Blockchain State Machine Representation

Jamsheed Shorish*
jamsheed@shorishresearch.com

Version 1.0.1
January 19, 2018

Abstract

We present a formalization of blockchain as a state machine, focusing upon permissionless blockchains due to general audience awareness of its most popular implementation, Bitcoin (permissioned blockchains are treated similarly without loss of generality). After presenting a typical Bitcoin transaction workflow, a general blockchain state representation is derived. It is demonstrated that the proper mathematical object defining the state of a blockchain is a topological fiber bundle, because it is not possible to globally 'parametrize' blocks (or ledgers of blocks) by time due to their dependence upon cryptographic hash functions. In addition, we specify a general transition function between blockchain states that is agnostic to the consensus mechanism used to write blocks into the ledger, and which is probabilistic in nature, so that blockchain may be regarded as a probabilistic state machine. We then interpret agents (both human and code-based, such as 'chaincode', 'smart contracts', or other artificial intelligence) as automata interacting with blockchain technology, drawing upon the theory of non-cooperative repeated interaction games. Finally, blockchain as a hierarchy of state machines is defined, and future research directions are presented using this hierarchy as a point of departure for modeling blockchain dynamics.

JEL Classification Codes: C63, C70

*Shorish Research, Belgium. *Shorish Research provides strategic consulting services built upon academic and real-world application expertise in the area of Computational Business.*

1 Introduction

Blockchain technology is poised to become, if its proponents prove to be correct, *the* major database innovation since the development of the Internet and the World Wide Web (see e.g. [BS18; Böh+15; Mar16; Lac16] for general examples of blockchain's penetration in the economics literature and popular press media). As a decentralized ledger technology (DLT), blockchain provides transparency, immutability and (in principle) decentralized governance. Such characteristics are currently being leveraged by consortia of large enterprises,[1] the open-source community[2] and government and educational institutions[3] at the proof-of-concept and production stages. Blockchain is also being studied from the socio-economic perspective, with applications to development economics ([All17]), innovation ([CG16]), and governance ([DFP16]) among others. According to an analysis by Deloitte Insights[4] there are currently over 86,000 blockchain development projects underway using the online collaboration platform Github, with nearly 27,000 projects being added in 2016 alone.

Beyond the hype, blockchain is a database–perhaps built upon (or out of) a 'world computer', such as the Turing-complete Ethereum Virtual Machine (EVM),[5] or more commonly as a decentralized network of participating 'nodes' that each possess a copy of the database and may (depending upon the consensus mechanism) participate in how the database is updated. Moreover, blockchain has a rich storage environment, capable of storing not just data records but also programming code, allowing for execution of such code if built upon a computing system like the EVM. These storage and execution capabilities allow blockchain to be defined heuristically as a finite automata or finite state machine,[6] but properly formalizing the state machine interpretation is challenging and hence most of the extant discussion of blockchain rest upon informal or anecdotal evidence of blockchain's state machine representation.

This is not enough to allow for a full study of blockchain's structure, implementation challenges and (of particular interest to parties using blockchain for business logic) strategic implications, where by *strategic* we mean the consideration by blockchain participants of the actions of other participants. This is particularly germane to blockchain technology, where participants are not just users themselves, but also automated representations that are coded and executed as part of the blockchain (e.g. 'smart contracts', 'chaincode', artificial intelligence entities, etc.). Strategic implications are crucial to formulate and build the types of contracting relationships that can exist on the blockchain, such as the aforementioned applications to land registry (cf. footnote 3) and to use cases such as identity registration and management, health record access, etc.

In this work we present a formalization of blockchain as a probabilistic state machine (PSM).[7] We focus upon *permissionless* or 'open' blockchains, due to general audience

[1] See e.g. R3/Corda, Enterprise Ethereum Alliance, LiquidShare.

[2] E.g. Hyperledger.

[3] In Belgium, "Toelichting proof of concepts in Vlaanderen[Illustrating proof of concepts in Flanders]", retrieved November 2017; in the U.K.,"HM Land Registry signals the start of its transformation", press release 18 July 2017; in Sweden, "Blockchain and Future House Purchases", retrieved November 2017.

[4] [TFS17]

[5] See Ethereum.org.

[6] See e.g. [Sip13] for an introduction and overview of finite automata.

[7] See e.g. [SY16] for a similar identification–there the authors concentrate on network participation uncertainty.

awareness of its most popular implementation, Bitcoin. But the analysis carries through with only minor adjustment for *permissioned*, or 'closed' blockchains, where a gatekeeper or other entity (or entities) whitelist participants in the network for particular activities (or sets of activities).[8] We show that the proper mathematical object that defines the state of a blockchain is a topological fiber bundle, because it is not in general possible to decompose a blockchain's state into a local product space of time and blockchain-specific data (such as blocks or ledger lists of blocks). This impossibility stems from blockchain's fundamental use of cryptographic hash functions, and is compounded by the fact that what is germane for decision-making is an equivalence relation over blocks, rather than individual blocks themselves. A general transition function between blockchain states is also derived that is 1) agnostic to the consensus mechanism used to write blocks into the ledger, and 2) probabilistic in nature, so that blockchain may be regarded as a probabilistic state machine.

Section 2 presents a quick introduction to blockchain's workflow, focusing upon the Bitcoin network as the leading (and origin) implementation of blockchain technology. In Section 3 the formalization of blockchain as a PSM is presented: Section 3.1 introduces definitions, while 3.2 derives the blockchain state representation. Sections 3.3 and 3.4 discuss agent actions and state transitions, respectively, and Sec. 3.5 puts everything together to define the PSM representation of a general blockchain. Section 4 then discusses how agents can themselves be considered as state machines, drawing upon an automata representation originally put forward for repeated game interaction in non-cooperative game theory. Finally, Section 5 concludes with an overall summary of the machine hierarchy from agent to blockchain, and proposes directions for future research.

2 Blockchain's Workflow: A Short Bitcoin Introduction

A blockchain as a DLT allows for transactions to be recorded which are transparent and, in principle, advantageous in that they do not require a third party to validate or verify the transactions process. Blockchain was introduced as the underlying technology for Bitcoin in Satoshi Nakamoto's seminal white paper in 2008/2009 ([Nak09]), and has subsequently seen an explosive growth of interest both from the public at large and from enterprise consortia. In seeking to formally describe how blockchain operates it is worth first quickly describing a typical blockchain workflow through the Bitcoin network (as of December 2017).

