```
┌──────────────────────── MODULE GraphStateSpace ────────────────────────┐
The graph representation of n-ary ordered state space and 2D state space used in CJupiter and
XJupiter, respectively.
EXTENDS JupiterCtx, GraphsUtil
├────────────────────────────────────────────────────────────────────────┤
IsSS(G) ≜   A state space is a digraph with labeled edges.
      ∧ IsGraph(G)  It is a digraph (represented by a record).
      ∧ G.node ⊆ (SUBSET Oid)  Each node represents a document state, i.e., a set of Oid.
      ∧ G.edge ⊆ [from : G.node, to : G.node, cop : Cop]  Each edge is labeled with a Cop operation.

EmptySS ≜ EmptyGraph
├────────────────────────────────────────────────────────────────────────┤
Locate(cop, ss) ≜   Locate the node in state space ss that matches the context of cop.
    CHOOSE n ∈ ss.node : n = cop.ctx

xForm(NextEdge(_, _, _), r, cop, ss) ≜   Transform cop with an operation sequence
    LET u ≜ Locate(cop, ss)              in state space ss at replica r.
        v ≜ u ∪ {cop.oid}
        RECURSIVE xFormHelper(_, _, _, _)
         xFormHelper(uh, vh, coph, xss) ≜   xss: eXtra ss created during transformation
            IF uh = ds[r]
             THEN [xcop ↦ coph,
                   xss ↦ xss,
                   lss  ↦ [node ↦ {vh},
                           edge ↦ {[from ↦ uh, to ↦ vh, cop ↦ coph]}]]
             ELSE LET e ≜ NextEdge(r, uh, ss)
                      copprime ≜ e.cop
                      uprime ≜ e.to
                      vprime ≜ vh ∪ {copprime.oid}
                       coph2copprime ≜ COT(coph, copprime)
                       copprime2coph ≜ COT(copprime, coph)
                  IN   xFormHelper(uprime, vprime, coph2copprime,
                         xss ⊕ [node ↦ {vprime},
                                edge ↦ {[from ↦ vh, to ↦ vprime,
                                         cop  ↦ copprime2coph],
                                        [from ↦ uprime, to ↦ vprime,
                                         cop  ↦ coph2copprime]}])
    IN   xFormHelper(u, v, cop, [node ↦ {v},
                                 edge ↦ {[from ↦ u, to ↦ v, cop ↦ cop]}])

xFormCopCops(cop, cops) ≜   Transform cop against cops (a sequence of Cop) on state space.
    LET RECURSIVE xFormCopCopsSSHelper(_, _, _)
          xFormCopCopsSSHelper(coph, copsh, xss) ≜
            LET u ≜ coph.ctx
                v ≜ u ∪ {coph.oid}
             uvSS ≜ [node ↦ {u, v}, edge ↦ {[from ↦ u, to ↦ v, cop ↦ coph]}]
```

```
                  IN  IF copsh = ⟨⟩ THEN [xcop ↦ coph, xss ↦ xss ⊕ uvSS, lss ↦ uvSS]
                      ELSE LET copprimeh ≜ Head(copsh)
                                  uprime ≜ u ∪ {copprimeh.oid}
                                  vprime ≜ u ∪ {coph.oid, copprimeh.oid}
                               coph2copprimeh ≜ COT(coph, copprimeh)
                               copprimeh2coph ≜ COT(copprimeh, coph)
                           IN  xFormCopCopsSSHelper(coph2copprimeh, Tail(copsh),
                                  xss ⊕ [node ↦ {u, v},
                                         edge ↦ {[from ↦ u, to ↦ v, cop ↦ coph],
                                                 [from ↦ u, to ↦ uprime, cop ↦ copprimeh],
                                                 [from ↦ v, to ↦ vprime, cop ↦ copprimeh2coph]}])
    IN  xFormCopCopsSSHelper(cop, cops, EmptySS)

xFormCopCopsShift(cop, cops, shift) ≜   shifting the first shift elements out of cops
    xFormCopCops(cop, SubSeq(cops, shift + 1, Len(cops)))
```

* Modification History

* Last modified Sat *Jan* 19 10:49:59 *CST* 2019 by *hengxin*

* Created *Wed Dec* 19 18:15:25 *CST* 2018 by *hengxin*