

# Lab Exercise: Action-Based Temporal Logics

Radu Mateescu  
Inria and LIG, Convecs project-team

January 14, 2022

## 1 Two-Place Buffer Specified in LNT

We consider a two-place buffer that accepts as input an infinite stream of bits along channel **PUT** and delivers it as output on channel **GET**. The buffer is modeled as the composition of two one-place cells according to the architecture shown in Figure 1.

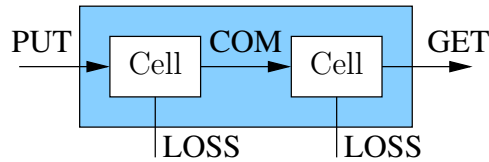


Figure 1: Architecture of the two-cell buffer

The behaviour of the two-place buffer is described in LNT as follows (file `buffer.lnt`). Each cell may receive a bit as input on channel **PUT**, store it, and then either deliver it as output on channel **GET**, or lose it by emitting a signal on channel **LOSS**, which denotes lossy communication. The buffer is composed of two cells synchronized on a channel **COM**, which is then hidden in order to keep visible only the **PUT** channel of the left cell and the **GET** channel of the right cell. This corresponds to the grey-colored box in Figure 1. The reliable version of the buffer is obtained by commenting out the lines denoting the choice of the **LOSS** action in process `Cell`.

```
module Buffer is

-- Data items to be stored in the buffer cells

