MODULE *BitcoinTransactions*

This spec captures the actions and states of bitcoin transactions in the context of the bitcoin blockchain. These actions will be used by the *LN* Contracts spec and other layer two contract specifications.

The focus of this module is to provide:

1. Way to generate transactions that accept input and generate outputs

2. Confirm transactions so that outputs can be spent.

3. Most importantly - provide a way to verify spend conditions without building the entire cryptography machinery. This enables spec authors to focus on what the conditions achieve instead of how those conditions are achieved.

Goal A: Move environment / bitcoin transaction actions and variables from Contracts to here

EXTENDS *Sequences*,
Integers

Define constants so that we can define finite sets for inputs, outputs and txids etc.

CONSTANTS *CSV*, Set of *CSV* values
VOUT, Set of *vout* values
TXID, Set of transaction ids
AMOUNT, Set of amounts that can be used
KEY, Set of all keys used for signatures
HASH Set of all hash preimages

$NoTxId \triangleq \text{CHOOSE } t : t \notin TXID$

$NoHash \triangleq \text{CHOOSE } h : h \notin HASH$

$SighashFlag \triangleq \{\text{“all”}, \text{“none”}, \text{“single”}, \text{“anyonecanpay”}\}$

$Input \triangleq [$
$\quad tx : TXID,$
$\quad vout : VOUT,$
$\quad sighash\_flag : SighashFlag,$ Parts of transactions covered by signature
$\quad signed\_by : Seq(KEY),$ One or more keys that have signed this input
$\quad hash\_preimage : HASH \cup \{NoHash\}$
$]$

Set of output types supported for building contracts.

Each output type will have to provide a means to verify an input trying to spend it.

$OutputTypes \triangleq \{\text{“p2wkh”}, \text{“multisig”}, \text{“multisig\_with\_csv”}, \text{“hash\_lock”}\}$

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

$Output \triangleq [$
$\quad type : OutputTypes,$

$csv : CSV \cup \{NoCSV\},$
$hash : HASH \cup \{NoHash\},$
$amount : AMOUNT$
]

$Tx \triangleq$ [
$id : TXID,$ A *TxID* breaks circular reference with *Input*
$inputs : Seq(Input),$ *Seq* instead of Set cause index is used for points
$outputs : Seq(Output)$
]

VARIABLES
$chain\_height,$
$mempool,$
$published$

$vars \triangleq \langle chain\_height,\ mempool,\ published \rangle$

$Init \triangleq$
$\land\ chain\_height = 1$
$\land\ mempool = \{\}$
$\land\ published = \{\}$

$TypeOK \triangleq$
$\land\quad mempool \in \text{SUBSET } Tx$
$\land\quad published \in \text{SUBSET } Tx$

$CreateP2PKHOutput(key,\ amount) \triangleq$ [
$type \mapsto$ "p2wkh",
$csv \mapsto NoCSV,$
$hash \mapsto NoHash,$
$amount \mapsto amount$
]

$CreateCoinbaseTx(txid,\ key,\ amount) \triangleq$ [
$id \mapsto txid,$
$inputs \mapsto \langle\rangle,$
$outputs \mapsto \langle CreateP2PKHOutput(key,\ amount) \rangle$
]

$AddCoinbaseToMempool(tx) \triangleq$
$\land\ tx \notin mempool$

$\land\ tx \notin published$
$\land\ mempool' = mempool \cup \{tx\}$
$\land$ UNCHANGED $\langle chain\_height,\ published \rangle$

$ConfirmMempoolTx\ \triangleq$
$\exists\, tx \in mempool :$
$\land\ tx \notin published$
$\land\ published' = published \cup \{tx\}$
$\land\ mempool' = mempool \setminus \{tx\}$
$\land\ chain\_height' = chain\_height + 1$

$Next\ \triangleq$
$\lor \exists\, k \in KEY,\ id \in TXID,\ a \in AMOUNT :$
$AddCoinbaseToMempool(CreateCoinbaseTx(id,\ k,\ a))$
$\lor\ ConfirmMempoolTx$

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