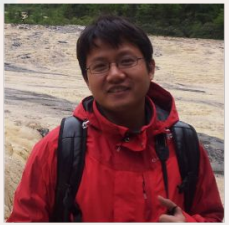
Are your Zero-Knowledge Proofs Correct?

Jon Stephens

CTO, Veridise

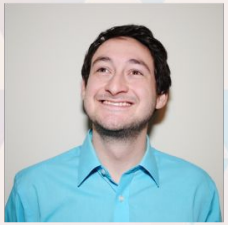
About Us

Isil Dillig, President
Stanford (BS'06, PhD'11)
CS Prof at UT Austin



Yu Feng, CEO
UT Austin (PhD'18)
CS Prof at UCSB

Jon Stephens, CTO
UT Austin,
Current PhD candidate



Ben Mariano, VP of R&D
UT Austin
Current PhD candidate

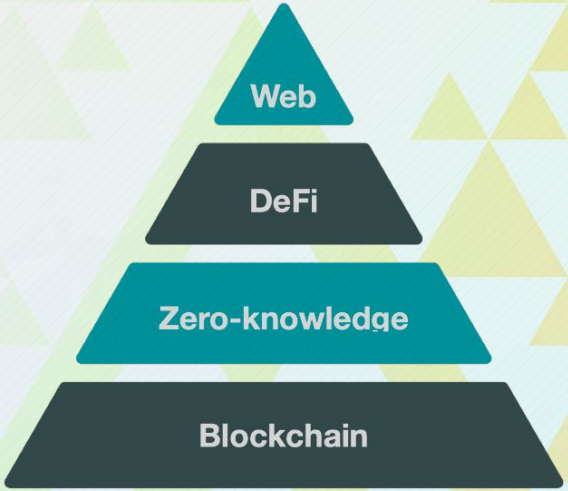
Shankara Pailoor, Sr. Scientist
UT Austin
Current PhD candidate

Jacob Van Geffen, Sr. Scientist
University of Washington
Current PhD candidate

About Us

Veridise provides state-of-the-art formal methods solutions for **all layers** in the blockchain ecosystem

Used to aid our DeFi, ZK Circuit and Blockchain audits. Now available in our SaaS!

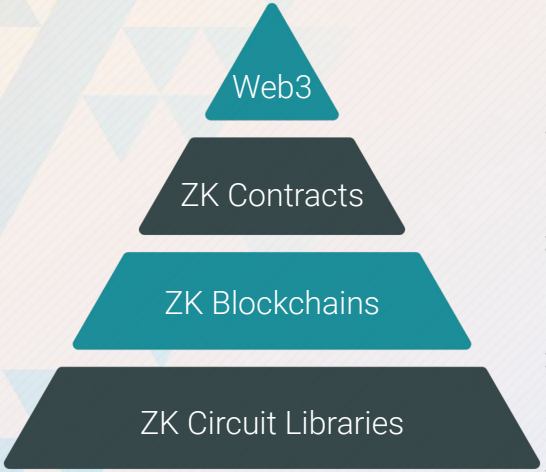


Section 1

Bugs in Zero-Knowledge Circuits

ZK in Web3

Zero-Knowledge Circuits are pervasive in L2



Semaphore, Tornado.cash, ...

ZkEVM, Scroll, Manta Network, Polygon Hermez, ...

CircomLib, ...

These circuits present new challenges!

DeFi Bugs

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

Beanstalk got jacked by a giant f

Crypto Bridge Nomad Drained of Nearly $200M in Exploit

The exploit calls the security of cross-chain toke

Harmony's $100M Hack Was Due to a Compromised Multi-Sig Scheme, Says Analyst

---

Tornado Cash

Oct 12, 2019 · 3 min read · Listen

Tornado.cash got hacked. By us.

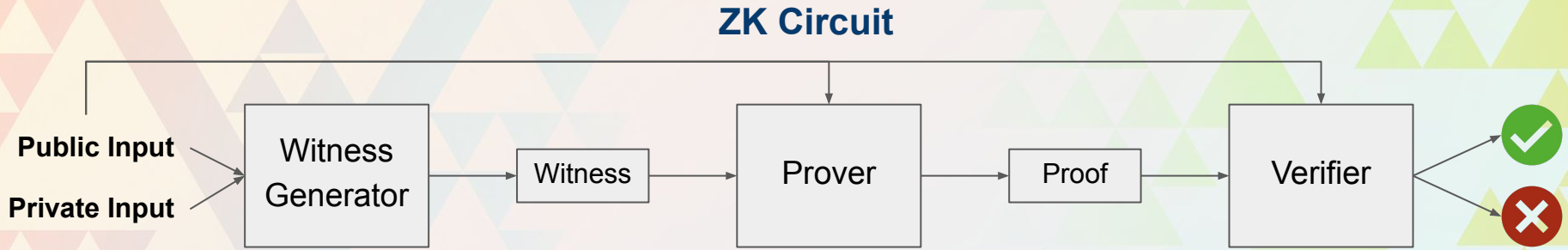
Polygon Dodges $850M Hack, Pays Record $2M Bounty

