

```

1  |----- MODULE CJupiter -----|
   | Model of our own CJupiter protocol. |
5  | EXTENDS JupiterSerial |
6  |-----|
7  VARIABLES
8    cseq,      cseq[c]: local sequence number at client  $c \in Client$ 
   | For all replicas: the  $n$ -ary ordered state space |
12   css,      css[r]: the  $n$ -ary ordered state space at replica  $r \in Replica$ 
13   cur      cur[r]: the current node of  $css$  at replica  $r \in Replica$ 
15   vars  $\triangleq \langle chins, cseq, css, cur, state, cincoming, sincoming, serialVars \rangle$ 
16 |-----|
   | A  $css$  is a directed graph with labeled edges, represented by a record with node field and edge field. |
   | Each node is characterized by its context, a set of oids. Each edge is labeled with an operation. |
23   IsCSS( $G$ )  $\triangleq$ 
24      $\wedge G = [node \mapsto G.node, edge \mapsto G.edge]$ 
25      $\wedge G.node \subseteq (SUBSET\ Oid)$ 
26      $\wedge G.edge \subseteq [from : G.node, to : G.node, cop : Cop]$ 
28   EmptySS  $\triangleq [node \mapsto \{\{\}\}, edge \mapsto \{\}]$ 
30   TypeOK  $\triangleq$ 
31      $\wedge TypeOKInt$ 
32      $\wedge TypeOKSerial$ 
33      $\wedge Comm(Cop)!TypeOK$ 
34      $\wedge cseq \in [Client \rightarrow Nat]$ 
   | For all replicas: the  $n$ -ary ordered state space |
38      $\wedge \forall r \in Replica : IsCSS(css[r])$ 
39      $\wedge cur \in [Replica \rightarrow SUBSET\ Oid]$ 
40 |-----|
41   Init  $\triangleq$ 
42      $\wedge InitInt$ 
43      $\wedge InitSerial$ 
44      $\wedge Comm(Cop)!Init$ 
45      $\wedge cseq = [c \in Client \mapsto 0]$ 
   | For all replicas: the  $n$ -ary ordered state space |
49      $\wedge css = [r \in Replica \mapsto EmptySS]$ 
50      $\wedge cur = [r \in Replica \mapsto \{\}]$ 
51 |-----|
   | Locate the node in  $rcss$  (the  $css$  at replica  $r \in Replica$ ) that matches the context  $ctx$  of  $cop$ . |
55   Locate( $cop, rcss$ )  $\triangleq$  CHOOSE  $n \in rcss.node : n = cop.ctx$ 
   | Take union of two state spaces  $ss1$  and  $ss2$ . |
59    $ss1 \oplus ss2 \triangleq [node \mapsto ss1.node \cup ss2.node, edge \mapsto ss1.edge \cup ss2.edge]$ 

```

xForm: Iteratively transform *cop* with a path through the *css* at replica $r \in \text{Replica}$, following the first edges.

```

64  $xForm(cop, r) \triangleq$ 
65   LET  $rcss \triangleq css[r]$ 
66    $u \triangleq Locate(cop, rcss)$ 
67    $v \triangleq u \cup \{cop.oid\}$ 
68   RECURSIVE  $xFormHelper(-, -, -, -, -, -)$ 
69   'h' stands for "helper";  $xcss$ : eXtra css created during transformation
70    $xFormHelper(uh, vh, coph, xcss, xcoph, xcurh) \triangleq$ 
71     IF  $uh = cur[r]$ 
72     THEN  $\langle xcss, xcoph, xcurh \rangle$ 
73     ELSE LET  $fedge \triangleq$  CHOOSE  $e \in rcss.edge :$ 
74        $\wedge e.from = uh$ 
75        $\wedge \forall uhe \in rcss.edge :$ 
76          $(uhe.from = uh \wedge uhe \neq e) \Rightarrow tb(e.cop.oid, uhe.cop.oid, serial[r])$ 
77        $uprime \triangleq fedge.to$ 
78        $fcop \triangleq fedge.cop$ 
79        $coph2fcop \triangleq COT(coph, fcop)$ 
80        $fcop2coph \triangleq COT(fcop, coph)$ 
81        $vprime \triangleq vh \cup \{fcop.oid\}$ 
82     IN  $xFormHelper(uprime, vprime, coph2fcop,$ 
83        $[xcss \text{ EXCEPT } !.node = @ \cup \{vprime\},$ 
84        $!.edge = @ \cup \{[from \mapsto vh, to \mapsto vprime, cop \mapsto fcop2coph],$ 
85        $[from \mapsto uprime, to \mapsto vprime, cop \mapsto coph2fcop]\},$ 
86        $coph2fcop, vprime)$ 
87   IN  $xFormHelper(u, v, cop, [node \mapsto \{v\}, edge \mapsto \{[from \mapsto u, to \mapsto v, cop \mapsto cop]\}], cop, v)$ 

