MODULE *UniversalPaxosStore*

Specification of the consensus protocol in *PaxosStore*.

See [*PaxosStore*@*VLDB*2017](*https* : *//www.vldb.org/pvldb/vol*10/*p*1730 − *lin.pdf*) by *Tencent*.

In this version (adopted from "*PaxosStore.tla*"):

- Client-restricted config (Ballot)
− *Message* types (*i.e.*, "Prepare", "Accept", "ACK") are deleted. No state flags (such as "Prepare", "Wait-Prepare", "Accept", "Wait-Accept" are needed.

EXTENDS *Integers*, *FiniteSets*

$Max(m,\, n) \triangleq$ IF $m > n$ THEN m ELSE n
$Injective(f) \triangleq \forall\, a,\, b \in$ DOMAIN $f : (a \neq b) \Rightarrow (f[a] \neq f[b])$

CONSTANTS
    Participant, the set of partipants
    Value the set of possible input values for *Participant* to propose

$None \triangleq$ CHOOSE $b : b \notin Value$
$NP \triangleq Cardinality(Participant)$ number of $p \in$ Participants

$Quorum \triangleq \{Q \in$ SUBSET $Participant : Cardinality(Q) * 2 \geq NP + 1\}$
ASSUME $QuorumAssumption \triangleq$
    $\wedge \quad \forall\, Q \in Quorum : Q \subseteq Participant$
    $\wedge \quad \forall\, Q1,\, Q2 \in Quorum : Q1 \cap Q2 \neq \{\}$

$Ballot \triangleq Nat$

$PIndex \triangleq$ CHOOSE $f \in [Participant \to 1\,..\,NP] : Injective(f)$
$Bals(p) \triangleq \{b \in Ballot : b\%NP = PIndex[p] - 1\}$ allocate ballots for $p \in Participant$

$State \triangleq [maxBal : Ballot \cup \{-1\},$
        $maxVBal : Ballot \cup \{-1\},\ maxVVal : Value \cup \{None\}]$

$InitState \triangleq [maxBal \mapsto -1,\ maxVBal \mapsto -1,\ maxVVal \mapsto None]$

For simplicity, in this specification, we choose to send the complete state of a participant each time. When receiving such a message, the participant processes only the "partial" state it needs.

$Message \triangleq [from : Participant,\ to :$ SUBSET $Participant,\ state : [Participant \to State]]$

VARIABLES
    state, *state*[*p*][*q*]: the state of $q \in Participant$ from the view of $p \in Participant$
    msgs the set of messages that have been sent

$vars \triangleq \langle state,\, msgs \rangle$

$TypeOK \triangleq$
    $\wedge \quad state \in [Participant \to [Participant \to State]]$
    $\wedge \quad msgs \subseteq Message$

```
57  Send(m) ≜ msgs′ = msgs ∪ {m}
58  ----------------------------------------------------------------------
59  Init ≜
60      ∧ state = [p ∈ Participant ↦ [q ∈ Participant ↦ InitState]]
61      ∧ msgs = {}
    p ∈ Participant starts the prepare phase by issuing a ballot b ∈ Ballot.
65  Prepare(p, b) ≜
66      ∧  b ∈ Bals(p)
67      ∧  state[p][p].maxBal < b
68      ∧  state′ = [state EXCEPT ![p][p].maxBal = b]
69      ∧  Send([from ↦ p, to ↦ Participant, state ↦ state′[p]])
    q ∈ Participant updates its own state state[q] according to the actual state pp of p ∈ Participant
    extracted from a message m ∈ Message it receives. This is called by OnMessage(q).

    Note: pp is m.state[p]; it may not be equal to state[p][p] at the time UpdateState is called.
78  UpdateState(q, p, pp) ≜
79      state′ = [state EXCEPT
80                  ![q][p].maxBal = Max(@, pp.maxBal),
81                  ![q][p].maxVBal = Max(@, pp.maxVBal),
82                  ![q][p].maxVVal = IF state[q][p].maxVBal < pp.maxVBal
83                                    THEN pp.maxVVal ELSE @,
84                  ![q][q].maxBal = Max(@, pp.maxBal),
85                  ![q][q].maxVBal = IF state[q][q].maxBal ≤ pp.maxVBal
86                                    THEN pp.maxVBal ELSE @,    make promise
87                  ![q][q].maxVVal = IF state[q][q].maxBal ≤ pp.maxVBal
88                                    THEN pp.maxVVal ELSE @]    accept
    q ∈ Participant receives and processes a message in Message.
92  OnMessage(q) ≜
93      ∃ m ∈ msgs :
94          ∧ q ∈ m.to
95          ∧ LET p ≜ m.from
96            IN  UpdateState(q, p, m.state[p])
97          ∧ IF  ∨ m.state[q].maxBal < state′[q][q].maxBal
98                ∨ m.state[q].maxVBal < state′[q][q].maxVBal
99            THEN Send([from ↦ q, to ↦ {m.from}, state ↦ state′[q]])
100           ELSE UNCHANGED msgs
    p ∈ Participant starts the accept phase by issuing the ballot b ∈ Ballot with value v ∈ Value.
105 Accept(p, b, v) ≜
106     ∧ b ∈ Bals(p)
107     ∧ ∃ Q ∈ Quorum : ∀ q ∈ Q : state[p][q].maxBal = b
108     ∧ ∨ ∀ q ∈ Participant : state[p][q].maxVBal = −1    free to pick its own value
109       ∨ ∃ q ∈ Participant :    v is the value with the highest maxVBal
110             ∧ state[p][q].maxVVal = v
111             ∧ ∀ r ∈ Participant : state[p][q].maxVBal ≥ state[p][r].maxVBal
```

```
112       ∧ state' = [state EXCEPT ![p][p].maxVBal = b, ![p][p].maxVVal = v]  accept
113       ∧ Send([from ↦ p, to ↦ Participant, state ↦ state'[p]])
114 ----------------------------------------------------------------------------
115   Next ≜ ∃ p ∈ Participant : ∨ OnMessage(p)
116                              ∨ ∃ b ∈ Ballot : ∨ Prepare(p, b)
117                                               ∨ ∃ v ∈ Value : Accept(p, b, v)
118   Spec ≜ Init ∧ □[Next]_vars
119 ============================================================================
```

UniversalPaxosStore satisfies the *Consistency* property.

```
123   ChosenP(p) ≜  the set of values chosen by p ∈ Participant
124       {v ∈ Value : ∃ b ∈ Ballot :
125                      ∃ Q ∈ Quorum : ∀ q ∈ Q : ∧ state[p][q].maxVBal = b
126                                               ∧ state[p][q].maxVVal = v}
127   chosen ≜ UNION {ChosenP(p) : p ∈ Participant}

129   Consistency ≜ Cardinality(chosen) ≤ 1
130   THEOREM Spec ⇒ □Consistency
131 ============================================================================
```

```
\ * Modification History
\ * Last modified Wed Aug 14 20:49:53 CST 2019 by hengxin
\ * Last modified Mon Jul 22 13:59:15 CST 2019 by pure_
\ * Last modified Mon Jun 03 21:26:09 CST 2019 by stary
\ * Last modified Wed May 09 21:39:31 CST 2018 by dell
\ * Created Mon Apr 23 15:47:52 GMT + 08:00 2018 by pure_
```