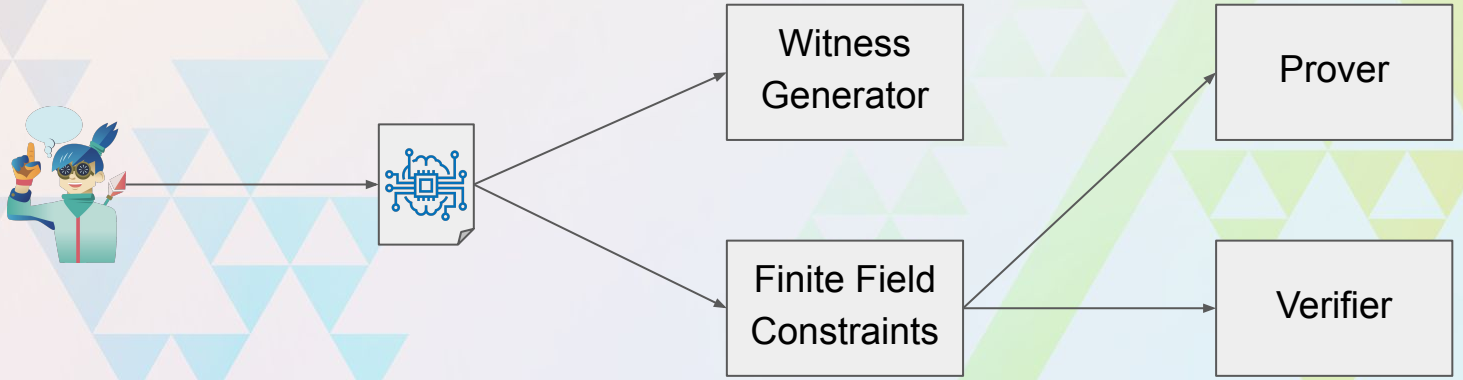
The team behind the Polygon protocol has paid ou
white hat hacker who discovered a critical vulnerab

Solana hit with another network incident causing degraded

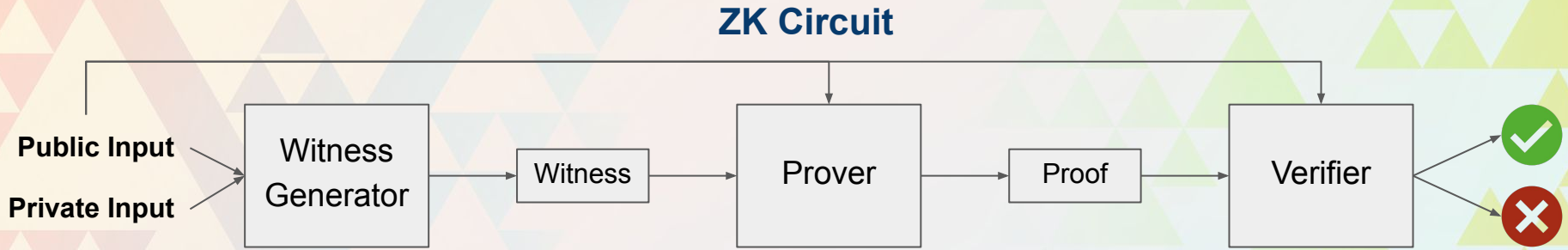
Solana's network goes down for the third time in less than six months

