```
1  ---------- MODULE StateSpace ----------
   The graph representation of n-ary ordered state space and 2D state space used in CJupiter and
   XJupiter, respectively.
6  EXTENDS JupiterCtx, GraphsUtil
7  ----------
8  IsSS(G) ≜  A state space is a digraph with labeled edges.
9      ∧ IsGraph(G)  It is a digraph (represented by a record).
10     ∧ G.node ⊆ (SUBSET Oid)  Each node represents a document state, i.e., a set of Oid.
11     ∧ G.edge ⊆ [from : G.node, to : G.node, cop : Cop]  Each edge is labeled with a Cop operation.

13 EmptySS ≜ EmptyGraph
14 ----------
15 Locate(cop, ss) ≜  Locate the (unique) node in state space ss that matches the context of cop.
16     CHOOSE n ∈ ss.node : n = cop.ctx

18 RECURSIVE ExtractCopSeq(_, _, _, _)  Extract Cop sequences starting with u in ss at replica r.
19 ExtractCopSeq(NextEdge(_, _, _), r, u, ss) ≜
20     IF u = ds[r] THEN ⟨⟩
21      ELSE LET e ≜ NextEdge(r, u, ss)
22           IN  ⟨e.cop⟩ ∘ ExtractCopSeq(NextEdge, r, e.to, ss)

24 xFormCopCopsSS(cop, cops) ≜  Transform cop against cops (a sequence of Cop) on state space.
25     LET RECURSIVE xFormCopCopsSSHelper(_, _, _)  Return the extra state space.
26         xFormCopCopsSSHelper(coph, copsh, xss) ≜  xss: the eXtra state space
27             LET u ≜ coph.ctx
28                 v ≜ u ∪ {coph.oid}
29              uvSS ≜ [node ↦ {u, v}, edge ↦ {[from ↦ u, to ↦ v, cop ↦ coph]}]
30              IN  IF copsh = ⟨⟩ THEN [xcop ↦ coph, xss ↦ xss ⊕ uvSS, lss ↦ uvSS]
31                  ELSE LET copprimeh ≜ Head(copsh)
32                           uprime ≜ u ∪ {copprimeh.oid}
33                           vprime ≜ u ∪ {coph.oid, copprimeh.oid}
34                       coph2copprimeh ≜ COT(coph, copprimeh)
35                       copprimeh2coph ≜ COT(copprimeh, coph)
36                   IN  xFormCopCopsSSHelper(coph2copprimeh, Tail(copsh),
37                         xss ⊕ [node ↦ {u, v},
38                                edge ↦ {[from ↦ u, to ↦ v, cop ↦ coph],
39                                        [from ↦ u, to ↦ uprime, cop ↦ copprimeh],
40                                        [from ↦ v, to ↦ vprime, cop ↦ copprimeh2coph]}])
41     IN  xFormCopCopsSSHelper(cop, cops, EmptySS)

43 xFormSS(cop, copprime) ≜  Transform cop against copprime on state space.
44     LET u ≜ cop.ctx  Return the extra state space.
45         v ≜ u ∪ {cop.oid}
46         uprime ≜ u ∪ {copprime.oid}
47         vprime ≜ u ∪ {cop.oid, copprime.oid}
48         cop2copprime ≜ COT(cop, copprime)
```

```
        copprime2cop ≜ COT(copprime, cop)
    IN  [node ↦ {u, v, uprime, vprime},
         edge ↦ {[from ↦ u, to ↦ v, cop ↦ cop],
                 [from ↦ u, to ↦ uprime, cop ↦ copprime],
                 [from ↦ v, to ↦ vprime, cop ↦ copprime2cop],
                 [from ↦ uprime, to ↦ vprime, cop ↦ cop2copprime]}]
```

* Modification History
* Last modified *Tue Jan* 08 14:33:51 *CST* 2019 by *hengxin*
* Created *Wed Dec* 19 18:15:25 *CST* 2018 by *hengxin*