Blockchains as Kripke Models: an Analysis of Atomic Cross-Chain Swap

Yoichi Hirai[0000−0003−2076−2750]

Independent Scholar
i@yoichihirai.com

Abstract. There is a protocol called "atomic cross-chain swap" that spans across multiple blockchains, but is it really atomic? We analyze the protocol using a modal logic for asynchronous communication. The modal logic allows us to identify some assumptions required for the "atomic" property as logical formulas. We first demonstrate that the atomicity fails without some temporal-epistemic assumptions. We further construct a proof that the atomicity holds with strong enough temporal-epistemic assumptions. In both analyses, we use Kripke models of the modal logic. This is the first analysis of multiple blockchains' interaction using a modal logic.

Keywords: modal logic, epistemic logic, asynchronous computation, blockchains

1 Introduction

This paper analyzes a concurrent, asynchronous protocol involving multiple blockchains using a modal logic called intuitionistic epistemic logic for asynchronous communication [14].

A blockchain is a singly linked list of data-blobs called blocks. A block contains a cryptographic hash value of the previous block's contents. The hash value serves as the link of the list. A single blockchain is useful for ensuring the integrity of a sequence of blocks because the latest block uniquely identifies the preceding sequence of blocks (assuming no hash collisions).

Today, some proof-of-work protocols (Bitcoin [19] and similar protocols) are spinning concurrent, asynchronous activities into fully sequential histories. These protocols have no fixed number of participants, so the protocols fall out of the traditional distributed protocols (e.g. Paxos [15] and Chandra-Toueg [3]). These blockchain protocols have no termination; they never yield final definite consensus but at most an ever-increasing confidence on one result. This is why these blockchain protocols fall out of scope of some impossibility theorems (e.g. FLP-theorem [9]) regarding distributed consensus. It turns out many people are willing to use such protocols without termination.

Once we start using multiple blockchains for performance reasons (as in Polkadot [25], Plasma [22], or "sharding" aproaches [10,18]), the asynchronous, concurrent reasoning is again required. As an example, this paper analyzes a protocol called “atomic cross-chain swap” [16,21]. Fig. 1 is a concise description of the protocol. In this protocol, two blockchains interact in an asynchronous manner, and they are claimed to establish an atomic swap together. The atomic swap is claimed to either succeed in both blockchains or fail in both.

Such atomicity between asynchronically communicating agents sound dubious to a student of modal epistemic logic, who learns that asynchronous communication never creates a new piece of common knowledge [4]. An atomic swap, if it is atomic as the name suggests, should result in common knowledge. This is because, if the swap atomically succeeds or fails, the result should be known to both parties, and there is no possibility that two parties see different situations, including their epistemic states. As a result, there should be an unlimited nesting of mutual knowledge of the form “X knows Y knows X knows Y knows $\cdots$ that the swap failed.” However, in the asynchronous setting, deep nesting of knowledge can be attained only after as many round-trip communication. Since the cross-chain atomic swap does not involve synchronous communication between the two blockchains, there must be some kind of assumptions supporting the “atomic” property. This paper clarifies those assumptions.

Our contributions are:

– defining the syntax (Subsect. 2.1) and the semantics (Subsect. 2.2) of logical formulas for reasoning about the cross-chain atomic swap,
– specifying hashlocks with logical formulas (Sect. 3),
– specifying desired atomicity of cross-chain atomic swaps as logical formulas (Binary-Outcome) and (Weak-Binary-Outcome) in Sect. 4,
– identifying two sets of assumptions that are not enough for the desired atomicity (Prop. 3 and Prop. 4), and
– identifying one set of assumptions that is enough for a form of atomicity (Prop. 5).

2 The Logic Used in this Paper

Our task is an instance of the general task of ensuring a desired property (in our case, atomicity) in all possible situations (in our case, protocol executions). If some situations refute the property, we can continue asking if we can restrict the possible situations to regain the desired property. We need a mathematical formalism to express the possible situations, our desired properties, and our assumptions on the possible situations.

We use the approach of mathematical logic [5]. The possible situations are represented as models. Each model contains states that are related temporally or epistemically. Logical formulas express properties of states of models. The syntax of the logic defines what sequence of symbols counts as logical formulas. The semantics of the logic defines which states of models satisfy a logical formula.

We use an extension of intuitionistic propositional logic. Intuitionistic logic originally modeled a mathematician who happens to be an intuitionist. When a proposition φ is known to hold, it holds forever. When the negation $\neg\varphi$ is known, that would also be remembered forever. When neither φ or $\neg\varphi$ is known, The canonical bitcoin atomic swap works as follows: Alice prepares a random secret `k` with a 20-byte hash `H=HASH160(k)` and then funds the following contract, in Bitcoin script [2]:

```
IF
  <BKey> CHECKSIGVERIFY
  HASH160 <H> EQUAL
ELSE
  <AKey> CHECKSIGVERIFY
  <ATime> CHECKLOCKTIMEVERIFY
ENDIF
```

Meaning of this contract: A signature from Bob's public key `BKey` in combination with secret `k` can spend the money. However as a fallback in case of cancellation, a signature from Alice's public key `AKey` lets her get a refund, but only after `ATime`. Bob does not know `k`, yet. He funds a similar contract on his blockchain using Alice's `H` value, but with a `BTime` expiring significantly sooner than `ATime`:

```
IF
  <AKey> CHECKSIGVERIFY
  HASH160 <H> EQUAL
ELSE
  <BKey> CHECKSIGVERIFY
  <BTime> CHECKLOCKTIMEVERIFY
ENDIF
```

Now, Alice may redeem Bob's contract but in doing so, she must reveal `k`. Bob can now see `k` which allows him to redeem Alice's contract. If the deal is called off then Bob is allowed to get a refund at `BTime`, and then Alice can get her refund after `ATime`. The above setup is generally perceived as perfectly secure and is being proposed for safe on-chain cryptocurrency exchanges that do not involve a third party.