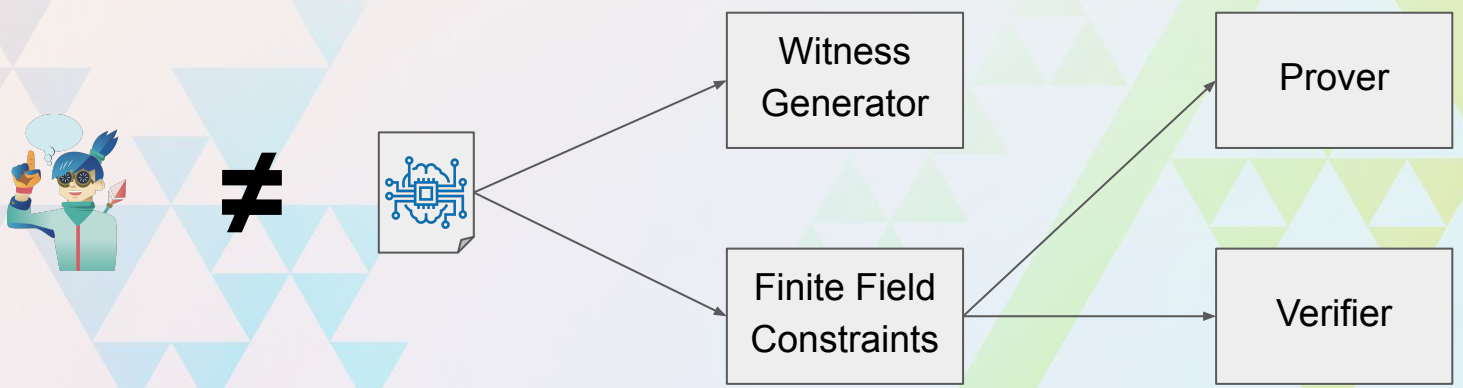
What is a Zero-Knowledge Circuit?



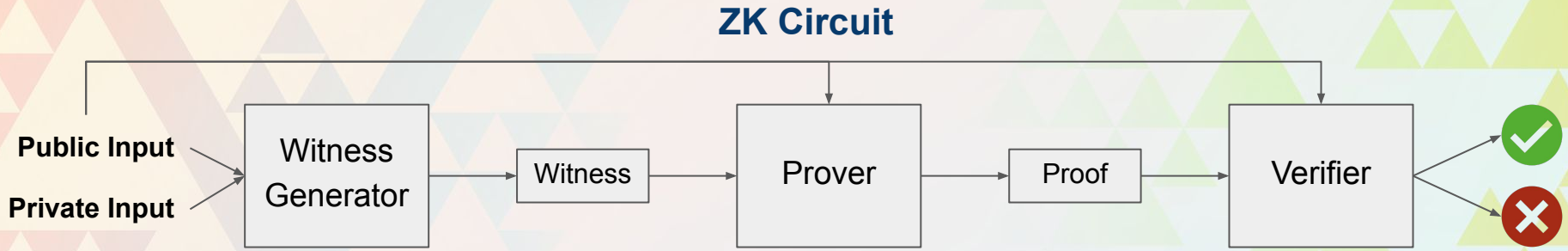


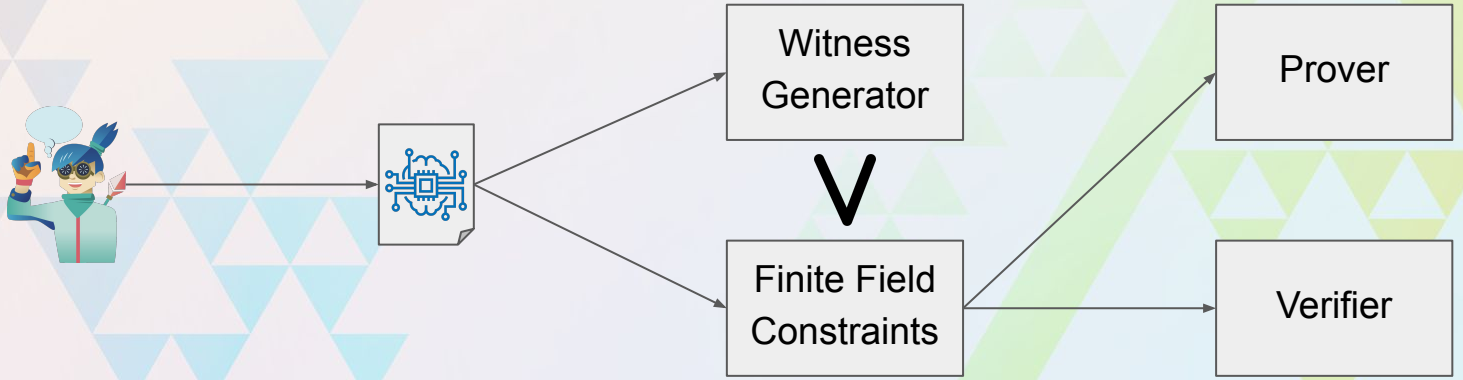
What is a Zero-Knowledge Circuit?