```

Perform *cop* at replica $r \in \text{Replica}$.

```

91  $Perform(cop, r) \triangleq$ 
92   LET  $xform \triangleq xForm(cop, r)$   $xform: \langle xcss, xcoph, xcur \rangle$ 
93    $xcss \triangleq xform[1]$ 
94    $xcop \triangleq xform[2]$ 
95    $xcur \triangleq xform[3]$ 
96   IN  $\wedge css' = [css \text{ EXCEPT } ![r] = @ \oplus xcss]$ 
97    $\wedge cur' = [cur \text{ EXCEPT } ![r] = xcur]$ 
98    $\wedge state' = [state \text{ EXCEPT } ![r] = Apply(xcop.op, @)]$ 
99 |

```

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

```

103  $DoOp(c, op) \triangleq$   $op$ : the raw operation generated by the client  $c \in \text{Client}$ 
104    $\wedge cseq' = [cseq \text{ EXCEPT } ![c] = @ + 1]$ 
105    $\wedge$  LET  $cop \triangleq [op \mapsto op, oid \mapsto [c \mapsto c, seq \mapsto cseq'[c]], ctx \mapsto cur[c]]$ 
106   IN  $\wedge Perform(cop, c)$ 
107    $\wedge Comm(Cop)!C\text{Send}(cop)$ 

```

```

109  $DoIns(c) \triangleq$ 
110    $\exists ins \in \{op \in Ins : op.pos \in 1 \dots (Len(state[c]) + 1) \wedge op.ch \in chins \wedge op.pr = Priority[c]\} :$ 

```

```

111       $\wedge DoOp(c, ins)$ 
112       $\wedge chins' = chins \setminus \{ins.ch\}$  We assume that all inserted elements are unique.

114   $DoDel(c) \triangleq$ 
115       $\exists del \in \{op \in Del : op.pos \in 1 \dots Len(state[c])\} :$ 
116       $\wedge DoOp(c, del)$ 
117       $\wedge UNCHANGED\ chins$ 

119   $Do(c) \triangleq$ 
120       $\wedge DoSerial(c)$ 
121       $\wedge \vee DoIns(c)$ 
122       $\vee DoDel(c)$ 
Client  $c \in Client$  receives a message from the Server.

126   $Rev(c) \triangleq$ 
127       $\wedge Comm(Cop)!CRev(c)$ 
128       $\wedge Perform(Head(cincoming[c]), c)$ 
129       $\wedge RevSerial(c)$ 
130       $\wedge UNCHANGED\ \langle chins, cseq \rangle$ 
131  |-----|
The Server receives a message.

135   $SRev \triangleq$ 
136       $\wedge Comm(Cop)!SRev$ 
137       $\wedge LET\ cop \triangleq Head(sincoming)$ 
138       $IN\ \wedge Perform(cop, Server)$ 
139       $\wedge Comm(Cop)!SSendSame(cop.oid.c, cop)$  broadcast the original operation
140       $\wedge SRevSerial$ 
141       $\wedge UNCHANGED\ \langle chins, cseq \rangle$ 
142  |-----|

143   $Next \triangleq$ 
144       $\vee \exists c \in Client : Do(c) \vee Rev(c)$ 
145       $\vee SRev$ 
Fairness: There is no requirement that the clients ever generate operations.

149   $Fairness \triangleq$ 
150       $WF_{vars}(SRev \vee \exists c \in Client : Rev(c))$ 

152   $Spec \triangleq Init \wedge \Box[Next]_{vars}$   $\wedge Fairness$  (We care more about safety.)
153  |-----|
The compactness of  $CJupiter$ : the  $CSSes$  at all replicas are the same.

157   $Compactness \triangleq$ 
158       $Comm(Cop)!EmptyChannel \Rightarrow Cardinality(Range(css)) = 1$ 

160  THEOREM  $Spec \Rightarrow Compactness$ 
161  |-----|

  \ * Modification History
  \ * Last modified Thu Dec 13 09:31:36 CST 2018 by hengxin
  \ * Created Sat Sep 01 11:08:00 CST 2018 by hengxin

```