# Heterogeneous Broadcast

Murdoch J. Gabbay

Against Cryptoeconomics Day

14 June 2025

# What this talk is about

I'll report on some consensus research by Isaac Sheff and myself in the past 12 months, to understand his ideas on *heterogeneous broadcast and consensus* algorithms.

Heterogeneous Bracha Broadcast (HBB) is a distributed broadcast protocol, which is a precursor to the Heterogeneous Paxos (HP) distributed consensus algorithm.

Anoma plans to use HBB and HP, but they do not yet exist: part of our brief was to create them.

By the end of this talk, I hope you'll have a flavour of what a heterogeneous broadcast or consensus protocol looks like, and the issues involved in creating one.

Let's recall the standard, homogeneous, problem of distributed broadcast and consensus:

# Distributed Broadcast and Consensus

Broadcast and consensus protocols help a set of participants decide/deliver/output/learn a value $v$ (proposed by a participant), in the presence of some bounded amount of faulty behaviour.

Relevant correctness properties are:

1. Agreement (safety guarantee):
   If correct (= live and follows the protocol) $p$ delivers $v$ and correct $p'$ delivers $v'$, then $v = v'$.
2. Totality for broadcast (liveness guarantee):
   If correct $p$ delivers, then every correct $p'$ eventually delivers.
3. Totality for consensus (liveness guarantee):
   Every correct $p$ eventually delivers $v$.

# Distributed Broadcast and Consensus

Thus, subject to assumptions on the number of faulty participants:

1. For broadcast, correct participants must deliver and agree, if any one correct process delivers.
2. For consensus, correct participants must deliver and agree.

# A problem

Before Anoma can use HBB and HP, someone has to invent them.

Isaac had proposed a novel HP algorithm, and I had developed a novel axiomatic method for analysing distributed protocols.

We used the proposed axiomatic method to analyse the proposed HP, to see what would happen. During this analysis:

1. we broke the original HP protocol design (it was wrong) and
2. we proposed two replacement HBB protocols.

Let's consider some motivation for homogeneous/heterogeneous broadcast, from daily life:

# An example from daily life (homogeneous version)

Consider my family:

$$\{\text{Mum},\ \text{Dad},\ \text{Son}\}.$$

Participants Mum and Dad communicate to deliver decisions $[v_1, v_2, \ldots]$ to Son.

Our protocol guarantees the following properties:

- Agreement: if Mum and Dad announce values $v_i$ and $v_i'$ respectively for the $i$th decision, then these values are equal.
  E.g. I should not tell my son that he has to shower before bed, and my wife says he doesn't.
- Totality: if Mum announces some decision $v_i$ then Dad will back her up, and vice versa.
  E.g. if my wife tells my son to shower, and he prevaricates, I will eventually confirm with 'do as your mother says'.

# Heterogeneous version: Mum, Dad, Gran, Auntie, Son

My son may sometimes expect heterogeneous values. This is a feature, not a bug.

Suppose my son is visiting Gran's house with Auntie. He knows this pair of adults may produce different values than his parents'.

More specifically, Mum, Dad, Gran, and Auntie will agree on most decisions, *but* where they differ, Mum and Dad will form one bloc and Gran and Auntie will form another. Note that:

1. There may be differences in values decided, but this is not chaos.
2. Son has nontrivial, heterogeneous, hierarchical expectations about who will remain in agreement with whom.
3. These expectations exist *a priori*, before algorithms are run.

# Framing the problem

Intuitively, in the heterogeneous case the universe might split into factions that agree within themselves but not necessarily between one another.

This may be a result of faulty behaviour, but it may also be desirable, e.g. if we are bridging between blockchains, or coordinating between groups of people or machines.

Participants don't necessarily know in advance who will turn out to be faulty or who might be communicating with whom.

We want to be able to make guarantees about behaviour, given suitable (externally but not internally known) assumptions on the state of the system.

# Heterogeneous $\propto$ and $\infty$

We annotate participants with a follows relation $p \propto p'$. If we expect this to be symmetric, we may call this relation should agree and write $p \infty p'$.

E.g. Son expects Dad $\infty$ Mum and Gran $\infty$ Auntie with near certainty, and Mum $\infty$ Gran with lesser certainty.

