Cyber-Component Verification Using HybridSAL

Ashish Tiwari

SRI International, Menlo Park, CA. ashish.tiwari@sri.com

Abstract. This tutorial describes the process of designing cyber-components of a complex system guided by formal verification.

1 The Tools

The tools used in the verification of cyber-controllers are the following:

Matlab Simulink Stateflow: We assume that the cyber-components are designed using Matlab's Simulink Stateflow language. Any changes to the controller models are performed using Matlab.
Preferred version: R13b

Matlab extension for specifying LTL properties: VU has developed scripts that extend the graphical user interface of Matlab Simulink/Stateflow with dialog boxes for specifying temporal logic properties using a pattern system. Desired properties of the controllers are attached to the Simulink subsystem or Stateflow chart using this extension.

MDL2MGACyber.exe: VU has developed a tool that translates a Matlab model (exported as a .mdl file) into an XML representation (cyber-composition language). Both the system and the LTL property are included in the XML file.
This tool is part of META release.

cybercomposition2hsal: SRI has developed a tool that translates models in the cyber composition language (.xml files) into the HybridSal formal verification language.
This tool is currently available as a separate script. It will soon be integrated into the META release.

hsalRA.exe: The HybridSal relational abstracter, combined with the infinite bounded model checkers.
The SAL tool is required to be installed separately. SAL executables are available from the SAL website. On Windows, SAL runs only under cygwin. The hybridsal tool is part of META release.

2 A Simple Exercise

We describe a simple design exercise using HybridSal. It consists of the following steps.

- Building/editing the cyber model and annotating it with LTL properties
- Converting the Matlab model into the cyber composition language
- Creating a HybridSal representation of the model and the properties
- Verifying the HybridSal model

2.1 Building/editing the cyber model

We start with controllers designed by VU. The Matlab files for these controllers are:

- Torque_Converter_control.mdl
- TorqueReductionSignal.mdl
- SimplifiedShiftController.mdl

These files can be opened in Simulink and edited. If the LTL-extension is installed, then LTL properties can be attached to these models directly by right clicking on the model.

Models (and the properties) are saved as .mdl files using Matlab.

2.2 Converting Matlab to Cyber Composition language

Using the `MDL2MGACyber.exe` tool, an XML representation of the models is generated. Corresponding to the above mdl files, we get the file

`SimplifiedControllerCyberWithProp.xml`

This file has all the three cyber-components and the attached properties.

2.3 Converting CyberComposition XML to HybridSal

The python script, `cybercomposition2hsal.py`, converts the XML into HybridSal. It is run as follows:

```
python cybercomposition2hsal.py SimplifiedControllersCyberWithProp.xml
```

Running the above script will generate a file called

`SimplifiedControllersCyberWithProp.hsal`.

Users can open this file and see the formal representation of the three cyber-components and the LTL properties.

2.4 Verifying the HybridSal model

The HybridSal file can now be model checked using the following command:

```
hsalRA.exe SimplifiedControllersCyberWithProp p1
```

where `p1` is the name of the LTL property being verified.

Running this command performs the following actions:

- A relational abstraction of the HybridSal model is first constructed. The result is stored in a file with the same filename, but with extension ".sal". In the above example, the tool will create a new file called

  ```
  SimplifiedControllersCyberWithProp.sal
  ```

- The SAL model is model-checked using an infinite-state model checker. The output is either a counter-example for the property, or a statement that no counter-example was found.

These two steps can be independently performed on the command-line as follows:

```
python HSalRelAbsCons.py SimplifiedControllersCyberWithProp.hsal
sal-inf-bmc -d 10 -v 3 SimplifiedControllersCyberWithProp p1
```

Design/property changes can be made using Matlab on the Simulink/Stateflow models. The verification process can then be repeated. Alternatively, it is also possible to edit the intermediate HybridSal or Sal files directly – this can save time – but the edits the not carried back to the Matlab models.

3 Remarks

In the above tutorial, only the cyber components were being analyzed. The plant model is completely abstracted. Hence, the analysis can be coarse. But, the analysis can still be useful, especially if care is taken in specifying the LTL properties. In particular, LTL properties should constrain the inputs of the controllers, and then check that the response of the controller is appropriate for that scenario. The predefined LTL templates may not be sufficient for this purpose.

The properties `p1` and `p2` in the HybridSal files were included to illustrate the utility. Property `p1` says that if the input `shift_requested` is consistently set to 1, then after a few steps, the output `gear_selected` of the controller is 1. This property was found to be invalid. Property `p2` says that if the input `shift_requested` is consistently set to 1, then after a few steps, the output `gear_selected` of the controller is at most 2. No counter-examples were found for this property. However, it is possible to fix the controller logic to make property `p1` true. Failure of `p1` was possibly a bug in the designed controller.