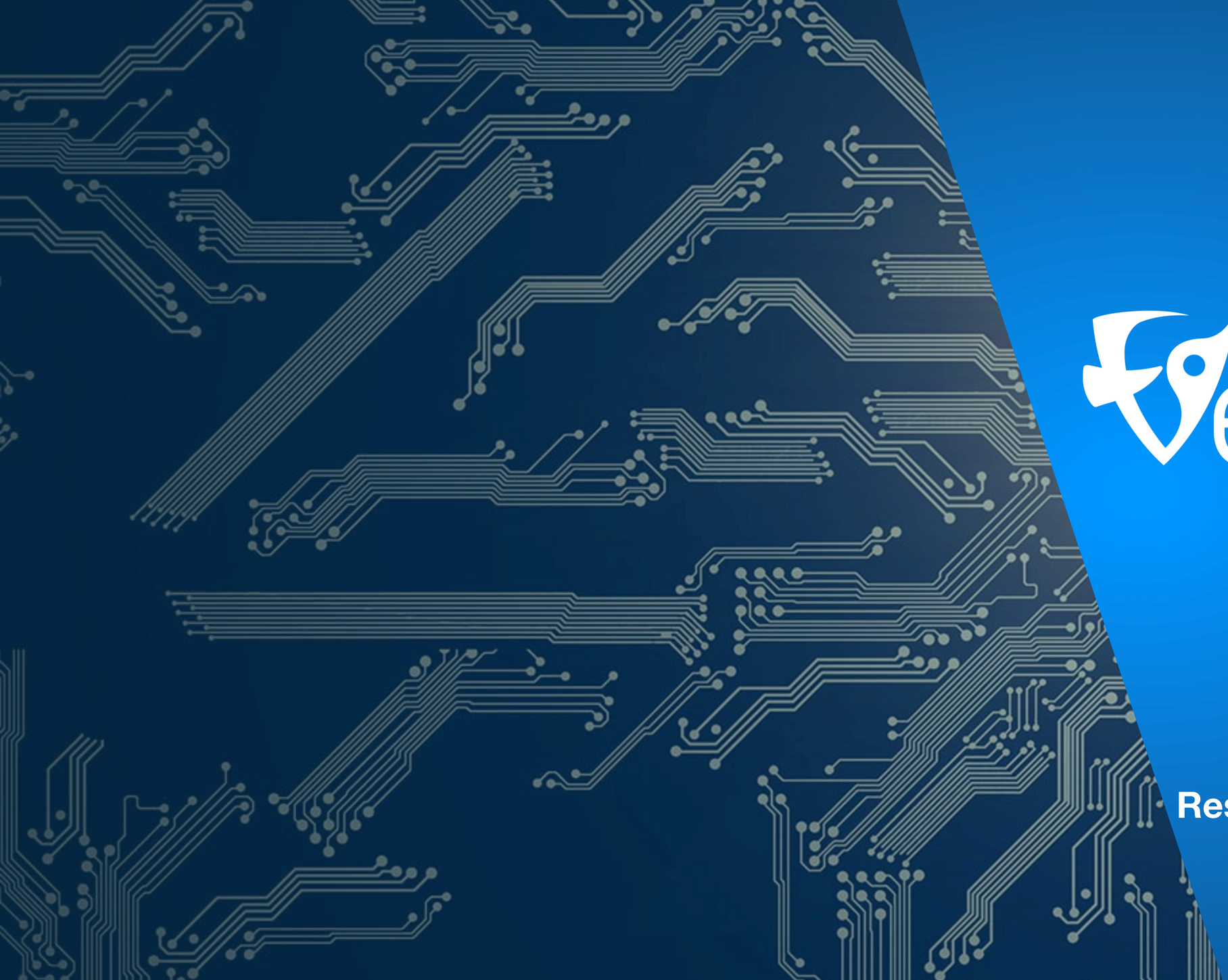


Veridise.