Fig. 1. A description of a cross-chain atomic swap, cited from [16] with cosmetic modifications.

according to the intuitionistic interpretation of disjunction $\vee$, the disjunction $\varphi \vee \neg\varphi$ is not known. In other words, whenever $\varphi \vee \psi$ is known, either φ is known, or ψ is known.

This intuitionistic reading of disjunction seems particularly useful for settlement of funds when φ means "Alice obtains the fund" and ψ means "Bob obtains the fund." The settlement should be final, and the finality is already captured by the persistent nature of intuitionistic logic.

Intuitionistic epistemic logic for asynchronous communication. Modal logic can express the fundamental assumptions about knowledge and time. For instance, the formula $K_A\varphi \supset K_A(K_A\varphi)$ says "if A knows φ, A knows that A knows φ". This formula is named "positive introspection." At first logical formulas look like merely a shorthand for sentences, but the symbolic treatments scale better than English sentences, especially when the modalities are nested.

Intuitionistic epistemic logic [14] was designed to reason about asynchronous communication. The logic can reason about temporal epistemic systems, but it has no explicit temporal modality. The Kripke model [5] of intuitionistic propositional logic is reused as the temporal frame. Originally Hirai [14] used the logic for waitfree communication on shared memory. In this paper we use the logic for asynchronous communication between multiple blockchains.

2.1 Language

Mathematical logic distinguishes syntax and semantics. Logical formulas themselves are just shapes without meaning. A separate criterion dictates when a model satisfies a formula. Of course, certain formulas are never satisfied, and these formulas represent falsehood. However, such interpretations come only after the definition (Def. 2) of semantics.

We first define which sequents of symbols count as logical formulas. The logical formulas contain names of agents and atomic propositions, so we define those first.

An *agent* is one of the four distinct symbols:

$$a ::= \mathrm{Alice}, \mathrm{Bob}, \mathrm{X}, \mathrm{Y}. \tag{1}$$

They are just distinct symbols, but informally, Alice and Bob are participants of the protocol in Fig. 1, and X and Y are blockchains.

We choose the following set of *atomic propositions*:

$$P ::= \mathrm{D}_1, \mathrm{D}_2, \mathtt{k}, \mathrm{A}_\mathrm{Y}, \mathrm{B}_\mathrm{X}. \tag{2}$$

Informally, D_1 holds whenever the wall clock shows more than one day ahead since the beginning of the protocol execution, and D_2 holds whenever more than two days. Also informally, $\mathtt{k}$ holds when Alice's secret is publicly visible. A_Y holds when the fund on blockchain Y is available to Alice. B_X when the fund on blockchain X is available to Bob. By introducing the last three atomic formulas, we are effectively assuming that the swaps on blockchains X and Y cannot be reversed. In practice, agents are supposed to ignore contents of too fresh blocks that might be orphaned[1].

Following Hirai [14], a *formula* is syntactically defined as

$$\varphi, \psi ::= \bot \mid P \mid (K_a\varphi) \mid (\varphi \vee \psi) \mid (\varphi \wedge \psi) \mid (\varphi \supset \psi). \qquad (3)$$

where symbols a and P are non-terminal symbols from (1) and (2). The symbol $\supset$ stands for implication; a formula of the form $(\varphi \supset \psi)$ says φ implies ψ. Negation $(\neg\varphi)$ is a shorthand for $(\varphi \supset \bot)$. We omit parentheses when there is no ambiguity.

Informal readings of the language. BHK-interpretation[2] [24, Ch. 1] is a proof-centric way of reading logical connectives ($\wedge$, $\vee$, $\supset$, $\bot$ and K_a). If one knows what counts as a proof of φ and what counts as a proof of ψ, BHK-interpretation tells what counts as a proof of more complicated formulas: $\varphi \wedge \psi$, $\varphi \vee \psi$ and $\varphi \supset \psi$.

H1 A proof of $\varphi \wedge \psi$ is given by presenting a proof of φ and a proof of ψ.
H2 A proof of $\varphi \vee \psi$ is given by presenting either a proof of φ or a proof of ψ (plus the stipulation that we want to regard the proof presented as evidence for $\varphi \vee \psi$).
H3 A proof of $\varphi \supset \psi$ is a construction which permits us to transform any proof of φ into a proof of ψ.
H4 Absurdity $\bot$ (contradiction) has no proof; a proof of $\neg\varphi$ is a construction which transforms any hypothetical proof of φ into a proof of a contradiction.

Hirai [13] extends the list with one clause about the epistemic modality:

HK A proof of $K_a\varphi$ is a construction that witnesses agent a's acknowledgment of a proof of φ and also contains the acknowledged proof.

In other words, a proof of $K_a\varphi$ is a proof of φ with a's signature. From a signed proof of φ, one can obtain an unsigned proof of φ, so, the formula $(K_a\varphi) \supset \varphi$ is always satisfied, as we see later in Prop. 7.