A Bitcoin network is a set of *nodes* $\mathcal{N}$, representing participants in the storage and manipulation of the underlying blockchain database. Time is continuous and is indexed by $t \in \mathbb{R}_+$, although blockchain updates are normally performed at discrete intervals. Each period a (usually strict) subset of nodes $N_t \subset \mathcal{N}$ is drawn and represents the active nodes in the network–the number of nodes $|N_t|$ active may thus change over time.[9]

[8] Permissioned blockchains are often referred to as 'private' blockchains–but as this terminology is also used to describe permission-only centralized blockchains, we shall use 'permissioned' to describe gatekeeper whitelisted, but decentralized, blockchain implementations.

[9] There is some evidence that nodes follow a preferential attachment network growth model, see e.g. [Kon+14], but there is apparently little extant research on the statistical distribution of participating nodes in the Bitcoin network. Note also that for a closed, permissioned blockchain network N_t may be

Each node possesses a full copy of the blockchain *ledger* L_t. The ledger is a record of every confirmed transaction in the network, and is (by definition in the Bitcoin network) the longest blockchain. A *block* B_t is a formatted list of transactions, and may be either *confirmed* or *unconfirmed*.[10] The ledger is a linked list of confirmed blocks, such that each block B_t references its immediate predecessor B_{t-1}, where the subscript indicates the time index of block confirmation and addition to L_t, which is discrete (an exception is B_0, the 'genesis' block, which has no predecessor). Thus we can schematically refer to a blockchain ledger as:

$$L_t := B_0 \cup B_1 \cup \ldots \cup B_t, \tag{1}$$

where the 'union' symbol $\cup$ is meant to facilitate the implication that the ledger L_t is a union of all transactions contained in all confirmed blocks.

A *transaction* T is a transfer of value between two users. A user is defined by an *address*, which is a unique identifier (Bitcoin is thus said to be a 'pseudonymous' network rather than anonymous, as an address uniquely identifies a user–note, however, that this mapping is not one-to-one, as a user may have multiple addresses). A block has a finite capacity for recording transactions–Bitcoin's capacity was originally 1MB, which allowed for (roughly) 2,000 transactions to be recorded per block.[11] Understanding how Bitcoin operates involves understanding how transactions are performed, and how they are collected into their respective block(s).

2.1 Bitcoin Transactions

When a user i decides to perform a bitcoin transaction T_t^i, they *broadcast* their transfer of value to the active network, N_t. This transaction is assumed for simplicity to occur at t and contains:

1. a list of *inputs*, which indicates all of the unspent bitcoin coming in to the address to be debited–each input is an output of a previous transaction contained in a confirmed block,

2. a list of *outputs*, which distributes the sum total of bitcoin from the inputs to one or more addresses, minus any specified transaction fee that goes to the block miner and is used to preferentially select transactions for block inclusion (see below).

In principle, every node $j \in N_t$ will receive T_t^i and add it to an unconfirmed block that is not already full–we can denote node j's unconfirmed block at a time $t + (\Delta t)_j$, when user i's transaction T_t^i arrives to j between time t and time $t + (\Delta t)_j$, by $B^j_{t+(\Delta t)_j}$. In practice, due to Internet network delays different nodes receive different unconfirmed transactions. Nodes also reorder transactions according to their proposed transaction fee, so that transactions with higher fees are placed higher in the block (and are thus

fixed and known, depending upon the whitelisting protocol implemented.

[10] The blockchain literature often uses the terminology *valid* and *invalid*–but because transactions within an 'invalid' block may be re-submitted and eventually become part of a 'valid block', confirmation would appear to be a more apt term than validation.

[11] An upgrade in September 2017 reduced transactions sizes by removing hash signature information from the transaction in the block, allowing for around 4,000 transactions to be recorded as of December 2017.

confirmed–and executed–earlier). It is this variation in transaction arrival time and reordering that allows for each potential new block in the blockchain to be different for each node.

2.2 Bitcoin Blocks

A block $B^j_{t+(\Delta t)_j}$ created by node j contains the following information:

1. a reference (hash) to the latest confirmed block in the ledger, $B_t \in L_t$,
2. a reference ('merkle root hash') based upon the block's transactions,
3. a unix timestamp, i.e. the number of seconds since 1 January, 1970 at midnight UTC,
4. a *target* for the blockchain mining problem (see sec. 2.3 below),
5. a *nonce* for the blockchain mining problem (see sec. 2.3 below), and
6. a list of transactions that have arrived between t and $t + (\Delta t)_j$ to node j, that have been potentially reordered from highest transaction fee to lowest.

The validity of the individual transactions, and of the block itself, depends upon cryptographically-signed transactions (this is where addresses come from) and cryptographic hashes of both the latest confirmed block (this validates transactions from currently accepted blockchain L_t) and the transactions contained in the block (this 'merkle root hash' uniquely references the transactions in the block so that they cannot be changed by the miner, or be corrupted without detection during network broadcast). This dependence on cryptography underlies the power of blockchain as a DLT–it allows for transactions, and the blocks that contain them, to be considered *tamper-proof* and hence immutable once recorded.

2.3 Bitcoin Mining and Consensus

When an unconfirmed block $B^j_{t+(\Delta t)_j}$ reaches capacity at time $t+(\Delta t)_j$, node j (or, perhaps more realistically, a collection or "pool" of nodes $\mathcal{N}_j \subset \mathcal{N}$) will attempt to be the first node (or pool) to *confirm* the block. Confirmation takes place by guessing the solution to a mathematical puzzle that depends upon the block target and block nonce that are encoded in the unconfirmed block. Every node that has a full unconfirmed block races against the other to be the first to guess the solution and hence "confirm" the block they are working on. The difficulty of the puzzle, which is again based upon cryptography, is dynamically adjusted by the network to generate an expected time to solution of 10 minutes–hence, on average, at least one block is confirmed every 10 minutes.

When a node j confirms a block at time $\hat{t} > t$, it immediately broadcasts the block to the network $N_{\hat{t}}$, and receiving nodes append that block to the end of their representation of the blockchain, i.e. $L_{\hat{t}} = L_t \cup B_{\hat{t}}$ where $B_{\hat{t}} := B^j_{t+(\Delta t)_j}$. If every node in $\mathcal{N}_{\hat{t}}$ were to receive this broadcast at the same time, then the ledger would be updated to a consistent state for everyone and new creation of blocks from new transactions would continue as above.

