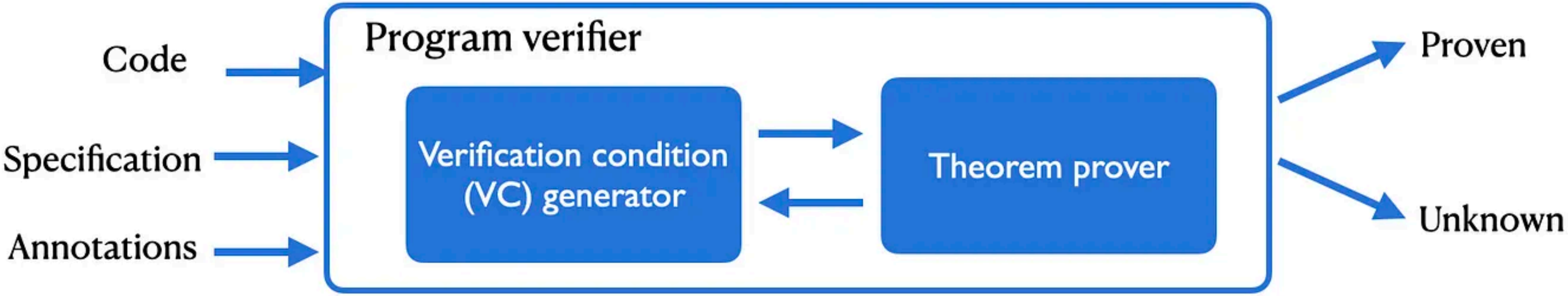
When in doubt, conservatively assume either path could be taken and merge information for different paths

$$x \in [-1,1]$$

Abstract Interpretation Tools in Web3

- Slither (TrailOfBits)
- Sailfish (Bose et al, Oakland’22)
- Vanguard (Veridise)

Static Analysis via Formal Verification



- Program implementation: Source code of the program, or intermediate representation
- The specification: A formal description of the property to be verified
- Human annotations (optional): Loop invariants, Contract invariants

Formal Specifications

<u>Formal specification:</u> Precise mathematical description of intended program behavior, typically in some formal logic

$$
\begin{aligned}
\Box\ \ &((\mathit{finish}(\mathit{bid}, \mathit{msg}\,.\,\mathit{value} = X \land \mathit{msg}\,.\,\mathit{sender} = L)) \\
&\quad \land \Diamond\ \mathit{finish}(\mathit{close}, L \neq \mathit{winner}) \\
&\quad \rightarrow \Diamond\ \mathit{send}(\mathit{to} = L \land \mathit{amt} = X))
\end{aligned}
$$

If auction closes with me not being the winner, I should eventually get back my bid

Formal Verification Tools in Web3

- Certora prover (Certora)
- K framework (Runtime Verification)

Different Flavors of Static Analysis

Formal verification checks program against provided specification

Abstract Interpretation

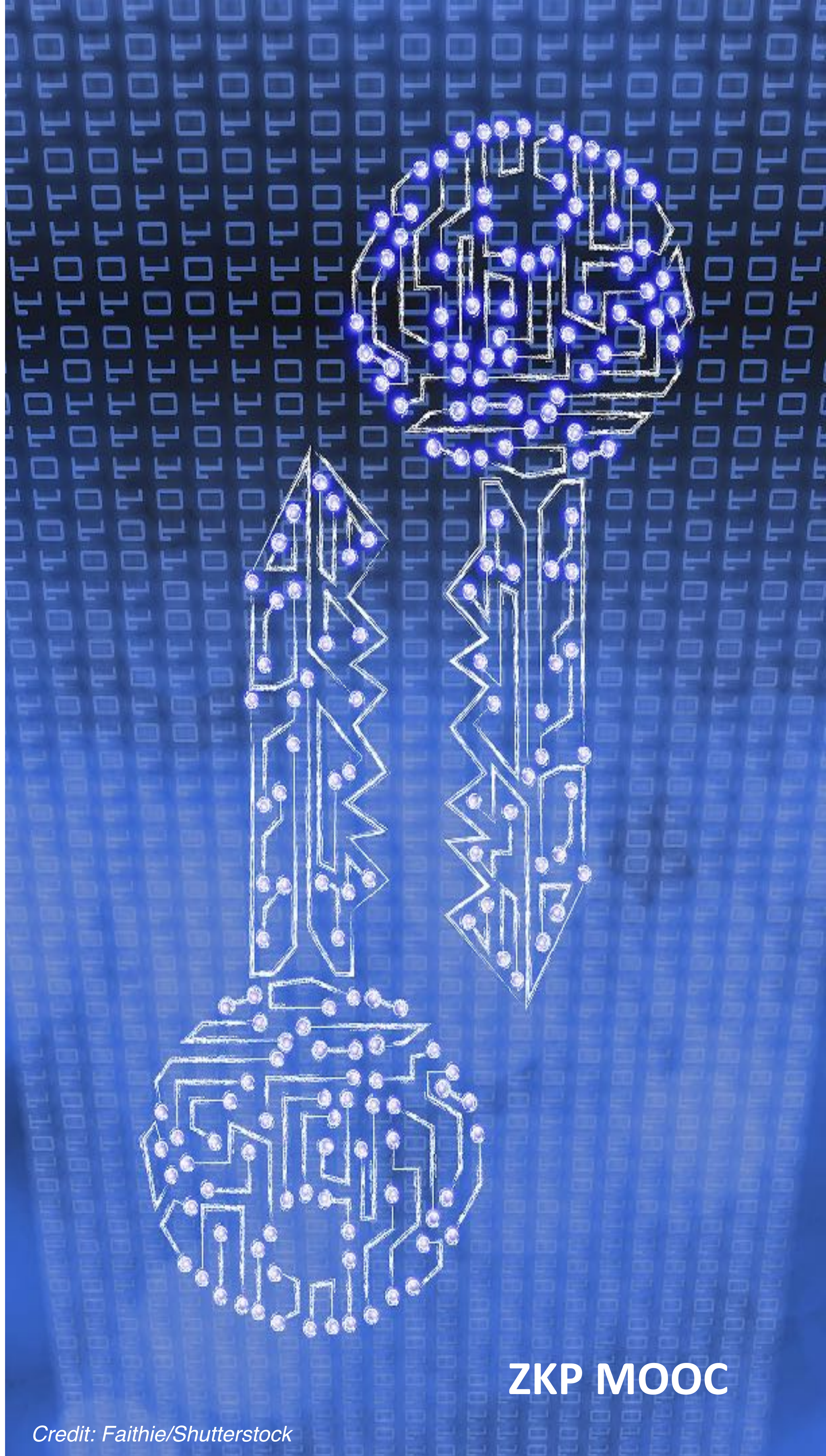
- ✎ Looks for known types of bugs
- ✓ Doesn’t require specifications

