MetaModelica – A Symbolic-Numeric Modelica Language and Comparison to Julia

Peter Fritzson    Adrian Pop    Martin Sjölund    Adeel Asghar

PELAB – Programming Environment Lab, Dept. of Computer and Information Science
Linköping University, SE-581 83 Linköping, Sweden
{peter.fritzson,adrian.pop,martin.sjolund,adeel.asghar}@liu.se

Abstract

The need for integrating system modeling with advanced tool capabilities is becoming increasingly pronounced. For example, a set of simulation experiments may give rise to new data that are used to systematically construct a series of new models, e.g. for further simulation and design optimization. Such combined symbolic-numeric capabilities have been pioneered by dynamically typed interpreted languages such as Lisp and Mathematica. Such capabilities are also relevant for advanced modeling and simulation applications but lacking in the standard Modelica language. Therefore, this is a topic of long-running design discussions in the Modelica Design group. One contribution in this direction is MetaModelica, that has been developed to extend Modelica with symbolic operations and advanced data structures, while preserving safe engineering practices through static type checking and a compilation-based efficient implementation. Another recent effort is Modia, implemented using the Julia macro mechanism, making it dynamically typed but also adding new capabilities. The Julia language has appeared rather recently and has expanded into a large and fast-growing ecosystem. It is dynamically typed, provides both symbolic and numeric operations, advanced data structures, and has a just-in-time compilation-based efficient implementation. Despite independent developments there are surprisingly many similarities between Julia and MetaModelica. This paper presents MetaModelica and its environment as a large case study, together with a short comparison to Julia. Since Julia may be important for the future Modelica, some integration options between Modelica tools and Julia are also discussed, including a possible approach for implementing MetaModelica (and OpenModelica) in Julia.

Keywords: Modelica, MetaModelica, symbolic, Julia, meta-programming, language, compilation

Figure 1. The integrated MetaModelica OMEdit-based development environment in debugging mode. Left: the package browser. Top: the active stack frames (including C routines) and breakpoints. Middle: text editing and breakpoint setting. Right: the local variables browser. The user can switch to modeling mode which has both textual and graphical editing.