However, it may be that multiple nodes confirm their block within a few seconds of each other, and due to network latency different subsets of nodes receive a different confirmed block first. In that case, there are multiple candidate blockchains for $L_{\hat{t}}$ and each node uses whichever local copy they have generated from the received block. To resolve this, recall that every block contains a reference to the previously confirmed block in the blockchain. If a node is working on a local chain that is not what the majority of the network is working on, then (sooner or later) they will start receiving confirmed blocks *with references to blocks they do not have* on their local blockchain copy. They will then abandon their local copy of the blockchain and download an updated copy from their peers, and commence working from the new copy. This ensures that on average, it is the *longest* blockchain that is the valid version of L_t–this is known as Bitcoin network *consensus*.

What happens to the transactions that were contained in a confirmed block that is not part of the longest chain? They are part of an 'orphan block' and are re-submitted to the list of pending transactions in the network, to be added to a new (unconfirmed) block. Thus, it is possible for transactions to be 'tied up' for some time in the Bitcoin network, as the throughput of Bitcoin is small (roughly 3 - 7 transactions per second).

3 Blockchain State Formalization and Representation

From the previous Section we see that at the time a blockchain transaction is created, there are both known factors, such as the state of the longest chain L_t, and unknown factors, such as which block will be the next to be confirmed (validated) and added to the chain. This probabilistic determination of which block comes next has been presented within the motivating example of Bitcoin, but almost all consensus mechanisms discussed in the literature (such as other 'flavors' of Proof-of-Work, Proof-of-Stake, (Practical) Byzantine Fault Tolerance, Proof-of-Importance, etc.) are stochastic either in the 'who wins the mining game' sense (common in open blockchains), or in the network robustness and fidelity sense (examining fault tolerance in closed blockchains). This combination of known state and unknown transition generates a representation of a blockchain workflow as a probabilistic state machine, and we shall formalize this in what follows. Accomplishing this will first require a *state space*, an *action* or *outcome* function, and finally a *transition function*. Discussing each in turn, the following sections outline the representation of the state space as a topological fiber bundle and discuss its consequences–perhaps surprisingly, it is not possible to globally 'tag' a block in a blockchain uniquely by time, as the use of cryptography prevents a global 'orientation' from being defined.

3.1 State Definitions

We loosely define the **state** of the blockchain to be *whatever data are required to be able to make a decision that influences the blockchain.* At this stage it doesn't matter if the decision-making entity is a human user, or a piece of code executed on the blockchain in e.g. a smart contract, chaincode, or an artificial intelligence. Blockchain information is conveniently contained in confirmed blocks, and so it stands to reason that a blockchain

state will be dependent upon how blocks are represented.[12]. This can be accomplished by following the same steps as in the Bitcoin workflow summarized in Section 2 and carefully following the aggregation of state information from transaction to block to blockchain ledger.

We formalize the state of a blockchain beginning with the fundamental subunit, the transaction. Define the *address space* $\mathcal{D}$ to be the discrete set of all possible transfer addresses–in the case of Bitcoin this is a set with $|\mathcal{D}| \simeq 2^{160}$. Values that can be transferred lie in $\mathbb{R}_{++}$, the strictly positive real orthant (we assume that values must be strictly positive to provide a minimum transfer, including a transaction fee if required).

A unique *input* address a possesses an available *balance* b_a, that represents the funds available for transfer–in the case of Bitcoin this would be the sum of all 'unspent transaction outputs' (UTXO) that are used as the input to the transaction, while in the case of other blockchains, such as Ethereum, it is a balance carried as part of the blockchain global store. A subset $D_o \subset \mathcal{D}$ is a list of output addresses of the transaction (which may be singular, if a single output address is the goal of the transfer and there are no transaction fees). Finally, a vector $v_o \in \mathbb{R}_{++}^{|D_o|}$ lists the amounts to be transferred to each output address in D_o. This is sufficient to define a transaction:

Definition 1. *A transaction T_t^i at time t for user i is a vector*

$$T_t^i := (a, b_a, D_o, v_o),$$

such that

$$b_a \geq v_A \cdot 1_{|A|} \tag{2}$$

and where

$$a \in \mathcal{D},\ b_a \in \mathbb{R}_{++},\ D_o \subset \mathcal{D},\ v_o \in \mathbb{R}_{++}^{|D_o|}.$$

Condition (2) is simply a consistency condition that the total available balance is at least as large as the planned transfers, including any transaction fees.

Note that the above characterization means that

$$T_t^i \in \mathbb{R}_{++}^{|D_o|+1} \times \mathcal{D}^{|D_o|+1} \supset \mathcal{T},$$

where we define the space of all possible transactions as $\mathcal{T}$,

$$\mathcal{T} := \mathbb{R}_{++}^{M} \times \mathcal{D}^{M_{\mathcal{T}}},$$

and where $M_{\mathcal{T}} < \infty$ is a large but finite number indicating the maximum number of addresses a transaction can specify (note it is understood that since $|D_o| \leq M_{\mathcal{T}}$ we treat $v_o \in \mathbb{R}^{|D_o|}$ as an element of $\mathbb{R}^M$ via the inclusion map $\iota_k : \mathbb{R}^k \to \mathbb{R}_{\mathcal{T}}^M$ with $k = |D_o|$).

Next, transactions are aggregated into blocks (initially unconfirmed, and then confirmed via the mining or other consensus mechanism–note that we suppress the dependence of the block on the block creation node in what follows for notational simplicity). A block depends upon, first, the timestamp $t \in T$ indicating when it has been created–for

[12]It is also true that any external-to-the-blockchain ("off-blockchain") information should be factored in, if a decision requires its value–in what follows we shall assume that such information is placed within the data a confirmed block possesses, perhaps through the interaction of the blockchain with a data 'oracle'.

simplicity, we assume that this is the time when all unconfirmed transactions have been inserted into the block.[13] A vector $T^{n_B} \in \mathcal{T}^{n_B}$ is a list of these n_B transactions, which is assumed to be finite. Finally, a *hash function* H takes as input the timestamp of the latest confirmed block in the longest blockchain $L_{\hat{t}}$ and the list of transactions in its block, T^{n_B}, and returns a k-dimensional vector of cryptographic hashes from a *hash space* $\mathcal{H}$:

Definition 2. *A block B_t at time t is a vector:*

$$B_t := (t, H(\hat{t}, T^{n_B}), T^{n_B}),$$

where

$$t \in T,\ T^{n_B} \subset \mathcal{T}^{n_B},\ H : T \times \mathcal{T}^{n_B} \to \mathcal{H}^k,\ n_B, k < \infty,$$

and $\mathcal{H}$ is the space of hashes.

