A Logic of Blockchain Updates

Kai Brünnler[1], Dandolo Flumini[2], and Thomas Studer[3]

[1] Bern University of Applied Sciences, Switzerland, kai.bruennler@bfh.ch
[2] ZHAW School of Engineering, Switzerland, dandolo.flumini@zhaw.ch
[3] University of Bern, Switzerland, tstuder@inf.unibe.ch

Abstract. Blockchains are distributed data structures that are used to achieve consensus in systems for cryptocurrencies (like Bitcoin) or smart contracts (like Ethereum). Although blockchains gained a lot of popularity recently, there are only few logic-based models for blockchains available. We introduce BCL, a dynamic logic to reason about blockchain updates, and show that BCL is sound and complete with respect to a simple blockchain model.

Keywords: blockchain, modal logic, dynamic epistemic logic

1 Introduction

Bitcoin [16] is a cryptocurrency that uses peer-to-peer technology to support direct user-to-user transactions without an intermediary such as a bank or credit card company. In order to prevent double spending, which is a common issue in systems without central control, Bitcoin maintains a complete and public record of all transactions at each node in the network. This ledger is called the *blockchain*.

The blockchain is essentially a growing sequence of blocks, which contain approved transactions and a cryptographic hash of the previous block in the sequence. Because the blockchain is stored locally at each node, any update to it has to be propagated to the entire network. Nodes that receive a transaction [1, 18]

1. first verify its validity (i.e., whether it is compatible with all preceeding transactions);
2. if it is valid, then it is added to the blockchain and
3. sent to all other nodes.

Blockchain technology, as a general solution to the Byzantine Generals' Problem [15], is now not only used for financial transactions but also for many other applications like, e.g., smart contracts [5].

Herlihy and Moir [11] propose to develop a logic of accountability to design and verify blockchain systems. In particular, they discuss blockchain scenarios to test (i) logics of authorization, (ii) logics of concurrency, and (iii) logics of incentives.

Halpern and Pass [10] provide a characterization of agents' knowledge when running a blockchain protocol using a variant of common knowledge.

In the present paper, we are not interested in accountability or aspects of common knowledge. We study the local, single agent perspective of a blockchain. That is we investigate steps 1. and 2. of the above procedure for receiving a transaction. Our approach is inspired by dynamic epistemic logic [7]. A given state of the local blockchain entails knowledge about the transactions that have taken place. We ask: *how does this knowledge change when a new block is received that might be added to the blockchain?* We develop a dynamic logic, BCL, with a semantics that is based on a blockchain model. The update operators of BCL are interpreted as receiving new blocks. It is the aim of this paper to investigate the dynamics of local blockchain updates.

The deductive system for BCL includes reduction axioms that make it possible to establish completeness by a reduction to the update-free case [13]. However, since blockchain updates are only performed if certain consistency conditions are satisfied, we use conditional reduction axioms similar to the ones developed by Steiner to model consistency preserving updates [19]. Moreover, unlike traditional public announcements [7], blockchain updates cannot lead to an inconsistent state, i.e., updates are total, like in [20].

We do not base BCL on an existing blockchain implementation but use a very simple model. First of all, the blockchain is a sequence of propositional formulas. Further, we maintain a list of provisional updates. Our blocks consist of two parts: a sequence number (called the index of the block) and a propositional formula. If a block is received, then the following case distinction is performed where i is the index of the block and l is the current length of the blockchain:

1. $i \leq l$. The block is ignored.
2. $i = l + 1$. If the formula of the block is consistent with the blockchain, then it is added to the blockchain; otherwise the

block is ignored. If the blockchain has been extended, then this procedure is performed also with the blocks stored in the list of provisional updates.

3. $i > l + 1$. The block is added to the list of provisional updates.

Although this is a simple model, it features two important logical properties of blockchains: consistency must be preserved and blocks may be received in the wrong order, in which case they are stored separately until the missing blocks have been received.

The main contribution of our paper from the point of view of dynamic epistemic logic is that we maintain a list of provisional updates. That means we support updates that do not have an immediate effect but that may lead to a belief change later only after certain other updates have been performed. BCL is the first logic that features provisional updates of this kind.

The paper is organized as follows. The next section introduces our blockchain model, the language of BCL, and its semantics. In Section 3, we introduce a deductive system for BCL. We establish soundness of BCL in Section 4. In Section 5, we show a normal form theorem for BCL, which is used in Section 6 to prove completeness of BCL. The final section studies some key principles of the dynamics of our blockchain logic and discusses future work.

We will only mention the lemmas that are needed to establish the main theorems of this paper; but we do not give any proofs because of lack of space. Detailed proofs can be found in the accompanying arXiv paper [3].

