MODULE *BlockGeneration*

Block generation specifies when and how braidpool miners generate blocks. The protocol to build current pool key and threshold signatures is assumed

EXTENDS
$\quad Sequences,$
$\quad Integers,$
$\quad DAG$

CONSTANT

$Miner,$	Set of miners
$ShareSeqNo,$	Share seq numbers each miner generates
$BlockReward$	Block reward in a difficulty period

VARIABLES

TODO: Replace these *last_.∗ variables with operators on DAG*

$last\_sent,$	*Function mapping miner to last sent share seq_no*
$share\_dag,$	*A DAG with valid shares for now implemented as a set*
$unpaid\_coinbases,$	*coinbases for braidpool blocks that* *haven t* been paid yet
$uhpo,$	Function mapping miner to unpaid balance
$pool\_key$	Current public key for *TS*

Share is a record of miner and sequence number. All shares are assumed to be mined at same difficulty

$Share \triangleq [miner : Miner, seq\_no : ShareSeqNo]$

Acks are the implicit acknowledgements sent with each share. These are the sequence number of shares receieved from each miner.

$Acks \triangleq \langle Share \rangle$

ShareDAG is used to track paths between shares

$ShareDAG \triangleq [node : Share, edge : Share \times Share]$

PublicKey is defined as sequence of miner identifier for now. The miners in a key sequence are the miners contributing to the key generated using *DKG*.

$PublicKey \triangleq Seq(Miner)$

Coinbase is a payment to a *DKG* public key with an value.

$Coinbase \triangleq [key : PublicKey, value : BlockReward]$

$NoVal \triangleq 0$

$Init \triangleq$

$\quad \land\ last\_sent = [m \in Miner \mapsto NoVal]$

```
        $\land\ share\_dag = [node \mapsto \{\},\ edge \mapsto \{\}]$
        $\land\ unpaid\_coinbases = \{\}$
        $\land\ uhpo = [m \in Miner \mapsto NoVal]$
        $\land\ pool\_key = \langle\rangle$

$TypeInvariant\ \triangleq$
        $\land\ last\_sent \in [Miner \rightarrow Int \cup \{NoVal\}]$
        $\land\ share\_dag.node \in \text{SUBSET}\ Share$
        $\land\ share\_dag.edge\ \in \text{SUBSET}\ (Share \times Share)$
        $\land\ unpaid\_coinbases \in \text{SUBSET}\ Coinbase$
        $\land\ uhpo \in [Miner \rightarrow Int \cup \{NoVal\}]$
        $\land\ pool\_key \in Seq(Miner)$

$vars\ \triangleq\ \langle last\_sent,\ share\_dag,\ unpaid\_coinbases,\ uhpo,\ pool\_key\rangle$
```

Send a share from a miner with a *seqno* = last share sent + 1 and in *ShareSeqNo*. The share is assumed to be successfully broadcast to all miners.

```
$SendShare\ \triangleq\ \exists\, m \in Miner,\ sno \in ShareSeqNo :$
            $\land\ sno = last\_sent[m] + 1$
            $\land\ last\_sent'\ = [last\_sent\ \text{EXCEPT}\ ![m] = @ + 1]$
            $\land\ share\_dag' = [share\_dag\ \text{EXCEPT}$
                                  Add share to node list of graph
                                 $!.node = @ \cup \{[miner \mapsto m,\ seq\_no \mapsto sno]\},$
                                  Add edge from share to all non $NoVal$ $last\_sent$
                                  This can be replaced by last share in $DAG$ from others
                                 $!.edge = @ \cup$
                                     $\{[miner \mapsto m,\ seq\_no \mapsto sno]\}$
                                     $\times$
                                     $\{[miner \mapsto mo,\ seq\_no \mapsto last\_sent[mo]] :$
                                             $mo \in \{mm \in Miner : last\_sent[mm] \neq NoVal\}\}]$
            $\land\ \text{UNCHANGED}\ \langle unpaid\_coinbases,\ uhpo,\ pool\_key\rangle$
```

StabiliseShare

RecvBitcoinBlock

FindBitcoinBlock

UpdatePoolKey

```
$Next\ \triangleq$
        $\lor\ SendShare$

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