Karlsruhe Institute of Technology

Verifying Payment Channels with TLA+

Matthias Grundmann

Introduction

Off-chain protocols are used to scale blockchain-based currencies

How to verify correctness / security properties of off-chain protocols in way that is accessible to protocol developers?

Specify protocol and properties in TLA$^+$ and run model-checker

Use case for this talk:
Payment Channel with the Lightning Network's protocol

DSN Research Group
KASTEL Institute

Goal and Challenges

Show that security property for payment channels is fulfilled or output counterexample.

Specify blockchain and transactions with signatures and hashes.

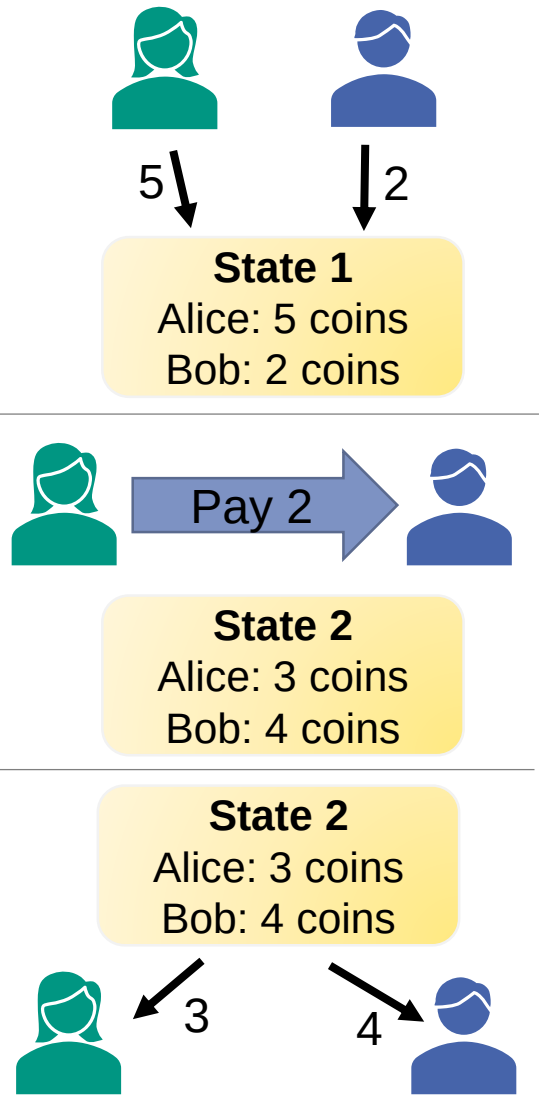
Specify time and adversarial behavior while keeping state space explorable.

Provide protocol developer with intuitive and understandable output for counterexamples.

Idea of Payment Channels

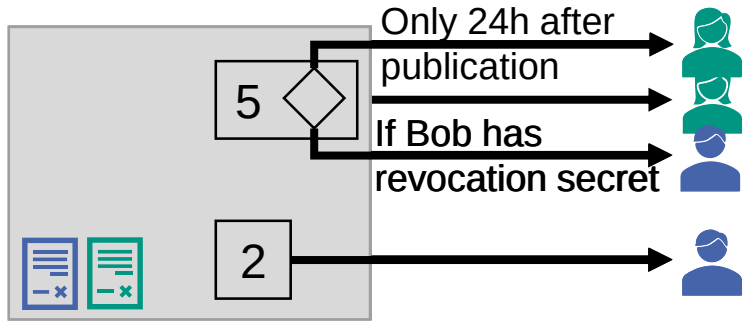
- A payment channel between two parties is based on a shared account.
- Open payment channel
  - Alice and Bob deposit coins into the shared account.
- Payment over channel
  - Alice and Bob agree to change the balances inside the account.
  - Can be repeated multiple times in both directions.
- Close channel
  - Alice and Bob receive their balance stored in the shared account.

