There are many situations especially in the context of sharding where you can have an event take place that is conditional on something else having happened in another shard or context, but where that something else cannot be verified for a relatively long time. In these cases, it would be nice to have that event "happen" immediately in some sense, so that the event can then itself be used as a cause for other events (eg. Alice sends 10 coins to Bob, Bob immediately sends the coins to Charlie).

This can happen in a few cases:

- Cross-shard transactions, where the execution process of one shard may not really

know the result of the execution process of another shard until some time delay

- Payments based on cryptoeconomic accumulators
- Payments based on ZKPs or other heavy computation, where that computation is done cryptoeconomically/interactively

Consider a UTXO-based HLL where individual transactions can have dependencies that cannot be instantly finalized, but can be instantly guessed with some accuracy. These dependencies could be inputs that exist in other shards, or they could be in the transaction verification itself. Each output would store in the state not just its own contents, but also a list of dependencies that have not yet been finalized.

After some period of time (eg. 2 months), any dependency can be removed; if a state object has no more dependencies, then it is considered "finalized". If a state object is created by a transaction whose inputs have dependencies, it inherits all of those dependencies.

Clients can use one of four methods to calculate state:

1. Trust nothing until finality (ie. 2 months)
2. Trust nothing until a privately selected waiting time shorter than 2 months
3. Personally perform all required computations associated with any dependencies
4. Rely on cryptoeconomic evidence for partial confirmations: if someone makes a message of the form "I put down 1 ETH and agree to lose it if this result is not this value", then assign that dependency a "strength" of 1 ETH. Go through the dependency graph and calculate the smallest amount of ETH needed to be destroyed to invalidate a given state object, and show that state object as partially confirmed, with confirmation strength depending on the ETH-denominated size of that "weakest link".

This can be extended to an account-based model, by treating every account as a series of UTXOs that happen to live at the same address; every transaction that touches some account has a version of that account as an input and a new version of that account as an output.

Given that an object may have many thousands of dependencies, it may not be efficient to store all dependencies of an object within that object's state. Instead, we can simply have a protocol where if an object's state is recalculated, then there is an incentive (paid out from the penalty paid by whoever published an incorrect claim) to send transactions to recalculate all downstream states. Alternatively, we can make the cryptoeconomic claims themselves be not claims on the correctness of one dependency of one object, but also on its entire ancestry tree; this way, recalculating everything downstream of an erroneous object will still be necessary, but there will at least be a stronger guarantee that there is someone who will pay for all of the transactions to do so.