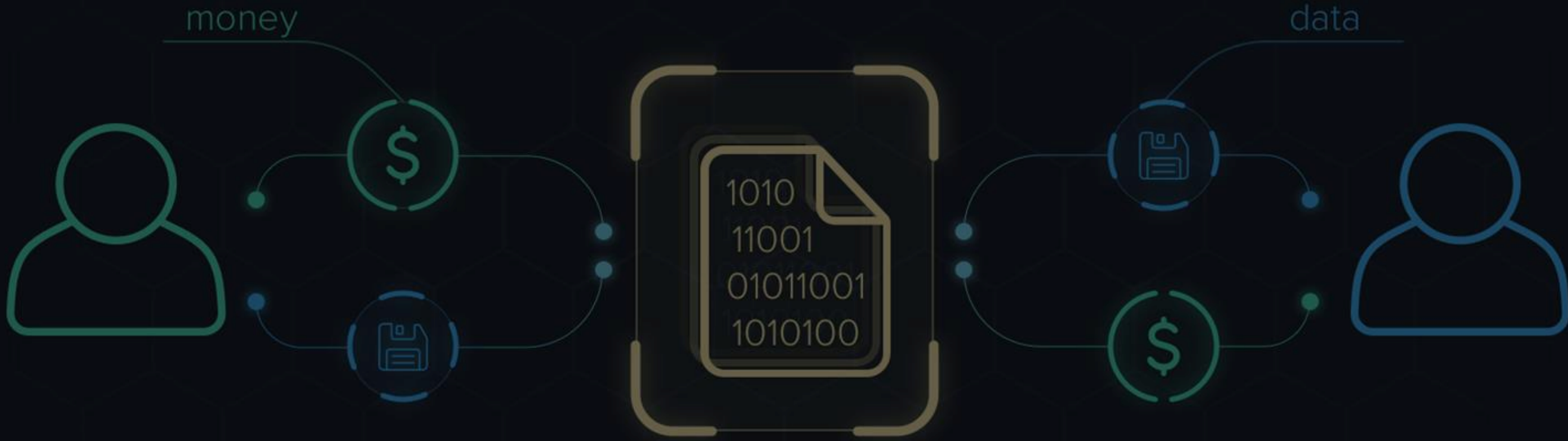
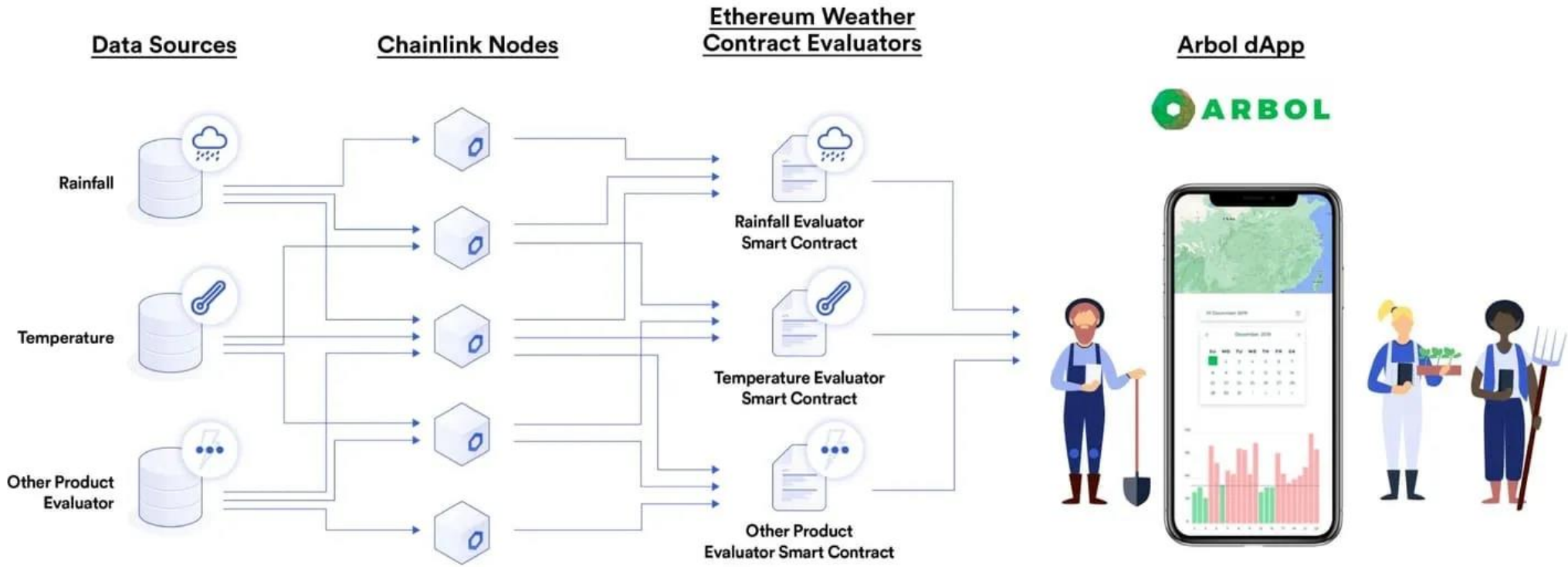
SMART CONTRACTS AND FORMAL VERIFICATION

