MODULE *CASPaxos*

This is a high-level specification of the *CASPaxos* algorithm from the paper "*CASPaxos*: Replicated State Machines without Logs" by *Denis Rystsov*.

Please go to https://*arxiv.org*/abs/1802.07000 for the paper.

This spec is adapted from that of *Paxos* consensus algorithm by *Leslie Lamport*, which can be found at https://*github.com*/tlaplus/Examples/blob/master/specifications/*PaxosHowToWinATuringAward*/*Paxos.tla*.

Search "$\langle + \rangle$" for the code added for *CASPaxos*.

TODO: It refines the spec in module Voting.

EXTENDS *Integers*

CONSTANTS
- *Value*, the set of values to be proposed and chosen from
- *Acceptor*, the set acceptors
- *Quorum* the quorum system on acceptors

$None \triangleq$ CHOOSE $v : v \notin Value$

ASSUME $\land \forall\, Q \in Quorum : Q \subseteq Acceptor$
$\land \forall\, Q1,\, Q2 \in Quorum : Q1 \cap Q2 \neq \{\}$

$Ballot \triangleq Nat$

$\langle + \rangle$ The set of all possible *CAS* operations. The *CAS* operations with $cmpVal = None$ are initialization operations. We assume that the new values (*i.e.*, $swapVal$) are not *None*.

$CASOperation \triangleq [cmpVal : Value \cup \{None\},\, swapVal : Value]$

$Message \triangleq$ the set of all possible messages that can be sent in the algorithm
- $[type : \{\text{"1a"}\},\, bal : Ballot]$
- $\cup\ [type : \{\text{"1b"}\},\, acc : Acceptor,\, bal : Ballot,\ mbal : Ballot \cup \{-1\},\, mval : Value \cup \{None\}]$
- $\cup\ [type : \{\text{"2a"}\},\, bal : Ballot,\, val : Value]$
- $\cup\ [type : \{\text{"2b"}\},\, acc : Acceptor,\, bal : Ballot,\, val : Value]$
- $\cup\ [type : \{\text{"response"}\},\, bal : Ballot]$ $\langle + \rangle$ the messages sent to the user

VARIABLES

$maxBal[a]$: the last ballot the acceptor $a \in Acceptor$ has voted for

maxBal,

$\langle maxVBal[a],\, maxVVal[a] \rangle$ is the vote with the largest ballot cast by acceptor $a \in Acceptor$. It equals $\langle -1,\, None \rangle$ if $a \in Acceptor$ has not cast any vote.

maxVBal, *maxVVal*,

msgs, the set of all messages that have been sent

ops $\langle + \rangle ops[b]$: the *CAS* operation to be proposed at ballot $b \in Ballot$

$vars \triangleq \langle maxBal,\, maxVBal,\, maxVVal,\, msgs,\, ops \rangle$

```
TypeOK ≜ ∧ maxBal  ∈ [Acceptor → Ballot ∪ {−1}]
         ∧ maxVBal ∈ [Acceptor → Ballot ∪ {−1}]
         ∧ maxVVal ∈ [Acceptor → Value ∪ {None}]
         ∧ msgs ⊆ Message
         ∧ ops ∈ [Ballot → CASOperation]  ⟨+⟩
```

```
Init ≜ ∧ maxBal  = [a ∈ Acceptor ↦ −1]
       ∧ maxVBal = [a ∈ Acceptor ↦ −1]
       ∧ maxVVal = [a ∈ Acceptor ↦ None]
       ∧ msgs = {}
       ⟨+⟩ops remains unchanged; we utilize TLC to explore all possible CAS operations.
       ∧ ops ∈ [Ballot → CASOperation]
```

```
Send(m) ≜ msgs′ = msgs ∪ {m}
```

The leader of ballot $b \in Ballot$ sends a $Phase1a$ message.

```
Phase1a(b) ≜
    ∧  Send([type ↦ "1a", bal ↦ b])
    ∧  UNCHANGED ⟨maxBal, maxVBal, maxVVal, ops⟩
```

The acceptor $a \in Acceptor$ receives a $Phase1a$ message and sends back a $Phase1b$ message.

