

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 {

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

$\wedge v1 \neq v2$

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

$\vee \exists l \in Learner : \{v1, v2\} \subseteq ready[a][l]$

 }

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 ∈ 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] ;
          ready[self][l] := ready[self][l] ∪ {v} ;
        }
      or
        with ( v ∈ V )
        with ( l ∈ Learner )
        with ( readyForV = {a ∈ Acceptor : v ∈ ready[a][l]} ) {
          when ∀ Q ∈ LG.quorums[l] :
            ∨ Q ∩ readyForV ≠ {}
            ∨ ∀ a ∈ Q : ProvenMalicious(a) ;
          ready[self][l] := ready[self][l] ∪ {v} ;
        }
      }
    }
    process ( byzAcceptor ∈ B ) {
l0:   while ( TRUE ) {
      with ( v ∈ V )
        echo[self] := echo[self] ∪ {v} ;
      with ( rdy ∈ [Learner → V] ) {
        ready[self] := [l ∈ Learner ↦ ready[self][l] ∪ {rdy[l]}] ;
      }
    }
  }
}

```

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

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

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

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

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

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

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