```
MODULE MCChainRep
Extend the ChainRep module
EXTENDS  chainrep, TLC
CONSTANT  MaxReq   maximum number of requests in any channel
VARIABLES
        last_read_val,
        last_committed_write,
        prevread

mcchain_newvars ≜ ⟨prevread, last_read_val, last_committed_write⟩
mcchainvars ≜ chainvars ∘ mcchain_newvars
------------------------------------------------------------------------
MCChain_TypeInvariant ≜
    ∧ TypeInvariant

    ∧ last_read_val ∈ [Object → [Adr → Val ∪ {NoVal}]]
    ∧ last_committed_write ∈ [Object → [Adr → Val ∪ {NoVal}]]
    ∧ prevread ∈ [Object → [Adr → Val ∪ {NoVal}]]

MCChain_Init ≜
    ∧ Init

    ∧ last_read_val = [c ∈ Object ↦ [a ∈ Adr ↦ NoVal]]
    ∧ last_committed_write = [c ∈ Object ↦ [a ∈ Adr ↦ NoVal]]
    ∧ prevread = [c ∈ Object ↦ [a ∈ Adr ↦ NoVal]]
------------------------------------------------------------------------
MCChain_Reply(r, type, reply) ≜
  ∨ ∧ type = "wr"
    ∧ last_committed_write' = [last_committed_write EXCEPT ![reply.object][reply.adr] = reply.val]
    ∧ prevread' = [prevread EXCEPT ![reply.object][reply.adr] = NoVal]  invalidate the previous read
    ∧ UNCHANGED last_read_val

  ∨ ∧ type = "rd"
    ∧ prevread' = [prevread EXCEPT ![reply.object][reply.adr] = reply.val]
    ∧ last_read_val' = [last_read_val EXCEPT ![reply.object][reply.adr] = reply.val]
    ∧ UNCHANGED ⟨last_committed_write⟩

MCChain_NoReply ≜
  UNCHANGED mcchain_newvars
------------------------------------------------------------------------
MCChain_ReplicaActions ≜
  ∨ Reconfiguration
  ∨ Recovery ∧ UNCHANGED mcchain_newvars
  ∨ ReplicaDeath ∧ UNCHANGED mcchain_newvars
```

```
    $\lor$ $ProcessMsg$

$MCChain\_Next \triangleq$
    $\lor$ $MasterActions \land$ UNCHANGED $mcchain\_newvars$
    $\lor$ $MCChain\_ReplicaActions$
    $\lor$ $ClientActions$

$MCChain\_Spec \triangleq MCChain\_Init \land \Box[MCChain\_Next]_{mcchainvars}$
```

```
$AddAllChannel(map, seq, a) \triangleq$
    LET $G[s \in Seq(Messages), ret \in [Val \rightarrow Nat]] \triangleq$
                IF $s = \langle\rangle$ THEN   $\lor \forall i \in 1 \, .. \, Len(s): s[i].adr \neq a$
                     $ret$ no more writes to this address
                ELSE
                     LET $m \triangleq Head(s)$IN
                          IF $m.adr \neq a$ THEN $G[Tail(s), ret]$
                           ELSE $G[Tail(s), [ret$ EXCEPT $![m.w] = @ + 1]]$
    IN    $G[seq, map]$

$map\_pending\_wrreq1(o, a) \triangleq$
    LET $Test(r) \triangleq$
                    $\land$ $master.health[r] \neq$ "dead"  $r$ is either alive or recovering
                    $\land$ $InSeq(master.chains[o], r)$
                    $\land$ $cache[r][o].in\_chain$ \* $r$ believes itself in chain
           $F[rep \in$ SUBSET $(Rep), ret \in [Val \rightarrow Nat]] \triangleq$
              IF $rep = \{\} \lor \forall r \in rep : \neg Test(r)$
                 THEN $ret$
                 ELSE LET $r \triangleq$ CHOOSE $r \in rep : Test(r)$
                           IN    $F[rep \setminus \{r\}, AddAllChannel(ret, channel[r][o].in, a)]$
    IN
         (  $Print($"$Pending\_wrreq$", $F[Rep, [v \in Val \mapsto 0]])$

$map\_pending\_wrreq(o, a) \triangleq$
   LET $Test(r) \triangleq$
           $\land$ $master.health[r] \neq$ "dead"  $r$ is either alive or recovering
           $\land$ $InSeq(master.chains[o], r)$
           $\land$ $r \neq Hd(o)$

         $IsClientReq(m) \triangleq$
           $m.type =$ "cliWrReq"

         $F[rep \in$ SUBSET $(Rep), ret \in [Val \rightarrow Nat]] \triangleq$
            IF $rep = \{\} \lor \forall r \in rep : \neg Test(r)$
             THEN $ret$
             ELSE LET $r \triangleq$ CHOOSE $r \in rep : Test(r)$
                     IN    $F[rep \setminus \{r\}, AddAllChannel(ret, SelectSeq(channel[r][o].in, IsClientReq), a)]$
```

IN

$AddAllChannel(AddAllChannel(F[Rep, [v \in Val \mapsto 0]], channel[Hd(o)][o].in, a), channel[Hd(o)][o].out, a)$

$map\_store(o, a) \triangleq$

```
IF $stat[Tl(o)].phase \neq$ "recover" \* the replica is not recovering
THEN $data[Tl(o)][o][a]$
ELSE \* tail is a new recovering replica, so we have to compute the real tail
  LET $oldtail \triangleq$

          $master.chains[o][Len(master.chains[o]) - 2]$
  IN  $(\ data[oldtail][o][a])$
  $last\_committed\_write[o][a]$
```

$map\_last\_read\_val(o, a) \triangleq last\_read\_val[o][a]$

$map\_pending\_rdreq(o, a) \triangleq 0$ use the quickrd version

$mapped\_ob \triangleq$ CHOOSE $o \in Object :$ TRUE
$mapped\_adr \triangleq$ CHOOSE $a \in Adr :$ TRUE

$ChainSS \triangleq$ INSTANCE $chain\_ss\_quickrd$ WITH

```
    $store \leftarrow map\_store(mapped\_ob, mapped\_adr),$
    $pending\_wrreq \leftarrow map\_pending\_wrreq(mapped\_ob, mapped\_adr),$
    $pending\_rdreq \leftarrow map\_pending\_rdreq(mapped\_ob, mapped\_adr),$
    $last\_read\_val \leftarrow map\_last\_read\_val(mapped\_ob, mapped\_adr)$
```

$ChainImplementsChainSS \triangleq ChainSS!SSQ\_Spec$

Constraints

$MaxChannelConstr \triangleq$ imposes a limit on the size of the channels at any time

```
  $\forall r \in Rep, o \in Object :$
      $\land Len(channel[r][o].in) < MaxReq$
      $\land Len(channel[r][o].out) < MaxReq$
```

A set of symmetry functions that make checking much faster.

$Perms \triangleq Permutations(Val) \cup Permutations(Rep) \cup Permutations(Adr) \cup Permutations(Object)$

$ReadLastCommitted \triangleq$

```
  $\forall o \in Object, a \in Adr :$
    $\forall r \in Rep :$
      $(\land stat[r].phase =$ "alive"
       $\land cache[r][o].in\_chain$
       $\land cache[r][o].right = NoRep)$ it believes it's tail
```

$\Rightarrow data[r][o][a] = last\_committed\_write[o][a]$

$MCAllInvariants \triangleq \land AllInvariants$

THEOREM $MCChain\_Spec \Rightarrow \Box MCAllInvariants$