```
┌──────────────────────── MODULE AbsJupiter ────────────────────────┐
Abstract Jupiter, inspired by the COT algorithm proposed by Sun and Sun. See their paper
published on TPDS′2009.
EXTENDS JupiterSerial
├────────────────────────────────────────────────────────────────────┤
VARIABLES
    copss    copss[r]: the state space (i.e., a set) of Cops maintained at replia r ∈ Replica

vars ≜ ⟨intVars, ctxVars, serialVars, copss⟩
├────────────────────────────────────────────────────────────────────┤
TypeOK ≜
    ∧  TypeOKInt
    ∧  TypeOKCtx
    ∧  TypeOKSerial
    ∧  Comm(Cop)!TypeOK
    ∧  copss ∈ [Replica → SUBSET Cop]
├────────────────────────────────────────────────────────────────────┤
Init ≜
    ∧ InitInt
    ∧ InitCtx
    ∧ InitSerial
    ∧ Comm(Cop)!Init
    ∧ 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⟩
            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])

Perform(cop, r) ≜
    LET xform ≜ xForm(cop, r)    ⟨xcop, xcopss⟩
        xcop ≜ xform[1]
        xcopssr ≜ xform[2]
```

```
        IN   ∧ state′  = [state EXCEPT ![r] = Apply(xcop.op, @)]
             ∧ copss′ = [copss EXCEPT ![r] = xcopssr ∪ {cop}]
────────────────────────────────────────────────────────────────
Client c ∈ Client issues an operation op.
DoOp(c, op) ≜   op: the raw operation generated by the client c ∈ Client
      ∧ LET cop ≜ [op ↦ op, oid ↦ [c ↦ c, seq ↦ cseq′[c]], ctx ↦ ds[c]]
          IN   ∧ Perform(cop, c)
               ∧ Comm(Cop)!CSend(cop)

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

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

Do(c) ≜
      ∧ DoCtx(c)
      ∧ DoSerial(c)
      ∧ ∨ DoIns(c)
        ∨ DoDel(c)
────────────────────────────────────────────────────────────────
Rev(c) ≜
      ∧ Comm(Cop)!CRev(c)
      ∧ Perform(Head(cincoming[c]), c)
      ∧ RevSerial(c)
      ∧ RevCtx(c)
      ∧ UNCHANGED chins
────────────────────────────────────────────────────────────────
SRev ≜
      ∧ Comm(Cop)!SRev
      ∧ LET cop ≜ Head(sincoming)
          IN   ∧ Perform(cop, Server)
               ∧ Comm(Cop)!SSendSame(cop.oid.c, cop)
      ∧ SRevSerial
      ∧ SRevCtx
      ∧ UNCHANGED chins
────────────────────────────────────────────────────────────────
Next ≜
      ∨ ∃ c ∈ Client : Do(c) ∨ Rev(c)
      ∨ SRev

Fairness ≜
```

$$\mathrm{WF}_{vars}(\mathit{SRev} \lor \exists\, c \in \mathit{Client} : \mathit{Rev}(c))$$

$$\mathit{Spec} \triangleq \mathit{Init} \land \Box[\mathit{Next}]_{vars} \land \mathit{Fairness}$$

$$\mathit{Compactness} \triangleq$$
$$\quad \mathit{Comm}(\mathit{Cop})!\mathit{EmptyChannel} \Rightarrow \mathit{Cardinality}(\mathit{Range}(\mathit{copss})) = 1$$

THEOREM $\mathit{Spec} \Rightarrow \mathit{Compactness}$

---

* Modification History
* Last modified *Fri Dec* 28 10:57:04 *CST* 2018 by *hengxin*
* Created *Wed Dec* 05 19:55:52 *CST* 2018 by *hengxin*