MODULE *Validity*

EXTENDS *Naturals*, *FiniteSets*, *Commons*

CONSTANT *NPROCESSES*
CONSTANT *NMESSAGES*
CONSTANT *CONFLICTR*(_, _)

Since this algorithm is for failure-free environments, the set of all processes is the same as the correct ones.

LOCAL $Processes \triangleq \{i : i \in 1 \,..\, NPROCESSES\}$
LOCAL $ChooseProcess \triangleq$ CHOOSE $x \in Processes$: TRUE
LOCAL $Create(id) \triangleq [id \mapsto id,\ d \mapsto Processes,\ o \mapsto ChooseProcess]$
LOCAL $AllMessages \triangleq \{Create(id) : id \in 1 \,..\, NMESSAGES\}$

VARIABLES
K,
$Pending$,
$Delivering$,
$Delivered$,
$PreviousMsgs$,
$Votes$,
$QuasiReliableChannel$

Initialize the instance for the Generic Multicast 0. The *INITIAL_MESSAGES* is a set with *NMESSAGES*, unordered, a tuple with the starting state *S*0 and the message.

$Algorithm \triangleq$ INSTANCE $GenericMulticast0$ WITH
$INITIAL\_MESSAGES \leftarrow \{\langle \text{“S0”}, m\rangle : m \in AllMessages\}$

Weak fairness is necessary.

$Spec \triangleq Algorithm!SpecFair$

If a correct process GM-Cast a message m to $m.d$, then some process in $m.d$ eventually GM-Deliver m .

We verify that all messages on the messages that will be sent, then we verify that exists a process on the existent processes that did sent the message and eventually exists a process on $m.d$ *that delivers the message*.

$Validity \triangleq$
$\forall\, m \in AllMessages :$
$m.o \in Processes \rightsquigarrow \exists\, q \in m.d : Algorithm!WasDelivered(q,\ m)$