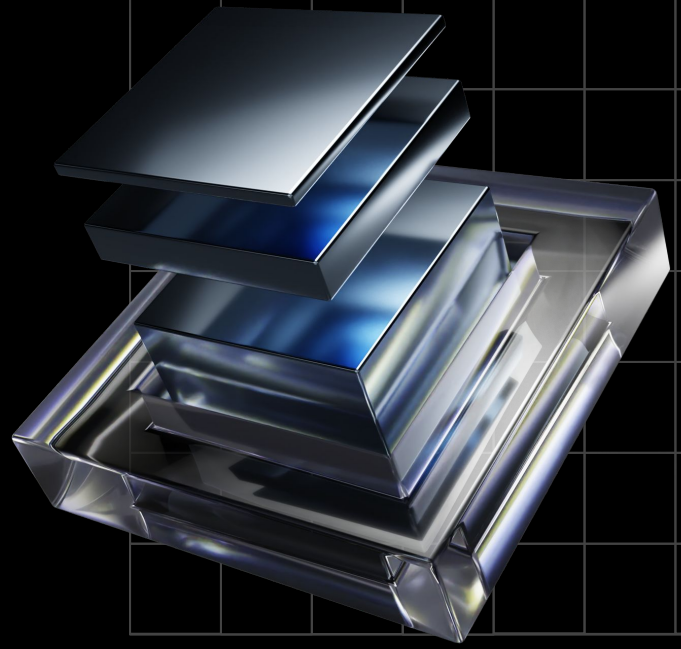
Essential

An Introduction to Declarative Programming for Blockchain Applications



Declarative Programming

Essential

V.S. Imperative programming

- *Example:*
  - *Declarative: SQL, Prolog, Haskel, ...*
  - *Imperative Rust, C++, ...*

Logic without control flow

No explicit algorithms

Minimize side effects

What v.s. How

Constraint Programming

Essential

Constraints on “Decision Variables”

Describe *properties* of a “solution” to be found

Example:

find x and y such that

- x * y = 100
- x + y = 25

One possible solution: x = 20 and y = 5

What does that have to do with Blockchain Apps?

Essential

Smart contracts describe state updates

Imperative contracts: state updates are a *side effect*

Constraint-based contracts are a better fit!

- Describe **allowed** state changes
- “Solution” = proposed state changes **computed off-chain**
- Validators verify constraints given a solution

Pint to the Rescue

Essential

Constraint-based DSL for declarative chains

Contract:
- Storage
- Set of predicates

Predicate:
- Variables
- Set of constraints

```
storage {
    counter: int,
}

predicate Increment {
    // Read the current value of the counter
    state counter: int = storage::counter;

    if counter == nil {
        // If the counter hasn't been set before, set it to 1
        constraint counter' == 1;
    } else {
        // If the counter has been set before, then increment it by 1
        constraint counter' == counter + 1;
    }
}
```

Demo