1 Proofs of Construction

- approach to proving the correctness of our state channel network protocol

- the nature of state channels tends to make the logic complex - value is moved between participants by exchanging statements about the distribution of assets held on-chain - inevitably you end up reasoning about the statements you hold and their interpretation by the chain, and the possible actions of the other participants in the channel

- inherent danger, funds are locked on-chain - participants act maliciously and stop cooperating at any point in the protocol - safe at all times

1.1 Channel Funding and Value

We will start by considering the interpretation of the outcome of a state channel. Suppose A is a participant in a state channel, L, that reaches an (allocation) outcome, ω, that allocates x coins to A. What does that mean for A? In particular, how much more can A withdraw from the system due to that outcome? Understanding this is key to analysing state channel networks.

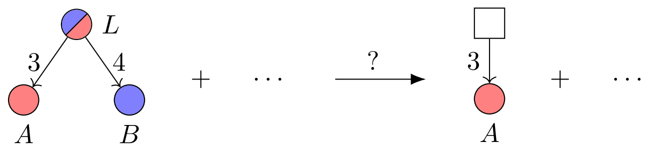

Figure 1

There is one case where the answer to these questions is very straightforward: where the channel L itself has enough coins in the adjudicator to cover the entire allocation. In this case, we say the channel is **directly funded**. If this happens, A will receive all x coins allocated to them in ω.

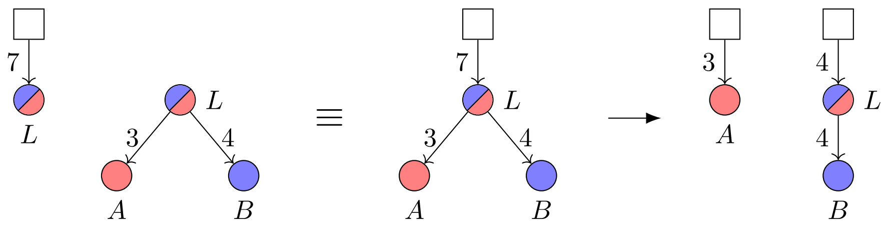

Figure 2

This is a good start, but the whole point of state channel *networks* is to move beyond the case where every channel needs to be directly funded. Suppose instead that L is not directly funded but there is another channel, L', that is. Further suppose that L' has reached an outcome where all its coins are allocated to L. Using this outcome, we know we can redistribute the coins in the adjudicator to L, recreating the situation above, where L was directly funded. Therefore, in this situation we also know that A will receive the x coins from the outcome of L, and that L can be considered to be funded, in some sense. Note that we did not actually need to perform the redistribution on-chain to reach this conclusion - we just needed to be able to reason that the outcome enabled us to.

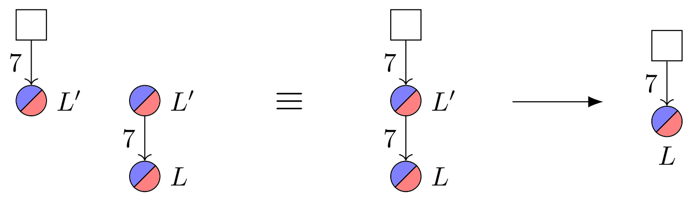

Figure 3

In the previous paragraph, we looked at the case where L' had already reached an outcome. In general, this will not be the case; indeed, the power of state channels comes from the ability to move between many potential outcomes as the interaction progresses. In order to say whether a channel is funded, we will need to start not with an outcome but with a **network state**. The network state, Σ, for a participant, A, consists of:

1. The state of the adjudicator:
   (a) The balances held
   (b) Any finalized outcomes
2. For each channel A is a participant of:
   (a) Signed statements that A has received
   (b) Signed statements that A has sent
   (c) Private information held by A

The private information always includes A's signing key for the channel and can also include information specific to the application running in the channel; for example, for a game of battleships the private information would include the positions of A's ships. Note that A's network state does not include a detailed model of which statements are held by specific other participants - just what A has sent and received. Generally we assume that all other participants are controlled by a single adversary, pooling their resources and statements.

