(* *MIT* License Copyright (c) 2020 *CQFN*
* https://*github.com*/cqfn/degitx/blob/master/LICENSE
*
* The module shows a conflict in the 3-phase commit protocol with the backup transaction manager.
* The problem lies in the different behavior of the Prime and Second (Backup) transaction managers.
* The $3PC$ protocol guarantees consistency if and only if ONE and ONLY TM makes decisions.
* at the moment at any stage of processing. And all nodes are available for this TM.
*)

MODULE $BackupCol$

EXTENDS $Sequences$, $FiniteSets$, TLC

CONSTANTS RM Set of all *Resourse* Managers. All available for all *TMs*

VARIABLES

$rmState$ Current state of every particular RM (set)

$isAnyAborted(RMS) \triangleq$
$\land \exists\, r \in RMS : rmState[r] =$ "aborted"

$isAnyWorking(RMS) \triangleq$
$\land \exists\, r \in RMS : rmState[r] =$ "working"

$allWorking(RMS) \triangleq$
$\land \forall\, r \in RMS : rmState[r] =$ "working"

$isAnyPrepared(RMS) \triangleq$
$\land \exists\, r \in RMS : rmState[r] =$ "prepared"

At least one node already gone to Prepared state. Or even more - to Committed state. Unreacheble with given assumptions.

$tmSendCommit(r) \triangleq$
$\land \exists\, rm \in RM : rmState[rm] \in \{$"prepared", "committed"$\}$
$\land rmState[r] =$ "prepared"
$\land rmState' = [rmState$ EXCEPT $![r] =$ "committed"$]$

As described in $3PC$: all nodes answered to Prime TM Working and go to Prepared state one by one.

$tmSendPrepare(r) \triangleq$
$\land rmState[r] =$ "working"
$\land rmState' = [rmState$ EXCEPT $![r] =$ "prepared"$]$

All nodes are in Working stage and Prime TM failed for a while. *BackUp TM* rise and sends "abort" message to nodes one-by-one

$tmBackSendAbort(r) \triangleq$
$\land rmState[r] =$ "working"
$\land rmState' = [rmState$ EXCEPT $![r] =$ "aborted"$]$

Predicate. Initial state here - all RM are in working stage (Trx is already began) Primarly TM received status "working" and going to continue transaction Some issue i the system, timeout for TM message. Nodes
BackUp TM

$$VCInit \triangleq \\ \land rmState = [rm \in RM \mapsto \text{"working"}]$$

Next state is: For *Primarly TM* - move all working nodes to "prepared" state For *BackUp TM* - rollback *Trx*. Move nodes to "aborted" state

$$VCNext \triangleq \\ \lor \exists\, r \in RM : \\ tmSendPrepare(r) \lor tmBackSendAbort(r) \lor tmSendCommit(r)$$

The invariants:

$$TypeOK \triangleq$$

The type-correctness invariant

$$\land \forall\, r \in RM : rmState[r] \in \{\text{"working"}, \text{"prepared"}, \text{"committed"}, \text{"aborted"}\}$$

$$Consistency \triangleq$$

A state predicate asserting that two *RMs* have not arrived to conflicting decisions. Prepared is the point on one-way direction to Committed stage. There is no Commit in *Git* to pull, but *Trx* is failed silently. However, it's impossible to roll back the *Trx* from Prepared state

$$\forall\, rm1, rm2 \in RM : \\ \neg(\land rmState[rm1] = \text{"aborted"} \\ \land rmState[rm2] \in \{\text{"prepared"}, \text{"committed"}\})$$