

EXTENDS *FiniteSets, Integers, Sequences, TLC*

Null $\triangleq 0$

Cowns $\triangleq 1 \dots 4$

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 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 $\triangleq \langle \textit{fuel}, \textit{queue}, \textit{scheduled}, \textit{running}, \textit{priority}, \textit{blocker}, \textit{mutor}, \textit{mute} \rangle$

Sleeping(*c*) $\triangleq \textit{scheduled}[c] \wedge (\textit{Len}(\textit{queue}[c]) = 0)$

Available(*c*) $\triangleq \textit{scheduled}[c] \wedge (\textit{Len}(\textit{queue}[c]) > 0)$

Overloaded(*c*) $\triangleq \textit{Len}(\textit{queue}[c]) > \textit{OverloadThreshold}$

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

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

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

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

RequiresPriority(*c*) \triangleq

$\vee \textit{Overloaded}(c)$

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

RECURSIVE *Blockers*($_$)

Blockers(*c*) \triangleq

IF *blocker*[*c*] = *Null* THEN {} ELSE {*blocker*[*c*]} \cup *Blockers*(*blocker*[*c*])

Prioritizing(*cs*) \triangleq

LET *unprioritized* $\triangleq \{c \in cs : \textit{priority}[c] < 1\}$ IN

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

$$\begin{aligned}
& \text{ValidMutor}(c) \triangleq \\
& \quad \vee (\text{priority}[c] = 1) \wedge \text{Overloaded}(c) \\
& \quad \vee (\text{priority}[c] = -1) \\
& \text{Init} \triangleq \\
& \quad \wedge \text{fuel} = \text{BehaviourLimit} \\
& \quad \wedge \text{queue} = [c \in \text{Cowns} \mapsto \langle \{c\} \rangle] \\
& \quad \wedge \text{scheduled} = [c \in \text{Cowns} \mapsto \text{TRUE}] \\
& \quad \wedge \text{running} = [c \in \text{Cowns} \mapsto \text{FALSE}] \\
& \quad \wedge \text{priority} = [c \in \text{Cowns} \mapsto \text{Null}] \\
& \quad \wedge \text{blocker} = [c \in \text{Cowns} \mapsto \text{Null}] \\
& \quad \wedge \text{mutor} = [c \in \text{Cowns} \mapsto \text{Null}] \\
& \quad \wedge \text{mute} = [c \in \text{Cowns} \mapsto \{\}] \\
& \text{Terminating} \triangleq \\
& \quad \wedge \forall c \in \text{Cowns} : \text{Sleeping}(c) \\
& \quad \wedge \text{UNCHANGED vars} \\
& \text{Acquire}(\text{cown}) \triangleq \\
& \quad \text{LET } \text{msg} \triangleq \text{CurrentMessage}(\text{cown}) \text{ IN} \\
& \quad \wedge \text{Available}(\text{cown}) \\
& \quad \wedge \text{cown} < \text{Max}(\text{msg}) \\
& \quad \wedge \text{IF } \exists c \in \text{msg} : \text{priority}[c] = 1 \text{ THEN} \\
& \quad \quad \text{LET } \text{prioritizing} \triangleq \text{Prioritizing}(\{c \in \text{msg} : c > \text{cown}\}) \text{ IN} \\
& \quad \quad \text{LET } \text{unmuting} \triangleq \text{LowPriority}(\text{prioritizing}) \text{ IN} \\
& \quad \quad \wedge \text{priority}' = [c \in \text{prioritizing} \mapsto 1] @@ \text{priority} \\
& \quad \quad \wedge \text{scheduled}' = (\text{cown} :> \text{FALSE}) @@ [c \in \text{unmuting} \mapsto \text{TRUE}] @@ \text{scheduled} \\
& \quad \text{ELSE} \\
& \quad \quad \wedge \text{scheduled}' = (\text{cown} :> \text{FALSE}) @@ \text{scheduled} \\
& \quad \quad \wedge \text{UNCHANGED } \langle \text{priority}, \text{mute} \rangle \\
& \quad \wedge \text{LET } \text{next} \triangleq \text{Min}(\{c \in \text{msg} : c > \text{cown}\}) \text{ IN} \\
& \quad \quad \wedge \text{blocker}' = (\text{cown} :> \text{next}) @@ \text{blocker} \\
& \quad \quad \wedge \text{LET } q \triangleq (\text{cown} :> \text{Tail}(\text{queue}[\text{cown}])) @@ \text{queue} \text{ IN} \\
& \quad \quad \quad \text{queue}' = (\text{next} :> \text{Append}(\text{queue}[\text{next}], \text{msg})) @@ q \\
& \quad \wedge \text{UNCHANGED } \langle \text{fuel}, \text{running}, \text{mutor}, \text{mute} \rangle \\
& \text{Prerun}(\text{cown}) \triangleq \\
& \quad \text{LET } \text{msg} \triangleq \text{CurrentMessage}(\text{cown}) \text{ IN} \\
& \quad \wedge \text{scheduled}[\text{cown}] \\
& \quad \wedge \neg \text{running}[\text{cown}] \\
& \quad \wedge \text{IF } \text{msg} = \{\} \text{ THEN FALSE ELSE } \text{cown} = \text{Max}(\text{msg}) \\
& \quad \wedge \text{priority}' = (\text{cown} :> \text{IF } \text{RequiresPriority}(\text{cown}) \text{ THEN 1 ELSE 0}) @@ \text{priority} \\
& \quad \wedge \text{running}' = (\text{cown} :> \text{TRUE}) @@ \text{running} \\
& \quad \wedge \text{blocker}' = [c \in \text{msg} \mapsto \text{Null}] @@ \text{blocker} \\
& \quad \wedge \text{UNCHANGED } \langle \text{fuel}, \text{queue}, \text{scheduled}, \text{mutor}, \text{mute} \rangle
\end{aligned}$$