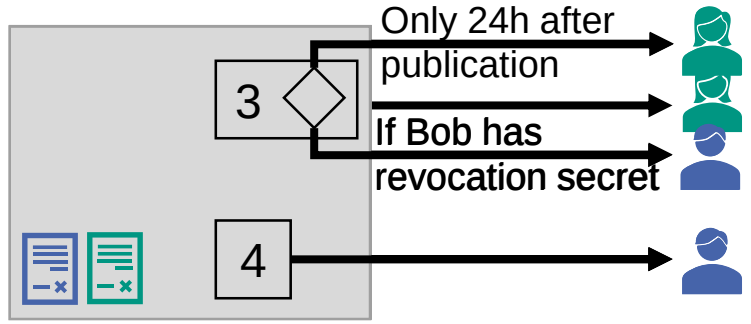
- Security properties
  - Each (honest) party can close the channel in the latest state.
  - No party can successfully close the channel in an outdated state.



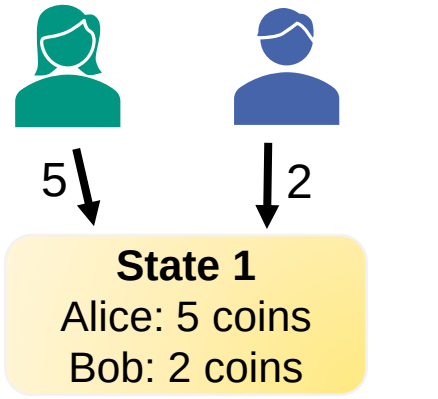
Lightning Network: Payment Channels

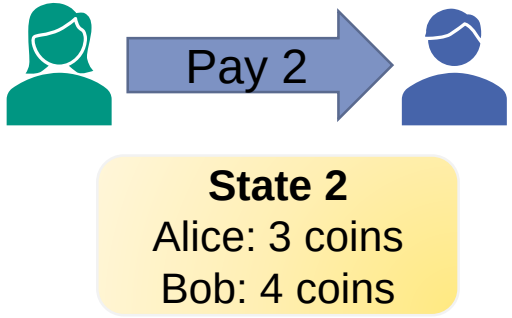
- State is encoded in commitment transactions.
- For each new state, a new commitment transaction is created.
- Old states need to be revocable.





(Construction is symmetrical for Bob.)





It's hard to see whether security guarantees are fulfilled. We specify this construction and the protocol to create it in TLA⁺ and show that it fulfills the security guarantees.

DSN Research Group
KASTEL Institute

Security Property

$$AliceGetsBalance \triangleq$$
$$\Diamond\Box((\neg AliceVars.HaveCheated) \Rightarrow AlicesOnChainBalance(LedgerTx,\ MAX\_TIME + 5) \geq AliceBalance)$$
$$BobGetsBalance \triangleq$$
$$\Diamond\Box((\neg BobVars.HaveCheated) \Rightarrow BobsOnChainBalance(LedgerTx,\ MAX\_TIME + 5) \geq BobBalance)$$

- “In any possible execution, an honest party finally gets (at least) their correct balance.”
  - Even it the other party publishes an outdated state.