Formal Verification

- ✓ Can find (prove absence of) any bug
- ✎ Requires specifications

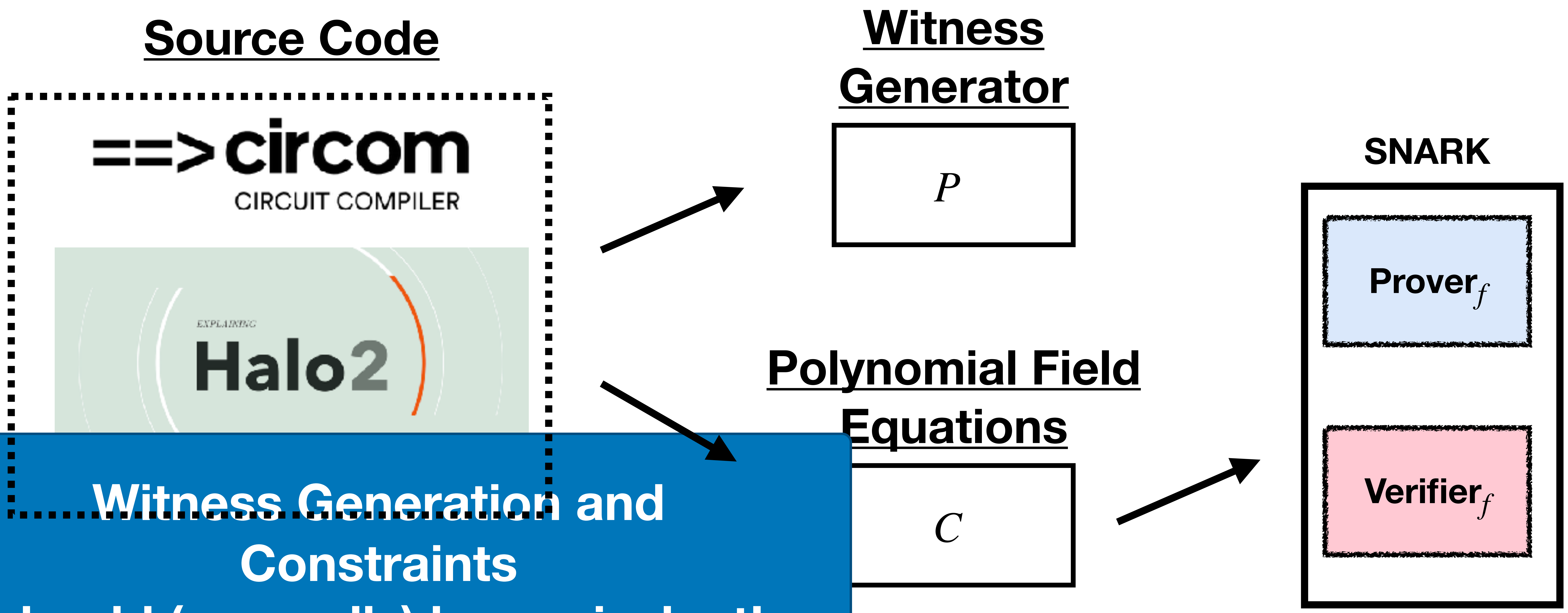
Section 2

Formal Methods in ZK: part I



Credit: Faithie/Shutterstock

Circuits Workflow



Witness Generation and Constraints should (generally) be equivalent!

Source Code: Witness Generation and Constraints

What is Equivalence

Program: P	**Set of Constraints**: C
Input: x	**Inputs**: x, y
Output: y	**Output**: $true$ or $false$

For every x, y. $P(x) = y$ *if and only if* $C(x, y)$ *is* $true$

Every input-output of *P* must satisfy *C*

Every *(x,y)* which satisfy *C* must be an input-out pair of *P*

?

How can this be violated?

Equivalence Violations

Two Requirements:

(1) Every input-out pair of P satisfies C

(2) For any x, y which satisfy C, $P(x) = y$

Overconstrained Bugs

Exists x, y where $P(x) = y$ but $C(x, y)$ is $false$

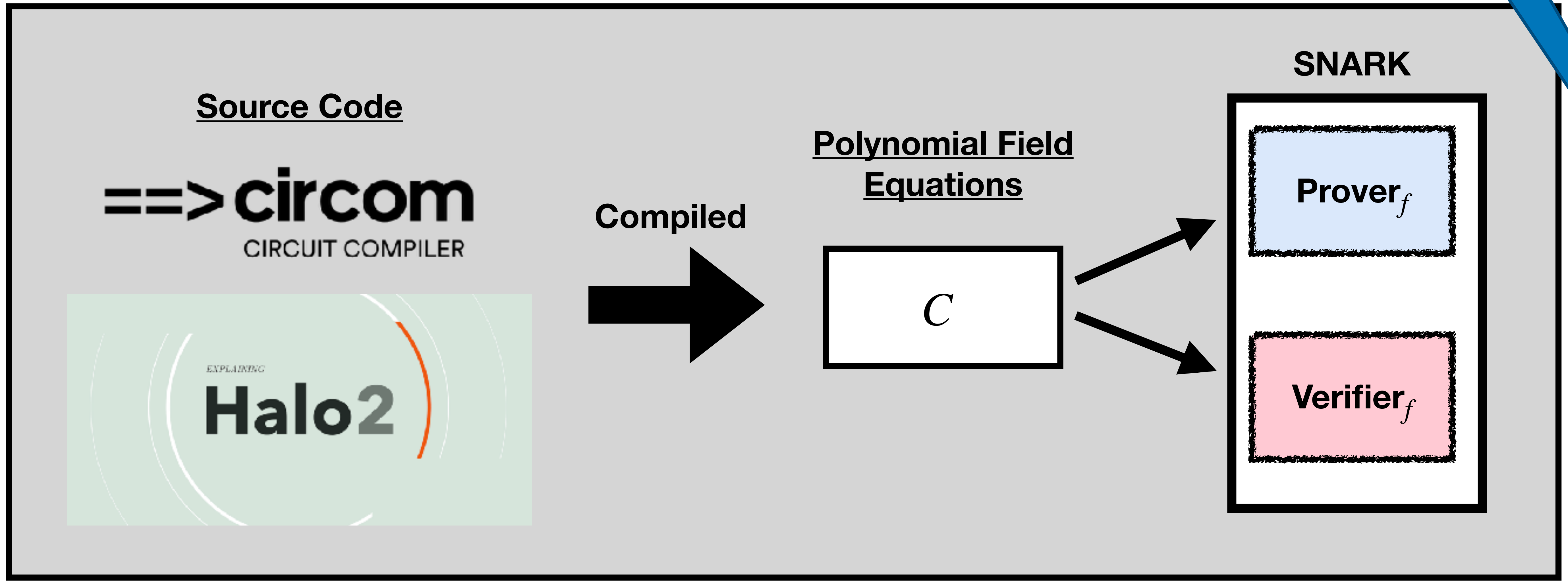
Most ZK languages (e.g., Circom, Halo2) add field equations as assertions to circuit!

Underconstrained Bugs

Exists x, y where $C(x, y)$ is $true$ but $P(x) \neq y$

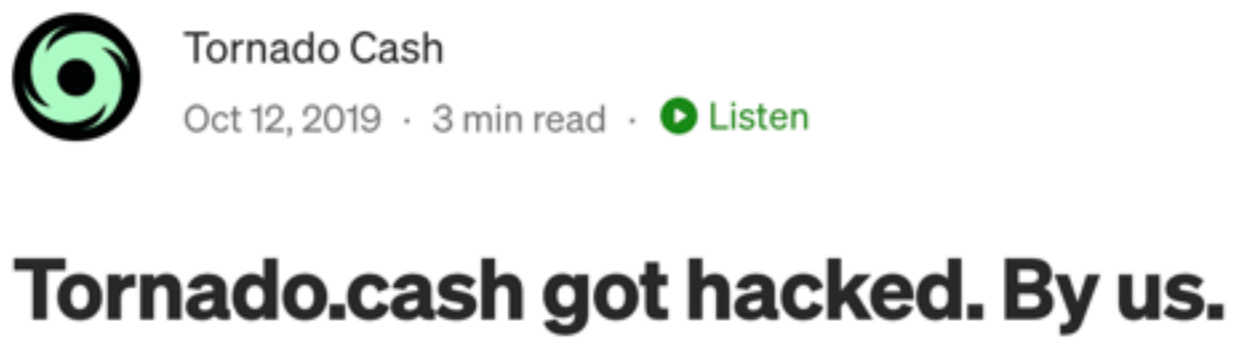
Why Do We Care

ZK Circuit Workflow



Could be used to drain all tokens

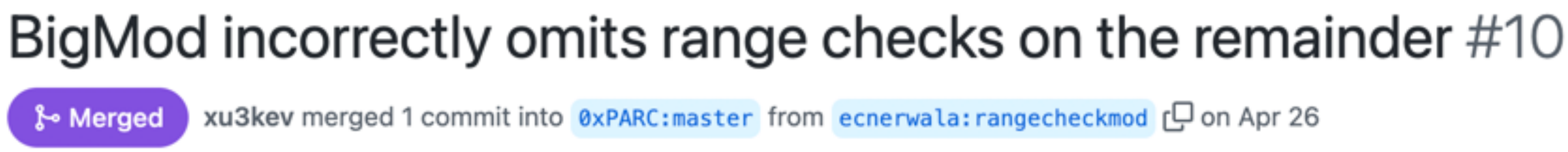
Underconstrained bugs: Verifier can accept bad inputs/ outputs



Tornado Cash

Oct 12, 2019 · 3 min read · Listen

Tornado.cash got hacked. By us.



