Formal Requirements Modeling for Simulation-Based Verification

Martin Otter¹, Nguyen Thuy², Daniel Bouskela², Lena Buffoni³, Hilding Elmqvist⁴, Peter Fritzson³, Alfredo Garro⁵, Audrey Jardin², Hans Olsson⁴, Maxime Payelleville⁶, Wladimir Schamai⁷, Eric Thomas⁶, Andrea Tundis⁵

¹Institute of System Dynamics and Control, DLR, Germany, Martin.Otter@dlr.de
³PELAB, Linköping University, Sweden, {Lena.Buffoni, Peter.Fritzson}@liu.se
⁴Dassault Systèmes AB, Sweden, {Hilding.Elmqvist, Hans.Olsson}@3ds.com
⁵DIMES, University of Calabria, Italy, {Alfredo.Garro, Andrea.Tundis}@unical.it
⁶Dassault Aviation, France, {Eric.Thomas, MP}@dassault-aviation.com
⁷Airbus Group Innovations, Germany, Wladimir.Schamai@airbus.com

This paper describes a proposal on how to model formal requirements in Modelica for simulation-based verification. The approach is implemented in the open source Modelica_Requirements library (see figure to the left). It requires extensions to the Modelica language, that have been prototypically implemented in the Dymola and Open-Modelica software. The design of the library is based on the FOrmal Require ment Modeling Language (FORM-L) defined by EDF, and on industrial use cases from EDF and Dassault Aviation.

The approach is to (a) provide the open source library Modelica_Requirements to define and model requirements in a formal way using 2- and 3-valued linear temporal logic and other description forms; (b) associating requirement models with behavioral models; (c) testing whether the defined requirements are violated by the system design currently studied when the underlying behavioral models are simulated.

For example the textually described requirement (from an EDF use case)

*When the MPS (Main Power Supply system) is switched off, signaled by Boolean Off, then the MPS must be declared Unavailable when it has been off for more than 6 accumulated seconds during any 10 seconds time window.*

can be modelled in the following, formal way with the Modelica_Requirements library:

![Modelica_Requirements Diagram](image-url)

**Component** minAccumulatedDuration2 outputs true, if in any time window of length 10 s variable Off was accumulated true for at least 6 s. This signal is the input to component during which requires that whenever the input is true, variable Unavailable must be true as well. In that case the block outputs Satisfied. If the input of during is true and Unavailable = false, the requirement is clearly violated and the during block outputs Violated (if the input is false, the block outputs Undecided).