MODULE *Contracts*

This spec captures the behaviour of commitment transactions on the two sides of a Lightning channel.

We model the various kinds of outputs a commitment transactions will have over its lifetime.

The state of the commitment transaction changes in reponse to the various actions like supersede, publish, etc are taken by parties.

We also do not deal with the communication protocol between nodes for creating and updating commitment transactions. This spec will only focus on the various commitment transaction and their lifecycle in response to interaction between parties and the blockchain.

We ignore the details of how transactions are signed and just mark transactions as signed. This lets us focus on the specifying the behaviour of the commitment transactions without dealing with lower level complexities.

The model defines the intial balance from *alice* to bob. TLA+ will handle situations where channels are balanced and when all the balance is on the other side.

TODO: Add actions for closing channels. Currenly we only have support for breach *tx* and the corresponding breach remedy txs.

TODO: Add *HTLCs*.

EXTENDS *Integers*,
TLC,
Sequences,
FiniteSets

CONSTANTS

CSV,	The *csv* value to use in contracts
InitialBalance	Initial balances for *alice* and bob

$SeqToSet(s) \triangleq \{s[i] : i \in \text{DOMAIN } s\}$

Current channel contracts only ever have two parties

$Party \triangleq \{\text{“alice”}, \text{“bob”}\}$

For the first revocation we only need two keys per party

$NumKey \triangleq 2$

Set of all keys

$Key \triangleq \{\langle p, k\rangle : p \in Party, k \in 0 \,..\, NumKey\}$

Value to capture missing *CSV* in output

$NoCSV \triangleq \text{CHOOSE } c : c \notin 0 \,..\, CSV$

Abstract out all outputs as meant to be spent by a party, is it signed by party and other party.

$Output \triangleq [party : Party,$
$\quad type : \{\text{“multisig”}, \text{“p2wkh”}\},$
$\quad csv : \{CSV\} \cup \{NoCSV\},$
$\quad amount : 0 \,..\, InitialBalance * 2]$ All the balance can be on one side

Multisig with no *csv* encumberance

$CreateMultisigOutput(party, amount) \triangleq$
$\quad [type \mapsto \text{“multisig”},$
$\quad party \mapsto party,$
$\quad csv \mapsto NoCSV,$
$\quad amount \mapsto amount]$

$CreateRSMCOutput(party, amount) \triangleq$
$\quad [type \mapsto \text{“multisig”},$
$\quad party \mapsto party,$
$\quad csv \mapsto CSV,$
$\quad amount \mapsto amount]$

$CreatePKOutput(party, amount) \triangleq$
$\quad [type \mapsto \text{“p2wkh”},$
$\quad party \mapsto party,$
$\quad csv \mapsto NoCSV,$
$\quad amount \mapsto amount]$

$NoSpendHeight \triangleq -1$

In contrast to txids, we simlpy use the party, index tuple to find the *tx* and the *vout* to get the output pointed to by the input

$Input \triangleq [party : Party, index : Int, vout : Int]$

Transaction record.

$Tx \triangleq [party : Party,$
$\quad index : Int,$
$\quad height : Int,$
$\quad inputs : Seq(Input),$
$\quad outputs : Seq(Output),$
$\quad party\_signed : \text{BOOLEAN} ,$
$\quad other\_party\_signed : \text{BOOLEAN}]$

VARIABLES

$alice\_cts$,	Commitment *tx* for *alice*
bob_cts,	Commitment *tx* for bob
$alice\_brs$,	Breach remedy transactions for *alice*
bob_brs,	Breach remedy transactions for bob
$mempool$,	The CT txs that have been broadcasted.

published, The CT that has been included in a block and confirmed.
index,
chain_height

$vars \triangleq \langle alice\_cts,\ bob\_cts,\ alice\_brs,\ bob\_brs,\ mempool,\ published,$
$chain\_height,\ index\rangle$

Helper function to get other party

$OtherParty(party) \triangleq$ CHOOSE $p \in Party : p \neq party$

The channel funding transaction. All commitment txs spend from the output of this tx.

$FundingTx \triangleq$
$[party \mapsto$ "alice", Only *alice* is funding
$index \mapsto 1,$
$height \mapsto 1,$
$inputs \mapsto \langle\rangle,$ FT inputs do not matter
$outputs \mapsto \langle CreateMultisigOutput($"alice", $InitialBalance)\rangle,$
$party\_signed \mapsto$ TRUE,
$other\_party\_signed \mapsto$ TRUE
$]$

Create a commitment transaction given the party, index and key to use.