The hash function H is, in some sense, the major innovation that sets blockchain apart from other DLT implementations. In the Bitcoin workflow described above, for example, H would return

1. a hash of the header of the latest confirmed block in the blockchain $L_{\hat{t}}$, where $\hat{t}$ is the timestamp of the last confirmed block,
2. a hash of the merkle root, which prevents accidental or malicious rewriting of the transactions in T^{n_B},
3. a *target* hash, which is the target the Proof-of-Work Bitcoin miner uses to find the solution of the hash puzzle, and
4. a *nonce* hash, which is a random value to increment successive guesses during the mining process.

(In the Bitcoin workflow each hash lives in $\mathcal{H} := \mathbb{Z}_{2^{256}}$, i.e. the set of integers from 0 to $2^{256} - 1$, and $k = 4$.)

As with transactions, we may view a block B_t as an element of a larger set:

$$B_t \in \mathbb{R}^2_{++} \times \mathcal{H}^k \times \mathcal{T} \supset \mathcal{B},$$

with

$$\mathcal{B} := \mathbb{R}^2_{++} \times \mathcal{H}^{M_\mathcal{H}} \times \mathcal{T},$$

where we treat the hash vector as being bounded by a large, but finite value $M_\mathcal{H} < \infty$ representing the finite capacity of a block, such that a hash vector of length k may be considered as an element of $\mathcal{H}^{M_\mathcal{H}}$ via the inclusion map $\iota_k : \mathcal{H}^k \to \mathcal{H}^{M_\mathcal{H}}$.

[13] In reality there is a great deal of variation in the accepted definition of a block timestamp for blockchain networks. In Bitcoin, for example,

> [a] timestamp is accepted as valid if it is greater than the median timestamp of previous 11 blocks, and less than the network-adjusted time + 2 hours. "Network-adjusted time" is the median of the timestamps returned by all nodes...As a result, block timestamps are not exactly accurate, and they do not even need to be in order. Block times are accurate only to within an hour or two. [con18]

3.2 State Representation

The foregoing section would appear to imply that since a blockchain ledger contains (by assumption) all information available to make a decision, it should be considered as 'the' state of the blockchain. Thus, we should be able to think of L_t at time t, say, as blockchain's state parametrized by t. Unfortunately this straightforward characterization is not generally true. First, and as will be discussed in further detail below, it is not the case that a state may be uniquely represented by a particular L_t–different blockchain ledgers may represent the same state. This is important because decision-making must be consistent, in the sense that the same state should lead to the same decisions (all other things equal)–having multiple representations of the same state means that the ledger L_t cannot be considered 'the' state but should rather be interpreted as one of many equivalent representations.

Second, the unique method by which blockchain secures itself from tampering, i.e. cryptography, prevents any state from being globally parametrized by t. In other words, if S is the state space then it cannot be represented *globally* by a product space of $T \times X$, where X is some object (e.g. a block B or a ledger L). We investigate this failure in detail in what follows.

In order to be as clear as possible, we first assume that the information contained in a block B_t confirmed at time t is sufficient for decision-making–other blocks in the past do not matter for today's decision. We relax this assumption later and show that whether we consider B_t or L_t as containing all available information does not change the topological structure of the state.

Under this assumption, we reiterate that, at least initially, we would expect the **state space** of a blockchain S to be defined globally, i.e. for all time and over all possible blocks, as a product space of time T and the block space $\mathcal{B}$, i.e.

$$S := T \times \mathcal{B},$$

so that we could, if we wished, simply refer to the state as the value of a block at a given time, i.e. "$S_t := B_t$".

To see that this is not so, recall that part of the specification of $\mathcal{B}$ relies upon cryptographic hashes of (for transactions) user identities, and (for blocks) block headers, the merkle root, etc. Cryptographic hash functions have the characteristics that they are 1) computationally infeasible to reverse, i.e. they are 'one-way hash' functions, and 2) arbitrarily nearby points in the domain are sent to images that are far apart in the range (see e.g. [BSP+95] for an overview). This latter property is especially important–it means that even if two block headers m_1 and m_2 (also called 'messages' to the hash function) differ by a single character, their hash images are very far apart from each other. This also means that it is generally not possible to reconstruct underlying messages by examining hash images which are 'close' to each other.

Proposition 1. *The blockchain state space S cannot be globally decomposed into the product space $T \times \mathcal{B}$.*

Proof. Fix a time $t \in T = \mathbb{R}_+$ and select two blocks B_1 and B_2 from $\mathcal{B}$ with the property that for any positive constants $\delta > 0$ and $M > 0$ and any two messages m_1 and m_2 to be

hashed in B_1 and B_2 (such as the header of the block) such that $\|m_1 - m_2\| < \delta$,

$$\|B_1 - B_2\| > M. \tag{3}$$

This is possible because cryptographic hash functions map arbitrarily nearby messages to arbitrarily distant images.

Suppose, to the contrary, that it were possible to globally decompose S into $T \times \mathcal{B}$. Parametrize the unit circle S^1 by $t \in T := \bar{\mathbb{R}} = \mathbb{R}_+ \cup \{\infty\}$ in the usual way, e.g. identify points $t = 0$ and $t = \infty$, so that $S^1 \simeq \bar{\mathbb{R}}/\sim$ where the equivalence relation '$\sim$' is provided by this identification.[14] We denote the parametrization by $s : T \to S^1$ and note that it is a homeomorphism (with the identified point removed).

Since S is globally a product space, this also defines a homeomorphism from S to the product space $S^1 \times \mathcal{B}$ via $s(t)$ (the identity mapping serves as the mapping from $\mathcal{B}$ to $\mathcal{B}$). We can then define a continuous function $f : S^1 \times \mathcal{B} \to \mathbb{R}$ such that

$$f(s(t), B_1) = +1,\ f(s(1/t), B_2) = -1,\ t \in T,$$

i.e. the function is constant *globally* around S^1 with a value depending upon B_1 and B_2.[15]

Consider an interval around $s(0)$ given by δ, $(s(\delta), s(1/\delta))$. As $\delta \to 0$ we have by assumption that $m_1 \to m_2$, $s(\delta) \to s(0)$ and $s(1/\delta) \to s(0)$, but

$$\lim_{\delta \to 0} f(s(\delta), B_1) \neq \lim_{\delta \to 0} f(s(1/\delta), B_2)$$

since $\|B_1 - B_2\| > M > 0$.