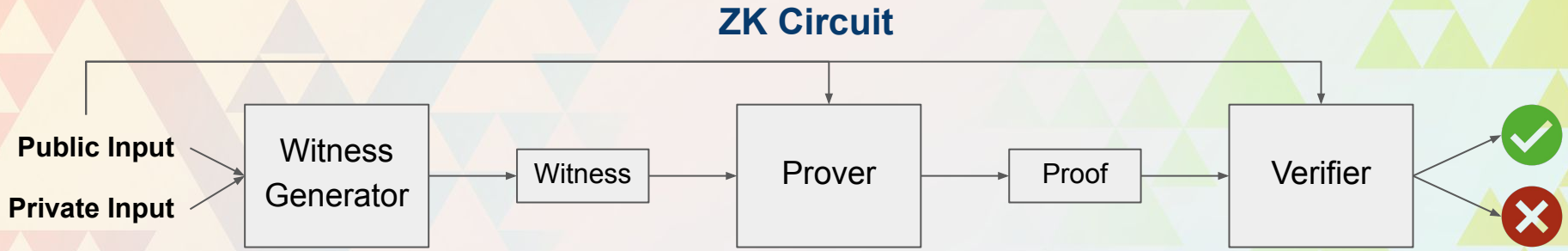


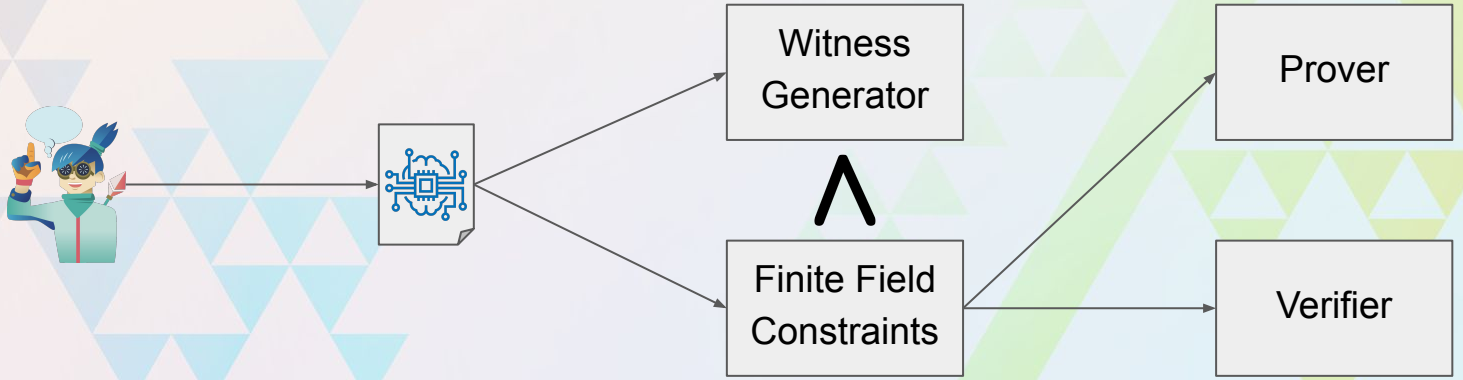
What is a Zero-Knowledge Circuit?





What is a Zero-Knowledge Circuit?





Formal Methods

Set of techniques for finding bugs and constructing proofs about software

Automated testing

- ✓ Method: Run program on constructed inputs
- ✓ Purpose: Bug finding

Static analysis

- ✓ Method: Analyze source code for specific classes of bugs
- ✓ Purpose: Bug finding + proof

Formal verification

- ✓ Method: Analyze source code wrt formal specification
- ✓ Purpose: Bug finding + proof

Stronger guarantees

More human effort

Section 2

Common Vulnerability Detection

Uniqueness Bugs

Constraints allow a single input to map to multiple outputs!

```
template Decoder(w) {
    signal input inp;
    signal output out[w];
    signal output success;
    var lc=0;

    for (var i=0; i<w; i++) {
        out[i] <-- (inp == i) ? 1 : 0;
        out[i] * (inp-i) === 0;
        lc = lc + out[i];
    }

    lc ==> success;
    success * (success -1) === 0;
}
```

Entries are zero except out[in] if in < w

$$out[i] * (inp - i) = 0$$

$$(inp - i) = 0 \qquad out[i] = 0$$

When $inp = i$, $out[i]$ can be any value and satisfy the above constraint.

