—————————————————— 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 focuss 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 liveness properties for breach / justice txs.

TODO: Add *HTLCs*.

EXTENDS *Integers*,
         TLC,
         Sequences,
         FiniteSets

CONSTANTS
    CSV,   The csv value to use in contracts
    Height,   The height up to which we run the spec
    NumTxs,   The number of commitment txs we want
    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
$NoCSV \triangleq \text{CHOOSE } c : c \notin 0 .. CSV$

Multisig outputs without *CSV* encumberance

1

$MultiSig \triangleq Party \times Party \times \{NoCSV\}$

Multisig outputs with CSV encumberance

$MultiSigWithCSV \triangleq Party \times Party \times \{CSV\}$

$P2WKH$ outputs, without encumbrance

$P2WKH \triangleq Key$

Set of all signatures for all commit txs. The signature in real world is related to the commit transaction. However, we leave out this complication of how the signature is generated. If there is a signature by a key on a tx, it is assumed it is correctly signed as per bitcoin's requirements

$Sig \triangleq \{\langle p, k \rangle : p \in Party, k \in 0 \ldots NumKey - 1\}$

Value to capture unsigned transactions

$NoSig \triangleq \text{CHOOSE } s : s \notin Sig$

$CT \triangleq [index \mapsto 0 \ldots NumTxs,$
$\qquad multisig \mapsto MultiSigWithCSV, pk \mapsto P2WKH,$
$\qquad local\_sig \mapsto Sig \cup \{NoSig\},$
$\qquad remote\_sig \mapsto Sig \cup \{NoSig\},$
$\qquad balance \mapsto 0 \ldots InitialBalance]$

$PublishId \triangleq \{\langle p, i, h \rangle : p \in Party, i \in 0 \ldots NumTxs, h \in 0 \ldots Height\}$
$NoSpend \triangleq \langle\rangle$

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
- $mempool\_ct$, The CT txs that have been broadcasted.
- $published\_ct$, The CT that has been included in a block and confirmed.
- $index$

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

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.

$CreateCT(party, i, key\_num, balance) \triangleq$
$\qquad [index \mapsto i,$
$\qquad multisig \mapsto \langle party, OtherParty(party), CSV \rangle,$
$\qquad pk \mapsto \langle party, key\_num \rangle,$

$\quad\quad\quad\quad\quad local\_sig \mapsto NoSig,$
$\quad\quad\quad\quad\quad remote\_sig \mapsto \langle OtherParty(party),\ key\_num \rangle,$
$\quad\quad\quad\quad\quad balance \mapsto balance]$

$Init\ \triangleq$

$\quad\quad \wedge\ alice\_cts = \{CreateCT(\text{``alice''},\ 0,\ 0,\ InitialBalance)\}$
$\quad\quad \wedge\ bob\_cts = \{CreateCT(\text{``bob''},\ 0,\ 0,\ InitialBalance)\}$
$\quad\quad \wedge\ alice\_brs = \{\}$
$\quad\quad \wedge\ bob\_brs = \{\}$
$\quad\quad \wedge\ mempool\_ct = \{\}$
$\quad\quad \wedge\ published\_ct = NoSpend$
$\quad\quad \wedge\ index = 1$

$TypeInvariant\ \triangleq$

$\quad\quad\quad \wedge\ \forall\, ct \in alice\_cts \cup bob\_cts:$
$\quad\quad\quad\quad \wedge\ ct.index \in Nat$
$\quad\quad\quad\quad \wedge\ ct.local\_sig \in Sig \cup \{NoSig\}$
$\quad\quad\quad\quad \wedge\ ct.remote\_sig \in Sig \cup \{NoSig\}$
$\quad\quad\quad\quad \wedge\ ct.pk \in P2WKH$
$\quad\quad\quad\quad \wedge\ ct.multisig \in MultiSigWithCSV$
$\quad\quad\quad \wedge\ \forall\, br \in alice\_brs \cup bob\_brs:$
$\quad\quad\quad\quad \wedge\ br.index \in Nat$
$\quad\quad\quad\quad \wedge\ br.pk \in P2WKH$
$\quad\quad\quad \wedge\ mempool\_ct \in \text{SUBSET } PublishId$
$\quad\quad\quad \wedge\ published\_ct \in PublishId \cup \{NoSpend\}$
$\quad\quad\quad \wedge\ index \in Nat$

