On Blockchain

Banan Mozher Mohammed Al-tayeb (banan@aims.ac.za)
بنان مظهر محمد الطيب
African Institute for Mathematical Sciences (AIMS)

Supervised by: Prof. J. W. Sanders
AIMS, South Africa

23 May 2019
Submitted in partial fulfillment of a structured masters degree at AIMS South Africa

Abstract

Blockchain has the potential to revolutionize several major applications on the Internet. It is thus crucial that a good understanding of blockchain exists for its appropriate use. In this essay, we give a formal specification to enhance the understanding and prove the correctness of blockchain. We also lay the groundwork to formally specify an abstract version of a cryptocurrency that uses a public blockchain. Finally, we mention some non-cryptocurrency applications of blockchain and provide details of the specification.

Keywords: Blockchain · Cryptocurrency · Z Language · Formal Specification

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

Banan

Banan Mozher Mohammed Al-tayeb, 23 May 2019

Contents

1. Introduction

In his paper Nakamoto (2008) he presented to the world a digital currency that could work without any central authority. His work was driven by frustration with the way central banks operated in America at that time. The first cryptocurrency used a novel technology named blockchain to decentralize the work of central banks. Blockchain is simply a distributed database stored on multiple independent sites which agree on what goes on the blockchain using a computationally taxing consensus mechanism. Since Nakamoto's paper blockchain has been used to support more than just the financial record keeping of bitcoin. It has been used in applications ranging from identity management to a smart storage solution.

There are many blog posts, books, and articles intending to explain blockchain. However, most of the materials we encounter explain it informally and with insignificant mathematical descriptions. In this essay we introduce some formal specifications to an abstract version of blockchain applied in bitcoin and related cryptocurrency protocols. Blockchain technology pulls from several fields which include cryptography, computer science, and mathematics. Although blockchain has attracted the interest of many researchers, but it seems that there is no strong theoretical foundation of blockchain.

The absence of such theory makes learning the technology thoroughly challenging. It also makes creating critical applications on top of it risky, because there is no formal study giving developers of applications a strong foundation to build on. Therefore, mathematics being an appropriate language of descriptions, it is essential to develop the theory behind blockchain using mathematics. With mathematical tools such as Z notation we are able to, for instance, specify the way blockchain works, explore its properties, and ensure that it works as expected and required.

In this work, we will investigate how blockchain works, and provide examples of its applications. Furthermore, we study the basic concepts behind a cryptocurrency and how it uses blockchain as a transaction keeping ledger. By keeping ledger we mean recording the history of transactions.

The rest of this essay is organised as follows. The first chapter reviews the background of blockchain. In the second chapter, we formally specify an abstract version of bitcoin using Z notation and Object Z. The last chapter concludes and gives some suggestions for future work.

2. Background

2.1 Blockchain Technology

A cryptocurrency is a digital currency in which encryption techniques are used to organize and control the generation of currency units and verification of funds transfer. Cryptocurrencies are backed using the so-called distributed ledger technologies. These technologies enable a consensus of replicated, shared, and synchronized digital data geographically spread across multiple sites, countries, or institutions. In this way, cryptocurrency systems operate independently of governing bodies such as a central bank.

Blockchain is one of those distributed public ledgers that enables peer-to-peer exchange in a secured, public and non-repudiable way. It can be programmed to record not just financial transactions but virtually everything of value (Kehrli, Accessed on May 2019a) (Tapscott and Tapscott, 2016). Blockchain came as a way to make a database both secured and widely distributed. It does so by storing blocks of information that are identical across its network, so blockchain cannot be controlled by any single entity and has no single point of failure.

Almost every node in the network stores its "own" copy of blockchain and each block contains a timestamp and a link to a previous block, to form a chain.

Two important properties result from these:

- Transparency: data is embedded within the network as a whole, by definition it is public.
- It cannot be corrupted: altering any unit of information on blockchain would mean using a huge amount of computing power to override the entire network (Rosic, Accessed on May 2019).

Since the launch of bitcoin in 2009, the structure and functionality of blockchains has evolved over the years. Blockchain experts split the development of blockchain into three main versions, which we discuss, briefly, next.

2.1.1 Blockchain versions. Blockchain 1.0 is the main component for most cryptocurrencies currency transfer like bitcoin.

Blockchain 2.0 is an evolution of blockchain protocol enabling not only to exchange transactions but rather code and programs in the form of smart contracts (Swan, 2015) (Kehrli, Accessed on May 2019b). This allows the decentralized transaction ledger functionality of blockchain to be used to register, confirm, and transfer all manner of contracts and property (Swan, 2015).

Blockchain 3.0 is blockchain decentralized applications beyond currency, finance, and markets particularly in the areas of government, health, science, literacy, culture, and art (Swan, 2015).

2.1.2 How blockchain works. Cryptocurrencies use blockchain to complete its transactions. Figure (2.1) illustrates how this is achieved. Let us consider the case where Alice wants to pay Bob in a certain cryptocurrency. In order to complete the transaction, Alice (buyer) creates and signs a transaction specifying the receiver and the desired amount. The transaction is then broadcast on the network, validated and retransmitted by every node (it reaches) in the network until it reaches (almost) every node. Alice's transaction and another group of valid transactions is made into a block by a miner. The block is then broadcast to every party in the network. Each node approves that the block is valid, and adds the block to its own blockchain. After the block containing Alice's transaction is recorded in blockchain and followed by enough new blocks, the transaction is then considered permanent and the recipient of the transaction, Bob, can spend the money sent to him.

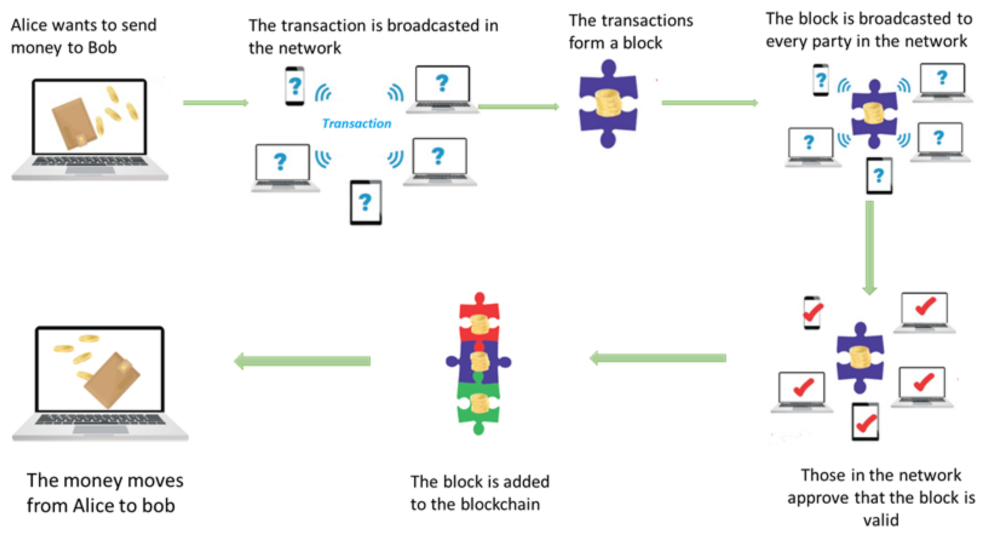
Alice wants to send money to Bob
The transaction is broadcasted in the network
Transaction
The transactions form a block
The block is broadcasted to every party in the network
Those in the network approve that the block is valid
The block is added to the blockchain
The money moves from Alice to bob

Figure 2.1: A simplified overview of how an exchange between two parties is recorded in blockchain. Original image http://cdn.fm.co.za/fm/images/Blockchain2303.png

After looking at how blockchain works, we explore some of the applications built on top of the technology.

