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,
TLC,
SequencesExt,
FiniteSetsExt

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
PARTY,	Parties participating in the *L2* protocol
KEY,	Set of keys for each party used in the *L2* protocol
HASH	Set of all hash preimages

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

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.

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

$NoCSV \triangleq \text{CHOOSE } c : c \notin CSV$
$MaxCSV \triangleq \text{CHOOSE } c \in CSV : \forall\, y \in CSV : c \geq y$
$NoHash \triangleq \text{CHOOSE } h : h \notin HASH$
$NoSpendHeight \triangleq -1$

All keys available for use by the parties
$Keys \triangleq PARTY \times KEY$

$Input \triangleq [$
$\quad txid : TXID,$

```
    index : VOUT,
    sighash_flag : SighashFlag,          Parts of transactions covered by signature
    signed_by : Seq(Keys),               One or more keys that have signed this input
    hash_preimage : HASH ∪ {NoHash}
]

Output ≜ [
    index : VOUT,
    type : OutputTypes,
    keys : Seq(Keys),                    Sig from these keys is required to spend
    csv : CSV ∪ {NoCSV},                 The CSV should have expired before spend
    hash : HASH ∪ {NoHash},              Pre-image required to spend
    amount : AMOUNT
]
```

```
VARIABLES
    chain_height,
    transactions,
    mempool,
    published
```

```
CreateP2WKHOutput(keys, amount) ≜ [
    index ↦ 1,
    type ↦ "p2wkh",
    keys ↦ keys,
    csv ↦ NoCSV,
    hash ↦ NoHash,
    amount ↦ amount
]

CreateMultisigOutput(keys, amount) ≜ [
    index ↦ 1,
    type ↦ "multisig",
    keys ↦ keys,
    csv ↦ NoCSV,
    hash ↦ NoHash,
    amount ↦ amount
]

CreateMultisigWithCSVOutput(keys, amount) ≜ [
    index ↦ 1,
    type ↦ "multisig_with_csv",
    keys ↦ keys,
    csv ↦ MaxCSV,
```

$hash \mapsto NoHash,$
$amount \mapsto amount$
]

Create a transaction spending the given output/id, and spendable by the given key.

$CreateP2WKHTx(spending\_output,\ id,\ output\_key,\ amount) \triangleq$ [
$inputs \mapsto \langle[txid \mapsto spending\_output[1],$
$index \mapsto spending\_output[2],$
$sighash\_flag \mapsto$ "all",
$signed\_by \mapsto transactions[spending\_output[1]].outputs[spending\_output[2]].keys,$
$hash\_preimage \mapsto NoHash]\rangle,$
$outputs \mapsto \langle CreateP2WKHOutput(output\_key,\ amount)\rangle$
]

Create a transaction spending the given output/id, and spendable by as a *multisig* of the given keys.

$CreateMultisigTx(spending\_output,\ id,\ output\_keys,\ amount) \triangleq$ [
$inputs \mapsto \langle[txid \mapsto spending\_output[1],$
$index \mapsto spending\_output[2],$
$sighash\_flag \mapsto$ "all",
$signed\_by \mapsto transactions[spending\_output[1]].outputs[spending\_output[2]].keys,$
$hash\_preimage \mapsto NoHash]\rangle,$
$outputs \mapsto \langle CreateMultisigOutput(output\_keys,\ amount)\rangle$
]

$CreateMultisigWithCSVTx(spending\_output,\ id,\ output\_keys,\ amount) \triangleq$ [
$inputs \mapsto \langle[txid \mapsto spending\_output[1],$
$index \mapsto spending\_output[2],$
$sighash\_flag \mapsto$ "all",
$signed\_by \mapsto transactions[spending\_output[1]].outputs[spending\_output[2]].keys,$
$hash\_preimage \mapsto NoHash]\rangle,$
$outputs \mapsto \langle CreateMultisigWithCSVOutput(output\_keys,\ amount)\rangle$
]

Choose keys to use in outputs. It is used by *AddSpendTxToMempool*.

We expect both this expression and the *AddSpendTxToMempool* action to be provided by the layer 2 protocol spec.

