$\overline{\qquad\qquad}$ MODULE *backpressure* $\overline{\qquad\qquad}$

EXTENDS *FiniteSets*, *Integers*, *Sequences*, *TLC*

$Null \triangleq 0$
$Cowns \triangleq 1 .. 4$
$BehaviourLimit \triangleq 4$
$OverloadThreshold \triangleq 2$
$PriorityLevels \triangleq \{-1, 0, 1\}$

$Min(s) \triangleq$ CHOOSE $x \in s : \forall y \in s \setminus \{x\} : y > x$
$Max(s) \triangleq$ CHOOSE $x \in s : \forall y \in s \setminus \{x\} : y < x$

$Range(f) \triangleq \{f[x] : x \in$ DOMAIN $f\}$

$Pick(s) \triangleq$ CHOOSE $x \in s :$ TRUE

$ReduceSet(op(\_, \_), set, acc) \triangleq$
  LET $f[s \in$ SUBSET $set] \triangleq$
    IF $s = \{\}$ THEN acc ELSE LET $x \triangleq Pick(s)$IN $op(x, f[s \setminus \{x\}])$
  IN $f[set]$

VARIABLES *fuel*, *queue*, *scheduled*, *running*, *priority*, *blocker*, *mutor*, *mute*
$vars \triangleq \langle fuel, queue, scheduled, running, priority, blocker, mutor, mute \rangle$

$Sleeping(c) \triangleq scheduled[c] \wedge (Len(queue[c]) = 0)$

$Available(c) \triangleq scheduled[c] \wedge (Len(queue[c]) > 0)$

$Overloaded(c) \triangleq Len(queue[c]) > OverloadThreshold$

$Muted(c) \triangleq c \in$ UNION $Range(mute)$

$CurrentMessage(c) \triangleq$ IF $Len(queue[c]) > 0$ THEN $Head(queue[c])$ ELSE $\{\}$

$LowPriority(cs) \triangleq \{c \in cs : priority[c] = -1\}$

$HighPriority(cs) \triangleq \{c \in cs : priority[c] = 1\}$

$RequiresPriority(c) \triangleq$
  $\vee Overloaded(c)$
  $\vee \exists m \in Range(queue[c]) : \exists k \in m \setminus \{c\} : priority[k] = 1$

RECURSIVE $Blockers(\_)$
$Blockers(c) \triangleq$
  IF $blocker[c] = Null$ THEN $\{\}$ ELSE $\{blocker[c]\} \cup Blockers(blocker[c])$
$Prioritizing(cs) \triangleq$
  LET $unprioritized \triangleq \{c \in cs : priority[c] < 1\}$IN
  $unprioritized \cup$ UNION $\{Blockers(c) : c \in unprioritized\}$

$ValidMutor(c) \triangleq$
  $\lor (priority[c] = 1) \land Overloaded(c)$
  $\lor (priority[c] = -1)$

$Init \triangleq$
  $\land fuel = BehaviourLimit$
  $\land queue = [c \in Cowns \mapsto \langle \{c\} \rangle]$
  $\land scheduled = [c \in Cowns \mapsto \text{TRUE}]$
  $\land running = [c \in Cowns \mapsto \text{FALSE}]$
  $\land priority = [c \in Cowns \mapsto Null]$
  $\land blocker = [c \in Cowns \mapsto Null]$
  $\land mutor = [c \in Cowns \mapsto Null]$
  $\land mute = [c \in Cowns \mapsto \{\}]$

$Terminating \triangleq$
  $\land \forall c \in Cowns : Sleeping(c)$
  $\land \text{UNCHANGED } vars$

