* Authors: *Santhosh Raju* < *santhosh@byisystems.com* >
*   Cherry *G. Mathew* < cherry@*byisystems.com* >
*   Fransisca *Andriani* < *sisca@byisystems.com* >
*

MODULE *VoucherLifeCycle*

This specification is of a Voucher and it's life cycle. This is based on the definiton of Vouchers in *RFC* 3506 with the tuple part decoupled.

Note: A new state called "phantom" was introduced to indicate the state of a voucher that is yet to be issued, once a voucher is issued it becomes a "valid" voucher. This is a one way transition and it cannot reversed.

CONSTANT V — The set of vouchers.

VARIABLE $vState$, — $vState[v]$ is the state of a voucher v.
$vlcState$ — The state of the voucher life cycle machine.
$vvlcState[v]$ is the state of the life cycle machine
for the voucher v.

$VTypeOK \triangleq$

The type-correctness invariant

$\land\ vState \in [V \rightarrow \{\text{“phantom”}, \text{“valid”}, \text{“redeemed”}, \text{“cancelled”}\}]$
$\land\ vlcState \in [V \rightarrow \{\text{“init”}, \text{“working”}, \text{“done”}\}]$

$VInit \triangleq$

The initial predicate.

$\land\ vState = [v \in V \mapsto \text{“phantom”}]$
$\land\ vlcState = [v \in V \mapsto \text{“init”}]$

We now define the actions that may be performed on the Vs, and then define the complete next-state action of the specification to be the disjunction of the possible V actions.

$Issue(v) \triangleq$
$\land\ vState[v] = \text{“phantom”}$
$\land\ vlcState[v] = \text{“init”}$
$\land\ vState' = [vState\ \text{EXCEPT}\ ![v] = \text{“valid”}]$
$\land\ vlcState' = [vlcState\ \text{EXCEPT}\ ![v] = \text{“working”}]$

$Transfer(v) \triangleq$
$\land\ vState[v] = \text{“valid”}$
$\land\ \text{UNCHANGED}\ \langle vState, vlcState \rangle$

$$\begin{array}{l}
\mathit{Redeem}(v) \triangleq \\
\quad \land\ \mathit{vState}[v] = \text{“valid”} \\
\quad \land\ \mathit{vlcState}[v] = \text{“working”} \\
\quad \land\ \mathit{vState}' = [\mathit{vState}\ \textsc{except}\ ![v] = \text{“redeemed”}] \\
\quad \land\ \mathit{vlcState}' = [\mathit{vlcState}\ \textsc{except}\ ![v] = \text{“done”}]
\end{array}$$

$$\begin{array}{l}
\mathit{Cancel}(v) \triangleq \\
\quad \land\ \mathit{vState}[v] = \text{“valid”} \\
\quad \land\ \mathit{vlcState}[v] = \text{“working”} \\
\quad \land\ \mathit{vState}' = [\mathit{vState}\ \textsc{except}\ ![v] = \text{“cancelled”}] \\
\quad \land\ \mathit{vlcState}' = [\mathit{vlcState}\ \textsc{except}\ ![v] = \text{“done”}]
\end{array}$$

$$\mathit{VNext} \triangleq \exists\, v \in V : \mathit{Issue}(v) \lor \mathit{Redeem}(v) \lor \mathit{Transfer}(v) \lor \mathit{Cancel}(v)$$

The next-state action.

$$\mathit{VConsistent} \triangleq$$

A state predicate asserting that a V started at a valid start state and has reached a valid final state at the end of the life cycle. V can be “valid” only when the state of the machine is “working”. It is an invariant of the specification.

$$\begin{array}{lll}
\land\ \forall\, v \in V : & \lor & \land\ \mathit{vlcState}[v] = \text{“done”} \\
 & & \land\ \mathit{vState}[v] \in \{\text{“redeemed”}, \text{“cancelled”}\} \\
 & \lor & \land\ \mathit{vlcState}[v] = \text{“init”} \\
 & & \land\ \mathit{vState}[v] = \text{“phantom”} \\
 & \lor & \land\ \mathit{vlcState}[v] = \text{“working”} \\
 & & \land\ \mathit{vState}[v] \in \{\text{“valid”}\}
\end{array}$$

$$\mathit{VSpec} \triangleq \mathit{VInit} \land \Box[\mathit{VNext}]_{\langle \mathit{vState},\, \mathit{vlcState} \rangle}$$

The complete specification of the protocol written as a temporal formula.

$$\textsc{theorem}\ \mathit{VSpec} \Rightarrow \Box(\mathit{VTypeOK} \land \mathit{VConsistent})$$

This theorem asserts the truth of the temporal formula whose meaning is that the state predicate $\mathit{VTypeOK} \land \mathit{VConsistent}$ is an invariant of the specification VSpec. Invariance of this conjunction is equivalent to invariance of both of the formulas $\mathit{VTypeOK}$ and $\mathit{VConsistent}$.

\* Modification History
\* Last modified *Tue Jun* 12 13:25:29 *IST* 2018 by Fox
\* Created *Fri Mar* 16 11:56:25 *SGT* 2018 by Fox