```
──────────────────── MODULE CJupiter ────────────────────
Model of our own CJupiter protocol.
EXTENDS StateSpace, JupiterSerial
─────────────────────────────────────────────────────────
VARIABLES
    css    css[r]: the n-ary ordered state space at replica r ∈ Replica

vars ≜ ⟨intVars, ctxVars, serialVars, css⟩
─────────────────────────────────────────────────────────
TypeOK ≜
    ∧  TypeOKInt
    ∧  TypeOKCtx
    ∧  TypeOKSerial
    ∧  Comm(Cop)!TypeOK
    ∧  ∀ r ∈ Replica : IsSS(css[r])
─────────────────────────────────────────────────────────
Init ≜
    ∧ InitInt
    ∧ InitCtx
    ∧ InitSerial
    ∧ Comm(Cop)!Init
    ∧ css = [r ∈ Replica ↦ EmptySS]
─────────────────────────────────────────────────────────
xForm: Iteratively transform cop with a path through the css at replica r ∈ Replica, following
the first edges.
xForm(cop, r) ≜
    LET rcss ≜ css[r]
        u ≜ Locate(cop, rcss)
        v ≜ u ∪ {cop.oid}
        RECURSIVE xFormHelper(_, _, _, _)
         'h' stands for "helper"; xcss: eXtra css created during transformation
        xFormHelper(uh, vh, coph, xcss) ≜
            IF uh = ds[r]
             THEN ⟨xcss, coph⟩
             ELSE LET fedge ≜ CHOOSE e ∈ rcss.edge :
                                  ∧ e.from = uh
                                  ∧ ∀ uhe  ∈ rcss.edge :
                                     (uhe.from = uh ∧ uhe ≠ e) ⇒ tb(e.cop.oid, uhe.cop.oid, serial[r])
                      uprime ≜ fedge.to
                      fcop ≜ fedge.cop
                      coph2fcop ≜ COT(coph, fcop)
                      fcop2coph ≜ COT(fcop, coph)
                       vprime ≜ vh ∪ {fcop.oid}
                  IN   xFormHelper(uprime, vprime, coph2fcop,
                           [xcss EXCEPT !.node = @ ∪ {vprime},
                                        !.edge = @ ∪ {[from ↦ vh, to ↦ vprime, cop ↦ fcop2coph],
                                                      [from ↦ uprime, to ↦ vprime, cop ↦ coph2fcop]}])
   IN    xFormHelper(u, v, cop, [node ↦ {v}, edge ↦ {[from ↦ u, to ↦ v, cop ↦ cop]}])

Perform cop at replica r ∈ Replica.
Perform(cop, r) ≜
    LET xform ≜ xForm(cop, r)   xform: ⟨xcss, xcop⟩
        xcss  ≜ xform[1]
        xcop  ≜ xform[2]
    IN  ∧ css′ = [css EXCEPT ![r] = @ ⊕ xcss]
        ∧ state′ = [state EXCEPT ![r] = Apply(xcop.op, @)]


Client c ∈ Client issues an operation op.
DoOp(c, op) ≜   op: the raw operation generated by the client c ∈ Client
      ∧ LET cop ≜ [op ↦ op, oid ↦ [c ↦ c, seq ↦ cseq′[c]], ctx ↦ ds[c]]
          IN   ∧ Perform(cop, c)
               ∧ UpdateDS(c, cop)
               ∧ Comm(Cop)!CSend(cop)

DoIns(c) ≜
    ∃ ins ∈ {op ∈ Ins : op.pos ∈ 1 . . (Len(state[c]) + 1) ∧ op.ch ∈ chins ∧ op.pr = Priority[c]} :
       ∧ DoOp(c, ins)
       ∧ chins′ = chins \ {ins.ch}   We assume that all inserted elements are unique.

DoDel(c) ≜
    ∃ del ∈ {op ∈ Del : op.pos ∈ 1 . . Len(state[c])} :
       ∧ DoOp(c, del)
       ∧ UNCHANGED chins

Do(c) ≜
      ∧ DoCtx(c)
      ∧ DoSerial(c)
      ∧ ∨ DoIns(c)
        ∨ DoDel(c)

Client c ∈ Client receives a message from the Server.
Rev(c) ≜
      ∧ Comm(Cop)!CRev(c)
      ∧ Perform(Head(cincoming[c]), c)
      ∧ RevSerial(c)
      ∧ RevCtx(c)
      ∧ UNCHANGED chins


The Server receives a message.
SRev ≜
    ∧ Comm(Cop)!SRev
    ∧ LET cop ≜ Head(sincoming)
```

```
        IN   ∧ Perform(cop, Server)
             ∧ Comm(Cop)!SSendSame(cop.oid.c, cop)   broadcast the original operation
     ∧ SRevSerial
     ∧ SRevCtx
     ∧ UNCHANGED chins
├──────────────────────────────────────────────────────────────────────┤
Next ≜
     ∨ ∃ c ∈ Client : Do(c) ∨ Rev(c)
     ∨ SRev
Fairness: There is no requirement that the clients ever generate operations.
Fairness ≜
     WF_vars(SRev ∨ ∃ c ∈ Client : Rev(c))

Spec ≜ Init ∧ □[Next]_vars   ∧ Fairness (We care more about safety.)
├──────────────────────────────────────────────────────────────────────┤
The compactness of CJupiter: the CSSes at all replicas are the same.
Compactness ≜
     Comm(Cop)!EmptyChannel ⇒ Cardinality(Range(css)) = 1

THEOREM Spec ⇒ Compactness
└──────────────────────────────────────────────────────────────────────┘
\* Modification History
\* Last modified Mon Dec 24 10:17:00 CST 2018 by hengxin
\* Created Sat Sep 01 11:08:00 CST 2018 by hengxin
```