MODULE *ReliableBroadcast*

EXTENDS *LearnerGraph*, *FiniteSets*

CONSTANTS

LG, the learner graph

B, the set of malicious acceptors

W, the set of well-behaved acceptors, *i.e.* honest and available

V the set of values that can be broadcast

ASSUME $B \cap W = \{\}$

ASSUME $IsValidLearnerGraph(LG)$

$Learner \triangleq LG.learners$
$Acceptor \triangleq LG.acceptors$

$HonestAcceptor \triangleq Acceptor \setminus B$
Note that *HonestAcceptor* is not necessary equal to W

```
--algorithm ReliableBroadcast{
  variables
    bcast ∈ (SUBSET V) \ {{}} ;   the value(s) broadcast; multiple values model a malicious sender
    echo = [a ∈ Acceptor ↦ {}] ;
    ready = [a ∈ Acceptor ↦ [l ∈ Learner ↦ {}]] ;
  define {
    ProvenMalicious(a) ≜ ∃ v1, v2 ∈ V :
        ∧  v1 ≠ v2
        ∧  ∨  {v1, v2} ⊆ echo[a]
           TODO: this is recursive:
           ∨ ∃ l1,l2 ∈ Learner :
              ∧ l1 ≠ l2
              ∧ v1 ∈ ready[a][l1]
              ∧ v2 ∈ ready[a][l2]
              ∧ ¬NotEntangled(l1, l2)
    NotEntangled(l1, l2) ≜
        ∧  l1 ≠ l2  a learner is always entangled with itself
        ∧  ∀ S ∈ LG.safeSets[⟨l1, l2⟩] :
              ∃ a ∈ S : ProvenMalicious(a)
  }
  fair process ( learner ∈ Learner )
    variables
      output = ⟨⟩ ;
  {
l0:  with ( v ∈ V ) {
       when ∃ Q ∈ LG.quorums[self] :
```

$\forall\, a \in Q : v \in ready[a][self]$ **;**

$output := v$ **;**

}

}

```
    process ( acceptor ∈ HonestAcceptor ) {
l0:     while ( TRUE )
        either
            with ( v ∈ V ) {
                when v ∈ bcast ∧ echo[self] = {} ;
                echo[self] := echo[self] ∪ {v} ;
            }
        or
            with ( v  ∈ V )
            with ( l  ∈ Learner )
            with ( Q ∈ LG.quorums[l] ) {
                when ready[self][l] = {} ;
                when ∀ a ∈ Q : v  ∈ echo[a] ;
                 check for conflicts:
                when ∀ l2 ∈ Learner : ∀ v2 ∈ V \ {v} :
                    v2 ∈ ready[self][l2] ⇒ NotEntangled(l, l2) ;
                ready[self][l] := ready[self][l] ∪ {v} ;
            }
        or
            with ( v ∈ V )
            with ( l1 ∈ Learner, l2 ∈ Learner ) {
                when ∀ Q ∈ LG.quorums[l1] : ∃ a2 ∈ Q :
                    ∧ v  ∈ ready[a2][l2]
                    ∧ v  ∈ bcast ;
                     and we need a proof for the ready message:
                      ∧ ∃ Q2 ∈ LG.quorums[l2] : ∀ a3 ∈ Q2 : v ∈ echo[a3];
                 check for conflicts:
                when ∀ l3 ∈ Learner \ {l1} : ∀ v2 ∈ V \ {v} :
                    v2 ∈ ready[self][l3] ⇒ NotEntangled(l1, l3) ;
                ready[self][l1] := ready[self][l1] ∪ {v} ;
            }
      }
    process ( byzAcceptor ∈ B ) {
l0:     while ( TRUE ) {
            either
            with ( v ∈ V )
                echo[self] := echo[self] ∪ {v}
            or
            with ( l ∈ Learner ) {
                with ( v ∈ V )
                    ready[self][l] := ready[self][l] ∪ {v} ;
            }
        }
    }
```

```
}
```

$TypeOK \triangleq$
$\land\ bcast \in (\text{SUBSET}\ V) \setminus \{\{\}\}$
$\land\ echo \in [Acceptor \to (\text{SUBSET}\ V)]$
$\land\ ready \in [Acceptor \to [Learner \to (\text{SUBSET}\ V)]]$
$\land\ output \in [Learner \to V \cup \{\langle\rangle\}]$

Two learners must agree if one of their safe sets is fully well-behaved:
$Entangled(l1,\ l2) \triangleq \exists\, S \in LG.safeSets[\langle l1,\ l2\rangle] :$
$S \cap B = \{\}$

$LiveLearner \triangleq \{l \in Learner :$
$\exists\, Q \in LG.quorums[l] : Q \subseteq W\}$

$Safety \triangleq$
$\land\ \forall\, l \in Learner :$
$\land\ pc[l] = \text{“Done”}$
$\land\ \exists\, Q \in LG.quorums[l] : Q \cap B = \{\}$ *SafeLearner*
$\Rightarrow output[l] \in bcast$
$\land\ \forall\, l1,\ l2 \in Learner :$
$\land\ Entangled(l1,\ l2)$
$\land\ pc[l1] = \text{“Done”}$
$\land\ pc[l2] = \text{“Done”}$
$\Rightarrow output[l1] = output[l2]$

$Liveness \triangleq$
$\land\ Cardinality(bcast) = 1 \Rightarrow$
$\forall\, l \in LiveLearner : \Diamond(pc[l] = \text{“Done”} \land bcast = \{output[l]\})$

This one is interesting (I think this is the best we can guarantee):
$\land\ \forall\, l1 \in Learner : \forall\, l2 \in LiveLearner : Entangled(l1,\ l2) \Rightarrow$
$\Box(pc[l1] = \text{“Done”} \Rightarrow \Diamond(pc[l2] = \text{“Done”}))$

$FairSpec \triangleq$
$\land\ Spec$
$\land\ \forall\, a \in W : \mathrm{WF}_{vars}(acceptor(a))$