PowerPoint Presentation in
Blockchain Technology
Fourth Year B. Tech
Sem 8 – A1. PA15. Krishnaraj Thadesar



What are Smart Contracts?

- Self-executing programs deployed on blockchain platforms.
- Code and state are immutable once deployed.
- Run deterministically, triggered by transactions.
- Popular platforms: Ethereum (Solidity), Cardano (Plutus), Tezos (Michelson).


Data Sources
Chainlink Nodes
Ethereum Weather Contract Evaluators
Arbol dApp
ARBOL
Rainfall
Temperature
Other Product Evaluator
Rainfall Evaluator Smart Contract
Temperature Evaluator Smart Contract
Other Product Evaluator Smart Contract

Common Smart Contract Vulnerabilities

Reentrancy attacks (e.g., The DAO hack, 2016)

Integer overflows/underflows (pre-Solidity 0.8.x)

Unchecked external calls and denial-of-service

Front-running and timestamp dependence

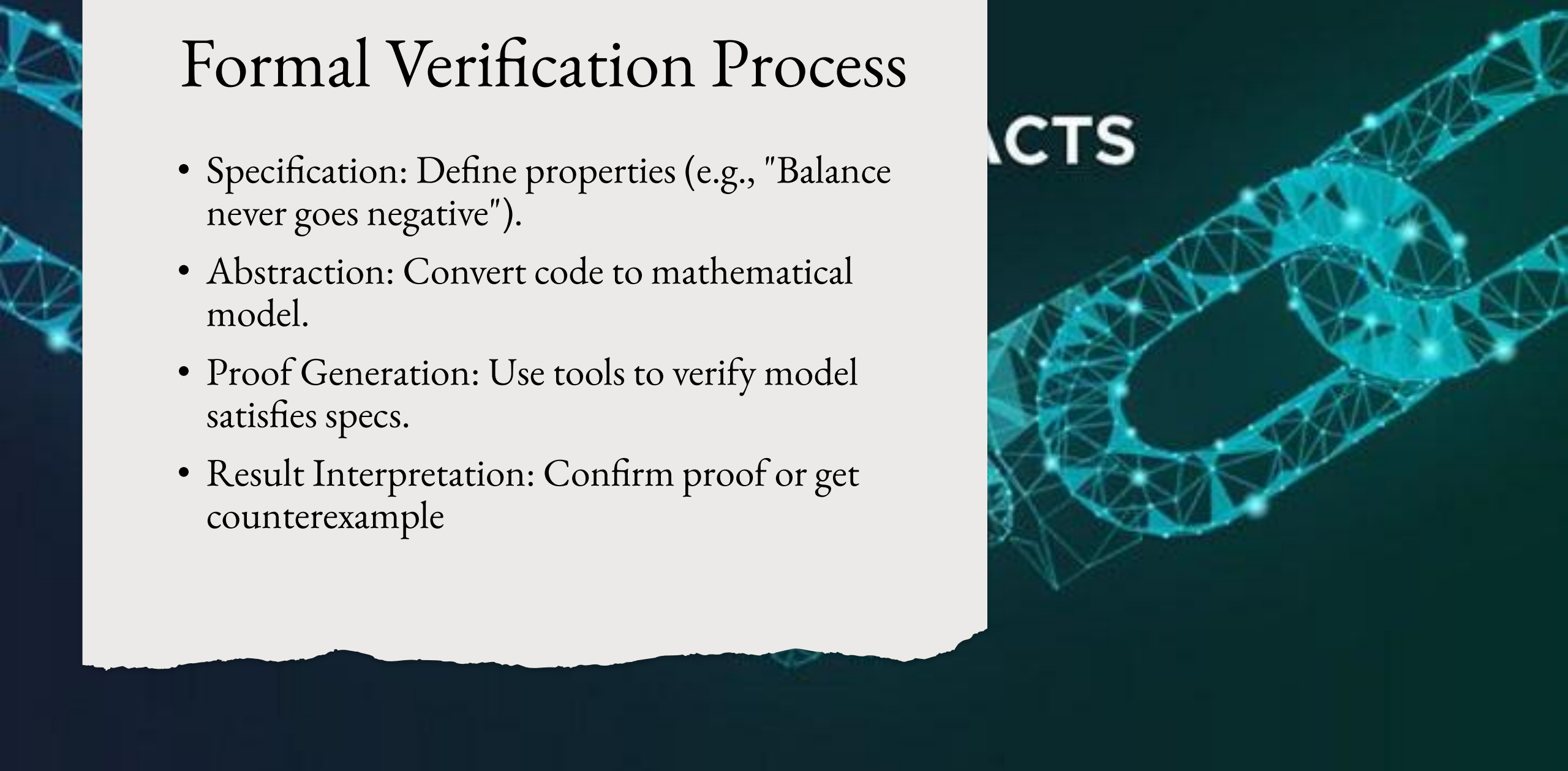
Smart contracts are non-upgradable without proxies — bugs are expensive.

What is Formal Verification?

- Mathematically proving a system satisfies a given specification.
- Converts smart contract logic to logical formulas (e.g., SMT, Hoare Logic).
- Ensures correctness, security, and completeness.
- Avoids reliance on testing alone (which is incomplete).

Formal Verification Process

