```
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 $r \in Replica$, following the first edges.
```
xForm(cop, r) ≜
    LET rcss ≜ css[r]
        u ≜ Locate(cop, rcss)
        v ≜ u ∪ {cop.oid}
        RECURSIVE xFormHelper(_, _, _, _)
        xFormHelper(uh, vh, coph, xcss) ≜   xcss: eXtra css created during transformation
            IF uh = ds[r] THEN [xcss ↦ xcss, xcop ↦ coph]
             ELSE LET fedge ≜ CHOOSE e ∈ rcss.edge :
                                  ∧ e.from = uh
                                  ∧ ∀ uhe ∈ rcss.edge \ {e} :
                                      (uhe.from = uh) ⇒ 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 ⊕ [node ↦ {vprime},
                                edge ↦ {[from ↦ vh, to ↦ vprime, cop ↦ fcop2coph],
                                        [from ↦ uprime, to ↦ vprime, cop ↦ coph2fcop]}])
```

```
50        IN   xFormHelper(u, v, cop, [node ↦ {v}, edge ↦ {[from ↦ u, to ↦ v, cop ↦ cop]}])

52  Perform(cop, r) ≜   Perform cop at replica r ∈ Replica.
53      LET xform ≜ xForm(cop, r)   xform: [xcss, xcop]
54      IN   ∧ css′ = [css EXCEPT ![r] = @ ⊕ xform.xcss]
55           ∧ state′ = [state EXCEPT ![r] = Apply(xform.xcop.op, @)]
56  ──────────────────────────────────────────────────────────────────────
57  DoOp(c, op) ≜
58        ∧ LET cop ≜ [op ↦ op, oid ↦ [c ↦ c, seq ↦ cseq′[c]], ctx ↦ ds[c]]
59          IN   ∧ Perform(cop, c)
60               ∧ Comm(Cop)!CSend(cop)

62  Do(c) ≜
63        ∧ DoCtx(c)
64        ∧ DoSerial(c)
65        ∧ DoInt(DoOp, c)

67  Rev(c) ≜
68        ∧ Comm(Cop)!CRev(c)
69        ∧ Perform(Head(cincoming[c]), c)
70        ∧ RevSerial(c)
71        ∧ RevCtx(c)
72        ∧ RevInt(c)

74  SRev ≜
75      ∧ Comm(Cop)!SRev
76      ∧ LET cop ≜ Head(sincoming)
77        IN   ∧ Perform(cop, Server)
78             ∧ Comm(Cop)!SSendSame(cop.oid.c, cop)   broadcast the original operation
79      ∧ SRevSerial
80      ∧ SRevCtx
81      ∧ SRevInt
82  ──────────────────────────────────────────────────────────────────────
83  Next ≜
84      ∨ ∃ c ∈ Client : Do(c) ∨ Rev(c)
85      ∨ SRev

87  Fairness ≜   There is no requirement that the clients ever generate operations.
88      WF_vars(SRev ∨ ∃ c ∈ Client : Rev(c))

90  Spec ≜ Init ∧ □[Next]_vars   ∧ Fairness (We care more about safety.)
91  ──────────────────────────────────────────────────────────────────────
92  Compactness ≜   Compactness of CJupiter: the CSSes at all replicas are the same.
93      Comm(Cop)!EmptyChannel ⇒ Cardinality(Range(css)) = 1

95  THEOREM Spec ⇒ Compactness
96  ──────────────────────────────────────────────────────────────────────
```

\ * Modification History
\ * Last modified *Mon Dec* 31 20:36:31 *CST* 2018 by *hengxin*
\ * Created Sat *Sep* 01 11:08:00 *CST* 2018 by *hengxin*