$MaxIndex(party\_cts)\ \triangleq$
$\quad\quad (\text{CHOOSE } x \in party\_cts: \forall\, y \in party\_cts: x.index \geq y.index).index$

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

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

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.

$SupersedeCommitmentTx(delta)\ \triangleq$
$\quad\quad \wedge$

LET
$key\_index \triangleq 1$
$last\_alice\_ct \triangleq LastCT(alice\_cts)$
$last\_bob\_ct \triangleq LastCT(bob\_cts)$

IN
$\land published\_ct = NoSpend$
$\land NumTxs > index$

$\land last\_alice\_ct.balance + delta > 0$
$\land last\_bob\_ct.balance - delta > 0$
$\land alice\_cts' = alice\_cts \cup$
    $\{CreateCT(\text{``alice''}, index, key\_index,$
      $last\_alice\_ct.balance + delta)\}$
$\land bob\_cts' = bob\_cts \cup$
    $\{CreateCT(\text{``bob''}, index, key\_index,$
      $last\_alice\_ct.balance - delta)\}$
$\land alice\_brs' = alice\_brs \cup$
    $\{[index \mapsto index, pk \mapsto \langle \text{``bob''}, key\_index\rangle]\}$
$\land bob\_brs' = bob\_brs \cup$
    $\{[index \mapsto index, pk \mapsto \langle \text{``alice''}, key\_index\rangle]\}$
$\land index' = index + 1$
$\land$ UNCHANGED $\langle mempool\_ct, published\_ct\rangle$

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

If the tx is the latest commitment transaction it is succesfuly spend.

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.

$PublishCommitment(party, height) \triangleq$
  LET $i \triangleq AnyCT.index$
  IN
    $\land mempool\_ct = \{\}$
    $\land mempool\_ct' = mempool\_ct \cup \{\langle party, i, height\rangle\}$
  $\land$ UNCHANGED $\langle alice\_cts, bob\_cts, alice\_brs, bob\_brs,$
        $published\_ct, index\rangle$

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

party is publishing the breach remedy tx when it is on index *CT*, and the chain is on height.

This tx is immediately published on chain.

TODO: We skip the *BR* going through the mempool and confirm it immeidiately. This can be improved too.

$PublishBR(party, in\_mempool, height) \triangleq$
    LET $cts \triangleq$ IF $party = $ "alice" THEN $alice\_cts$ ELSE bob_cts
    IN
        $\land published\_ct = NoSpend$ No CT is confirmed on chain yet
        $\land mempool\_ct \neq \{\}$ Only if some CT has been published
        $\land in\_mempool[1] = OtherParty(party)$ CT was broadcastt by the other party
        $\land in\_mempool[2] < MaxIndex(cts)$ Revoked CT was broadcast
        $\land height - in\_mempool[2] < CSV$ Can only publish BR if CSV hasn't expired
          Record which index was published at what height
        $\land published\_ct' = \langle party, in\_mempool[2], height \rangle$
    $\land$ UNCHANGED $\langle alice\_cts, bob\_cts, alice\_brs, bob\_brs,$
                      $mempool\_ct, index \rangle$

$Next \triangleq$
    $\lor \exists\, d \in \{-1, 1\} : SupersedeCommitmentTx(d)$
    $\lor \exists\, p \in Party, h \in 0\, ..\, Height : PublishCommitment(p, h)$
    $\lor \exists\, p \in Party, h \in 0\, ..\, Height, m \in mempool\_ct : PublishBR(p, m, h)$

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