Smart Contracts and Formal Reasoning: “Should we trust in code after all?”

Rules: Logic and Applications,
National Technical University of Athens
19/12/2018

Nikos Triantafyllou
University of the Aegean, i4m Lab

UNIVERSTIY OF THE AEGEAN

Blockchain Origins: A brief history

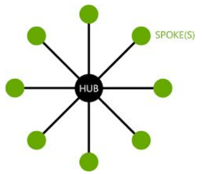
- **Satoshi Nakamoto** released the **Bitcoin White Paper** outlining a *purely peer to peer electronic cash/digital asset transfer system*
- **First** popular implementation of Blockchain
- **Ethereum**, **Hyperledger**, etc.

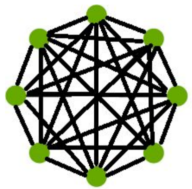
What is a blockchain?

- Distributed database that maintains a continuously growing list of transactions secured from **tampering** and **revision**.
- **Blocks** contain a **timestamp** and a **link** to a previous block.
- The first implementation of a blockchain was a public ledger of cryptocurrency transactions known as **Bitcoin**.
- This has led to the development of various decentralized platforms, which allow the execution of tamper free programs, called smart contracts, on top of such a blockchain.

What is a blockchain?

- Transactions
    - Blockchain is a historical archive of decisions and actions taken
    - Proof of history, provides provenance
- Immutable
    - once written to the chain, the blocks can be changed, but it is extremely difficult to do so
    - In DBA terms, **Blockchains are Write and Read only**
- Decentralized Peers
    - each NODE has a copy of the ledger





What is a blockchain?

- Consensus
  - Ensures that the next block in a blockchain is the one and only version of the truth
  - Keeps adversaries from derailing the system and successfully forking the chain
- Smart Contracts
  - Computer code
  - Provides business logic layer prior to block submission

Blockchain	Smart Contracts?	Language	
Bitcoin	No		
Ethereum	Yes	Solidity	
Hyperledger	Yes	Various	GoLang, C++, etc, depends
Others	Depends	Depends	

Why are blockchains useful?

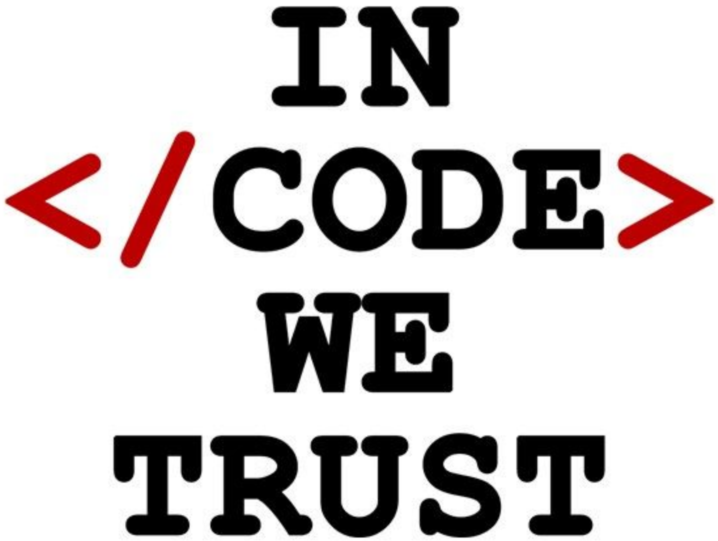
- tamper-proof, data structure
  - No central trusted authority exists
  - Participating parties do not **trust** each other
- Improved traceability
- Enhanced security
  - protecting sensitive data, blockchain has an opportunity to really change how critical information is shared by helping to prevent fraud and unauthorized activity.
- execution of smart contracts
  - **Enforce** the negotiation or performance of a contract
  - Allows for **fair-exchange** (blockchain is the mediator)
  - No direct interaction between parties
  - **Open/verifiable** business logic

In code we trust!

or
Understanding the need for Formal Methods

But Should we?

- Open business logic
- Immutability
- Verifiability



Testing strategies?

