

```

1  ┌────────────────────────── MODULE op_counter ───────────────────────────┐
2  EXTENDS Integers, Sequences, TLC
3  CONSTANTS N
4
5  Procs  $\triangleq$  1 .. N
6
7
8  --algorithm op_counter
9  variables
10   ops = [j ∈ Procs ↦ ⟨⟩];  to broadcast operations
11
12   send a operation to all
13  macro Broadcast(o) begin
14   ops := [j ∈ Procs ↦ Append(ops[j], o)];
15  end macro ;
16
17   receive and process operations, one by one
18  macro Update(v) begin
19   if Len(ops[self]) > 0 then
20     if Head(ops[self]) = "I" then
21       v := v + 1 ;
22     elsif Head(ops[self]) = "D" then
23       v := v + 1 ;
24     end if ;
25     ops[self] := Tail(ops[self]);  clear processed operation
26   end if ;
27  end macro ;
28
29  process Counter ∈ Procs
30  variables
31   count = 0 ;  local counter
32  begin Main:
33   while TRUE do
34     Update(count) ;
35     either Increment:
36       Broadcast("I") ;
37     or Decrement:
38       Broadcast("D") ;
39     end either ;
40   end while ;
41  end process ;
42
43  end algorithm ;
44
45  BEGIN TRANSLATION
46  VARIABLES ops, pc, count
47
48  vars  $\triangleq$  ⟨ops, pc, count⟩
49
50  ProcSet  $\triangleq$  (Procs)

```

```

52 Init  $\triangleq$  Global variables
53       $\wedge ops = [j \in Procs \mapsto \langle \rangle]$ 
54      Process Counter
55       $\wedge count = [self \in Procs \mapsto 0]$ 
56       $\wedge pc = [self \in ProcSet \mapsto \text{"Main"}]$ 

58 Main(self)  $\triangleq$   $\wedge pc[self] = \text{"Main"}$ 
59       $\wedge \text{IF } Len(ops[self]) > 0$ 
60      THEN  $\wedge \text{IF } Head(ops[self]) = \text{"I"}$ 
61          THEN  $\wedge count' = [count \text{ EXCEPT } ![self] = count[self] + 1]$ 
62          ELSE  $\wedge \text{IF } Head(ops[self]) = \text{"D"}$ 
63              THEN  $\wedge count' = [count \text{ EXCEPT } ![self] = count[self] + 1]$ 
64              ELSE  $\wedge \text{TRUE}$ 
65               $\wedge count' = count$ 
66           $\wedge ops' = [ops \text{ EXCEPT } ![self] = Tail(ops[self])]$ 
67          ELSE  $\wedge \text{TRUE}$ 
68           $\wedge \text{UNCHANGED } \langle ops, count \rangle$ 
69       $\wedge \vee \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"Increment"}]$ 
70       $\vee \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"Decrement"}]$ 

72 Increment(self)  $\triangleq$   $\wedge pc[self] = \text{"Increment"}$ 
73       $\wedge ops' = [j \in Procs \mapsto Append(ops[j], \text{"I"})]$ 
74       $\wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"Main"}]$ 
75       $\wedge count' = count$ 

77 Decrement(self)  $\triangleq$   $\wedge pc[self] = \text{"Decrement"}$ 
78       $\wedge ops' = [j \in Procs \mapsto Append(ops[j], \text{"D"})]$ 
79       $\wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"Main"}]$ 
80       $\wedge count' = count$ 

82 Counter(self)  $\triangleq$  Main(self)  $\vee$  Increment(self)  $\vee$  Decrement(self)

84 Next  $\triangleq$   $(\exists self \in Procs : Counter(self))$ 

86 Spec  $\triangleq$  Init  $\wedge \Box [Next]_{vars}$ 

88 END TRANSLATION

90 Eventual Convergence:
91 Safety:  $i, j : C(xi) = C(xj)$  implies that the abstract states of  $i$  and  $j$  are equivalent.
92 Liveness:  $i, j : \text{f } C(xi)$  implies that, eventually,  $\text{f } C(xj)$ .
93 Convergence  $\triangleq$   $(\forall i, j \in Procs : count[i] = count[j])$ 

95 
    \ * Modification History
    \ * Last modified Thu Dec 13 17:11:59 PST 2018 by ocosta
    \ * Created Sat Dec 01 16:58:15 PST 2018 by ocosta
  

```