MODULE *AtomicBroadcast*

This module is the abstraction for the Atomic Broadcast, a primitive for group communication. A process can broadcast a message to its local group, where all members will deliver in the same order.

We use a sequence to maintain the same order on all processes. New messages are added to the back and removed from the front. A group has its own order within, whereas there are no ordering requirements across groups.

LOCAL INSTANCE *Naturals*
LOCAL INSTANCE *Sequences*

Number of groups.
CONSTANT *NGROUPS*

Number of processes.
CONSTANT *NPROCESSES*

The sequences of initial messages.
CONSTANT *INITIAL_MESSAGES*

VARIABLES

    The underlying buffer that holds all the messages.
    AtomicBroadcastBuffer

Broadcast the message to the given group. We add the message at the back of every process' sequence within this group.

```
$ABroadcast(g, m) \triangleq$
    $\land AtomicBroadcastBuffer' = [$
        $AtomicBroadcastBuffer$ EXCEPT $![g] = [$
            $p \in$ DOMAIN $AtomicBroadcastBuffer[g] \mapsto$
                $Append(AtomicBroadcastBuffer[g][p], m)]]$
```

Deliver the message to the process in the specific group. If there is a message in the buffer, we pass it to the callback and consume it.

```
$ABDeliver(g, p, Fn(\_)) \triangleq$
    $\land Len(AtomicBroadcastBuffer[g][p]) > 0$
    $\land Fn(Head(AtomicBroadcastBuffer[g][p]))$
```

$$\begin{aligned}
&\land\ \mathit{AtomicBroadcastBuffer}' = [\\
&\quad \mathit{AtomicBroadcastBuffer}\ \textsc{except}\ ![g][p] =\\
&\qquad \mathit{Tail}(\mathit{AtomicBroadcastBuffer}[g][p])]
\end{aligned}$$

Initialize the algorithm with the configuration values. The processes within a group will have the same sequence of messages in the same order.

$$\begin{aligned}
&\mathit{Init}\ \triangleq\\
&\quad \land\ \mathit{AtomicBroadcastBuffer} = [\\
&\qquad g \in 1 \mathrel{..} \mathit{NGROUPS} \mapsto [\\
&\qquad\quad p \in 1 \mathrel{..} \mathit{NPROCESSES} \mapsto \mathit{INITIAL\_MESSAGES}[g]]]
\end{aligned}$$