BigMod incorrectly omits range checks on the remainder #10

Merged xu3kev merged 1 commit into `0xPARC:master` from `ecnerwala:rangecheckmod` on Apr 26

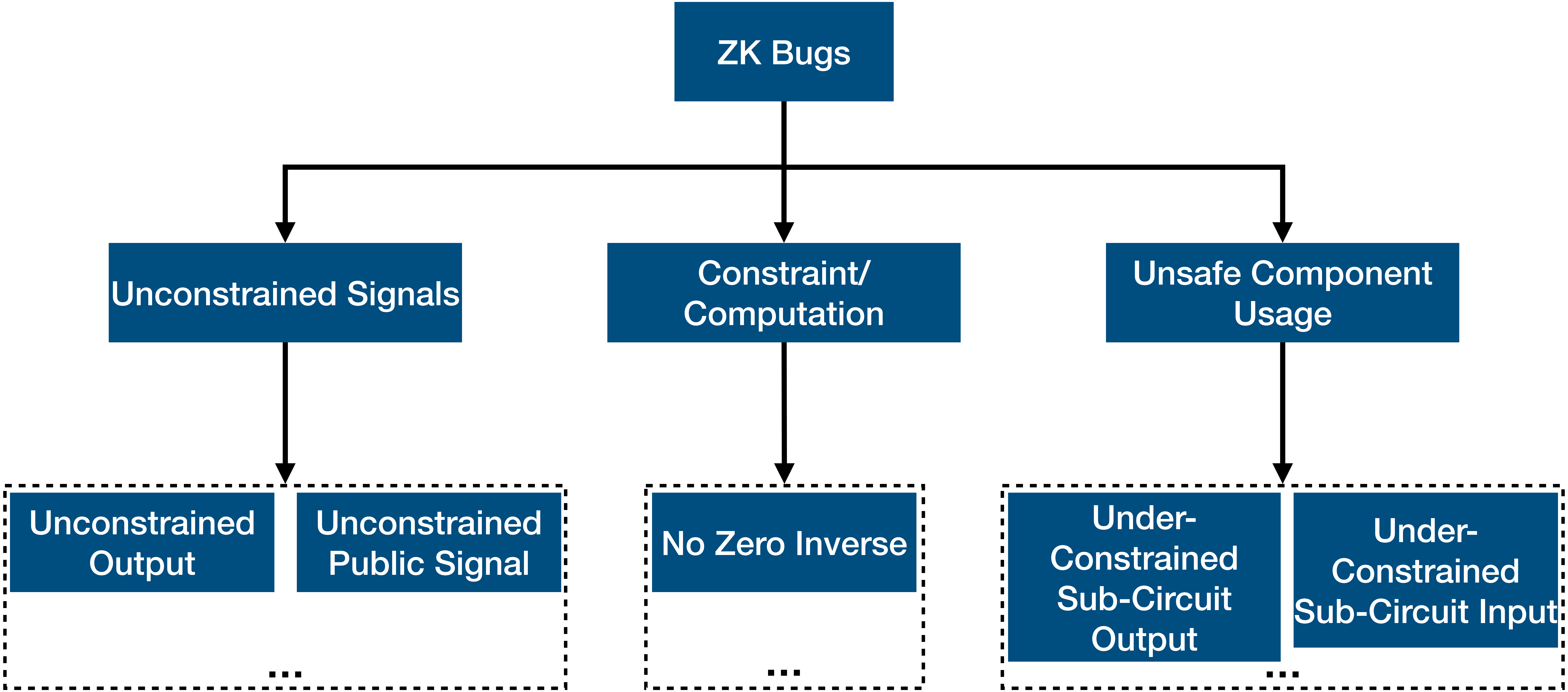
Double spend

Disclosure of recent vulnerabilities

We have recently patched two severe bugs in Aztec 2.0. The first was found by an Aztec engineer and the second by community members.

1. Lack of range constraints for the `tree_index` variable

A Taxonomy of ZK Bugs



ZK Bugs

Unconstrained Signals

Constraint/ Computation

Unsafe Component Usage

Unconstrained Output

Unconstrained Public Signal

...

No Zero Inverse

...

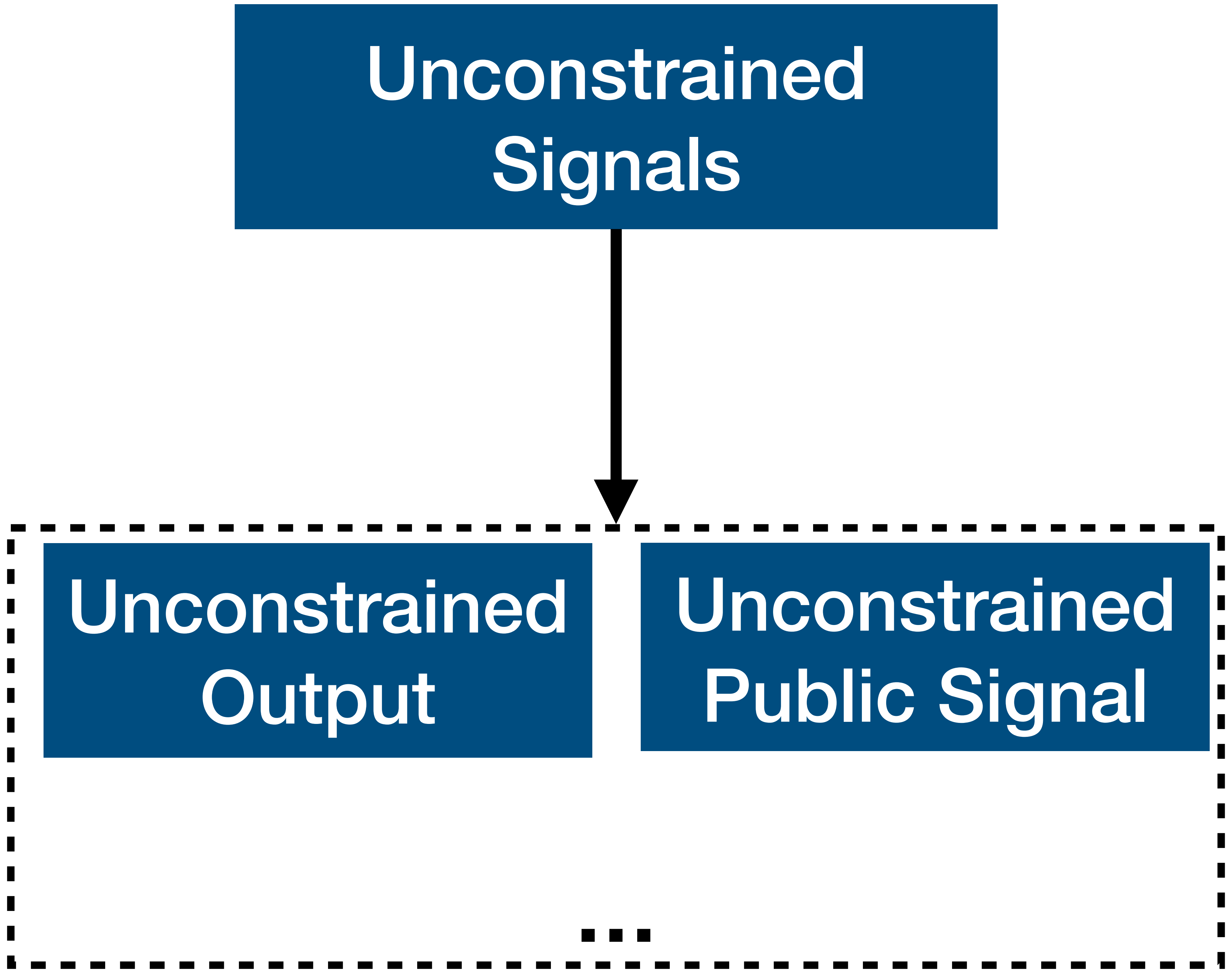
Under-Constrained Sub-Circuit Output

Under-Constrained Sub-Circuit Input

...

Unconstrained Signals

Corresponds to signals whose constraints always evaluate to *true*, accepting everything



Underconstrained Output

Buggy Implementation

```
template Num2Bits(n) {
    signal input in;
    signal output out[n];
    var lc1 = 0;

    var e2 = 1;
    for (var i = 0; i < n-1; i++) {
        out[i] <-- (in >> i) & 1;
        out[i] * (out[i] - 1) === 0;
        lc1 += out[i] * e2;
        e2 = e2 + e2;
    }

    lc1 === in;
}
```

Developer added constraints

Constraints for $n = 3$

out_2 is underconstrained

$$\begin{cases} \text{input } in \\ \text{output } out_0, out_1, out_2 \\ out_0 \cdot (out_0 - 1) = 0 \\ out_1 \cdot (out_1 - 1) = 0 \\ out_0 + 2 * out_1 = in \end{cases}$$

