# Literature review of Formal Methods applied to Blockchain technology

Henoch Vitureira
Instituto Politécnico de Setúbal
Setúbal, Portugal
2016041081@estudantes.ips.pt

## ABSTRACT

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum enim nibh, aliquet sit amet ex suscipit, efficitur fringilla enim. Vestibulum elementum fringilla sodales. Cras malesuada sollicitudin consectetur. Cras elit nibh, tempus elementum dui vitae, sagittis porttitor elit. In sollicitudin purus neque, eu tristique felis varius in. Morbi sed commodo eros. Pellentesque ornare sed felis et bibendum.

## CCS CONCEPTS

• **Software and its engineering** → **Process validation**; • **Security and privacy** → **Cryptography**.

## KEYWORDS

Software Quality, Formal Methods, Blockchain, Smart contracts

## 1 INTRODUCTION

The rising complexity of features offered by software gives us the ability to use it for the most integrity and reliability dependent needs of our lives. Information systems that rely on these base principles can be financial and health related services, and in most recent times, blockchain based applications. Blockchain, being a decentralized and distributed network protocol that can be used as the core of a monetary system that performs peer-to-peer transactions, it needs the assurance of coordination and consensus of its economy's state [1]. Knowing that the protocol of a given blockchain project is what dictates how transactions and their coordination is maintained, the assurance of security and safety is a given. Formal methods can be introduced in order to provide unequivocal evidence that a given blockchain system and its consensus algorithm are secure and in according to expected software quality.

Formal methods are a rigorous description of a system or process using mathematics that aims to provide evidence of its reliability and robustness, according the specification in question [6].

Contrasting with normal testing, formal verification is performed during the design process, it ensure the design is correct and allows the verification of any situation, while regular testing cannot test for every possible input [5].

This paper is organized as follows. Section 2 introduces applications of formal modeling of blockchain technology. Section 3 expands on the use of formal mathematical descriptions of a blockchain's consensus rule. Section 4 explored how formal modeling can be used to verify and assure quality to smart contracts using Colored Petri Nets (CPN). Section 5 elaborates on the security requirements of a blockchain system. Section **??** is a brief conclusion.

## 2 FORMAL MODELING APPLICATION

Blockchain systems can be divided into five main layers [1]: Data layer, Consensus layer, Smart contract layer and Application layer. As seen on Figure 1.

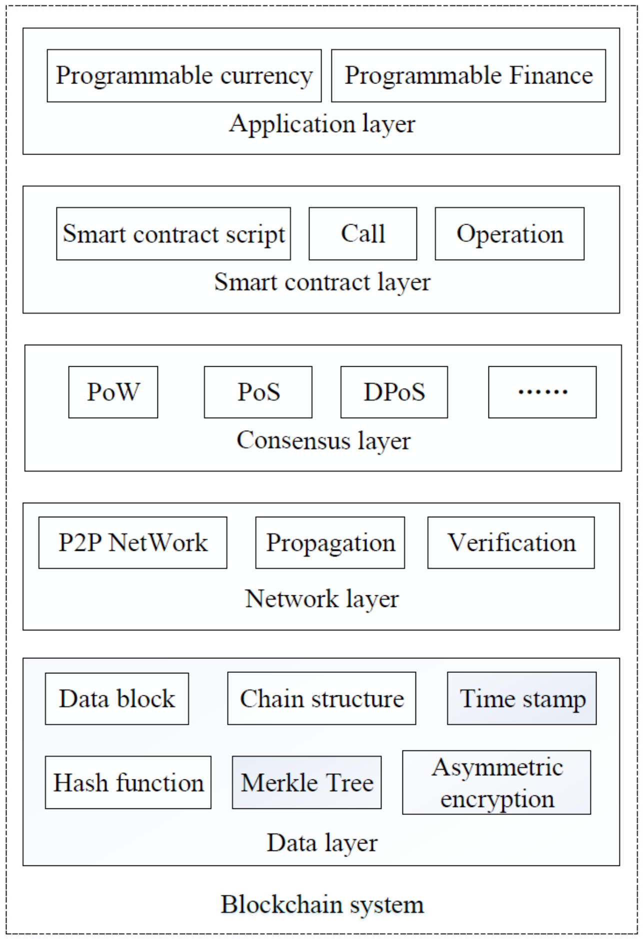

**Figure 1: Blockchain infrastructure, from [1]**

The data layer is responsible for block generation, the construction of the blockchain and storage [1]. The network layer connects the data and filters packets, and also manages the nodes. The rules of how the blockchain protocol must work are defined on the consensus layer. This includes agreement between nodes, fault tolerance and data consistency. Smart contracts have their execution lifecycle on the smart contact layer. The application layer is the upper level layer of used applications such as games or decentralized financial services.