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

$$\begin{aligned}
& \text{LET } \textit{invalid_keys} \triangleq \{c \in \text{DOMAIN } \textit{mute} : (\textit{priority}[c] = 0) \vee \textit{Sleeping}(c)\} \text{IN} \\
& \text{LET } \textit{unmuting} \triangleq \text{UNION } \textit{Range}([k \in \textit{invalid_keys} \mapsto \textit{LowPriority}(\textit{mute}[k])]) \text{IN} \\
& \quad \wedge \textit{unmuting} \neq \{\} \\
& \quad \wedge \textit{priority}' = [c \in \textit{unmuting} \mapsto 0] @@ \textit{priority} \\
& \quad \wedge \textit{mute}' = [c \in \textit{invalid_keys} \mapsto \{\}] @@ \textit{mute} \\
& \quad \wedge \textit{scheduled}' = [c \in \textit{unmuting} \mapsto \text{TRUE}] @@ \textit{scheduled} \\
& \quad \wedge \text{UNCHANGED } \langle \textit{fuel}, \textit{queue}, \textit{running}, \textit{blocker}, \textit{mutor} \rangle \\
\\
& \textit{Run}(\textit{cown}) \triangleq \\
& \quad \vee \textit{Acquire}(\textit{cown}) \\
& \quad \vee \textit{Prerun}(\textit{cown}) \\
& \quad \vee \textit{Send}(\textit{cown}) \\
& \quad \vee \textit{Complete}(\textit{cown}) \\
\\
& \textit{Next} \triangleq \textit{Terminating} \vee \exists c \in \textit{Cowns} : \textit{Run}(c) \vee \textit{Unmute} \\
\\
& \textit{Spec} \triangleq \\
& \quad \wedge \textit{Init} \\
& \quad \wedge \Box[\textit{Next}]_{\textit{vars}} \\
& \quad \wedge \forall c \in \textit{Cowns} : \text{WF}_{\textit{vars}}(\textit{Run}(c)) \\
& \quad \wedge \text{WF}_{\textit{vars}}(\textit{Unmute}) \\
\\
& \textit{MessageLimit} \triangleq \\
& \quad \text{LET } \textit{msgs} \triangleq \textit{ReduceSet}(\text{LAMBDA } c, \textit{sum} : \textit{sum} + \textit{Len}(\textit{queue}[c]), \textit{Cowns}, 0) \text{IN} \\
& \quad \textit{msgs} \leq (\textit{BehaviourLimit} + \textit{Max}(\textit{Cowns})) \\
\\
& \textit{RunningIsScheduled} \triangleq \\
& \quad \forall c \in \textit{Cowns} : \textit{running}[c] \Rightarrow \textit{scheduled}[c] \wedge (c = \textit{Max}(\textit{CurrentMessage}(c))) \\
\\
& \textit{CownNotMutedBySelf} \triangleq \forall c \in \textit{Cowns} : c \notin \textit{mute}[c] \\
\\
