MODULE *DAGConsensus*

Specification, at a high level of abstraction, of a very simple DAG-based *BFT* consensus protocol.

Model-checking with *TLC* seems intractable beyond 4 rounds, even with sequentialization.

EXTENDS *DomainModel*

```
  --algorithm $DAGConsensus${
    variables
        $vs = \{\}$,    the vertices of the $DAG$
        $es = \{\}$ ;   the edges of the $DAG$
    define {
        $VerticeQuorums(r) \triangleq$
            $\{VQ \in \text{SUBSET } vs :$
                   $\land\ \forall v \in VQ : Round(v) = r$
                   $\land\ \{Node(v) : v \in VQ\} \in Quorum\}$
        $Committed(v) \triangleq$
            $\land\ v \in vs$
            $\land\ Round(v)\%2 = 0$
            $\land\ Node(v) = Leader(Round(v))$
            $\land\ \{Node(p) : p \in Parents(v, es)\} \in Quorum$
        $Correctness \triangleq \forall v1, v2 \in vs :$
            $\land\ Committed(v1)$
            $\land\ Committed(v2)$
            $\land\ Round(v1) \leq Round(v2)$
            $\Rightarrow Reachable(v2, v1, es)$
     }
    process ( $node \in N$ )
        variables
            $round = 0$ ;   current round
    {
l0:     while ( TRUE ) {
            either with ( $v = \langle self, round \rangle$ ) {
                 add a new vertex to the $DAG$ and go to the next round:
                $vs := vs \cup \{v\}$ ;
                if ( $round > 0$ )
                with ( $vq \in VerticeQuorums(round - 1)$ )
                    $es := es \cup \{\langle v, pv \rangle : pv \in vq\}$ ;
                $round := round + 1$
            }
           or {
                 join a higher round
                with ( $r \in \{r \in R : r > round\}$ )
                    $round := r$
            }
```

$\qquad\quad \}$

$\qquad \}$

$\}$

$TypeOK \triangleq$

$\quad \land\ \forall v \in vs : Node(v) \in N \land Round(v) \in Nat$

$\quad \land\ \forall e \in es :$

$\qquad\quad \land\ e = \langle e[1],\ e[2] \rangle$

$\qquad\quad \land\ \{e[1],\ e[2]\} \subseteq vs$

$\qquad\quad \land\ Round(e[1]) > Round(e[2])$

$\quad \land\ \forall n \in N : round[n] \in Nat$

Model-checking stuff:

Sequentialization constraints, which enforce a particular ordering of the actions. Because of how actions commute, the set of reachable states remains unchanged.

$SeqConstraints(n) \triangleq$

wait for all nodes to finish previous rounds:

$\quad \land (round[n] > 0 \Rightarrow \forall n2 \in N : round[n2] \geq round[n])$

wait for all nodes with lower index to leave the round:

$\quad \land \forall n2 \in N : NodeIndex(n2) < NodeIndex(n) \Rightarrow round[n2] > round[n]$

$SeqNext \triangleq (\exists self \in N : SeqConstraints(self) \land node(self))$

$SeqSpec \triangleq Init \land \Box[SeqNext]_{vars}$

Example assignment of leaders to rounds (changes every 2 rounds):

$ModLeader(r) \triangleq NodeSeq[((r \div 2) \% Cardinality(N)) + 1]$

$StateConstraint \triangleq$

$\quad \text{LET } Max(S) \triangleq \text{CHOOSE } x \in S : \forall y \in S : y \leq x\text{IN}$

$\qquad \forall n \in N : round[n] \in 0 \,..\, (Max(R) + 1)$

$Falsy1 \triangleq \neg($

$\quad \exists v1,\ v2 \in vs :$

$\qquad \land\ v1 \neq v2$

$\qquad \land\ Committed(v1)$

$\qquad \land\ Committed(v2)$

$)$

$Falsy2 \triangleq \neg($

$\quad Committed(\langle Leader(2),\ 2 \rangle)$

$)$