Acknowledgements
We would like to thank Eveline Lehmann and Nenad Savic for carefully reading a previous version of this paper.

This work was partially supported by the SNSF grant 165549.

2 A simple blockchain logic

The set of all natural numbers is denoted by $\mathbb{N} := \{0, 1, 2, \ldots\}$. The set of positive natural numbers is denoted by $\mathbb{N}^+ := \{1, 2, \ldots\}$. We use ω for the least ordinal such that $\omega > n$, for all $n \in \mathbb{N}$.

Let $\sigma = \langle \sigma_1, \ldots, \sigma_n \rangle$ be a finite sequence. We define its *length* by $\mathsf{len}(\sigma) := n$. For an infinite sequence $\sigma = \langle \sigma_1, \sigma_2, \ldots \rangle$ we set

$\mathsf{len}(\sigma) := \omega$. For a (finite or infinite) sequence $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_i, \ldots \rangle$ we set $(\sigma)_i := \sigma_i$. The *empty sequence* is denoted by $\langle\rangle$ and we set $\mathsf{len}(\langle\rangle) := 0$. We can append x to a finite sequence $\sigma := \langle \sigma_1, \ldots, \sigma_n \rangle$, in symbols we set $\sigma \circ x := \langle \sigma_1, \ldots, \sigma_n, x \rangle$. We will also need the set of all components of a sequence σ and define

$$\mathsf{set}(\sigma) := \{x \mid \text{there is an } i \text{ such that } x = \sigma_i\}.$$

In particular, we have $\mathsf{set}(\langle\rangle) := \emptyset$. Moreover, we use the shorthand $x \in \sigma$ for $x \in \mathsf{set}(\sigma)$.

We start with a countable set of atomic propositions $\mathcal{AP} := \{P0, P1, \ldots\}$. The set of formulas $\mathcal{L}_{\mathsf{cl}}$ of classical propositional logic is given by the following grammar

$$A ::= \bot \mid P \mid A \to A \quad ,$$

where $P \in \mathcal{AP}$.

In order to introduce the language $\mathcal{L}_{\mathsf{B}}$ for blockchain logic, we need another countable set of special atomic propositions $\mathcal{AQ} := \{Q1, Q2, \ldots\}$ that is disjoint with $\mathcal{AP}$. We will use these special propositions later to keep track of the length of the blockchain. The formulas of $\mathcal{L}_{\mathsf{B}}$ are now given by the grammar

$$F ::= \bot \mid P \mid Q \mid F \to F \mid \Box A \mid [i, A]F \quad ,$$

where $P \in \mathcal{AP}$, $Q \in \mathcal{AQ}$, $A \in \mathcal{L}_{\mathsf{cl}}$, and $i \in \mathbb{N}^+$. The operators of the form $[i, A]$ are called *blockchain updates* (or simply *updates*).

Note that in $\mathcal{L}_{\mathsf{B}}$ we cannot express higher-order knowledge, i.e., we can only express knowledge about propositional facts but not knowledge about knowledge of such facts.

For all languages in this paper, we define further Boolean connectives (e.g. for negation, conjunction, and disjunction) as usual. Moreover, we assume that unary connectives bind stronger than binary ones.

For $\mathcal{L}_{\mathsf{cl}}$ we use the semantics of classical propositional logic. A *valuation* v is a subset of $\mathcal{AP}$ and we define the truth of an $\mathcal{L}_{\mathsf{cl}}$-formula A under v, in symbols $\mathsf{v} \models A$ as usual. For a set Γ of $\mathcal{L}_{\mathsf{cl}}$-formulas, we write $\mathsf{v} \models \Gamma$ if $\mathsf{v} \models A$ for all $A \in \Gamma$. The set Γ

is *satisfiable* if there is a valuation v such that $\mathsf{v} \models \Gamma$. We say Γ *entails* A, in symbols $\Gamma \models A$, if for each valuation v we have

$$\mathsf{v} \models \Gamma \quad \text{implies} \quad \mathsf{v} \models A.$$

Now we introduce the blockchain semantics for $\mathcal{L}_{\mathsf{B}}$.

