```
1  ---------- MODULE CJupiter ----------
   Model of our own CJupiter protocol.
5  EXTENDS StateSpace, JupiterSerial
6  ----------
7  VARIABLES
8      css      css[r]: the n-ary ordered state space at replica r ∈ Replica
10 vars ≜ ⟨intVars, ctxVars, serialVars, css⟩
11 ----------
12 TypeOK ≜
13     ∧  TypeOKInt
14     ∧  TypeOKCtx
15     ∧  TypeOKSerial
16     ∧  Comm(Cop)!TypeOK
17     ∧  ∀ r ∈ Replica : IsSS(css[r])
18 ----------
19 Init ≜
20     ∧ InitInt
21     ∧ InitCtx
22     ∧ InitSerial
23     ∧ Comm(Cop)!Init
24     ∧ css = [r ∈ Replica ↦ EmptySS]
25 ----------
   xForm: Iteratively transform cop with a path through the css at replica r ∈ Replica, following
   the first edges.
30 xForm(cop, r) ≜
31     LET rcss ≜ css[r]
32         u ≜ Locate(cop, rcss)
33         v ≜ u ∪ {cop.oid}
34         RECURSIVE xFormHelper(_, _, _, _)
35          'h' stands for "helper"; xcss: eXtra css created during transformation
36         xFormHelper(uh, vh, coph, xcss) ≜
37             IF uh = ds[r]
38              THEN [xcss ↦ xcss, xcop ↦ coph]
39              ELSE LET fedge ≜ CHOOSE e ∈ rcss.edge :
40                                 ∧ e.from = uh
41                                 ∧ ∀ uhe ∈ rcss.edge :
42                                     (uhe.from = uh ∧ uhe ≠ e) ⇒ tb(e.cop.oid, uhe.cop.oid, serial[r])
43                       uprime ≜ fedge.to
44                       fcop ≜ fedge.cop
45                       coph2fcop ≜ COT(coph, fcop)
46                       fcop2coph ≜ COT(fcop, coph)
47                       vprime ≜ vh ∪ {fcop.oid}
48                   IN  xFormHelper(uprime, vprime, coph2fcop,
49                         [xcss EXCEPT !.node = @ ∪ {vprime},
```

```
50                                  !.edge = @ ∪ {[from ↦ vh, to ↦ vprime, cop ↦ fcop2coph],
51                                                [from ↦ uprime, to ↦ vprime, cop ↦ coph2fcop]})
52      IN    xFormHelper(u, v, cop, [node ↦ {v}, edge ↦ {[from ↦ u, to ↦ v, cop ↦ cop]}])
```

Perform cop at replica $r \in Replica$.

```
56  Perform(cop, r) ≜
57      LET xform ≜ xForm(cop, r)   xform: [xcss, xcop]
58      IN  ∧ css' = [css EXCEPT ![r] = @ ⊕ xform.xcss]
59          ∧ state' = [state EXCEPT ![r] = Apply(xform.xcop.op, @)]
60  ----------------------------------------------------------------------
```

Client $c \in Client$ issues an operation op.

```
64  DoOp(c, op) ≜   op: the raw operation generated by the client c ∈ Client
65          ∧ LET cop ≜ [op ↦ op, oid ↦ [c ↦ c, seq ↦ cseq'[c]], ctx ↦ ds[c]]
66            IN  ∧ Perform(cop, c)
67                ∧ UpdateDS(c, cop)
68                ∧ Comm(Cop)!CSend(cop)

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

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

80  Do(c) ≜
81          ∧ DoCtx(c)
82          ∧ DoSerial(c)
83          ∧ ∨ DoIns(c)
84            ∨ DoDel(c)
```

Client $c \in Client$ receives a message from the $Server$.

```
88  Rev(c) ≜
89          ∧ Comm(Cop)!CRev(c)
90          ∧ Perform(Head(cincoming[c]), c)
91          ∧ RevSerial(c)
92          ∧ RevCtx(c)
93          ∧ UNCHANGED chins
94  ----------------------------------------------------------------------
```

The $Server$ receives a message.

```
98  SRev ≜
99      ∧ Comm(Cop)!SRev
100     ∧ LET cop ≜ Head(sincoming)
101       IN  ∧ Perform(cop, Server)
102           ∧ Comm(Cop)!SSendSame(cop.oid.c, cop)   broadcast the original operation
```

```
103         ∧ SRevSerial
104         ∧ SRevCtx
105         ∧ UNCHANGED chins
106 ----------------------------------------------------------------------
107 Next ≜
108         ∨ ∃ c ∈ Client : Do(c) ∨ Rev(c)
109         ∨ SRev
    Fairness: There is no requirement that the clients ever generate operations.
113 Fairness ≜
114         WF_vars(SRev ∨ ∃ c ∈ Client : Rev(c))

116 Spec ≜ Init ∧ □[Next]_vars   ∧ Fairness (We care more about safety.)
117 ----------------------------------------------------------------------
    The compactness of CJupiter: the CSSes at all replicas are the same.
121 Compactness ≜
122         Comm(Cop)!EmptyChannel ⇒ Cardinality(Range(css)) = 1

124 THEOREM Spec ⇒ Compactness
125 ======================================================================
    \* Modification History
    \* Last modified Mon Dec 24 10:33:33 CST 2018 by hengxin
    \* Created Sat Sep 01 11:08:00 CST 2018 by hengxin
```