

```

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 StateSpace |
8  |-----|
9  | VARIABLES |
   | The 2D state spaces (2ss, 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 |-----|
20 |   TypeOK ≜ |
21 |     ∧ TypeOKInt |
22 |     ∧ TypeOKCtx |
23 |     ∧ Comm(Cop)! TypeOK |
24 |     ∧ ∀ c ∈ Client : IsSS(c2ss[c]) ∧ IsSS(s2ss[c]) |
25 |-----|
26 |   Init ≜ |
27 |     ∧ InitInt |
28 |     ∧ InitCtx |
29 |     ∧ Comm(Cop)! Init |
30 |     ∧ c2ss = [c ∈ Client ↦ EmptySS] |
31 |     ∧ s2ss = [c ∈ Client ↦ EmptySS] |
32 |-----|
   | xForm: iteratively transform cop with a path through the 2D state space ss at some client. |
37 |   xForm(cop, ss, cur) ≜ |
38 |     LET u ≜ Locate(cop, ss) |
39 |     v ≜ u ∪ {cop.oid} |
40 |     RECURSIVE xFormHelper(−, −, −, −) |
41 |     xFormHelper(uh, vh, coph, xss) ≜ xss: eXtra ss created during transformation |
42 |     IF uh = cur THEN [xss ↦ xss, xcop ↦ coph] |
43 |     ELSE LET e ≜ CHOOSE e ∈ ss.edge : e.from = uh ∧ ClientOf(e.cop) ≠ ClientOf(cop) |
44 |       copprime ≜ e.cop |
45 |       uprime ≜ e.to |
46 |       vprime ≜ vh ∪ {copprime.oid} |
47 |       coph2copprime ≜ COT(coph, copprime) |
48 |       copprime2coph ≜ COT(copprime, coph) |
49 |       IN xFormHelper(uprime, vprime, coph2copprime, |
50 |         xss ⊕ [node ↦ {vprime}, |
51 |           edge ↦ {[from ↦ vh, to ↦ vprime, cop ↦ copprime2coph], |
52 |             [from ↦ uprime, to ↦ vprime, cop ↦ coph2copprime]}]) |
53 |       IN xFormHelper(u, v, cop, [node ↦ {v}, edge ↦ {[from ↦ u, to ↦ v, cop ↦ cop]}]) |
54 |-----|

```

Client $c \in \text{Client}$ perform operation cop .

58 $\text{ClientPerform}(\text{cop}, c) \triangleq$
59 LET $xform \triangleq xForm(\text{cop}, c2ss[c], ds[c])$ $xform: [xss, xcop]$
60 IN $\wedge c2ss' = [c2ss \text{ EXCEPT } ![c] = @ \oplus xform.xss]$
61 $\wedge state' = [state \text{ EXCEPT } ![c] = Apply(xform.xcop.op, @)]$

Client $c \in \text{Client}$ generates an operation op .

65 $\text{DoOp}(c, op) \triangleq$
66 LET $\text{cop} \triangleq [op \mapsto op, oid \mapsto [c \mapsto c, seq \mapsto cseq'[c]], ctx \mapsto ds[c]]$
67 IN $\wedge \text{ClientPerform}(\text{cop}, c)$
68 $\wedge \text{UpdateDS}(c, \text{cop})$
69 $\wedge \text{Comm}(\text{Cop})!C\text{Send}(\text{cop})$

71 $\text{DoIns}(c) \triangleq$
72 $\exists ins \in \{op \in \text{Ins} : op.pos \in 1 \dots (Len(state[c]) + 1) \wedge op.ch \in chins \wedge op.pr = Priority[c]\} :$
73 $\wedge \text{DoOp}(c, ins)$
74 $\wedge chins' = chins \setminus \{ins.ch\}$ We assume that all inserted elements are unique.

76 $\text{DoDel}(c) \triangleq$
77 $\exists del \in \{op \in \text{Del} : op.pos \in 1 \dots Len(state[c])\} :$
78 $\wedge \text{DoOp}(c, del)$
79 $\wedge \text{UNCHANGED } chins$

81 $\text{Do}(c) \triangleq$
82 $\wedge \text{DoCtx}(c)$
83 $\wedge \vee \text{DoIns}(c)$
84 $\vee \text{DoDel}(c)$
85 $\wedge \text{UNCHANGED } s2ss$

Client $c \in \text{Client}$ receives a message from the *Server*.

89 $\text{Rev}(c) \triangleq$
90 $\wedge \text{Comm}(\text{Cop})!C\text{Rev}(c)$
91 $\wedge \text{LET } \text{cop} \triangleq \text{Head}(cincoming[c])$ the received (transformed) operation
92 IN $\text{ClientPerform}(\text{cop}, c)$
93 $\wedge \text{RevCtx}(c)$
94 $\wedge \text{UNCHANGED } \langle chins, s2ss \rangle$

The *Server* performs operation cop .

99 $\text{ServerPerform}(\text{cop}) \triangleq$
100 LET $c \triangleq \text{ClientOf}(\text{cop})$
101 $scur \triangleq ds[\text{Server}]$
102 $xform \triangleq xForm(\text{cop}, s2ss[c], scur)$ $xform: [xss, xcop]$
103 $xcop \triangleq xform.xcop$
104 $xcur \triangleq scur \cup \{cop.oid\}$
105 IN $\wedge s2ss' = [cl \in \text{Client} \mapsto$
106 IF $cl = c$
107 THEN $s2ss[cl] \oplus xform.xss$

```

108             ELSE  $s2ss[cl] \oplus [node \mapsto \{xcur\},$ 
109                  $edge \mapsto \{[from \mapsto scur, to \mapsto xcur, cop \mapsto xcop]\}]$ 
110             ]
111              $\wedge state' = [state \text{ EXCEPT } ![Server] = Apply(xcop.op, @)]$ 
112              $\wedge Comm(Cop)!SSendSame(c, xcop)$  broadcast the transformed operation
113         The Server receives a message.
114     SRev  $\triangleq$ 
115          $\wedge Comm(Cop)!SRev$ 
116          $\wedge \text{LET } cop \triangleq Head(sincoming)$ 
117         IN  $ServerPerform(cop)$ 
118          $\wedge SRevCtx$ 
119          $\wedge \text{UNCHANGED } \langle chins, c2ss \rangle$ 
120
121 Next  $\triangleq$ 
122      $\vee \exists c \in Client : Do(c) \vee Rev(c)$ 
123      $\vee SRev$ 
124
125 Fairness  $\triangleq$ 
126      $WF_{vars}(SRev \vee \exists c \in Client : Rev(c))$ 
127
128 Spec  $\triangleq Init \wedge \Box[Next]_{vars} \wedge Fairness$ 
129
130 In Jupiter (not limited to XJupiter), each client synchronizes with the server. In XJupiter, this
131 is expressed as the following CSSync property.
132
133 CSSync  $\triangleq$ 
134      $\forall c \in Client : (ds[c] = ds[Server]) \Rightarrow c2ss[c] = s2ss[c]$ 
135
136 \ * Modification History
137 \ * Last modified Mon Dec 24 11:38:04 CST 2018 by hengxin
138 \ * Created Tue Oct 09 16:33:18 CST 2018 by hengxin

```