Developer:

- Unit tests
- Integration tests

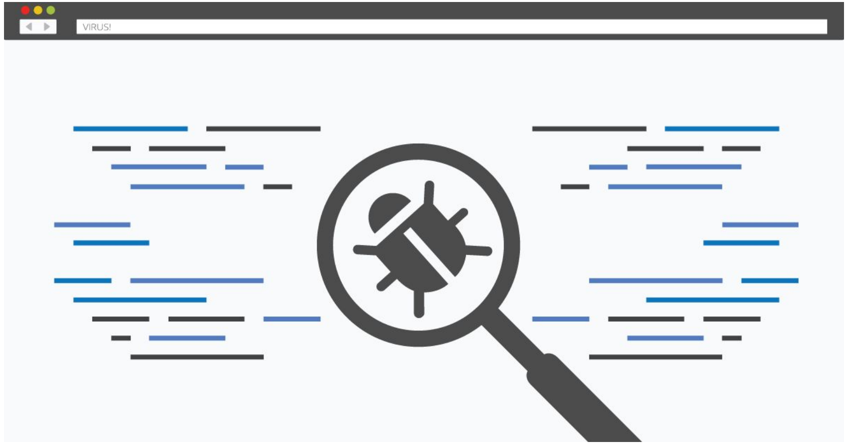
QA:

- Functional tests
- Performance tests
- Stress tests
- Failure tests

Security requires **reasoning**!

Informal Proofs

- Require deep thinking which promotes a better understanding of the system/algorithm
- Hard to get right!



High complexity
Errors (bugs) can be found in proofs as well

Automated reasoning is required!

Formal Methods

- Precise specification of system/algorithm
- Tools to validate correctness
    - Computer handles complexity and correctness
- Human intuition makes reasoning possible

$\mapsto p$

$|$ Intermediate steps

$|\ q \wedge \sim q$

$|\ \_\_\_\_\_\_\_\_$

$\therefore p \rightarrow (q \wedge \sim q)$	: CP
$\sim (q \wedge \sim q) \rightarrow \sim p$	: Transposition
$(\sim q \vee q) \rightarrow \sim p$	: DeMorgan
$\sim q \vee q$	: EMI
$\sim p$	: Modus Ponens

But Should we?

- Blind trust in critical systems is not a good idea
- Open/Verifiable code does not mean **correct code**
- **Examples**:
    - theDAO hack
    - Parity freeze
    - Parity’s multisig wallet
- **Fixing** (if possible) is very expensive (hard forks, updating clients etc.)

Maybe yes!

- Raise the bar on security
- Automated reasoning in mathematical logic to provide **additional assurances**
- Formal verification **allows us to prove conclusively that certain error states can never occur.**

Key point

- ***“The introduction of a blockchain doesn’t magically make the system secure”***
- Companies proposing to join or use blockchains should ensure that they are **designed** and **configured appropriately** and processes are supported by their own internal controls**
- **Formal Methods can help!**

**https://www.icas.com/technical-resources/the-interaction-between-blockchain-and-corporate-reporting

A case study:

RegTech Project verification

What is the deRegtech Project?

- **Based on:** Blockchain Technology, Algorithmic Financial Contract Standards, and Document Engineering methods and techniques.
- deRegTech project deploys a **permissioned blockchain** that provides a distributed ledger for **collecting**, **publishing** and **storing** **information** related to the **creation** and **evolution** of **financial contracts**.

System Overview

When a contract is agreed between two counter-parties:

