

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*{

Global variables:

variables

$bcast \in (\text{SUBSET } V) \setminus \{\{\}\}$; the value(s) broadcast; multiple values model a malicious sender
 $echo = [a \in Acceptor \mapsto \{\}];$ echo messages sent by the acceptors
 ready messages sent by the acceptors; note that acceptors get ready per learner:
 $ready = [a \in Acceptor \mapsto [l \in Learner \mapsto \{\}]];$
 $fd = [a \in Acceptor \mapsto \{\}];$ failure-detector output

define {

$NotEntangled(self, l1, l2) \triangleq$
 $\wedge l1 \neq l2$ a learner is always entangled with itself
 $l1$ and $l2$ are don't have to agree if none of their safe sets are completely well-behaved:
 $\wedge \forall S \in LG.safeSets[\langle l1, l2 \rangle]:$
 $\exists a \in S : a \in fd[self]$
 }

Each process get outputs from a failure-detector that eventually identifies all malicious processes (we enforce this eventuality by making the failure-detector process below a fair process):

fair process ($fd \in \{\text{"failure-detector"}\}$)

{
 l0: **while** ($\exists a \in HonestAcceptor : \exists b \in B : b \notin fd[a]$)
 with ($a \in HonestAcceptor, b \in B$) {
 when $b \notin fd[a];$
 $fd[a] := fd[a] \cup \{b\};$
 }
 }

```

fair process ( learner ∈ Learner )
  variables
    output =  $\langle \rangle$  ;
  {
l0: with ( v ∈ 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  $\forall a \in Q : v \in echo[a] ;$ 
      check for conflicts:
      when  $\forall l2 \in Learner : \forall v2 \in V \setminus \{v\} :$ 
         $v2 \in ready[self][l2] \Rightarrow NotEntangled(self, l, l2) ;$ 
      ready[self][l] := ready[self][l] ∪ {v} ;
    }
  }
  or
    with ( v ∈ V )
    with ( l1 ∈ Learner, l2 ∈ Learner ) {
      when v ∈ bcast ;
      when  $\forall Q \in LG.quorums[l1] :$ 
         $\exists a2 \in Q : v \in ready[a2][l2] ;$ 
      check for conflicts:
      when  $\forall l3 \in Learner \setminus \{l1\} : \forall v2 \in V \setminus \{v\} :$ 
         $v2 \in ready[self][l3] \Rightarrow NotEntangled(self, l1, l3) ;$ 
      ready[self][l1] := ready[self][l1] ∪ {v} ;
    }
  }
}

```

```

process ( byzAcceptor  $\in B$  ) {
l0:   while ( TRUE ) {
      either
      with ( v  $\in V$  )
        echo[self] := echo[self]  $\cup$  {v}
      or
      with ( l  $\in$  Learner ) {
        with ( v  $\in V$  )
          ready[self][l] := ready[self][l]  $\cup$  {v};
      }
    }
}

```

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

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

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

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

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

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

$FairSpec \triangleq$

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