Test 2.b: Communicating Processes (with abstracted counterexample)

1 Test instructions

This test consists in debugging a given system which does not respect a given safety property because of a bug in the specification. We ask you to discover the bug and describe its cause in an email that you will have to send us. We provide you the following information (in this pdf): (i) a short system description; (ii) the system specification written in LNT code; (iii) a short property description and the property in MCL code; (iv) a counterexample; (v) an abstracted counterexample.

The abstracted counterexample is a version of the counterexample where we highlighted choices in the model that can contribute to the cause of the bug. These choices correspond to states in the model where: (a) there is a transition leading to a correct behaviour (that always satisfies the property) (b) there is a transition leading to an incorrect behaviour (that always violates the property), (c) there is a transition leading to both correct and incorrect behaviour.

You do not need to use any software tools to perform this test, it should be sufficient to exploit the information contained in this document. You will have a maximum amount of time of 30 minutes to understand the specification and perform the debugging. At the end, you must write your results in an email. The email must contains the following fields:

- TEST NAME: the name of the test ('Test 2.b Communicating Processes' in this case);
- BUG DESCRIPTION: describe the cause of the bug in about two or three sentences (in English language); if you can point out a precise line in the LNT code that caused the bug, please indicate its number in the email; if you reach the time limit of 30 minutes without finding the bug, write "Bug not found";
- TIME: measure and provide the time (in minutes) you spent to discover the bug; you must start measuring time when you start reading Section 2 below; note that you must not take into account time spent in writing the email; if you reach 30 minutes without finding the bug, you can leave this field empty;

- ABSTRACTED COUNTEREXAMPLE: tell in one sentence if and why the abstracted counterexample helped you in understanding the bug.

You must send us the email at the following address:
gianluca.barbon AT inria DOT fr

2 Communicating Processes: system description

The system provided with this test case is composed of three main processes: a producer process, a consumer process and a process that can be both a consumer and a producer. Each process can loop infinitely or break the loop and terminate the execution. A deployer process is also present in order to deploy the three other processes. We present the LNT code of the specification in Appendix A.

3 The property

The provided property prevents a process to consume if something has not been produced before. Figure 1 shows the property in MCL. The MCL formula states that it should be impossible to have a consumption event (`CONSUME`) which has not been preceded by a production event (`PRODUCE`).

```
[  (not "PRODUCE")* . "CONSUME" .
   true*
] false
```

Figure 1: MCL property

The property in Figure 1 is not satisfied by the system. Figure 2 shows a counterexample generated with the Evaluator tool (from the CADP toolbox). Note that the code provided with this test case is syntactically correct and compiles. The bug arises from the violation of the given property.

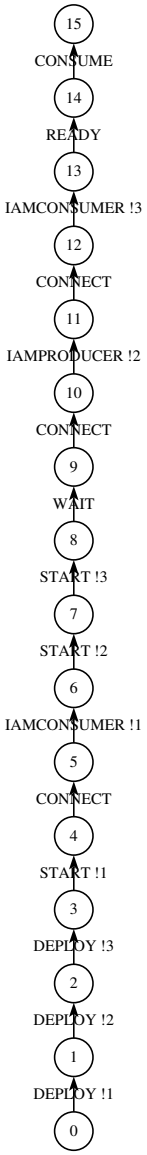

Figure 2: Counterexample.

4 Abstracted Counterexample

Figure 3 shows the abstraction of the counterexample depicted in Figure 2. The corresponding choices are shown on the right-hand side of the Figure.

