```
- module or_set -
 2 EXTENDS Integers, Sequences, TLC
   Constants N, Data
 5 \ Procs \stackrel{\triangle}{=} 1 \dots N
    --algorithm or_set
 8
    variables
9
      ops = [j \in Procs \mapsto \langle \rangle]; to broadcast operations
10
    define atSource phases
12
      \bot Lookup(set, e) \stackrel{\triangle}{=} \exists s \in set : s.val = e
13
      \_Add(proc, e) \triangleq \{[key \mapsto ToString(proc) \circ ToString(e), val \mapsto e]\}
15
      \_Remove(set, e) \stackrel{\Delta}{=} \text{IF } \_Lookup(set, e) \text{ THEN}
17
        {CHOOSE s \in set : s.val = e}
18
       ELSE
19
        {}
20
    end define;
21
     send a operation to all
23
    macro Broadcast(o, s)begin
24
      if s \neq \{\} then
25
        ops := [j \in Procs \mapsto Append(ops[j], [op \mapsto o, set \mapsto s])];
26
      end if
27
    end macro;
28
     receive and process operations, one by one
30
    macro Update(s)begin
31
      if Len(ops[self]) > 0 then
32
33
        if Head(ops[self]).op = "A" then
          s := s \cup Head(ops[self]).set;
34
         elsif Head(ops[self]).op = "R" then
35
          s := s \setminus Head(ops[self]).set;
36
        end if;
37
        ops[self] := Tail(ops[self]); clear processed operation
38
      end if;
39
    end macro;
40
    process Set \in Procs
42
    variables
43
      set = \{\}; local set of pairs [key \mapsto "", val \mapsto ""]
44
    begin Main:
45
      while TRUE do
46
        Update(set);
47
        either Add:
48
           with var \in Data do select a random value to add
49
```

```
Broadcast("A", _Add(self, var));
50
            end with;
51
         or Remove:
52
            with var \in Data do select a random value to remove
53
              Broadcast("R", \_Remove(set, var));
54
            end with;
55
         end either;
56
       end while ;
57
    end process;
58
    end algorithm ;
      BEGIN TRANSLATION
62
    VARIABLES ops, pc
      define statement
65
    \_Lookup(set, e) \stackrel{\triangle}{=} \exists s \in set : s.val = e
    \_Add(proc, e) \triangleq \{ [key \mapsto ToString(proc) \circ ToString(e), val \mapsto e] \}
    \_Remove(set, e) \stackrel{\triangle}{=} \text{IF } \_Lookup(set, e) \text{ THEN}
       \{\text{CHOOSE } s \in set : s.val = e\}
71
72
     ELSE
      {}
73
    Variable set
    vars \stackrel{\Delta}{=} \langle ops, pc, set \rangle
    ProcSet \triangleq (Procs)
    Init \stackrel{\triangle}{=} Global variables
81
                \land ops = [j \in Procs \mapsto \langle \rangle]
82
                Process Set
83
                \land set = [self \in Procs \mapsto \{\}]
84
                \land pc = [self \in ProcSet \mapsto "Main"]
85
    Main(self) \stackrel{\triangle}{=} \wedge pc[self] = "Main"
87
                         \wedge IF Len(ops[self]) > 0
88
                                THEN \land IF Head(ops[self]).op = "A"
89
                                                THEN \land set' = [set \ EXCEPT \ ! [self] = set[self] \cup Head(ops[self]).set]
90
                                                ELSE \land IF Head(ops[self]).op = "R"
91
                                                                THEN \land set' = [set \ EXCEPT \ ! [self] = set[self] \setminus Head(ops[self])
92
                                                                ELSE \land TRUE
93
                                                                         \wedge set' = set
94
                                         \land ops' = [ops \ EXCEPT \ ![self] = Tail(ops[self])]
95
                                ELSE \land TRUE
96
                                         \land UNCHANGED \langle ops, set \rangle
97
                         \land \lor \land pc' = [pc \text{ EXCEPT } ! [self] = \text{``Add''}]
```

98

```
\lor \land pc' = [pc \text{ EXCEPT } ![self] = \text{``Remove''}]
99
      Add(self) \stackrel{\Delta}{=} \wedge pc[self] = \text{``Add''}
101
                         \wedge \exists var \in Data:
102
                              IF (\_Add(self, var)) \neq \{\}
103
                                   THEN \land ops' = [j \in Procs \mapsto Append(ops[j], [op \mapsto "A", set \mapsto (\_Add(self, var))])]
104
                                  ELSE \land TRUE
105
                                            \wedge ops' = ops
106
                         \land pc' = [pc \text{ EXCEPT } ![self] = \text{"Main"}]
107
                         \wedge set' = set
108
      Remove(self) \stackrel{\Delta}{=} \wedge pc[self] = "Remove"
110
                              \land \exists var \in Data:
111
                                   IF (\_Remove(set[self], var)) \neq \{\}
112
                                        THEN \land ops' = [j \in Procs \mapsto Append(ops[j], [op \mapsto "R", set \mapsto (\_Remove(set[self], op \mapsto "R")])
113
                                        ELSE \land TRUE
114
                                                \wedge ops' = ops
115
                              \land pc' = [pc \text{ EXCEPT } ! [self] = \text{"Main"}]
116
                              \wedge set' = set
117
      Set(self) \triangleq Main(self) \vee Add(self) \vee Remove(self)
     Next \triangleq (\exists self \in Procs : Set(self))
     Spec \stackrel{\triangle}{=} Init \wedge \Box [Next]_{vars}
123
       END TRANSLATION
125
       Eventual Convergence:
127
       Safety: i, j: C(xi) = C(xj) implies that the abstract states of i and j are equivalent.
128
       Liveness: i, j: f C(xi) implies that, eventually, f C(xj).
129
      Convergence \triangleq (\forall i, j \in Procs : set[i] = set[j])
130
132
      * Last modified Thu Dec 13 17:12:10 PST 2018 by ocosta
      \* Created Sat Dec~01~19:34:11~PST~2018 by ocosta
```