

MODULE *IFifo*

This module defines *IFifo* to be the TLA+ formula representing the abstract program *IFifo* of Section 7.6.1 of the book “A Science of Concurrent Programs” by *Leslie Lamport*. It is a standard linearizable specification of a *Fifo* (first-in-first-out) queue with a set *EnQers* of processes that perform Enqueue operations and a set of *DeQers* processes that perform Dequeue operations. Linearizability means that each of these two operations are performed by a sequence of three atomic actions:

- A Begin action that modifies the interface variables and, for an Enqueue operation, provides the value to be enqueued.
- A Do action that changes only internal variables and performs the Enqueue or Dequeue operation to the queue, whose contents are described by an internal variable.
- An End action that modifies the interface variables and, for a Dequeue operation, returns the value dequeued.

This module is instantiated by the module *IPOFifo_pqs*.

EXTENDS *Integers*, *Sequences*, *TLC*

CONSTANTS	<i>EnQers</i> , <i>DeQers</i> ,	The sets of enqueueer and dequeuer processes.
	<i>Data</i> ,	The set of data items that can be enqueued
	<i>NoData</i> ,	An arbitrary value not an element of <i>Data</i> .
	<i>Done</i> , <i>Busy</i>	Two arbitrary values, signifying an operation's status.

ASSUME \wedge *Done* \notin *Data*
 \wedge *Busy* \notin *Data*
 \wedge *NoData* \notin *Data*

The variables:

- *enq* and *deq* are the interface variables that are changed by the Begin and End actions as follows:
 - deq*[*d*]: Set to *Busy* by *BeginDeq*(*d*) and to the dequeued value by *EndDeq*(*d*)
 - enq*[*e*]: Set by *BeginEnq*(*e*) to the value to be enqueued, and to *Done* by *EndEnq*(*d*)
- *queue* is the internal variable holding the contents of the queue.
- *enqInner* and *deqInner* are internal variables that hold the state of a process during an operation as follows:
 - deqInner*[*d*]: Set to *NoData* by *BeginDeq*(*d*) and to the dequeued value by *DoDeq*(*d*).
 - enqInner*[*e*]: Set by *BeginEnq*(*e*) to *Busy*, and by *DoEnq*(*e*) to *Done*.

VARIABLES *enq*, *deq*, *queue*, *enqInner*, *deqInner*
 vars \triangleq \langle *enq*, *deq*, *queue*, *enqInner*, *deqInner* \rangle

TypeOK \triangleq \wedge *enq* \in [*EnQers* \rightarrow *Data* \cup {*Done*}]
 \wedge *deq* \in [*DeQers* \rightarrow *Data* \cup {*Busy*}]
 \wedge *queue* \in Seq(*Data*)
 \wedge *enqInner* \in [*EnQers* \rightarrow {*Done*, *Busy*}]
 \wedge *deqInner* \in [*DeQers* \rightarrow *Data* \cup {*NoData*}]

$$\begin{aligned}
Init &\triangleq \wedge enq = [e \in EnQers \mapsto Done] \\
&\wedge deq \in [DeQers \rightarrow Data] \\
&\wedge queue = \langle \rangle \\
&\wedge enqInner = [e \in EnQers \mapsto Done] \\
&\wedge deqInner = deq \\
\\
BeginEnq(e) &\triangleq \wedge enq[e] = Done \\
&\wedge \exists D \in Data : enq' = [enq \text{ EXCEPT } ![e] = D] \\
&\wedge enqInner' = [enqInner \text{ EXCEPT } ![e] = Busy] \\
&\wedge UNCHANGED \langle deq, queue, deqInner \rangle \\
\\
DoEnq(e) &\triangleq \wedge enqInner[e] = Busy \\
&\wedge queue' = Append(queue, enq[e]) \\
&\wedge enqInner' = [enqInner \text{ EXCEPT } ![e] = Done] \\
&\wedge UNCHANGED \langle deq, enq, deqInner \rangle \\
\\
EndEnq(e) &\triangleq \wedge enq[e] \neq Done \\
&\wedge enqInner[e] = Done \\
&\wedge enq' = [enq \text{ EXCEPT } ![e] = Done] \\
&\wedge UNCHANGED \langle deq, queue, enqInner, deqInner \rangle \\
\\
BeginDeq(d) &\triangleq \wedge deq[d] \neq Busy \\
&\wedge deq' = [deq \text{ EXCEPT } ![d] = Busy] \\
&\wedge deqInner' = [deqInner \text{ EXCEPT } ![d] = NoData] \\
&\wedge UNCHANGED \langle enq, queue, enqInner \rangle \\
\\
DoDeq(d) &\triangleq \wedge deq[d] = Busy \\
&\wedge deqInner[d] = NoData \\
&\wedge queue \neq \langle \rangle \\
&\wedge deqInner' = [deqInner \text{ EXCEPT } ![d] = Head(queue)] \\
&\wedge queue' = Tail(queue) \\
&\wedge UNCHANGED \langle enq, deq, enqInner \rangle \\
\\
EndDeq(d) &\triangleq \wedge deq[d] = Busy \\
&\wedge deqInner[d] \neq NoData \\
&\wedge deq' = [deq \text{ EXCEPT } ![d] = deqInner[d]] \\
&\wedge UNCHANGED \langle enq, queue, enqInner, deqInner \rangle \\
\\
Next &\triangleq \vee \exists e \in EnQers : BeginEnq(e) \vee DoEnq(e) \vee EndEnq(e) \\
&\vee \exists d \in DeQers : BeginDeq(d) \vee DoDeq(d) \vee EndDeq(d) \\
\\
IFifo &\triangleq Init \wedge \Box [Next]_{vars} \\
\\
Liveness &\triangleq \wedge \forall e \in EnQers : WF_vars(DoEnq(e) \vee EndEnq(e)) \\
&\wedge \forall d \in DeQers : WF_vars(DoDeq(d) \vee EndDeq(d)) \\
\\
ILiveFifo &\triangleq IFifo \wedge Liveness
\end{aligned}$$

* Modification History
* Last modified Sat *Oct* 19 16:17:25 *CEST* 2024 by *lamport*
* Created *Thu Sep* 15 06:53:21 *PDT* 2022 by *lamport*