

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

$bcast \in (\text{SUBSET } V) \setminus \{\{\}\}$; the value(s) broadcast; multiple values model a malicious sender

$echo = [a \in Acceptor \mapsto \{\}];$

$ready = [a \in Acceptor \mapsto [l \in Learner \mapsto \{\}]];$

define {

$ProvenMalicious(a) \triangleq \exists v1, v2 \in V :$

$\wedge v1 \neq v2$

$\wedge \{v1, v2\} \subseteq echo[a]$

$NotEntangled(l1, l2) \triangleq$

$\wedge l1 \neq l2$ a learner is always entangled with itself

$\wedge \forall S \in LG.safeSets[\langle l1, l2 \rangle] :$

$\exists a \in S : ProvenMalicious(a)$

}

fair process ($learner \in Learner$)

variables

$output = \langle \rangle;$

{

l0: **with** ($v \in V$) {

when $\exists Q \in LG.quorums[self] :$

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

$output := v;$

}

}

```

    process ( acceptor  $\in$  HonestAcceptor ) {
l0:   while ( TRUE )
      either
        with (  $v \in V$  ) {
          when  $v \in bcast \wedge echo[self] = \{\}$ ;
           $echo[self] := echo[self] \cup \{v\}$ ;
        }
      or
        with (  $v \in V$  )
        with (  $l \in Learner$  )
        with (  $Q \in LG.quorums[l]$  ) {
          when  $ready[self][l] = \{\}$ ;
          when  $\forall a \in Q : v \in echo[a]$ ;
          check for conflicts:
          when  $\forall l2 \in Learner : \forall v2 \in V \setminus \{v\} :$ 
             $v2 \in ready[self][l2] \Rightarrow NotEntangled(l, l2)$ ;
           $ready[self][l] := ready[self][l] \cup \{v\}$ ;
        }
      or
        with (  $v \in V$  )
        with (  $l1 \in Learner, l2 \in Learner$  ) {
          when  $v \in bcast$ ;
          when  $\forall Q \in LG.quorums[l1] :$ 
             $\exists a2 \in Q : v \in ready[a2][l2]$ ;
          check for conflicts:
          when  $\forall l3 \in Learner \setminus \{l1\} : \forall v2 \in V \setminus \{v\} :$ 
             $v2 \in ready[self][l3] \Rightarrow NotEntangled(l1, l3)$ ;
           $ready[self][l1] := ready[self][l1] \cup \{v\}$ ;
        }
      }
    }
  }
}

```

$$\begin{aligned}
TypeOK &\triangleq \\
&\wedge \text{bcast} \in (\text{SUBSET } V) \setminus \{\{\}\} \\
&\wedge \text{echo} \in [\text{Acceptor} \rightarrow (\text{SUBSET } V)] \\
&\wedge \text{ready} \in [\text{Acceptor} \rightarrow [\text{Learner} \rightarrow (\text{SUBSET } V)]] \\
&\wedge \text{output} \in [\text{Learner} \rightarrow V \cup \{\langle \rangle\}]
\end{aligned}$$

Two learners must agree if one of their safe sets is fully well-behaved:

$$\begin{aligned}
Entangled(l1, l2) &\triangleq \exists S \in LG.\text{safeSets}[\langle l1, l2 \rangle] : \\
&S \cap B = \{\}
\end{aligned}$$

$$\begin{aligned}
LiveLearner &\triangleq \{l \in \text{Learner} : \\
&\exists Q \in LG.\text{quorums}[l] : Q \subseteq W\}
\end{aligned}$$

$$\begin{aligned}
Safety &\triangleq \\
&\wedge \forall l \in \text{Learner} : \\
&\quad \wedge pc[l] = \text{"Done"} \\
&\quad \wedge \exists Q \in LG.\text{quorums}[l] : Q \cap B = \{\} \text{ SafeLearner} \\
&\quad \Rightarrow output[l] \in \text{bcast} \\
&\wedge \forall l1, l2 \in \text{Learner} : \\
&\quad \wedge Entangled(l1, l2) \\
&\quad \wedge pc[l1] = \text{"Done"} \\
&\quad \wedge pc[l2] = \text{"Done"} \\
&\quad \Rightarrow output[l1] = output[l2]
\end{aligned}$$

$$\begin{aligned}
Liveness &\triangleq \\
&\wedge Cardinality(\text{bcast}) = 1 \Rightarrow \\
&\quad \forall l \in LiveLearner : \Diamond(pc[l] = \text{"Done"} \wedge \text{bcast} = \{output[l]\}) \\
&\quad \text{This one is interesting (I think this is the best we can guarantee):} \\
&\wedge \forall l1 \in \text{Learner} : \forall l2 \in LiveLearner : Entangled(l1, l2) \Rightarrow \\
&\quad \Box(pc[l1] = \text{"Done"} \Rightarrow \Diamond(pc[l2] = \text{"Done"}))
\end{aligned}$$

$$\begin{aligned}
FairSpec &\triangleq \\
&\wedge Spec \\
&\wedge \forall a \in W : WF_{vars}(acceptor(a))
\end{aligned}$$