type Bit is
  range 0 .. 1 of Nat
end type
```

```

-- Channels for synchronization and communication

channel Null_Channel is
  ()
end channel

channel Bit_Channel is
  (Bit)
end channel

-- Two-cell buffer

process Main [PUT:Bit_Channel, GET:Bit_Channel, LOSS:Null_Channel] is
  hide COM:Bit_Channel in
    par
      COM -> Cell [PUT, COM, LOSS]
      || COM -> Cell [COM, GET, LOSS]
    end par
  end hide
end process

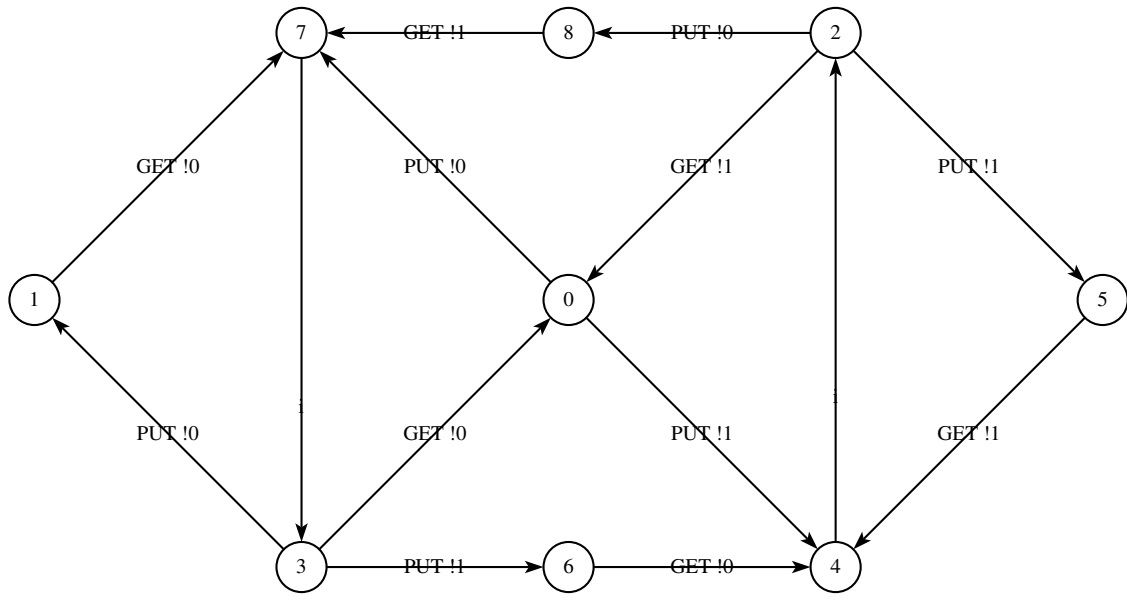
-- Buffer cell

process Cell [PUT:Bit_Channel, GET:Bit_Channel, LOSS:Null_Channel] is
  var n:Bit, empty:Bool in
    n := 0;
    empty := true;
    loop
      if empty then
        PUT (?n)
      else
        select
          LOSS      (* modeling unreliable communication *)
          []
          GET (n)
        end select
      end if;
      empty := not (empty)
    end loop
  end var
end process

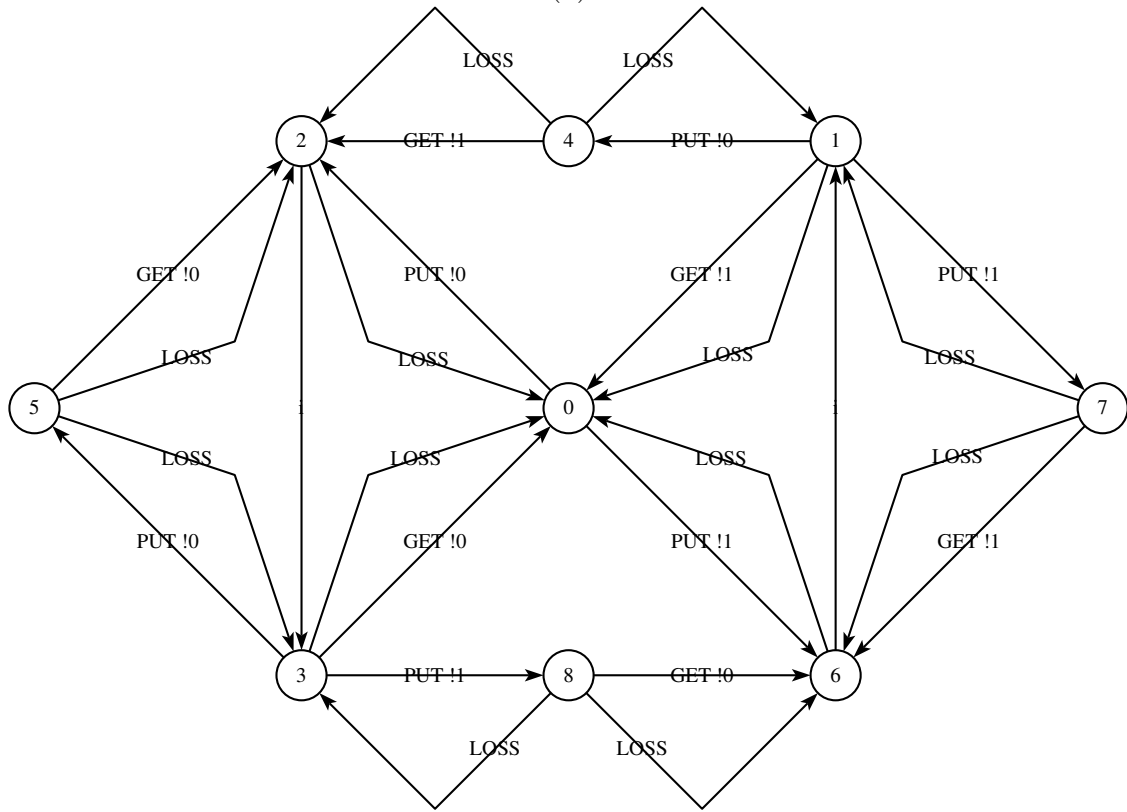
end module

```

The LTS models of the two-place buffer (reliable and lossy, available in files `buffer_2_reliable.bcg` and `buffer_2_lossy.bcg`) are shown in Figure 2.



(a)



(b)

Figure 2: LTS of the two-place buffer: (a) reliable, (b) lossy

In the following, we will specify several safety and liveness properties about the two-place buffer using MCL.