It is possible to use languages to define the standards of a blockchain system with mathematical rigor. One of them is SDL (Specification and Description Language), an Object-Oriented formal language defined by the International Telecommunications Unit. It is capable of mathematically describing the structure, behavior, and data of real-time and distributed communicating systems. It is also used by IEEE for standard definition [1].

With SDL, we can describe a structure at the levels of system, block or process. A system represents the object to describe. It is responsible for the communication with the outer environment through the appropriate channels. A system can contain one or more blocks and blocks contain one or more processes. The majority of the functionality of an SDL system is described on the process level. Comparing to conventional programming languages, modules are represented by blocks in SDL, and funcionalities are represented by processes. In [1], the authors propose a hierarchical model focused on the consensus layer.

## 2.1 Data Structure

In [1] the modelled blockchain system used an Improved Byzantine Fault Tolerant Algorithm as the consensus algorithm. To define the data structure of a block in SDL, we need to represent them as a pair with a header BT and body BH, where:

$$B \equiv (B_H, B_T) \quad \text{with} \quad B_H \equiv (\mathbb{H}_p, \mathbb{H}_a, \mathbb{H}_t, \mathbb{H}_m, \mathbb{H}_h)$$

$$\text{and} \quad B_T \equiv (T_1, T_2, ..., T_{\mathbb{H}_l})$$

In these expressions, $\mathbb{H}_p$ represents the hash value of the previous block, $\mathbb{H}_p$ the hash value of the current block, $\mathbb{H}_t$ it's timestamp, $\mathbb{H}_m$ the merkle tree root (the hash of the transactions of a block), and $\mathbb{H}_h$ the current block's height.

The the block' structure definition can be seen on Listing 1.

```
newtype blockchain struct
  prehash integer;
  hash integer;
  length integer;/*the length of blocks*/
  merkleroot integer;
  ti Time;/*Timestamp*/
  translist list;
endnewtype;

newtype list
 array(maxit, integer)
endnewtype;

syntype maxit =Integer constants 0:25
endsyntype;
```

**Listing 1: Definition of a block's structure**

A blockchain can be regarded as a state machine [1] that contains a starting state, a non-empty state, the input transaction set, a state transition function, and an acceptance state set. Which can be formally defined as:

$$S \equiv (Q, \Sigma, \delta, s, F)$$

Where:

- Q is the non-empty state set, which represents all the states.
- $\Sigma$ is the set of the newly generated and consensus blocks.
- $\delta$ represents the state transition function, $\delta : Q \times \Sigma \longrightarrow Q$.
- s would be the starting state, which is the state of the system when it initializes. $\delta : s \in Q$.
- F representes the acceptance state set. $F \subseteq Q$.

The state of the model at the first block generation is the starting state. As transactions are made, the model a leader node to generate the block, which is transmitted to the other nodes on the blockchain network, triggering its validation for consensus. A consensus block stores the hash of the previous one and adds it to the end of the blockchain and completes the transfer of the blockchain state.

*2.1.1 Diagrams.* With a modeling language like SDL, we can use diagrams to define blockchain structural entities, like the core system of the consensus layer and blocks. Process flowcharts can model the flow of information and input validation, making a process like node validations of blocks and its subprocess of synchronization and voting can clearly illustrated.

## 3 CONSENSUS RULE

Blockchain system may choose to use consensus rules taken from popular protocols such as Bitcoin, but an alternative rule can be implemented, and also be formally modelled [2].

On a consensus rule, fault tolerances requirements are of utmost importance, including the collusion of nodes to alter a ledger. In [2], the author discusses a way to formally describe these tolerances with a scalable verification.

Endorsement policies, the rule that define the criteria of agreement os transaction results between nodes, can be described as a threshold function as follows:

$$T(m, \text{node}_1, ..., \text{node}_n), m \in \mathbb{N}$$

Where $T(2, \text{node}_1, \text{node}_2, \text{node}_3)$ would be an endorsement policy that means "among the three nodes in the argument, at least two nodes must return the same execution result". With this, we can mathematically model faulty state verification.