For refinement: This action implements the $IncreaseMaxBal(a, b)$ action of the Voting algorithm for $b = m.bal$.

```
Phase1b(a) ≜
  ∧ ∃ m ∈ msgs :
      ∧ m.type = "1a"
      ∧ m.bal > maxBal[a]
      ∧ maxBal′ = [maxBal EXCEPT ![a] = m.bal]
      ∧ Send([type ↦ "1b", acc ↦ a, bal ↦ m.bal,
              mbal ↦ maxVBal[a], mval ↦ maxVVal[a]])
  ∧ UNCHANGED ⟨maxVBal, maxVVal, ops⟩
```

In the $Phase2a(b, v)$ action, the ballot b leader sends a type "2a" message asking the acceptors to vote for some value computed based on v in ballot number b.

For refinement: the enabling conditions of the action–its first two conjuncts–ensure that the second through fourth conjuncts of the four enabling conditions of action $VoteFor(a, b, v)$ in module Voting will be true when acceptor a receives that message.

```
Phase2a(b, v) ≜
  ∧ ¬∃ m ∈ msgs  : m.type = "2a" ∧ m.bal = b
  ∧ ∃ Q ∈ Quorum :
      LET Q1b ≜ {m ∈ msgs  : ∧ m.type = "1b"
                             ∧ m.acc ∈ Q
                             ∧ m.bal = b}
          Q1bv ≜ {m ∈ Q1b : m.mbal ≥ 0}
      IN  ∧ ∀ a ∈ Q : ∃ m ∈ Q1b : m.acc = a
```

```
                    ∧ ∨ ∧ Q1bv = {}   ⟨+⟩ CAS(None, v) as an initialization operation
                        ∧ ops[b].cmpVal = None   ⟨+⟩
                      ∨ ∃ m ∈ Q1bv :   ⟨+⟩ CAS(v, ops[b].swapVal) as an atomic compare-and-swap operation
                          ∧ m.mval = v
                          ∧ ∀ mm ∈ Q1bv : m.mbal ≥ mm.mbal
                          ∧ ops[b].cmpVal = v   ⟨+⟩
    ∧ Send([type ↦ "2a", bal ↦ b, val ↦ ops[b].swapVal])   ⟨+⟩ val ↦ ops[b].swapVal
    ∧ UNCHANGED ⟨maxBal, maxVBal, maxVVal, ops⟩
```

The $Phase2b(a)$ action describes what $a \in Acceptor$ does when it receives a phase $2a$ message $m \in msgs$, which is sent by the leader of ballot $m.bal$ asking acceptors to vote for $m.val$ in that ballot.

For refinement: The enabling condition of the $Phase2b(a)$ action together with the receipt of the phase $2a$ message m implies that the $VoteFor(a, m.bal, m.val)$ action of module Voting is enabled and can be executed.

```
Phase2b(a) ≜
  ∧ ∃ m ∈ msgs :
      ∧ m.type = "2a"
      ∧ m.bal ≥ maxBal[a]
      ∧ maxBal' = [maxBal EXCEPT ![a] = m.bal]
      ∧ maxVBal' = [maxVBal EXCEPT ![a] = m.bal]
      ∧ maxVVal' = [maxVVal EXCEPT ![a] = m.val]
      ∧ Send([type ↦ "2b", acc ↦ a, bal ↦ m.bal, val ↦ m.val])
  ∧ UNCHANGED ⟨ops⟩
```

⟨+⟩ The leader of ballot $b \in Ballot$ responds to the user.

```
Respond(b) ≜
  ∧ ¬∃ m ∈ msgs   : m.type = "response" ∧ m.bal = b
  ∧ ∃ Q ∈ Quorum :
      LET Q2b ≜ {m ∈ msgs : ∧ m.type = "2b"
                            ∧ m.acc ∈ Q
                            ∧ m.bal = b}
      IN  ∀ a ∈ Q : ∃ m ∈ Q2b : m.acc = a
  ∧ Send([type ↦ "response", bal ↦ b])
  ∧ UNCHANGED ⟨maxBal, maxVBal, maxVVal, ops⟩
```

```
Next ≜
  ∨ ∃ b ∈ Ballot :
      ∨ Phase1a(b)
      ∨ ∃ v ∈ Value : Phase2a(b, v)
      ∨ Respond(b)   ⟨+⟩
  ∨ ∃ a ∈ Acceptor :
      ∨ Phase1b(a)
      ∨ Phase2b(a)

Spec ≜ Init ∧ □[Next]_vars
```

\ * Modification History
\ * Last modified *Wed Jul* 27 00:15:28 *CST* 2022 by *hengxin*
\ * Created *Tue Jul* 20 23:30:00 *CST* 2022 by *hengxin*