Definition 1. *A* block *is a pair* $[i, A]$ *where* A *is an* $\mathcal{L}_{\mathsf{cl}}$*-formula and* $i \in \mathbb{N}^+$*. We call* i *the* index *and* A *the* formula *of the block* $[i, A]$*. We define functions* ind *and* fml *by* $\mathsf{ind}[i, A] := i$ *and* $\mathsf{fml}[i, A] := A$*.*

Definition 2. *A* model $\mathsf{M} := (\mathsf{I}, \mathsf{BC}, \mathsf{PU}, \mathsf{v})$ *is a quadruple where*

1. I *is a set of* $\mathcal{L}_{\mathsf{cl}}$*-formulas*
2. BC *is a sequence of* $\mathcal{L}_{\mathsf{cl}}$*-formulas*
3. PU *is a finite sequence of blocks*
4. v *is a valuation, i.e.* $\mathsf{v} \subseteq \mathcal{AP}$

such that

$$\mathsf{I} \cup \mathsf{set}(\mathsf{BC}) \text{ is satisfiable} \tag{1}$$

and

$$\text{for each block } [i, A] \in \mathsf{PU} \text{ we have } i > \mathsf{len}(\mathsf{BC}) + 1. \tag{2}$$

The components of a model $(\mathsf{I}, \mathsf{BC}, \mathsf{PU}, \mathsf{v})$ have the following meaning:

1. I models initial background knowledge.
2. BC is the blockchain.
3. PU stands for *provisional updates*. The sequence PU consists of those blocks that have been announced but that could not yet be added to the blockchain because their index is too high. Maybe they will be added to BC later (i.e., after the missing blocks have been added).
4. v states which atomic propositions are true.

We need some auxiliary definition in order to precisely describe the blockchain dynamics.

Definition 3. *1. Let* PU *be a finite sequence of blocks. Then we let* $\mathsf{find}(i, \mathsf{PU})$ *be the least* $j \in \mathbb{N}^+$ *such that there is an* $\mathcal{L}_{\mathsf{cl}}$*-formula* A *with* $[i, A] = (\mathsf{PU})_j$*.*

2. *Let* $\sigma = \langle \sigma_1, \ldots, \sigma_{i-1}, \sigma_i, \sigma_{i+1}, \ldots \rangle$ *be a sequence. We set*

$$\mathsf{remove}(i, \sigma) := \langle \sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots \rangle.$$

3. *Given a set of* $\mathcal{L}_{\mathsf{cl}}$*-formulas* I*, a sequence of* $\mathcal{L}_{\mathsf{cl}}$*-formulas* BC*, and a finite sequence of blocks* PU*, then the* chain completion $\mathsf{complete}(\mathsf{I}, \mathsf{BC}, \mathsf{PU})$ *is computed according to Algorithm 1.*

Algorithm 1 Chain Completion Algorithm: complete

Input: $(\mathsf{I}, \mathsf{BC}, \mathsf{PU})$
1: $n \leftarrow \mathsf{len}(\mathsf{BC}) + 1$
2: **while** $[n, A] \in \mathsf{PU}$ for some formula A **do**
3: $\quad i \leftarrow \mathsf{find}(n, \mathsf{PU})$
4: $\quad B \leftarrow \mathsf{fml}((\mathsf{PU})_i)$
5: $\quad \mathsf{remove}(i, \mathsf{PU})$
6: $\quad$ **if** $\mathsf{I} \cup \mathsf{set}(\mathsf{BC}) \cup \{B\}$ is satisfiable **then**
7: $\quad\quad \mathsf{BC} \leftarrow \mathsf{BC} \circ B$
8: $\quad\quad n \leftarrow \mathsf{len}(\mathsf{BC}) + 1$
9: $\quad$ **end if**
10: **end while**
11: **for** $i \in \mathsf{len}(\mathsf{PU}), \ldots, 1$ **do**
12: $\quad$ **if** $\mathsf{ind}((\mathsf{PU})_i) < n$ **then**
13: $\quad\quad \mathsf{remove}(i, \mathsf{PU})$
14: $\quad$ **end if**
15: **end for**
16: **return** $(\mathsf{BC}, \mathsf{PU})$

Let us comment on the chain completion procedure. The numbers refer to the lines in Algorithm 1.

- 1: n is the index a block must contain so that it could be added to the blockchain BC.
- 2: '$[n, A] \in \mathsf{PU}$ for some formula A' means that PU contains a block that could be added to BC.
- 3–5: Find the next formula B that could be added to BC and remove the corresponding block from PU.
- 6: '$\mathsf{I} \cup \mathsf{set}(\mathsf{BC}) \cup \{B\}$ is satisfiable' means that B is consistent with the current belief. This test guarantees that (1) will always be satisfied.
- 7,8: Update the blockchain BC with B.

11–15: Remove all blocks from PU whose index is less than or equal to the current length of the blockchain BC. Because the blockchain never gets shorter, these block will never be added. Removing them guarantees that (2) will always be satisfied.

Note if BC and PU satisfy condition (2) in the definition of a model, then the chain completion algorithm will return BC and PU unchanged.

