
MODULE *IPOFifo*

This module defines the abstract program *IPOFifo* described in Section 7.6.2 of the book “A Science of Concurrent Programs” by *Leslie Lamport*. It is instantiated by the module *IPOFifo_pqs*.

The constants are the same as for the abstract program *Fifo* of Section 7.6.1 except for *Ids*, which is an arbitrary set of identifiers. (The constant *NoData* is not needed.)

CONSTANTS *EnQers*, *DeQers*, *Data*, *Busy*, *Done*, *Ids*

ASSUME $\wedge \text{Done} \notin \text{Data}$
 $\wedge \text{Busy} \notin \text{Data}$

Datums $\triangleq \text{Data} \times \text{Ids}$

The set of all values that can be enqueued in *elts*.

NonElt $\triangleq \langle \rangle$

We could let *NonElt* be any value not in *Datums*; this is a simple choice.

THE VARIABLES: *enq*, *deq* : The same interface variables as *Fifo*.

elts : The set of elements currently enqueued.

before : The relation \prec on a set *elts* is defined the way mathematicians generally define a relation on a set: as a subset of *elts* \times *elts*, where $d \prec e$ is defined to mean $\langle d, e \rangle$ is in that set. The value of *before* is the set that defines the \prec relation on the set *elts*.

adding[*e*] : If enqueuer *e* is performing an Enqueue operation, then this is the element it has put in *elts*. Otherwise, it equals *NonElt*.

VARIABLES *enq*, *deq*, *elts*, *before*, *adding*

POv $\triangleq \langle \text{enq}, \text{deq}, \text{elts}, \text{before}, \text{adding} \rangle$

$e1 \prec e2 \triangleq \langle e1, e2 \rangle \in \text{before}$

POTypeOK $\triangleq \wedge \text{enq} \in [\text{EnQers} \rightarrow \text{Data} \cup \{\text{Done}\}]$
 $\wedge \text{deq} \in [\text{DeQers} \rightarrow \text{Data} \cup \{\text{Busy}\}]$
 $\wedge \text{elts} \in \text{SUBSET } \text{Datums}$
 $\wedge \text{before} \in \text{SUBSET } (\text{elts} \times \text{elts})$
 $\wedge \text{adding} \in [\text{EnQers} \rightarrow \text{Datums} \cup \{\text{NonElt}\}]$

POInit $\triangleq \wedge \text{enq} = [e \in \text{EnQers} \mapsto \text{Done}]$
 $\wedge \text{elts} = \{\}$
 $\wedge \text{deq} \in [\text{DeQers} \rightarrow \text{Data}]$
 $\wedge \text{before} = \{\}$
 $\wedge \text{adding} = [e \in \text{EnQers} \mapsto \text{NonElt}]$

beingAdded is essentially the set of elements being enqueued by an Enqueue operation that has not completed.

beingAdded $\triangleq \{\text{adding}[e] : e \in \text{EnQers}\} \setminus \{\text{NonElt}\}$

THE ACTIONS: We add the *PO* to the names of the Begin and End actions to avoid name clashes with the *Fifo* abstract program.

The *BeginPOEnq(e)* action for enqueueer *e* adds an element to *elts* and updates the \prec relation (by changing *before*). It enqueues an arbitrary data item *D*, giving it an *id* such that $\langle D, id \rangle$ is not in *elts* or in *beingAdded*. We can't have the action add to *elts* an element that equals *adding[e2]* for an enqueueer *e2* $\neq e$ even if that element has been removed from *elts* by a Dequeue operation, because *beingAdded* is used in updating the \prec relation, and that could cause incorrect \prec relations to be added. The action also sets *adding[e]*.

$$\begin{aligned}
\text{BeginPOEnq}(e) &\triangleq \\
&\wedge \text{enq}[e] = \text{Done} \\
&\wedge \exists D \in \text{Data} : \\
&\quad \exists id \in \{i \in \text{Ids} : \langle D, i \rangle \notin (\text{elts} \cup \text{beingAdded})\} : \\
&\quad \wedge \text{enq}' = [\text{enq} \text{ EXCEPT } ![e] = D] \\
&\quad \wedge \text{elts}' = \text{elts} \cup \{\langle D, id \rangle\} \\
&\quad \wedge \text{before}' = \text{before} \cup \{\langle el, \langle D, id \rangle \rangle : el \in (\text{elts} \setminus \text{beingAdded})\} \\
&\quad \wedge \text{adding}' = [\text{adding} \text{ EXCEPT } ![e] = \langle D, id \rangle] \\
&\wedge \text{deq}' = \text{deq}
\end{aligned}$$

The *EndPOEnq(e)* and *BeginNdDeq(d)* for an enqueueer *e* and a dequeuer *d* do the obvious things.

$$\begin{aligned}
\text{EndPOEnq}(e) &\triangleq \wedge \text{enq}[e] \neq \text{Done} \\
&\quad \wedge \text{enq}' = [\text{enq} \text{ EXCEPT } ![e] = \text{Done}] \\
&\quad \wedge \text{adding}' = [\text{adding} \text{ EXCEPT } ![e] = \text{NonElt}] \\
&\quad \wedge \text{UNCHANGED } \langle \text{deq}, \text{elts}, \text{before} \rangle \\
\\
\text{BeginPODeq}(d) &\triangleq \wedge \text{deq}[d] \neq \text{Busy} \\
&\quad \wedge \text{deq}' = [\text{deq} \text{ EXCEPT } ![d] = \text{Busy}] \\
&\quad \wedge \text{UNCHANGED } \langle \text{enq}, \text{elts}, \text{before}, \text{adding} \rangle
\end{aligned}$$

The *EndPODeq(d)* action removes from *elts* an arbitrary element *el* that is no preceded in the \prec relation by any other element in *elts*. Remember that *el[1]* is the data item contained in the element *el*.

$$\begin{aligned}
\text{EndPODeq}(d) &\triangleq \wedge \text{deq}[d] = \text{Busy} \\
&\quad \wedge \exists el \in \text{elts} : \\
&\quad \quad \wedge \forall el2 \in \text{elts} : \neg (el2 \prec el) \\
&\quad \quad \wedge \text{elts}' = \text{elts} \setminus \{el\} \\
&\quad \quad \wedge \text{deq}' = [\text{deq} \text{ EXCEPT } ![d] = el[1]] \\
&\quad \quad \wedge \text{before}' = \text{before} \cap (\text{elts}' \times \text{elts}') \\
&\quad \wedge \text{UNCHANGED } \langle \text{enq}, \text{adding} \rangle
\end{aligned}$$

$$\begin{aligned}
\text{PONext} &\triangleq \vee \exists e \in \text{EnQers} : \text{BeginPOEnq}(e) \vee \text{EndPOEnq}(e) \\
&\quad \vee \exists d \in \text{DeQers} : \text{BeginPODeq}(d) \vee \text{EndPODeq}(d)
\end{aligned}$$

$$\text{POFifo} \triangleq \text{POInit} \wedge \Box[\text{PONext}]_{POv}$$

\ * Modification History
 \ * Last modified Sat Oct 19 16:12:55 CEST 2024 by lamport
 \ * Created Thu Sep 15 07:19:08 PDT 2022 by lamport