We can now proceed with some definitions of funding and value:

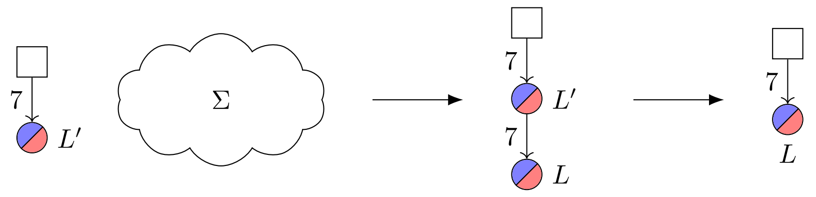

Figure 4

A channel χ is **funded** with x coins for participant A with network state Σ, if A has an unbeatable strategy for obtaining a state where χ is directly funded with x coins, starting from Σ.

The **value** for participant A of a network state Σ is the maximum x for which A has an unbeatable strategy for obtaining a state where the balance of A's address in the adjudicator is x coins, starting from Σ

The concept of an unbeatable strategy can involve a full range of actions allowed within the protocol including signing statements, refusing to sign statements, launching/responding to on-chain challenges and calling on-chain operations to redistribute funds. We will cover this in more detail in section 1.3.

1.2 Constructing

Now that we have defined what we mean by a channel being funded and a state having value, we can start to talk about the state channel network constructions that will form the bulk of the paper. A construction specifies both the network state for each participant and a sequence of states that can be used to reach it. Presenting a construction will follow the same rough pattern:

1. Show that a given network state funds a channel
2. Show you can build it from a known state, using a sequence of value-preserving single channel updates

The second point here is really important and often neglected in state channel work - often the hardest part is not finding a construction that funds a channel but demonstrating how the participants can safely reach this state. The single channel update requirement is a decision we have made when designing the protocol. What this means is that we do not allow atomic updates across multiple channels; each update to the system comprises sending or receiving a single statement applying to just one channel. This keeps channel updates are independent, which makes it a lot easier to reason about finalizability on a per-channel basis. We make protocol design harder but remove complexity that would otherwise be present at the engineering stage.

We require that the sequence of state transitions must be value preserving for each participants involved. While the power of state channel networks comes from being able to move value off-chain, opening and closing channels can be viewed as rewriting the existing state in a different form and therefore should not change the value. We furthermore make the assumption that participants will be willing to make any transition that preserves their value, meaning that value-preservation is both a necessary and sufficient property for constructing network states.

We call this last assumption the **Simple Transition Rule**. The assumption here is slightly subtle, when combined with the fact that our value calculations ignore the cost of the on-chain redistribution operations required to extract the value from the system. If you were to consider this cost, then moving from a simpler to a more complicated construction would slightly decrease the value of the network for the participants. In practice, the simple transition rule incorporates the assumption that the utility from being able to fund channels off-chain will outweigh the slight increase in cost in the worst-case scenario. Modelling the cost of the on-chain operations is beyond the scope of this paper.

1.3 Unbeatable Strategies

In the definitions of value and funding, we talked about having an unbeatable strategy for obtaining some state on-chain. The means that whatever actions (or lack of actions) other participants and external parties might take, the target state is still obtainable. This is not the easiest definition to work with: to show that a strategy is unbeatable it seems that you have to consider all possible actions other parties could take. In this section, we will break this down and give some tools to make it easier to show that a strategy is unbeatable.

We start by outlining the rules for interacting with the blockchain. When evaluating whether a strategy is unbeatable, we make the following assumptions about blockchain transactions:

1. **Transactions are unimpeded**: given that the current time is t and $\epsilon > 0$, then it is possible for any party to apply any operation, O, on-chain before time $t + \epsilon$.

2. **Transactions *can* be front-run**: given two parties, p_1 and p_2, and two operations, O_1 and O_2, there is no way for p_1 to ensure that they can apply O_1 to the chain before p_2 applies O_2.

The first assumption sidesteps issues of censorship, chain congestion and timing considerations around the creation of blocks. In practice, this assumption should hold if ϵ is sufficiently large, which can be accomplished by picking sensible channel timeouts. The second assumption rules out any strategies that rely on executing a given transaction on-chain before someone else executes a different one.

We will next break an unbeatable strategy for obtaining a state into two stages: finalization and redistribution.

