MODULE *LinearSnapshot*

This module is part of the *AfekSimplified* example in Section 6 of "Auxiliary Variables in TLA+". The other modules in that example are *Linearizability*, *NewLinearSnapshot*, *NewLinearSnapshotPS*, *AfekSimplified*, and *AfekSimplifiedH*.

In their 1993 *JACM* paper "Atomic Snapshots of Shared Memory", *Afek*, Attiya, *Dolev*, *Gafni*, *Merritt*, and *Shavit* defined what they called a single-writer atomic snapshot object. We call it simply a snapshot object. A snapshot object is a data object with two kinds of processes, readers and writers, and whose state is a memory consisting of an array of registers, one register for each writer. A writer i can execute a command that sets the value of register value i, returning a "null value"; a reader can execute a command that returns the current value of the object, leaving its state unchanged. They present an algorithm that implements a linearizable snapshot object. Here, we define precisely what a linearizable snapshot object is. We specify it as an instance of the specification of linearizability in module *Linearizability*, which should be read before this module.

We declare the constants *Readers*, *Writers*, and *RegVals* to be the sets of readers, writers, and register values, and we declare the initial value *InitVal* of the registers. Readers and writers should be thought of as roles rather than processes, since a physical process can act as both a reader and a writer. We assume that the sets *Readers* and Writers are disjoint.

$$\textsc{constants}\ \mathit{Readers},\ \mathit{Writers},\ \mathit{RegVals},\ \mathit{InitRegVal}$$

$$\begin{array}{ll}\textsc{assume} & \land\ \mathit{Readers} \cap \mathit{Writers} = \{\} \\ & \land\ \mathit{InitRegVal} \in \mathit{RegVals}\end{array}$$

We define *Procs* to be the value to be substituted for the constant parameter of that name in the *Linearizability* module.

$$\mathit{Procs} \triangleq \mathit{Readers} \cup \mathit{Writers}$$

MemVals is the set of memory values, and *InitMem* is its initial value. They are used to instantiate the *ObjValues* and *InitObj* constants of the *Linearizability* module.

$$\begin{array}{lcl}\mathit{MemVals} & \triangleq & [\mathit{Writers} \rightarrow \mathit{RegVals}] \\ \mathit{InitMem} & \triangleq & [i \in \mathit{Writers} \mapsto \mathit{InitRegVal}]\end{array}$$

NotMemVal is the value returned by a write operation, and *NotRegVal* is the command issued by a reader.

$$\begin{array}{lcl}\mathit{NotMemVal} & \triangleq & \textsc{choose}\ v : v \notin \mathit{MemVals} \\ \mathit{NotRegVal} & \triangleq & \textsc{choose}\ v : v \notin \mathit{RegVals}\end{array}$$

We now define the values to be instantiated for the remaining constant paramters of module *Linearizability*.

$$\begin{array}{l}\mathit{Commands}(i) \triangleq \textsc{if}\ i \in \mathit{Readers}\ \textsc{then}\ \{\mathit{NotMemVal}\} \\ \qquad\qquad\qquad\qquad\qquad\qquad\ \ \textsc{else}\ \ \mathit{RegVals}\end{array}$$

$$\begin{array}{l}\mathit{Outputs}(i) \triangleq \textsc{if}\ i \in \mathit{Readers}\ \textsc{then}\ \mathit{MemVals} \\ \qquad\qquad\qquad\qquad\qquad\quad\ \ \textsc{else}\ \ \{\mathit{NotRegVal}\}\end{array}$$

$$\mathit{InitOutput}(i) \triangleq \textsc{if}\ i \in \mathit{Readers}\ \textsc{then}\ \mathit{InitMem}\ \textsc{else}\ \ \mathit{NotRegVal}$$

$$\begin{array}{l}\mathit{Apply}(i,\ \mathit{cmd},\ \mathit{obj}) \triangleq \textsc{if}\ i \in \mathit{Readers} \\ \qquad\qquad\qquad\qquad\qquad\ \textsc{then}\ [\mathit{newState} \mapsto \mathit{obj},\ \mathit{output} \mapsto \mathit{obj}] \\ \qquad\qquad\qquad\qquad\qquad\ \textsc{else}\ \ [\mathit{newState} \mapsto [\mathit{obj}\ \textsc{except}\ ![i] = \mathit{cmd}],\end{array}$$

$$output \quad \mapsto NotRegVal]$$

We prefer to use a variable named *mem* to instantiate the variable object of module *Linearizability*.

VARIABLES $mem, interface, istate$

Remember that parameters of an instantiated module whose instantiated values are not given in the WITH statement are instantiated by identifiers with the same name in the current module.

INSTANCE $Linearizability$ WITH $ObjValues \leftarrow MemVals,\ InitObj \leftarrow InitMem,$
$object \leftarrow mem$

TLC does not automatically check assumptions in instantiated modules.

ASSUME $LinearAssumps$

\* Modification History
\* Last modified Sat *Oct* 22 01:36:13 *PDT* 2016 by *lamport*
\* Created *Tue Oct* 04 02:26:20 *PDT* 2016 by *lamport*