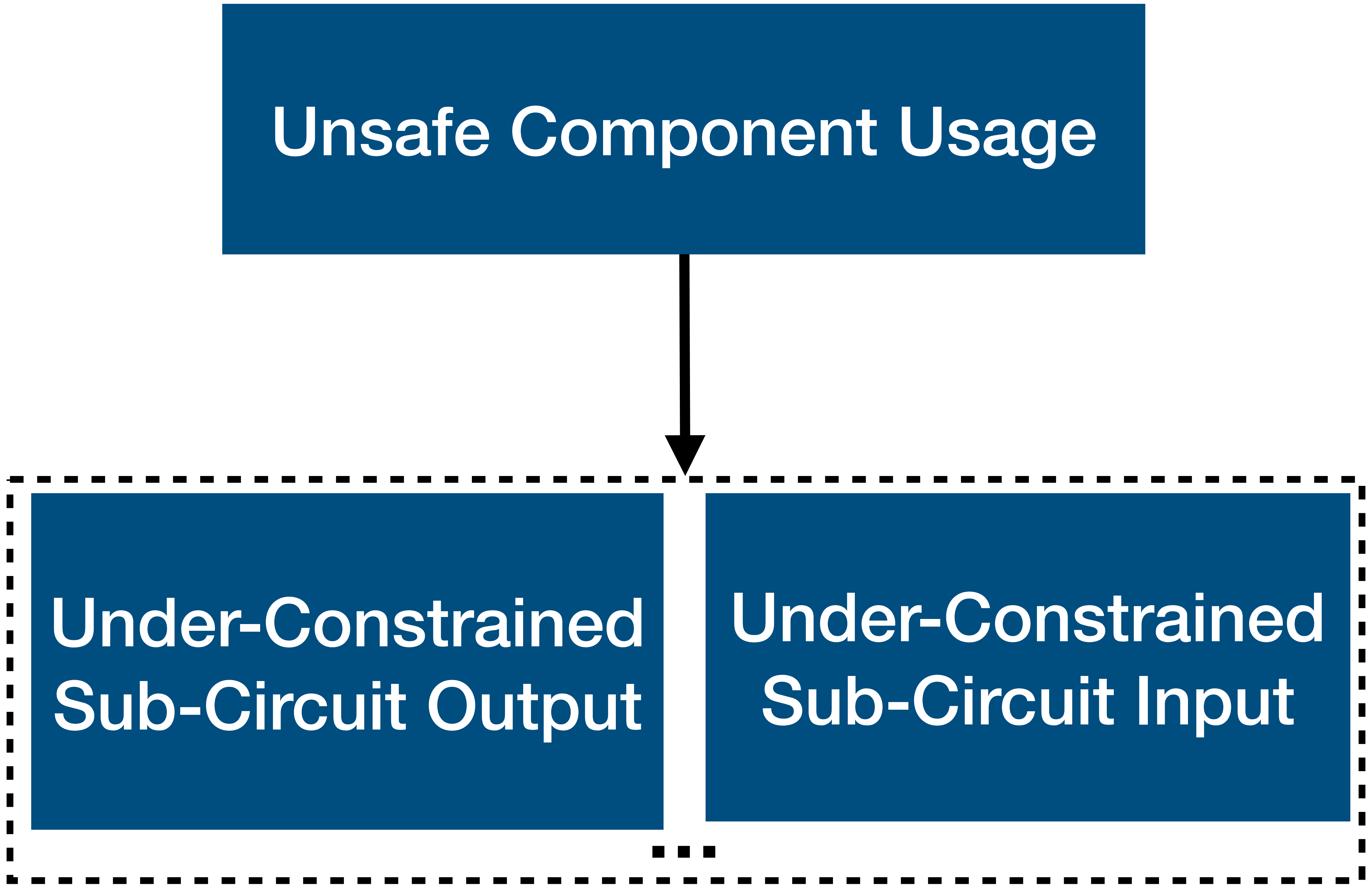
Attacker can pass in any value for out_2

https://github.com/iden3/circomlib/blob/master/circuits/bitify.circom

Unsafe Component Usage

Sub-circuits often assume constraints are placed on inputs and outputs

Corresponds to cases where the use of a sub-circuit do not follow



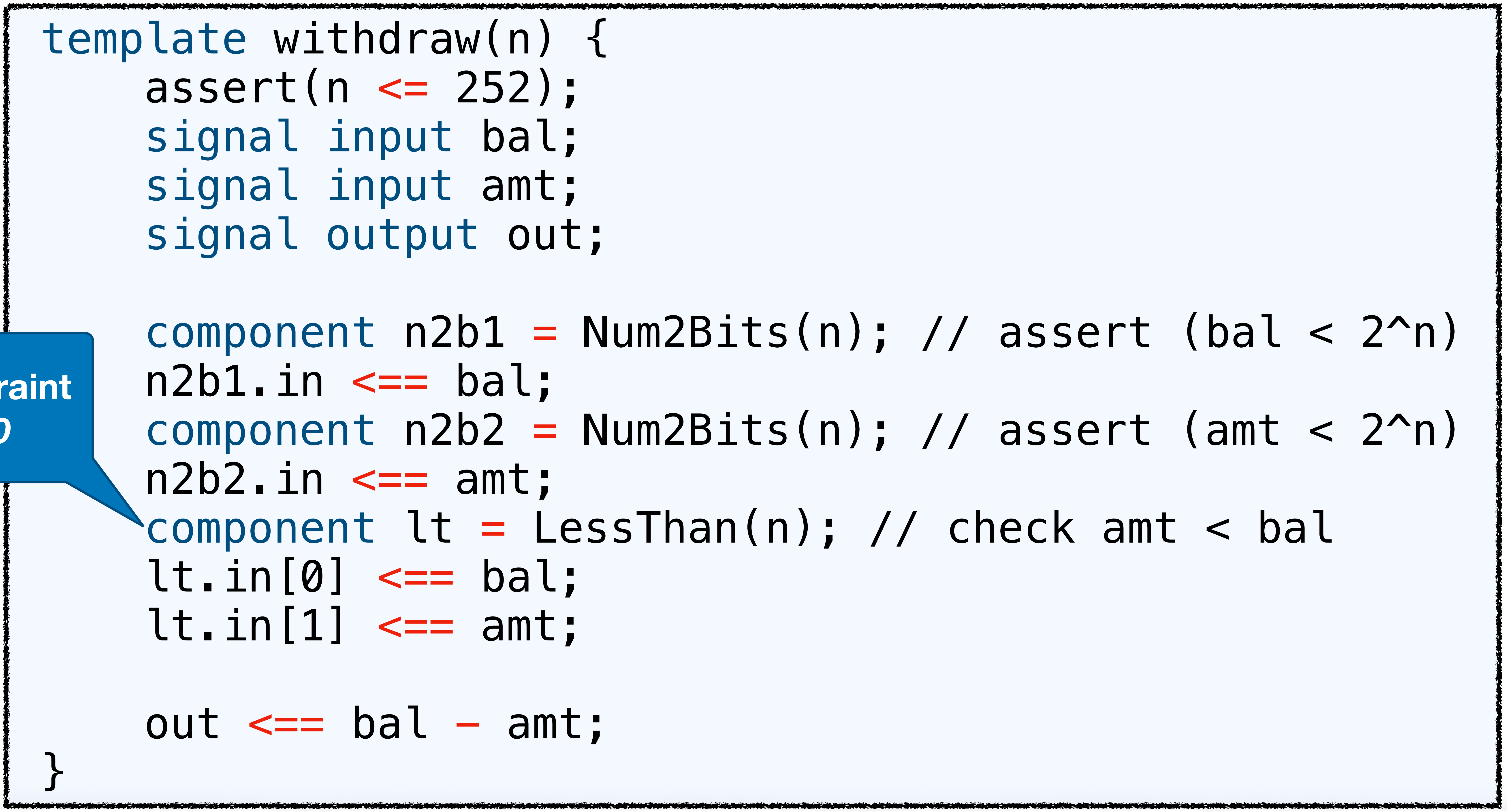
Unsafe Component Usage

Under-Constrained Sub-Circuit Output

Under-Constrained Sub-Circuit Input

...

Example: Under-Constrained Sub-Circuit Output



```
template withdraw(n) {
    assert(n <= 252);
    signal input bal;
    signal input amt;
    signal output out;

    component n2b1 = Num2Bits(n); // assert (bal < 2^n)
    n2b1.in <== bal;
    component n2b2 = Num2Bits(n); // assert (amt < 2^n)
    n2b2.in <== amt;
    component lt = LessThan(n); // check amt < bal
    lt.in[0] <== bal;
    lt.in[1] <== amt;

    out <== bal - amt;
}
```