ZK Workshop
—Day 4—

Shankara Pailoor
Research Scientist, Veridise

Veridise. | Recap

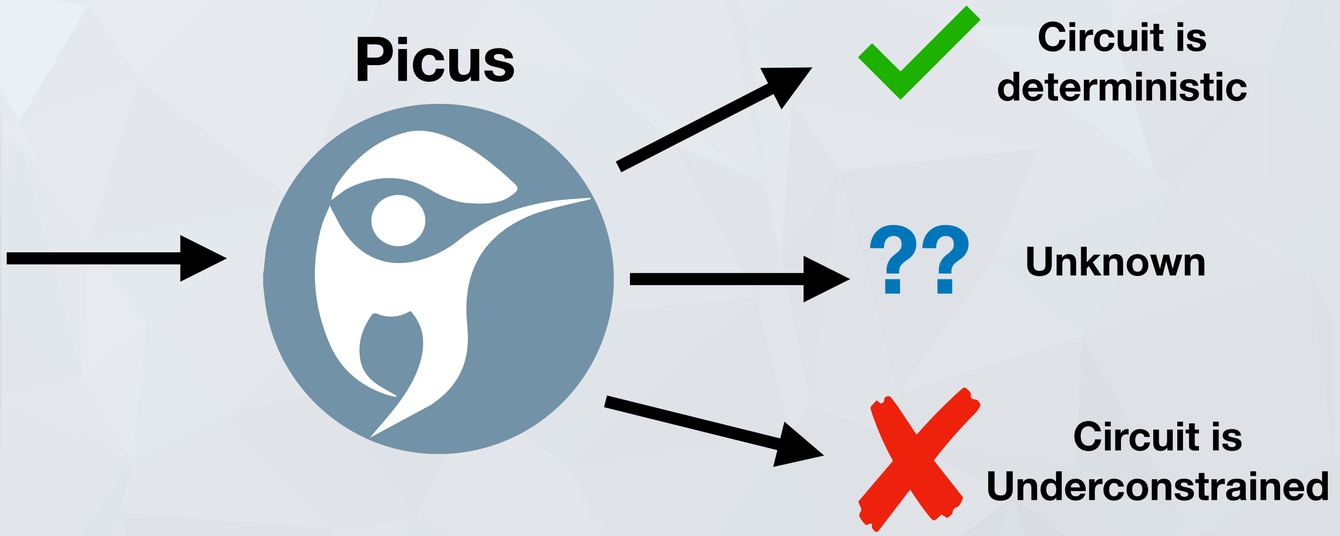
- Introduction to Circom
- Witness Generation vs. Constraints
- Constraint Dependence Graphs
- Use ZK Vanguard to find bugs

Veridise. | Picus - 30k feet view

Picus is a *verifier* that *automatically* checks whether a ZK circuit is *underconstrained*

Circuit

```
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;
}
```



Veridise. | What to expect!

Theoretical Part

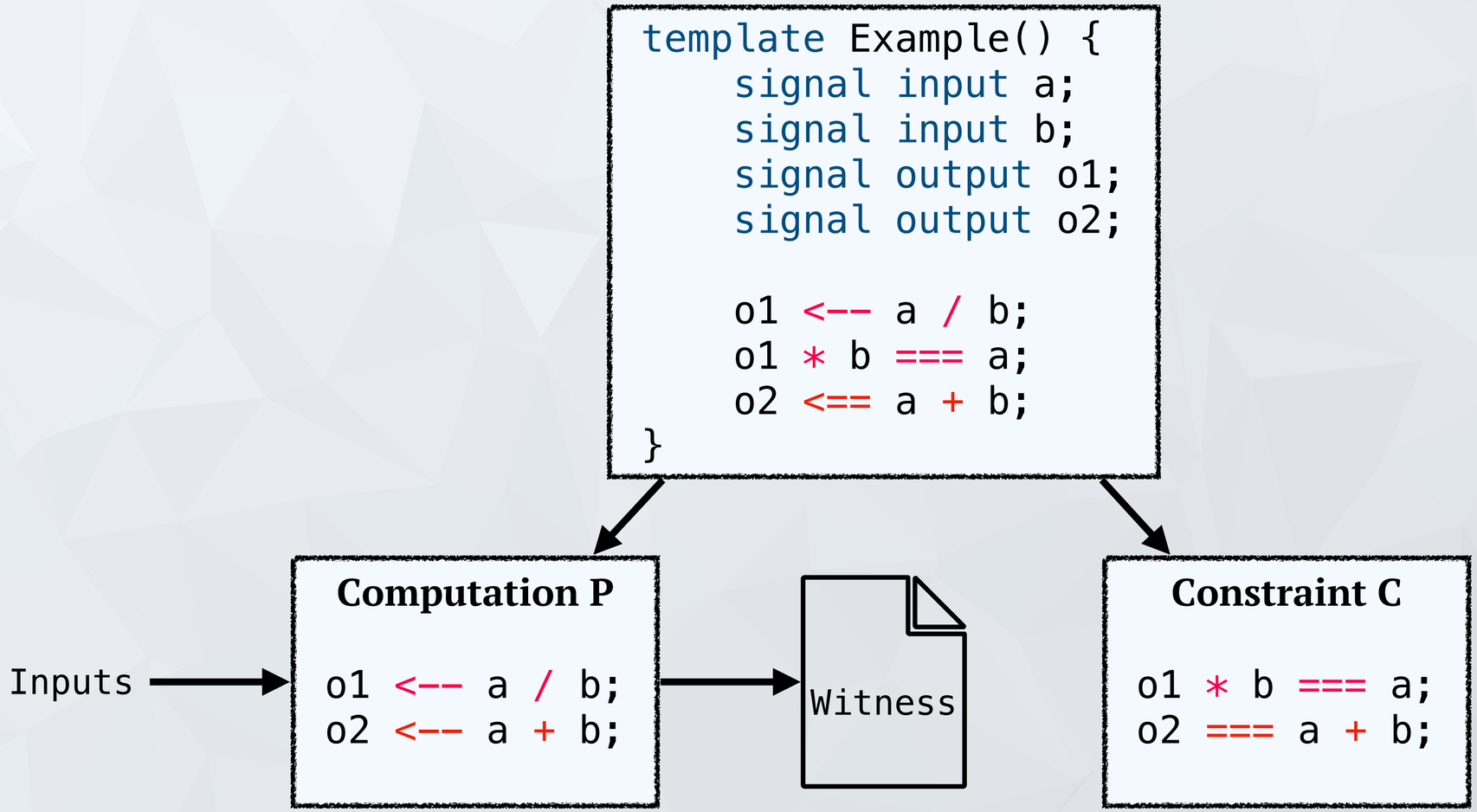
- **Learn what underconstrained circuits are.**
  - Why they are an important class of bugs.
- **Overview of Picus.**
  - How it works.
  - How to use it.