## 3.1 Faulty States

Let $\mathbb{B}$ and $\mathbb{N}$ be sets of Boolean and natural numbers, respectively. The state of the $i$-th node can be defined as: $s_i = (s_i.e, s_i.v)$, where for each transaction execution, $s_i.e \in \mathbb{B}$ is the existence of the result, and $s_i.v \in \mathbb{B}$ is the value of the result (correct or wrong).

Using the threshold function previously shown, $T(m, s_1, ..., s_n) = (T.e, T.v)$. With this, let

$$H_= \left( \sum_i^n I(s_i.e \wedge (s_i.v = b)) \geq m \right)$$

Where:

- $m$ is a threshold

- $b \in \mathbb{B}$
- $I : \mathbb{B} \mapsto \{0, 1\}$ is the indicator function

With the expression above, the author of [2] infers:

$$T.e = H_{\text{true}} \vee H_{\text{false}}, \quad T.v = \neg H_{\text{false}}$$

The endorsement policy $EP = (EP.e, EP.v)$ can be formed with $T$. When $EP.e$ = true, the policy is considered accordingly followed and the involved ledger is updated with the value of $EP.v$.

Mathematical descriptions like the ones above can also be applied to the fault tolerance of organizations participating in the blockchain in question. With formal modeling, a consensus rule can be tested and benchmarked after giving the test case some arbitrary requirements that are in conformity with the scope. The test results are used to judge the viability and scalability of the rule.

## 4 SMART CONTRACTS

Smart contracts are the backend code of decentralized applications that live in blockchains such as Ethereum. This code cannot be changed once it is deployed to a production Net, so quality must be assured in order to guarantee that specifications are followed and that vulnerabilities are inexistent [5]. With the logic and mathematical modeling of formal verification, we avoid edge test cases that involve unforeseen inputs or conditions.

A smart contract's execution after its required verifications originates a transfer on a blockchain. Ethereum is a blockchain system that allows is to deploy decentralized applications based on smart contracts, which are written in Solidity [5]. Solidity has a limitation that makes smart contracts have a no parameters, no return value *function()* that is anonymous and defined as the fallback function [3]. When a contract sends Ether (the Ethereum blockchain token) but does not not call any function, the fallback function is called. If this function does not exist int he contract, the transaction is cancelled. This limitation can be used by hackers to exploit the contract, like the DAO smart contract exploit in 2016 [3].

A formal verification method using CPN modeling is proposed in [3]. In a scenario where a crowd funding smart contract (CFD) is considered, [3] proposes a two layers contract modeling, the CFD layer model that establishes the overall structure of the contract, and a bottom layer that contains the AddTo layer and the WithDraw layer, which implement deposit and withdrawing functions.

The CFD layer, shown on Figure 2, presents the entity Account, that represents the user, Bank that represent the contract account, and Withdraw and AddTo, that represent withdrawing and depositing operations.

The AddTo layer, shown on Figure 3, models the user's depositing operations, where there are 3 transactions and 8 entities in the model that describe the steps of a given transaction in the blockchain and it's conditions.

A similar to the on seen on Figure 3 was also presented in [3] for the WithDraw layer to describe the inherent processes of withdraw operations and all their conditions.

Models like the proposed CPN model can be tested for resilience against malicious behavior using another model that attempts to

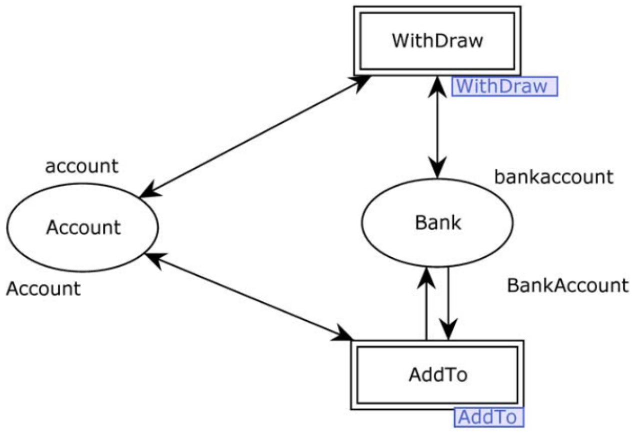

Figure 2: CFD Layer, from [3]