Suitable generalisations of Agreement and Totality are:

1. Agreement: If correct $p$ delivers $v$ and correct $p'$ delivers $v'$ and $p' \propto p$, then $v = v'$.
2. Totality: If correct $p$ delivers $v$, then every correct $p'$ such that $p' \propto p$, eventually delivers $v$.

# Heterogeneous $\propto$ and $\infty$

The behaviour of $\propto$ and $\infty$ can be nontrivial.

Example: Suppose Jamie $\propto$ Isaac $\propto$ Chrisopher.

Now suppose I learn that hackers have used AI to steal Christopher's identity and clone his likeness.

Obviously, now I can't follow any information from Christopher. But the question is, do I know whether Isaac knows about the identity theft?

If I believe Isaac $\propto$ Christopher, then I cannot follow Isaac, because the hackers might be feeding him false information via their copy of Christopher.

So whom I $\propto$ or $\infty$ is determined not just about what I know or believe about Christopher: it's also about what I know or believe Isaac knows or believes about Christopher.

# How to make this all formal?

So far, this has all been intuition and examples. We will make it formal, using mathematics.

We will need:

1. *Semitopology*, for quorums.
2. *History structures*, for assessing whom to trust.
3. *A modal logic*, for formally asserting the rules of the protocol.

I will define each in turn.

You might like to strap yourself in for this bit . . .

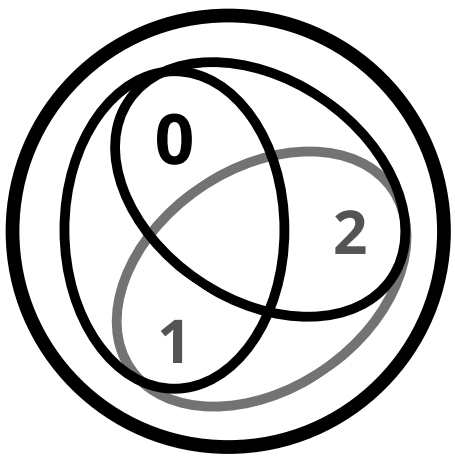
# Background: semitopology

A semitopology $(\mathrm{P}, \mathrm{Open})$ ($\approx$ quorum system) is a set of

- participants P and
- open sets or quorums $\mathrm{Open} \subseteq \mathit{powerset}(\mathrm{P})$.

You might recognise this as generalising the axioms of *point-set topology*, by removing the condition that an intersection of open sets must be open.

Where $\#\mathrm{P} = 3f + 1$, it is typical to take $O \in \mathrm{Open}$ when $\#O = 0$ or $\#O \geq 2f + 1$:



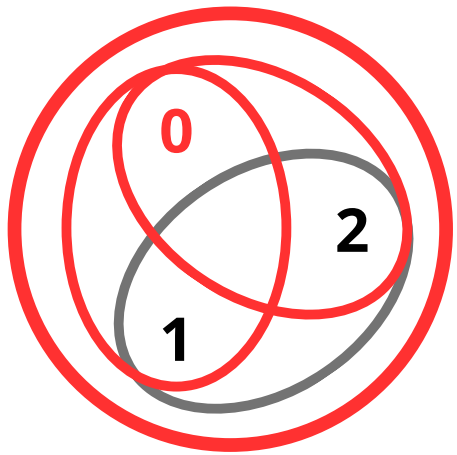

$$P = \{0, 1, 2\}$$

# Background: semitopology

Semitopology has a rich mathematical theory which I mostly elide, except for this one definition: if $p \in \mathrm{P}$ and $O \in \mathrm{Open}$ then call $O$ a neighbourhood of $p$ when $p \in O$. Define

$$nbhd(p) = \{O \in \mathrm{Open} \mid p \in O\}.$$

I'll use neighbourhoods later. Here are the neighbourhoods of 0:



# Motivation for history structures

The notion of $p \propto p'$ is inherently epistemic. This is a fancy way of saying that the validity of $p \propto p'$ depends on who knows what.

1. 'It's raining' is not epistemic.
   Validity just depends on whether it *is* raining.
2. 'I trust you' is epistemic.
   Validity depends what I think I know about your behaviour *and* that of other participants, *and* what I think you know. E.g. 'nice boy but easily led'.

$p \propto p'$ implies that $p$ trusts the communication channels between it and $p'$ *and* $p$ trusts the communication channels of $p'$.

All of this detail is packed into the word 'heterogeneous'.

