MODULE *StateCounter*

TLA+ module for State-based Counter. See its implementation in paper *Burckhardt@POPL'*2014.
We check that the State-based Counter satisfies the strong eventual convergence property (*SEC*)

EXTENDS *Naturals*, *Sequences*, *Bags*, *TLC*

CONSTANTS

Protocol variables.

Replica, the set of replicas

Auxiliary variables for model checking.

Max *Max*[*r*]: the maximum number of the *Inc*() event replica $r \in Replica$ can issue; for finite-state model checking

VARIABLES

Protocol variables.

vc, *vc*[*r*][*r*]: the current value of the counter vector at replica $r \in Replica$

incoming, *incoming*[*r*]: incoming messages at replica $r \in Replica$

Auxiliary variables for model checking.

inc, *inc*[*r*]: the number of *Inc*() events issued by the replica $r \in Replica$

sendAllowed *sendAllowed*[*r*]: is the replica $r \in Replica$ allowed to send a message

The type correctness predicate.

$$\begin{aligned} TypeOK \triangleq\ & \land vc \in [Replica \to [Replica \to Nat]] \\ & \land inc \in [Replica \to Nat] \\ & \land sendAllowed \in [Replica \to \{0, 1\}] \\ & \land incoming \in [Replica \to SubBag(SetToBag(Nat))] \setminus^* \text{message ordering is not important; using bag (}i.e.\text{, mul} \end{aligned}$$

The initial state predicate.

$$\begin{aligned} Init \triangleq\ & \land vc = [r \in Replica \mapsto [s \in Replica \mapsto 0]] \\ & \land incoming = [r \in Replica \mapsto EmptyBag] \\ & \land inc = [r \in Replica \mapsto 0] \\ & \land sendAllowed = [r \in Replica \mapsto 0] \end{aligned}$$

Replica $r \in Replica$ issues an *Read*() event.

Replica $r \in Replica$ issues an *Inc*() event.

$$\begin{aligned} Inc(r) \triangleq\ & \land inc[r] < Max[r] \\ & \quad \text{each replica } r \in Replica \text{ can issue at most } Max[r] Inc() \text{ events.} \\ & \land vc' = [vc \text{ EXCEPT } ![r][r] = @ + 1] \quad \text{current counter } +1 \\ & \land inc' = [inc \text{ EXCEPT } ![r] = @ + 1] \\ & \land sendAllowed' = [sendAllowed \text{ EXCEPT } ![r] = 1] \\ & \land \text{UNCHANGED } \langle incoming \rangle \end{aligned}$$

Broadcast a message m to all replicas except the sender s.

$$\begin{aligned}
&Broadcast(s,\, m) \triangleq\ [r \in Replica \mapsto \\
&\qquad \text{IF } s = r \\
&\qquad \text{THEN } incoming[s] \\
&\qquad \text{ELSE } incoming[r] \oplus SetToBag(\{m\})]
\end{aligned}$$

Replica r issues a *Send*() event, sending an update message.

$$\begin{aligned}
&Send(r) \triangleq\ \land\ sendAllowed[r] = 1 \\
&\qquad \land\ incoming' = Broadcast(r,\, vc[r]) \quad \text{broadcast } vc[r] \text{ to other replicas} \\
&\qquad \land\ sendAllowed' = [sendAllowed \text{ EXCEPT } ![r] = 0] \\
&\qquad \land\ \text{UNCHANGED } \langle vc,\, inc\rangle
\end{aligned}$$

Replica r issues a *Receive*() event, receiving an update message.

$$SetMax(r,\, s) \triangleq \text{IF } r > s \text{ THEN } r \text{ ELSE } s$$

$$\begin{aligned}
&Receive(r) \triangleq\ \land\ incoming[r] \neq EmptyBag \quad \text{there are accumulated increments from other replicas} \\
&\qquad \land \exists\, m \in BagToSet(incoming[r]) : \quad \text{message reordering can be tolerant} \\
&\qquad\quad (\land \forall\, s \in Replica : vc' = [vc \text{ EXCEPT } ![r][s] = SetMax(@,\, m[s])] \\
&\qquad\qquad \land\ incoming' = [incoming \text{ EXCEPT } ![r] = @ \ominus SetToBag(\{m\})]) \quad \text{each message is delivered ex} \\
&\qquad \land\ sendAllowed' = [sendAllowed \text{ EXCEPT } ![r] = 1] \\
&\qquad \land\ \text{UNCHANGED } \langle inc\rangle
\end{aligned}$$

The Next-state relation.

$$Next \triangleq\ \land \exists\, r \in Replica : Inc(r) \lor Send(r) \lor Receive(r)$$

The specification.

$$\begin{aligned}
&vars \triangleq \langle vc,\, incoming,\, inc,\, sendAllowed\rangle \\
&Spec \triangleq Init \land \Box[Next]_{vars} \land \mathrm{WF}_{vars}(Next)
\end{aligned}$$

The correctness of counter: Eventual *Convergence* (*EC*), Quiescent Consistency (*QC*), and Strong Eventual *Convergence* (*SEC*).

Eventual Consistency (*EC*) If clients stop issuing *Incs*, then the counters at all replicas will be eventually the same.

$$\begin{aligned}
&Convergence \triangleq\ \land \forall\, r,\, s \in Replica : vc[r] = vc[s] \\
&\qquad\qquad \land \exists\, r,\, s \in Replica : vc[r][s] \neq 0 \quad \text{excluding the initial state} \\
&EC \triangleq \Diamond Convergence
\end{aligned}$$

Quiescent Consistency (*QC*) If the system is at quiescent, then the counters at all replicas must be the same.

$$\begin{aligned}
&AccBroadcast \triangleq \forall\, r \in Replica : sendAllowed[r] = 0 \quad \text{all } r \in Replica \text{ are not allowed to send} \\
&MessageDelivery \triangleq \forall\, r \in Replica : incoming[r] = EmptyBag \quad \text{all messages have been delivered} \\
&QConvergence \triangleq \forall\, r,\, s \in Replica : vc[r] = vc[s] \quad \text{no } counter[r] \neq 0
\end{aligned}$$

$QC \triangleq \Box(AccBroadcast \wedge MessageDelivery \Rightarrow QConvergence)$

Strong Eventual Consistency (SEC)

* Modification History
* Last modified Sat *Aug* 11 11:49:53 *CST* 2018 by *zfwang*
* Created *Fri Aug* 03 09:57:12 *CST* 2018 by *zfwang*