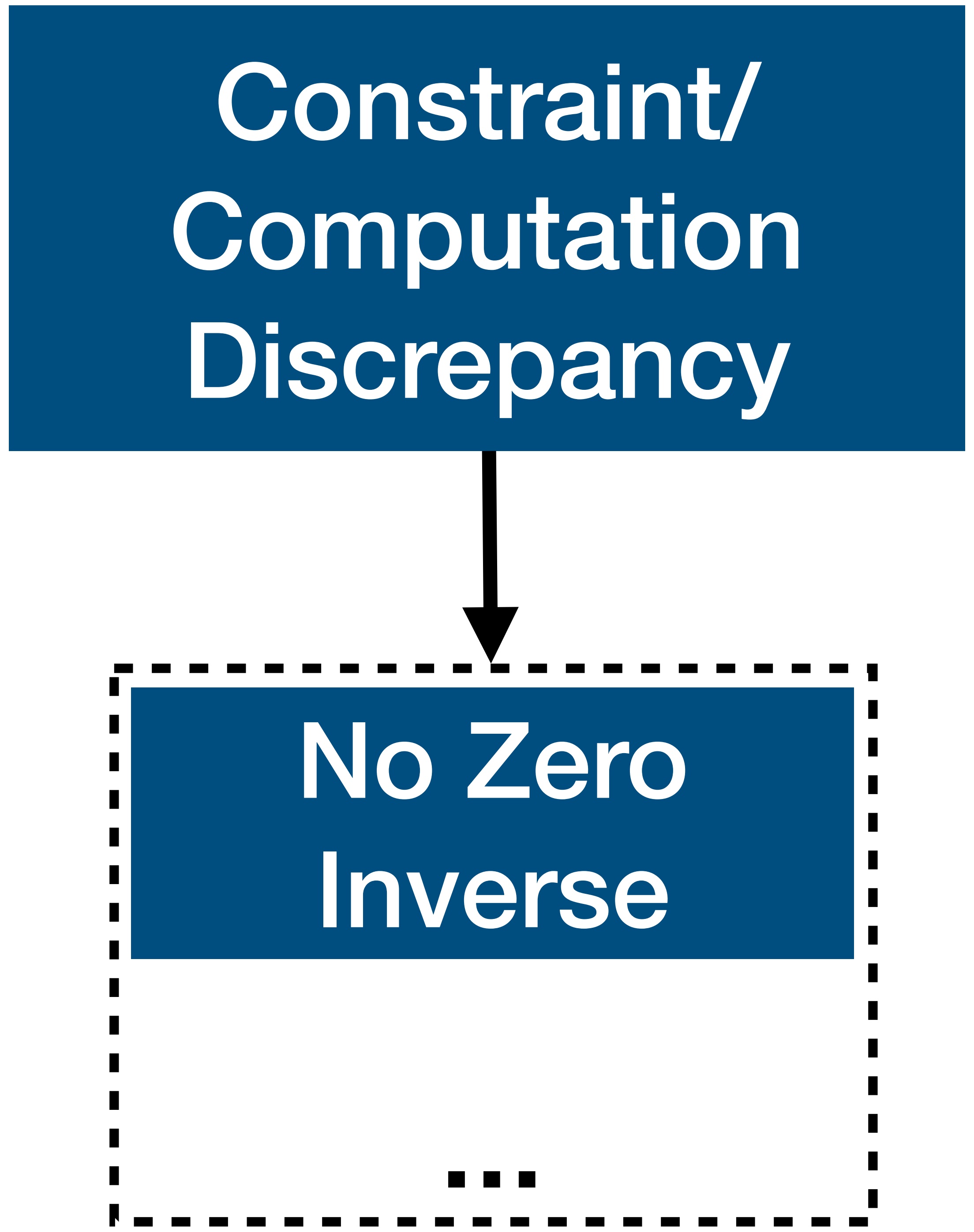
Missing constraint
lt.out === 0

Without the missing constraint, attacker can withdraw more funds than they have

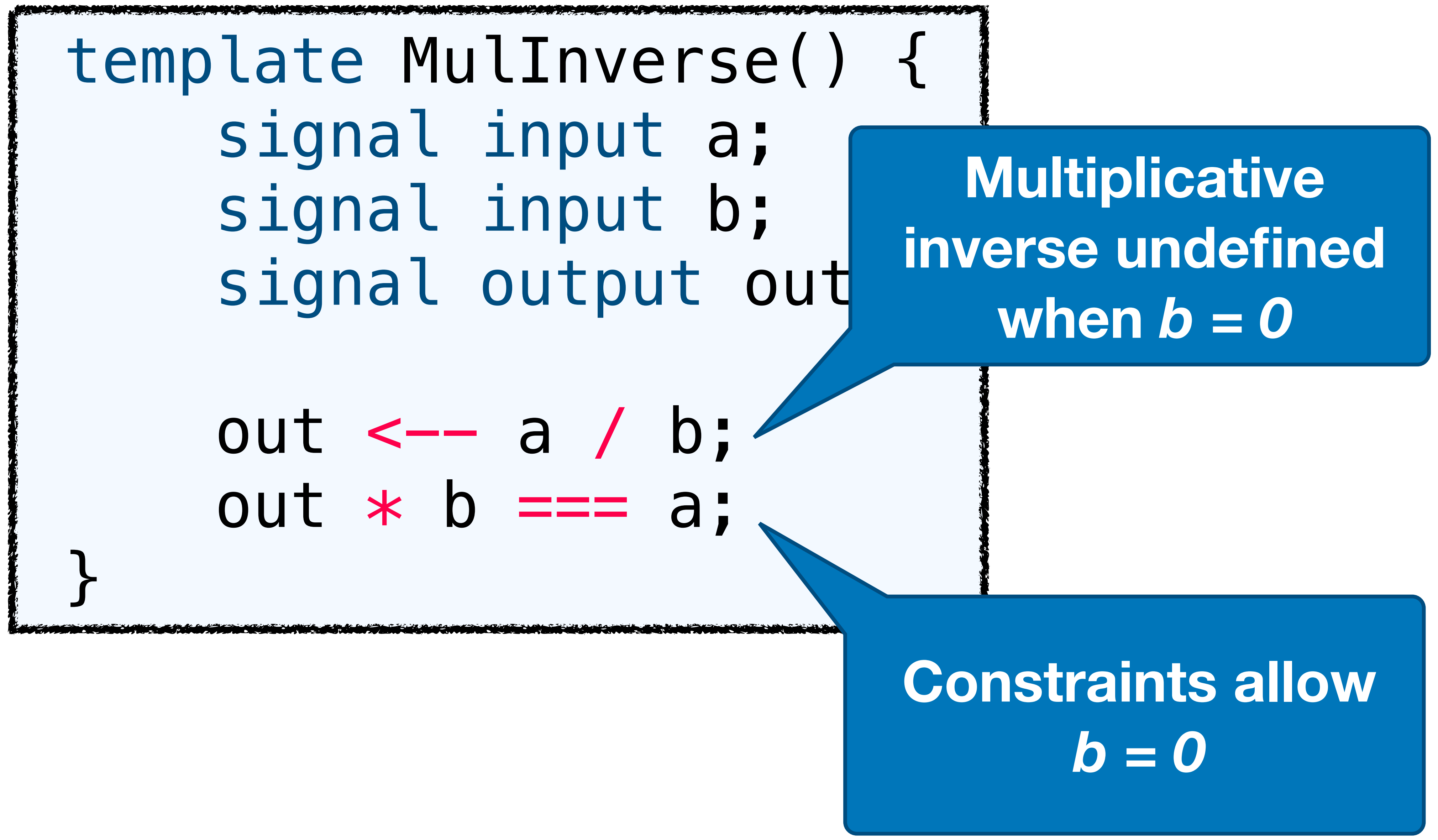
Constraint/Computation Discrepancy

Not all computation can be directly expressed as a constraint

Corresponds to constraints that do not capture a computation's semantics



Example: No Zero Inverse



```
template MulInverse() {
    signal input a;
    signal input b;
    signal output out

    out <-- a / b;
    out * b === a;
}
```

Multiplicative inverse undefined when *b = 0*

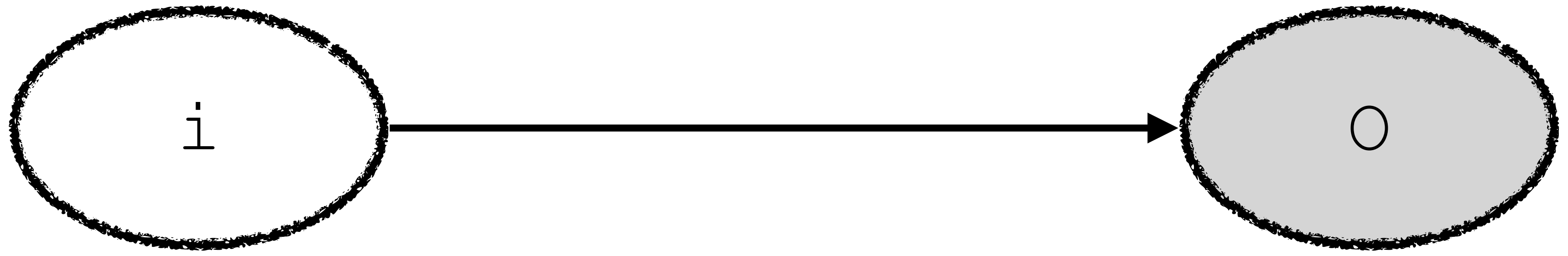
Constraints allow *b = 0*

Accepts arbitrary *out* when *a* and *b* are 0!