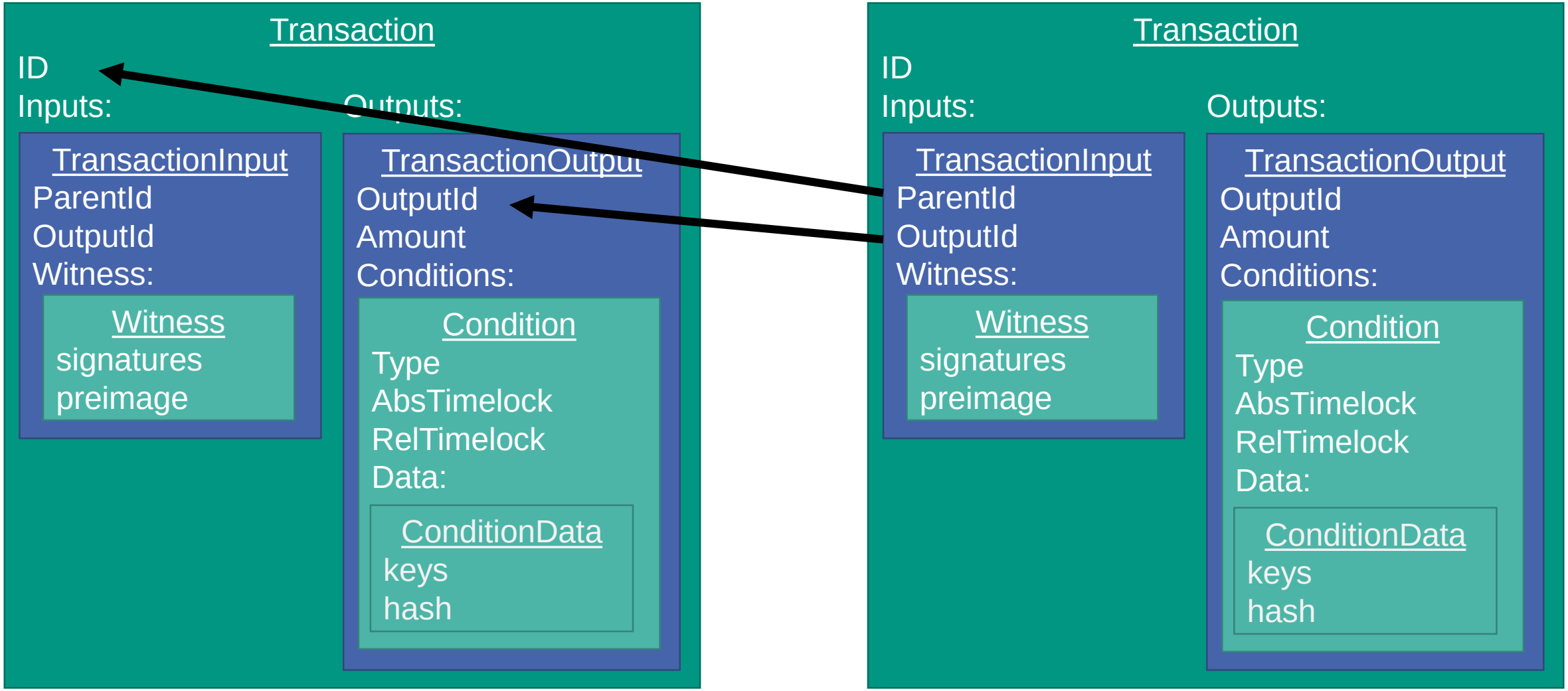
DSN Research Group
KASTEL Institute

Specification of the Blockchain

- To specify the construction of payment channels, we need a specification of transactions and the blockchain.

- Requirements
  - Specify all aspects required by Payment Channel Protocol
  - Keep it as simple as possible for performance of Model Checking
  - Must be instantiable by Bitcoin so that counterexamples can be transferred to the real world

Specifying Transactions in the UTXO Model



DSN Research Group
KASTEL Institute

Specification of the Blockchain

- Signatures

Requirements	Approach
Signature can only be created using corresponding key	To sign a transaction with key k, the key k is added to an input's witness
Verifiable whether signature and key correspond	
Sending signature of transaction to other party	Send whole signed transaction

- Hash functions

Requirements	Approach
Hash can be calculated from preimage	Modeled as Identity function Clear from context, whether an element is preimage or hash
Verifiable whether preimage and hash correspond	
Preimage cannot be computed from hash	Never compute preimage from hash in specification

- Transaction IDs

Requirements	Approach
Unique ID for each transaction	We use an arbitrary integer value. For each transaction created or modified, a new unique value is chosen.
If $ID_1 = ID_2$, then $transaction_1 = transaction_2$	