The BHK-interpretation explains the logical connectives. We have to interpret the atomic formulas so that we know what count as proofs of the atomic formulas. A proof of `k` is the secret generated by Alice shown in public. A proof of D_1 and D_2 could be some real-world information only available one day or two days after a certain point in time. A proof of A_Y is an onchain proof that Alice can spend the fund on blockchain Y. A proof of B_X is an onchain proof that Bob can spend the fund on blockchain X. If the cross-chain atomic swap is really atomic, A_Y and B_X both should hold or neither (Sect. 4).

[1] A block is orphaned when it belongs to a blockchain that is not considered canonical anymore. This sometimes happens after branching blockchains are formed.
[2] BHK stands for Brouwer-Heyting-Kolmogorov.

2.2 Models

A model (Def. 1) is a set of states equipped with some relations and functions. We will be using small, finite models to refute the desired atomicity of the protocol (Props. 3 and 4). We will also be reasoning about arbitrary models to establish the atomicity (Prop. 5).

Definition 1 (Def. 2.3, [14]). *Let A denote the set of agents. A model $\langle W, \prec, (f_a)_{a \in A}, \rho \rangle$ is a tuple with following properties:*

1. *$\langle W, \preceq \rangle$ is a partially ordered set whose elements are called states,*
2. *for each agent $a \in A$, a function $f_a \colon W \to W$ satisfies*
   (a) $f_a(w) \preceq w$,
   (b) $f_a(f_a(w)) = f_a(w)$, and
   (c) $w \preceq v$ implies $f_a(w) \preceq f_a(v)$
3. *Let* Atom *be the set of atomic propositions and $\mathcal{P}(W)$ is the powerset of W. $\rho\colon \mathrm{Atom} \to \mathcal{P}(W)$ is a function such that each $\rho(P)$ is upward-closed with respect to $\preceq$. In other words, $w' \succeq w \in \rho(P)$ implies $w' \in \rho(P)$.*

Definition 2. *We define a relation $M, w \models \varphi$ (pronounced "the model M at state w satisfies φ") of a model $M = \langle W, \preceq, (f_a)_{a \in A}, \rho \rangle$, a state $w \in W$ and a formula φ. The definition is inductive on the structure of φ.*

(Case $\varphi = \bot$) *$M, w \models \bot$ never holds.*
(Case $\varphi = P$) *for an atomic formula P, $M, w \models P$ if and only if $w \in \rho(P)$.*
(Case $\varphi = K_a\psi$) *$M, w \models K_a\psi$ if and only if $M, f_a(w) \models \psi$.*
(Case $\varphi = \psi_0 \wedge \psi_1$) *$M, w \models \psi_0 \wedge \psi_1$ if and only if both $M, w \models \psi_0$ and $M, w \models \psi_1$ hold.*
(Case $\varphi = \psi_0 \vee \psi_1$) *$M, w \models \psi_0 \vee \psi_1$ if and only if $M, w \models \psi_0$ or $M, w \models \psi_1$ holds.*
(Case $\varphi = \psi_0 \supset \psi_1$) *$M, w \models \psi_0 \supset \psi_1$ if and only if for any $w' \in W$ with $w' \succeq w$, the relation $M, w' \models \psi_0$ implies the relation $M, w' \models \psi_1$.*

Since $\neg\varphi$ is an abbreviation of $\varphi \supset \bot$, the relation $M, w \models \neg\varphi$ holds if and only if no $v \succeq w$ satisfies $M, v \models \varphi$.

Informal interpretation of the model. When a state w satisfies a proposition φ, a proof of φ is available in the state. When two states are ordered $v \preceq w$, they are temporarily related. Every proof available in the past state v is also available in the future state w. So, any formula satisfied in v is also satisfied in w (Prop. 6). Such monotonicity is not found in the real world, where people can forget and proofs can be lost. When we analyze cryptographic protocols, it is prudent to assume that attackers do not forget a once-learned secret. The treatment also has shortcomings. Most importantly, our analysis assumes the finality of transactions on blockchains. Nonetheless we will find that more assumptions are necessary.

The state $f_a(w)$ is agent a's latest state seen from w. Every proof in $f_a(w)$ are available in w. Moreover, the proofs available in $f_a(w)$ are all signed by a and made available in w, so, if $f_a(w)$ contains a proof of φ, w contains a proof of

$K_a\varphi$. Such situation typically occurs when a sends a message from state $f_a(w)$ and the message arrives in state w. We assume such messages contain all current knowledge of a at $f_a(w)$.

We consider any $f_a(w)$ as a local state of a and thus $f_a(f_a(w))$ is always equal to $f_a(w)$. As a result, $K_a\varphi$ and $K_aK_a\varphi$ are always equivalent (so the property "positive introspection" holds.

Our models can distinguish (a) asynchronous round-trip between Alice and Bob from (b) synchronous communication between them (Fig. 2). In the asynchronous case, if $f_{\text{Alice}}f_{\text{Bob}}f_{\text{Alice}}(v)$ satisfies φ, v satisfies $K_{\text{Alice}}K_{\text{Bob}}K_{\text{Alice}}\varphi$ but not necessarily $K_{\text{Alice}}K_{\text{Bob}}K_{\text{Alice}}K_{\text{Bob}}\varphi$. In the synchronous case, if w satisfies φ, w also satisfies $K_{\text{Alice}}K_{\text{Bob}}\cdots K_{\text{Bob}}\varphi$ with any repetition of K_{Alice} and K_{Bob}.