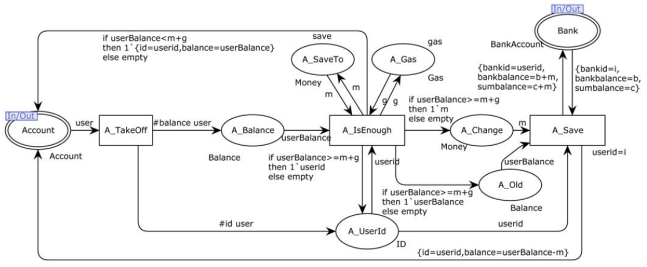

Figure 3: AddTo Layer, from [3]

exploit the CFD contract. We can plan our attack model to test any operation and condition, such as balance altering, assurance of reception of funds, etc. Smart contract's vulnerabilities can be taken advantage of by regular users executing expected operations, not only by badly intended actors with knowledge in computer science, so it makes sense that scenarios of no attack can also be part of the model checking. CPN tools like Space State Tools can used to see the status of each step of a contract's execution during a test scenario [3], easing the discovert of vulnerabilities.

## 5 SECURITY

As mentioned in section 2, formal analysis and verification is used to assure security of a blockchain's core structures and consensus rule. But we can cover the security-requirements for backend blockchain protocol and mechanisms to ensure the assumptions and scripting language [4].

In [4], the authors enumerate security requirements of a blockchain system by their layers, consisting of cryptography layer, backbone protocol, application protocol, application logic, implementation and operation. These requirements can be seen in Table 1.

We can implement formal methods in the layers Implementation, Backbone protocol, Application protocol and Language for Smart Contract. In the implementation layer we have both software and hardware implementation pf security mechanisms, including the

**Table 1: Blockchain Technology layers and security considerations**

| Layer | Security Consideration | Standard |
|---|---|---|
| Operation | Key Management, Audit, Backup | ISO/IEC 27000 |
| Implementation | Program Code, Secure Hardware | ISO/IEC 15408 |
| Application Logic | Scripting Language for Financial Transaction, Contract | Secure coding guides |
| Application Protocol | Privacy protection, Secure transaction | ISO/IEC 29128 |
| Backbone Protocol | PSP, Consensus, Merkle Tree | ISO/IEC 29128 |
| Cryptography | ECDSA, SHA-2, RIPEMD160 | NIST, ISO |

cryptographic algorithm, protocols and key management mechanisms. In the Backbone protocol and application protocol deals with cryptographic protocols. The security of the source code of a smart contracts are also mention in [4], which can be modeled and verified according to requirements, as shown in section 4.

## REFERENCES

[1] Zhangbo Duan, Hongliang Mao, Zhidong Chen, Xiaomin Bai, Kai Hu, and Jean Pierre Talpin. 2018. Formal modeling and verification of blockchain system. *ACM International Conference Proceeding Series* 86 (2018), 231–235. https://doi.org/10.1145/3177457.3177485

[2] Ryo Kawahara. 2020. Verification of customizable blockchain consensus rule using a formal method. *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2020* (2020), 14–16. https://doi.org/10.1109/ICBC48266.2020.9169472

[3] Zhentian Liu and Jing Liu. 2019. Formal verification of blockchain smart contract based on colored petri net models. *Proceedings - International Computer Software and Applications Conference* 2 (2019), 555–560. https://doi.org/10.1109/COMPSAC.2019.10265

[4] Shin'Ichiro Matsuo. 2017. How formal analysis and verification add security to blockchain-based systems. *Proceedings of the 17th Conference on Formal Methods in Computer-Aided Design, FMCAD 2017* (2017), 1–4. https://doi.org/10.23919/FMCAD.2017.8102228

[5] Yvonne Murray and David A. Anisi. 2019. Survey of formal verification methods for smart contracts on blockchain. *2019 10th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2019 - Proceedings and Workshop* 2 (2019), 1–6. https://doi.org/10.1109/NTMS.2019.8763832

[6] Gerard O'Regan. 2014. *Software Quality Assurance Software Quality Assurance.* 143–150. https://doi.org/10.1007/978-3-319-06106-1_9