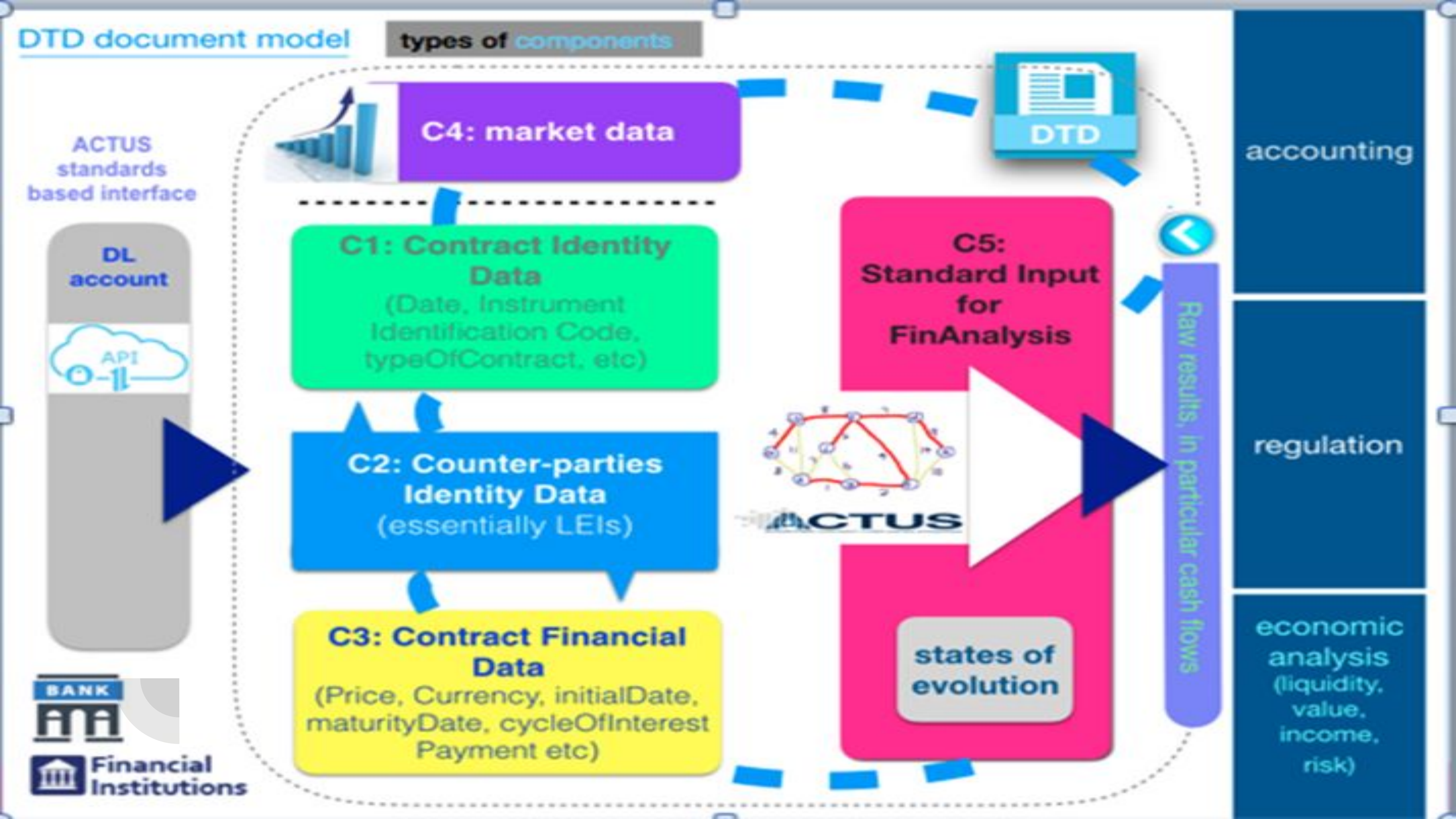
- jointly submit their report to the blockchain part of the deRegTech Service.
- **smart contracts** process these data, based on:
  - **ACTUS** standards and produce a DTD, in the form of a transaction and risk report.
  - follow a specific data model that implements a number of requirements made public recently

Regulatory Authority supervising these counter-parties can:

- obtain a list of all **reports in the system** (automatically)
- obtain for each such report all the **related information** (called state variables) for this contract.

The Regulatory Authority incorporates these data and functionality to its own financial/risk analysis system(s) to assess the risks undertaken by the counter-parties.

.