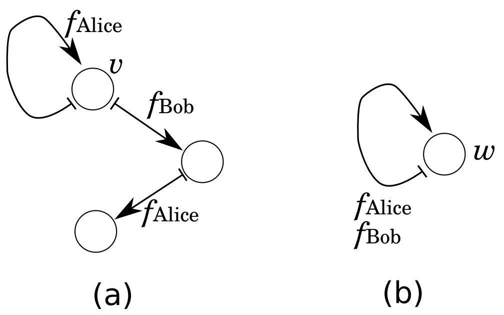

Fig. 2. Kripke models showing (a) asynchronous round-trip and (b) synchronous communication between Alice and Bob. Circles represent the states of models. In (a), the state v contains Bob's message from $f_{\text{Bob}}(v)$, which in turn contains Alice's message from $f_{\text{Alice}}f_{\text{Bob}}(v)$.

3 Common Assumptions

3.1 For the Hashlock on Blockchain X

The cross-chain atomic swap (Fig. 1) is a protocol based on a primitive called "a hashlock." We need to specify the hashlocks using logical formulas.

The most relevant property is a hashlock's ability to settle payments. After a finite amount of time, a hashlock is able to dictate whether the locked fund already belongs to Bob or never. Concretely on blockchain X, after two days, either Bob has obtained the fund and the secret has been revealed ($\mathrm{B}_{\mathrm{X}} \wedge \mathtt{k}$) or Bob will never get the fund ($\neg\mathrm{B}_{\mathrm{X}}$).

$$K_{\mathrm{X}}(\mathrm{D}_2 \supset ((\mathrm{B}_{\mathrm{X}} \wedge \mathtt{k}) \vee \neg\mathrm{B}_{\mathrm{X}})). \qquad \text{(X-live1)}$$

A hashlock can be unlocked using a secret. On blockchain X, if two days have not passed yet, if Bob provides the secret ($K_{\text{Bob}}\mathtt{k}$), Bob obtains the fund (B_{X}).

$$K_{\mathrm{X}}(\mathrm{D}_2 \vee (K_{\text{Bob}}\mathtt{k} \supset \mathrm{B}_{\mathrm{X}})). \qquad \text{(X-live2)}$$

Here, we could not say $(\neg D_2) \supset \cdots$ because that formula is of any relevance only at a state with no future satisfying D_2. The formula $\neg D_2$ is satisfied only when the world ends before two days pass.

The above two properties are about what a hashlock is supposed to do. There is one important thing that a hashlock is not supposed to do. The hashlock grants the fund to Bob only when Bob authenticates himself with the secret `k`:

$$\mathrm{B_X} \supset K_{\mathrm{X}} K_{\mathrm{Bob}} \mathtt{k}. \qquad \text{(X-safe)}$$

3.2 For the Hashlock on Blockchain Y

We can formulate the same properties of the hashlock on the other blockchain. After one day, on blockchain Y, either the fund is given or not given:

$$K_{\mathrm{Y}}(\mathrm{D_1} \supset ((\mathrm{A_Y} \wedge \mathtt{k}) \vee \neg \mathrm{A_Y})). \qquad \text{(Y-live1)}$$

Before one day, on blockchain Y, if the hash is known, the fund is given:

$$K_{\mathrm{Y}}(\mathrm{D_1} \vee (K_{\mathrm{Alice}} \mathtt{k} \supset \mathrm{A_Y})). \qquad \text{(Y-live2)}$$

Alice cannot obtain the fund on the blockchain Y unless she reveals the hash:

$$\mathrm{A_Y} \supset K_{\mathrm{Y}} \mathtt{k}. \qquad \text{(Y-safe)}$$

3.3 For the temporal ordering of day one and day two

If two days have passed, one day has already passed, too:

$$\mathrm{D_2} \supset \mathrm{D_1}. \qquad \text{(Days)}$$

4 Reasoning about Atomicity

4.1 A Failure on Binary Outcome

In general, atomicity states that the protocol cleanly succeeds or fails, without leaving an incomplete success. In our case, the two unlocking events on blockchains X and Y need to succeed both or fail both. One way to express this as a logical formula goes like this; after two days, at least one of the two allowed cases happens:

$$\mathrm{D_2} \supset ((\mathrm{A_Y} \wedge \mathrm{B_X}) \vee ((\neg \mathrm{A_Y}) \wedge (\neg \mathrm{B_X}))). \qquad \text{(Binary-Outcome)}$$

However, the already introduced axioms do not guarantee (Binary-Outcome).

Proposition 3 *There is a model that satisfies all of (Y-live1), (Y-live2), (Y-safe), (X-live1), (X-live2), (X-safe) and (Days) at every state, but does not satisfy (Binary-Outcome) at a state.*

Proof. By constructing a model M and a state w (Fig. 3) so that M satisfies all assumptions at every state but w does not satisfy (Binary-Outcome). The state w in Fig. 3 does not satisfy $\neg \mathrm{A_Y}$ because there is a future state satisfying $\mathrm{A_Y}$. On the other hand, w does not satisfy $\mathrm{A_Y}$ either. So, without looking at $\mathrm{B_X}$ or $\neg\mathrm{B_X}$, we can conclude that w does not satisfy $(\mathrm{A_Y} \wedge \mathrm{B_X}) \vee ((\neg \mathrm{A_Y}) \wedge (\neg \mathrm{B_X}))$. However, w does satisfy $\mathrm{D_2}$. So w does not satisfy the implication (Binary-Outcome). □

Informally speaking, on state w in Fig. 3, two days have passed but neither blockchain has produced visible blocks since the beginning of the protocol.