# History structure: our model of knowledge

1. Fix arbitrary disjoint sets of participants P and events Event.
2. Define PreHistory inductively by

$$\text{PreHistory} = \mathit{fin}(\text{P} \times \text{Event} \times \text{PreHistory}).$$

   *fin* means finite powerset, so a prehistory is a finite set of participant-event-prehistory tuples.
3. For $H, H' \in \text{PreHistory}$, $p \in \text{P}$, and $E \in \text{Event}$ define

$$H' \in H \quad \text{when} \quad \exists p' \in \text{P}, E' \in \text{Event}.(p', E', H') \in H.$$

4. Call $H \in \text{PreHistory}$ transitive when $\forall H'.H' \in H \Rightarrow H' \subseteq H$.
5. Let History be the the set of hereditarily transitive prehistories. (Pleasingly reminiscent of von Neumann ordinals.)

Thus, a history is a prehistory all of whose elements are histories. Or:

*A history is a finite transitively closed set of participant-event-history tuples.*

# Example history structure (Jinx)

$$
\begin{aligned}
H_0 &= \{ && && \} \\
H_1 &= \{(\text{Jamie}, && \text{Echo}(\text{Morning! I love you}), && H_0) && \} \\
H_2 &= \{(\text{JW}, && \text{Echo}(\text{I love you too}), && H_1) && \} \cup H_1 \\
H_3 &= \{(\text{Jamie}, && \text{Echo}(\text{Did you sleep well?}), && H_2) && \} \cup H_2 \\
H'_3 &= \{(\text{JW}, && \text{Echo}(\text{Did you sleep well?}), && H_2) && \} \cup H_2 \\
H_4 &= \{(\text{JW}, && \text{Echo}(\text{Jinx!}), && H_3 \cup H'_3) && \} \cup H_3 \cup H'_3
\end{aligned}
$$

# A modal logic over histories

Call a tuple $\mathcal{M} = (\mathcal{H}, \mathrm{P}, \mathrm{Open})$ a model, where $\mathcal{H}$ is a history and $(\mathrm{P}, \mathrm{Open})$ is a semitopology.

I find it helpful to understand distributed algorithms in logical time, which represents a logical (not algorithmic) stage in an algorithm.

- In Paxos, the logical time is the 'round' (a natural number).
- The relevant notion of logical time in HBB is History (this was not obvious).

# Algorithmic time examples

### Phase 1 [ edit ]

#### Phase 1a: *Prepare* [ edit ]

A Proposer creates a message, which we call a Prepare. The message is identified with a unique number, *n*, which must be greater than any number previously used in a Prepare message by this Proposer. Note that *n* is not the value to be proposed; it is simply a unique identifier of this initial message by the Proposer. In fact, the Prepare message needn't contain the proposed value (often denoted by *v*).

The Proposer chooses at least a Quorum of Acceptors[*how?*] and sends the Prepare message containing *n* to them. A Proposer should not initiate Paxos if it cannot communicate with enough Acceptors to constitute a Quorum.

#### Phase 1b: *Promise* [ edit ]

The Acceptors wait for a Prepare message from any of the Proposers. When an Acceptor receives a Prepare message, the Acceptor must examine the identifier number, *n*, of that message. There are two cases:

1. If *n* is higher than every previous proposal number received by the Acceptor (from any Proposer), then the Acceptor must return a message (called a *Promise*) to the Proposer, indicating that the Acceptor will ignore all future proposals numbered less than or equal to *n*. The Promise must include the highest number among the Proposals that the Acceptor previously accepted, along with the corresponding accepted value.

   > This article **is missing information** about handling the first Prepare message. Please expand the article to include this information. Further details may exist on the talk page. *(July 2024)*

2. If *n* is less than or equal to any previous proposal number received by the Acceptor, the Acceptor needn't respond and can ignore the proposal. However, for the sake of optimization, sending a denial, or *negative acknowledgement* (*NAK*), response would tell the Proposer that it can stop its attempt to create consensus with proposal *n*.

