```
MODULE XJupiter
Specification of the Jupiter protocol described in CSCW'2014 by Xu, Sun, and Li.
EXTENDS StateSpace
VARIABLES
    c2ss,    c2ss[c]: the 2D state space (2ss, for short) at client c ∈ Client
    s2ss     s2ss[c]: the 2D state space maintained by the Server for client c ∈ Client
vars ≜ ⟨intVars, ctxVars, c2ss, s2ss⟩
TypeOK ≜
    ∧  TypeOKInt
    ∧  TypeOKCtx
    ∧  ∀ c ∈ Client : IsSS(c2ss[c]) ∧ IsSS(s2ss[c])
Init ≜
    ∧ InitInt
    ∧ InitCtx
    ∧ c2ss = [c ∈ Client ↦ EmptySS]
    ∧ s2ss = [c ∈ Client ↦ EmptySS]
xForm(cop, ss, cur) ≜  Transform cop with an operation sequence in 2D state space ss.
    LET u ≜ Locate(cop, ss)
        v ≜ u ∪ {cop.oid}
        RECURSIVE xFormHelper(_, _, _, _)
        xFormHelper(uh, vh, coph, xss) ≜   xss: eXtra ss created during transformation
            IF uh = cur THEN [xss ↦ xss, xcop ↦ coph]
            ELSE LET e ≜ CHOOSE e ∈ ss.edge : e.from = uh ∧ ClientOf(e.cop) ≠ ClientOf(cop)
                     copprime ≜ e.cop
                     uprime ≜ e.to
                     vprime ≜ vh ∪ {copprime.oid}
                     coph2copprime ≜ COT(coph, copprime)
                     copprime2coph ≜ COT(copprime, coph)
                 IN  xFormHelper(uprime, vprime, coph2copprime,
                         xss ⊕ [node ↦ {vprime},
                                edge ↦ {[from ↦ vh, to ↦ vprime, cop ↦ copprime2coph],
                                        [from ↦ uprime, to ↦ vprime, cop ↦ coph2copprime]}])
    IN  xFormHelper(u, v, cop, [node ↦ {v}, edge ↦ {[from ↦ u, to ↦ v, cop ↦ cop]}])
ClientPerform(c, cop) ≜
    LET xform ≜ xForm(cop, c2ss[c], ds[c])  xform: [xss, xcop]
    IN  ∧ c2ss′ = [c2ss EXCEPT ![c] = @ ⊕ xform.xss]
        ∧ SetNewAop(c, xform.xcop.op)
ServerPerform(cop) ≜
```

```
48      LET c ≜ ClientOf(cop)
49       scur ≜ ds[Server]
50      xform ≜ xForm(cop, s2ss[c], scur)   xform: [xss, xcop]
51       xcop ≜ xform.xcop
52       xcur ≜ scur ∪ {cop.oid}
53       IN   ∧ s2ss' = [cl ∈ Client ↦
54                         IF cl = c
55                          THEN s2ss[cl] ⊕ xform.xss
56                          ELSE s2ss[cl] ⊕ [node ↦ {xcur},
57                                           edge ↦ {[from ↦ scur, to ↦ xcur, cop ↦ xcop]}]]
58            ∧ SetNewAop(Server, xcop.op)
59            ∧ Comm!SSendSame(c, xcop)
60 ------------------------------------------------------------------------
61 DoOp(c, op) ≜
62       LET cop ≜ [op ↦ op, oid ↦ [c ↦ c, seq ↦ cseq[c]], ctx ↦ ds[c]]
63        IN   ∧ ClientPerform(c, cop)
64             ∧ Comm!CSend(cop)

66 Do(c) ≜
67       ∧ DoInt(DoOp, c)
68       ∧ DoCtx(c)
69       ∧ UNCHANGED s2ss

71 Rev(c) ≜
72       ∧ RevInt(ClientPerform, c)
73       ∧ RevCtx(c)
74       ∧ UNCHANGED s2ss

76 SRev ≜
77      ∧ SRevInt(ServerPerform)
78      ∧ SRevCtx
79      ∧ UNCHANGED c2ss
80 ------------------------------------------------------------------------
81 Next ≜
82      ∨ ∃ c ∈ Client : Do(c) ∨ Rev(c)
83      ∨ SRev

85 Fairness ≜
86      WF_vars(SRev ∨ ∃ c ∈ Client : Rev(c))

88 Spec ≜ Init ∧ □[Next]_vars   ∧ Fairness
89 ------------------------------------------------------------------------
90 CSSync ≜   Each client c ∈ Client is synchonized with the Server.
91      ∀ c ∈ Client : (ds[c] = ds[Server]) ⇒ c2ss[c] = s2ss[c]

93 THEOREM Spec ⇒ □CSSync
94 ------------------------------------------------------------------------
```

\ * Modification History
\ * Last modified *Tue Jan* 08 07:13:59 *CST* 2019 by *hengxin*
\ * Created *Tue Oct* 09 16:33:18 *CST* 2018 by *hengxin*