Practical Part

- **Quiz**

Veridise. | Circuit Correctness

- Recall ZK Programs consist of two parts: **Computation** and **Constraints**



Veridise. | Circuit Correctness

- Recall ZK Programs consist of two parts: **Computation** and **Constraints**
- **Computation** is a normal computer program $P(x, w)$ where (w possibly secret) that returns some value y
- **Constraints** are polynomial equations $C(x, w, y)$
- **Ideally:** Constraints and Computation are *equivalent*!

Veridise. | Equivalence

- We say a witness generation program $P(x, w) = y$ is equivalent to constraints $C(x, w, y)$ if and only if every execution trace of P is a satisfying assignment to C and vice-versa

- An execution trace for P is a mapping from signals in P to values obtained when executing the program on some input

- A satisfying assignment for constraints C is a mapping from variables in C to values that make C evaluate to true.

Veridise. | Example

Computation P

```
o1 <-- a / b;
o2 <-- a + b;
```

Constraint C

```
o1 * b === a;
o2 === a + b;
```

Execution Trace

$$\{a \to 4,\ b \to 2,\ o1 \to 2,\ o2 \to 6\}$$

Satisfying Assignment

$$\{a \to 2,\ b \to 1,\ o1 \to 1,\ o2 \to 3\}$$

Question: Are these equivalent?

Veridise. | Underconstrained Bugs

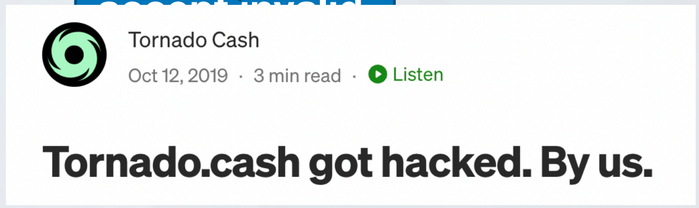
- If computation and constraints are equivalent, then if P is deterministic, then C should be as well.