### Phase 2 [ edit ]

#### Phase 2a: *Accept* [ edit ]

If a Proposer receives Promises from a Quorum of Acceptors, it needs to set a value *v* to its proposal. If any Acceptors had previously accepted any proposal, then they'll have sent their values to the Proposer, who now must set the value of its proposal, *v*, to the value associated with the highest proposal number reported by the Acceptors, let's call it *z*. If none of the Acceptors had accepted a proposal up to this point, then the Proposer may choose the value it originally wanted to propose, say *x*.[19]

The Proposer sends an *Accept* message, *(n, v)*, to a Quorum of Acceptors with the chosen value for its proposal, v, and the proposal number *n* (which is the same as the number contained in the *Prepare* message previously sent to the Acceptors). So, the *Accept* message is either *(n, v=z)* or, in case none of the Acceptors previously accepted a value, *(n, v=x)*.

This *Accept* message should be interpreted as a "request", as in "Accept this proposal, please!".

#### Phase 2b: *Accepted* [ edit ]

If an Acceptor receives an Accept message, *(n, v)*, from a Proposer, it must accept it if and only if it has *not* already promised (in Phase 1b of the Paxos protocol) to only consider proposals having an identifier greater than *n*.

If the Acceptor has not already promised (in Phase 1b) to only consider proposals having an identifier greater than *n*, it should register the value *v* (of the just received *Accept* message) as the accepted value (of the Protocol), and send an *Accepted* message to the Proposer and every Learner (which can typically be the Proposers themselves). Learners will learn the decided value *only after* receiving Accepted messages from a majority of acceptors, i.e. *not* after receiving just the first Accept message.

Else, it can ignore the Accept message or request.

**Algorithm 4:** Asynchronous Binding Crusader Agreement for Byzantine Faults instance $id$ ($BCA_{Byz}$)

**Input:** $x$

1: $approvedVals = \{\}$
2: send $\langle \text{echo}, id, x \rangle$ to all
3: **upon** receiving $\langle \text{echo}, id, v \rangle$ for the same non-$\perp$ $v$ from $t + 1$ parties:
4: **if** haven't echoed $v$ yet **then** send $\langle \text{echo}, id, v \rangle$ to all
5: **upon** receiving $\langle \text{echo}, id, v \rangle$ for the same non-$\perp$ $v$ from $n - t$ parties:
6: $approvedVals = approvedVals \bigcup v$
7: **if** haven't sent any echo2 message yet **then** send $\langle \text{echo2}, id, v \rangle$ to all
8: **wait until**
9: (1) $|approvedVals| > 1$ or
10: (2) received $\langle \text{echo2}, id, v \rangle$ for the same non-$\perp$ $v$ from $n - t$ parties
11: **if** (1) **then** send $\langle \text{echo3}, id, \perp \rangle$ to all
12: **else** send $\langle \text{echo3}, id, v \rangle$ to all
13: **wait until**
14: (1) $|approvedVals| > 1$ and received echo3 messages from at least $n - t$ parties or
15: (2) received $\langle \text{echo3}, id, v \rangle$ for the same non-$\perp$ $v$ from at least $n - t$ parties
16: **if** (1) **then** decide $\perp$
17: **else** decide $v$

# A modal logic over histories

Define predicates by

$$\phi ::= \bot \mid \phi \Rightarrow \phi \mid \forall p.\phi \mid E \mid \Downarrow\phi \mid \Diamond\phi \mid \boxdot_{l_1 \ldots l_n} \phi \mid \mathsf{seq} \mid p \propto p'.$$

Read

- $\Downarrow$ as *in the past*;
- $\Diamond$ as *someone*;
- $\boxdot_{l_1 \ldots l_n}$ as *for some* $l_1 \ldots l_n$*-quorum intersection*; and
- seq as *sequential*.

Interpretations of these symbols will follow.

# A modal logic over histories

Recall a model $\mathcal{M} = (\mathcal{H}, \mathsf{P}, \mathsf{Open})$ is a tuple such that $\mathcal{H}$ is a history structure and $(\mathsf{P}, \mathsf{Open})$ is a semitopology.