2.1.3 Blockchain applications. Blockchain technology, with its three versions, has many different and interesting applications. We mention some of them here.

- **Cryptocurrencies**

  Cryptocurrencies rely on blockchain to store and execute transactions.

- **Time Stamped Proof of Existence**

  In theory, information on blockchain is immutable. This fact enables a simple yet strong application to basic versions of blockchain. MetroGnomo is an open-source project based on blockchain like distributed ledger, which provides its users with a timestamped proof that a document (or an event) existed at the stamped time (Z/Yen Group, Accessed on May 2019). The project can support a wide range of real-world use cases and has even developed an application prototype to provide proof of physical visits to locations by saving geolocation coordinates (MetroGnomo.com, Accessed on May 2019).

- **Smart Contracts**

  Smart contracts are autonomous computer programs that, once started, execute automatically and mandatorily the conditions defined beforehand in the contract terms (Crosby et al., 2016).

  The Ethereum project is enabling smart contracts using blockchain technology. Ethereum aims to implement a globally decentralized, un-own-able, digital computer for executing peer-to-peer contracts (Kehrli, Accessed on May 2019a).

- **Distributed Registry**

  Blockchain technology can be used to protect and decentralize information with a strong focus on maintaining user anonymity. Governments, businesses and even schools can store different forms of data safely with no fear of alteration or loss (Snow et al., 2014).

- **Distributed Cloud**

  Blockchain allows for the implementation of a distributed cloud in a way that significantly increases security, privacy, and data control, without the need for a central third party (Wilkinson et al., 2014). Companies like Storj and Codus used blockchain to implement distributed cloud services.

- **Decentralized Messages**

  Using blockchain to enable peer-to-peer Encrypted Instant Messaging system that keeps its users' privacy. Origin, a decentralized marketplace DApp, introduced its implementation of such a system in a blog post by Micah Alcorn (Accessed on May 2019).

- **Decentralized Voting**

  Blockchain enables a vote to be non-modifiable, non-repudiable and cannot be registered in multiple ways. Thus, making every online voting initiative as secure and verifiable as a public election (Reply, company).

- **Protection of Intellectual Property**

  Blockchain can be used for registering and protecting intellectual property (IP)(Rosic, Accessed on May 2019). The emerging digital art industry offers services for privately registering the exact contents of any digital asset (any file, image, health record, software, etc.) to blockchain (Swan, 2015).

- **Internet of Things (IoT)**

  Blockchain technology solves the scalability, privacy, and reliability concerns in the Internet of Things domain. Blockchain technology can be used in tracking billions of connected devices, enabling the processing of transactions and coordination between devices.

  This decentralized approach would eliminate single points of failure, creating a more resilient ecosystem for devices to run on. Furthermore, the cryptographic algorithms used by blockchain would make consumers' data more private (Malviya, 2016).

Although, blockchain technology is only 10 years old, it still has many more applications than mentioned here. However, it is crucial not to become infatuated with the idea behind blockchain and to stay critical about where blockchain can make the biggest impact and be of greater usage.

In the following section, we look at the main components of a cryptocurrency and how they come together to make a real transaction.

2.2 How Cryptocurrencies Work

In this section, we give a general description of how cryptocurrencies work. The definitions and explanations included are a generalization of their definition in bitcoin and applicable to a wide range of other popular cryptocurrencies.

2.2.1 Cryptocurrency basic concepts. All cryptocurrencies share some common basic concepts. These concepts are important for understanding how cryptocurrencies work and so we will explain them in this section.

Keys

Every cryptocurrency user has two type of keys, a public key (address) used for receiving coins in the network and a private key used for signing user transactions. Both keys are generated using cryptographic algorithms like elliptic curve cryptography (Antonopoulos, 2014).

Wallet

Software that holds all cryptocurrency user's public and private keys. It is used to send, receive, and store coins from the cryptocurrency (Antonopoulos, 2014). There are many different types of wallet available for users

- *Software (desktop) Wallet* is a wallet installed on the user's computer that is controlled by him. Ideally, only the user has access to the private keys that store his cryptocurrency. This type of wallet usually stores the entire blockchain on the user's computer, making him a full node. A software wallet that does not store the entire blockchain is called a Nano wallet (DeMartino, Accessed on May 2019).
- *Mobile Wallet* is a wallet that is run from a smartphone app (DeMichele, Accessed on May 2019).
- *Online Wallets* are web-based wallets where the user's data is hosted on a real or virtual server. Some online wallets are hybrid wallets allowing encryption of private data before being sent to the online server (DeMichele, Accessed on May 2019).
- *Hardware Wallet* use dedicated hardware that is specifically built to hold cryptocurrency and keep it secure. This includes USB devices. These devices can go online to make transactions and get data and then can be taken offline for transportation and security (DeMichele, Accessed on May 2019).
- *Paper Wallets* are made by users by printing a QR code for both public and private keys. This allows them to both spend and receive cryptocurrency using a paper wallet. However, they are limited in use because they cannot be used to make online transactions (DeMichele, Accessed on May 2019).

Transaction

A transaction is a transfer of cryptocurrency from one address to another. More precisely, a transaction is a signed data structure expressing a transfer of value (Antonopoulos, 2014).

Transactions usually contain the transaction version, the number of inputs and outputs, the transaction creation time, where inputs are the outputs of some older transactions in the network.

There are two type of transactions

- *Transfer transaction* is a normal transaction to transfer coins from the sender to receiver.
- *Coin Issuance (Coinbase) transaction* is a transaction that contains newly mined coins. They are included in every block as the very first transaction and are meant as a reward for solving a proof of work puzzle (Antonopoulos, 2014).

Difficulty

A network-wide setting that controls how much computation is required to produce a proof of work (Antonopoulos, 2014).

Difficulty retarget

A difficulty at which all the computation in the network will find blocks (Antonopoulos, 2014).

Miner

A miner is a network node that finds valid proof of work for new blocks, usually with very powerful CPU (Antonopoulos, 2014).

Mining

Mining is the process of creating a new block using proof of work and is done by the miner (Antonopoulos, 2014).

Consensus Mechanism

"In distributed ledgers, a consensus mechanism is the way in which a majority, (or, in some mechanisms, all) of network members agree on the value of a piece of data or a proposed transaction, which then updates the ledger. In other words, a consensus mechanism is a set of rules and procedures that maintain a coherent set of facts among the participating nodes (Seibold and Samman, 2016)." The most famous consensus mechanism used in cryptocurrencies is proof of work.

Proof of Work is a piece of data which is difficult (costly, time-consuming) to produce but easy for others to verify and which satisfies certain requirements (BitcoinWiki, Accessed on May 2019b). For a transaction to be complete a hard-mathematical problem must be solved.

Transaction fees

The sender of a transaction often includes a fee to the network for processing the requested transaction (Antonopoulos, 2014).

Coin generation

A certain number of coins is included in each new block as part of the reward given to the first miner who found the proof of work solution. This is the way new coins are generated in cryptocurrencies (Antonopoulos, 2014).