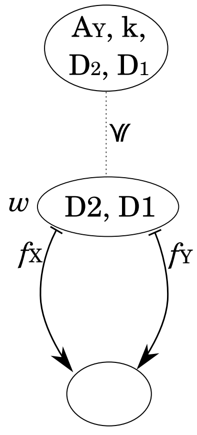

Fig. 3. A model M and a state w for the proof of Prop. 3. Three circles represent the three states of M. Each function f_a is identity whenever not explicitly shown. The $\preceq$ relation holds whenever two states are connected through dashed lines and arrows in a bottom-to-top way.

4.2 A Failure on a Weaker Binary Outcome

We can require that both blockchains contain blocks produced after two days have passed:

$$K_{\mathrm{X}}\mathrm{D}_2 \supset (K_{\mathrm{Y}}\mathrm{D}_2 \supset ((\mathrm{A_Y} \wedge \mathrm{B_X}) \vee ((\neg \mathrm{A_Y}) \wedge \neg \mathrm{B_X}))). \quad \text{(Weak-Binary-Outcome)}$$

This new proposition is strictly weaker than the old one. All states satisfying (Binary-Outcome) also satisfy (Weak-Binary-Outcome), but the inverse is not always the case. For instance, state w in Fig. 3 does not satisfy (Binary-Outcome), but it satisfies (Weak-Binary-Outcome).

Proposition 4 *There is a model that satisfies (Y-live1), (Y-live2), (Y-safe), (X-live1), (X-live2), (X-safe) and (Days) at all states, but does not satisfy (Weak-Binary-Outcome) at some states.*

Proof. By constructing a model M and a state w in it (Fig. 4). □

Fig. 4 demonstrates a lack of communication between the two chains. More specifically, although the hashlock on blockchain Y is unlocked, Bob fails to use the secret revealed on blockchain Y to unlock the hashlock on blockchain X.

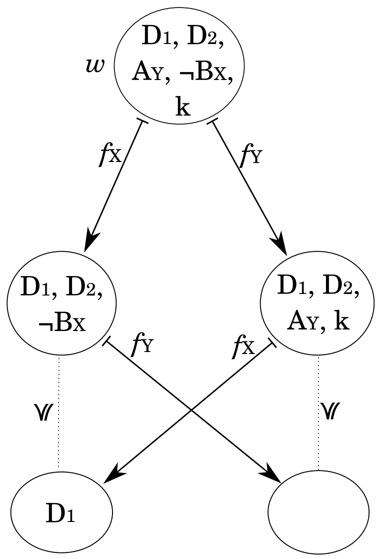

Fig. 4. A model M and a state w for proving Prop. 4. Five circles represent the five states of M. Each function f_a is identity whenever not explicitly shown. The $\preceq$ relation holds whenever two states are reachable following dashed lines and arrows in a bottom-to-top way.

4.3 Enough Assumptions for Atomicity

To remedy the situation, we need to assume certain communication between the two chains. Especially, contents on blockchain Y should be read by Bob and transmitted over to blockchain X in a timely manner.

In order to talk about the timing restrictions, we add two more agents in the language: $1\frac{1}{4}$ and $1\frac{1}{2}$ that represent "$1\frac{1}{4}$ (resp. $1\frac{1}{2}$) days from the beginning of protocol execution." In the models, $f_{1\frac{1}{2}}(w)$ is equal to w when the wall-clock time at w is less than one-and-half days from the beginning of protocol execution. Otherwise, $f_{1\frac{1}{2}}(w)$ is a previous state where the wall-clock time is less than one-and-half days from the beginning.

Now we can spell out assumptions; whenever blockchain Y has a record at one-and-quarter days, Bob should have read and submitted the record to blockchain X by one-and-half days:

$$(K_{\mathrm{Y}}K_{1\frac{1}{4}}\varphi) \supset K_{\mathrm{X}}K_{1\frac{1}{2}}K_{\mathrm{Bob}}K_{\mathrm{Y}}K_{1\frac{1}{4}}\varphi. \qquad \text{(Bob-has-chance)}$$

We have defined a set of infinitely many logical formulas where φ is replaced with arbitrary logical formulas.

When blockchain Y contains records at the two-day moment, it also contains witnesses from the one-and-quarter-day moment, saying that the hashlock had already been settled; either Alice had used the secret to unlock the hashock, or Alice would never unlock it:

$$K_{\mathrm{Y}}(\mathrm{D}_2 \supset K_{1\frac{1}{4}}((\mathrm{A}_{\mathrm{Y}} \wedge \mathtt{k}) \vee (\neg \mathrm{A}_{\mathrm{Y}}))). \tag{Y-timed1}$$

Finally, if Bob ever gets to know the secret, Alice should have opened the hashlock. In other words, Alice does not leak the secret without getting the fund in blockchain Y:

$$K_{\mathrm{Bob}}\mathtt{k} \supset \mathrm{A}_{\mathrm{Y}}. \tag{Alice-opsec}$$

Blockchain X at one-and-half days should allow Bob to unlock the hashlock:

$$K_{\mathrm{X}} K_{1\frac{1}{2}}((K_{\mathrm{Bob}}\mathtt{k}) \supset \mathrm{B}_{\mathrm{X}}). \tag{X-live$1\frac{1}{2}$}$$

When we impose those formulas at every state, the desired weak binary outcome property holds.

Proposition 5 *If a model M satisfies (X-live2), (Y-timed1), (Alice-opsec), (Bob-has-chance), (X-live$1\frac{1}{2}$) at every state, M also satisfies (Weak-Binary-Outcome) at every state.*

Before proving this proposition, we need some preparations.