- Blockchain
  - Modeled by the set of all published transactions

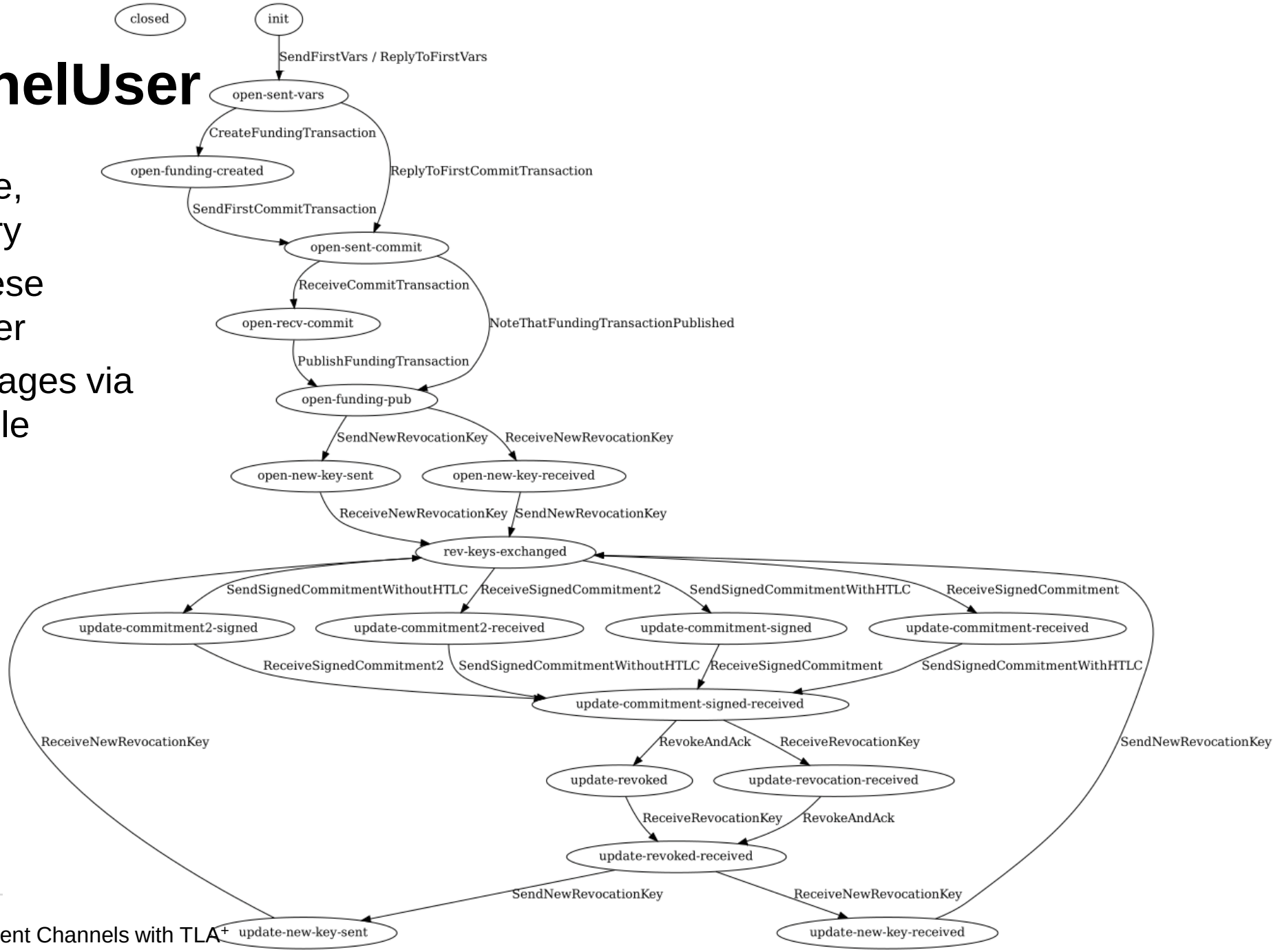
DSN Research Group
KASTEL Institute

Specification of Payment Channel Protocol

- Specification of possible actions of two users (Alice and Bob) defined by PaymentChannelUser
- Start from initial state
  - Ledger contains one transaction with an output spendable by Alice
- Next state: Any action of Alice or Bob

PaymentChannelUser

- User stores state name, variables, and inventory
- Actions manipulate these variables and the ledger
- Users exchange messages via global message variable