We interpret validity of predicates with respect to $\mathcal{M}$, a participant $p \in \mathsf{P}$, and a logical time $H \in \mathcal{H}$, thus the validity judgement has the form

$$p, H \vDash_{\mathcal{M}} \phi.$$

We now define it:

# A modal logic over histories (easy bits)

Recall that $\mathcal{H}$ is a set of $(p, E, H)$-tuples, and $\phi$ is (in)valid in $\mathcal{H}$ at some $p$ and $H$:

$$\begin{array}{lcl}
p, H \models_{\mathcal{M}} \bot & \Leftrightarrow & \bot \\
p, H \models_{\mathcal{M}} \phi \Rightarrow \phi' & \Leftrightarrow & p, H \models \phi \Rightarrow p, H \models_{\mathcal{M}} \phi' \\
p, H \models_{\mathcal{M}} \forall p'.\phi & \Leftrightarrow & \forall p'' \in \mathrm{P}. p, H \models \phi[p' := p''] \\
p, H \models_{\mathcal{M}} E & \Leftrightarrow & (p, E, H) \in \mathcal{H} \\
p, H \models_{\mathcal{M}} \Downarrow \phi & \Leftrightarrow & \exists H' \in H.\ p, H' \models_{\mathcal{M}} \phi \\
p, H \models_{\mathcal{M}} \Diamond \phi & \Leftrightarrow & \exists p' \in \mathrm{P}.\ p', H \models_{\mathcal{M}} \Downarrow \phi
\end{array}$$

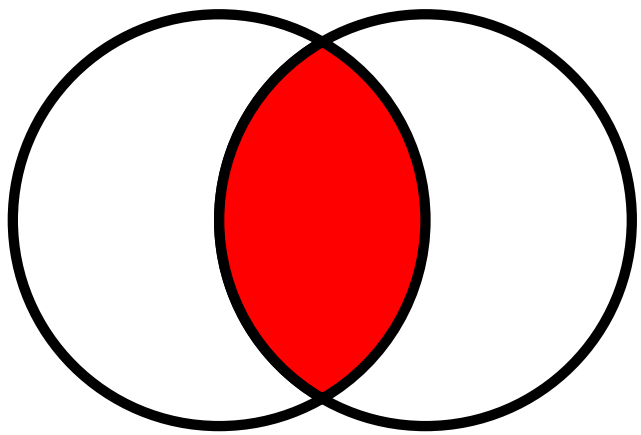
We can express other logical connectives in the usual ways, e.g.

$$\neg\phi = (\phi \Rightarrow \bot) \quad \phi \vee \phi' = (\neg\phi \Rightarrow \phi') \quad \exists p'.\phi = (\neg \forall p'.\neg\phi).$$

# A modal logic over histories: quorum intersections

$$p, H \models_{\mathcal{M}} \boxdot_{l_1 \ldots l_n} \phi \;\Leftrightarrow$$
$$\exists O_1 \in nbhd(l_1) \ldots O_n \in nbhd(l_n). \forall p' \in O_1 \cap \ldots \cap O_n. (p', H \models_{\mathcal{M}} \Downarrow \phi)$$

This expresses that there exists *some* quorum intersection between quorums of $l_1, \ldots, l_n$, *all* of whose participants have $\phi$. E.g for $l_1$ and $l_2$:



(Image credit: Wikipedia.)

A useful special case is $\boxdot_l \phi$, which means that there exists an $l$-quorum of participants *all* of whom have $\phi$.

# A modal logic over histories: quorum intersections

Also write $\diamondsuit_{I_1 \ldots I_n} \phi$ for $\neg \boxdot_{I_1 \ldots I_n} \neg \phi$ (the *de Morgan dual*). Then:

$$p, H \models_{\mathcal{M}} \diamondsuit_{I_1 \ldots I_n} \phi \;\Leftrightarrow$$
$$\forall O_1 \in nbhd(I_1) \ldots O_n \in nbhd(I_n). \exists p' \in O_1 \cap \ldots \cap O_n. (p', H \models_{\mathcal{M}} \Downarrow \phi)$$

This expresses that *all* $I_1 \ldots I_n$-quorum intersections contain *some* participant with $\phi$.

# A modal logic over histories: sequentiality

$$p, H \vDash_{\mathcal{M}} \text{seq} \;\Leftrightarrow\; \forall E, E' \in \text{Event}.\; p, H \vDash_{\mathcal{M}} \Downarrow E \wedge \Downarrow E' \Rightarrow p, H \vDash_{\mathcal{M}} \Downarrow(E \wedge \Downarrow E') \vee \Downarrow(E' \wedge \Downarrow E)$$

This says that events at $p$ in $H$ occur *in sequence*.

We assume of the real world that each event is cryptographically signed by whichever participant performed it, so $p$ can't just report an $E'$ for some other $p'$ if that didn't actually happen — but $p$ can *omit* historial events when it reports its events to *other* participants.

