```
MODULE or_set
EXTENDS Integers, Sequences, TLC
CONSTANTS N, Data

Procs ≜ 1 .. N

--algorithm or_set
variables
  ops = [j ∈ Procs ↦ ⟨⟩] ;   to broadcast operations

define   atSource phases
  _Lookup(set, e) ≜ ∃ s ∈ set : s.val = e

  _Add(proc, e) ≜ {[key ↦ ToString(proc) ∘ ToString(e), val ↦ e]}

  _Remove(set, e) ≜ IF _Lookup(set, e) THEN
    {CHOOSE s ∈ set : s.val = e}
   ELSE
    {}
end define ;

 send a operation to all
macro Broadcast(o, s)begin
  if s ≠ {} then
    ops := [j ∈ Procs ↦ Append(ops[j], [op ↦ o, set ↦ s])] ;
  end if
end macro ;

 receive and process operations, one by one
macro Update(s)begin
  if Len(ops[self]) > 0 then
    if Head(ops[self]).op = "A" then
      s := s ∪ Head(ops[self]).set ;
     elsif Head(ops[self]).op = "R" then
      s := s \ Head(ops[self]).set ;
    end if ;
    ops[self] := Tail(ops[self]) ;   clear processed operation
  end if ;
end macro ;

process Set ∈ Procs
variables
  set = {} ;   local set of pairs [key ↦ "", val ↦ ""]
begin Main:
  while TRUE do
    Update(set) ;
    either Add:
      with var ∈ Data do   select a random value to add
```

```
50          Broadcast("A", _Add(self, var)) ;
51        end with ;
52      or Remove:
53        with var ∈ Data do   select a random value to remove
54          Broadcast("R", _Remove(set, var)) ;
55        end with ;
56      end either ;
57    end while ;
58  end process ;

60  end algorithm ;
62   BEGIN TRANSLATION
63  VARIABLES ops, pc

65   define statement
66  _Lookup(set, e) ≜ ∃ s ∈ set : s.val = e

68  _Add(proc, e) ≜ {[key ↦ ToString(proc) ∘ ToString(e), val ↦ e]}

70  _Remove(set, e) ≜ IF _Lookup(set, e) THEN
71    {CHOOSE s ∈ set : s.val = e}
72   ELSE
73    {}

75  VARIABLE set

77  vars ≜ ⟨ops, pc, set⟩

79  ProcSet ≜ (Procs)

81  Init ≜   Global variables
82          ∧ ops = [j ∈ Procs ↦ ⟨⟩]
83           Process Set
84          ∧ set = [self ∈ Procs ↦ {}]
85          ∧ pc = [self ∈ ProcSet ↦ "Main"]

87  Main(self) ≜  ∧ pc[self] = "Main"
88                ∧ IF Len(ops[self]) > 0
89                      THEN  ∧ IF Head(ops[self]).op = "A"
90                                  THEN  ∧ set′ = [set EXCEPT ![self] = set[self] ∪ Head(ops[self]).set]
91                                  ELSE  ∧ IF Head(ops[self]).op = "R"
92                                              THEN  ∧ set′ = [set EXCEPT ![self] = set[self] \ Head(ops[self])
93                                              ELSE  ∧ TRUE
94                                                    ∧ set′ = set
95                                  ∧ ops′ = [ops EXCEPT ![self]  = Tail(ops[self])]
96                      ELSE  ∧ TRUE
97                                  ∧ UNCHANGED ⟨ops, set⟩
98                ∧ ∨ ∧ pc′ = [pc EXCEPT ![self] = "Add"]
```

```
                    ∨ ∧ pc' = [pc EXCEPT ![self] = "Remove"]

Add(self) ≜ ∧ pc[self] = "Add"
            ∧ ∃ var ∈ Data :
                IF (_Add(self, var)) ≠ {}
                    THEN ∧ ops' = [j ∈ Procs ↦ Append(ops[j], [op ↦ "A", set ↦ (_Add(self, var))])]
                    ELSE ∧ TRUE
                         ∧ ops' = ops
            ∧ pc' = [pc EXCEPT ![self] = "Main"]
            ∧ set' = set

Remove(self) ≜ ∧ pc[self] = "Remove"
               ∧ ∃ var ∈ Data :
                   IF (_Remove(set[self], var)) ≠ {}
                       THEN ∧ ops' = [j ∈ Procs ↦ Append(ops[j], [op ↦ "R", set ↦ (_Remove(set[self],
                       ELSE ∧ TRUE
                            ∧ ops' = ops
               ∧ pc' = [pc EXCEPT ![self] = "Main"]
               ∧ set' = set

Set(self) ≜ Main(self) ∨ Add(self) ∨ Remove(self)

Next ≜ (∃ self ∈ Procs : Set(self))

Spec ≜ Init ∧ □[Next]_vars

END TRANSLATION

\* Modification History
\* Last modified Wed Dec 12 17:54:27 PST 2018 by ocosta
\* Created Sat Dec 01 19:34:11 PST 2018 by ocosta
```