Detecting Uniqueness Bugs

Static Analysis of Constraints (ECNE)

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

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

SAT means the circuit is underconstrained

	Pros	**Cons**
Static Analysis	Scalable	Many False Positives
SMT Solver	Precise	Doesn't Scale

Picus

Polynomial Field Equations

$\mathscr{P}$

Picus

If it can prove $\mathscr{P}$ is constrained

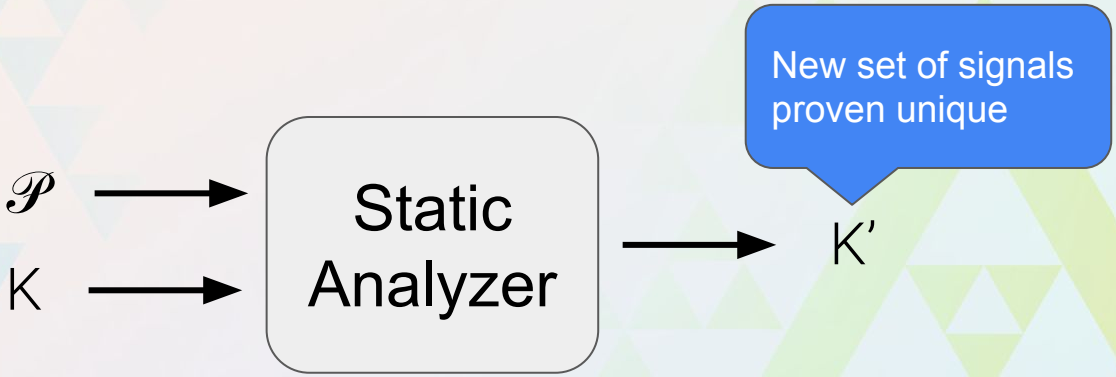
If it can prove $\mathscr{P}$ is unconstrained

?? Otherwise

Combine the strengths of Static Analysis and SMT!

Static Analysis Phase

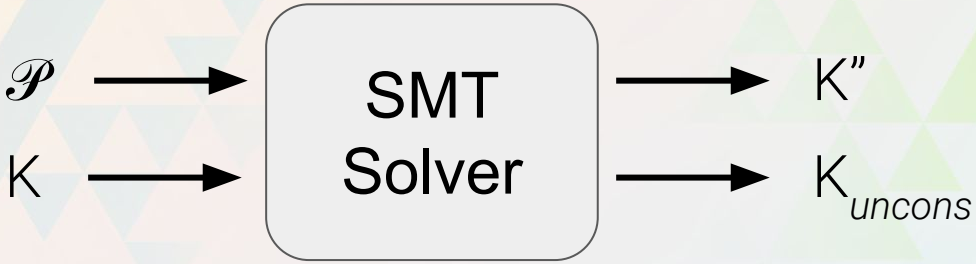
Takes Polynomial Constraints ($\mathscr{P}$) and set of signals proven unique (K) as input



If Output Signals ⊆K', return ✓

Otherwise send K' to SMT Phase

SMT Phase



If Output Signals ⊆K', return ✓

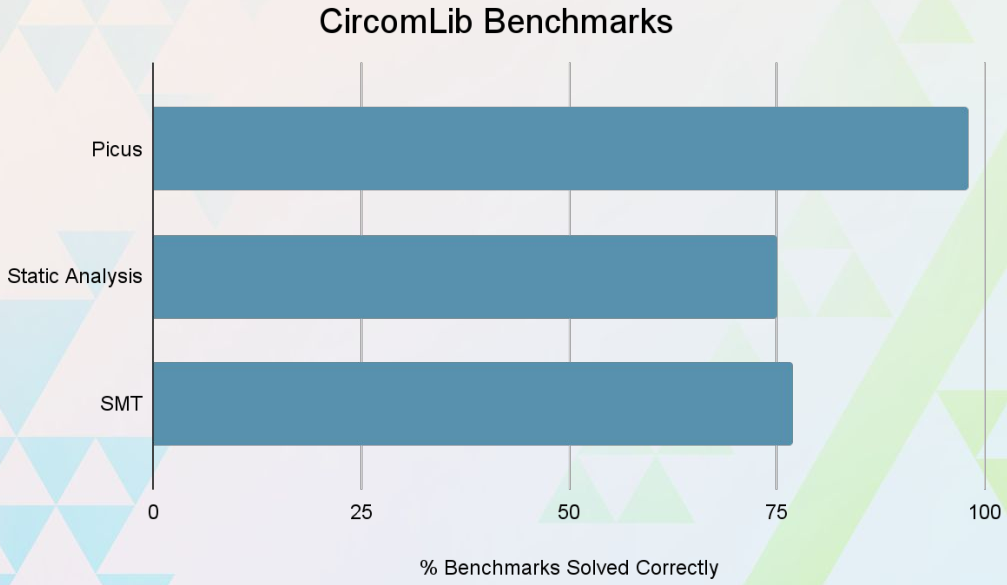
If Output Signals ∩ K_{uncons} ≠∅, return ✗

If K = K", return ??

