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*.

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 ≜ CHOOSE v : v ∉ Value

ASSUME  ∧ ∀ Q ∈ Quorum : Q ⊆ Acceptor
        ∧ ∀ Q1, Q2 ∈ Quorum : Q1 ∩ Q2 ≠ {}
────────────────────────────────────────────────────────────
Ballot ≜  Nat
```

Added for *CASPaxos*.

The set of all possible *CAS* operations. The *CAS* operations with *cmpVal* = *None* are initialization operations. We do not allow the new value (*swapVal*) to be *None*.

```
CASOperation ≜ [cmpVal : Value ∪ {None}, swapVal : Value]

Message ≜   the set of all possible messages that can be sent in the algorithm
         [type  : {"1a"}, bal  : Ballot]
    ∪    [type  : {"1b"}, acc : Acceptor, bal : Ballot,
          mbal : Ballot ∪ { − 1}, mval : Value ∪ {None}]
    ∪    [type  : {"2a"}, bal  : Ballot, val : Value]
    ∪    [type  : {"2b"}, acc : Acceptor, bal : Ballot, val : Value]
────────────────────────────────────────────────────────────
```

maxBal − Is the same as the variable of that name in the Voting algorithm.
maxVBal
maxVVal − As in the Voting algorithm, a vote is a ⟨*ballot*, *value*⟩ pair. The pair ⟨*maxVBal*[*a*], *maxVVal*[*a*]⟩ is the vote with the largest ballot number cast by acceptor a . It equals ⟨ − 1, *None*⟩ if a has not cast any vote.

```
VARIABLES
    maxBal,
    maxVBal,
    maxVVal,
    msgs,        the set of all messages that have been sent
    ops          ops[b ∈ Ballot]: the CAS operation to be proposed at ballot b
                 added for CASPaxos
```

```
vars ≜ ⟨maxBal, maxVBal, maxVVal, msgs, ops⟩
├──────────────────────────────────────────────────────────────┤
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}
├──────────────────────────────────────────────────────────────┤
TODO: define the CAS(cmpVal, swapVal) interface


The leader of ballot b ∈ Ballot sends a Phase1a message.
Phase1a(b) ≜
    ∧   Send([type ↦ "1a", bal ↦ b])
    ∧   UNCHANGED ⟨maxBal, maxVBal, maxVVal, ops⟩
The acceptor a ∈ 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   added for CASPaxos
                     ∨ ∃ 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   added for CASPaxos
    ∧ Send([type ↦ "2a", bal ↦ b, val ↦ ops[b].swapVal])   modified for CASPaxos: 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.

TODO: to finish it

```
Respond(b) ≜ FALSE

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

Spec ≜ Init ∧ □[Next]_vars
```