Adversarial Behavior

- A dishonest user can cheat at any time
  - “Typical Cheating”
    Publish a transaction that is not the latest state
  - “Opportunistic Cheating”
    Publish a transaction that spends any outputs spendable using the keys in the user’s inventory
  - “Crash”
    Stop interacting with the other party
- Sending invalid messages not modeled as it is detectable by the other party.

$$
\begin{aligned}
&Cheat \triangleq \\
&\quad \land\ Honest = \text{FALSE} \\
&\quad \land\ \neg \text{ENABLED}\ RedeemHTLCAfterClose \\
&\quad \land\ \ \lor \exists\, tx \in Inventory.transactions : \\
&\qquad \land\ tx.id \neq Vars.LatestCommitTransactionId \quad \text{This case would be closing} \\
&\qquad \land\ tx.id \neq Vars.fundingTxId \\
&\qquad \land\ \text{LET}\ signedTx \triangleq SignTransaction(tx) \\
&\qquad\ \ \text{IN}\ \ Ledger!PublishTx(signedTx) \\
&\qquad \land\ State' = \text{“closed”} \quad \text{Necessary because } RedeemHTLCAfterClose \text{ requires } State = \text{“closed”} \\
&\qquad \land\ \text{UNCHANGED}\ \langle TIOUsedTransactionIds \rangle \\
&\quad\ \lor \exists\, tx \in MyPossibleNewTransactionsWithoutOwnParents(LedgerTx) : \\
&\qquad \land\ TIO!MarkAsUsed(tx.id) \\
&\qquad \land\ Ledger!PublishTx(tx) \\
&\qquad \land\ \text{UNCHANGED}\ \langle State \rangle \\
&\quad\ \lor\ \land \exists\, tx \in LedgerTx : tx.id = Vars.fundingTxId \\
&\qquad\ \land\ State' = \text{“closed”} \\
&\qquad\ \land\ \text{UNCHANGED}\ \langle LedgerTx,\ TIOUsedTransactionIds \rangle \\
&\quad \land\ Vars' = [Vars\ \text{EXCEPT}\ !.HaveCheated = \text{TRUE}] \\
&\quad \land\ \text{UNCHANGED}\ \langle Message,\ Inventory,\ Balance,\ PendingBalance \rangle \\
&\quad \land\ \text{UNCHANGED}\ UnchangedVars
\end{aligned}
$$