That's the episemic part: reporting events while omitting context, i.e. out of sequence. Nonsequentiality indicates epistemological hacking!

# A modal logic over histories

Define

$$\text{safe} = \forall p'.\diamondsuit_{pp'} \text{seq}.$$

Thus,

$$p, H \models_{\mathcal{M}} \text{safe} \;\Leftrightarrow\; p, H \models_{\mathcal{M}} \forall p'.\diamondsuit_{pp'} \text{seq}.$$

$p$ is safe when it has sequential quorum intersections with every participant.

If $p, H \models_{\mathcal{M}} \phi$ does not depend on the choice of participant $p$, then omit it and just write $H \models_{\mathcal{M}} \phi$. Define

$$H \models_{\mathcal{M}} p_1 \propto p_2 \;\Leftrightarrow\; p_2, H \models_{\mathcal{M}} \text{safe}.$$

So $p_1$ follows $p_2$ at logical time $H$ when $p_2$ is safe at that time (and if it's not, it doesn't).

# The protocol (axiomatic presentation)

Finally we can express an HBB protocol.

| | |
|---|---|
| (**Echo0**?) | $\mathrm{echo}(0, v) \Rightarrow \Diamond \mathrm{propose}(v)$ |
| (**Echo1**?) | $\mathrm{echo}(1, I, v) \Rightarrow \boxdot_I \mathrm{echo}(0, v)$ |
| (**EchoDup**) | $\mathrm{echo}(0, v) \Rightarrow \Downarrow \mathrm{echo}(0, v') \Rightarrow v{=}v'$ |
| (**Vote**?) | $\mathrm{vote}(I, I', v) \Rightarrow I \propto I' \wedge \Diamond \mathrm{echo}(1, I', v)$ |
| (**Deliver**?) | $\mathrm{deliver}(I, I', v) \Rightarrow \boxdot_I \mathrm{vote}(I, I', v)$ |

Note that this is short and precise.

We did need to define history structures and learn some logic to get to this point, but that's a feature in the sense that the complexity is factored into the complexity inherent in logical time (history structures) and the complexity inherent in the specific protocol.

# Agreement property

Lemma: $H \vDash_{\mathcal{M}} \Diamond\text{deliver}(I_2, I_2', v_2) \wedge \Diamond\text{deliver}(I_1, I_1', v_1)$ and $H \vDash_{\mathcal{M}} \diamondsuit_{I_1 I_2}\text{seq}$ implies $v_1 = v_2$.

Proof: Suppose $H \vDash_{\mathcal{M}} \Diamond\text{deliver}(I_2, I_2', v_2) \wedge \Diamond\text{deliver}(I_1, I_1', v_1)$. By (**Deliver**?) we have

- an $I_2$-quorum of past $\text{vote}(I_2, I_2', v_2)$, and
- an $I_1$-quorum of past $\text{vote}(I_1, I_1', v_1)$.

By assumption, these quorums intersect at a sequential participant $q$. Assume wlog $H' \in H$ and

$$q, H' \vDash_{\mathcal{M}} \text{vote}(I_2, I_2', v_2) \wedge \Downarrow\text{vote}(I_1, I_1', v_1).$$

By (**Vote**?) $I_2'$ is safe at logical time $H'$, so it has sequential quorum intersections with $I_1'$. Following (**Echo1**?), there exist an $I_2'$-quorum of $\text{echo}(0, v_2)$ and an $I_1'$-quorum of $\text{echo}(0, v_1)$ and so a sequential participant in their intersection that has echoed both, in sequence! By (**EchoDup**) $v_1 = v_2$, and we're done.

# What have we learned?

1. This stuff is nontrivial. There are a lot of definitions and finding them was not obvious. It's taken months.
2. The axioms expressing the protocol are fairly straightforward.
3. There's a large design space of possible protocols. We've only found two that work — I present the simpler one! I haven't found a directed way to search the design space.
4. You get used to this world of histories and sequential and safe participants.
   It's a lot to understand in this talk, but as a conceptual structure it seems a solid foundation for future work.
5. The logical approach was (for me) *absolutely necessary* to understand the protocol. I find it highly effective.

# Future work

1. Write this up as a conference or (perhaps more likely) journal paper.
2. Implement it.
3. Build a model-checker for history structures might speed up exploration of the state space.
4. Improve the protocols. They work, but they're either a bit weak or a bit slow. It would be nice, at least, to have a fast path.