Tool sim2lus – Guidelines

Before using the tool, we need to configure some environment parameters:

- On ensisun, use the command: **source /usr/local/mdl2lus2osek/SETENV.sh**.
- On ensibm, use the command: **source /usr/local/lustre-lego/ALIRE.ensibm**

One can add the content of these files in the configuration file (such as .bashrc). 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 the discrete-time part of a Simulink model. 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.

These limitations are explained in the following points.

- **Solving methods**. Since Simulink semantics depends on the simulation method, 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 **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.

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. 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.

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

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.
- The model **MAY need to be saved under the version 13** of Simulink before translation.