———————————————— 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

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.
$Output \triangleq [type : \{\text{"multisig"}, \text{"p2wkh"}\},$
$\qquad\qquad party : Party,$
$\qquad\qquad csv : \{CSV\} \cup \{NoCSV\},$

1

$$amount : 0 .. InitialBalance * 2]$$ All the balance can be on one side

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

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

$NoSpendHeight \triangleq\ -1$

Transaction record.

TODO: Track output being spent.

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

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.
$index,$	
$chain\_height$	

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

Helper function to get other party

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

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

$CreateCT(party,\ i,\ key\_num,\ amount,\ other\_amount) \triangleq$
$\qquad [party \mapsto party,$
$\qquad\ index \mapsto i,$

2

$$height \mapsto NoSpendHeight,$$
$$outputs \mapsto \langle CreateRSMCOutput(party,\ amount),$$
$$CreatePKOutput(OtherParty(party),\ other\_amount)\rangle,$$
$$party\_signed \mapsto \text{FALSE},$$
$$other\_party\_signed \mapsto \text{TRUE}]$$

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

$CreateBR(party,\ i,\ amount) \triangleq$
$$[party \mapsto party,$$
$$index \mapsto i,$$
$$height \mapsto NoSpendHeight,$$
$$outputs \mapsto \langle CreatePKOutput(OtherParty(party),\ amount)\rangle,$$
$$party\_signed \mapsto \text{TRUE},$$
$$other\_party\_signed \mapsto \text{FALSE}]$$

$Init \triangleq$

  Balanced channel to start with
$\wedge\ alice\_cts = \{CreateCT(\text{``alice''},\ 0,\ 0,\ InitialBalance,\ 0)\}$
$\wedge\ bob\_cts = \{CreateCT(\text{``bob''},\ 0,\ 0,\ 0,\ InitialBalance)\}$
$\wedge\ alice\_brs = \{CreateBR(\text{``alice''},\ 0,\ InitialBalance)\}$
$\wedge\ bob\_brs\ = \{CreateBR(\text{``bob''},\ 0,\ InitialBalance)\}$
$\wedge\ mempool = \{\}$
$\wedge\ published = \{\}$
$\wedge\ index = 1$
$\wedge\ chain\_height = 1$ The genesis block is the *FT*

$TypeInvariant \triangleq$
$\wedge\ index \in Nat$
$\wedge\ alice\_cts \in \text{SUBSET}\ Tx$
$\wedge\ bob\_cts \in \text{SUBSET}\ Tx$
$\wedge\ alice\_brs \in \text{SUBSET}\ Tx$
$\wedge\ bob\_brs\ \in \text{SUBSET}\ Tx$
$\wedge\ mempool \in \text{SUBSET}\ Tx$
$\wedge\ published \in \text{SUBSET}\ Tx$

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

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

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

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.

$SupersedeCommitmentTx(delta) \triangleq$
 $\wedge$
  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
   $\wedge published = \{\}$
   $\wedge last\_alice\_ct.outputs[1].amount - delta > 0$
   $\wedge last\_alice\_ct.outputs[2].amount + delta \leq InitialBalance$
   $\wedge alice\_cts' = alice\_cts \cup$
    $\{CreateCT(\text{``alice''}, index, key\_index,$
     $last\_alice\_ct.outputs[1].amount - delta,$
     $last\_alice\_ct.outputs[2].amount + delta)\}$
   $\wedge bob\_cts' = bob\_cts \cup$
    $\{CreateCT(\text{``bob''}, index, key\_index,$
     $last\_bob\_ct.outputs[1].amount + delta,$
     $last\_bob\_ct.outputs[2].amount - delta)\}$
   $\wedge alice\_brs' = alice\_brs \cup$
    $\{CreateBR(\text{``alice''}, index, last\_alice\_ct.outputs[1].amount)\}$
   $\wedge bob\_brs' = bob\_brs \cup$
    $\{CreateBR(\text{``bob''}, index, last\_alice\_ct.outputs[1].amount)\}$
   $\wedge index' = index + 1$
 $\wedge$ 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.

$BroadcastCommitment(party) \triangleq$
 $\wedge alice\_cts \neq \{\}$
 $\wedge bob\_cts \neq \{\}$
 $\wedge$

LET

    $key\_index \triangleq 1$ *TODO*, manage key numbers

    $cts \triangleq$ IF $party \;= $ "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

    $\land published = \{\}$

    $\land mempool' = mempool \cup \{ct\}$

$\land$ 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 rule is to make sure the CSV has expired, and that is handled at the time of inserting the tx into *mempool*

$ConfirmMempoolTx \triangleq$

    $\exists \, tx \in mempool :$

        $\land tx \notin published$ Tx is not already confirmed

        $\land mempool' = mempool \setminus \{tx\}$

        $\land chain\_height' = chain\_height + 1$

        $\land published' = published \cup \{[tx$ 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

$BroadcastBR(party) \triangleq$

    $\land$

        LET

            $cts \triangleq$ IF $party =$ "alice" THEN $alice\_cts$ ELSE bob_cts

            $brs \triangleq$ IF $party =$ "alice" THEN bob_brs ELSE $alice\_brs$

        IN

            $\exists \, in\_mempool \in mempool :$

                CT was broadcast by the other party

                $\land in\_mempool.outputs[1].party = OtherParty(party)$

                Revoked CT was broadcast

                $\land in\_mempool.index < MaxIndex(cts)$

                This party already signed the ct as local sig

                $\land in\_mempool.other\_party\_signed =$ TRUE

                CSV hasn't expired - given FT is at height 1

                $\land chain\_height < CSV$

                $\land mempool' = mempool \cup \{$CHOOSE $b \in brs : b.index = in\_mempool.index\}$

    $\land$ UNCHANGED $\langle alice\_cts, \; bob\_cts, \; alice\_brs, \; bob\_brs,$
                    $index, \; published, \; chain\_height \rangle$

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

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

$Liveness \triangleq \exists\, p \in Party,\, d \in \{-1, 1\}:$
   $\mathrm{WF\_} vars(PublishBR(p) \lor SupersedeCommitmentTx(d))$
$Liveness \triangleq \exists\, d \in \{1\} : \mathrm{WF}_{vars}(SupersedeCommitmentTx(d))$

$FairSpec \triangleq Spec \land Liveness$

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