Lemma 1. *Let* I *be a set of* $\mathcal{L}_{\mathsf{cl}}$*-formulas and let* BC *be a sequence of* $\mathcal{L}_{\mathsf{cl}}$*-formulas such that* $\mathsf{I} \cup \mathsf{set}(\mathsf{BC})$ *is satisfiable. Let* PU *be an arbitrary finite sequence of blocks. For* $(\mathsf{BC}', \mathsf{PU}') := \mathsf{complete}(\mathsf{I}, \mathsf{BC}, \mathsf{PU})$ *we find that*

1. $\mathsf{I} \cup \mathsf{set}(\mathsf{BC}')$ *is satisfiable and*
2. *for each block* $[i, A] \in \mathsf{PU}'$ *we have* $i > \mathsf{len}(\mathsf{BC}') + 1$.

Definition 4. *Let* $\mathsf{M} := (\mathsf{I}, \mathsf{BC}, \mathsf{PU}, \mathsf{v})$ *be a model and* $[i, A]$ *be a block. The* updated model $\mathsf{M}^{[i,A]}$ *is defined as* $(\mathsf{I}, \mathsf{BC}', \mathsf{PU}', \mathsf{v})$ *where*

$$(\mathsf{BC}', \mathsf{PU}') := \mathsf{complete}(\mathsf{I}, \mathsf{BC}, \mathsf{PU} \circ [i, A]).$$

Remark 1. Note that $\mathsf{M}^{[i,A]}$ is well-defined: by Lemma 1 we know that $\mathsf{M}^{[i,A]}$ is indeed a model.

Definition 5. *Let* $\mathsf{M} := (\mathsf{I}, \mathsf{BC}, \mathsf{PU}, \mathsf{v})$ *be a model. We define the* truth *of an* $\mathcal{L}_{\mathsf{B}}$*-formula* F *in* M*, in symbols* $\mathsf{M} \models F$*, inductively by:*

1. $\mathsf{M} \not\models \bot$*;*
2. $\mathsf{M} \models P$ *if* $P \in \mathsf{v}$ *for* $P \in \mathcal{AP}$*;*
3. $\mathsf{M} \models Qi$ *if* $i \leq \mathsf{len}(\mathsf{BC})$ *for* $Qi \in \mathcal{AQ}$*;*
4. $\mathsf{M} \models F \to G$ *if* $\mathsf{M} \not\models F$ *or* $\mathsf{M} \models G$*;*
5. $\mathsf{M} \models \Box A$ *if* $\mathsf{I} \cup \mathsf{set}(\mathsf{BC}) \models A$*;*
6. $\mathsf{M} \models [i, A]F$ *if* $\mathsf{M}^{[i,A]} \models F$.

We define validity only with respect to the class of models that do not have provisional updates.

Definition 6. *We call a model* $\mathsf{M} = (\mathsf{I}, \mathsf{BC}, \mathsf{PU}, \mathsf{v})$ initial *if* $\mathsf{PU} = \langle\rangle$. *A formula* F *is called* valid *if* $\mathsf{M} \models F$ *for all initial models* M.

3 The deductive system BCL

In order to present an axiomatic system for our blockchain logic, we need to formalize an *acceptance condition* stating whether a received block can be added to the blockchain. That is we need a formula $\mathsf{Acc}(i, A)$ expressing that the formula A is consistent with the current beliefs and the current length of the blockchain is $i - 1$. Thus if $\mathsf{Acc}(i, A)$ holds, then the block $[i, A]$ will be accepted and added to the blockchain. The truth definition for the atomic propositions $Qi \in \mathcal{AQ}$ says that Qi is true if the blockchain contains at least i elements. That means the formula $Q(i-1) \wedge \neg Qi$ is true if the blockchain contains exactly $i - 1$ elements. This leads to the following definition of $\mathsf{Acc}(i, A)$ for $i \in \mathbb{N}^+$:

$$\mathsf{Acc}(i, A) := \begin{cases} \neg Qi \wedge \neg\Box\neg A & \text{if } i = 1 \\ Q(i-1) \wedge \neg Qi \wedge \neg\Box\neg A & \text{if } i > 1 \end{cases}$$

As desired, we find that if $\mathsf{Acc}(i, A)$ is true, then the chain completion algorithm can append the formula A to the blockchain (see Lemma 2 later).

An $\mathcal{L}_{\mathsf{B}}$-formula is called compliant if the blockchain updates occur in the correct order. Formally, we use the following definition.