- Specification: Define properties (e.g., "Balance never goes negative").
- Abstraction: Convert code to mathematical model.
- Proof Generation: Use tools to verify model satisfies specs.
- Result Interpretation: Confirm proof or get counterexample



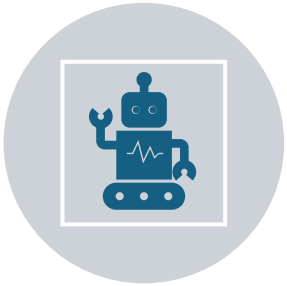
Tools and Frameworks

Solidity + SMTChecker (built-in with Solidity ≥ 0.5.10)
Certora Prover – Symbolic execution and formal specs (used by Aave, Compound)
KEVM – Formal semantics of EVM using K Framework
Coq / F* / Why3 – For writing and verifying critical components
VeriSmart, MythX, Manticore – for hybrid formal+symbolic analysis

Real-World Usage

MakerDAO: Verified modules for stablecoin logic.

Tezos: Supports formal verification at protocol level.



Algorand: Smart contracts in PyTeal can be verified using Coq-based tools.

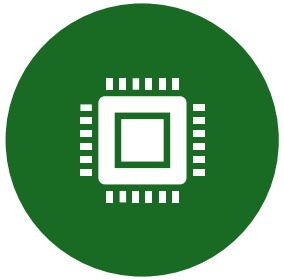
Audits of large DeFi protocols increasingly include formal proofs.

Conclusion

Smart contracts handle **billions in value** — correctness is critical.

Formal verification provides **stronger guarantees** than testing or audits.

Tooling is improving but still requires **mathematical expertise**.



Future: More **user-friendly formal methods** integrated in dev pipelines.