MODULE *GraphStateSpace*

The graph representation of n-ary ordered state space and $2D$ state space used in *CJupiter* and *XJupiter*, respectively.

EXTENDS *JupiterCtx*, *GraphUtils*

$IsSS(G) \triangleq$ A state space is a digraph with labeled edges.
$\quad \land\ IsGraph(G)$ It is a digraph (represented by a record).
$\quad \land\ G.node \subseteq (\text{SUBSET } Oid)$ Each node represents a document state, *i.e.*, a set of *Oid*.
$\quad \land\ G.edge \subseteq [from : G.node,\ to : G.node,\ cop : Cop]$ Labeled with a *Cop* operation.

$EmptySS \triangleq EmptyGraph$

$Locate(cop,\ ss) \triangleq$ Locate the node in state space ss that matches the context of cop.
$\quad \text{CHOOSE } n \in ss.node : n = cop.ctx$

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

$xFormCopCops(cop,\ cops) \triangleq$ Transform cop against cops on state space.
$\quad \text{LET RECURSIVE } xFormCopCopsSSHelper(\_,\ \_,\ \_)$
$\qquad xFormCopCopsSSHelper(coph,\ copsh,\ xss) \triangleq$
$\qquad\quad \text{LET } u \triangleq coph.ctx$
$\qquad\qquad\quad v \triangleq u \cup \{coph.oid\}$
$\qquad\qquad uvSS \triangleq [node \mapsto \{u,\ v\},$

```
                    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 *Tue Feb* 05 11:52:00 *CST* 2019 by anonymous
* Created *Wed Dec* 19 18:15:25 *CST* 2018 by anonymous