Reward

A miner is given a reward for each new block they add to blockchain. This reward consists of transaction fees and newly generated coins (Antonopoulos, 2014).

Block

A grouping of transactions, marked with a timestamp, and a fingerprint of the previous block (Antonopoulos, 2014).

Confirmed Transaction

A transaction gets confirmed when it is included in a block in blockchain, and it has one confirmation (Antonopoulos, 2014).

Confirmation

Once a transaction is included in a block, it has one confirmation. As soon as another block is mined on the same blockchain, the transaction has two confirmations, and so on. Six or more confirmations is considered sufficient proof that a transaction cannot be reversed (Antonopoulos, 2014).

Genesis block

The first block in blockchain used to initialize the cryptocurrency (Antonopoulos, 2014).

Network

A peer-to-peer network that propagates transactions and blocks to every cryptocurrency node on it (Antonopoulos, 2014).

Full node

A node in the network that stores the full content of the cryptocurrency's blockchain (Antonopoulos, 2014).

Untraceability

For each incoming transaction, all possible senders are equally probable (Saberhagen, 2013).

Unlinkability

For any two outgoing transactions, it is impossible to prove they were sent to the same person (Saberhagen, 2013).

2.2.2 How it all works. To understand how a cryptocurrency works we look at an example of a transaction in a cryptocurrency system. Suppose that Alice wishes to send x amount of money to Bob. Bob shares his address (publicly) with Alice. Alice then uses her wallet to create a transaction in which she sends x coins to Bob plus the transaction fees. The wallet then broadcasts the transaction in the network by sending it to nodes connected with it. When a node receives a valid transaction it forwards it to other nodes in the network, otherwise, a transaction is ignored. This process continues until (almost) all nodes in the network have received the transaction.

Once the transaction reaches a miner, they validate it and add it to their transactions pool (mem-pool). The miner adds the transaction to transactions from other users and attempts to create a valid block by solving the proof of work problem or other kinds of consensus mechanism used in the cryptocurrency. Miners add transactions to the current block that they are mining to collect the transactions' fees. When the miner successfully creates a valid block, they broadcast it to connected nodes, where these nodes validate the block and send it to nodes connected with them. If the node is a full node it then adds the block to its copy of blockchain. Alice's transaction is considered confirmed, and the money is transferred from Alice to Bob when the block containing Alice's transaction is added to blockchain. After that, Bob has to wait for a certain number of blocks to be added to blockchain before spending the money sent to him by Alice in a new transaction.

Nodes use proof of work to identify valid blocks. Only valid blocks are part of blockchain. However, since many miners around the world are working on creating (mining) valid blocks, every once in a while two valid blocks are created at the "same time". The protocol in such a situation is that a node will consider the block which it received first to be the valid one. This creates a "fork" in a cryptocurrency's blockchain, where the network's nodes disagree on the most recent blocks. This can go one for some time until one of the two blockchains surpasses the other in length.

Nodes consider the longest chain to be the valid one and all nodes switch over to the longest chain, resolving the fork. In bitcoin the longest chain is the chain which took the greatest amount of computational power to create. This definition prevents a malicious node from making a chain with the highest number of blocks but with low difficulty (BitcoinWiki, Accessed on May 2019a).

The way that bitcoin and other cryptocurrencies work has made it possible for it to acquire several properties, some of which are necessary for digital currency. We discuss them in the upcoming section and how they follow from blockchain characteristics.

2.3 Properties of Cryptocurrencies

Bitcoin uses blockchain as a ledger to store its users' information. As we have mentioned previously in Section (2.1) bitcoin's blockchain is publicly stored by every full node in the network and every block in blockchain contains the hash of the previous block's header. This gives the network and cryptocurrency several properties. Some of these properties are:

- **Traceability and Linkability of Transactions**: Transactions in bitcoin contain the sender's and receiver's public address. So every transaction becomes fully traceable and linkable in terms of the definitions in Subsection (2.2.1). Which may pose a threat to the user's privacy, as an entity who has access to your public key will be able to track all of their transactions and find out their financial history (in the network). To counter this threat it is recommended to use a different public address for every new transaction. Nonetheless, a user's true identity can be still be revealed. In (Reid and Harrigan) they show how they were able to deanonymize some of the addresses in the bitcoin network, successfully.

  Another consequence of transactions being fully traceable and linkable is that the coins of a cryptocurrency become non-fungible. It means that, unlike fiat currencies, every coin in bitcoin has its entire history attached to it. Which may make the value of different coins differ depending on the type of history they have.

- **No Double Spend (Spend Once)**: Every digital piece of data can be replicated an infinite amount of time with almost no cost. This creates one of the biggest challenges facing digital currencies since the dawn of the Internet. Given that, how can it be guaranteed that a malevolent entity will not successfully resend or re-spend money in a digital currency system? Bitcoin guarantees this by using blockchain. Let us consider a scenario where a malevolent entity tries to double spend inside of bitcoin.

  For a transaction to be considered valid its inputs must be the outputs of some older transactions without being the input of any other valid transaction in bitcoin's blockchain at the same time. If a scammer is trying to double spend some coins; he will have to convince two different parties that he paid them. To do this he will create two transactions with the same input and two different outputs. Then, he will send one of the transactions to the network, including one of the sellers (outputs). Since the first transaction is valid it will quickly become part of a block in bitcoin's blockchain. The seller of the first transaction can confirm that he was paid, and sends the product or service to the scammer. When the scammer receives the product he will "quickly" mine a new block containing the second transaction to replace the block with the first transaction. This event will fork bitcoin's blockchain and the scammer has to mine another block before other miners to make its version of blockchain the longest one. If everything goes well the scammer would have successfully spent his coins twice. Furthermore, he will have scammed the first seller; since the block containing the transaction paying him has been overwritten.

The scenario above outlines how to scam in bitcoin. However, using patience and the difficulty of mining the threat can be easily averted. Simply, the first seller will have to wait a full hour before sending the scammer the product. This builds up enough work (6 or more blocks) on top of the transaction paying him that it becomes computationally impossible for the scammer to replace all the blocks after the block containing it. This is especially true since all the other miners in the network are mining new blocks all the time.

In the case that the scammer is able to secure 51% of the computational power in the bitcoin network he would be able to produce blocks much faster than the rest of the honest miners and waiting for an hour will not stop him from double spending. Nonetheless, it will be extremely expensive and impractical to try to gain such computational power just to double spend. It would be more profitable for the scammer to use his power to mine honest new blocks and collect their rewards.

- **Byzantine Fault Tolerance:** If a node in the bitcoin network fails (goes offline or has high latency) the rest of the network is unaffected and continues to work following the network protocol as if the failed node was never part of the network. When the node is repaired it can easily rejoin the network and resume its activities by first catching up to the rest of the network and requesting blocks that it missed. Therefore, the cryptocurrency network is Byzantine fault tolerant (Wikipedia, Accessed on May 2019), which is a result of using proof of work as the consensus mechanism. When a node receives a block it can independently of other nodes verify its validity and make the correct decision regarding adding it to its blockchain. So most nodes are in consensus about bitcoin's blockchain most of the time. A Byzantine fault describes a condition of a computer system, specifically distributed, when the system fails to determine the failure status of a component. The cause of confusion in the system is that the component could have different states for different components, which makes reaching a consensus on its state troublesome. A computing system that is able to correctly handle such a condition is called Byzantine fault tolerant (Wikipedia, Accessed on May 2019).

