```
---------------------------- MODULE EWD998 ----------------------------
TLA+ specification of an algorithm for distributed termination detection on a ring, due to
Shmuel Safra, published as EWD 998: Shmuel Safra's version of termination detection.
https://www.cs.utexas.edu/users/EWD/ewd09xx/EWD998.PDF

EXTENDS Integers, FiniteSets, Functions, SequencesExt, Randomization

CONSTANT
    @type: Int;
  N
ASSUME NAssumption ≜ N ∈ Nat \ {0}   At least one node.

Node  ≜ 0 .. N − 1
Color ≜ {"white", "black"}
Token ≜ [pos : Node, q : Int, color : Color]

VARIABLES
   @type: Int → Bool;
 active,       activation status of nodes
   @type: Int → Str;
 color,        color of nodes
   @type: Int → Int;
 counter,      nb of sent messages − nb of rcvd messages per node
   @type: Int → Int;
 pending,      nb of messages in transit to node
   @type: [ pos: Int, q: Int, color: Str ];
 token         token structure

vars ≜ ⟨active, color, counter, pending, token⟩

TypeOK ≜
  ∧ active ∈ [Node → BOOLEAN ]
  ∧ color ∈ [Node → Color]
  ∧ counter ∈ [Node → Int]
  ∧ pending ∈ [Node → Nat]
  ∧ token ∈ Token
------------------------------------------------------------------------

Init ≜
    EWD840 but nodes
  ∧ active ∈ [Node → BOOLEAN ]
  ∧ color ∈ [Node → Color]
   Rule 0
  ∧ counter = [i ∈ Node ↦ 0]  c properly initialized
  ∧ pending = [i ∈ Node ↦ 0]
  ∧ token ∈ [pos : {0}, q : {0}, color : {"black"}]

InitiateProbe ≜
```

Rules $1 + 5 + 6$

$\land$ $token.pos = 0$

$\land$ previous round not conclusive if:

$\quad\lor$ $token.color =$ "black"

$\quad\lor$ $color[0] =$ "black"

$\quad\lor$ $counter[0] + token.q > 0$

$\land$ $token' = [pos \mapsto N - 1,\ q \mapsto 0,\ color \mapsto \text{"white"}]$

$\land$ $color' = [color \text{ EXCEPT } ![0] = \text{"white"}]$

The state of the nodes remains unchanged by token-related actions.

$\land$ UNCHANGED $\langle active, counter, pending \rangle$

$PassToken(i) \triangleq$

Rules $2 + 4 + 7$

$\land$ $\neg active[i]$ If machine i is active, keep the token.

$\land$ $token.pos = i$

$\land$ $token' = [pos \mapsto token.pos - 1,$

$\qquad q \mapsto token.q + counter[i],$

$\qquad color \mapsto \text{IF } color[i] = \text{"black" THEN "black" ELSE } token.color]$

$\qquad color \mapsto color[i]\ ]$

$\land$ $color' = [color \text{ EXCEPT } ![i] = \text{"white"}]$

The state of the nodes remains unchanged by token-related actions.

$\land$ UNCHANGED $\langle active, counter, pending \rangle$

$System \triangleq \lor InitiateProbe$
$\qquad\qquad \lor \exists\, i \in Node \setminus \{0\} : PassToken(i)$

$SendMsg(i) \triangleq$

Only allowed to send msgs if node i is active.

$\land$ $active[i]$

Rule 0

$\land$ $counter' = [counter \text{ EXCEPT } ![i] = @ + 1]$

Non-deterministically choose a receiver node.

$\land$ $\exists\, j \in Node \setminus \{i\} : pending' = [pending \text{ EXCEPT } ![j] = @ + 1]$

Note that we don't blacken node i as in $EWD840$ if node i
sends a message to node j with $j > i$

$\land$ UNCHANGED $\langle active, color, token \rangle$

$RecvMsg(i) \triangleq$

$\land$ $pending[i] > 0$

$\land$ $pending' = [pending \text{ EXCEPT } ![i] = @ - 1]$

Rule 0

$\land$ $counter' = [counter \text{ EXCEPT } ![i] = @ - 1]$

Rule 3

$\land$ $color' = [color \text{ EXCEPT } ![i] = \text{"black"}]$

Receipt of a message activates i.