But this implies that if we interpret the value of f as an *orientation*, we cannot assign a unique orientation to the point $s(0)$–the orientation is different depending upon the direction around S^1 we travel. The resulting manifold resembles a Möbius strip, which *cannot* be globally decomposed into a product space of S^1 and $\mathcal{B}$ (note that rather than the usual line segment as the 'fiber' over S^1 for the Möbius strip, here we have a block with a particular orientation, given by f, that acts as the fiber). But then neither can S be globally decomposed into T and $\mathcal{B}$, as this contradicts the assumption that S was homeomorphic to $S^1 \times \mathcal{B}$. □

Thus we see that in spite of its simple construction out of transactions as the fundamental unit, cryptographic hash functions prevent the blockchain state from having a globally decomposable structure into 'time slices'.

Fortunately a local decomposition is always possible, i.e. we can adopt the *fiber bundle* as the natural arena for the blockchain state space.

Definition 3. *The blockchain state space is a fiber bundle $(S, T, \mathcal{B}, \pi)$, where S is the total state space, $\mathcal{B}$ is the standard fiber, T is the base space, and $\pi : S \to \mathcal{B}$ is the projection operator (discussed in further detail below).*

[14] Note that the proof also goes through if we parametrize S^1 using stereographic projection from $\mathbb{R}$ to S^1, where in that case points $t = \infty$ and $t = -\infty$ are identified. This has the advantage of building the limiting neighborhoods used in the proof around the now non-identified point $t = 0$, but the present proof sidesteps the question of indexing time by $\mathbb{R}$ and suffers no inconsistency from the 'endpoint stitching' parametrization.

[15] We are free to do this because of (3).

We let S be an $(n+1)$-dimensional manifold and $\mathcal{B}$ an n-dimensional manifold, but we cannot generally endow $\mathcal{B}$ with a vector space structure, due to the existence of cryptographic hash functions (as discussed in Proposition 1 above). As before we let T be given by $\mathbb{R}_+$ and will interpret $t \in T$ as time.

Consider next an *atlas* $\mathcal{A}$ over the base space, i.e. a set of *charts* $(U_\alpha, \phi_\alpha)_{\alpha \in \mathcal{I}}$ with neighborhoods U_α, homeomorphisms ϕ_α, and $\mathcal{I}$ an index set, such that

$$\phi_\alpha : \pi^{-1}(U_\alpha) \subset S \to U_\alpha \times \mathcal{B}$$

defines a local trivialization mapping a subset of S locally to a product space of time and the block space $\mathcal{B}$. We need to ensure that in a non-empty overlap between two trivializations $(U_\alpha, \phi_\alpha), (U_\beta, \phi_\beta) \in \mathcal{A}$ we can consistently define a transition function between trivializations at the same location $s \in S$. Note that in what follows, whenever we have only two charts we will usually denote them by (U, ϕ_U) and (V, ϕ_V) rather than by (U_α, ϕ_α) and (U_β, ϕ_β) for notational exposition.

Select (U, ϕ_U) and (V, ϕ_V) from $\mathcal{A}$ such that $U \cap V \neq \varnothing$. Select a point $t \in U \cap V$ and ask what the local trivializations are at this point. We have, for $s \in \pi^{-1}(t)$,

$$\phi_U(s) = (t, B_U),\ \phi_V(s) = (t, B_V),$$

where $B_U, B_V \in \mathcal{B}$ are the representations under ϕ_U and ϕ_V of the fiber over t. But what does it mean for a block to have two different representations?

3.2.1 Block Representations and Equivalence

We have defined the state of the blockchain as "all relevant information required for making decisions". A block $B \in \mathcal{B}$ is, strictly speaking, a non-unique representation of the information required for decision-making (where, to repeat, by 'decision-maker' we include both human entities and also (semi-) autonomous agents such as smart contracts). There are necessary components of B, such as the nonce hash for Bitcoin mining, which are required for the blockchain to operate but which are not germane to decision-making. In other words, two different blocks B_1 and B_2 having different nonce hashes *represent the same information for decision-making, and form a part of the same underlying state.* Note that we are agnostic about what constitutes sufficient information for decision-making, and so there may be other ways that syntactically distinct blocks convey the same information–for some cases, for example, it may be that the transaction address hashes are not relevant, only their associated balances. In that case, two blocks with the same balance state but different address hash structure would nevertheless be considered "the same".

We can formalize this "sameness" by declaring that two blocks which represent the same underlying information are equivalent, and thus partition the space $\mathcal{B}$ of blocks into sets of equivalence classes. In order to do this, we use the fact that the state space is a fiber bundle, possessing local trivializations that act as different representations of an underlying 'standard fiber'. Here a 'standard fiber' is just a representative member of an equivalence class of blocks:

Definition 4. *Given two charts $(U, \phi_U) \in \mathcal{A}$ and $(V, \phi_V) \in \mathcal{A}$ with $U \cap V \neq \varnothing$ and blocks*

$B_1, B_2 \in \mathcal{B}$, we declare B_1 ***equivalent*** *to B_2 at $t \in U \cap V$, denoted $B_1 \sim_t B_2$, if*

$$\pi \circ \phi_U^{-1}(t, B_1) = \pi \circ \phi_V^{-1}(t, B_2) = t.$$

The set of all equivalence classes over $\mathcal{B}$ at t is $\mathcal{B}/\sim_t$.

Although it is not the case that the blockchain state is globally parametrized by time, Definition 4 still allows us to recover the idea that cryptographically distinct blocks nevertheless can share a 'sameness' from the point of view of decision-making. This sameness is derived from the *local* decomposition of the blockchain state into a product space from Definition 3, where charts from the atlas $\mathcal{A}$ serve as the cryptographically distinct representations of a 'standard block'.

3.3 Actions and Outcomes

An individual agent, whether a human being or a smart contract, performs actions in response to the state of the blockchain. These actions in turn generate an **outcome** of either an individual's action or, more typically, of the collective actions of many agents. A simple example would again be a transfer of bitcoin on the Bitcoin network. Alice posts a transfer of X bitcoin to Bob. This transfer posting is the action of Alice, and depends upon the state of Bitcoin's blockchain (i.e., the Bitcoin addresses of Alice and Bob). The outcome of the transfer posting is that X bitcoin are debited against Alice's account and credited against Bob's account. In this case, the outcome is *determined*, in that there is no randomness–precisely X is transferred from Alice's address to Bob's address.

We can think of transactions which are more complicated. For example, an insurance contract would be contingent upon not just an agent's action (such as filing a claim), but also upon external data, such as the type of accident, damage done, and fault. The agent's action is again deterministic–a claim is either filed, or it is not–but the outcome is determined by the realization of what were, *when the contract was formed*, a set of random variables.