Otherwise invoke Static Analysis phase with K"

Results

Picus outperforms Static Analysis and SMT



Section 3

Functional Correctness Checking

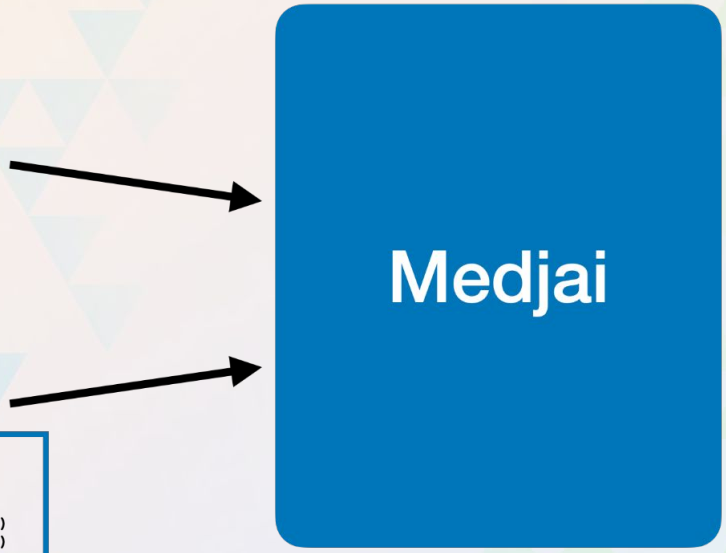
Medjai

Source Code

```
func move(src : felt, dst : felt, rad : Uint256):
    check_validity(rad)

    # Sub from src
    let (dai_src) = _dai.read(src)
    let (dai_src) = sub(dai_src, rad)
    _dai.write(src, dai_src)

    # Add to dst
    let (dai_dst) = _dai.read(dst)
    let (dai_dst) = add(dai_dst, rad)
    _dai.write(dst, dai_dst)

    return ()
end
```

Specification

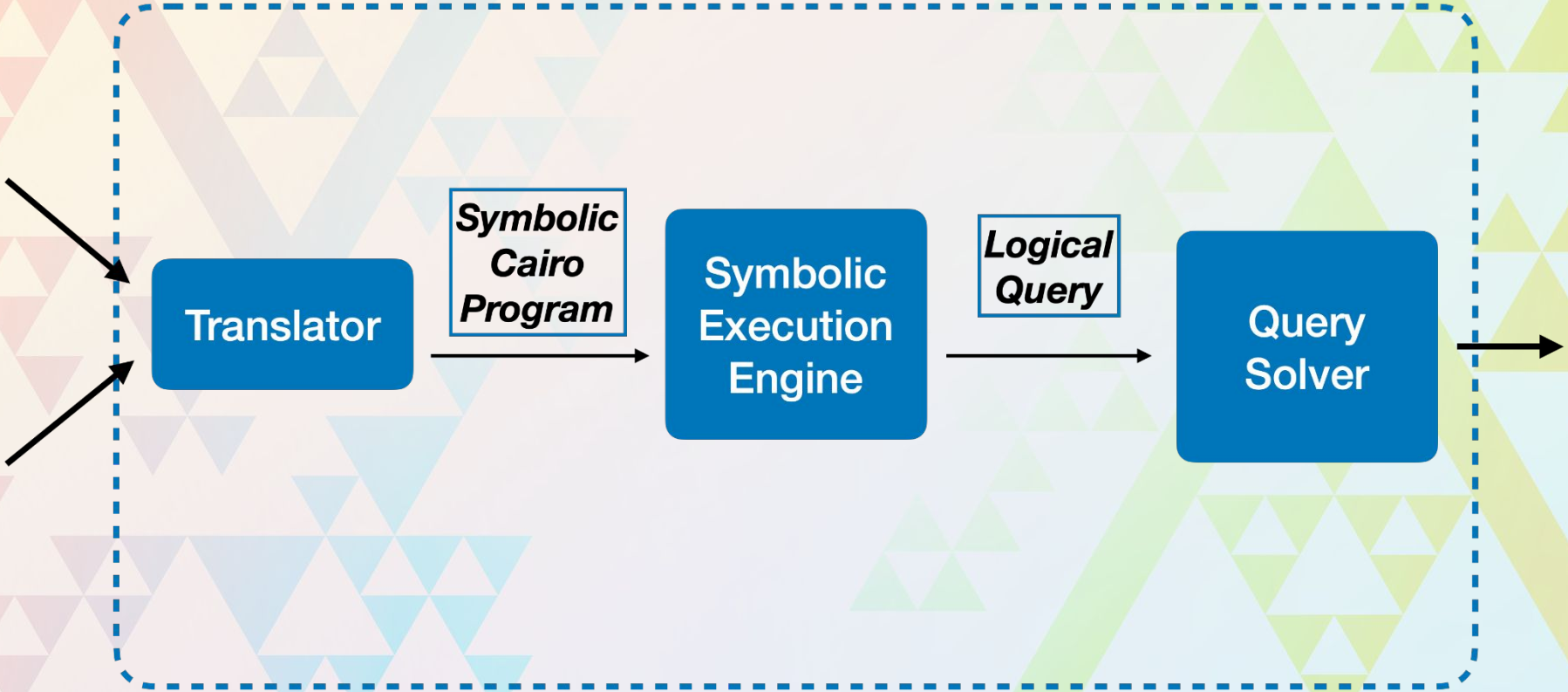
```
vars: felt other
spec: finished(contract.move(src, dst, rad),
               other != src && other != dst
               |=>
                   _dai[dst] >= old(_dai[dst])
               && _dai[src] <= old(_dai[src])
               && _dai[other] = old(_dai[other])
```



