Exercises on LNT

1 Exercise 1: Star Topology

The aim of this exercise is to specify a star network topology. The system consists of a set of nodes and of a router, which forwards the messages emitted by the nodes. All messages from node to node thus pass through the router.

Each node has an IP address represented abstractly by a natural number (1, 2, 3, ...). Processes need to handle IP address lists, represented by the type LIP defined below.

```
type LIP is list of Nat end type

function empty (lid : LIP) : Bool is
    case lid in
        nil -> return true
    |   cons (any Nat, any LIP) -> return false
    end case
end function
```

Write three functions head (which returns the first element of a given non-empty list), tail (which returns a given non-empty list without its first element), and member (which checks whether a given natural number belongs to a given list).

Messages contain two addresses, corresponding to the sender and receiver addresses. To simplify the model, we abstract out the remainder of the information exchanged between nodes. Define a channel Message, which allows messages of this form to be typed.

Now, write the two processes node and router, which are described as follows.

The node process is parameterized by its own address and by the list of addresses of the nodes to which it must send messages. The behaviour of process node is described as follows:

- If the address list is not empty, then the node may send on event send a message to the node whose address is at the head of the list. In this case, this address is removed from the list.
- At any time, the node may also receive on event receive a message sent to itself. In this case, the sender address is added at the head of the list, so that the current node sends the next message prioritarily to the node from which it received its last message.

The router process is parameterized by the list of addresses of the known nodes. The behaviour of the router process is described as follows: At any time, it can receive on event send a message sent by an arbitrary node. It then checks whether the addresses of the sender and of the receiver are known and, if so, forwards the message to the receiver on event receive. Otherwise, an invalid event occurs and the process gets ready to process the next message.

Below is an example of instantiation of such a system, in which three nodes interact. The notation $\{a_1, \ldots, a_n\}$ is a shorthand for `cons(`a_1`, cons(...,` `cons(`a_n`, nil)...))`.

```
par send, receive in
   par
      node [send, receive] (1, {})
   ||
      node [send, receive] (2, {1})
   ||
      node [send, receive] (3, {})
   end par
 ||
   router [send, receive, invalid] ({1, 2, 3})
end par
```

Draw the LTS corresponding to this concrete example.

2 Exercise 2: Digital Circuit

We propose to describe in LNT a digital circuit such as the one depicted in Figure 1, representing each gate of the circuit by an LNT process.

First, define the following four LNT processes:

- The process AND_gate is parameterized by three LNT events: Input_left, Input_Right, and Output. It receives a Boolean value on event Input_left and Input_Right, in any order, then sends on event Output the Boolean conjunction of the received values.
- The process OR_gate is defined similarly, but it sends the Boolean disjunction of the received values.
- The process NOT_gate is parameterized by two LNT events: Input and Output. It sends on event Output the Boolean negation of the Boolean value received on event Input.
- At last, the process Wire is similar to process NOT_gate, but it sends the received Boolean value instead of its negation.

Draw the LTS corresponding to each of these four processes.

We then define the LNT process Digital_Circuit corresponding to the example of Figure 1. This process corresponds to an instantiation of the different gate processes. It takes as parameters events Input_0 to Input_2 corresponding to its three inputs and events Output_0 to Output_3 corresponding to its four outputs. No hypothesis is made neither on the input reading order (which may thus occur in any order), nor on the output writing order. Internal LNT events will have to be defined to enable all data exchanges. Those events must not be part of the profile of process Digital_Circuit.

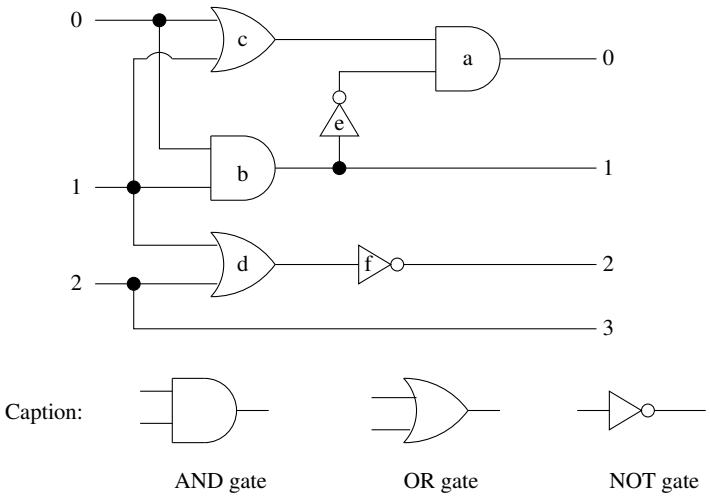

Figure 1: An example of circuit