Finally, there may be contracts where collectively the actions of agents do not 'pin down' an outcome, but instead themselves create a distribution of possible outcomes. Consider, for example, the actions of agents as determining the distribution of winning hands of poker, i.e. an 'action profile' is a list of games of poker played, and the outcome is the set of all hands which won, associated with their relative frequency of occurrence. Such an outcome is a distribution of hands over the support (winning hands of poker), and since the outcome is dependent upon the stochastic realization of multiple decks of cards, it is *probabilistic*.

For simplicity, we shall assume that while outcomes may be deterministic or stochastic, agents are constrained to select a single action that influences the blockchain state. That is, regardless of whether they are themselves using (in game theoretic parlance) a "pure" or a "mixed" strategy, they submit one and one one action to the blockchain, rather than a set of actions:

Definition 5. *A blockchain agent i's* ***action function*** *is a surjection $a_i : S \to A_i$, where A_i is a set of possible actions that agent i may take that influence the blockchain's state. An* ***action profile*** *$a := (a_1(s), \ldots, a_N(s))$ is an N-dimensional vector of agent actions, $a \in \Pi_{i=1}^{N} A_i =: A$, where N is the number of agents.*

3.4 State Transition

Once a set of agents has created an action profile a, the blockchain still needs to incorporate this profile into the current state in order to generate the next state. Recall that every blockchain implementation generates a new block from a consensus mechanism. For public, permissionless blockchains a consensus mechanism may use e.g. Proof-of-Work or Proof-of-Stake, where competition between nodes serves to incentivize the creation of new blocks. Permissioned blockchains, on the other hand, may use validator nodes at the block level, or transaction participants at the transactions level, to determine whether or not new transactions are valid and hence should be included in a new block. In either case, any given transaction (or block) is not *certain* to be included in the blockchain as part of a new state representation. As discussed earlier, Bitcoin's blockchain consensus mechanism, Proof-of-Work, is stochastic–there is some probability that a particular node (or pool of nodes) will solve the computational puzzle first, and hence include their candidate block as a new part of the longest chain (on average, every 10 minutes). For consensus mechanisms which do not rely upon competition, and which may then be deterministic, peer-to-peer communication is nevertheless uncertain (peers required for e.g. endorsing or validating a transaction may be unavailable, or may be subject to delays/network lag, etc.).

A general formulation of blockchain state transitions should include some method of handling uncertainty. In the current context we extend the discussion on the state of the blockchain by focusing only upon blockchain *representations*, i.e. those members from the set of equivalence classes $\mathcal{B}/\sim$ that are considered, from the point of decision-making, to be equivalent to one another. This is sufficient to induce a measure of the set of representations that provide the probabilistic transition function between blockchain states.

Definition 6. *A **consensus probability measure** $\mu : \mathcal{B} \to [0,1]$ is a mapping from the set of blocks to the unit interval, such that*

$$\mu(B) > 0 \Leftrightarrow B \text{ is a } \textit{possible} \text{ future block, and}$$

$$\sum_{B \in \mathcal{B}} \mu(B) = 1.$$

We denote the space of all consensus probability measures by $\mathcal{P}_c$.

One or more possible blocks may belong to the same equivalence class (cf. Definition 4). We can induce a measure over equivalence classes simply by summing the measures over the representatives of the class: given any representative block $B \in [B]$,

$$\mu([B]) := \sum_{B \in [B]} \mu(B).$$

Definition 7. *A **state probability measure** μ_S over the state space S is defined as the total weight of the equivalence class of blocks representing each state, i.e. if given a chart $(U, \phi_U) \in \mathcal{A}$ with $t \in U$ and*

$$s_{[B]} := \phi(t, B)$$

we have

$$\mu_S(s_{[B]}) := \mu([B]).$$

Paralleling the consensus probability measure we denote the space of state probability measures by $\mathcal{P}_S$.

Using the state probability measure we can define the (probabilistic) transition mapping for the blockchain:

Definition 8. *A **state transition** mapping $\tau : S \times A \times \mathcal{P}_S \rightrightarrows \mathcal{P}(S)$ is a correspondence assigning to a state $s \in S$, an action profile $a \in A$, and a state probability measure $\mu_S \in \mathcal{P}_S$ a set W in the power set $\mathcal{P}(S)$, i.e.*

$$W := \tau(s, a, \mu_S) \in \mathcal{P}(S)$$

is the set of possible future states.

3.5 Probabilistic State Machine Representation

The foregoing analysis has led us to a formalization of blockchain as a state machine–note that although we have assumed throughout that the sufficient information for decision-making has been limited a block, rather than a ledger of blocks, without loss of generality we may assume that the fiber bundle representation of the state space locally decomposes to a product space of time and a *ledger space* $\mathcal{L}$, where a ledger $L \in \mathcal{L}$ is a sequence of blocks (cf. equation 1). This is true because a sequence of blocks is simply a list of transactions, and a list of transactions may itself be considered (again, from the point of view of decision-making) as one large block.

Thus the definition of the state machine representation of blockchain, given below, can encode a ledger space into the projection mapping π of the fiber bundle, and the definition may be seen as equally valid for 'memoryless' blockchains with block space $\mathcal{B}$, where a block is sufficient for decision-making, as for full ledger blockchains with ledger space $\mathcal{L}$.

Definition 9. *A blockchain is a **probabilistic state machine** with state space S, fiber bundle representation $S \xrightarrow{\pi} T$ and state transition mapping τ such that given $s \in S$, $a \in A$ and $\mu_S \in \mathcal{P}_S$, the next state $s' \in S$ is*

$$s' \in \tau(s, a, \mu_S).$$

4 Agent Automata: Rubinstein's Machines

In examining the foundations and interactions in repeated (non-cooperative) games, [Rub86] Rubinstein defined a formal mapping between agents that engage in a repeated game and a 'finite automata' that represents the agent, so that agents are restricted in the complexity of the strategies they use.[16] In addition to formally defining how such a 'machine' can represent individual preferences and actions, and how the resulting outcomes can be encoded as a 'state' to condition upon for future actions, Rubinstein also investigated how agents would select between alternative machines with differing complexity costs. By

[16]This limit on the complexity, or on the computational power, of an agent places this analysis firmly within the context of behavioral economics (developed by Herbert Simon and others), an increasingly important research area as artificial intelligence gains traction in mainstream digital interactions, such as 'chatbots'.

optimizing over the space of machines, agents would then select the best machine across the possible states of a repeated game. This analysis was extended further by [AR88] and many others to characterize the resulting equilibrium strategies and associated payoffs.