Other party hands this CT to this party, therefore it is signed by other party.

$CreateCT(party,\ i,\ key\_num,\ amount,\ other\_amount) \triangleq$
$[party \mapsto party,$
$index \mapsto i,$
$height \mapsto NoSpendHeight,$
Input for CT is the FT *multisig* output (1, 1)
$inputs \mapsto \langle[party \mapsto$ "alice", $index \mapsto 1,\ vout \mapsto 1]\rangle,$
$outputs \mapsto \langle CreateRSMCOutput(party,\ amount),$
$CreatePKOutput(OtherParty(party),\ other\_amount)\rangle,$
$party\_signed \mapsto$ FALSE,
$other\_party\_signed \mapsto$ TRUE$]$

Breach remedy transactions are handled as presigned transactions instead of by passing private keys around. This is different from the Poon-Dryja LN paper.

The party creates this tx, signs it and sends it to the other party.

$CreateBR(party,\ i,\ amount) \triangleq$
$[party \mapsto party,$
$index \mapsto i,$
$height \mapsto NoSpendHeight,$
BR spend the $RSMC$ output from the corresponding index CT.
$inputs \mapsto \langle[party \mapsto OtherParty(party),\ index \mapsto i,\ vout \mapsto 1]\rangle,$
Spending BR output will give the balance to party

$outputs \mapsto \langle CreatePKOutput(party,\ amount)\rangle$,
$party\_signed \mapsto$ TRUE,
The other party presigns the BR so that this party can spend it
$TODO$: switch to exchanging private keys for the BR instead
$other\_party\_signed \mapsto$ TRUE]

$Init \triangleq$
Balanced channel to start with
$\land\ alice\_cts = \{CreateCT(\text{“alice”},\ 2,\ 0,\ InitialBalance,\ 0)\}$
$\land\ bob\_cts = \{CreateCT(\text{“bob”},\ 2,\ 0,\ 0,\ InitialBalance)\}$
$\land\ alice\_brs = \{CreateBR(\text{“bob”},\ 2,\ InitialBalance)\}$
$\land\ bob\_brs\ \ = \{CreateBR(\text{“alice”},\ 2,\ 0)\}$ Bob did not add funds
$\land\ mempool = \{\}$
$\land\ published = \{FundingTx\}$
$\land\ index = 3$
$\land\ chain\_height = 1$ The genesis block is the FT

$TypeInvariant \triangleq$
$\land\ index \in Int$
$\land\ alice\_cts \in$ SUBSET Tx
$\land\ bob\_cts \in$ SUBSET Tx
$\land\ alice\_brs \in$ SUBSET Tx
$\land\ bob\_brs\ \ \in$ SUBSET Tx
$\land\ mempool \in$ SUBSET Tx
$\land\ published \in$ SUBSET Tx

$LastCT(party\_cts) \triangleq$
CHOOSE $ct \in party\_cts : \forall\ y \in party\_cts : ct.index \geq y.index$

$MaxIndex(party\_cts) \triangleq$
$(LastCT(party\_cts)).index$

$AnyCT \triangleq$ (CHOOSE $ct \in alice\_cts \cup bob\_cts$: TRUE)

Create commitment transaction as well as the corresponding beach remedy txs.

Breach remedy transactions are pre-signed transactions instead of they private key being sent over to the other party.

delta is the balance going from *alice* to bob. We allow negative balances to enable payments in other other direction.

Parties are free to keep creating CT even if FT is spent. They will not be usable, but the protocol does not disallow this.

$SupersedeCommitmentTx(delta) \triangleq$
$\land$
LET
$key\_index \triangleq 1$ $TODO$: manage key numbers

$last\_alice\_ct \triangleq LastCT(alice\_cts)$
$last\_bob\_ct \triangleq LastCT(bob\_cts)$

IN

Create CTs till channel is not closed

$\land\ published = \{FundingTx\}$
$\land\ last\_alice\_ct.outputs[1].amount - delta > 0$
$\land\ last\_alice\_ct.outputs[2].amount + delta \leq InitialBalance$
$\land\ alice\_cts' = alice\_cts \cup$
$\{CreateCT(\text{“alice”}, index, key\_index,$
$last\_alice\_ct.outputs[1].amount - delta,$
$last\_alice\_ct.outputs[2].amount + delta)\}$
$\land\ bob\_cts' = bob\_cts \cup$
$\{CreateCT(\text{“bob”}, index, key\_index,$
$last\_bob\_ct.outputs[1].amount + delta,$
$last\_bob\_ct.outputs[2].amount - delta)\}$