Definition 7. *An $\mathcal{L}_{\mathsf{B}}$-formula F is* compliant *if no occurrence of a $[i, A]$-operator in F is in the scope of some $[j, B]$-operator with $j > i$.*

Now we can define a deductive system for BCL. It is formulated in the language $\mathcal{L}_{\mathsf{B}}$ and consists of the following axioms:

(PT)	Every instance of a propositional tautology
(K)	$\Box(F \to G) \to (\Box F \to \Box G)$
(D)	$\neg\Box\bot$
(Q)	$Qi \to Qj$ if $i > j$
(A1)	$[i, A]\bot \to \bot$
(A2)	$[i, A]P \leftrightarrow P$ for $P \in \mathcal{AP}$
(A3.1)	$\mathsf{Acc}(i, A) \to ([i, A]Qi \leftrightarrow \top)$ for $Qi \in \mathcal{AQ}$
(A3.2)	$\neg\mathsf{Acc}(i, A) \to ([i, A]Qi \leftrightarrow Qi)$ for $Qi \in \mathcal{AQ}$
(A3.3)	$[i, A]Qj \leftrightarrow Qj$ for $Qj \in \mathcal{AQ}$ and $i \neq j$
(A4)	$[i_1, A_1] \dots [i_k, A_k](F \to G) \leftrightarrow$ $([i_1, A_1] \dots [i_k, A_k]F \to [i_1, A_1] \dots [i_k, A_k]G)$
(A5.1)	$\mathsf{Acc}(i, A) \to ([i, A]\Box B \leftrightarrow \Box(A \to B))$
(A5.2)	$\neg\mathsf{Acc}(i, A) \to ([i, A]\Box B \leftrightarrow \Box B)$
(A6)	$[h_1, C_1] \dots [h_k, C_k][i, A][j, B]F \leftrightarrow$ $[h_1, C_1] \dots [h_k, C_k][j, A][i, B]F$ for $i \neq j$

Note that in (A6), we may choose k to be 0, in which case the axiom has the form $[i, A][j, B]F \leftrightarrow [j, A][i, B]F$ for $i \neq j$.

In order to formulate the rules of BCL, we need the following notation. Let $H(P)$ be a formula that may contain occurrences of the atomic proposition P. By $H(F)$, we denote the result of simultaneously replacing each occurrence of P in $H(P)$ with the formula F. The rules of BCL are:

$$(\mathsf{MP})\,\frac{F \qquad F \to G}{G} \qquad (\mathsf{NEC})\,\frac{A}{\Box A} \qquad (\mathsf{SUB})\,\frac{F \leftrightarrow G}{H(F) \leftrightarrow H(G)}$$

where (SUB) can only be applied if $H(F) \leftrightarrow H(G)$ is a compliant formula.

Remark 2. Our semantics includes infinite blockchains: in a given model $(\mathsf{I}, \mathsf{BC}, \mathsf{PU}, \mathsf{v})$, the sequence BC may have infinite length. If we want to exclude such models, then we have to add an infinitary rule

$$\frac{Qi \quad \text{for all } i \in \mathbb{N}^+}{\bot}$$

to BCL. This rule states that some Qi must be false, which means that BC has finite length.

4 Soundness

Before we can establish soundness of BCL, we have to show some preparatory lemmas.

Lemma 2. *Let* $\mathsf{M} := (\mathsf{I}, \mathsf{BC}, \langle\rangle, \mathsf{v})$ *be an initial model. Further let* $(\mathsf{I}, \mathsf{BC}', \mathsf{PU}', \mathsf{v}) := \mathsf{M}^{[i,A]}$ *for some block* $[i, A]$.

1. *If* $\mathsf{M} \models \mathsf{Acc}(i, A)$, *then* $\mathsf{BC}' = \mathsf{BC} \circ A$. *In particular, this yields* $\mathsf{len}(\mathsf{BC}') = i$ *and for each* j *with* $j \neq i$,

$$M \models Qj \quad \textit{if and only if} \quad \mathsf{M}^{[i,A]} \models Qj.$$

2. *If* $\mathsf{M} \not\models \mathsf{Acc}(i, A)$, *then* $\mathsf{BC}' = \mathsf{BC}$.

Lemma 3. *Each axiom of* BCL *is valid.*

Lemma 4. *Let* $\mathsf{M} = (\mathsf{I}, \mathsf{BC}, \mathsf{PU}, \mathsf{v})$ *be an arbitrary model and let* $[i, A]$ *be a block such that* $i > \mathsf{len}(\mathsf{BC}) + 1$. *Then we have* $\mathsf{M}^{[i,A]} = (\mathsf{I}, \mathsf{BC}, \mathsf{PU} \circ [i, A], \mathsf{v})$.