Rubinstein's definition of an automaton, or 'machine' for a player i, requires the existence of (cf. [OR94], p. 140)

1. a set of states Q_i,
2. an action set A_i,
3. an initial state $q_i^0 \in Q_i$,
4. a mapping associating states to actions $f_i : Q_i \to A_i$, and finally
5. a transition function associating states and actions to states, $\tau_i : Q_i \times \Pi_i A_i \to Q_i$.

In the present work we can see that Rubinstein's finite automata approach neatly incorporates an agent (defined as a human blockchain participant, an external data 'oracle', or a blockchain-present 'chaincode' or smart contract) into the blockchain state machine representation. As discussed, agents interact with the blockchain through their selection of an action, and they collectively select an action profile (cf. Definition 5). Thus, at a naive level we can make the following associations to connect *any* agent to a 'machine' *à la* Rubinstein (objects on the left are from [OR94], while objects on the right are from this work):

1. $Q_i \Leftrightarrow S$,
2. $A_i \Leftrightarrow A_i$,
3. $q_i^0 \Leftrightarrow B_0 = L_0$,
4. $f_i \Leftrightarrow a_i$,
5. $\tau_i \Leftrightarrow \tau$.

Notice that what are idiosyncratic states in Rubinstein's approach (the set Q_i) are here global blockchain states (S), meaning that if a state were to include private information as it pertains to decision-making, it would need to be specified off-blockchain. We have sidestepped this issue in the current analysis by assuming that all required information is available on-blockchain, so that 'idiosyncratic' here would mean a restriction of an agent's use of the global blockchain state S to a maximal subset $Q_i \subseteq S$ of available information used in the action function a_i. Note also that the blockchain transition function τ imputes a machine transition function τ_i, by mapping elements of S (via Q_i) to other elements of S (also in Q_i)–it is in this sense that Q_i is 'maximal'.

However, it should be understood that the states Q_i as defined by Rubinstein reflect the *internal* states of the machine representation of the agent, and not (necessarily) the global blockchain states S. The naive characterization above would only be valid if all possible machine states could only be recorded *on* the blockchain, as would be the case with e.g. chaincode or a smart contract. By contrast, a machine representation of a human being's decision-making would likely require an intermediate mapping from S to Q_i, making them distinct, so that the external global blockchain state would in turn affect an individual's

internal machine state in some fashion. This is a crucial point–it is known that in some situations, such as with infinitely repeated games, a complete characterization of the set of strategies available to a player requires a machine with an *infinite* number of states! We leave this level of behavioral modeling, which is fascinating in its own right, and simply present the naive view of a finite state machine with 'direct' state dependence $Q_i \simeq S$, as a sufficient demonstration, or proof-of-concept, that blockchain participants may be thought of as *finite* automata.

5 Machine Hierarchy and Future Research

Although the richness of human behavior in all likelihood cannot be captured by a state machine (finite or otherwise), innovations such as the Internet and DLTs continue to redefine how human agents can interact with their digital counterparts, be they artificially intelligent agents, bits of code encapsulated in blockchain smart contracts, or 'avatars' acting on behalf of their human counterparts. The discussion presented here has demonstrated that this interaction can be thought of as a hierarchy of behavior within the state machine paradigm:

- individual agents may be represented by finite automata, which interact with each other and the blockchain;
- this interaction defines an execution of state-dependent code (via the blockchain protocol, smart contract, etc.); and
- the entire blockchain state itself transitions in a probabilistic fashion from one state to the next, as defined in the framework of Section 3.

A schematic representation of this hierarchical nesting of state machines is given in Figure 1.

The utility of such a framework is twofold: first, it is largely pedagogical, in that it sets out explicitly what is being specified in a blockchain and how that specification evolves over time. This, in turn, provides a useful point of departure to build more realistic (and complex) models of behavior, to examine a blockchain's state equilibria (or 'path of motion'), its ability to connect agents with no prior relationship in a system of trust, and its ability to potentially replace costly third-party contract intermediaries with a transparent, immutable ledger of transactions.

Accepting the blockchain framework as setting the boundaries for behavioral interaction opens the door for studying not just the (usually non-cooperative) game theoretic foundations of e.g. consensus mechanisms, which is already well-established in the literature, but also the equilibrium outcomes of large-scale, 'multi-signature' contracting arrangements found more often in cooperative game theory. This in turn provides an opportunity to turn the question around by defining a desired outcome that a blockchain contracting procedure must achieve, and then tackling the resulting *mechanism design* problem that must be solved in order to support that outcome. In this sense blockchain can move beyond its roots as a DLT and become a way to fashion autonomous contracting relationships that are properly designed to mitigate problems with incentive compatibility and enforcement, while at the same time preserving the transparency and immutability that have placed it at the forefront of database innovation to date.