$ChooseOutputKeys(output\_type) \triangleq$
IF $output\_type =$ "p2wkh"
THEN $SetToSeq(\text{CHOOSE } k \in kSubset(1,\ Keys) : \text{TRUE})$
ELSE $SetToSeq(\text{CHOOSE } k \in kSubset(2,\ Keys) : \text{TRUE})$

Return TRUE if the given output uses a key for the provided party

$OutputOwnedByParty(o,\ p) \triangleq$
$p \in \{k[1] : k \in ToSet(transactions[o[1]].outputs[o[2]].keys)\}$

$ConfirmedTransactions \triangleq$
$\{p \in \text{DOMAIN } transactions : published[p] \neq NoSpendHeight\}$

$AllOutputs \triangleq$
$\text{UNION } \{\{txid\} \times \{o.index : o \in ToSet(transactions[txid].outputs)\} : txid \in ConfirmedTransactions\}$

$SpentOutputs \triangleq$
$\{\langle i.txid,\ i.index\rangle : i \in$
$\text{UNION } \{ToSet(transactions[txid].inputs) : txid \in ConfirmedTransactions\}$
$\}$

$UnspentOutputs \triangleq AllOutputs \setminus SpentOutputs$

Add a coinbase *tx* spendable with a pk. No verification is required here as no prevout is being spent.

$AddP2WKHCoinbaseToMempool(id,\ keys,\ amount) \triangleq$
$\land\ id \notin mempool$
$\land\ published[id] = NoSpendHeight$
$\land\ transactions' = [transactions \text{ EXCEPT } ![id] = [inputs \mapsto \langle\rangle,$
$outputs \mapsto \langle CreateP2WKHOutput(keys,\ amount)\rangle]]$
$\land\ mempool' = mempool \cup \{id\}$
$\land \text{ UNCHANGED } \langle chain\_height,\ published\rangle$

Add a coinbase *tx* with a *multisig* output spendable by signature from all keys.

We don't do threshold signatures for simplicity.

$AddMultisigCoinbaseToMempool(id,\ keys,\ amount) \triangleq$
$\land\ id \notin mempool$
$\land\ published[id] = NoSpendHeight$
$\land\ transactions' = [transactions \text{ EXCEPT } ![id] = [inputs \mapsto \langle\rangle,$
$outputs \mapsto \langle CreateMultisigOutput(keys,\ amount)\rangle]]$
$\land\ mempool' = mempool \cup \{id\}$
$\land \text{ UNCHANGED } \langle chain\_height,\ published\rangle$

Confirm transaction from *mempool*.

$ConfirmMempoolTx(id) \triangleq$
$\land\ id \in mempool$
$\land\ published[id] = NoSpendHeight$
$\land \text{ LET } tx \triangleq transactions[id]$
IN
$\land\ chain\_height' = chain\_height + 1$ Each *tx* is in it's own block
$\land\ published' = [published \text{ EXCEPT } ![id] = chain\_height']$
$\land\ mempool' = mempool \setminus \{id\}$
$\land \text{ UNCHANGED } \langle transactions\rangle$

Add a new transaction to *mempool*.

The transaction is created and added to *mempool*.

The transaction is constructed such that it is a valid transaction.

input_type specifies the type of published output to select to spend.

output_type specifies the type of new output to create.

$$
\begin{array}{l}
AddSpendTxToMempool(id,\ amount,\ input\_type,\ output\_type) \triangleq \\
\quad \exists\, spending\_output \in UnspentOutputs : \\
\qquad \land\ id \notin mempool \\
\qquad \land\ transactions[spending\_output[1]].outputs[spending\_output[2]].type = input\_type \\
\qquad \land\ transactions' = [transactions \text{ EXCEPT } ![id] = \\
\qquad\qquad \text{CASE } (output\_type = \text{“p2wkh”}) \rightarrow \\
\qquad\qquad\qquad\qquad CreateP2WKHTx(spending\_output,\ id, \\
\qquad\qquad\qquad\qquad\quad ChooseOutputKeys(output\_type),\ amount) \\
\qquad\qquad\quad \square\ (output\_type = \text{“multisig”}) \rightarrow \\
\qquad\qquad\qquad\qquad CreateMultisigTx(spending\_output,\ id, \\
\qquad\qquad\qquad\qquad\quad ChooseOutputKeys(output\_type),\ amount) \\
\qquad\qquad\quad \square\ (output\_type = \text{“multisig\_with\_csv”}) \rightarrow \\
\qquad\qquad\qquad\qquad CreateMultisigWithCSVTx(spending\_output,\ id, \\
\qquad\qquad\qquad\qquad\quad ChooseOutputKeys(output\_type),\ amount) \\
\qquad\quad] \\
\qquad \land\ mempool' = mempool \cup \{id\} \\
\qquad \land\ \text{UNCHANGED } \langle chain\_height,\ published \rangle
\end{array}
$$