Lemma 5. *Let* $\mathsf{M} = (\mathsf{I}, \mathsf{BC}, \langle\rangle, \mathsf{v})$ *be an initial model and let* $[i, A]$ *be a block such that* $i \leq \mathsf{len}(\mathsf{BC}) + 1$. *Then* $\mathsf{M}^{[i,A]}$ *is an initial model, too.*

Lemma 6. *Let* $(\mathsf{I}, \mathsf{BC}, \mathsf{PU}, \mathsf{v})$ *be a model and* F *be an* $\mathcal{L}_{\mathsf{B}}$*-formula such that for each* $[i, A]$ *occurring in* F *we have* $i > \mathsf{len}(\mathsf{BC}) + 1$. *Then*

$$(\mathsf{I}, \mathsf{BC}, \mathsf{PU}, \mathsf{v}) \models F \quad \textit{if and only if} \quad (\mathsf{I}, \mathsf{BC}, \langle\rangle, \mathsf{v}) \models F.$$

Now we can show that the rule (SUB) preserves validity.

Lemma 7. *Let* $H(P), F, G$ *be* $\mathcal{L}_{\mathsf{B}}$*-formulas such that* $H(F) \leftrightarrow H(G)$ *is compliant. We have that*

$$\textit{if } F \leftrightarrow G \textit{ is valid, then } H(F) \leftrightarrow H(G) \textit{ is valid, too.}$$

We have established that the axioms of BCL are valid and that (SUB) preserves validity. It is easy to see that the rules (MP) and (NEC) also preseve validity. Soundness of BCL follows immediately.

Corollary 1. *For each formula F we have*

$$\vdash F \quad \textit{implies} \quad F \textit{ is valid.}$$

Remark 3. The reduction axiom (A3.3) does not hold in non-initial models. Indeed, let $\mathsf{M} := (\emptyset, \langle\rangle, \langle[2, \top]\rangle, \emptyset)$. We find that $\mathsf{M}^{[1,P]} = (\emptyset, \langle P, \top\rangle, \langle\rangle, \emptyset)$. Hence $\mathsf{M}^{[1,P]} \models Q2$, which is $\mathsf{M} \models [1, P]Q2$. But we also have $\mathsf{M} \not\models Q2$.

Remark 4. The above remark also implies that a block necessitation rule would not be sound, that is the validity of F does not entail the validity of $[i, A]F$. Indeed, the axiom $[1, P]Q2 \leftrightarrow Q2$ is valid; but the formula $[2, \top]([1, P]Q2 \leftrightarrow Q2)$ is not valid as shown in the previous remark.

Remark 5. The rule (SUB) would not preserve validity if we drop the condition that the conclusion must be compliant. Indeed, let us again consider the valid formula $[1, P]Q2 \leftrightarrow Q2$. Without the compliance condition, the rule (SUB) would derive $[2, P'][1, P]Q2 \leftrightarrow [2, P']Q2$, which is not a valid formula.

5 Normal form

Remember that a formula is compliant if the blockchain updates occur in the correct order. In this section, we establish a normal form theorem for our simple blockchain logic.

Definition 8. *A* base formula *is a formula that has one of the following forms (which include the case of no blockchain updates):*

1. $[i_1, A_1] \dots [i_m, A_m]\bot$
2. $[i_1, A_1] \dots [i_m, A_m]P$ *with* $P \in \mathcal{AP} \cup \mathcal{AQ}$
3. $[i_1, A_1] \dots [i_m, A_m]\Box B$

Formulas in normal form *are given as follows:*

1. *each compliant base formula is in normal form*
2. *if F and G are in normal form, then so is $F \to G$.*

Remark 6. As an immediate consequence of this definition, we obtain that for each formula F,

$$\text{if } F \text{ is in normal form, then } F \text{ is compliant.}$$

The following theorem states that for each formula, there is a provably equivalent formula in normal form. The proof is by induction on the structure of F.

Theorem 1. *For each $\mathcal{L}_{\mathsf{B}}$-formula F, there is an $\mathcal{L}_{\mathsf{B}}$-formula G in normal form such that $\vdash F \leftrightarrow G$.*

6 Completeness

We first show that BCL is complete for modal formulas. The modal language $\mathcal{L}_{\mathsf{M}}$ consists of all update-free $\mathcal{L}_{\mathsf{B}}$-formulas. Formally, $\mathcal{L}_{\mathsf{M}}$ is given by the following grammar

$$F ::= \bot \mid P \mid Q \mid F \to F \mid \Box A \quad ,$$

where $P \in \mathcal{AP}$, $Q \in \mathcal{AQ}$, and $A \in \mathcal{L}_{\mathsf{cl}}$.

