```
------------------------------ MODULE CJupiter ------------------------------
Model of our own CJupiter protocol.
EXTENDS JupiterSerial, GraphsUtil
-----------------------------------------------------------------------------
VARIABLES
    css     css[r]: the n-ary ordered state space at replica r ∈ Replica

vars ≜ ⟨intVars, ctxVars, serialVars, css⟩
-----------------------------------------------------------------------------
A css is a directed graph (defined in moudule GraphsUtil) with labeled edges, Each node is
characterized by its context, a set of oids. Each edge is labeled with an operation.
IsCSS(G) ≜
    ∧ G = [node ↦ G.node, edge ↦ G.edge]
    ∧ G.node ⊆ (SUBSET Oid)
    ∧ G.edge ⊆ [from : G.node, to : G.node, cop : Cop]

TypeOK ≜
    ∧   TypeOKInt
    ∧   TypeOKCtx
    ∧   TypeOKSerial
    ∧   Comm(Cop)!TypeOK
    ∧   ∀ r ∈ Replica : IsCSS(css[r])
-----------------------------------------------------------------------------
Init ≜
    ∧ InitInt
    ∧ InitCtx
    ∧ InitSerial
    ∧ Comm(Cop)!Init
    ∧ css = [r ∈ Replica ↦ EmptyGraph]
-----------------------------------------------------------------------------
Locate the node in rcss (the css at replica r ∈ Replica) that matches the context ctx of cop.
Locate(cop, rcss) ≜ CHOOSE n ∈ rcss.node : n = cop.ctx
xForm: Iteratively transform cop with a path through the css at replica r ∈ Replica, following
the first edges.
xForm(cop, r) ≜
    LET rcss ≜ css[r]
        u ≜ Locate(cop, rcss)
        v ≜ u ∪ {cop.oid}
        RECURSIVE xFormHelper(_, _, _, _, _)
         'h' stands for "helper"; xcss: eXtra css created during transformation
        xFormHelper(uh, vh, coph, xcss, xcoph) ≜
            IF uh = ds[r]
             THEN ⟨xcss, xcoph⟩
             ELSE LET fedge ≜ CHOOSE e ∈ rcss.edge :
                                  ∧ e.from = uh
```

```
                                        $\land \forall uhe \in rcss.edge :$
                                            $(uhe.from = uh \land uhe \neq e) \Rightarrow tb(e.cop.oid, uhe.cop.oid, serial[r])$
                                $uprime \triangleq fedge.to$
                                $fcop \triangleq fedge.cop$
                                $coph2fcop \triangleq COT(coph, fcop)$
                                $fcop2coph \triangleq COT(fcop, coph)$
                                 $vprime \triangleq vh \cup \{fcop.oid\}$
                            IN   $xFormHelper(uprime, vprime, coph2fcop,$
                                   $[xcss$ EXCEPT $!.node = @ \cup \{vprime\},$
                                     $!.edge = @ \cup \{[from \mapsto vh, to \mapsto vprime, cop \mapsto fcop2coph],$
                                                $[from \mapsto uprime, to \mapsto vprime, cop \mapsto coph2fcop]\}],$
                                     $coph2fcop)$
    IN   $xFormHelper(u, v, cop, [node \mapsto \{v\}, edge \mapsto \{[from \mapsto u, to \mapsto v, cop \mapsto cop]\}], cop)$

Perform cop at replica $r \in Replica$.
$Perform(cop, r) \triangleq$
    LET $xform \triangleq xForm(cop, r)$    $xform$: $\langle xcss, xcop \rangle$
        $xcss \triangleq xform[1]$
        $xcop \triangleq xform[2]$
    IN  $\land css' = [css$ EXCEPT $![r] = @ \oplus xcss]$
        $\land state' = [state$ EXCEPT $![r] = Apply(xcop.op, @)]$
--------------------------------------------------------------------------------
Client $c \in Client$ issues an operation $op$.
$DoOp(c, op) \triangleq$    $op$: the raw operation generated by the client $c \in Client$
        $\land$ LET $cop \triangleq [op \mapsto op, oid \mapsto [c \mapsto c, seq \mapsto cseq'[c]], ctx \mapsto ds[c]]$
            IN  $\land Perform(cop, c)$
                $\land UpdateDS(c, cop)$
                $\land Comm(Cop)!CSend(cop)$

$DoIns(c) \triangleq$
    $\exists ins \in \{op \in Ins : op.pos \in 1 \,..\, (Len(state[c]) + 1) \land op.ch \in chins \land op.pr = Priority[c]\} :$
        $\land DoOp(c, ins)$
        $\land chins' = chins \setminus \{ins.ch\}$    We assume that all inserted elements are unique.

$DoDel(c) \triangleq$
    $\exists del \in \{op \in Del : op.pos \in 1 \,..\, Len(state[c])\} :$
        $\land DoOp(c, del)$
        $\land$ UNCHANGED $chins$

$Do(c) \triangleq$
      $\land DoCtx(c)$
      $\land DoSerial(c)$
      $\land \lor DoIns(c)$
         $\lor DoDel(c)$
Client $c \in Client$ receives a message from the $Server$.
```

```
Rev(c) ≜
     ∧ Comm(Cop)!CRev(c)
     ∧ Perform(Head(cincoming[c]), c)
     ∧ RevSerial(c)
     ∧ RevCtx(c)
     ∧ UNCHANGED chins

The Server receives a message.
SRev ≜
    ∧ Comm(Cop)!SRev
    ∧ LET cop ≜ Head(sincoming)
      IN    ∧ Perform(cop, Server)
            ∧ Comm(Cop)!SSendSame(cop.oid.c, cop)   broadcast the original operation
    ∧ SRevSerial
    ∧ SRevCtx
    ∧ UNCHANGED chins

Next ≜
    ∨ ∃ c ∈ Client : Do(c) ∨ Rev(c)
    ∨ SRev
Fairness: There is no requirement that the clients ever generate operations.
Fairness ≜
    WF_vars(SRev ∨ ∃ c ∈ Client : Rev(c))

Spec ≜ Init ∧ □[Next]_vars   ∧ Fairness (We care more about safety.)

The compactness of CJupiter: the CSSes at all replicas are the same.
Compactness ≜
    Comm(Cop)!EmptyChannel ⇒ Cardinality(Range(css)) = 1

THEOREM Spec ⇒ Compactness
```

\* Modification History
\* Last modified *Wed Dec* 19 11:35:32 *CST* 2018 by *hengxin*
\* Created Sat *Sep* 01 11:08:00 *CST* 2018 by *hengxin*