& \textit{LowPriorityMuted} \triangleq \forall c \in \textit{Cowns} : (\textit{priority}[c] = -1) \Rightarrow \textit{Muted}(c) \\
\\
& \textit{WillScheduleCown} \triangleq \exists c \in \textit{Cowns} : \\
& \quad \vee \textit{scheduled}[c] \\
& \quad \vee \\
& \quad \wedge \textit{priority}[c] = -1 \\
& \quad \wedge \exists k \in \text{DOMAIN } \textit{mute} : (c \in \textit{mute}[k]) \wedge (\textit{priority}[k] = 0) \\
\\
& \textit{Nonblocking} \triangleq \\
& \quad \forall c \in \textit{Cowns} : \forall m \in \textit{Range}(\textit{queue}[c]) : \\
& \quad \quad \neg(\exists h \in \textit{HighPriority}(m) : \exists l \in \textit{LowPriority}(m) : (h < c) \wedge (l \leq c)) \\
\\
& \textit{RunningNotBlocked} \triangleq \\
& \quad \forall c \in \textit{Cowns} : \textit{running}[c] \Rightarrow (\forall k \in \textit{CurrentMessage}(c) : \textit{blocker}[k] = \textit{Null}) \\
\\
& \textit{Acquired}(c) \triangleq \exists k \in \textit{Cowns} : (k > c) \wedge (c \in \text{UNION } \textit{Range}(\textit{queue}[k])) \\
& \textit{UnscheduledByMuteOrAcquire} \triangleq
\end{aligned}$$

$$\begin{aligned}
& \forall c \in \text{Cowns} : \neg((\text{priority}[c] = -1) \vee \text{Acquired}(c)) \equiv \text{scheduled}[c] \\
& \text{BehaviourAcquisition} \triangleq \\
& \quad \forall c \in \text{Cowns} : \forall k \in \text{UNION Range}(\text{queue}[c]) : (k < c) \Rightarrow \neg \text{scheduled}[k] \\
& \text{AcquiredBy}(a, b) \triangleq (a < b) \wedge (a \in \text{UNION Range}(\text{queue}[b])) \\
& \text{AcquiredOnce} \triangleq \\
& \quad \forall a \in \text{Cowns} : \forall b \in \text{Cowns} : \forall c \in \text{Cowns} : \\
& \quad \quad (\text{AcquiredBy}(a, b) \wedge \text{AcquiredBy}(a, c)) \Rightarrow (b = c) \\
& \text{SelfInCurrentMessage} \triangleq \\
& \quad \forall c \in \text{Cowns} : (\text{Len}(\text{queue}[c]) > 0) \Rightarrow (c \in \text{CurrentMessage}(c)) \\
& \text{Termination} \triangleq \Diamond \Box (\forall c \in \text{Cowns} : \text{Sleeping}(c)) \\
& \text{SomeCownWillBeScheduled} \triangleq \Box \Diamond (\exists c \in \text{Cowns} : \text{scheduled}[c])
\end{aligned}$$
