MODULE *message_queue3*

```
EXTENDS TLC, Integers, Sequences
CONSTANTS MaxQueueSize

  --algorithm message_queue

variable queue = ⟨⟩ ;

define
    BoundedQueue ≜ Len(queue) ≤ MaxQueueSize
end define ;

macro add_to_queue(val)begin
    await Len(queue) < MaxQueueSize ;
    queue := Append(queue, val) ;
end macro ;

procedure add_to_queue(val = "")begin
    Add:
        await Len(queue) < MaxQueueSize ;
        queue := Append(queue, val) ;
        return ;
end procedure ;

process writer = "writer"
begin Write:
    while TRUE do
        call add_to_queue("msg")
    end while ;
end process ;

process reader ∈ {"r1", "r2"}
variables current_message = "none" ;
begin Read:
    while TRUE do
        await queue ≠ ⟨⟩ ;
        current_message := Head(queue) ;
        queue := Tail(queue) ;
        either
            skip ;
        or
            NotifyFailure:
                current_message := "none" ;
                call add_to_queue(self) ;
        end either ;
    end while ;
end process ;
```

end algorithm ;

```
 BEGIN TRANSLATION (chksum(pcal) = "cc8acc17" ∧ chksum(tla) = "887d62be")
VARIABLES queue, pc, stack

 define statement
BoundedQueue ≜ Len(queue) ≤ MaxQueueSize

VARIABLES val, current_message

vars ≜ ⟨queue, pc, stack, val, current_message⟩

ProcSet ≜ {"writer"} ∪ ({"r1", "r2"})

Init ≜  Global variables
        ∧ queue = ⟨⟩
         Procedure add_to_queue
        ∧ val = [self ∈ ProcSet ↦ ""]
         Process reader
        ∧ current_message = [self ∈ {"r1", "r2"} ↦ "none"]
        ∧ stack = [self ∈ ProcSet ↦ ⟨⟩]
        ∧ pc = [self ∈ ProcSet ↦ CASE self = "writer" → "Write"
                                    □ self ∈ {"r1", "r2"} → "Read"]

Add(self) ≜ ∧ pc[self] = "Add"
            ∧ Len(queue) < MaxQueueSize
            ∧ queue' = Append(queue, val[self])
            ∧ pc' = [pc EXCEPT ![self] = Head(stack[self]).pc]
            ∧ val' = [val EXCEPT ![self] = Head(stack[self]).val]
            ∧ stack' = [stack EXCEPT ![self] = Tail(stack[self])]
            ∧ UNCHANGED current_message

add_to_queue(self) ≜ Add(self)

Write ≜ ∧ pc["writer"] = "Write"
        ∧ ∧ stack' = [stack EXCEPT !["writer"] = ⟨[procedure ↦  "add_to_queue",
                                                  pc        ↦  "Write",
                                                  val       ↦  val["writer"]]⟩
                                                 ∘ stack["writer"]]
          ∧ val' = [val EXCEPT !["writer"] = "msg"]
        ∧ pc' = [pc EXCEPT !["writer"] = "Add"]
        ∧ UNCHANGED ⟨queue, current_message⟩

writer ≜ Write

Read(self) ≜ ∧ pc[self] = "Read"
             ∧ queue ≠ ⟨⟩
             ∧ current_message' = [current_message EXCEPT ![self] = Head(queue)]
             ∧ queue' = Tail(queue)
```

```
                  $\land \lor \land$ TRUE
                        $\land pc' = [pc$ EXCEPT $![self] =$ "Read"$]$
                     $\lor \land pc' = [pc$ EXCEPT $![self] =$ "NotifyFailure"$]$
                  $\land$ UNCHANGED $\langle stack, val \rangle$

$NotifyFailure(self) \triangleq$ $\land pc[self] =$ "NotifyFailure"
                          $\land current\_message' = [current\_message$ EXCEPT $![self] =$ "none"$]$
                          $\land \land stack' = [stack$ EXCEPT $![self] = \langle [procedure \mapsto$ "add_to_queue",
                                                                  $pc \mapsto$ "Read",
                                                                  $val \mapsto val[self]] \rangle$
                                                                  $\circ\ stack[self]]$
                             $\land val' = [val$ EXCEPT $![self] = self]$
                          $\land pc' = [pc$ EXCEPT $![self] =$ "Add"$]$
                          $\land queue' = queue$

$reader(self) \triangleq Read(self) \lor NotifyFailure(self)$

$Next \triangleq writer$
          $\lor (\exists self \in ProcSet : add\_to\_queue(self))$
          $\lor (\exists self \in \{$"r1", "r2"$\} : reader(self))$

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

 END TRANSLATION
```

* Modification History
* Last modified Sat *Aug* 27 21:45:47 *CST* 2022 by *wengjialin*
* Created Sat *Aug* 27 18:19:37 *CST* 2022 by *wengjialin*