DTD document model
types of components
ACTUS standards based interface
DL account
API
BANK
Financial Institutions
C4: market data
C1: Contract Identity Data
(Date, Instrument Identification Code, typeOfContract, etc)
C2: Counter-parties Identity Data
(essentially LEIs)
C3: Contract Financial Data
(Price, Currency, initialDate, maturityDate, cycleOfInterest Payment etc)
DTD
C5: Standard Input for FinAnalysis
ACTUS
states of evolution
Raw results, in particular cash flows
accounting
regulation
economic analysis (liquidity, value, income, risk)

Important Issues

1. **Data validation**
   a. Is the information inserted in the system accurate?
2. **Access control policies**
   a. Who gains access to which part of the available information

Goals

- How can we develop a f**ormal framework for reasoning about smart contracts**?
    - Reasoning about smart contract business logic.
    - Implementing business logic correctly.
- Minimum Safety Property:

> *"It is not possible to have a "confirmed" contract in deRegTech system without the the approval of all involved parties first".*

Core Ontology for Blockchains

We can identify in a Blockchain system the following basic structures;

- **Subject**
  - The elements of the sort Subject, are used to denote the **users of the blockchain**.
- **Object**
  - Objects denote the **entities on which the actions** of the system **are applied**.
- **Actions**
  - The Action domain contains all the **actions permitted in a blockchain system**
  - The actions defined in our system are the following: createAccount, createContract, updateContract, validateContract, getReport
- **Transactions**
  - The elements of the Transaction domain denote a desire or a **request by the subject to execute an action on the object** of the transaction.

State Transition System and Blockchain

- The information contained within a Blockchain constantly changes!
- To address this, we define a new structure, called **State**, which represents the **state space** of the blockchain system.
- A new **constant** is declared, ***init*** *: → State*, which denotes the **initial state of the system** (i.e. it represents the genesis block of the blockchain).
- Three **constructor** functions are declared, which define how a **new state of the system can be derived by a previous one**, ***sendTransaction***, ***validateBlock*** *and* ***Tick***.

State Transition System and Blockchain

- **sendTransaction**: *State Transaction* → *State*, denotes that a **new transaction** is sent to the system.
- **validateBlock**: *State Transaction Transaction* → *State*, denotes that a set of received transactions were considered as valid and their actions took effect **altering the state of the blockchain** (i.e. represents the mining of a new block in the blockchain).
- ***Tick***: *State* → *State*, denotes the **passing of time** and is required because the information retrieved by a smart contract may change depending on this.

State Transition System and Blockchain

Two more functions are defined;

- ***pendingTransactions***, which denotes the transactions submitted to the system but are not yet verified, i.e. the transactions which are pending validation.
- ***objects***, which given an element of the sort State returns a set of object sorted elements and **denotes the objects that belong to the blockchain at the given state** of the system.

A blockchain can thus be thought of as a **State Transition system,** where:

- each **state consists**: of the status of the core entities of the system, and
- each **state transition function**: takes as input a previous state of the system and a transaction and gives as output a new state.

Reasoning with Algebraic Specifications

- **Algebraic specification** method is considered as one of the major **formal methods.**
- Systems are specified/designed based on **algebraic modeling**.
- The specifications/designs are tested/**verified against requirements using algebraic techniques.**
- The **behavior** of systems can be nicely modeled by **algebras**.
- CafeOBJ is an algebraic specification language.

Formal verification of the desired goal

- Using the OTS/CafeOBJ approach, we successfully verified that the specification satisfies the desired system property.
- The full specification of the proposed system and the proofs can be found at CafeOBJ@NTUA [https://cafeobjntua.wordpress.com/].

Key Takeaways

- Blockchains build trust
- To trust code, testing is not enough
- Blockchain **benefits** come at a cost:
    a. Design Error Resilience
- Formal Methods could be a feasible answer to addressing this problem
    a. Correctness by Design Engineering
- Risk reporting using a blockchain is feasible
- May **aid** regulatory authorities and society at large in overshighting the global financial system

Who is involved?

- Petros Kavassalis, University of the Aegean, Information Management Lab (i4M Lab), pkavassalis@atlantis-group.gr
- Harris Papadakis, University of the Aegean, Information Management Lab (i4M Lab), adanar@atlantis-group.gr
- Petros Stefaneas, National Technical University of Athens, Logic and Formal Methods Group (λ-ForM), petros@math.ntua.gr
- Katerina Ksystra, University of the Aegean, Information Management Lab (i4M Lab), katerinaksystra@aegean.gr
- Nikolaos Triantafyllou, University of the Aegean, Information Management Lab (i4M Lab), triantafyllou.ni@aegean.gr

Thank you for your attention!

Questions?