System Design: Concurrency, Real-Time, Stochastics, and Analog/Digital
Master of Science in Informatics at Grenoble – December 2021

Homework Project #1

1. Motivation

The project is about "upward" translation from an old, lower-level language (namely, LOTOS) to a recent, higher-level language (namely, LNT). This project expects that students will:

- discover specifications of realistic, non-trivial concurrent systems;
- become familiar with reading description of concurrent systems written in "classical" process calculi (e.g., LOTOS, which is itself derived from Hoare's CSP and Milner's CCS);
- understand the differences and similarities between "classical" process calculi and modern specification languages such as LNT;
- generate Labelled Transition Systems (LTSs) and experiment with bisimulations and temporal-logic formulas.

2. Documentation

You can find detailed information about LOTOS here: https://cadp.inria.fr/tutorial/#lotos

You can find detailed information about LNT here: https://cadp.inria.fr/tutorial/#lnt

Details about the translation from LOTOS to LNT can be found in the attached slide show "translation.pdf" (39 pages).

Notice that a "downward" translation from LNT to LOTOS is done automatically by the **lnt2lotos** translator of the CADP toolbox. However, the reverse "upward" translation is difficult to automate, as it requires human decisions, common sense, and good taste to produce readable higher-language descriptions.

Students can find the CADP toolbox in the Virtual Machine prepared by Wendelin Serwe for MOSIG students. The toolbox is also installed at ENSIMAG, on all servers of room E103 (from ensipc501 to ensipc540, but ensipc539). Finally, students can obtain and install the CADP toolbox on their personal machine by filling in the following form: https://cadp.inria.fr/registration .

3. General Instructions

Each student team has between one and four LOTOS specifications to translate into LNT. The cumulated size is around 3300 lines of LOTOS (including blank lines and comments) for each team.

Detailed information about the examples to be processed will be given into a particular mail sent to each student team.

Each example is provided into a directory containing several types of files:

- There is usually a file named **=READ_ME.txt** that gives explanations about the example.

- The source LOTOS files are contained into files named *.**lotos** and, possibly, *.**lib** if "**library** … **endlib**" directives are used (such directives behave in the same way as **#include** directives in C/C++). You will have to replace these files with *.**lnt** files containing LNT code (notice that, in LNT, the distinction between *.**lotos** and *.**lib** files disappears: using the LNT system of modules, there are only *.**lnt** files).

- There may be additional C code (hand-written implementation of LOTOS abstract data types and functions) contained in files named *.**f** and *.**t** (where "f" and "t" respectively stand for functions and types). You will have to rename these files into *.**fnt** and *.**tnt**, respectively.

- There is also a file named **demo.svl** (or **demos.svl**) that specifies how the example can be verified using the CADP toolbox. You will have to adapt these files to reuse them, which can be done easily by changing, in the file, the "**.lotos**" extensions into "**.lnt**". If more changes are needed, you can contact the Help Desk (see Section 6 below).

- Finally, there can be other types of files (e.g., *.**aut**, *.**bcg**, *.**mcl**, *.**seq**) used for the verifications. These files should be kept unchanged.

Here are some examples of LOTOS-to-LNT translation (the original LOTOS specifications can be found in the sub-directory named “LOTOS”):

https://cadp.inria.fr/ftp/demos/demo_28
https://cadp.inria.fr/ftp/demos/demo_29
https://cadp.inria.fr/ftp/demos/demo_32

4. Translation Guidelines

Formal specifications are not only intended to be processed by computers; they are also intended to be read by humans, especially implementers, who will turn them into executable code, and maintainers, who will use them as documentation to understand the behaviour of the system and evolve it on the long run. Therefore, it is suitable to write them in an elegant, readable form, so that others will have pleasure reading them; this is also true for programs in general, but even more for formal specifications.

Guidelines for the translation from LOTOS to LNT are given in the “translation.pdf” document. Consult also the slides of Frédéric Lang’s lectures on LNT, and the Reference Definition of LNT. Here are additional recommendations.

To help us evaluating your project, the correspondence between the LOTOS and LNT specifications has to be preserved. To do so, please:

- Respect the style and conventions of the source LOTOS specifications.

- Refrain the desire to rename identifiers (or to change their case to lower or upper) even if

you don't like them. In particular, the gate and function names should not be modified as they appear on the transitions of the generated LTSs.

- Keep the comments given in the LOTOS specifications, possibly updating them where needed to make sense in the LNT specifications.

- Format your LNT code by using three spaces for indentation. Do not use the <Tab> character for this. Unfortunately, there is no LNT pretty printer available at the moment.

- If you have problems to provide LNT channels (an information that does not exist in LOTOS specifications), you can use the "**any**" channel.

If your text editor is Emacs, Gedit, Jedit, Nano, Notepad++, Pluma or Vi/Vim, you can find style files for the LNT language in the $CADP/ext directory of the CADP toolbox. These style files provide syntax highlighting.

5. Verification

- You can compile a LOTOS specification using the following command:
  lotos.open *file***.lotos generator** *file***.bcg**

  This will produce a binary file named *file***.bcg** containing the Labelled Transition System (LTS) corresponding to *file***.lotos.** This operation may fail if the LTS is too large. You can insert the "**-monitor**" option after "**generator**" to observe the growth of the LTS.

- Similarly, you can compile a LNT specification using the following command:
  lnt.open *file***.lnt generator** *file***.bcg**
  This command will first translate *file***.lnt** to *file***.lotos** and will fail if a file named *file***.lotos** already exists in the current directory. It is therefore recommended to put the original LOTOS files in a sub-directory named, e.g., LOTOS.

- You can obtain information about an LTS by typing the command:
  bcg_info *file***.bcg**

- You can check that a LOTOS specification *file1***.lotos** is equivalent to an LNT specification *file2***.lnt** by comparing their respective LTSs modulo strong bisimulation. This is done by typing the command:
  bcg_cmp -strong *file1***.bcg** *file2***.bcg**

  If the answer is TRUE, both specifications are equivalent. If the answer is false, then you must understand why they are different, in order to correct the LNT specification. This can be done as follows:
  bcg_cmp -strong -diag *diff***.bcg** *file1***.bcg** *file2***.bcg**
  and by inspecting visually the diagnostic file *diff***.bcg** if it is not too large:
  bcg_edit *diff.bcg*

  You can also execute step-by-step a LOTOS or LNT specification by typing:
  lotos.open *file***.lotos ocis**
  or

lnt.open *file*.**lnt ocis**

- Ensure that your LNT specifications are correct by running the (modified) SVL script. This file can be executed by typing the shell command **svl** in the directory containing the example. The LTS(s) produced from the LNT specification(s) should be equivalent (for strong bisimulation) to the LTS(s) produced from the original LOTOS specification(s).

6. Help Desk

- In you face unexpected problems, you can ask for support by send an e-mail to Hubert.Garavel@inria.fr and Frederic.Lang@inria.fr (include both of them as recipients).

- We would appreciate if you could share the result of your trials at writing LNT code: this will be extremely useful for testing the LNT compilers that are currently under development. To do so, execute the following shell commands:

```
cd /tmp
tar cf - `ls -d ${USER}_lnt.open_*` > lnt.tar
gzip lnt.tar
rm -rf ${USER}_lnt.open_*
```

  and send the resulting file /tmp/lnt.tar.gz to Hubert.Garavel@inria.fr . The LNT code fragments will be anonymized and used to produce new test cases, either correct or incorrect (incorrect LNT programs are useful to test the reaction of the LNT compilers).