In this chapter, we gave an overview of what a blockchain is and how cryptocurrencies use it to function. We also discussed blockchain applications and some of the properties cryptocurrencies acquire from using blockchain technology. In the next chapter we will formalize the concepts discussed here using Z notation and Object Z.

3. Specification in Z Notation

In this chapter, we give a specification for a general cryptocurrency, as described in the previous chapter using, Z notation and Object Z. The specification abstracts the different components of the bitcoin network. At the same time, we specify some parts as they are in bitcoin. We start by specifying an abstract digital currency.

3.1 An Abstract Digital Currency Specification

In this section, we formally specify an abstract digital currency. The specification includes a centralized ledger which models a database that stores the users' data. In addition, we specify only one operation: the creation of a transaction between two parties.

The state of the currency is described by the function cl which has a domain of all the ids of users in the ledger, and the positive real numbers as its range. The function binds every id to its balance in the digital currency. The Z formal specification is given by schema SysState below.

$$\begin{array}{|l}
\multicolumn{1}{l}{\textit{SysState}} \\ \hline
cl : IDs \mathrel{+\!\!\!\!\to} \mathbb{R}^{\geq 0} \\ \hline
\end{array}$$

A transaction in the system needs the buyer's and seller's ids and the amount sent. Every successful transaction will modify the state of the ledger, and hence will change the values returned by cl when given the buyer and seller as inputs.

$$\begin{array}{|l}
\multicolumn{1}{l}{\textit{Transaction}} \\ \hline
\Delta SysState \\
buyer?, seller? : IDs \\
amount? : \mathbb{R}^{>0} \\ \hline
cl' = cl \oplus \{(buyer?, cl(buyer?) - amount?), (seller?, cl(seller?) + amount)\} \\ \hline
\end{array}$$

The schema $Transaction$ together with $SysState$ ensures "implicitly" that the system state will change if and only if $cl(buyer) \geq amount$.

In our specification, we ignore a lot of important details for an actual system. We ignore the addition of new participants and the mechanism of adding funds to the system. This specification, however, gives us a good starting point for specifying our cryptocurrency system in the next section.

3.2 A Cryptocurrency Specification

In this section we try to capture the most import components of a cryptocurrency system in our formal specification. Every coming subsection will address one or more schemas describing different parts of our abstract version of a cryptocurrency network.

3.2.1 Data types. In this subsection, we formally specify all of the data types used in the remainder of the specification.

The smallest meaningful piece of data in a cryptocurrency is a transaction. As in the abstract digital currency, a cryptocurrency transaction consists of a buyer and a seller which are agents and the amount sent in the transaction. However, a transaction here is a data type. Furthermore, it has a timestamp, which specifies the time the transaction was created. The timestamp is necessary for ensuring the validity of a transaction.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\mathbb{T}} \\
\hline
time : \mathbb{TS} \\
buyer, seller : Agent \\
amount : \mathbb{R}^{\geq 0} \\
\hline
buyer \neq seller \\
\hline
\end{array}
$$

We use timestamp denoted $\mathbb{TS}$ and $Agent$ as basic data types that tell time and identify users in the network respectively. In $\mathbb{T}$ the condition $buyer \neq seller$ is called an invariant and it establishes that the buyer and seller are different agents. The invariant condition must hold for every instance of type $\mathbb{T}$.

A collection of valid transactions forms a block in the network. We define the type $Block$ to be a data type with a header object of type $Header$ and a set of transactions $txs : \mathbb{P}\,\mathbb{T}$.

$$
\begin{array}{|l}
\multicolumn{1}{l}{Block} \\
\hline
header : Header \\
txs : \mathbb{P}\,\mathbb{T} \\
\hline
\end{array}
$$

The header of a block contains several important pieces of information, namely

- A timestamp object $time : \mathbb{TS}$ specifying the time the block was created.
- A hash $hashPrevBlock : \mathbb{B}^{256}$ of the header of the previous block in blockchain.
- A $target : \mathbb{B}^{256}$ which specifies the amount of computational power required to mine the block. When $target$ is converted from binary to hexadecimal it will have a large number of leading zeros (very small number).
- A $nonce : \mathbb{N}$ that is used in the mining process.
- A $height : \mathbb{N}$ which specifies the number of the block in blockchain.

$$
\begin{array}{|l}
\multicolumn{1}{l}{Header} \\
\hline
time : \mathbb{TS} \\
hashPrevBlock : \mathbb{B}^{256} \\
target : \mathbb{B}^{256} \\
nonce : \mathbb{N} \\
height : \mathbb{N} \\
\hline
\end{array}
$$

All of the types that we have defined so far have been very simple. They did not have any complicated condition nor did they have any operations. The upcoming type is different. We define the most important and complex type in our specification as a class using object Z.

As we mentioned in Chapter (2) a chain of blocks forms a blockchain. So, in our formal specification, a blockchain is a collection of valid blocks of type $\mathbb{P}\, Block$ and some other pieces of information. We model blockchain's $blocks$ as a set rather than a sequence to prevent duplication of blocks and also we can always recreate the sequence from the set using height as our ordering attribute.

The other pieces of information that specify a $Blockchain$ object are:

- $length : \mathbb{N}$ is the number of blocks in blockchain. Clearly $\# blocks = length$.
- $reward : \mathbb{R}^{\geq 0}$ is the number of coins the miner is allowed to give himself if he successfully mines a block first.
- $target : \mathbb{B}^{256}$ which is similar to $target$ in the $Block$ schema, however, this target is the target used to mine the latest blocks in blockchain. This information is not actually stored in the real versions of a blockchain but it is obtained by looking at the blocks in it. We have added it to help make our specification neater.

$Blockchain$

$\mathrm{Genesis} : Block$
$maxTarget : \mathbb{B}^{256}$

$\mathrm{Genesis}\,.height = 1$

$bvalid(\mathrm{Genesis})$

$blocks : \mathbb{P}\, Block$
$length : \mathbb{N}$
$reward : \mathbb{R}^{\geq 0}$
$target : \mathbb{B}^{256}$

$\forall\, b : Block \bullet b \in blocks \;\wedge\; bvalid(b)$

$\textsc{Init}$

$blocks = \{\mathrm{Genesis}\}$

$length = 1$

$reward = \text{“max”}$

$target = \text{“starget”}$

There are two other pieces of information which are usually hard-coded into the software managing the network, and so the are added in an open nameless schema at the start of the $Blockchain$ class. The first is $Genesis : Block$, which is the first block in blockchain and it is valid by definition and has $height$ 1. The second is $maxTarget : \mathbb{B}^{256}$ which is the largest value that the target object can take in blockchain, so if $target = maxTarget$ in a block, it will be the “easiest” block to mine.

