Towards a Formalized Modelica Subset

Lucas Satabin[1] Jean-Louis Colaço[1] Olivier Andrieu[1] Bruno Pagano[1]

[1]Esterel Technologies/ANSYS SBU, France, firstname.lastname@ansys.com

Originally designed to address multi-physics simulation, Modelica integrates constructions to describe discrete system controllers since its specification version 3.3. I makes it possible to describe the physical environment as well as the system controllers within the same model. The ability to write controllers directly in Modelica makes it tempting to directly generate the actual embedded code from the model coupled with its simulation. Once this has been stated, there is only one small step that leads to considering embedding the generated code in safety-critical environments, such as airplanes.

Software that is embedded into such critical systems must respect qualification activities. Among them, we can find processes that must be followed for embedded code. These processes state requirements expecting strong guarantees on the language and tools that are used during developement. In particular essential guarantees are *determinism* and *absence of ambiguities*. One of this process is the DO-178C (2011) normative text used in aeronautics.

In this paper, we develop a formalization framework for a practical Modelica subset based in Thiele et al. (2012). The goal of this formalization is to provide a comprehensive and consistent description of the available constructs and how they interact. This approach has the advantage of being non ambiguous, in comparison to natural language formulation which cannot exhibit evidence that it cannot be interpreted in various contradictory ways. Furthermore, based on this formalism, it is straight forward to write an implementation that respects it.

This paper focuses on the static name resolution problem, and shows how it can be described within our framework. This example shall emphasize the advantages of such a formalization and illustrate how it can be used to solve ambiguities in the language design to meet the need of precise requirements in a qualified development process.

Even if this formalization is intereseting *per se*, it is also a first step that is necessary to define a complete modeling language that can be used to design safety critical applications. Our approach takes its origins from our previous experience on the Scade 6 language, the industrial dialect of the Lustre dataflow language, introduced in Halbwachs et al. (1991), used to produce qualifiable code to embed.

References

DO-178C. DO-178C Software Considerations in Airborne Systems and Equipment Certification, December 2011.

Nicolas Halbwachs, Paul Caspi, Pascal Raymond, and Daniel Pilaud. The synchronous dataflow programming language lustre. In *Proceedings of the IEEE*, 1991.

Bernhard Thiele, Stefan-Alexander Schneider, and Pierre R Mai. A Modelica Sub-and Superset for Safety-Relevant Control Applications. In *Proceedings of the Ninth International Modelica Conference*, 2012.