```
MODULE AbsJupiter
Abstract Jupiter, inspired by the COT algorithm proposed by Sun and Sun. See their paper published on TPDS'2009.
EXTENDS JupiterSerial

VARIABLES
    cseq,     cseq[c]: local sequence number at client c ∈ client
    ds,       ds[r]: document state at replica r ∈ Replica
    copss     copss[r]: the state space (i.e., a set) of Cops maintained at replia r ∈ Replica

vars ≜ ⟨intVars, serialVars, cseq, ds, copss⟩

TypeOK ≜
    ∧ TypeOKInt
    ∧ TypeOKSerial
    ∧ Comm(Cop)!TypeOK
    ∧ cseq ∈ [Client → Nat]
    ∧ ds ∈ [Replica → SUBSET Oid]
    ∧ copss ∈ [Replica → SUBSET Cop]

Init ≜
    ∧ InitInt
    ∧ InitSerial
    ∧ Comm(Cop)!Init
    ∧ cseq = [c ∈ Client ↦ 0]
    ∧ ds = [r ∈ Replica ↦ {}]
    ∧ copss = [r ∈ Replica ↦ {}]

RECURSIVE xForm(_, _)
xForm(cop, r) ≜
  LET ctxDiff ≜ ds[r] \ cop.ctx   THEOREM : cop.ctx ⊆ ds[r]
      RECURSIVE xFormHelper(_, _, _)
      xFormHelper(coph, ctxDiffh, copssr) ≜   'h' stands for "helper"
        IF ctxDiffh = {}
        THEN ⟨coph, copssr ∪ {coph}⟩
        ELSE LET foph ≜ CHOOSE op ∈ ctxDiffh :   the first op (specifically, oid) in serial
                          ∀ opprime ∈ ctxDiffh :
                            opprime ≠ op ⇒ tb(op, opprime, serial[r])
                 fcophDict ≜ {op ∈ copssr : op.oid = foph ∧ op.ctx = coph.ctx}
                 fcoph ≜ CHOOSE op ∈ fcophDict : TRUE   THEOREM : Cardinality(fophDict) = 1
                 cophx ≜ COT(coph, fcoph)
                 fcophx ≜ COT(fcoph, coph)
             IN  xFormHelper(cophx, ctxDiffh \ {foph}, copssr ∪ {cophx, fcophx})
  IN xFormHelper(cop, ctxDiff, copss[r])
```

```
48  Perform(cop, r) ≜
49      LET xform ≜ xForm(cop, r)   ⟨xcop, xcopss⟩
50          xcop ≜ xform[1]
51           xcopssr ≜ xform[2]
52       IN   ∧ state' = [state EXCEPT ![r] = Apply(xcop.op, @)]
53            ∧ ds' = [ds EXCEPT ![r] = @ ∪ {cop.oid}]
54            ∧ copss' = [copss EXCEPT ![r] = xcopssr]
55 ----------------------------------------------------------------------
   Client c ∈ Client issues an operation op.
59  DoOp(c, op) ≜   op: the raw operation generated by the client c ∈ Client
60        ∧ cseq' = [cseq EXCEPT ![c] = @ + 1]
61        ∧ LET cop ≜ [op ↦ op, oid ↦ [c ↦ c, seq ↦ cseq'[c]], ctx ↦ ds[c]]
62          IN   ∧ Perform(cop, c)
63               ∧ Comm(Cop)!CSend(cop)

65  DoIns(c) ≜
66      ∃ ins ∈ {op ∈ Ins : op.pos ∈ 1 .. (Len(state[c]) + 1) ∧ op.ch ∈ chins ∧ op.pr = Priority[c]} :
67        ∧ DoOp(c, ins)
68        ∧ chins' = chins \ {ins.ch}   We assume that all inserted elements are unique.

70  DoDel(c) ≜
71      ∃ del ∈ {op ∈ Del : op.pos ∈ 1 .. Len(state[c])} :
72        ∧ DoOp(c, del)
73        ∧ UNCHANGED chins

75  Do(c) ≜
76        ∧ DoSerial(c)
77        ∧ ∨ DoIns(c)
78          ∨ DoDel(c)
79 ----------------------------------------------------------------------
80  Rev(c) ≜
81        ∧ Comm(Cop)!CRev(c)
82        ∧ Perform(Head(cincoming[c]), c)
83        ∧ RevSerial(c)
84        ∧ UNCHANGED ⟨chins, cseq⟩
85 ----------------------------------------------------------------------
86  SRev ≜
87       ∧ Comm(Cop)!SRev
88       ∧ LET cop ≜ Head(sincoming)
89         IN   ∧ Perform(cop, Server)
90              ∧ Comm(Cop)!SSendSame(cop.oid.c, cop)
91       ∧ SRevSerial
92       ∧ UNCHANGED ⟨chins, cseq⟩
93 ----------------------------------------------------------------------
94  Next ≜
95       ∨ ∃ c ∈ Client : Do(c) ∨ Rev(c)
```

```
96        ∨ SRev

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

101 Spec ≜ Init ∧ □[Next]_vars  ∧ Fairness
102 ----------------------------------------------------------------
103 Compactness ≜
104       Comm(Cop)!EmptyChannel ⇒ Cardinality(Range(copss)) = 1

106 THEOREM Spec ⇒ Compactness
107 ================================================================
      \* Modification History
      \* Last modified Sat Dec 08 17:34:44 CST 2018 by hengxin
      \* Created Wed Dec 05 19:55:52 CST 2018 by hengxin
```