We need the collection $\mathsf{BCL}^{\Box}$ of all BCL axioms that are given in $\mathcal{L}_{\mathsf{M}}$. The usual satisfaction relation for Kripke models is denoted by $\models_{\Box}$.

Lemma 8. *For each $\mathcal{L}_{\mathsf{M}}$-formula F we have*

$$F \text{ is valid} \quad \text{implies} \quad \vdash F.$$

We establish completeness for compliant formulas using a translation from compliant formulas to provably equivalent update-free formulas. We start with defining a mapping h that eliminates update operators.

Definition 9. *The mapping* h *from* $\{[i, A]F \mid F \in \mathcal{L}_{\mathsf{M}}\}$ *to* $\mathcal{L}_{\mathsf{M}}$ *is inductively defined by:*

$$\begin{aligned}
\mathsf{h}([i, A]\bot) &:= \bot \\
\mathsf{h}([i, A]P) &:= P \quad \text{for } P \in \mathcal{AP} \\
\mathsf{h}([i, A]Qi) &:= \mathsf{Acc}(i, A) \vee Qi \\
\mathsf{h}([i, A]Qj) &:= Qj \quad \text{for } Qj \in \mathcal{AQ} \text{ and } i \neq j \\
\mathsf{h}([i, A](F \to G)) &:= \mathsf{h}([i, A]F) \to \mathsf{h}([i, A]G) \\
\mathsf{h}([i, A]\Box B) &:= (\mathsf{Acc}(i, A) \wedge \Box(A \to B)) \vee (\neg\mathsf{Acc}(i, A) \wedge \Box B)
\end{aligned}$$

The mapping h corresponds to the reduction axioms of BCL. Thus it is easy to show the following lemma by induction on the structure of F.

Lemma 9. *Let F be an $\mathcal{L}_{\mathsf{B}}$-formula of the form $[i, A]G$ such that $G \in \mathcal{L}_{\mathsf{M}}$. We have that $\vdash F \leftrightarrow \mathsf{h}(F)$.*

We define a translation t from $\mathcal{L}_{\mathsf{B}}$ to $\mathcal{L}_{\mathsf{M}}$

Definition 10. *The mapping $\mathsf{t} : \mathcal{L}_{\mathsf{B}} \to \mathcal{L}_{\mathsf{M}}$ is inductively defined by:*

$$\begin{aligned} \mathsf{t}(\bot) &:= \bot \\ \mathsf{t}(P) &:= P \quad \text{for } P \in \mathcal{AP} \cup \mathcal{AQ} \\ \mathsf{t}(F \to G) &:= \mathsf{t}(F) \to \mathsf{t}(G) \\ \mathsf{t}(\Box A) &:= \Box A \\ \mathsf{t}([i, A]F) &:= \mathsf{h}([i, A]\mathsf{t}(F)) \end{aligned}$$

Lemma 10. *For each compliant formula F, we have*

$$\vdash F \leftrightarrow \mathsf{t}(F).$$

Theorem 2. *For each compliant $\mathcal{L}_{\mathsf{B}}$-formula F we have*

$$F \text{ is valid} \quad \text{implies} \quad \vdash F.$$

Combining Theorem 1 and Theorem 2 easily yields completeness for the full language.

Theorem 3. *For each $\mathcal{L}_{\mathsf{B}}$-formula F we have*

$$F \text{ is valid} \quad \text{implies} \quad \vdash F.$$

7 Conclusion

We have presented BCL, a dynamic logic to reason about updates in a simple blockchain model. Our semantics does not have the full complexity of the blockchains used in Bitcoin or Ethereum, yet it exhibits two key properties of blockchains: blockchain extensions must preserve consistency and blocks may be received in the wrong order. Note, however, that although receiving blocks in the wrong order is an important logical possibility, it only happens rarely in practice:

in the Bitcoin protocol the average generation time of a new block is 10 minutes; the average time until a node receives a block is only 6.5 seconds [6].

In order to illustrate the dynamics of our simple blockchain logic, we state some valid principles of BCL:

Persistence: $\Box A \to [i, B]\Box A$. Beliefs are persistent, i.e., receiving a new block cannot lead to a retraction of previous beliefs.

Consistency: $[i, B]\neg\Box\bot$. Receiving a new block cannot result in inconsistent beliefs.

Success: $\mathsf{Acc}(i, A) \to [i, A]\Box A$. If a block $[i, A]$ is acceptable, then A is believed after receiving $[i, A]$.[4]

Failure: $(Qi \vee \neg Q(i-1)) \to ([i, B]\Box A \leftrightarrow \Box A)$. If the current length of the blockchain is not $i - 1$, then receiving a block $[i, B]$ will not change the current beliefs.