OR

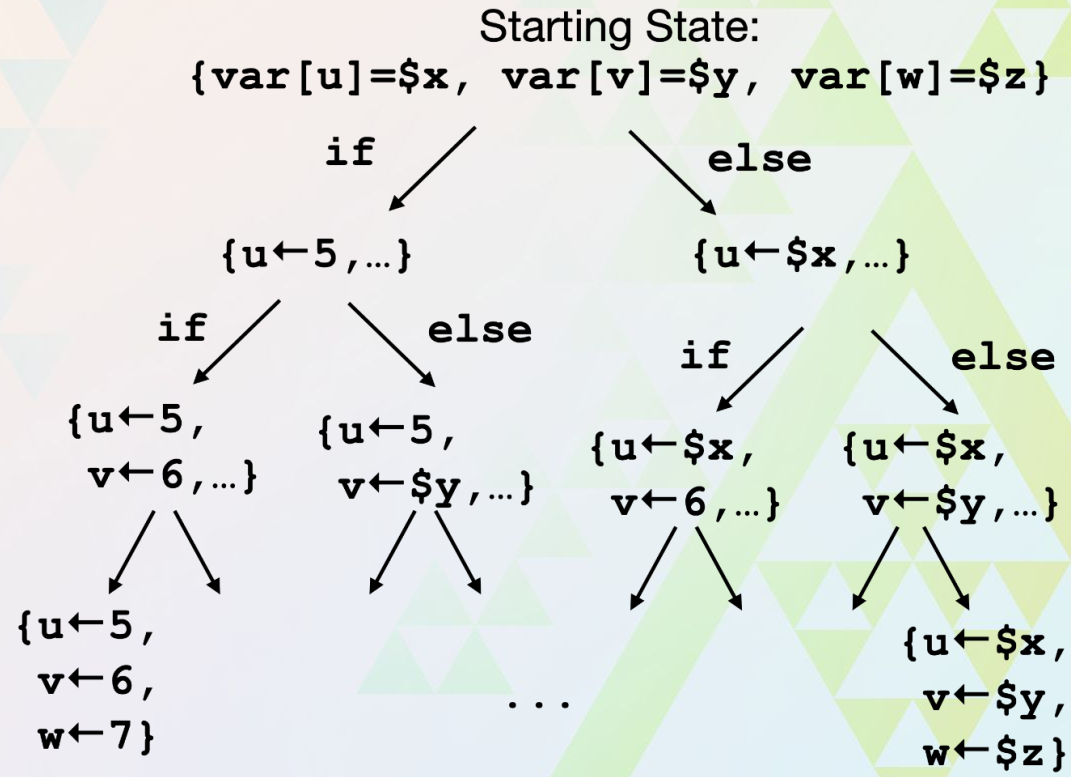
when
src=...
dst=...
rad=...