As an example, the reachability of two successive PUT actions carrying different values can be expressed as follows:

```
<
  true* .
  "PUT !0" .
  (not ("PUT !0" or "PUT !1"))* .
  "PUT !1"
> true
```

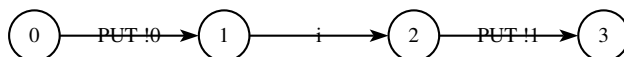
Every property must be checked on both LTSs using `evaluator3`, with the diagnostic generation option, which will help interpreting the verification verdicts. For example, assuming the property above is stored in a file `exists_put.mcl`, the command below checks it on the LTS of the reliable buffer:

```
bcg_open buffer_2_reliable.bcg evaluator3 -diag d_exists_put.bcg exists_put.mcl
```

The positive diagnostic (witness) produced in file `d_exists_put.bcg` can be visualized using the `bcg_edit` tool:

```
bcg_edit d_exists_put.bcg
```

This diagnostic is a sequence of three actions shown below:



You can also use the property patterns `INEVITABLE(A)`, `FAIR(A)`, and `CYCLE(R)` defined as macros in the file `macros.mcl`. This file must be included by adding a directive `library macros.mcl end_library` at the beginning of the MCL file containing the formula of interest.

## 2 Safety properties

Informally, safety properties express that “something bad never happens” during the execution of the system.

### 2.1 FIFO property

*Two different messages put one after the other in the buffer must be delivered in the same order.*

Specify first the property using action predicates containing only the channels PUT and GET. Does the property hold on the lossy buffer? Propose a modification of the property, by taking into account the actions on the LOSS channel, such that it yields the same verdict on both LTSs.

## 2.2 Causality of the first GET

*Initially, no GET action can occur before a PUT.*

## 2.3 Ordering of PUT

*Every time the buffer becomes full (i.e., after two successive PUT), it is impossible to reach a PUT action before a GET.*

Specify first the property using action predicates containing only the channels PUT and GET. Does the property hold on the lossy buffer? Propose a modification of the property, by taking into account the actions on the LOSS channel, such that it yields the same verdict on both LTSs.

## 2.4 Ordering of GET

*Every time the buffer becomes empty (i.e., after two successive GET), it is impossible to reach a GET action before a PUT.*

## 2.5 Combining safety properties

Write a single modality that combines the three safety properties above. Try to optimize it by factoring the prefixes and/or suffixes of regular formulas so as to minimize the number of \*-operators.

# 3 Liveness properties

Informally, liveness properties express that “something good eventually happens” during the execution of the system.

## 3.1 Inevitable transmission of messages

*Every message put into the buffer will be inevitably delivered.*

Specify the property using the INEVITABLE(A) property pattern and check it on both LTSs (modeling the reliable and lossy buffer). Does the property hold on the lossy buffer? Justify the verification verdict using the diagnostic produced by the model checker.

## 3.2 Fair transmission of messages

*Every message put into the buffer will be fairly delivered.*

Specify the property using the FAIR(A) property pattern and check it on both LTSs (modeling the reliable and lossy buffer). Does the property hold on the lossy buffer? Justify the verification verdict using the diagnostic produced by the model checker.

### 3.3 Reachability of full buffer

*There exists a sequence leading to a state where the buffer is full (i.e., a sequence containing two consecutive PUT actions without a GET in between).*

*Every time the buffer becomes full, it is possible to empty it.*

Specify these two properties first by using action predicates containing only the channels PUT and GET. When appropriate, justify the verification verdicts using the diagnostic produced by the model checker. Propose a modification of the second property (by taking into account the LOSS action) such that it yields the same verdict on both LTSs.

### 3.4 Normal execution of a FIFO cycle

*There exists an infinite execution during which two different consecutive messages are put into the buffer and then delivered in the same order.*

Specify the property and justify the verification verdicts using the diagnostic produced by the model checker.