```
1  ---------- MODULE XJupiter ----------
   Specification of the Jupiter protocol described in CSCW'2014 by Yi Xu, Chengzheng Sun, and
   Mo Li. We call it XJupiter, with 'X' for "Xu".
7  EXTENDS JupiterCtx, GraphsUtil
8  ----------
9  VARIABLES
       The 2D state spaces (ss, for short). Each client maintains one 2D state space. The server
       maintains n 2D state spaces, one for each client.
15     c2ss,    c2ss[c]: the 2D state space at client c ∈ Client
16     s2ss     s2ss[c]: the 2D state space maintained by the Server for client c ∈ Client
18 vars ≜ ⟨intVars, ctxVars, c2ss, s2ss⟩
19 ----------
   Direction flags for edges in 2D state spaces and OT.
23 Local ≜ 0
24 Remote ≜ 1
   A 2D state space is a directed graph with labeled edges. It is represented by a record with node
   field and edge field. Each node is characterized by its context, a set of operations. Each edge is
   labeled with an operation and a direction flag indicating whether this edge is LOCAL or REMOTE.
   For clarity, we denote edges by records instead of tuples.
33 IsSS(G) ≜
34     ∧ G = [node ↦ G.node, edge ↦ G.edge]
35     ∧ G.node ⊆ (SUBSET Oid)
36     ∧ G.edge ⊆ [from : G.node, to : G.node, cop : Cop, lr : {Local, Remote}]
38 TypeOK ≜
39     ∧  TypeOKInt
40     ∧  TypeOKCtx
41     ∧  Comm(Cop)!TypeOK
42     ∧  ∀ c ∈ Client : IsSS(c2ss[c]) ∧ IsSS(s2ss[c])
43 ----------
44 Init ≜
45     ∧ InitInt
46     ∧ InitCtx
47     ∧ Comm(Cop)!Init
48     ∧ c2ss = [c ∈ Client ↦ EmptyGraph]
49     ∧ s2ss = [c ∈ Client ↦ EmptyGraph]
50 ----------
   Locate the node in the 2D state space ss which matches the context ctx of cop.
54 Locate(cop, ss) ≜ CHOOSE n ∈ ss.node : n = cop.ctx
   xForm: iteratively transform cop with a path through the 2D state space ss at some client,
   following the edges with the direction flag d.
60 xForm(cop, ss, current, d) ≜
61     LET u ≜ Locate(cop, ss)
62         v ≜ u ∪ {cop.oid}
63         RECURSIVE xFormHelper(_, _, _, _, _)
```

```
64          'h' stands for "helper"; xss: eXtra ss created during transformation
65        xFormHelper(uh, vh, coph, xss, xcoph) ≜
66            IF uh = current
67             THEN ⟨xss, xcoph⟩
68             ELSE LET e ≜ CHOOSE e ∈ ss.edge : e.from = uh ∧ e.lr = d
69                      uprime ≜ e.to
70                      copprime ≜ e.cop
71                      coph2copprime ≜ COT(coph, copprime)
72                      copprime2coph ≜ COT(copprime, coph)
73                       vprime ≜ vh ∪ {copprime.oid}
74                  IN   xFormHelper(uprime, vprime, coph2copprime,
75                          [node ↦ xss.node ∪ {vprime},
76                           edge ↦ xss.edge ∪ {[from ↦ vh, to ↦ vprime, cop ↦ copprime2coph, lr ↦ d],
77                                              [from ↦ uprime, to ↦ vprime, cop ↦ coph2copprime, lr ↦ 1 − d]}],
78                                            coph2copprime)
79    IN   xFormHelper(u, v, cop, [node ↦ {v}, edge ↦ {[from ↦ u, to ↦ v, cop ↦ cop, lr ↦ 1 − d]}], cop)
80 ⊢──────────────────────────────────────────────────────────────────────────────────────────────────────⊣
   Client c ∈ Client perform operation cop guided by the direction flag d.
84 ClientPerform(cop, c, d) ≜
85    LET xform ≜ xForm(cop, c2ss[c], ds[c], d)   xform: ⟨xss, xcop⟩
86          xss ≜ xform[1]
87         xcop ≜ xform[2]
88    IN   ∧ c2ss′ = [c2ss EXCEPT ![c] = @ ⊕ xss]
89         ∧ state′ = [state EXCEPT ![c] = Apply(xcop.op, @)]
   Client c ∈ Client generates an operation op.
93 DoOp(c, op)  ≜
94       LET cop ≜ [op ↦ op, oid ↦ [c ↦ c, seq ↦ cseq′[c]], ctx ↦ ds[c]]
95           IN   ∧ ClientPerform(cop, c, Remote)
96                ∧ UpdateDS(c, cop)
97                ∧ Comm(Cop)!CSend(cop)

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

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

109 Do(c) ≜
110      ∧ DoCtx(c)
111      ∧ ∨ DoIns(c)
112        ∨ DoDel(c)
```

```
113         ∧ UNCHANGED s2ss
    Client c ∈ Client receives a message from the Server.
117 Rev(c) ≜
118         ∧ Comm(Cop)!CRev(c)
119         ∧ LET cop ≜ Head(cincoming[c])  the received (transformed) operation
120             IN  ClientPerform(cop, c, Local)
121         ∧ RevCtx(c)
122         ∧ UNCHANGED ⟨chins, s2ss⟩
123 ├──────────────────────────────────────────────────────────────────────┤
    The Server performs operation cop.
127 ServerPerform(cop) ≜
128     LET c ≜ cop.oid.c
129       scur ≜ ds[Server]
130       xform ≜ xForm(cop, s2ss[c], scur, Remote)  xform: ⟨xss, xcop⟩
131       xss ≜ xform[1]
132       xcop ≜ xform[2]
133       xcur ≜ scur ∪ {cop.oid}
134     IN  ∧ s2ss′ = [cl ∈ Client ↦
135                     IF cl = c
136                     THEN s2ss[cl] ⊕ xss
137                     ELSE s2ss[cl] ⊕ [node ↦ {xcur},
138                                      edge ↦ {[from ↦ scur, to ↦ xcur,
139                                               cop ↦ xcop, lr ↦ Remote]}]
140                   ]
141         ∧ state′ = [state EXCEPT ![Server] = Apply(xcop.op, @)]
142         ∧ Comm(Cop)!SSendSame(c, xcop)  broadcast the transformed operation
    The Server receives a message.
146 SRev ≜
147         ∧ Comm(Cop)!SRev
148         ∧ LET cop ≜ Head(sincoming)
149             IN  ServerPerform(cop)
150         ∧ SRevCtx
151         ∧ UNCHANGED ⟨chins, c2ss⟩
152 ├──────────────────────────────────────────────────────────────────────┤
153 Next ≜
154         ∨ ∃ c ∈ Client : Do(c) ∨ Rev(c)
155         ∨ SRev

157 Fairness ≜
158     WF_vars(SRev ∨ ∃ c ∈ Client : Rev(c))

160 Spec ≜ Init ∧ □[Next]_vars  ∧ Fairness
161 ├──────────────────────────────────────────────────────────────────────┤
    In Jupiter (not limited to XJupiter), each client synchronizes with the server. In XJupiter, this
    is expressed as the following CSSync property.
```

166 $CSSync \triangleq$

167 $\quad \forall\, c \in Client : (ds[c] = ds[Server]) \Rightarrow c2ss[c] = s2ss[c]$

168

\ * Modification History
\ * Last modified *Wed Dec* 19 11:41:44 *CST* 2018 by *hengxin*
\ * Created *Tue Oct* 09 16:33:18 *CST* 2018 by *hengxin*