Medjai


Translator
Symbolic Cairo Program
Symbolic Execution Engine
Logical Query
Query Solver

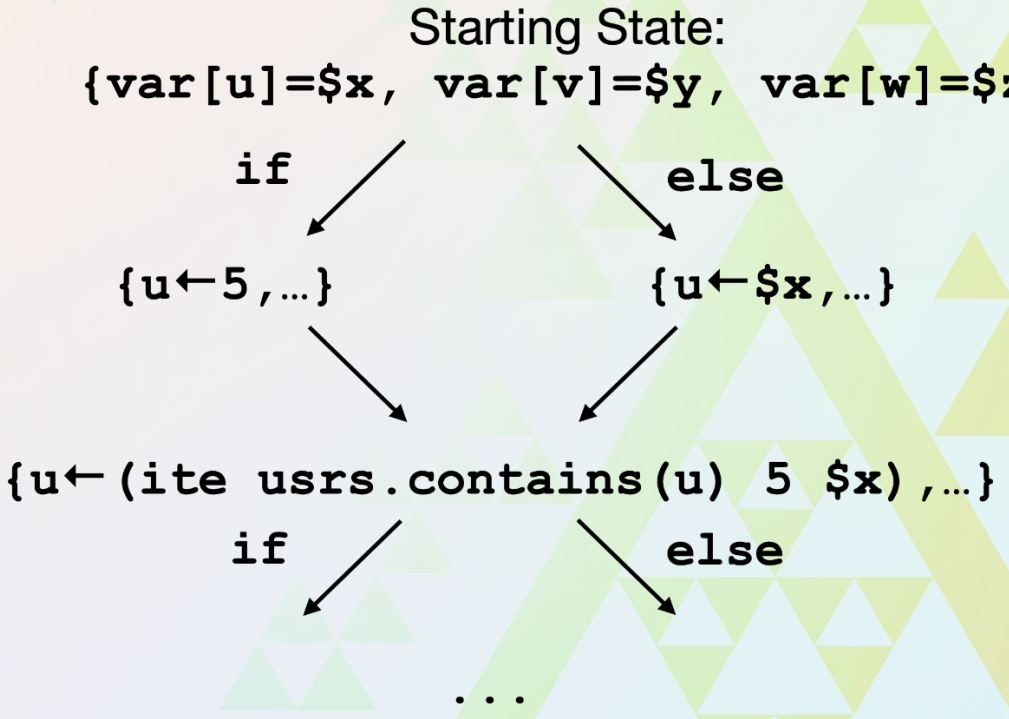
Symbolic Execution

```
# u, v, w : symbolic felts
func my_func(u, v, w, ...):
    if usrs.contains(u):
        var[u] = 5
    end

    if usrs.contains(v):
        var[v] = 6
    end

    if usrs.contains(w):
        var[w] = 7
    end

    ...
```



Optimization

```
# u, v, w : symbolic felts
func my_func(u, v, w, ...):
    if usrs.contains(u):
        var[u] = 5
    end

    if usrs.contains(v):
        var[v] = 6
    end

    if usrs.contains(w):
        var[w] = 7
    end

    ...
```



Example

Goal: Verify and find bugs in ZK Smart Contracts

```
func move(src : felt, dst : felt, rad : Uint256):


    # Sub from src
    let (dai_src) = _dai.read(src)
    let (dai_src) = sub(dai_src, rad)
    _dai.write(src, dai_src)

    # Add to dst
    let (dai_dst) = _dai.read(dst)
    let (dai_dst) = add(dai_dst, rad)
    _dai.write(dst, dai_dst)

    return ()
end
```

Specification:

_dai[dst] >= old(_dai[dst])
_dai[src] <= old(_dai[src])

Medjai Output:

when

`rad=Uint256(2**129, 2**129)`

Example

Goal: Verify and find bugs in ZK Smart Contracts

```
func move(src : felt, dst : felt, rad : Uint256):
    uint256_check(rad)

    # Sub from src
    let (dai_src) = _dai.read(src)
    let (dai_src) = sub(dai_src, rad)
    _dai.write(src, dai_src)

    # Add to dst
    let (dai_dst) = _dai.read(dst)
    let (dai_dst) = add(dai_dst, rad)
    _dai.write(dst, dai_dst)

    return ()
end
```

Specification:

```
_dai[dst] >= old(_dai[dst])
_dai[src] <= old(_dai[src])
```

Medjai Output:

Example

Goal: Verify and find bugs in ZK Smart Contracts

maciejka fix fold, suggested by Veridise

```
func mov                                    rad : Uint256):
    ui
    # Sub
    let (dai_src)
    let (dai_src) = sub(
    _dai.write(src, dai_src)

    # Add to dst
    let (dai_dst) = _dai.read(dst)
    let (dai_dst) = add(dai_dst, rad)
    _dai.write(dst, dai_dst)

    return ()
end
```

Specification:

```
dai[dst] >= old(_dai[dst])
src] <= old(_dai[src])
```

Thank you!

Jon Stephens

CTO, Veridise

jon@veridise.com

@VeridiseInc

Picus Repository

Medjai Repository