We require that every block in *blockchain.blocks* to be valid; that is every transaction in it is valid and the hash of its header is less than the target requirement when they are compared in hexadecimal. Then a boolean function *bvalid* can be defined by:

$$
\begin{array}{|l}
\mathit{bvalid} : \mathit{Block} \rightarrow \mathbb{B} \\
\hline
\exists\, \mathit{bc} : \mathit{Blockchain} \bullet \forall\, b : \mathit{Block} \bullet \\
\qquad \mathit{bvalid}(b) \Leftrightarrow \forall\, \mathit{tx} : \mathbb{T} \bullet \\
\qquad\qquad\qquad \mathit{tx} \in b.\mathit{txs} \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad \mathit{bc}.\mathit{tvalid}(\mathit{tx}) \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad \mathit{hash}(b.\mathit{header}) \leq b.\mathit{header}.\mathit{target}
\end{array}
$$

When a cryptocurrency is launched its blockchain will be in its initial state. In the initial state blockchain of a cryptocurrency contains only the first block and the reward and target are at their initial values, “max” and “starget” respectively, usually set by the founder/s.

After formally specifying *Blockchain* we can now define more complex classes with different operations.

3.2.2 Wallets. Wallets are the software used by end users to make transactions and they also show when users receive transactions. A wallet has one owner but usually contains a large number of public addresses. Our wallet specification merges the two concepts under owner which is an instance of type *Agent*. Since our cryptocurrency has a public blockchain the *balance* of each user is visible to every part of the network.

$$
\begin{array}{|l}
\hline
\textit{Wallet} \\
\upharpoonright (\mathit{owner}, \mathit{balance}) \\
\\
\quad \begin{array}{|l}
\hline
\mathit{balance} : \mathbb{R}^{\geq 0} \\
\mathit{owner} : \mathit{Agent} \\
\hline
\end{array} \\
\\
\quad \begin{array}{|l}
\textsc{Init} \\
\hline
\mathit{balance} = 0 \\
\hline
\end{array} \\
\ldots \\
\hline
\end{array}
$$

The owner of a wallet can create a transaction using his wallet. He would need to specify the *seller* and the *amount* he wants to send. The wallet will check the validity of the transaction, using the function *tvalid*, and broadcast it. If the transaction is found to be invalid the wallet ignores it. A real wallet would usually warn the user with a message if that is the case. We include the specification of *tvalid* is in the *FullNode* schema in Subsection (3.2.4).

$$\begin{array}{|l}
\textit{createTx} \\
\hline
\Delta balance \\
time? : \mathbb{TS} \\
seller? : Agent \\
amount? : \mathbb{R}^{>0} \\
tx! : \mathbb{T} \\
\hline
tx!.time = time \\
tx!.buyer = owner \\
tx!.seller = seller? \\
tx!.amount = amount? \\
\exists\, bc : Blockchain \bullet bc.tvalid(tx!) \\
balance' = balance - amount? \\
\hline
\end{array}$$

By design wallets will not allow an invalid transaction to be created, which is reflected in the schema *createTx*. A valid transaction will deplete the *balance* of the *wallet* sending it.

A wallet needs to be able to receive transactions on behalf of its owner. The schema *receiveTx* specifies this operation as part of the *wallet* class.

$$\begin{array}{|l}
\textit{receiveTx} \\
\hline
\Delta balance \\
tx? : \mathbb{T} \\
\hline
tx?.seller = owner \\
\exists\, bc : Blockchain \bullet \exists\, b : Block \bullet \\
\qquad b \in bc.txs \;\land\; tx? \in b.txs \;\land\; confirmed(b) \\
balance' = balance + tx?.amount \\
\hline
\end{array}$$

When the *wallet* receives a $tx : \mathbb{T}$ it will check that this transaction is valid and is part of a confirmed block before reflecting the gain in the owner's balance. To determined if a block in a blockchain is old enough to be confirmed we define two boolean functions *confirmed* and *age* by:

$$\begin{array}{|l}
confirmed : Block \to \mathbb{B} \\
\hline
\exists\, bc : Blockchain \bullet \forall\, b : Block \bullet \\
\qquad confirmed(b) \Leftrightarrow b \in bc.blocks \;\land\; age(b) \;\geq\; 1\ hour
\end{array}$$

$$\begin{array}{|l}
age : Block \to Age \\
time? : \mathbb{TS} \\
\hline
\forall\, b : Block \bullet age(b) = time? - b.time
\end{array}$$

The 1 hour threshold for a block to become confirmed was an "arbitrary" choice made by Satoshi in (Nakamoto, 2008). Currently depending on the type of transaction of interest different parties may require older blocks (1 day old blocks for coinbase transactions) to accept a transaction as payment.

There are different types of wallet software with different functionalities. Nonetheless, the wallet we specified is sufficient to interact with our system. The complete class schema is shown below.

Wallet

$\upharpoonright (owner, balance)$

$$balance : \mathbb{R}^{\geq 0}$$
$$owner : Agent$$

Init

$$balance = 0$$

createTx

$$\Delta balance$$
$$time? : \mathbb{TS}$$
$$seller? : Agent$$
$$amount? : \mathbb{R}^{>0}$$
$$tx! : \mathbb{T}$$

$$tx!.time = time$$
$$tx!.buyer = owner$$
$$tx!.seller = seller$$
$$tx!.amount = amount$$
$$\exists\, bc : Blockchain \bullet bc.tvalid(tx!)$$
$$balance' = balance - amount$$

receiveTx

$$\Delta balance$$
$$tx? : \mathbb{T}$$

$$tx?.seller = owner \, \exists\, bc : Blockchain \bullet \exists\, b : Block \bullet$$
$$\quad b \in bc.txs \;\wedge\; tx? \in b.txs \;\wedge\; confirmed(b)$$
$$balance' = balance + tx?.amount$$

3.2.3 Pools. Every full node in a cryptocurrency stores a pool of all the valid transactions sent to it by other nodes in the same network. Hence, if we model pool as a class $Pool$ then its state can be reflected by a set $pooltxs$ of valid transactions. Moreover, in the initial state $pooltx$ is empty.

Pool

$$pooltxs : \mathbb{P}\,\mathbb{T}$$

$$\exists\, bc : Blockchain \bullet\ \forall\, tx : \mathbb{T} \bullet$$
$$tx \in pooltxs \Leftrightarrow bc.tvalid(tx)$$

INIT

$$pooltxs = \varnothing$$

addTx

$$\Delta pooltxs$$
$$tx? : \mathbb{T}$$

$$pooltxs' = pooltxs \cup \{tx?\}$$

selectTxs

$$\Delta pooltxs$$
$$txs! : \mathbb{P}\,\mathbb{T}$$

$$txs! \subseteq pooltxs$$
$$pooltxs' = pooltxs \setminus txs!$$

The *pool*'s state can be altered by two operations. Firstly, in the case of receiving a valid transaction the transaction will be added to the *pool*'s transactions. Secondly, a subset of the transactions from *pooltxs* can be removed and sent out for a certain purpose (creation of a new block).

3.2.4 Full nodes. Full nodes are the most vital type of nodes in a cryptocurrency. They store complete copies of the cryptocurrency's blockchain and usually (but not always) are miners who mine new blocks and, thus add new coins to the system. Other types of nodes are light nodes and end users. In our specification, we assume that miners and full nodes are the same and focus on them.

A *fullnode* state is given by:

- A *wallet* : *Wallet* which it uses to send and receive coins.
- A *blockchain* : *Blockchain* which must be in sync (most of the time) with blockchains of the other full nodes linked to it.
- A *pool* : *Pool* containing the most recent valid transactions the node received.
- A *block* : *Block* which is the current block being mined.
- A set *linkedNodes* : $\mathbb{P}$ *FullNode* of the full nodes that can communicate directly with the node. It is usually a small subset of all the full nodes in the network.