Proposition 6 (Kripke monotinicity [14]) *$M, w \models \varphi$ and $w \preceq w'$ imply $M, w' \models \varphi$.*

Proof. By structural induciton on φ. □

Proposition 7 *Any model M at any state w satisfies any $(K_a\varphi) \supset \varphi$.*

Proof. For any w' with $w' \succeq w$, we assume $M, w' \models K_a\varphi$ and claim $M, w' \models \varphi$. By the semantics of K_a, $f_a(w')$ satisfies φ. By the definition of a model, $f_a(w') \preceq w'$ holds. By Prop. 6, w' satisfies φ. □

With these two auxiliary propositions, we are ready to continue.

Proof (of Prop. 5). We take an arbitrary state v in such a model M. And we take an arbitrary state w with $w \succeq v$. We assume $M, w \models K_{\mathrm{X}}\mathrm{D}_2$ and $M, w \models K_{\mathrm{Y}}\mathrm{D}_2$. It is enough to show that w satisfies $(\mathrm{A}_{\mathrm{Y}} \wedge \mathrm{B}_{\mathrm{X}}) \vee ((\neg \mathrm{A}_{\mathrm{Y}}) \wedge (\neg \mathrm{B}_{\mathrm{X}}))$.

Since w satisfies $K_{\mathrm{Y}}\mathrm{D}_2$, $f_{\mathrm{Y}}(w)$ satisfies D_2. Since w satisfies (Y-timed1), $f_{\mathrm{Y}}(w)$ satisfies $\mathrm{D}_2 \supset K_{1\frac{1}{4}}((\mathrm{A}_{\mathrm{Y}} \wedge \mathtt{k}) \vee (\neg \mathrm{A}_{\mathrm{Y}}))$. So $f_{\mathrm{Y}}(w)$ satisfies $K_{1\frac{1}{4}}((\mathrm{A}_{\mathrm{Y}} \wedge \mathtt{k}) \vee (\neg \mathrm{A}_{\mathrm{Y}}))$. That is to say w satisfies $K_{\mathrm{Y}} K_{1\frac{1}{4}}((\mathrm{A}_{\mathrm{Y}} \wedge \mathtt{k}) \vee (\neg \mathrm{A}_{\mathrm{Y}}))$. By (Bob-has-chance), w also satisfies $K_{\mathrm{X}} K_{1\frac{1}{2}} K_{\mathrm{Bob}} K_{\mathrm{Y}} K_{1\frac{1}{4}}((\mathrm{A}_{\mathrm{Y}} \wedge \mathtt{k}) \vee (\neg \mathrm{A}_{\mathrm{Y}}))$. In other words, $f_{1\frac{1}{4}} f_{\mathrm{Y}} f_{\mathrm{Bob}} f_{1\frac{1}{2}} f_{\mathrm{X}}(w)$ satisfies $\mathrm{A}_{\mathrm{Y}} \wedge \mathtt{k}$ (the positive case) or $\neg \mathrm{A}_{\mathrm{Y}}$ (the negative case).

(The positive case) By (X-live$1\frac{1}{2}$), w satisfies $K_{\mathrm{X}} K_{1\frac{1}{2}}((K_{\mathrm{Bob}}\mathtt{k}) \supset \mathrm{B}_{\mathrm{X}})$. In other words, $f_{1\frac{1}{2}} f_{\mathrm{X}}(w)$ satisfies $(K_{\mathrm{Bob}}\mathtt{k}) \supset \mathrm{B}_{\mathrm{X}}$. Since $f_{\mathrm{Bob}} f_{1\frac{1}{2}} f_{\mathrm{X}}(w)$ is a future of $f_{1\frac{1}{4}} f_{\mathrm{Y}} f_{\mathrm{Bob}} f_{1\frac{1}{2}} f_{\mathrm{X}}(w)$, by Kripke monotonicity, $f_{\mathrm{Bob}} f_{1\frac{1}{2}} f_{\mathrm{X}}(w)$ satisfies $\mathrm{A}_{\mathrm{Y}} \wedge \mathtt{k}$. So, $f_{1\frac{1}{2}} f_{\mathrm{X}}(w)$ satisfies $K_{\mathrm{Bob}}\mathtt{k}$. The same state also satisfies $(K_{\mathrm{Bob}}\mathtt{k}) \supset \mathrm{B}_{\mathrm{X}}$. As a result, $f_{1\frac{1}{2}} f_{\mathrm{X}}(w)$ satisfies B_{X}. By Kripke monotonicity, w satisfies B_{X}. Also by Kripke monotonicity, w satisfies A_{Y}.

(The negative case) Since $f_{1\frac{1}{2}} f_{\mathrm{X}}(w) \succeq f_{1\frac{1}{4}} f_{\mathrm{Y}} f_{\mathrm{Bob}} f_{1\frac{1}{2}} f_{\mathrm{X}}(w)$, by Kripke monotonicity, $f_{1\frac{1}{2}} f_{\mathrm{X}}(w)$ also satisfies $\neg\mathrm{A}_{\mathrm{Y}}$. Since $f_{1\frac{1}{2}} f_{\mathrm{X}}(w)$ satisfies (Alice-opsec), by the semantics of $\supset$, $f_{1\frac{1}{2}} f_{\mathrm{X}}(w)$ satisfies $\neg K_{\mathrm{Bob}}\mathtt{k}$. Since $f_{\mathrm{X}}(w) \succeq f_{1\frac{1}{2}} f_{\mathrm{X}}(w)$, by Kripke monotonicity, $f_{\mathrm{X}}(w)$ also satisfies $\neg K_{\mathrm{Bob}}\mathtt{k}$. We claim that w satisfies $\neg\mathrm{B}_{\mathrm{X}}$. For that, seeking contradiction, We assume some $x \succeq w$ satisfies B_{X}. State x satisfies (X-safe), so by the semantics of $\supset$, x also satisfies $K_{\mathrm{X}} K_{\mathrm{Bob}}\mathtt{k}$. In other words, $f_{\mathrm{X}}(x)$ satisfies $K_{\mathrm{Bob}}\mathtt{k}$. However, since $f_{\mathrm{X}}(x) \succeq f_{\mathrm{X}}(w)$, this contradicts $f_{\mathrm{X}}(w)$ satisfying $\neg K_{\mathrm{Bob}}\mathtt{k}$. □