$\land\ active' = [active \text{ EXCEPT } ![i] = \text{TRUE}]$
$\land\ \text{UNCHANGED } \langle token \rangle$

$Deactivate(i) \triangleq$
$\quad \land\ active[i]$
$\quad \land\ active' = [active \text{ EXCEPT } ![i] = \text{FALSE}]$
$\quad \land\ \text{UNCHANGED } \langle color, counter, pending, token \rangle$

$Environment \triangleq \exists\, i \in Node : SendMsg(i) \lor RecvMsg(i) \lor Deactivate(i)$

$Next \triangleq$
$\quad System \lor Environment$

$Spec \triangleq Init \land \Box[Next]_{vars} \land \mathrm{WF}_{vars}(System)$

Bound the otherwise infinite state space that TLC has to check.

$StateConstraint \triangleq$
$\quad \land\ \forall\, i \in Node : counter[i] \leq 3 \land pending[i] \leq 3$
$\quad \land\ token.q \leq 9$

Main safety property: if there is a white token at node 0 and there are no in-flight messages then every node is inactive.

$terminationDetected \triangleq$
$\quad \land\ token.pos = 0$
$\quad \land\ token.color = \text{“white”}$
$\quad \land\ token.q + counter[0] = 0$
$\quad \land\ color[0] = \text{“white”}$
$\quad \land\ \neg active[0]$
$\quad \land\ pending[0] = 0$

Sum of the values $f[x]$, for $x \in S \subseteq \text{DOMAIN } f$.

$Sum(f, S) \triangleq FoldFunctionOnSet(+, 0, f, S)$

The number of messages on their way. “in-flight”

$B \triangleq Sum(pending, Node)$

The system has terminated if no node is active and there are no in-flight messages.

$Termination \triangleq$
$\quad \land\ \forall\, i \in Node : \neg active[i]$
$\quad \land\ B = 0$

$TerminationDetection \triangleq$
$\quad terminationDetected \Rightarrow Termination$

Interval of nodes between a and b: this is just $a \,..\, b$, but the following definition helps *Apalache* to construct a bounded set.

$$Rng(a, b) \triangleq \{i \in Node : a \leq i \land i \leq b\}$$

Safra's inductive invariant

$Inv \triangleq$
$\quad \land P0 :: B = Sum(counter, Node)$
(Ai: $t < i < N$: machine $nr.i$ is passive) $\land$
(Si: $t < i < N$: $ci.i$) $= q$
$\quad \land \lor P1 :: \land \forall i \in Rng(token.pos + 1, N - 1) : active[i] = \text{FALSE}$ machine $nr.i$ is passive
$\qquad \land \text{IF } token.pos = N - 1$
$\qquad\quad \text{THEN } token.q = 0$
$\qquad\quad \text{ELSE } token.q = Sum(counter, Rng(token.pos + 1, N - 1))$
(Si: $0 \leq i \leq t$: $c.i) + q > 0$.
$\quad \lor P2 :: Sum(counter, Rng(0, token.pos)) + token.q > 0$
Ei: $0 \leq i \leq t$: machine $nr.i$ is black.
$\quad \lor P3 :: \exists i \in Rng(0, token.pos) : color[i] = \text{"black"}$
The token is black.
$\quad \lor P4 :: token.color = \text{"black"}$

The inductive invariant combined with the type invariant

$TypedInv \triangleq$
$\quad \land TypeOK$
$\quad \land Inv$

Liveness property: termination is eventually detected.

$Liveness \triangleq$
$\quad Termination \leadsto terminationDetected$

The algorithm implements the specification of termination detection in a ring with asynchronous communication. The parameters of module *AsyncTerminationDetection* are instantiated by the symbols of the same name of the present module.

$TD \triangleq \text{INSTANCE } AsyncTerminationDetection$

$TDSpec \triangleq TD!Spec$

THEOREM $Spec \Rightarrow TDSpec$

Checked with *TLC* in 01/2021 with two cores on a fairly modern desktop and the given state constraint *StateConstraint* above:

N	Diameter	Distinct States	States	Time

3	60	1.3m	10.1m	42 s
4	105	219m	2.3b	50 m