We require that a node is in its initial state if it is working on a new block with no transactions and a zero nonce.

FullNode

$\upharpoonright (blockchain)$

$wallet : Wallet$
$blockchain : Blockchain$
$pool : Pool$
$block : Block$
$linkedNodes : \mathbb{P}\, FullNode$

Init

$block.txs = \varnothing$

$block.header.nonce = 0$

$\ldots$

Other than keeping a full copy of blockchain, a node also keeps on creating new blocks. Creating a block changes the state of the node from the initial state. It affects the state of *block* and *pool* objects.

createBlock

$\Delta(block, pool)$
$time? : \mathbb{TS}$
$coinbaseTx : \mathbb{T}$

$coinbaseTx.time = time?$

$coinbaseTx.buyer = \text{“none”}$

$coinbaseTx.seller = wallet.owner$

$$\begin{aligned} coinbaseTx.amount = {} & (blockchain.recalcReward \\ & \wedge \\ & block.header.height \bmod 210,000 = 0) \\ & [] \\ & blockchain.reward \end{aligned}$$

$block.txs = pool.selectTxs \cup \{coinbaseTx\} block.header.time = time?$

$block.header.height = blockchain.length + 1$

$$\begin{aligned} block.header.target = {} & (blockchain.recalcTarget \\ & \wedge \\ & block.header.height \bmod 2016 = 0) \\ & [] \\ & blockchain.target \end{aligned}$$

$$\begin{aligned} & \exists\, b : Block \bullet \forall\, b_1 : Block \bullet \\ & \quad b, b_1 \in blockchain \;\wedge\; b \neq b_1 \\ & \quad \wedge \\ & \quad b.header.time > b_1.header.time \\ & \quad \wedge \\ & \quad block.header.hashPrevBlock = hash(b.header) \end{aligned}$$

The node starts by creating the $coinbaseTx : \mathbb{T}$, which has no $buyer$ so we set its value to "none", and adding it to $block.txs$. $coinbaseTx$ rewards the miner/node if he successfully mines the block first. The value of the reward halves every 210,000 blocks and becomes zero after enough time has passed, and $coinbaseTx.amount$ follows it. So, in the future miners will be need to rely on transaction fees as an incentive to keep mining new blocks. We do not incorporate fees in our formal specification. In the specification, we require that $coinbaseTx.amount$ is either given the recalculated value of the reward or the value already stored in $blockchain.reward$. Only one of the two cases can happen each time since the reward value is recalculated if and only if the condition to recalculate it is met (210,000 blocks have mined since the last recalculation).

After creating $coinbaseTx$, the node chooses some transactions from his pool to add to $block.txs$. He also uses the current time and length of blockchain to set the values for $block.header.time$ and $block.header.height$. The two values left to specify are $blockchain.header.target$ and $block.header.haskPrevBlock$. The simpler between the two, $block.header.haskPrevBlock$, is given the hash of the header of a previous block. The other one follows the example of $blockchain$ and takes either blockchain's global value or a new recalculated value if the block number meets a particular condition.

The schema $createBlock$ produces no outputs. However, the changes it made to $block$ is utilized by the schema $mineBlock$.

We know that mining is the process of finding the solution to a hard computing problem. We abstract the process of mining to finding the value of $block.nonce$ such that the $hash(block.header) \leq blockchain.target$.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{mineBlock}} \\
\hline
\Delta block \\
newBlock! : Block \\
\hline
\exists\, n \in \mathbb{N} \bullet \\
\qquad hash(block.header) \leq blockchain.target \;\wedge\; block.header.nonce = n \\
newBlock! = block \\
\hline
\end{array}
$$

If the node is victorious at mining the block, it will quickly broadcast it to the other nodes in the network. Incidentally, while mining the node can receive a $newBlock$ from any other node in the network. Adding $newBlock$ to the node's $blockchain$ will alter the states of the $blockchain$, $block$ and $pool$ instances.

$addBlock$

$$\Delta(blockchain, block, pool)$$
$$newBlock? : Block$$

$$
\begin{aligned}
&newBlock?.header.height \geq blockchain.length + 1\\
&bvalid(newBlock?)\\
&blockchain'.blocks = blockchain.blocks \cup \{newBlock?\}\\
&\qquad\qquad\qquad\qquad [\,]\\
&\qquad\qquad\qquad\qquad ([node \in linkedNodes] \bullet\\
&\qquad\qquad\qquad\qquad node.requestBlock(blockchain.length + 1)\\
&\qquad\qquad\qquad\qquad \gg addBlock)\\
&blockchain'.length = blockchain.length + 1\\
&pool'.pooltxs = pool.pooltxs \setminus newBlock.txs\\
&FullNode.INIT
\end{aligned}
$$

We require that $newblock$ has height greater than the length of blockchain to ensure that it will add to the length of blockchain. Moreover, it must also be a valid block. A valid new block must be added on top of the existing $blockchain$. But, it may happen that $blockchain$ is missing some of the blocks before $newblock$. If $blockchain$ is missing some blocks it will have to request them from the nodes linked to it. When the node has caught up with all of the blocks in his neighbour's blockchains he can add $newblock$ and make the necessary adjustment to the relevant variables. He will also create another new block and start mining again.

When a node requests a block it does not have from another node it will give the required block height and gets the block it requested as $newBlock$ so it can use $addBlock$ to add it to its $blockchain$.

$requestBlock$

$$height? : Block$$
$$newBlock! : Block$$

$$
\begin{aligned}
&\exists\, b : Block \bullet b.header.height = height\\
&newBlock = b
\end{aligned}
$$

We require that the reward is halved every 210,000 blocks.

$recalcReward$

$$\Delta blockchain$$

$$
\begin{aligned}
&blockchain.length + 1 \bmod 210,000 = 0\\
&blockchain.reward' = blockchain.reward/2
\end{aligned}
$$

We also require that the target is recalculated every 2016 blocks.

$$
\begin{array}{|l}
\hline
\textit{recalcTarget} \\
\Delta \textit{blockchain} \\
\textit{exTime}, \textit{acTime} : \mathbb{R}^{>0} \\
\hline
\textit{blockchain.length} + 1 \bmod 2016 = 0 \\
\exists\, b_1, b_2 : \textit{Block} \bullet \\
\quad b_1.\textit{header.height} = \textit{blockchain.length} \\
\quad \wedge \\
\quad b_2.\textit{header.height} = \textit{blockchain.length} - 10 \\
\textit{exTime} = 20160 \\
\textit{acTime} = b_1.\textit{header.time} - b_2.\textit{header.time} \\
\textit{blockchain.target}' = (\textit{exTime}/\textit{acTime}) * (\textit{maxTarget}/\textit{blockchain.target}) \\
\hline
\end{array}
$$

The target's new value depends on the time it took for nodes to mine the last 2016 blocks. If it took less time than 10 minutes on average to mine a block, then the target value diminishes accordingly.

The reason full nodes are important is that they are the only parties in the network that are independently capable of checking the validity of a transaction using their *blockchain*. Then a boolean function *tvalid* can be defined by:

$$
\begin{array}{|l}
\textit{tvalid} : \mathbb{T} \rightarrow \mathbb{B} \\
\hline
\forall\, tx : \mathbb{T} \bullet \\
\quad \textit{buyerBalance} = \textit{amountReceived}(tx.\textit{buyer}, tx.\textit{time}) - \ \textit{amountSent}(tx.\textit{buyer}, tx.\textit{time}) \\
\quad \textit{tvalid}(tx) \Leftrightarrow \textit{buyerBalance} \geq tx.\textit{amount}
\end{array}
$$

So a transaction is valid if its buyer's balance can cover the amount he wishes to send.

