Cours ISC

TP -- Translation from Simulink to Lustre

The goal of the following excercises is to understand the basic principles of the tool **sim2lus** for translation from Simulink models to Lustre programs. **The guidelines of the tool, which explain which models are translatable, and a short manual are given in the next part of the document**.

Exercise 1. Download some examples of Simulink models from the webpage of this course.
To use the tool **sim2lus**, we need to set some environment parameters before invoking sim2lus.

Experiment with the provided examples to understand the conditions for translatability and the required options of Simulink models to translate.

Exercise 2. Extract the Simulink model of the controller for the robot and translate it to Lustre.

Tool sim2lus -- Guidelines

The tool is available either on ensipc or directly on the local machines.

Before using the tool, we need to configure some environment parameters. Add the content of these files in the configuration file (such as .bashrc).

```
#### OSEK + LUSTRE
# seach a local install (ensisun)
# or use the teacher install (ensipc)
####
for idir in /usr/local /user/5/raymond ; do
   if [[ -f $idir/mdl2lus2osek/SETENV.sh ]]; then
      source $idir/mdl2lus2osek/SETENV.sh
      break
    fi
done
```

Then the tool is invoked simply by the command ``sim2lus".

1. Supported Simulink features

Not all the Simulink models can be translated to Lustre.

We only translate discrete-time Simulink models. Concretely, this means blocks of the "Discrete" library, generic mathematical operators such as sum, gain, logical and relational operators, other useful operators such as switches, and, finally, subsystems or triggered subsystems.

We only translate a Simulink model with open input, which is the case of embedded controllers.

The “semantics” of a Simulink model can be roughly defined by its simulation traces. Therefore, the simulation options should be provided together with the model. Not all the simulation options are supported by the tool. The limitations are explained in the following:

- **Solving methods**. We restrict the translation only to one method, namely, "**solver: fixed-step, discrete**" and "**mode: auto**". This means that Simulink models must be saved AND simulated correctly under the above simulation method before feeding them to **sim2lus**.

  If the controller to be translated is part of a bigger Simulink model containing both discrete and continuous part, we suggest that you copy/paste it to a new model, correct the parameters and simulate the new one. If Simulink does not give an error then you can use this model with the translator.

- **The tool does not translate S-functions or Matlab functions**. Such functions are often helpful. On the other hand, they can also create side-effects, which is something to be avoided and contrary to the "functional programming" spirit of Lustre.

- **Sampling time**. As the Simulink models to be translated are, in principle, controllers embedded in larger models containing both discrete and continuous parts, we assume that for every input of the model to be translated (i.e., every input of the controller) the sampling time is explicitly specified. This also helps the user to see the boundaries of the discrete and the continuous parts in the original model.

- **Type checks**. We want the Lustre program produced by the translator to type check if and only if the original Simulink model "type checks" (i.e., is not rejected by Simulink because of type errors). However, the behavior of the type checking mechanism of Simulink depends on the simulation method and the "Boolean logic signals" flag (BLS). Thus, we also **assume that BLS is on**. When set, BLS insists that inputs and outputs of logical blocks (and, or, not) be of type boolean. Not only this is good modeling and programming practice, it also makes type inference more precise and simplifies the verification of the translated Simulink models using Lustre-based model-checking tools.

- We also have to **set the "algebraic loop" detection mechanism of Simulink to the strictest degree**, which rejects models with such loops. These loops correspond to cyclic data dependencies in the same instant in Lustre. The Lustre compiler rejects such programs.

- The parameters of the blocks should be **numerical values**. For example, if the gain of a Gain block is given by a symbolical variable the numerical value of which is defined elsewhere, then we need to substitute the variable by its numerical value.

It should also be noted that Simulink is a product evolving in time. This evolution has an impact on the semantics of the tool. For instance, earlier versions of Simulink had weaker type-checking rules.

2. Usual known blocks

The translator groups blocks into three categories; "non-translatable", "known blocks" and the rest.

The first category contains blocks that we do not need to translate but are commonly found in Simulink models. This includes the "Probe" and "Scope" blocks. These blocks should not have outputs and also not affect the type or the Sample Time of any signal in the system.

In the second category we have the blocks for which we know the exact specifications such as

number of inputs and outputs, how they behave according to type and clock inference and how they are translated into Lustre code.

The last category contains the rest of the blocks. These blocks behave as "neutral" with respect to clock and type, meaning that they do not change anything between inputs and outputs. So during the type and clock inference phase no special manipulation is done. As for the code generation, since they are unknown blocks for **sim2lus** they are translated as external nodes.

Usual known Blocks

Sources
Inport, Constant, DiscretePulseGenerator,
Random, DataTypeConversion
Sinks
Outport, Ground, Terminator
Discrete
UnitDelay
Zero-OrderHold
Math Operations
Sum, Product, Gain, CombinatorialLogic,
LogicalOperator, RelationalOperator
Signal Routing
Mux, Demux, BusCreator, BusSelector, Switch
Signal Attributes
DataTypeConversion Ports & Subsystems
Subsystem, Trigger, Enable
various categories
Saturation, Lookup2D

Non-translatable blocks

Sinks
Scope, Probe, Display
and others

3. How to use the tool sim2lus

Recall that we need to set some environment parameters (PATH, etc) before invoking sim2lus.

Before translating, one needs to check the options of the Simulink to translate as follows.

- Supported options in .mdl models (check in the "Simulation parameters" box under "Simulation" menu, once you have opened a .mdl file):

o The "solver" option must be set to "fixed-step, discrete"
o The "mode" option must be set to "auto"
o The "boolean logic signals" ("advanced" menu) flag must be set to "on"
o The "algebraic loop" flag ("diagnostics" menu) must be set to "error"

- Inputs cannot have inherited sample times unless the option ``--monoperiodic|-mp" is passed to **sim2lus** during translation.

If the translation fails, **check all the above conditions**.