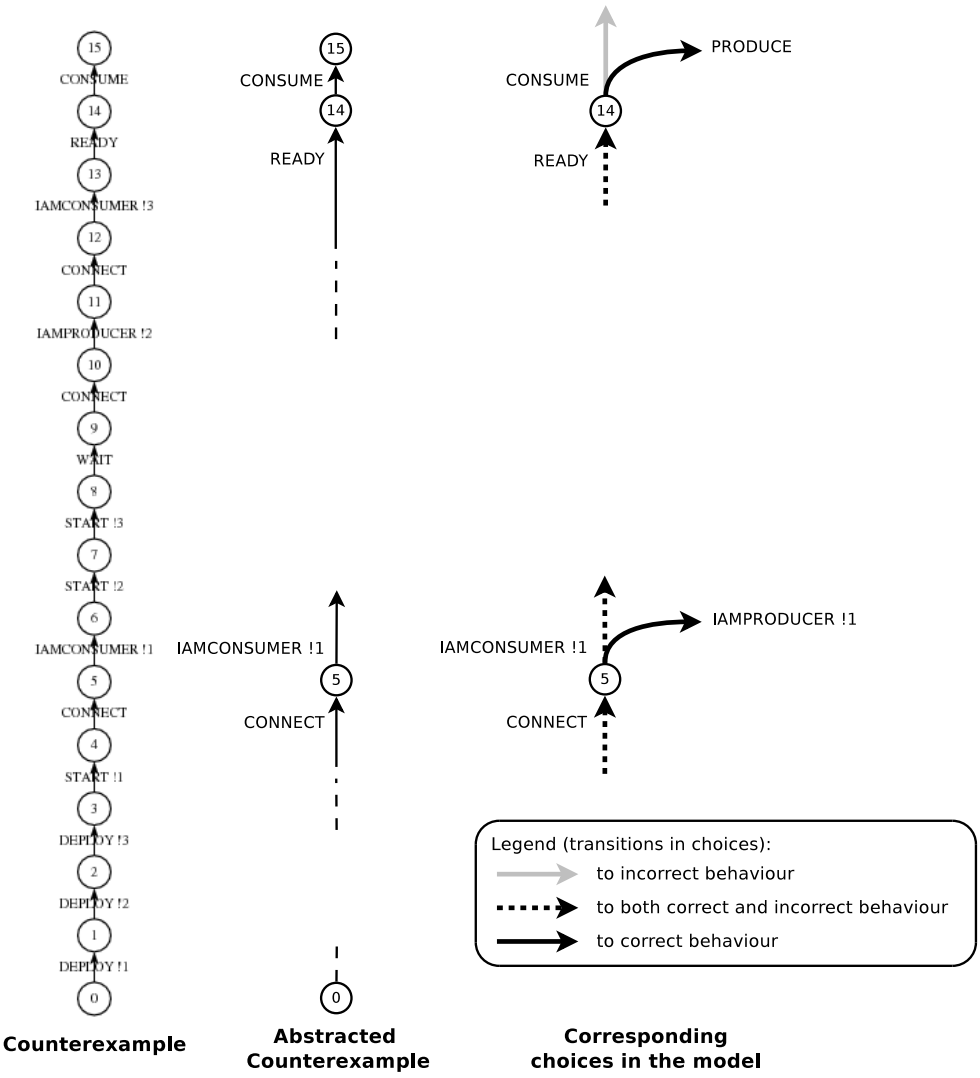

Figure 3: Abstracted counterexample and corresponding choices.

Appendix A LNT code

```
module COMMPROC is

-------------------------------------------------------------------------------

   channel WHOAMI_C is
      (nat)
   end channel

-------------------------------------------------------------------------------

   (*
    * DEPLOYER deploys other processes.
    *)
   process DEPLOYER [ DEPLOY : WHOAMI_C
                    ] is
      par
         DEPLOY(1 of nat) || DEPLOY(2 of nat) || DEPLOY(3 of nat)
      end par
   end process

   (*
    * PRODCONS can be a producer or a consumer.
    *)
   process PRODCONS [ CONNECT, READY, SYNC, WAIT : any,
                      DEPLOY, START, IAMPRODUCER, IAMCONSUMER : WHOAMI_C,
                      CONSUME, PRODUCE : any
                    ] is
      var whoami : bool in
         DEPLOY(1 of nat);
         START(1 of nat);
         CONNECT;
         select
            whoami := true; IAMPRODUCER(1 of nat)
         []
            whoami := false; IAMCONSUMER(1 of nat)
         end select;
         WAIT;
         READY;
         if (whoami) then
           loop L in
               select
                  NULL [] break L
               end select;
               par
                  WAIT || PRODUCE; SYNC
               end par
           end loop
         else
            loop L in
               select
                  NULL [] SYNC [] break L
               end select;
               par
                  WAIT || CONSUME
               end par
            end loop
         end if
      end var
   end process

   process PRODUCER [ CONNECT, READY, SYNC, WAIT : any,
                      DEPLOY, START, IAMPRODUCER : WHOAMI_C,
                      PRODUCE : any
                    ] is
      DEPLOY(2 of nat);
```

```
66          START(2 of nat);
67          CONNECT;
68          IAMPRODUCER(2 of nat);
69          READY;
70          loop L in
71             select
72                WAIT [] PRODUCE; SYNC [] break L
73             end select
74          end loop
75       end process
76
77       process CONSUMER [ CONNECT, READY, SYNC, WAIT : any,
78                          DEPLOY, START, IAMCONSUMER : WHOAMI_C,
79                          CONSUME : any
80                        ] is
81          DEPLOY(3 of nat);
82          START(3 of nat);
83          CONNECT;
84          IAMCONSUMER(3 of nat);
85          READY;
86          WAIT;
87          loop L in
88             select
89                WAIT [] SYNC; CONSUME [] break L
90             end select
91          end loop
92       end process
93
94       process MAIN [ CONNECT, READY, SYNC, WAIT : any,
95                      DEPLOY, START, IAMPRODUCER, IAMCONSUMER : WHOAMI_C,
96                      CONSUME, PRODUCE : any
97                    ] is
98          par DEPLOY in
99             DEPLOYER[DEPLOY]
100         ||
101            par READY, SYNC in
102               PRODCONS[CONNECT, READY, SYNC, WAIT, DEPLOY, START, IAMPRODUCER, IAMCONSUMER,
103                        CONSUME, PRODUCE]
104            ||
105               PRODUCER[CONNECT, READY, SYNC, WAIT, DEPLOY, START, IAMPRODUCER, PRODUCE]
106            ||
107               CONSUMER[CONNECT, READY, SYNC, WAIT, DEPLOY, START, IAMCONSUMER, CONSUME]
108            end par
109         end par
110      end process
111
112   ------------------------------------------------------------------------------
113
114   end module
```