Informal reading of the result. We have identified a set of sufficient assumptions that supports the atomicity. We should now reflect on the meaning of these logical formulas, but before that we have to evaluate our choice of the logic. When we modeled the cross-chain atomic swap using intuitionistic epistemic logic, before introducing any axioms, we effectively assumed that all propositions are stable. That is, once they become true, they remain true forever. This stability seems appropriate for the revealed secret `k`, but questionable for heads of blockchains. For Bitcoin, the head of the chain sometimes jumps to other branches. We discuss approaches to circumvent this problem in Sect. 6.

(Bob-has-chance) is about Bob's ability to read from the blockchain Y and submit the obtained knowledge to the blockchain X, which in turn requires availability of both blockchains. (Alice-opsec) is about Alice's ability to keep the secret when she chooses not to unlock the hashlock. This also requires pre-image resistance of the hash function. The assumptions (Y-timed1) and (X-live2) are about the behavior of the onchain scripts. These two assumptions need to be backed by program analysis. A formal notation of onchain programs like SoK [1] might ease such program analysis.

5 Related Work

Emerson and Clarke [7] already regarded Kripke models as states of communicating processes. They proposed a method for automatically generating finite state machines that represent states of communicating processes. They describe a procedure to decide whether such finite state machines exist for a specification.

Smart contract verification. Luu et al. [17] define a lightweight semantics of Ethereum. Nikolic et al. [20] use the same semantics to capture bugs spanning multiple Ethereum transactions. Sergey and Aquinas [23] gave an insight that concurrent reasoning applies to Ethereum contracts' interactions. None of these works treats the interaction of multiple blockchains.

Halpern and Pass's knowledge-based analysis on the epistemic property of a blockchain. Halpern and Pass [12] also analyze the communication ability of a blockchain using a modal epistemic logic. The reasoning framework of Halpern and Pass is based on the tradition of model epistemic logic, admittedly more faithfully than our work is, because their "runs and systems" [8] model has been very popular among computer scientists.

Their focus of attention is different from ours. Halpern and Pass [12] look at a blockchain as a communication medium between agents, and propose a weak form of common knowledge that the blockchain provides. To seek a form of common knowledge, their analysis needs to consider entering and leaving agents. Our analysis never required a global view of all agents.

The difference seems to come from different roles of blockchains. Halpern and Pass seem to regard a blockchain as a mechanism for public attestation of valid contracts between agents. In our analysis, Alice is never interested in Bob's knowledge, or vice versa. The formulas in our analysis do not involve nesting of epistemic modalities like $K_{\mathrm{Alice}}K_{\mathrm{Bob}}\cdots$. The participants are interested to see value transfers recorded on the finalized blocks, but not interested in whether other agents have seen those blocks.

We analyze a situation where multiple blockchains are involved, and, we have identified some synchrony conditions for specific agents (Bob-has-chance), (Y-timed1), (X-live$1\frac{1}{2}$) that are useful for a specific protocol, while Halpern and Pass assume a global parameter to limit delays of all messages.

Hirai's intuitionistic epistemic logic. Hirai [14] characterized waitfree communication over sequentially consistent shared memory using intuitionistic epistemic logic. That work targetted shared memory multi-thread computation. There, a well-known property called sequential consistency was represented as an axiom type[3]. In this paper, we are figuring out desired properties out of Kripke models that reveal missing assumptions (Props. 3 and 4). Modal logics are useful not only for explaining known properties but for identifying unspecified requirements.

Gleissenthall and Rybalchenko [11] use a more expressive logic with separate temporal modalities and epistemic modalities for characterizing sequential consistency, linearizability and eventual consistency. Their logic is more suitable to express liveness properties (e.g., "if Alice tries to do something infinitely often, she eventually succeeds").

6 Discussion

The biggest remaining problem is the lurking unsoundness with respect to the reality. In our modeling, any satisfied formula remains true in the future. Blockchain developers call our assumption finality. Bitcoin does not provide finality. Sometimes blockchains fork and all branches except one are orphaned. In those cases, a history becomes abandoned. There are three ways to deal with this problem:

[3] An axiom type is a logical formula with free variables like φ and ψ that can be substituted by any formulas.

1. assume finality and hold agents responsible if they trust blocks too early,
2. model the probability that blocks are final, and
3. model all forking branches.

Our current treatment is 1. Agents are required to ignore too fresh blocks and only take the contents of older blocks into their knowledge base, and our analysis breaks down when the agents are unluckily not patient enough. Our treatment is in line with the cryptocurrency exchanges' treatment of blockchains. Halpern and Pass [12] talk about probabilistic treatments, which supposedly would support the approach 2. The approach 3. seems not yet explored, but should be an interesting topic for modal logicians.