Finalization happens on a per-channel basis, with different channels finalizing independently. This makes it easier to reason about which outcomes are possible. In general, we cannot assume that the outcome will be known; we might have to take multiple possible outcomes through to the redistribution step. The

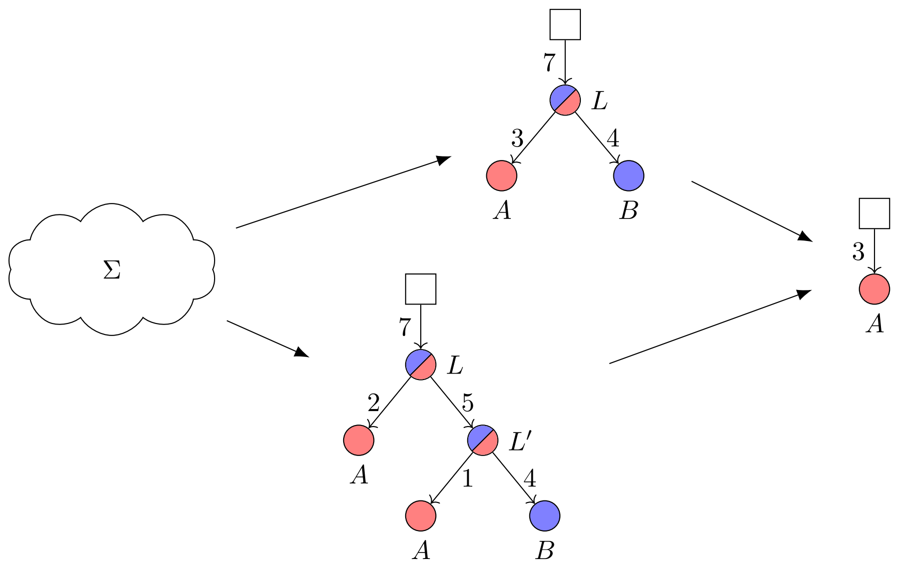

Figure 5: *Cool, huh?*

finalization step depends heavily on the rules of the state channel framework. We will cover finalization in more detail in section 1.4.

Reasoning about when a redistribution strategy is unbeatable, depends heavily on the protocol involved. We will cover the logic here in the sections on Turbo and Nitro protocol. In Turbo, it turns out that the answer is simple: any strategy that works is unbeatable. In Nitro, it is more complicated to show that redistribution strategies are unbeatable but we provide a few tools to help.

1.4 Finalizable Outcomes

We say an outcome, Ω, is **finalizable** for participant A, if A has an unbeatable strategy for finalizing this outcome in the adjudicator. We use the notation $[\chi \mapsto \Omega]_A$, to represent a state of a channel, χ, where the outcome, Ω, is finalizable by A.

$$[\chi \mapsto \Omega]_A \xrightarrow{\text{A's unbeatable strategy}} [\![\chi \mapsto \Omega]\!] \qquad (1)$$

It follows from the definition that exactly one of the following statements is true about a channel χ at any point in time:

1. The outcome of χ has already been finalized on-chain: $[\![\chi \mapsto \Omega]\!]$
2. Participant p is the unique participant with one or more finalizable outcome(s), $\Omega_1, \ldots, \Omega_m$. We write this $[\chi \mapsto \Omega_1, \ldots, \Omega_m]_p$.
3. There are at least two participants, $P = \{p_1, \ldots, p_m\}$, who share the same finalizable outcome, Ω. We write this $[\chi \mapsto \Omega]_{p_1,\ldots,p_m}$.
4. There are no participants with any finalizable outcomes.

The definition of finalizability excludes the case where two different finalizable outcomes are held by different participants, as in this case at least one participant's strategy would be beatable by the other participant's strategy. None of the protocols we present make use of the final situation, where no participant has a finalizable outcome, and we believe this state should generally be avoided.

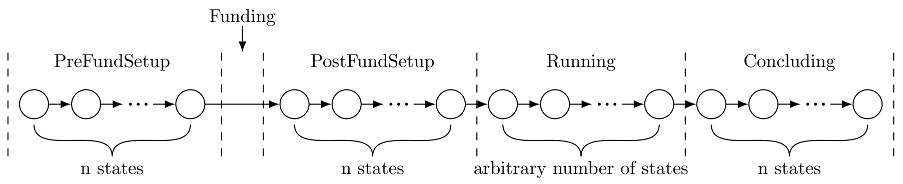

Figure 6: *Cool, huh?*

In the special case where the outcome of a channel is finalizable by all its participants, we say that the outcome is **universally finalizable**. For a ForceMove channel, this happens at the following points in its lifecycle:

1. After the first n states have been broadcast. In this state, we say the channel is at the **funding point**.
2. When a single conclusion proof is known to each participant. In this state, we say the channel is in the **concluded state**.

It is an important property of ForceMove that all channels have one universally finalizable state at the beginning of their lifecycle and one at the end [1].

