

```

1  |----- MODULE CJupiter -----|
   | Model of our own CJupiter protocol. |
5  | EXTENDS StateSpace, JupiterSerial |
6  |-----|
7  | VARIABLES
8  |     css      | css[r]: the n-ary ordered state space at replica  $r \in Replica$  |
10 | vars  $\triangleq$   $\langle intVars, ctxVars, serialVars, css \rangle$ 
11 |-----|
12 | TypeOK  $\triangleq$ 
13 |      $\wedge$  TypeOKInt
14 |      $\wedge$  TypeOKCtx
15 |      $\wedge$  TypeOKSerial
16 |      $\wedge$  Comm(Cop)!TypeOK
17 |      $\wedge$   $\forall r \in Replica : IsSS(css[r])$ 
18 |-----|
19 | Init  $\triangleq$ 
20 |      $\wedge$  InitInt
21 |      $\wedge$  InitCtx
22 |      $\wedge$  InitSerial
23 |      $\wedge$  Comm(Cop)!Init
24 |      $\wedge$   $css = [r \in Replica \mapsto EmptySS]$ 
25 |-----|
   | xForm: Iteratively transform cop with a path through the css at replica  $r \in Replica$ , following |
   | the first edges. |
30 | xForm(cop, r)  $\triangleq$ 
31 |     LET rcss  $\triangleq$  css[r]
32 |     u  $\triangleq$  Locate(cop, rcss)
33 |     v  $\triangleq$  u  $\cup$  {cop.oid}
34 |     RECURSIVE xFormHelper(–, –, –, –)
35 |     xFormHelper(uh, vh, coph, xcsc)  $\triangleq$  | xcsc: eXtra css created during transformation
36 |     IF uh = ds[r] THEN [xcsc  $\mapsto$  xcsc, xcop  $\mapsto$  coph]
37 |     ELSE LET fedge  $\triangleq$  CHOOSE  $e \in rcsc.edge$  :
38 |          $\wedge e.from = uh$ 
39 |          $\wedge \forall uhe \in rcsc.edge \setminus \{e\} :$ 
40 |              $(uhe.from = uh) \Rightarrow tb(e.cop.oid, uhe.cop.oid, serial[r])$ 
41 |         uprime  $\triangleq$  fedge.to
42 |         fcop  $\triangleq$  fedge.cop
43 |         coph2fcop  $\triangleq$  COT(coph, fcop)
44 |         fcop2coph  $\triangleq$  COT(fcop, coph)
45 |         vprime  $\triangleq$  vh  $\cup$  {fcop.oid}
46 |     IN xFormHelper(uprime, vprime, coph2fcop,
47 |         xcsc  $\oplus$  [node  $\mapsto$  {vprime},
48 |             edge  $\mapsto$  {[from  $\mapsto$  vh, to  $\mapsto$  vprime, cop  $\mapsto$  fcop2coph],
49 |                 [from  $\mapsto$  uprime, to  $\mapsto$  vprime, cop  $\mapsto$  coph2fcop]}])

```

IN $xFormHelper(u, v, cop, [node \mapsto \{v\}, edge \mapsto \{[from \mapsto u, to \mapsto v, cop \mapsto cop]\}])$
 52 $Perform(cop, r) \triangleq$ Perform cop at replica $r \in Replica.$
 53 LET $xform \triangleq xForm(cop, r)$ $xform: [xcss, xcop]$
 54 IN $\wedge css' = [css \text{ EXCEPT } ![r] = @ \oplus xform.xcss]$
 55 $\wedge SetNewAop(r, xform.xcop.op)$

56
 57 $DoOp(c, op) \triangleq$
 58 \wedge LET $cop \triangleq [op \mapsto op, oid \mapsto [c \mapsto c, seq \mapsto cseq'[c]], ctx \mapsto ds[c]]$
 59 IN $\wedge Perform(cop, c)$
 60 $\wedge Comm(Cop)!CSend(cop)$

61
 62 $Do(c) \triangleq$
 63 $\wedge DoCtx(c)$
 64 $\wedge DoSerial(c)$
 65 $\wedge DoInt(DoOp, c)$

66
 67 $Rev(c) \triangleq$
 68 $\wedge Comm(Cop)!CRev(c)$
 69 $\wedge Perform(Head(cincoming[c]), c)$
 70 $\wedge RevSerial(c)$
 71 $\wedge RevCtx(c)$
 72 $\wedge RevInt(c)$

73
 74 $SRev \triangleq$
 75 $\wedge Comm(Cop)!SRev$
 76 \wedge LET $cop \triangleq Head(sincoming)$
 77 IN $\wedge Perform(cop, Server)$
 78 $\wedge Comm(Cop)!SSendSame(cop.oid.c, cop)$ broadcast the original operation
 79 $\wedge SRevSerial$
 80 $\wedge SRevCtx$
 81 $\wedge SRevInt$

82
 83 $Next \triangleq$
 84 $\vee \exists c \in Client : Do(c) \vee Rev(c)$
 85 $\vee SRev$

86
 87 $Fairness \triangleq$ There is no requirement that the clients ever generate operations.
 88 $WF_{vars}(SRev \vee \exists c \in Client : Rev(c))$

89
 90 $Spec \triangleq Init \wedge \Box [Next]_{vars} \wedge Fairness$ (We care more about safety.)

91
 92 $Compactness \triangleq$ Compactness of $CJupiter$: the $CSSes$ at all replicas are the same.
 93 $Comm(Cop)!EmptyChannel \Rightarrow Cardinality(Range(css)) = 1$

94 THEOREM $Spec \Rightarrow Compactness$

* Modification History
* Last modified *Tue Jan 01 11:36:24 CST 2019* by *hengxin*
* Created Sat *Sep 01 11:08:00 CST 2018* by *hengxin*