MODULE *backpressure*

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

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

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

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

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

$ReduceSet(op(\_, \_), set, acc) \triangleq$
  LET $f[s \in \text{SUBSET } set] \triangleq$
    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*
$vars \triangleq \langle fuel, queue, scheduled, running, priority, blocker, mutor, mute\rangle$

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

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

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

$Muted(c) \triangleq c \in \text{UNION } Range(mute)$

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

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

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

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

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
  $unprioritized \cup \text{UNION } \{Blockers(c) : c \in unprioritized\}$

$$
\begin{array}{l}
ValidMutor(c) \triangleq \\
\quad \lor (priority[c] = 1) \land Overloaded(c) \\
\quad \lor (priority[c] = -1)
\end{array}
$$

$$
\begin{array}{l}
Init \triangleq \\
\quad \land fuel = BehaviourLimit \\
\quad \land queue = [c \in Cowns \mapsto \langle \{c\} \rangle] \\
\quad \land scheduled = [c \in Cowns \mapsto \text{TRUE}] \\
\quad \land running = [c \in Cowns \mapsto \text{FALSE}] \\
\quad \land priority = [c \in Cowns \mapsto 0] \\
\quad \land blocker = [c \in Cowns \mapsto Null] \\
\quad \land mutor = [c \in Cowns \mapsto Null] \\
\quad \land mute = [c \in Cowns \mapsto \{\}]
\end{array}
$$

$$
\begin{array}{l}
Terminating \triangleq \\
\quad \land \forall\, c \in Cowns\colon Len(queue[c]) = 0 \\
\quad \land Assert(\forall\, c \in Cowns : Sleeping(c), \text{“}Termination\ with\ unscheduled\ cowns\text{”}) \\
\quad \land \forall\, c \in Cowns : Sleeping(c) \\
\quad \land \text{UNCHANGED } vars
\end{array}
$$

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

$$
\begin{array}{l}
Prerun(cown) \triangleq \\
\quad \text{LET } msg \triangleq CurrentMessage(cown) \text{IN} \\
\quad \land scheduled[cown] \\
\quad \land \neg running[cown] \\
\quad \land \text{IF } msg = \{\} \text{ THEN FALSE ELSE } cown = Max(msg) \\
\quad \land priority' = (cown :> \text{IF } RequiresPriority(cown) \text{ THEN } 1 \text{ ELSE } 0) \mathbin{@@} priority \\
\quad \land running' = (cown :> \text{TRUE}) \mathbin{@@} running \\
\quad \land blocker' = [c \in msg \mapsto Null] \mathbin{@@} blocker
\end{array}
$$

$\land$ 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 $priority[Min(receivers)] = 1$ THEN
LET $prioritizing \triangleq Prioritizing(\{Min(receivers)\})$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$]$ @@ $scheduled$
$\land\ priority' =$
$(cown :>$ IF $Len(queue[cown]) = 1$ THEN 0 ELSE $priority[cown])$ @@
$[c \in msg \setminus \{cown\} \mapsto$ IF $Len(queue[c]) = 0$ THEN 0 ELSE $priority[c]]$ @@
$priority$
$\land$ UNCHANGED $\langle mute\rangle$

$\land\ queue' = (cown :> Tail(queue[cown]))\ @@\ queue$
$\land\ running' = (cown :> \text{FALSE})\ @@\ running$
$\land\ mutor' = (cown :> Null)\ @@\ mutor$
$\land\ \text{UNCHANGED}\ \langle fuel,\ blocker\rangle$

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

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

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

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

Invariants

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

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

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

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

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

$Nonblocking\ \triangleq$

$\forall\, c \in \mathit{Cowns} : \forall\, m \in \mathit{Range}(\mathit{queue}[c]) :$
$\neg(\exists\, h \in \mathit{HighPriority}(m) : \exists\, l \in \mathit{LowPriority}(m) : (h < c) \land (l \leq c))$