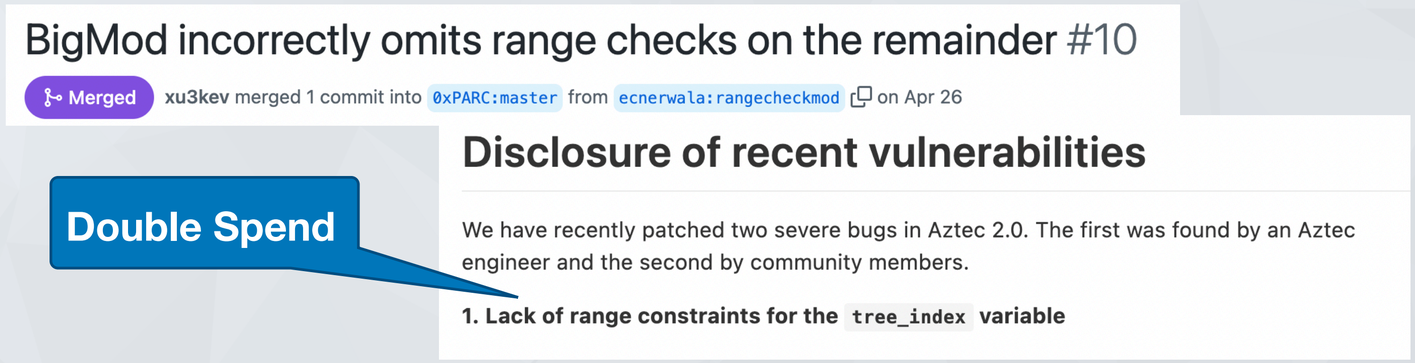
- A circuit C is **deterministic** if for any input x, w and any pair of outputs y, y', if $C(x, w, y)$ and $C(x, w, y')$ then $y = y'$

Could be used to drain all tokens

cuit is **underconstrained** (nondeterministic) if it is *not deterministic*.

r could





Double Spend

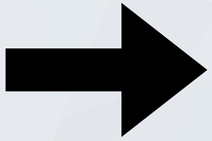
Veridise. | Example

BuggyExample.circom

```
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;
}
```



Constraints for $n = 3$

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

out_2 **is underconstrained**

Attacker can pass in any value for out_2

Veridise. | Picus Demo

Demo Through Saas!

Veridise. | Existing Strategies

Static Analysis of Constraints

Apply predefined rules to quickly detect if circuit is properly constrained

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

Since z is function of x and y is a function of both x and z we infer y is uniquely determined by input x

SMT

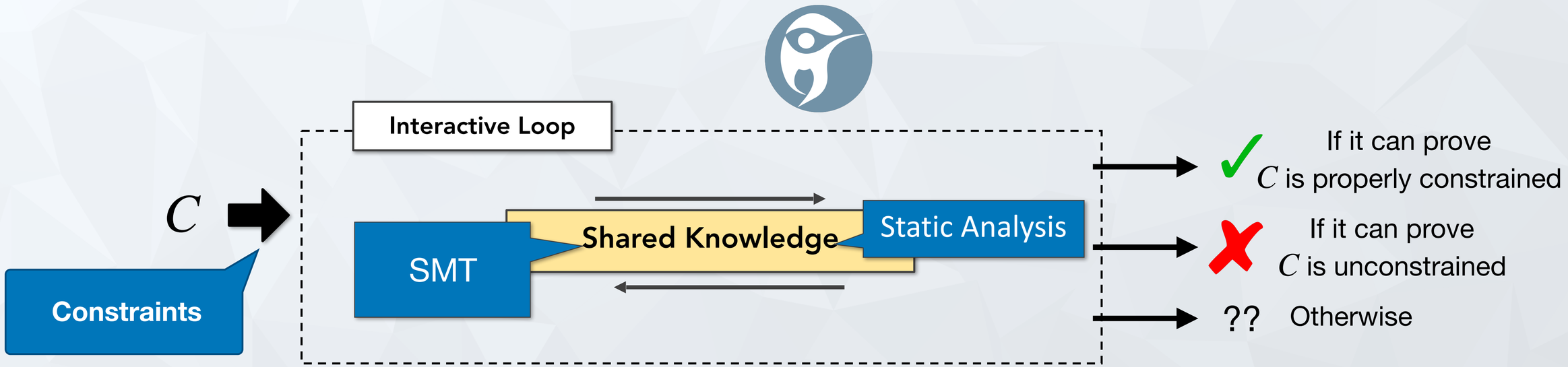
Underconstrained can be expressed as SMT query

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

SAT means the circuit is underconstrained

Strategy	Pros	Cons
Analysis	**Scalable**	Many False Positives
SMT	**Precise**	Can’t Scale

Veridise. | Picus



Combine the strengths of Static Analysis and SMT-based reasoning!