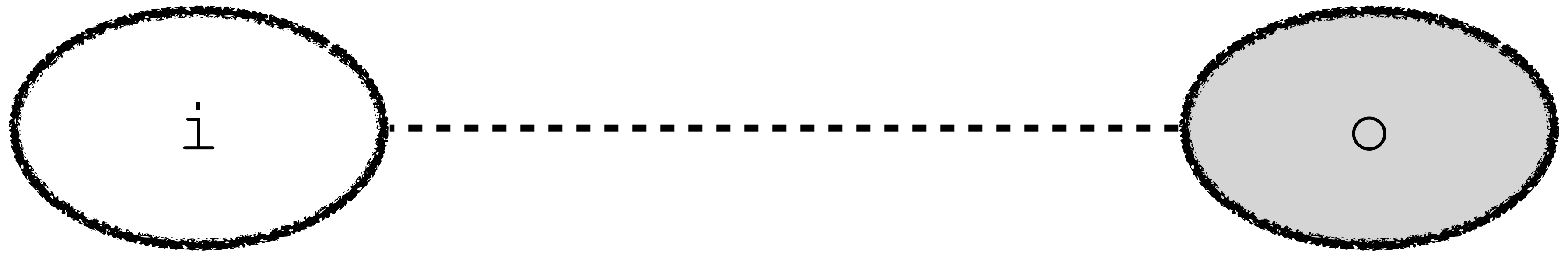
Circuit Dependence Graphs (CDG)

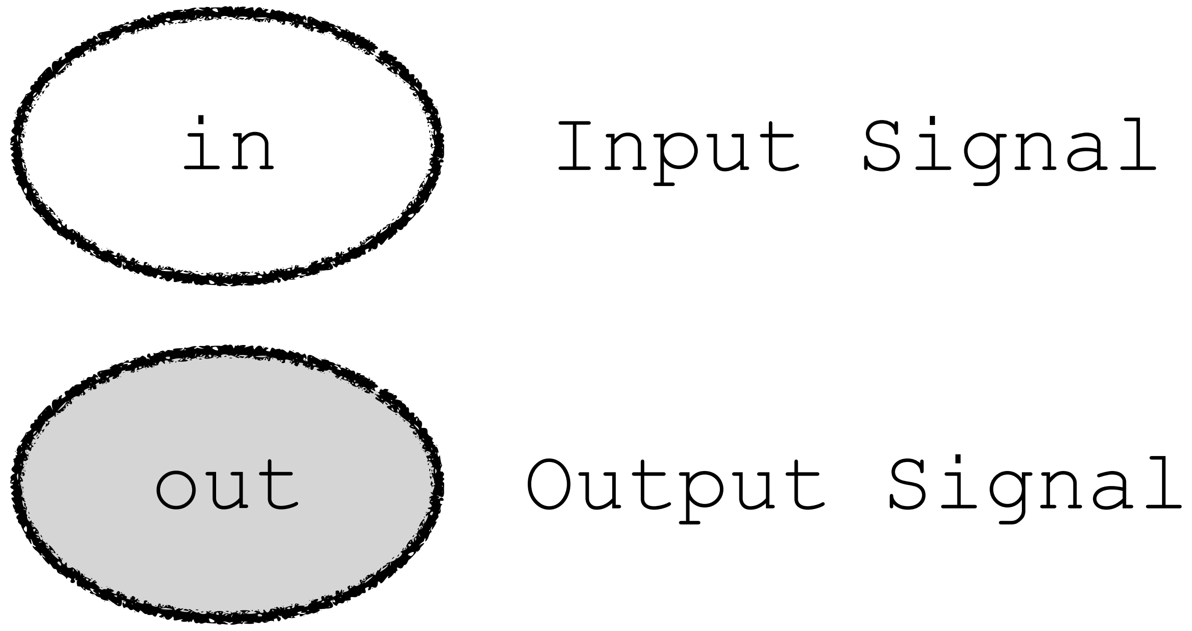
***Goal:** Identify discrepancies between computation and constraints*

```
o <-- i
```

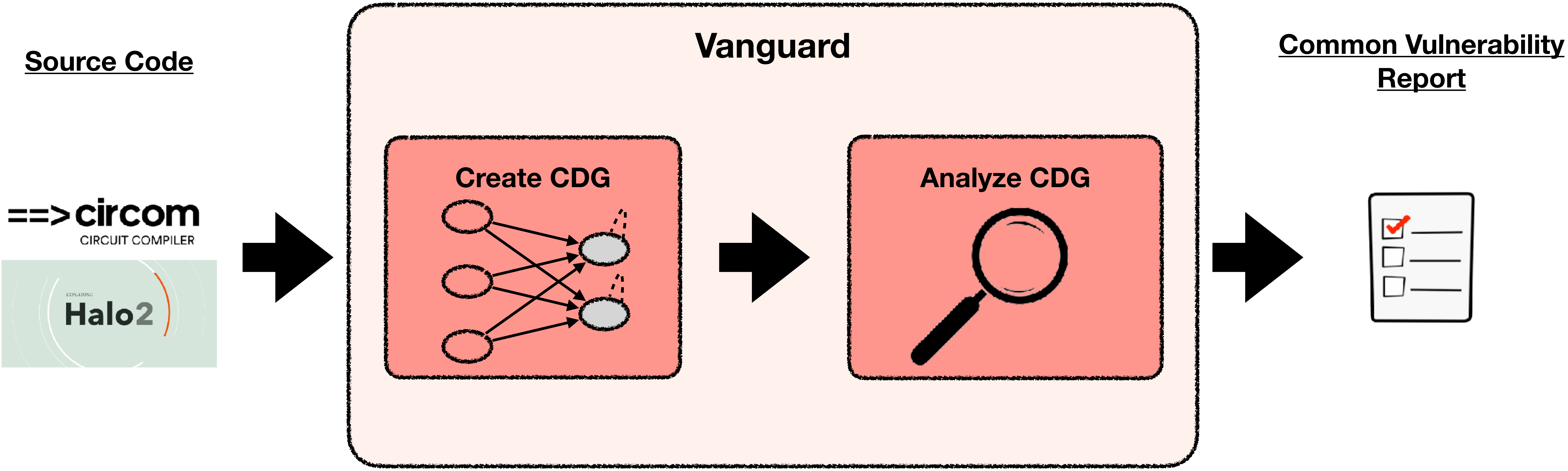


```
o === i
```





Vanguard Static Analysis



Source Code

Vanguard

Create CDG

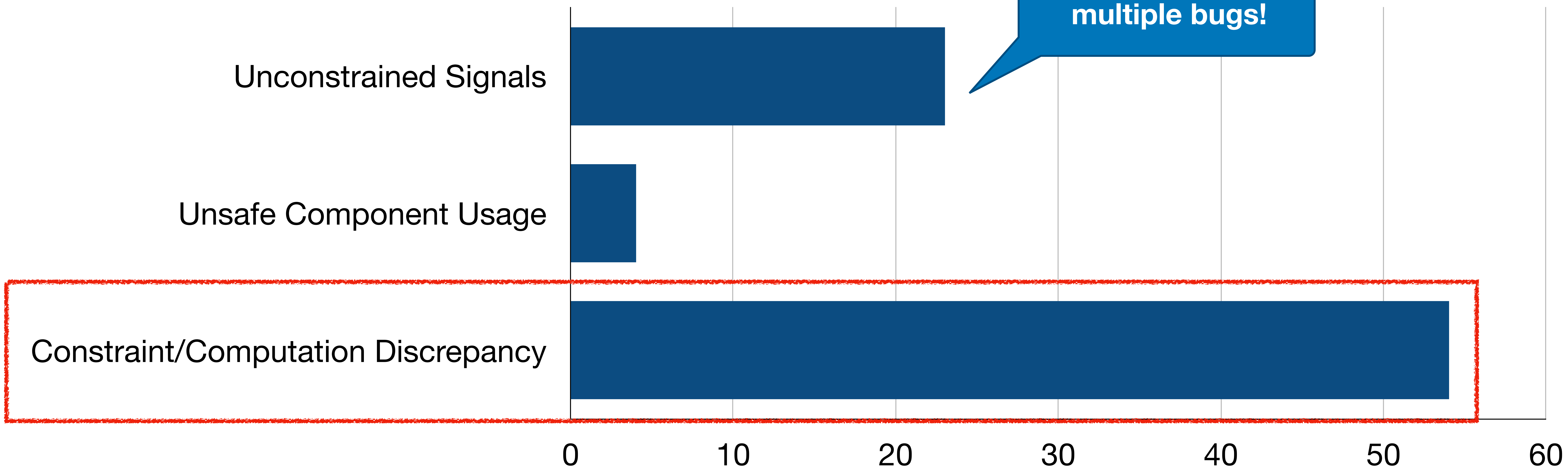
Analyze CDG

Common Vulnerability Report

Used to evaluate **258** circuits from **17** public Circom projects on Github

Evaluation Results

Identified **32** previously unknown vulnerabilities!