Alice's gets a BR it can immediately spend when corresponding CT is spen, and vice versa

$\land\ alice\_brs' = alice\_brs \cup$
$\{CreateBR(\text{“bob”}, index, last\_alice\_ct.outputs[1].amount)\}$
$\land\ bob\_brs' = bob\_brs \cup$
$\{CreateBR(\text{“alice”}, index, last\_bob\_ct.outputs[1].amount)\}$
$\land\ index' = index + 1$

$\land$ UNCHANGED $\langle mempool, published, chain\_height\rangle$

Broadcast a commitment transaction to the blockchain. The commitment is first signed. The protocol allows all commitments to be broadcast, what happens next depends on the status of the commitment transaction.

If the tx is the latest commitment transaction it can be spent later.

If not, it gives the other party a chance to spend the breach remedy tx.

$TODO$: We only spec CSV (self) commitment transaction. We need to handle the non-CSV output being published and co-op closes.

$BroadcastCommitment(party) \triangleq$

$\land\ alice\_cts \neq \{\}$
$\land\ bob\_cts \neq \{\}$
$\land$

LET

$key\_index \triangleq 1$ $TODO$, manage key numbers
$cts \triangleq$ IF $party = \text{“alice”}$ THEN $alice\_cts$ ELSE bob_cts
$ct \triangleq$ CHOOSE $ct \in cts$: TRUE

IN

The commitment is not already in $mempool$

$\land\ ct \notin mempool$

No commitment has already been confirmed

$\land\ published = \{FundingTx\}$

$\land\ mempool' = mempool \cup \{[ct \text{ EXCEPT } !.party\_signed = \text{TRUE}]\}$
$\land$ UNCHANGED $\langle alice\_cts,\ bob\_cts,\ alice\_brs,\ bob\_brs,$
$published,\ index,\ chain\_height\rangle$

Confirm any transaction from $mempool-$ this indeed is sparta. Any $mempool$ tx can be confirmed. So we model just that.

The only requirement is to make sure the CSV has expired.

$ConfirmMempoolTx \triangleq$
$\exists\, tx \in mempool :$
$\land \exists\, o \in SeqToSet(tx.outputs) :$
$\lor\ o.type = \text{“multisig”} \land o.csv < chain\_height$ CSV expired
$\lor\ o.type = \text{“p2wkh”} \land o.csv = NoCSV$ Without a CSV
$\land\ tx \notin published$ Tx is not already confirmed
$\land\ mempool' = mempool \setminus \{tx\}$
$\land\ chain\_height' = chain\_height + 1$
$\land\ published' = published \cup \{[tx \text{ EXCEPT } !.height = chain\_height']\}$
$\land$ UNCHANGED $\langle alice\_cts,\ bob\_cts,\ alice\_brs,\ bob\_brs,$
$index\rangle$

Broadcast a breach remedy transaction in response to a commitment transaction.

party is broadcasting the tx

$BroadcastBR \triangleq$
$\land \exists\, \langle m,\ b\rangle \in mempool \times (alice\_brs \cup bob\_brs) :$
$\land\ published = \{FundingTx\}$ Channel is not closed yet
$\land\ m.outputs[1].type = \text{“multisig”}$
Offending tx in $mempool$
$\land\ chain\_height - 1 < m.outputs[1].csv$
$\land\ m.party = b.party$
$\land\ mempool' = mempool \cup \{m\}$
$\land$ UNCHANGED $\langle alice\_cts,\ bob\_cts,\ alice\_brs,\ bob\_brs,$
$index,\ published,\ chain\_height\rangle$

$Next \triangleq$
$\lor \exists\, d \in 1\,..\,2 : SupersedeCommitmentTx(d)$
$\lor \exists\, p \in Party : BroadcastCommitment(p)$
$\lor\ BroadcastBR$
$\lor\ ConfirmMempoolTx$

$Spec \triangleq\ Init \land \Box[Next]_{\langle vars\rangle}$

$Liveness \triangleq\ \text{WF}_{vars}(BroadcastBR)$

$FairSpec \triangleq\ Spec \land Liveness$

$TODO-$ Add $BalanceInvariant$: Sum of all amounts on all txs = $InitialBalance$