$Acquire(cown) \triangleq$
 LET $msg \triangleq CurrentMessage(cown)$ IN
  $\land Available(cown)$
  $\land cown < Max(msg)$
  $\land \text{IF } \exists c \in msg : priority[c] = 1 \text{ THEN}$
    LET $prioritizing \triangleq Prioritizing(\{c \in msg : c > cown\})$ IN
    LET $unmuting \triangleq LowPriority(prioritizing)$ IN
      $\land priority' = [c \in prioritizing \mapsto 1] @@ priority$
      $\land scheduled' = (cown :> \text{FALSE}) @@ [c \in unmuting \mapsto \text{TRUE}] @@ scheduled$
    ELSE
      $\land scheduled' = (cown :> \text{FALSE}) @@ scheduled$
      $\land \text{UNCHANGED } \langle priority, mute \rangle$
  $\land \text{LET } next \triangleq Min(\{c \in msg : c > cown\})$ IN
    $\land blocker' = (cown :> next) @@ blocker$
    $\land \text{LET } q \triangleq (cown :> Tail(queue[cown])) @@ queue$ IN
    $queue' = (next :> Append(queue[next], msg)) @@ q$
  $\land \text{UNCHANGED } \langle fuel, running, mutor, mute \rangle$

$Prerun(cown) \triangleq$
 LET $msg \triangleq CurrentMessage(cown)$ IN
  $\land scheduled[cown]$
  $\land \neg running[cown]$
  $\land \text{IF } msg = \{\} \text{ THEN FALSE ELSE } cown = Max(msg)$
  $\land priority' = (cown :> \text{IF } RequiresPriority(cown) \text{ THEN } 1 \text{ ELSE } 0) @@ priority$
  $\land running' = (cown :> \text{TRUE}) @@ running$
  $\land blocker' = [c \in msg \mapsto Null] @@ blocker$
  $\land \text{UNCHANGED } \langle fuel, queue, scheduled, mutor, mute \rangle$

$Send(cown) \triangleq$
 LET $senders \triangleq CurrentMessage(cown)$IN
 $\land running[cown]$
 $\land fuel > 0$
 $\land \exists receivers \in$ SUBSET $Cowns :$
  $\land Cardinality(receivers) > 0$
  $\land queue' =$
  $(Min(receivers) :> Append(queue[Min(receivers)], receivers)) @@ queue$
  $\land$ IF $\exists c \in receivers : priority[c] = 1$ THEN
   LET $prioritizing \triangleq Prioritizing(receivers)$IN
   LET $unmuting \triangleq LowPriority(prioritizing)$IN
   $\land priority' = [c \in prioritizing \mapsto 1] @@ priority$
   $\land scheduled' = [c \in unmuting \mapsto$ TRUE$] @@ scheduled$
   $\land$ LET $mutors \triangleq \{c \in receivers \setminus senders : ValidMutor(c)\}$IN
    IF
     $\land mutors \neq \{\}$
     $\land mutor[cown] = Null$
     $\land \forall c \in senders : priority[c] = 0$
     $\land \forall c \in senders : c \notin receivers$ $TODO$: justify
    THEN
     $\land mutor' = (cown :> Min(mutors)) @@ mutor$
    ELSE
     $\land$ UNCHANGED $\langle mutor \rangle$
    ELSE
     $\land$ UNCHANGED $\langle scheduled, priority, mutor \rangle$
 $\land fuel' = fuel - 1$
 $\land$ UNCHANGED $\langle running, blocker, mute \rangle$

$Complete(cown) \triangleq$
 LET $msg \triangleq CurrentMessage(cown)$IN
 $\land running[cown]$
 $\land$ IF $mutor[cown] \neq Null$ THEN
  LET $muting \triangleq \{c \in msg : priority[c] = 0\}$IN
  $\land priority' = [c \in muting \mapsto -1] @@ priority$
  $\land mute' = (mutor[cown] :> mute[mutor[cown]] \cup muting) @@ mute$
  $\land scheduled' = [c \in msg \mapsto c \notin muting] @@ scheduled$
  ELSE
  $\land scheduled' = [c \in msg \mapsto$ TRUE$] @@ scheduled$
  $\land$ UNCHANGED $\langle priority, mute \rangle$
 $\land queue' = (cown :> Tail(queue[cown])) @@ queue$
 $\land running' = (cown :>$ FALSE$) @@ running$
 $\land mutor' = (cown :> Null) @@ mutor$
 $\land$ UNCHANGED $\langle fuel, blocker \rangle$