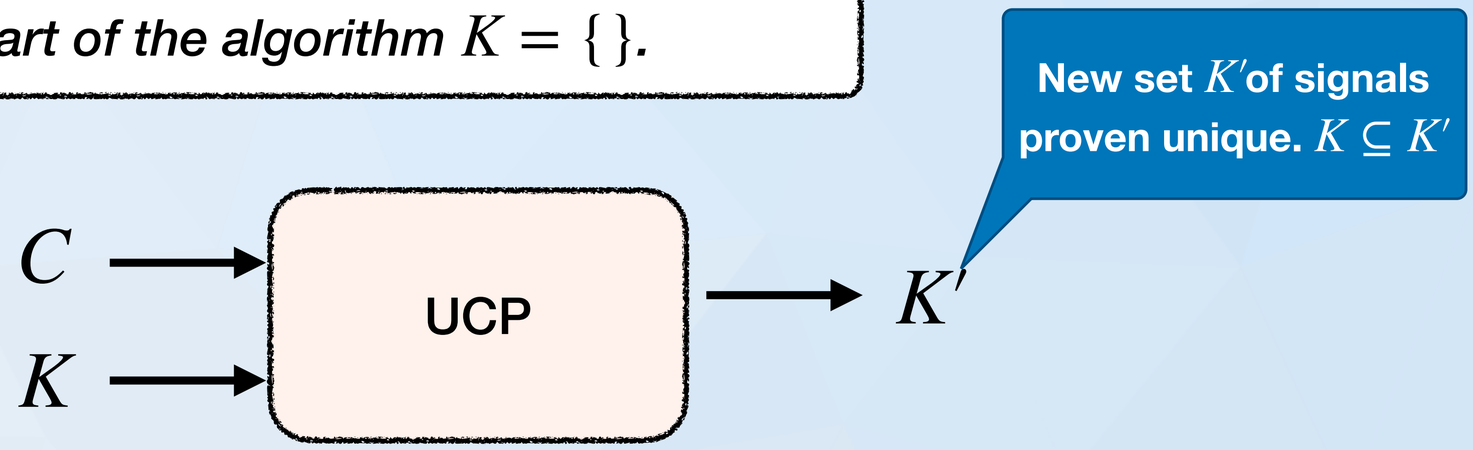
Static Analysis and SMT phases interact in a loop

Partial Results from one phase can be useful for another

Uniqueness Constraint Propagator (UCP) (Static Analysis)

Takes as input field equations C, and set of signals K proven unique.

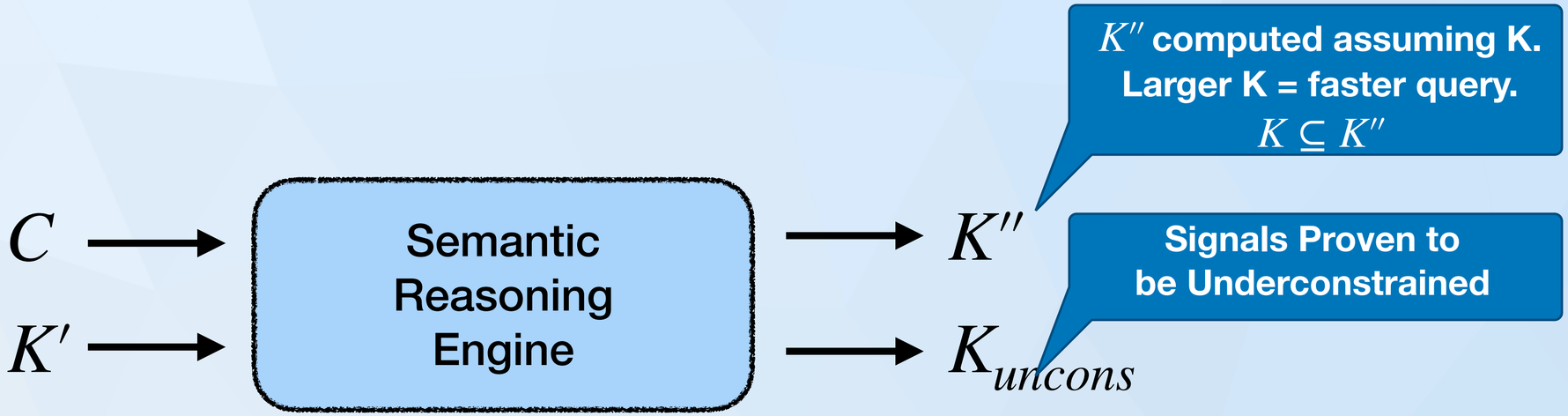
At the start of the algorithm $K = \{\}$.



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

Otherwise we send K' as input to SMT Phase

Semantic Reasoning Engine (SMT)



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.**

Veridise. | Tips for using Picus when auditing

- For larger Circuits, try using Picus on individual components first to make sure they are not underconstrained.

- If the Circuit is instantiated with a large parameter, first try and see if it is underconstrained when using a smaller parameter.

- If you think there is an underconstrained bug in part of a circuit, try and extract it from the rest and run Picus on the extracted portion.

Conclusion

Questions?