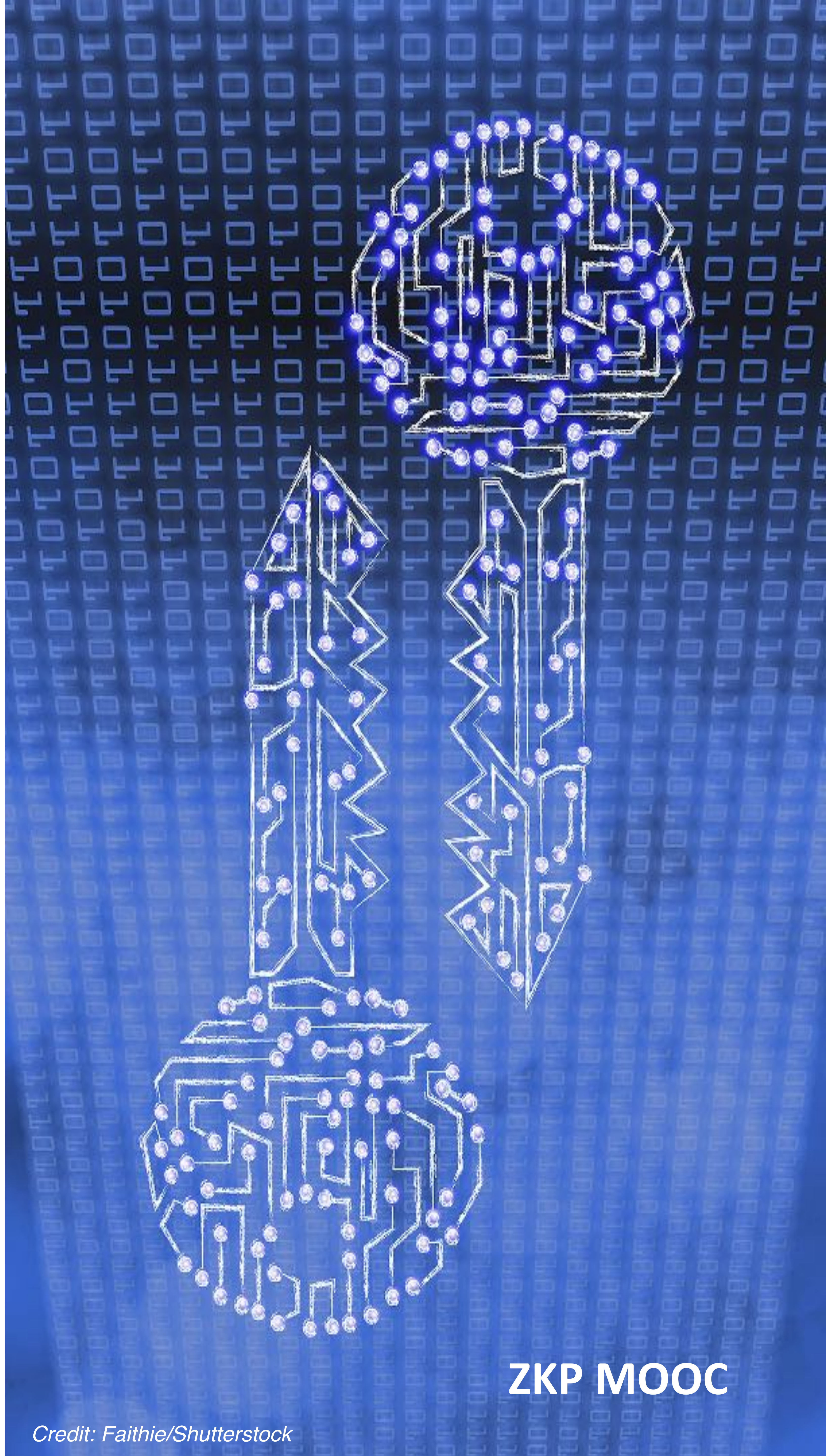


Some Circuits had multiple bugs!

Developers have the most difficulty reasoning about a computation's semantics!

Section 3

Formal Methods in ZK: part II



Credit: Faithie/Shutterstock

Existing Strategies

Static Analysis of Constraints (SA)

Apply predefined rules to quickly detect if circuit is properly constrained

$$\begin{cases} \text{input } x \\ \text{output } y \\ z = 3x + 4 \\ y = z + 2x \end{cases}$$

Since y is linear in x, z we immediately infer it is not under constrained

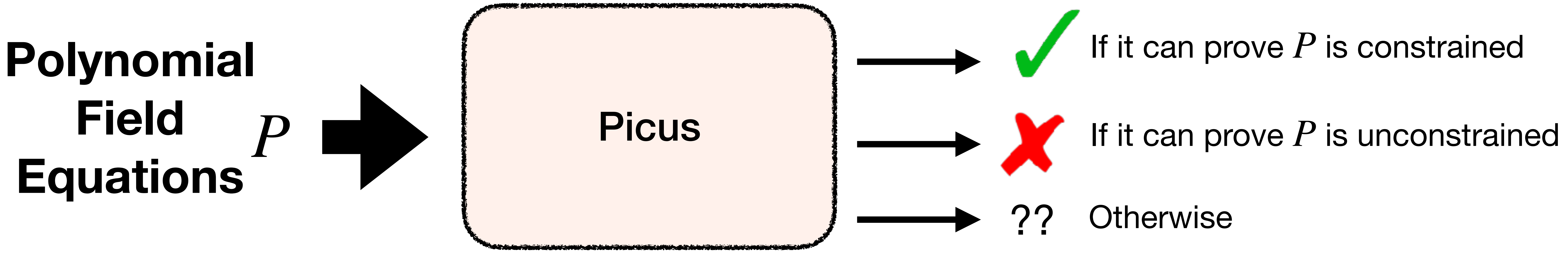
SMT Solver

Underconstrained can be expressed as SMT query

$$\exists y_1, y_2 \,.\, P[y_1/y] \wedge P[y_2/y] \wedge y_1 \neq y_2$$

SAT means the circuit is underconstrained

Picus


Polynomial Field Equations
P
Picus
✓ If it can prove P is constrained
✗ If it can prove P is unconstrained
?? Otherwise

Combine the strengths of Static Analysis and SMT!

Fast but imprecise!

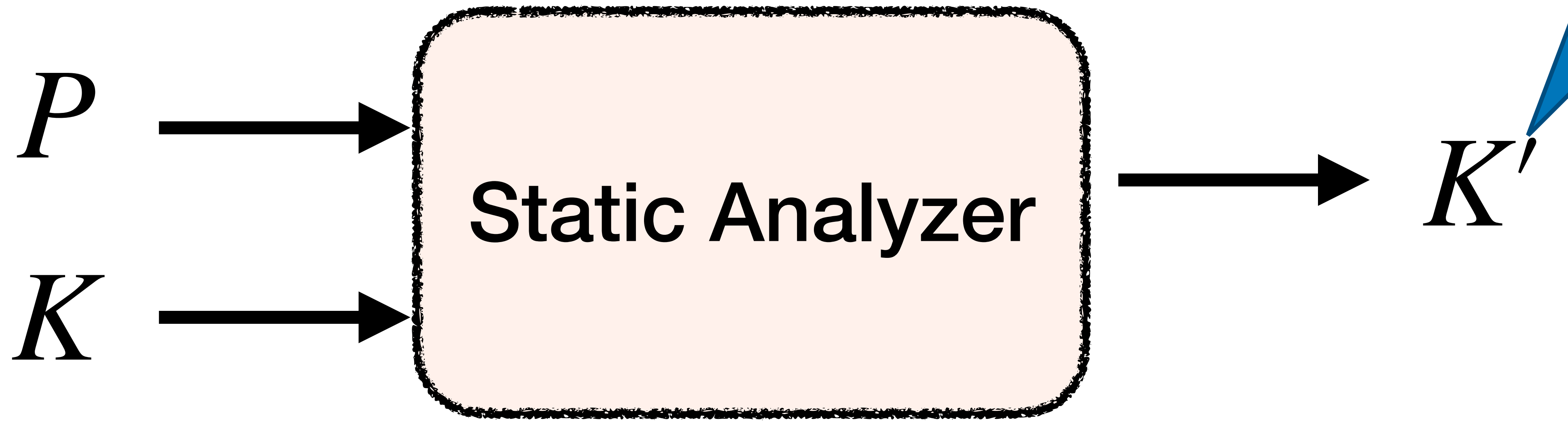
Precise but slow!

Static Analysis and SMT phases interact in a loop

Static Analysis Phase

Takes as input field equations P, and set of signals K proven

At the start of the algorithm $K = \{\}$.