$Unmute \triangleq$

$\text{LET } invalid\_keys \triangleq \{c \in \text{DOMAIN } mute : (priority[c] = 0) \lor Sleeping(c)\}\text{IN}$
$\text{LET } unmuting \triangleq \text{UNION } Range([k \in invalid\_keys \mapsto LowPriority(mute[k])])\text{IN}$
$\quad \land unmuting \neq \{\}$
$\quad \land priority' = [c \in unmuting \mapsto 0] @@ priority$
$\quad \land mute' = [c \in invalid\_keys \mapsto \{\}] @@ mute$
$\quad \land scheduled' = [c \in unmuting \mapsto \text{TRUE}] @@ scheduled$
$\quad \land \text{UNCHANGED } \langle fuel, queue, running, blocker, mutor \rangle$

$Run(cown) \triangleq$
$\quad \lor Acquire(cown)$
$\quad \lor Prerun(cown)$
$\quad \lor Send(cown)$
$\quad \lor Complete(cown)$

$Next \triangleq Terminating \lor \exists c \in Cowns : Run(c) \lor Unmute$

$Spec \triangleq$
$\quad \land Init$
$\quad \land \Box[Next]_{vars}$
$\quad \land \forall c \in Cowns : \text{WF}_{vars}(Run(c))$
$\quad \land \text{WF}_{vars}(Unmute)$

$MessageLimit \triangleq$
$\quad \text{LET } msgs \triangleq ReduceSet(\text{LAMBDA } c, sum : sum + Len(queue[c]), Cowns, 0)\text{IN}$
$\quad msgs \leq (BehaviourLimit + Max(Cowns))$

$RunningIsScheduled \triangleq$
$\quad \forall c \in Cowns : running[c] \Rightarrow scheduled[c] \land (c = Max(CurrentMessage(c)))$

$CownNotMutedBySelf \triangleq \forall c \in Cowns : c \notin mute[c]$

$LowPriorityMuted \triangleq \forall c \in Cowns : (priority[c] = -1) \Rightarrow Muted(c)$

$WillScheduleCown \triangleq \exists c \in Cowns :$
$\quad \lor scheduled[c]$
$\quad \lor$
$\quad\quad \land priority[c] = -1$
$\quad\quad \land \exists k \in \text{DOMAIN } mute : (c \in mute[k]) \land (priority[k] = 0)$

$Nonblocking \triangleq$
$\quad \forall c \in Cowns : \forall m \in Range(queue[c]) :$
$\quad\quad \neg(\exists h \in HighPriority(m) : \exists l \in LowPriority(m) : (h < c) \land (l \leq c))$

$RunningNotBlocked \triangleq$
$\quad \forall c \in Cowns : running[c] \Rightarrow (\forall k \in CurrentMessage(c) : blocker[k] = Null)$

$Acquired(c) \triangleq \exists k \in Cowns : (k > c) \land (c \in \text{UNION } Range(queue[k]))$
$UnscheduledByMuteOrAcquire \triangleq$

$\forall\, c \in Cowns : \neg((priority[c] = -1) \vee Acquired(c)) \equiv scheduled[c]$

$BehaviourAcquisition \triangleq$
$\quad \forall\, c \in Cowns : \forall\, k \in \text{UNION } Range(queue[c]) : (k < c) \Rightarrow \neg scheduled[k]$

$SelfInCurrentMessage \triangleq$
$\quad \forall\, c \in Cowns : (Len(queue[c]) > 0) \Rightarrow (c \in CurrentMessage(c))$

$Termination \triangleq \Diamond\Box(\forall\, c \in Cowns : Sleeping(c))$

$SomeCownWillBeScheduled \triangleq \Box\Diamond(\exists\, c \in Cowns : scheduled[c])$