To calculate the balance of any agent at a specific point in time we compute the difference between the number of coins he has received and those which he has sent. The schemas *amountSent* and *amountReceived* calculate these numbers by scanning through the entire *blockchain* up to the specified time.

$$
\begin{array}{|l}
\hline
\textit{amountSent} \\
\textit{agent}? : \textit{Agent} \\
\textit{time}? : \mathbb{TS} \\
\textit{amountS}! : \mathbb{R}^{\geq 0} \\
\textit{sTxs} : \mathbb{P}\, \mathbb{T} \\
\hline
\forall\, b : \textit{Block} \bullet b \in \textit{blockchain.blocks} \ \wedge \ b.\textit{header.time} \leq \textit{time}? \\
\quad \forall\, tx : \mathbb{T} \bullet \\
\qquad tx \in \textit{sTxs} \Leftrightarrow tx \in b.\textit{txs} \ \wedge \ tx.\textit{buyer} = \textit{agent}? \\
\textit{amoutS}! = \sum_{tx \in \textit{sTxs}} tx.\textit{amount} \\
\hline
\end{array}
$$

amountReceived

$$\begin{array}{l}
agent? : Agent \\
time? : \mathbb{TS} \\
amountR! : \mathbb{R}^{\geq 0} \\
rTxs : \mathbb{P}\,\mathbb{T} \\
\hline
\forall\, b : Block \bullet b \in blockchain.blocks \;\wedge\; b.header.time \leq time? \\
\quad \forall\, tx : \mathbb{T} \bullet \\
\qquad tx \in rTxs \Leftrightarrow tx \in b.txs \;\wedge\; tx.seller = agent? \\
amountR! = \sum_{tx \in rTxs} tx.amount
\end{array}$$

The *FullNode* class is the largest class in our specification, and we connect all of its components in one schema below.

FullNode

$$\upharpoonright (blockchain)$$

$$\begin{array}{l}
wallet : Wallet \\
blockchain : Blockchain \\
pool : Pool \\
block : Block \\
linkedNodes : \mathbb{P}\, FullNode
\end{array}$$

Init

$$\begin{array}{l}
block.txs = \varnothing \\
block.header.nonce = 0
\end{array}$$

addBlock

$$\begin{array}{l}
\Delta(blockchain, block, pool) \\
newBlock? : Block \\
\hline
newBlock?.header.height \geq blockchain.length + 1 \\
bvalid(newBlock?) \\
blockchain'.blocks = blockchain.blocks \cup \{newBlock?\} \\
\qquad\qquad\qquad\qquad [] \\
\qquad\qquad\qquad\qquad ([node \in linkedNodes] \bullet \\
\qquad\qquad\qquad\qquad node.requestBlock(blockchain.length + 1) \\
\qquad\qquad\qquad\qquad \gg addBlock) \\
blockchain'.length = blockchain.length + 1 \\
pool'.pooltxs = pool.pooltxs \setminus newBlock.txs \\
FullNode.INIT
\end{array}$$

requestBlock

$$\begin{array}{l}
height? : Block \\
newBlock! : Block \\
\hline
\exists\, b : Block \bullet b.header.height = height \\
newBlock = b
\end{array}$$

createBlock

$\Delta(block, pool)$
$time? : \mathbb{TS}$
$coinbaseTx : \mathbb{T}$

$coinbaseTx.time = time?$

$coinbaseTx.buyer = none$

$coinbaseTx.seller = wallet.owner$

$$\begin{aligned} coinbaseTx.amount = {} & (blockchain.recalcReward \\ & \wedge \\ & block.header.height \bmod 210{,}000 = 0) \\ & [] \\ & blockchain.reward \end{aligned}$$

$block.txs = pool.selectTxs \cup \{coinbaseTx\} block.header.time = time?$

$block.header.height = blockchain.length + 1$

$$\begin{aligned} block.header.target = {} & (blockchain.recalcTarget \\ & \wedge \\ & block.header.height \bmod 2016 = 0) \\ & [] \\ & blockchain.target \end{aligned}$$

$$\begin{aligned} & \exists\, b : Block \bullet \forall\, b_1 : Block \bullet \\ & \quad b, b_1 \in blockchain \;\wedge\; b \neq b_1 \\ & \quad \wedge \\ & \quad b.header.time > b_1.header.time \\ & \quad \wedge \\ & \quad block.header.hashPrevBlock = hash(b.header) \end{aligned}$$

mineBlock

$\Delta block$
$newblock! : Block$

$$\begin{aligned} & \exists\, n \in \mathbb{N} \bullet \\ & \quad hash(block.header) \leq blockchain.target \;\wedge\; block.header.nonce = n \end{aligned}$$

$newblock! = block$

recalcReward

$\Delta blockchain$

$blockchain.length + 1 \bmod 210{,}000 = 0$

$blockchain.reward' = blockchain.reward/2$

recalcTarget

$\Delta blockchain$
$exTime, acTime : \mathbb{R}^{>0}$

$blockchain.length + 1 \bmod 2016 = 0$

$$\begin{aligned} & \exists\, b_1, b_2 : Block \bullet \\ & \quad b_1.header.height = blockchain.length \\ & \quad \wedge \\ & \quad b_2.header.height = blockchain.length - 10 \end{aligned}$$

$exTime = 20160$

$acTime = b_1.header.time - b_2.header.time$

$blockchain.target' = (exTime/acTime) * (maxTarget/blockchain.target)$

$tvalid : \mathbb{T} \rightarrow \mathbb{B}$

$\forall\, tx : \mathbb{T} \bullet$
$\quad buyerBalance = amountReceived(tx.buyer, tx.time) - \ amountSent(tx.buyer, tx.time)$
$\quad tvalid(tx) \Leftrightarrow buyerBalance \geq tx.amount$

amountSent

$agent? : Agent$
$time? : \mathbb{TS}$
$amountS! : \mathbb{R}^{\geq 0}$
$sTxs : \mathbb{P}\,\mathbb{T}$

$\forall\, b : Block \bullet b \in blockchain.blocks \ \wedge \ b.header.time \leq time?$
$\quad \forall\, tx : \mathbb{T} \bullet$
$\qquad tx \in sTxs \Leftrightarrow tx \in b.txs \ \wedge \ tx.buyer = agent?$
$amoutS! = \sum_{tx \in sTxs} tx.amount$

amountReceived

$agent? : Agent$
$time? : \mathbb{TS}$
$amountR! : \mathbb{R}^{\geq 0}$
$rTxs : \mathbb{P}\,\mathbb{T}$

$\forall\, b : Block \bullet b \in blockchain.blocks \ \wedge \ b.header.time \leq time?$
$\quad \forall\, tx : \mathbb{T} \bullet$
$\qquad tx \in rTxs \Leftrightarrow tx \in b.txs \ \wedge \ tx.seller = agent?$
$amountR! = \sum_{tx \in rTxs} tx.amount$

3.2.5 Network. The last class we formalize is an abstract description of part of the interaction between the nodes. There is no actual entity representing this class. It is however, the way observers see the *network*. A cryptocurrency is described by its participants list $nodes : \mathbb{P}\, FullNode$ and $blockchain : Blockchain$. We require that *blockchain* is the longest *blockchain* between all the *nodes*' blockchains. We enforce this condition in the invariant in the schema's state. In our specification, we ignore the threat of a malevolent node creating a long chain of low-difficulty blocks, and consider the longest chain to be the chain with the greatest number of valid blocks. We also assumed that every blockchain is valid, since by definition it contains only valid blocks. A *newNode* can join the *network* by simply copying the *blockchain*.