If a participant has no finalizable outcomes, their analysis of the network needs to be performed in terms of their **enabled outcomes**. The enabled outcomes for a participant, p, is defined as the set of outcomes that p has no strategy to prevent from being finalized. We write the set of enabled outcomes for p as $[\chi \mapsto \Omega_1 \ldots \Omega_m]_{(p)}$.

For any participant, p, in a channel, χ, exactly one of the following statements is true at a given point in time:

1. p has at least one finalizable outcome.

[1] If a channel does not end with a conclusion proof, it ends with an expired on-chain challenge, in which case the outcome is already finalized on-chain.

2. p has at least two enabled outcomes.

Note that if a participant has only enabled a single outcome, that outcome must be finalizable for them.

1.5 Consensus Game

Another important example of universally finalizable states comes from the **consensus game**. The consensus game is a ForceMove *application*, which means it specifies a certain set of transitions rules that can be used to define the allowed state transitions for a ForceMove channel. We will make heavy use of the consensus game throughout the paper.

The consensus game provides a way for participants to move from one universally finalizable outcome to another, provided that they all agree. One participant proposes the new outcome, Ω_2. On their turn, each subsequent participant decides whether to accept the transition to the new outcome or whether to cancel the transition and return to the old outcome, Ω_2.

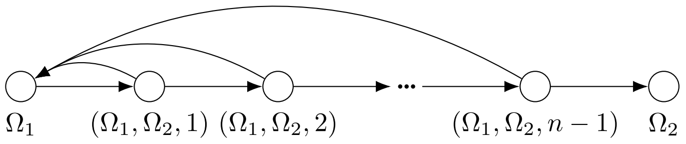
Ω_1
$(\Omega_1, \Omega_2, 1)$
$(\Omega_1, \Omega_2, 2)$
$(\Omega_1, \Omega_2, n-1)$
Ω_2

Figure 7: *A consensus game transition from Ω_1 to Ω_2, for a channel with n participants. The counter records how many participants have approved the transition. If all participants agree, they finish in a state with outcome Ω_2. Any participant can reject the transition, returning to the state with Ω_1.*

Throughout the only enabled outcomes for any participant are Ω_1 and Ω_2. In particular, a participant has the finalizable outcome $[\chi \mapsto \Omega_1]_p$ until they approve the transition, and then enabled outcomes $[\chi \mapsto \Omega_1, \Omega_2]_{(p)}$ until they receive the final state. When the final state is broadcast, every participant has the finalizable outcome $[\chi \mapsto \Omega_2]_p$.

1.6 Outcomes First

In practice, it is hard to write networks states down concisely. Instead, we will write our constructions in terms of outcomes and use the properties of the consensus game to reason that (a) network states exist that lead to this outcome and (b) we can find a sequence of network states to transition from one outcome to another.

In particular

- if I have two network outcomes that differ in the outcome of a single CG channel

- then I can find a sequence of single-update network states that interpolate between them

- write down a sequence of outcomes - update one channel at a time - and have the same value to all participants

- the start and conclude states are also finalizable

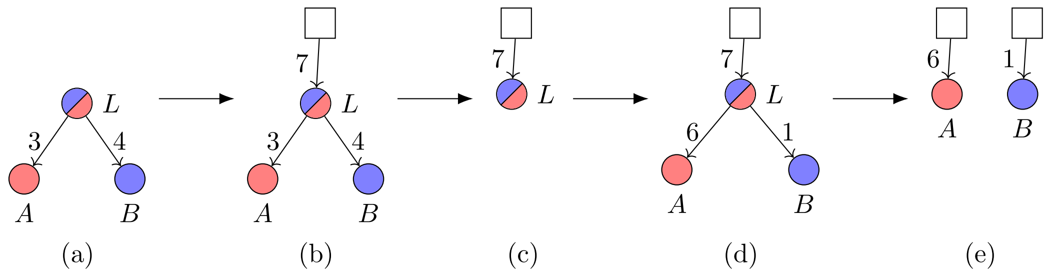

Figure 8: *In (a), A and B have exchanged the first two states in the channel L, bringing them to the funding point. At this point the channel is not yet funded. In step (b), both participants have deposited into the adjudicator. In step (c), the channel L is running. A and B do not have a finalizable outcome and the ultimate outcome is governed by the rules given by the channel's game library. In step (d), A and B have created a conclusion proof and therefore have another universally finalizable outcome. They are then able to finalize this outcome on-chain and withdraw their funds in (e).*