MODULE *SimpleAllocator*

Specification of an allocator managing a set of resources: - *Clients* can request sets of resources whenever all their previous
requests have been satisfied.
- Requests can be partly fulfilled, and resources can be returned even before the full request has been satisfied. However, clients only have an obligation to return resources after they have obtained all resources they requested.

EXTENDS $FiniteSets$, TLC

CONSTANTS

$Clients$,	set of all clients
$Resources$	set of all resources

ASSUME

$IsFiniteSet(Resources)$

VARIABLES

$unsat$,	set of all outstanding requests per process
$alloc$	set of resources allocated to given process

$$\begin{aligned} &TypeInvariant \triangleq\\ &\quad \land\ unsat \in [Clients \rightarrow \text{SUBSET } Resources]\\ &\quad \land\ alloc \in [Clients \rightarrow \text{SUBSET } Resources] \end{aligned}$$

Resources are available iff they have not been allocated.

$$available \triangleq Resources \setminus (\text{UNION } \{alloc[c] : c \in Clients\})$$

Initially, no resources have been requested or allocated.

$$\begin{aligned} &Init \triangleq\\ &\quad \land\ unsat = [c \in Clients \mapsto \{\}]\\ &\quad \land\ alloc = [c \in Clients \mapsto \{\}] \end{aligned}$$

A client c may request a set of resources provided that all of its previous requests have been satisfied and that it doesn't hold any resources.

$$\begin{aligned} &Request(c, S) \triangleq\\ &\quad \land\ unsat[c] = \{\} \land alloc[c] = \{\}\\ &\quad \land\ S \neq \{\} \land unsat' = [unsat \text{ EXCEPT } ![c] = S]\\ &\quad \land\ \text{UNCHANGED } alloc \end{aligned}$$

Allocation of a set of available resources to a client that requested them (the entire request does not have to be filled).

$$\begin{aligned} &Allocate(c, S) \triangleq\\ &\quad \land\ S \neq \{\} \land S \subseteq available \cap unsat[c]\\ &\quad \land\ alloc' = [alloc \text{ EXCEPT } ![c] = @ \cup S]\\ &\quad \land\ unsat' = [unsat \text{ EXCEPT } ![c] = @ \setminus S] \end{aligned}$$

Client c returns a set of resources that it holds. It may do so even before its full request has been honored.

$$
\begin{aligned}
&\mathit{Return}(c, S) \triangleq \\
&\quad \land S \neq \{\} \land S \subseteq \mathit{alloc}[c] \\
&\quad \land \mathit{alloc}' = [\mathit{alloc} \text{ EXCEPT } ![c] = @ \setminus S] \\
&\quad \land \text{UNCHANGED } \mathit{unsat}
\end{aligned}
$$

The next-state relation.

$$
\begin{aligned}
&\mathit{Next} \triangleq \\
&\quad \exists\, c \in \mathit{Clients},\ S \in \text{SUBSET } \mathit{Resources} : \\
&\qquad \mathit{Request}(c, S) \lor \mathit{Allocate}(c, S) \lor \mathit{Return}(c, S)
\end{aligned}
$$

$$\mathit{vars} \triangleq \langle \mathit{unsat},\ \mathit{alloc} \rangle$$

The complete high-level specification.

$$
\begin{aligned}
&\mathit{SimpleAllocator} \triangleq \\
&\quad \land \mathit{Init} \land \Box[\mathit{Next}]_{\mathit{vars}} \\
&\quad \land \forall\, c \in \mathit{Clients} : \mathrm{WF}_{\mathit{vars}}(\mathit{Return}(c,\ \mathit{alloc}[c])) \\
&\quad \land \forall\, c \in \mathit{Clients} : \mathrm{SF}_{\mathit{vars}}(\exists\, S \in \text{SUBSET } \mathit{Resources} : \mathit{Allocate}(c, S))
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{ResourceMutex} \triangleq \\
&\quad \forall\, c1,\ c2 \in \mathit{Clients} : c1 \neq c2 \Rightarrow \mathit{alloc}[c1] \cap \mathit{alloc}[c2] = \{\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{ClientsWillReturn} \triangleq \\
&\quad \forall\, c \in \mathit{Clients} : \mathit{unsat}[c] = \{\} \leadsto \mathit{alloc}[c] = \{\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{ClientsWillObtain} \triangleq \\
&\quad \forall\, c \in \mathit{Clients},\ r \in \mathit{Resources} : r \in \mathit{unsat}[c] \leadsto r \in \mathit{alloc}[c]
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{InfOftenSatisfied} \triangleq \\
&\quad \forall\, c \in \mathit{Clients} : \Box\Diamond(\mathit{unsat}[c] = \{\})
\end{aligned}
$$

Used for symmetry reduction with *TLC*

$$\mathit{Symmetry} \triangleq \mathit{Permutations}(\mathit{Clients}) \cup \mathit{Permutations}(\mathit{Resources})$$

The following version states a weaker fairness requirement for the clients: resources need be returned only if the entire request has been satisfied.

$$
\begin{aligned}
&\mathit{SimpleAllocator2} \triangleq \\
&\quad \land \mathit{Init} \land \Box[\mathit{Next}]_{\mathit{vars}}
\end{aligned}
$$

$\land \forall c \in \mathit{Clients} : \mathrm{WF}_{vars}(\mathit{unsat}[c] = \{\} \land \mathit{Return}(c, \mathit{alloc}[c]))$
$\land \forall c \in \mathit{Clients} : \mathrm{SF}_{vars}(\exists S \in \textsc{subset}\ \mathit{Resources} : \mathit{Allocate}(c, S))$

$\textsc{theorem}\ \mathit{SimpleAllocator} \Rightarrow \Box \mathit{TypeInvariant}$
$\textsc{theorem}\ \mathit{SimpleAllocator} \Rightarrow \Box \mathit{ResourceMutex}$
$\textsc{theorem}\ \mathit{SimpleAllocator} \Rightarrow \mathit{ClientsWillReturn}$
$\textsc{theorem}\ \mathit{SimpleAllocator}2 \Rightarrow \mathit{ClientsWillReturn}$
$\textsc{theorem}\ \mathit{SimpleAllocator} \Rightarrow \mathit{ClientsWillObtain}$
$\textsc{theorem}\ \mathit{SimpleAllocator} \Rightarrow \mathit{InfOftenSatisfied}$
* The following do not hold: *
* $\textsc{theorem}\ \mathit{SimpleAllocator}2 \Rightarrow \mathit{ClientsWillObtain}$ *
* $\textsc{theorem}\ \mathit{SimpleAllocator}2 \Rightarrow \mathit{InfOftenSatisfied}$ *