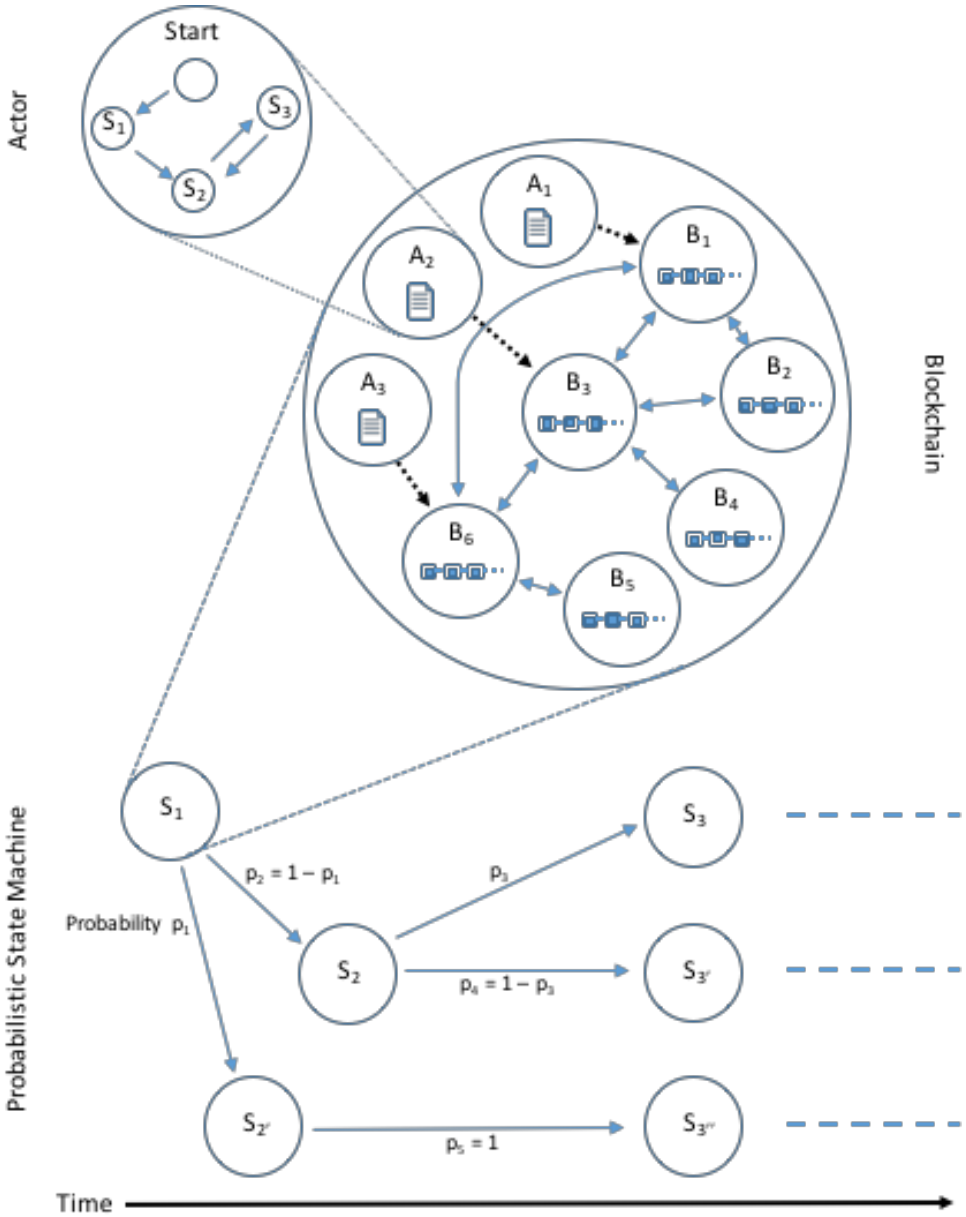

Figure 1: Hierarchical state machines in a blockchain: **A** nodes are agents (human, chaincode, smart contracts, etc.), **B** nodes are ledger nodes (validators, miners, etc.), **S** nodes represent the blockchain state, and probabilities are imputed from the state transition mapping τ. Each agent is represented as a (finite) automaton, while the blockchain evolves according to a probabilistic state machine.

References

[All17] Darcy W. E. Allen. "Discovering and Developing the Blockchain Crypto-Economy". In: *SSRN Electronic Journal* (2017). DOI: 10.2139/ssrn.2815255.

[AR88] Dilip Abreu and Ariel Rubinstein. "The Structure of Nash Equilibrium in Repeated Games with Finite Automata". In: *Econometrica* 56.6 (Nov. 1988), p. 1259. DOI: 10.2307/1913097.

[Böh+15] Rainer Böhme et al. "Bitcoin: Economics, Technology, and Governance". In: *Journal of Economic Perspectives* 29.2 (May 2015), pp. 213–238. DOI: 10.1257/jep.29.2.213.

[BS18] Aleksander Berentsen and Fabian Schar. "A Short Introduction to the World of Cryptocurrencies". In: *Review* 100.1 (2018), pp. 1–16. DOI: 10.20955/r.2018.1-16.

[BSP+95] Shahram Bakhtiari, Reihaneh Safavi-Naini, Josef Pieprzyk, et al. "Cryptographic hash functions: A survey". In: *Centre for Computer Security Research, Department of Computer Science, University of Wollongong, Australie* (1995).

[CG16] Christian Catalini and Joshua Gans. *Some Simple Economics of the Blockchain*. Tech. rep. Dec. 2016. DOI: 10.3386/w22952.

[con18] bitcoinwiki contributors, ed. *bitcoin wiki: Block timestamp*. retrieved January 2018. URL: https://en.bitcoin.it/wiki/Block_timestamp.

[DFP16] Sinclair Davidson, Primavera De Filippi, and Jason Potts. "Economics of Blockchain". In: *SSRN Electronic Journal* (2016). DOI: 10.2139/ssrn.2744751. URL: https://ssrn.com/abstract=2744751.

[Kon+14] Dániel Kondor et al. "Do the Rich Get Richer? An Empirical Analysis of the Bitcoin Transaction Network". In: *PLoS ONE* 9.2 (Feb. 2014). Ed. by Matjaž Perc, e86197. DOI: 10.1371/journal.pone.0086197.

[Lac16] Naomi Lachance. "Not Just Bitcoin: Why The Blockchain Is A Seductive Technology To Many Industries". In: *National Public Radio Online* (May 2016). URL: https://www.npr.org/sections/alltechconsidered/2016/05/04/476597296/not-just-bitcoin-why-blockchain-is-a-seductive-technology-to-many-industries.

[Mar16] Bernard Marr. "How Blockchain Technology Could Change The World". In: *Forbes* (May 2016). URL: https://www.forbes.com/sites/bernardmarr/2016/05/27/how-blockchain-technology-could-change-the-world/#6609318c725b.

[Nak09] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: (2009). Retrieved December 2017. URL: https://bitcoin.org/en/bitcoin-paper.

[OR94] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press Ltd, July 12, 1994. 368 pp. ISBN: 0262650401.

[Rub86] Ariel Rubinstein. "Finite automata play the repeated prisoner's dilemma". In: *Journal of Economic Theory* 39.1 (June 1986), pp. 83–96. DOI: 10.1016/0022-0531(86)90021-9.

[Sip13] Michael Sipser. *Introduction to the Theory of Computation*. Third Edition. Cengage Learning, 2013.

[SY16] Kenji Saito and Hiroyuki Yamada. "What's So Different about Blockchain? — Blockchain is a Probabilistic State Machine". In: *2016 IEEE 36th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, June 2016. DOI: 10.1109/icdcsw.2016.28.

[TFS17] Jesus Leal Trujillo, Steve Fromhart, and Val Srinivas. "Evolution of blockchain technology: Insights from the GitHub platform". In: *Deloitte Insights* (Nov. 2017). URL: https://www2.deloitte.com/insights/us/en/industry/financial-services/evolution-of-blockchain-github-platform.html.