```
1  ---------- MODULE AbsJupiter ----------
   Abstract Jupiter, inspired by the COT algorithm proposed by Sun and Sun; see TPDS'2009.
5  EXTENDS JupiterSerial, SetStateSpace
6  ----------
7  VARIABLES
8      copss    copss[r]: the state space (i.e., a set) of Cop maintained at replia r ∈ Replica
10 vars ≜ ⟨intVars, ctxVars, serialVars, copss⟩
11 ----------
12 TypeOK ≜
13     ∧  TypeOKInt
14     ∧  TypeOKCtx
15     ∧  TypeOKSerial
16     ∧  copss ∈ [Replica → SUBSET Cop]
17 ----------
18 Init ≜
19     ∧ InitInt
20     ∧ InitCtx
21     ∧ InitSerial
22     ∧ copss = [r ∈ Replica ↦ {}]
23 ----------
24 NextCop(r, cop, ss, ctx) ≜  Return the next fcop ∈ Cop against which cop is to be transformed.
25     LET foid ≜ CHOOSE oid ∈ ctx :  the first oid in ctx according to serial[r]
26                    ∀ id ∈ ctx \ {oid} : tb(oid, id, serial[r])
27     IN  CHOOSE fcop ∈ ss :  THEOREM : Existence of fcop
28             fcop.oid = foid ∧ fcop.ctx = cop.ctx
30 Perform(r, cop) ≜
31     LET xform ≜ xForm(NextCop, r, cop, copss[r])   [xcop, xss]
32     IN  ∧ copss' = [copss EXCEPT ![r] = xform.xss]
33         ∧ SetNewAop(r, xform.xcop.op)
35 ClientPerform(c, cop) ≜ Perform(c, cop)
37 ServerPerform(cop) ≜
38     ∧ Perform(Server, cop)
39     ∧ Comm!SSendSame(ClientOf(cop), cop)
40 ----------
41 DoOp(c, op)  ≜
42       LET cop ≜ [op ↦ op, oid ↦ [c ↦ c, seq ↦ cseq[c]], ctx ↦ ds[c]]
43       IN  ∧ ClientPerform(c, cop)
44           ∧ Comm!CSend(cop)
46 Do(c) ≜
47       ∧ DoInt(DoOp, c)
48       ∧ DoCtx(c)
```

```
        ∧ DoSerial(c)

Rev(c) ≜
        ∧ RevInt(ClientPerform, c)
        ∧ RevCtx(c)
        ∧ RevSerial(c)

SRev ≜
      ∧ SRevInt(ServerPerform)
      ∧ SRevCtx
      ∧ SRevSerial
------------------------------------------------------------------------
Next ≜
      ∨ ∃ c ∈ Client : Do(c) ∨ Rev(c)
      ∨ SRev

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

Spec ≜ Init ∧ □[Next]_vars  ∧ Fairness
------------------------------------------------------------------------
QC ≜  Quiescent Consistency
      Comm!EmptyChannel ⇒ Cardinality(Range(state)) = 1
THEOREM Spec ⇒ □QC

SEC ≜  Strong Eventual Consistency
      ∀ r1, r2 ∈ Replica :
         ds[r1] = ds[r2] ⇒ state[r1] = state[r2]
THEOREM Spec ⇒ □SEC

Compactness ≜  Compactness of state space
     Comm!EmptyChannel ⇒ Cardinality(Range(copss)) = 1
THEOREM Spec ⇒ □Compactness
========================================================================
\* Modification History
\* Last modified Mon Jan 28 20:06:39 CST 2019 by hengxin
\* Created Wed Dec 05 19:55:52 CST 2018 by hengxin
```