———————————————— MODULE $GraphStateSpace$ ————————————————

1

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 $IsSS(G) \triangleq$ A state space is a digraph with labeled edges.

9       $\wedge IsGraph(G)$ It is a digraph (represented by a record).

10      $\wedge G.node \subseteq (\text{SUBSET } Oid)$ Each node represents a document state, $i.e.$, a set of Oid.

11      $\wedge G.edge \subseteq [from : G.node, to : G.node, cop : Cop]$ Each edge is labeled with a Cop operation.

13 $EmptySS \triangleq EmptyGraph$

14 ├———┤

15 $Locate(cop, ss) \triangleq$ Locate the node in state space ss that matches the context of cop.

16   CHOOSE $n \in ss.node : n = cop.ctx$

18 $xForm(NextEdge(\_, \_, \_), r, cop, ss) \triangleq$ Transform cop with an operation sequence

19   LET $u \triangleq Locate(cop, ss)$ in state space ss at replica r.

20         $v \triangleq u \cup \{cop.oid\}$

21   RECURSIVE $xFormHelper(\_, \_, \_, \_)$

22         $xFormHelper(uh, vh, coph, xss) \triangleq$ xss: $eXtra$ ss created during transformation

23   IF $uh = ds[r]$

24   THEN $[xcop \mapsto coph,$

25                    $xss \mapsto xss,$

26                    $lss \mapsto [node \mapsto \{vh\},$

27                        $edge \mapsto \{[from \mapsto uh, to \mapsto vh, cop \mapsto coph]\}]]$

28   ELSE LET $e \triangleq NextEdge(r, uh, ss)$

29                    $copprime \triangleq e.cop$

30                    $uprime \triangleq e.to$

31                    $vprime \triangleq vh \cup \{copprime.oid\}$

32                    $coph2copprime \triangleq COT(coph, copprime)$

33                    $copprime2coph \triangleq COT(copprime, coph)$

34   IN $xFormHelper(uprime, vprime, coph2copprime,$

35                        $xss \oplus [node \mapsto \{vprime\},$

36                            $edge \mapsto \{[from \mapsto vh, to \mapsto vprime,$

37 $cop \mapsto copprime2coph],$

38                                   $[from \mapsto uprime, to \mapsto vprime,$

39 $cop \mapsto coph2copprime]\}])$

40   IN $xFormHelper(u, v, cop, [node \mapsto \{v\},$

41                        $edge \mapsto \{[from \mapsto u, to \mapsto v, cop \mapsto cop]\}])$

43 $xFormCopCops(cop, cops) \triangleq$ Transform cop against cops (a sequence of Cop) on state space.

44   LET RECURSIVE $xFormCopCopsSSHelper(\_, \_, \_)$

45         $xFormCopCopsSSHelper(coph, copsh, xss) \triangleq$

46   LET $u \triangleq coph.ctx$

47               $v \triangleq u \cup \{coph.oid\}$

48               $uvSS \triangleq [node \mapsto \{u, v\}, edge \mapsto \{[from \mapsto u, to \mapsto v, cop \mapsto coph]\}]$

```
49              IN    IF copsh = ⟨⟩ THEN [xcop ↦ coph, xss ↦ xss ⊕ uvSS, lss ↦ uvSS]
50                    ELSE LET copprimeh   ≜  Head(copsh)
51                             uprime      ≜  u ∪ {copprimeh.oid}
52                             vprime      ≜  u ∪ {coph.oid, copprimeh.oid}
53                             coph2copprimeh   ≜  COT(coph, copprimeh)
54                             copprimeh2coph   ≜  COT(copprimeh, coph)
55                         IN    xFormCopCopsSSHelper(coph2copprimeh, Tail(copsh),
56                                 xss ⊕ [node ↦ {u, v},
57                                        edge ↦ {[from ↦ u, to ↦ v, cop ↦ coph],
58                                                [from ↦ u, to ↦ uprime, cop ↦ copprimeh],
59                                                [from ↦ v, to ↦ vprime, cop ↦ copprimeh2coph]}])
60    IN    xFormCopCopsSSHelper(cop, cops, EmptySS)
61
```

\ * Modification History
\ * Last modified *Tue Jan* 15 20:46:21 *CST* 2019 by *hengxin*
\ * Created *Wed Dec* 19 18:15:25 *CST* 2018 by *hengxin*

2