DSN Research Group
KASTEL Institute

Time Progression

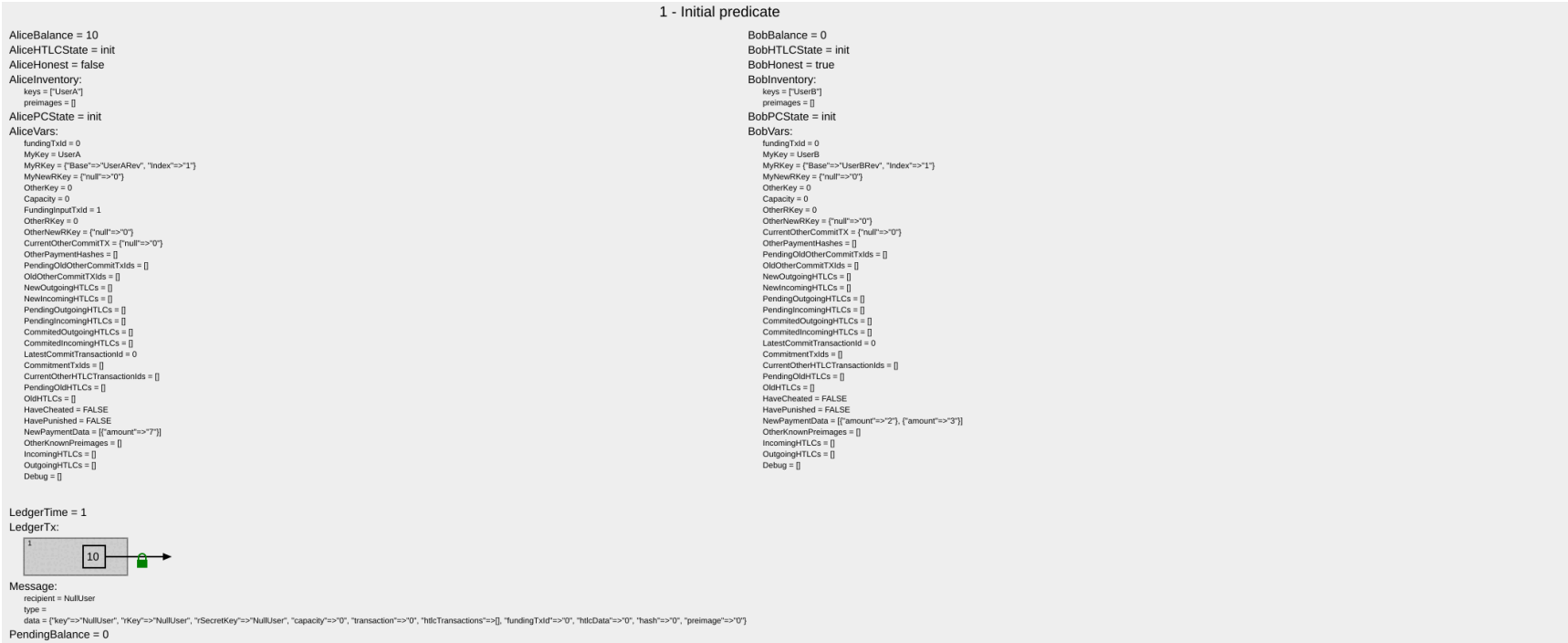
- Weak fairness: Users perform actions they can
- Time/Block count progresses at arbitrary times
- Requirement: Users need to perform actions before a certain time T
- Specification: Users specify a limit for time progression depending on certain conditions (“NextTimedStepTime”)
  - As long as the condition is fulfilled, time does not advance further than time T
  - After an action has caused the condition to fail, time can advance
- Improvement to limit states:
  - Time “jumps” to NextTimedStepTime instead of incrementing in smaller steps
  - Open to show that this does not skip relevant cases

$$
\begin{aligned}
&AdvanceLedgerTime \triangleq \\
&\quad \land LedgerTime' \in \{i \in 0 \,..\, MAX\_TIME : \\
&\qquad \land i > LedgerTime \\
&\qquad \land i \leq AlicePC!NextTimedStepTime \\
&\qquad \land i \leq BobPC!NextTimedStepTime \\
&\qquad \land (i \in \{AlicePC!NextTimedStepTime,\ BobPC!NextTimedStepTime,\ MAX\_TIME\})\} \\
&\quad \land \text{UNCHANGED } \langle LedgerTx,\ AliceInventory,\ BobInventory,\ AliceVars,\ BobVars,\ AlicePCState,\ BobPCSta
\end{aligned}
$$

DSN Research Group
KASTEL Institute

Understanding Counterexamples

- Visualization of complete state
- Flip-book style animation to navigate through states that led to a counterexample
- Work in progress



DSN Research Group
KASTEL Institute

Results

- Size of specification: ~ 1,200 LoC

- Modeled scenario: Alice starts with 10, pays 7 to Bob. Bob pays 3 and 2 to Alice.
  - 1,131,490 distinct states
  - Run time on notebook (two workers on i5-7300U): ~ 2 hours

- Lesson Learned: Run time of TLC depends not only on the number of successor states but also on the number of ‘non-successor’ states
  - Run time is also spent on branches of the next-state action that evaluate to FALSE
  - Challenge: Such expressions should evaluate to FALSE as early as possible

Conclusion

- Contributions
  - TLA$^+$-Specification of transactions and blockchain that can be reused for other off-chain protocols
  - TLA$^+$- Specification of payment channel protocol and visualization of counterexamples
  - Model-checking shows that security property is fulfilled by payment channel construction
  - Specification at https://github.com/kit-dsn/payment-channel-tla

- Future work
  - Multi-hop payments: from Payment Channel to Payment Channel Network
  - Specify other protocol (e.g., watchtower protocol) and verify security properties

- Contact: matthias.grundmann@kit.edu