```
──────────────── MODULE Validity ────────────────
EXTENDS Naturals, FiniteSets, Commons

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

─────────────────────────────────────────────────

LOCAL Processes ≜ 1 . . NPROCESSES
LOCAL Groups ≜ 1 . . NGROUPS
LOCAL ProcessesInGroup ≜ [g ∈ Groups ↦ Processes]

LOCAL AllMessages ≜ CreateMessages(NMESSAGES, Groups, Processes)
LOCAL MessagesCombinations ≜ CreatePossibleMessages(AllMessages)
─────────────────────────────────────────────────

VARIABLES K, PreviousMsgs, Delivered, Votes, MemoryBuffer,
    QuasiReliableChannel, AtomicBroadcastBuffer
```

Initialize the instance for the Generic Multicast 1. The *INITIAL_MESSAGES* is a sequence, totally ordered within a group, wherein the elements are tuples with the message, state, and timestamp.

```
Algorithm ≜ INSTANCE GenericMulticast1 WITH
    INITIAL_MESSAGES ← [
        g ∈ Groups ↦
            TotallyOrdered(MessagesCombinations[(g%NMESSAGES) + 1])]
─────────────────────────────────────────────────
```

Weak fairness is necessary.

```
Spec ≜ 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 ≜
    ∀ m ∈ AllMessages :
        m.o[1] ∈ Groups ∧ m.o[2] ∈ Processes
            ⇝ ∃ g ∈ m.d :
                ∃ p ∈ ProcessesInGroup[g] : Algorithm! WasDelivered(g, p, m)
─────────────────────────────────────────────────
```