MODULE *Buffer*

$Id: *Buffer.tla* 260 2010-03-08 16:27:45Z charpov $

This module simulates a producer-consumer example as it could be written using *Java* threads. In particular, we want to demonstrate the risk of deadlock when producers and consumers wait on the same object.

EXTENDS *Naturals*, *Sequences*

CONSTANTS *Producers*, the (nonempty) set of producers
Consumers, the (nonempty) set of consumers
BufCapacity, the maximum number of messages in the bounded buffer
Data the set of values that can be produced and/or consumed

$$\begin{array}{lll}
\text{ASSUME} & \land\ Producers \neq \{\} & \text{at least one producer} \\
 & \land\ Consumers \neq \{\} & \text{at least one consumer} \\
 & \land\ Producers \cap Consumers = \{\} & \text{no thread is both consumer and producer} \\
 & \land\ BufCapacity > 0 & \text{buffer capacity is at least 1} \\
 & \land\ Data \neq \{\} & \text{the type of data is nonenpty}
\end{array}$$

VARIABLES *buffer*, the buffer, as a sequence of objects
waitSet the wait set, as a set of threads

$$Participants \triangleq Producers \cup Consumers$$

$$RunningThreads \triangleq Participants \setminus waitSet$$

$$\begin{array}{ll}
TypeInv \triangleq & \land\ buffer \in Seq(Data) \\
 & \land\ Len(buffer) \in 0\,..\,BufCapacity \\
 & \land\ waitSet \subseteq Participants
\end{array}$$

$$\begin{array}{ll}
Notify \triangleq & \text{IF } waitSet \neq \{\} \\
 & \quad \text{THEN } \exists\, x \in waitSet : waitSet' = waitSet \setminus \{x\} \\
 & \quad \text{ELSE UNCHANGED } waitSet
\end{array}$$

corresponds to method *notify*() in *Java*

$$NotifyAll \triangleq waitSet' = \{\}$$

corresponds to method *notifyAll*() in *Java*

$$Wait(t) \triangleq waitSet' = waitSet \cup \{t\}$$

corresponds to method *wait*() is *Java*

$$Init \triangleq buffer = \langle\rangle \land waitSet = \{\}$$

$$\begin{array}{ll}
Put(t, m) \triangleq & \text{IF } Len(buffer) < BufCapacity \\
 & \quad \text{THEN } \land\ buffer' = Append(buffer, m) \\
 & \quad \phantom{\text{THEN }} \land\ Notify \\
 & \quad \text{ELSE } \land\ Wait(t) \\
 & \quad \phantom{\text{ELSE }} \land\ \text{UNCHANGED } buffer
\end{array}$$

$$\begin{array}{ll}
Get(t) \triangleq & \text{IF } Len(buffer) > 0 \\
 & \quad \text{THEN } \land\ buffer' = Tail(buffer) \\
 & \quad \phantom{\text{THEN }} \land\ Notify \\
 & \quad \text{ELSE } \land\ Wait(t) \\
 & \quad \phantom{\text{ELSE }} \land\ \text{UNCHANGED } buffer
\end{array}$$

$$\begin{array}{ll}
Next \triangleq \exists\, t \in RunningThreads : & \lor\ t \in Producers \land \exists\, m \in Data : Put(t, m) \\
 & \lor\ t \in Consumers \land Get(t)
\end{array}$$

$$Prog \triangleq Init \land \Box[Next]_{\langle buffer,\, waitSet\rangle}$$

$$NoDeadlock \triangleq \Box(RunningThreads \neq \{\})$$

THEOREM $Prog \Rightarrow \Box\, TypeInv \land NoDeadlock$