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

$None \triangleq \text{CHOOSE } v : v \notin Value$

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

$Ballot \triangleq Nat$

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*.

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

$Message \triangleq$ the set of all possible messages that can be sent in the algorithm

$[type : \{\text{"1a"}\},\, bal : Ballot]$
$\cup\ [type : \{\text{"1b"}\},\, acc : Acceptor,\, bal : Ballot,$
$\quad mbal : Ballot \cup \{-1\},\, mval : Value \cup \{None\}]$
$\cup\ [type : \{\text{"2a"}\},\, bal : Ballot,\, val : Value]$
$\cup\ [type : \{\text{"2b"}\},\, acc : Acceptor,\, bal : Ballot,\, val : Value]$

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

VARIABLES
- *maxBal*,
- *maxVBal*,
- *maxVVal*,
- *msgs*, the set of all messages that have been sent
- *ops* *ops*[$b \in Ballot$]: the *CAS* operation to be proposed at ballot b

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

$TypeOK \triangleq$
- $maxBal \in [Acceptor \to Ballot \cup \{-1\}]$
- $maxVBal \in [Acceptor \to Ballot \cup \{-1\}]$
- $maxVVal \in [Acceptor \to Value \cup \{None\}]$
- $msgs \subseteq Message$
- $ops \in [Ballot \to CASOperation]$

$Init \triangleq$
- $maxBal = [a \in Acceptor \mapsto -1]$
- $maxVBal = [a \in Acceptor \mapsto -1]$
- $maxVVal = [a \in Acceptor \mapsto None]$
- $msgs = \{\}$
- *ops* remains unchanged; we utilize *TLC* to explore all possible *CAS* operations.
- $ops \in [Ballot \to CASOperation]$

$Send(m) \triangleq msgs' = msgs \cup \{m\}$

TODO: define the $CAS(cmpVal,\ swapVal)$ interface

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

$Phase1a(b) \triangleq$
- $Send([type \mapsto$ "1a", $bal \mapsto b])$
- UNCHANGED $\langle maxBal,\ maxVBal,\ maxVVal,\ ops \rangle$

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 $b = m.bal$.

$Phase1b(a) \triangleq$
- $\exists\, m \in msgs :$
  - $m.type =$ "1a"
  - $m.bal > maxBal[a]$
  - $maxBal' = [maxBal$ EXCEPT $![a] = m.bal]$
  - $Send([type \mapsto$ "1b", $acc \mapsto a,\ bal \mapsto m.bal,$
    $mbal \mapsto maxVBal[a],\ mval \mapsto maxVVal[a]])$
- UNCHANGED $\langle maxVBal,\ maxVVal,\ ops \rangle$

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.

$Phase2a(b,\ v) \triangleq$
- $\neg\exists\, m \in msgs \ : m.type =$ "2a" $\wedge\ m.bal = b$
- $\exists\, Q \in Quorum :$
  - LET $Q1b \triangleq \{m \in msgs \ : \wedge\ m.type =$ "1b"
    $\wedge\ m.acc \in Q$

```
                                        ∧ 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
              ∨ ∃ 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 *Phase*2*b*(*a*) action describes what $a \in Acceptor$ does when it receives a phase 2*a* 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 *Phase*2*b*(*a*) action together with the receipt of the phase 2*a* 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
```