There are several open issues for future work. Let us only mention two of them. Although blockchains are called *chains*, the data structure that is actually used is more tree-like and there are different options how to choose the valid branch: Bitcoin currently uses the branch that has the greastest proof-of-work effort invested in it [16] (for simplicity we can think of it as the longest branch); but recent research shows that the GHOST rule [18] (used, e.g., in Ethereum [21]) provides better security at higher transaction throughput. We plan to extend BCL so that it can handle tree-like structures and the corresponding forks of the chain. In particular, this requires some form of probability logic to model the fact that older transactions have smaller probability of being reversed [9, 16, 18].

In a multi-agent setting, each agent (node) has her own instance of a blockchain. Justification logics [2] could provide a formal approach to handle this. Evidence terms could represent blockchain instances and those instances can be seen as justifying the agents' knowledge about the accepted transactions. This approach would require to develop new dynamic justification logics [4, 17, 14]. Moreover, if the underlying blockchain model supports forks of the chain, then we need justification logics with probability operators [12].

[4] We call this prinicple *success*; but it is not related to the notion of a *successful formula* as studied in dynamic epistemic logic, see, e.g., [8].

References

1. Antonopoulos, A.M.: Mastering Bitcoin: Unlocking Digital Crypto-Currencies. O'Reilly Media, Inc. (2014)
2. Artemov, S.N.: Explicit provability and constructive semantics. Bulletin of Symbolic Logic 7(1), 1–36 (Mar 2001)
3. Brünnler, K., Flumini, D., Studer, T.: A logic of blockchain updates. E-print 1707.01766, arXiv.org (2017)
4. Bucheli, S., Kuznets, R., Studer, T.: Realizing public announcements by justifications. Journal of Computer and System Sciences 80(6), 1046–1066 (2014)
5. Buterin, V.: Ethereum: A next-generation smart contract and decentralized application platform (2013), https://github.com/ethereum/wiki/wiki/White-Paper, retrieved 2 Feb. 2017
6. Decker, C., Wattenhofer, R.: Information propagation in the Bitcoin network. In: 13th IEEE International Conference on Peer-to-Peer Computing. pp. 1–10 (2013)
7. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic, Synthese Library, vol. 337. Springer (2007)
8. van Ditmarsch, H., Kooi, B.: The secret of my success. Synthese 151(2), 201–232 (2006)
9. Grunspan, C., Pérez-Marco, R.: Double spend races. ArXiv e-prints 1702.02867 (2017)
10. Halpern, J.H., Rafael, P.: A knowledge-based analysis of the blockchain protocol. In: Lang, K. (ed.) TARK 2017. pp. 324–335. No. 251 in EPTCS (2017)
11. Herlihy, M., Moir, M.: Blockchains and the logic of accountability: Keynote address. In: LICS '16. pp. 27–30 (2016)
12. Kokkinis, I., Maksimović, P., Ognjanović, Z., Studer, T.: First steps towards probabilistic justification logic. Logic Journal of IGPL 23(4), 662–687 (2015)
13. Kooi, B.: Expressivity and completeness for public update logics via reduction axioms. Journal of Applied Non-Classical Logics 17(2), 231–253 (2007)
14. Kuznets, R., Studer, T.: Update as evidence: Belief expansion. In: Artemov, S.N., Nerode, A. (eds.) Logical Foundations of Computer Science, International Symposium, LFCS 2013, San Diego, CA, USA, January 6–8, 2013, Proceedings, Lecture Notes in Computer Science, vol. 7734, pp. 266–279. Springer (2013)
15. Lamport, L., Shostak, R., Pease, M.: The byzantine generals problem. ACM Trans. Program. Lang. Syst. 4(3), 382–401 (1982)
16. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2009)
17. Renne, B.: Public communication in justification logic. Journal of Logic and Computation 21(6), 1005–1034 (Dec 2011), published online July 2010
18. Sompolinsky, Y., Zohar, A.: Secure high-rate transaction processing in bitcoin. In: Böhme, R., Okamoto, T. (eds.) Financial Cryptography and Data Security 2015, Revised Selected Papers. pp. 507–527. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
19. Steiner, D.: A system for consistency preserving belief change. In: Artemov, S., Parikh, R. (eds.) Proceedings of Rationality and Knowledge. pp. 133–144. 18th ESSLLI, Association for Logic, Language and Information (2006)
20. Steiner, D., Studer, T.: Total public announcements. In: Artemov, S., Nerode, A. (eds.) LFCS 2007. LNCS, vol. 4514, pp. 498–511. Springer (2007)
21. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger, EIP-150 revision (2017), https://ethereum.github.io/yellowpaper/paper.pdf, retrieved 2 Feb. 2017