Network

$nodes : \mathbb{P}\, FullNode$
$blockchain : Blockchain$

$\forall\, node : nodes \bullet$
$\quad blockchain = node.blockchain$
$\quad \Leftrightarrow$
$\quad \{o : nodes \setminus \{node\} \mid node.blockchain.length < o.blockchain.length\} = \phi$

INIT

$nodes = \phi$
$blockchain.INIT$

join

$\Delta nodes$
$newNode? : FullNode$

$newNode.requestBlock(blockchain.length)$
$nodes' = nodes \cup \{newNode\})$

addBlock

$\Delta blockchain$

$newBlock = []node : nodes \bullet node.mineBlock$
$\forall\, node : nodes \bullet node.addBlock(newBlock)$

$forked : Network \rightarrow \mathbb{B}$

$\forall\, network : Network \bullet forked(network) \Leftrightarrow$
$\quad \forall\, node : nodes \bullet blockchain.length = node.blockchain.length$
$\quad \wedge$
$\quad \{s : nodes \setminus \{node\} \mid s.blockchain \neq node.blockchain\} \neq \phi$

The process of adding a new block to the network happens when a node mines a block successfully and the other nodes accept this block. We do not specify a line to modify the network's *blockchain* in the schema *addBlock* since it is modified automatically by the class invariant. The invariant ensures as well that *Network.blockchain.reward* and *Network.blockchain.target* and recalculated whenever necessary. One interesting boolean function we choose to define is *forked*, it will help us recognized when the network's blockchain becomes forked.

In this chapter we gave a formal specification for an abstract cryptocurrency network. A cryptocurrency is defined by its nodes which each store their own blockchain and use wallets to send and receive transactions. In the next chapter we conclude and give some ideas for future work.

4. Conclusion

In this project, we have given a formal specification of blockchain technology. We started by investigating the underlying concepts behind blockchain and the primary components of cryptocurrencies. Furthermore, we gave our formal specification for an abstract cryptocurrency.

Specifying blockchain formally helped us in deepening our understanding of blockchain and the way cryptocurrencies work. It also made us pay close attention to some details we brushed over when looking at general explanations. Specifying a concept that has a large number of materials available on it was a daunting task. However, it was satisfying to look at how simple mathematical concepts can be used to describe a concept with a vast number of actual applications. Using Z notation to specify blockchain was a two edged sword. It helped in ensuring that the specification was accurate and correctly reflects the system, but at the same time a minor error can render the entire work invalid.

For future work we would like to prove mathematically that our specification correctly models blockchain. Since, in our essay our specification has targeted only finical application of blockchain, one could also make the specification more abstract and general to include other types non-financial applications.

References

Antonopoulos, A. M. *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*, pages xix–xxi. O'Reilly Media, Inc., 2014. URL https://github.com/bitcoinbook/bitcoinbook/blob/develop/glossary.asciidoc.

BitcoinWiki. Block, Accessed on May 2019a. URL https://en.bitcoin.it/wiki/Block.

BitcoinWiki. Proof of Work, Accessed on May 2019b. URL https://en.bitcoin.it/wiki/proof-of-work.

Crosby, M., Pattanayak, P., Verma, S., and Kalyanaraman, V. Blockchain Technology: Beyond Bitcoin. *Applied Innovation*, 2:6–10, 2016.

DeMartino, I. The Many Types and Functions of Bitcoin Wallets, Accessed on May 2019. URL https://cointelegraph.com/news/the-many-types-and-functions-of-Bitcoin-wallets.

DeMichele, T. What Is a Cryptocurrency Wallet?, Accessed on May 2019. URL http://cryptocurrencyfacts.com/what-is-a-cryptocurrency-wallet/.

Hameed, S. and Farooq, S. The Art of Crypto Currencies A Comprehensive Analysis of Popular Cryptocurrencies. *International Journal of Advanced Computer Science and Applications*, 7(12):426–435, 2016. URL https://arxiv.org/pdf/1711.11073.pdf.

Kehrli, J. Blockchain Explained, Accessed on May 2019a. URL https://www.niceideas.ch/roller2/badtrash/entry/blockchain-explained-beta.

Kehrli, J. Blockchain 2.0 - From Bitcoin Transactions to Smart Contract Applications, Accessed on May 2019b. URL https://www.niceideas.ch/roller2/badtrash/entry/blockchain-2-0-from-bitcoin.

Malviya, H. How Blockchain will Defend IOT. *SSRN*, 2016. URL https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2883711.

MetroGnomo.com. MetroGnomo – FAQs, Accessed on May 2019. URL https://www.metrognomo.com/faq/.

Micah Alcorn. Introducing Origin Messaging: Decentralized, Secure, and Auditable, Accessed on May 2019. URL https://medium.com/originprotocol/introducing-origin-messaging-decentralized-secure-and-auditable-13c16fe0f13e.

Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. URL https://bitcoin.,org/bitcoin.pdf.

Reid, F. and Harrigan, M. An Analysis of Anonymity in the Bitcoin System. In *Security and privacy in social networks*. URL https://arxiv.org/pdf/1107.4524.pdf%3Forigin%3Dpublication_detail.

Reply(company). Ballotchain Online Voting System Powered by Blockchain, Accessed on May 2019. URL http://www.reply.com/en/content/ballotchain.

Rosic, A. What Is Blockchain Technology? A Step-by-Step Guide for Beginners, Accessed on May 2019. URL https://blockgeeks.com/guides/what-is-blockchain-technology/.

Saberhagen, N. v. CryptoNote v2.0. *CryptoNote Whitepaper*, 2013. URL https://cryptonote.org/whitepaper.pdf.

Seibold, S. and Samman, G. Consensus: Immutable Agreement for the Internet of Value. *KPMG Whitpaper*, 2016. URL https://assets.kpmg/content/dam/kpmg/pdf/2016/06/kpmg-blockchain-consensus-mechanism.pdf.

Snow, P., Deery, B., Lu, J., Johnston, D., and Kirby, P. Factom Business Processes Secured by Immutable Audit Trails on the Blockchain. *Factom Whitepaper*, November 2014.

Swan, M. *Blockchain: Blueprint for a New Economy*. O'Reilly Media, Inc., 2015.

Tapscott, D. and Tapscott, A. *Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World*. Portfolio-Penguin, 2016.

Wikipedia. Byzantine Fault, Accessed on May 2019. URL https://en.wikipedia.org/wiki/Byzantine_fault.

Wilkinson, S., Boshevski, T., Brandoff, J., and Buterin, V. Storj a Peer-to-Peer Cloud Storage Network. *Storj Whitepaper*, 2014. URL https://storj.io/storj2014.pdf.

Z/Yen Group. Smart Ledger (aka Blockchain) Technology, Accessed on May 2019. URL https://www.zyen.com/work/types-work/mutual-distributed-ledger-aka-blockchain-technology.