MODULE *LocalBlockChain*

This is for the block chain info in one node. The very basic constraint here is that there is only one root node, *i.e.*, the *genisis* node. Another two constains shall be :
1. Two final blocks must be on the same branch;
2. Two blocks with the same proposer, or committer, must be on the same branch. In the *PoD* implementation, the $1st$ constraint is true for good node, while the $2nd$ can be archieved by deposit.

EXTENDS *Naturals*, *Sequences*

CONSTANT *Validator*

VARIABLE *UsedIds*

VARIABLES *Self*

VARIABLE *block*,
*block_prepares*,
*block_commits*

$$\begin{aligned} Block \triangleq\ & [parent : Nat, \\ & id : Nat, \\ & proposer : Validator, \\ & state : \{\text{“init”}, \text{“prepared”}, \text{“committed”}\}] \\ & \cup \\ & [val : \text{“genisis”}, state : \text{“committed”}, id : 0, proposer : \text{“0”}] \end{aligned}$$

********** Helpers **********

$$NextBlock[n \in block] \triangleq \text{CHOOSE } v \in block : v.parent = n.id$$

$$PrevBlock[n \in block] \triangleq \text{CHOOSE } v \in block : n.parent = v.id$$

$$\begin{aligned} TailBlock[n \in block] \triangleq\ & \text{IF } \exists\, t \in block : t.parent = n.id \\ & \text{THEN } TailBlock[NextBlock[n]] \\ & \text{ELSE } n \end{aligned}$$

$$\begin{aligned} FollowingBlock[n \in block] \triangleq\ & \{n\} \cup \text{IF } \exists\, t \in block : t.parent = n.id \\ & \text{THEN } FollowingBlock[NextBlock[n]] \\ & \text{ELSE } \{\} \end{aligned}$$

$$GenisisBlock \triangleq [val \mapsto \text{“genisis”}, state \mapsto \text{“committed”}, id \mapsto 0, proposer \mapsto \text{“0”}]$$

$$AllTails \triangleq \{t \in block : TailBlock[t] = t\}$$

$$\begin{aligned} ChainWithTail[n \in block] \triangleq\ & \text{IF } n.id = 0 \\ & \text{THEN } \{n\} \\ & \text{ELSE } \{n\} \cup ChainWithTail[PrevBlock[n]] \end{aligned}$$

$$AllChains \triangleq \{ChainWithTail[t] : t \in AllTails\}$$

$BlockContributer[n \in block] \triangleq \text{IF } n.id = 0 \text{ THEN } \{\}$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{ELSE } \{n.proposer\} \cup n.prepares \cup \{n.commits\}$

$BlockPrepares[n \in block] \triangleq block\_prepares[n.id]$
$BlockCommits[n \in block] \triangleq block\_commits[n.id]$

*FindContributedTail*. We don't have this helper for performance reason.

---

********** Actions **********

$BCTypeOK \triangleq \land block \subseteq Block$
$\qquad\qquad\qquad\quad \land Self \in Validator$

$BCInit \triangleq \land block = \{GenisisBlock\}$
$\qquad\qquad\quad \land block\_prepares = [id \in Nat \mapsto \{\}]$
$\qquad\qquad\quad \land block\_commits = [id \in Nat \mapsto \{\}]$

$BCGenBlockWithTail(tail) \triangleq [parent \mapsto tail.id,$
$\qquad\qquad\qquad\qquad id \mapsto \text{CHOOSE } t : t \in Nat \land t \notin UsedIds,$
$\qquad\qquad\qquad\qquad proposer \mapsto Self,$
$\qquad\qquad\qquad\qquad state \mapsto \text{“init”}]$

$BCAddBlock(b) \triangleq block \cup \{b\}$

$BCChangeBlockStatus(b, new\_status) \triangleq block' = block \setminus \{b\}$

$BCPrepareBlock(b, v) \triangleq [block\_prepares \text{ EXCEPT } ![b.id] = block\_prepares[b.id] \cup \{v\}]$

$BCCommitBlock(b, v) \triangleq [block\_commits \text{ EXCEPT } ![b.id] = block\_commits[b.id] \cup \{v\}]$

---

********** Consistency **********

For a block chain, the final status consistency requirement is like this:
If two blocks are committed, they must be on the same branch.

$BCFinalStatusConsistency \triangleq \forall a, b \in \{t \in block : t.state = \text{“committed”}\} :$
$\qquad\qquad\qquad\qquad\qquad\qquad FollowingBlock[a] \cap FollowingBlock[b] \neq \{\}$

We define a block's contributor as the validator who propose, prepare or commit
the block. Then the requirement is like this:
For any two blocks, a and $b$, in which a and $b$ belong to different branches,
a's contributer must not be $b$'s contributer.

$BCContributerConsistency \triangleq \forall ch1, ch2 \in AllChains :$
$\qquad\qquad\qquad\qquad\qquad\qquad \text{LET } dch1 \triangleq ch1 \setminus (ch1 \cap ch2)$
$\qquad\qquad\qquad\qquad\qquad\qquad\quad dch2 \triangleq ch2 \setminus (ch1 \cap ch2)$
$\qquad\qquad\qquad\qquad\qquad\qquad \text{IN } \forall b1 \in dch1, b2 \in dch2 :$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad BlockContributer[b1] \cap BlockContributer[b2] = \{\}$

$BCConsistency \triangleq \lor BCFinalStatusConsistency$

$\vee$ *BCContributerConsistency*

\ * Modification History
\ * Last modified Sat *Feb* 03 17:47:34 *CST* 2018 by *xuepeng*
\ * Created Sat *Jan* 27 16:13:27 *CST* 2018 by *xuepeng*