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

$\mathit{Acquired}(c) \triangleq \exists\, k \in \mathit{Cowns} : (k > c) \land (c \in \text{UNION}\ \mathit{Range}(\mathit{queue}[k]))$
$\mathit{UnscheduledByMuteOrAcquire} \triangleq$
$\forall\, c \in \mathit{Cowns} : \neg((\mathit{priority}[c] = -1) \lor \mathit{Acquired}(c)) \equiv \mathit{scheduled}[c]$

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

$\mathit{AcquiredBy}(a,\, b) \triangleq (a < b) \land (a \in \text{UNION}\ \mathit{Range}(\mathit{queue}[b]))$
$\mathit{AcquiredOnce} \triangleq$
$\forall\, a \in \mathit{Cowns} : \forall\, b \in \mathit{Cowns} : \forall\, c \in \mathit{Cowns} :$
$(\mathit{AcquiredBy}(a,\, b) \land \mathit{AcquiredBy}(a,\, c)) \Rightarrow (b = c)$

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

$\mathit{HighPriorityInQueue} \triangleq$
$\forall\, c \in \mathit{Cowns} : (\mathit{priority}[c] = 1) \Rightarrow$
$\exists\, k \in \mathit{Cowns} : c \in \text{UNION}\ \mathit{Range}(\mathit{queue}[k])$

$\mathit{Required}(c) \triangleq \exists\, k \in \mathit{Cowns} : (k < c) \land (c \in \text{UNION}\ \mathit{Range}(\mathit{queue}[k]))$
$\mathit{SleepingIsNormalOrRequired} \triangleq$
$\forall\, c \in \mathit{Cowns} : \mathit{Sleeping}(c) \Rightarrow ((\mathit{priority}[c] = 0) \lor \mathit{Required}(c))$

$\mathit{FreeCandy} \triangleq \forall\, c \in \mathit{Cowns} : ((\mathit{priority}[c] = 1) \Rightarrow (\mathit{Len}(\mathit{queue}[c]) > 0 \lor \neg \mathit{scheduled}[c]))$

$\mathit{MuteSetsDisjoint} \triangleq$
$\forall\, c \in \mathit{Cowns} : \forall\, k \in \mathit{Cowns} :$
$((\mathit{mute}[c] \cap \mathit{mute}[k]) \neq \{\}) \Rightarrow (c = k)$

https://*github.com*/tlaplus/Examples/blob/master/specifications/*TransitiveClosure*/*TransitiveClosure.tla*#*L*114

$TC(R) \triangleq$
LET
$S \triangleq \{r[1] : r \in R\} \cup \{r[2] : r \in R\}$
RECURSIVE $TCR(\_)$
$TCR(T) \triangleq$
IF $T = \{\}$ THEN R
ELSE
LET
$r \triangleq$ CHOOSE $s \in T$: TRUE
$RR \triangleq TCR(T \setminus \{r\})$
IN
$RR \cup \{\langle s,\, t\rangle \in S \times S : \langle s,\, r\rangle \in RR \land \langle r,\, t\rangle \in RR\}$

IN
$TCR(S)$

$CylcicTransitiveClosure(R(\_, \_)) \triangleq$
LET $s \triangleq \{\langle a, b\rangle \in Cowns \times Cowns : R(a, b)\}$
IN $\exists\, c \in Cowns : \langle c, c\rangle \in TC(s)$

$MutedBy(a, b) \triangleq (a \in mute[b]) \land (priority[a] = -1)$
$AcyclicTCMute \triangleq \neg CylcicTransitiveClosure(MutedBy)$

$Obstructs(a, b) \triangleq$
$\lor\ AcquiredBy(a, b)$
$\lor\ (\neg Acquired(a) \land MutedBy(a, b))$

$Foo \triangleq \neg CylcicTransitiveClosure(Obstructs)$

Temporal Properties

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

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