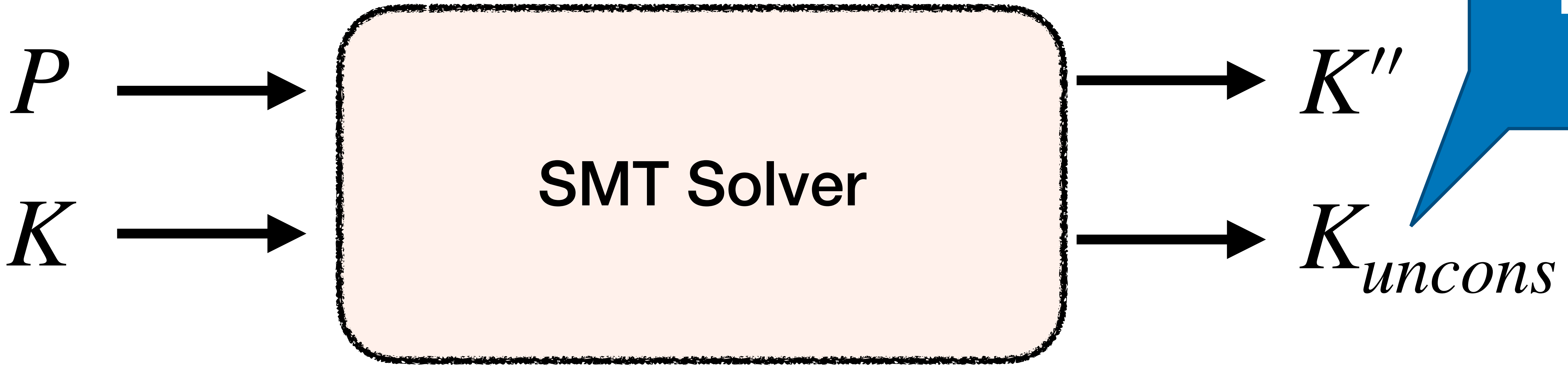


New set $K'$of signals proven unique. $K \subseteq K'$

If OutputSignals $\subseteq K'$ **we return** ✓

Otherwise we send K' as input to SMT Phase

SMT Phase



K'' **computed assuming K.**
Larger K = faster query.
$K \subseteq K''$

If OutputSignals $\subseteq K''$ **we return** ✓

If OutputSignals $\cap K_{uncons} \neq \emptyset$ **we return** ✗

If $K = K''$ **we return** **??**

Otherwise we send K'' **to Static Analysis phase and repeat.**

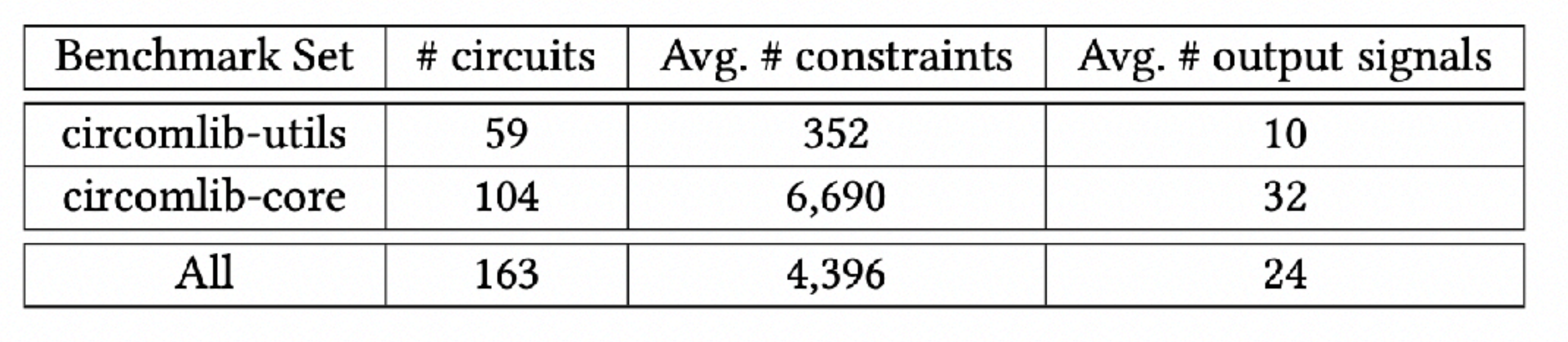
Picus Results

Picus Output

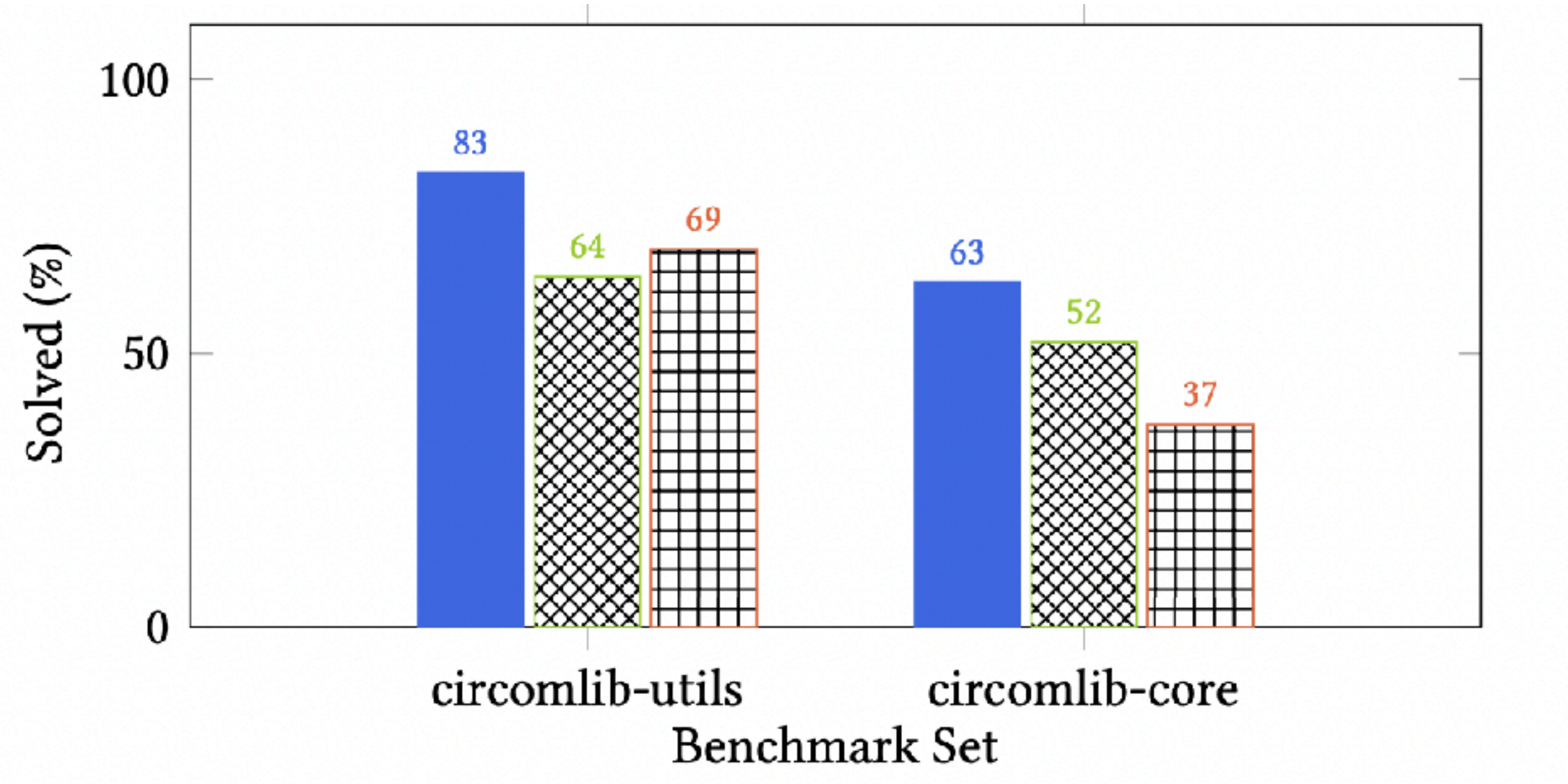
```
$ ./picus-solve.sh ./benchmarks/motivating/
adder.r1cs
# number of constraints: 9
# parsing alternative r1cs...
# configuring precondition...
safe.
```

Guaranteed to have no underconstrained signals!

Evaluation



Benchmark Set	# circuits	Avg. # constraints	Avg. # output signals
circomlib-utils	59	352	10
circomlib-core	104	6,690	32
All	163	4,396	24



Conclusion

- Automated Detection of Underconstrained Circuits for Zero-Knowledge Proofs, PLDI'23
- Practical Security Analysis of Zero-Knowledge Proof Circuits
- Certifying Zero-Knowledge Circuits with Refinement Types

https://github.com/Veridise/Picus

Medium https://veridise.medium.com/

https://veridise.com/ @VeridiseInc