Another discrepancy is the requirement that agents remember all knowledge. This discrepancy does not matter when a protocol is shown not to work because incomplete memory doesn't work better than complete memory. On the other hand, once a protocol is shown to work, the concrete implementation of the protocol can optimize away irrelevant knowledge.

For more convenience, an automatic decision procedure is desirable that can judge whether a desired property is valid given some assumptions. Is there always a finite model that refutes an invalid property? Moreover, when one develops an on-chain program, their possible behavior should be spelled out automatically as logical formulas.

Our biggest diversion from the traditional treatment of knowledge is the treatment of blockchains as agents. Usually network participants or processes are treated as agents, but not a data structure maintained on the network. Since the semantics of intuitionistic epistemic logic never relies on a global state, we never had to identify a blockchain from a global view point. Given a local w, $f_{\mathrm{X}}(w)$ is blockchain X's state just according to w. If we used the traditional S5 epistemic logic [6], we would define which two states blockchain X can distinguish, but that criterion would assume an undisputed global identification of blockchain X.

The take away for blockchain protocol designers is that, for the atomic cross-chain swap to be atomic, availability and safety assumptions are necessary. For better certainly, merged blocks (e.g. Aspen's checkpoints [10]) will be effective. A merged block belongs to multiple blockchains all at once or none at all. With a merged block, two blockchains can share common knowledge in the same way Alice and Bob share common knowledge in Fig. 2 (b).

7 Conclusion

We used Kripke models of intuitionistic epistemic logic to see what kind of assumptions are necessary for "atomic" property of the atomic cross-chain swaps (Props. 3 and 4). We also showed a set of assumptions is enough for the "atomic" property to hold (Prop. 5). For cross-chain atomic swaps to be atomic, external agents' abilities to read and write blocks within a limited timeframe is crucial. To our knowledge, this is the first analysis of inter-blockchain communication using a modal logic.

References

1. Atzei, N., Bartoletti, M., Cimoli, T., Lande, S., Zunino, R.: SoK: Unraveling bitcoin smart contracts. In: Bauer, L., Küsters, R. (eds.) Principles of Security and Trust. pp. 217–242. Springer International Publishing (2018)
2. Bitcoin Wiki: Script (2010–2018), `https://en.bitcoin.it/wiki/Script`, accessed on 2018-03-12
3. Chandra, T.D., Toueg, S.: Unreliable failure detectors for reliable distributed systems. J. ACM 43(2), 225–267 (1996)
4. Chandy, K.M., Misra, J.: How processes learn. Distributed Computing 1(1), 40–52 (1986)
5. van Dalen, D.: Logic and Structure. Springer (1994)
6. Ditmarsch, H.v., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Springer, 1st edn. (2007)
7. Emerson, E., Clarke, E.M.: Using branching time temporal logic to synthesize synchronization skeletons. Science of Computer Programming 2(3), 241–266 (1982)
8. Fagin, R., Halpern, J.Y., Vardi, M.Y., Moses, Y.: Reasoning About Knowledge. MIT Press (1995)
9. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. J. ACM 32(2), 374–382 (1985)
10. Gencer, A.E., van Renesse, R., Sirer, E.G.: Short paper: Service-oriented sharding for blockchains. In: FC 2017. pp. 393–401. Springer (2017)
11. von Gleissenthall, K., Rybalchenko, A.: An epistemic perspective on consistency of concurrent computations. In: CONCUR 2013. pp. 212–226. Springer (2013)
12. Halpern, J.Y., Pass, R.: A knowledge-based analysis of the blockchain protocol. In: TARK 2017. EPTCS, vol. 251, pp. 324–335 (2017)
13. Hirai, Y.: An intuitionistic epistemic logic for asynchronous communication. Master's thesis, the University of Tokyo (2010)
14. Hirai, Y.: An intuitionistic epistemic logic for sequential consistency on shared memory. In: LPAR-16. pp. 272–289 (2010)
15. Lamport, L.: The part-time parliament. ACM Trans. Comput. Syst. 16(2), 133–169 (1998)
16. Lundeberg, M.B.: Advisory: secret size attack on cross-chain hash lock smart contracts (2018), `https://gist.github.com/markblundeberg/7a932c98179de2190049f5823907c016`, accessed on 2018-03-07
17. Luu, L., Chu, D.H., Olickel, H., Saxena, P., Hobor, A.: Making smart contracts smarter. In: CCS '16. pp. 254–269. ACM (2016)
18. Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P.: A secure sharding protocol for open blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 17–30. CCS '16, ACM (2016)
19. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), `https://bitcoin.org/bitcoin.pdf`, accessed on 2018-03-20
20. Nikolic, I., Kolluri, A., Sergey, I., Saxena, P., Hobor, A.: Finding The Greedy, Prodigal, and Suicidal Contracts at Scale. ArXiv e-prints (2018)
21. Nolan, T.: Re: Alt chains and atomic transfers (2013), `https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949`, accessed on 2018-03-12
22. Poon, J., Buterin, V.: Plasma: Scalable autonomous smart contracts (2017), `https://plasma.io/plasma.pdf`, accessed on 2018-03-07

23. Sergey, I., Hobor, A.: A concurrent perspective on smart contracts. In: FC 2017. pp. 478–493. Springer (2017)
24. Troelstra, A.S., Van Dalen, D.: Constructivism in mathematics: an introduction, vol. 1. Elsevier (1988)
25. Wood, G.: Polkadot: vision for a heterogeneous multi-chain framework (2016), `https://github.com/w3f/polkadot-white-paper/blob/master/PolkaDotPaper.pdf`, accessed on 2018-03-07