

Type Safety of Equation-Based Object-Oriented Modeling Languages

David Broman and Peter Fritzson

Department of Computer and Information Science, Linköping University, Sweden

Background

Computer aided modeling and simulation of complex physical systems, using components from multiple domains, such as electrical, mechanical, and hydraulic, have in recent years witnessed a significant growth of interest. General-purpose simulation tools, e.g., Simulink, using block diagrams and causal connections have dominated the area for years. However, in the last decade novel languages, (e.g., Modelica[1], gPROMS[2], and VHDL-AMS[3]) based on acausal modeling using Differential Algebraic Equations (DAEs), have evolved. We call these kind of languages Equation-based Object-Oriented (EEO).



Figure 1. An ABB-robot that could be modeled and simulated by an equation-based object-oriented (EEO)-language.

The Problems

Large and Complex Languages

- Currently available EEO-languages are large and complex, which make them hard to analyze and reason about.
- To the best of our knowledge, no EEO-language has a complete precise formal semantics including formal type system. Hence, no type-soundness proofs exist for this kind of languages.

Early Error Detection

No complete solution exists for:

- Detecting and debugging under- or over-constrained systems of equations at the component and type system level.
- Static Unit checking. Detect and automatically convert between physical units (e.g., newton, meter, and ampere).

Separately Compiled Components

In for example Modelica, no sound solution exists to separately compile components and still make full use of symbolic transformations.

What is an EEO-Language?

Equation-based Object-Oriented (EEO)-languages typically possess the following characteristics:

- The continuous-time behavior is described by using Differential Algebraic Equations (DAEs).
- Graphical modeling using well-defined graphical components.
- Textual modeling of components using a high-level language syntax.
- Making use of OO-concepts, such as inheritance, subtyping, and abstraction.

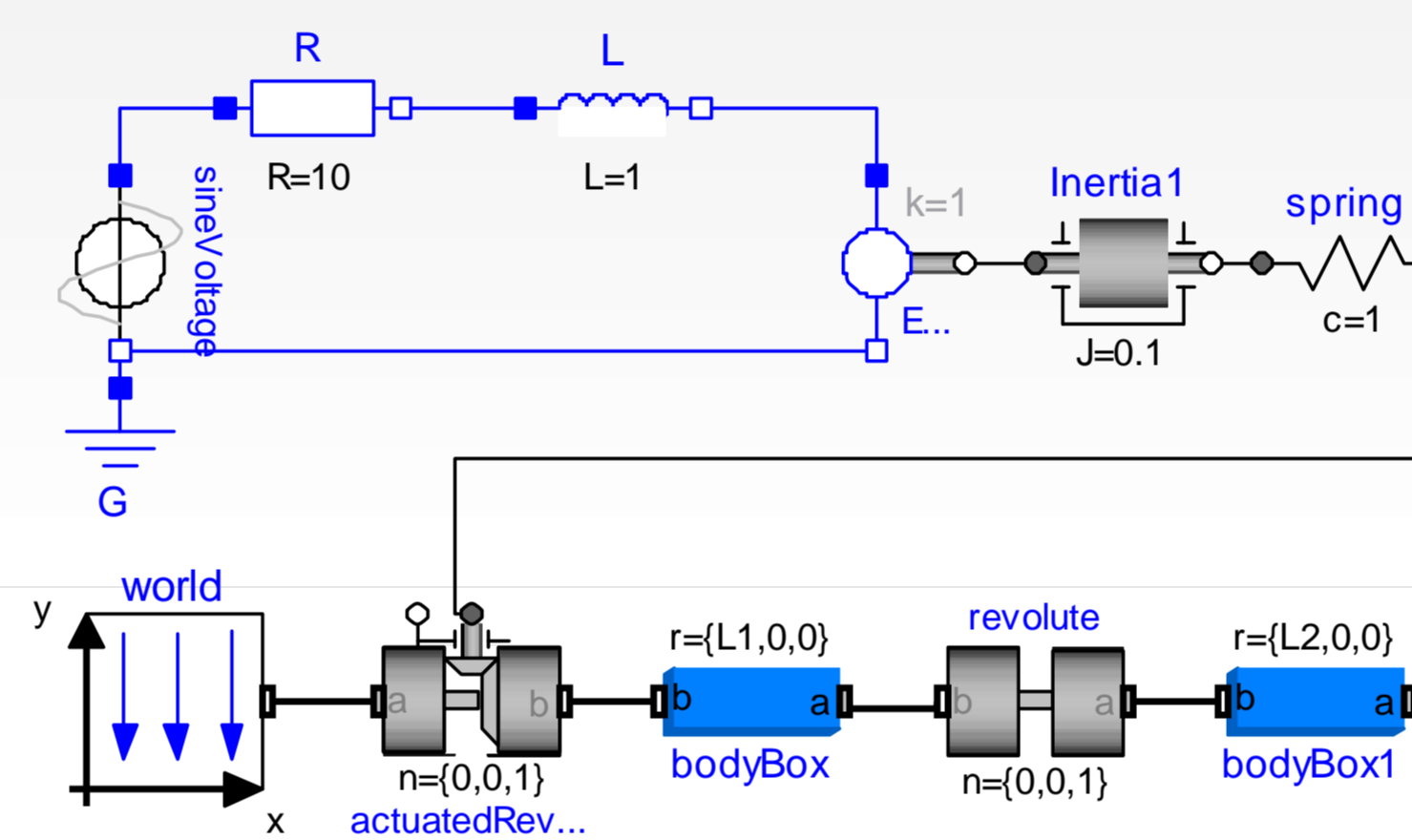


Figure 2. Modelica model of a motor connected to a double pendulum. Illustrates multi-domain usage, where an electrical circuit is combined with mechanical components.

Objectives and Approach

Project Objectives

- Improve the theoretical foundation of EEO-languages.
- Enable engineers to detect modeling errors at an early stage.

Our Approach

Define a formally specified kernel language, called Modeling Kernel Language (MKL) that:

- Describes the core concepts of EEO-languages.
- Makes use of static type checking.
- Is minimal and expressive.

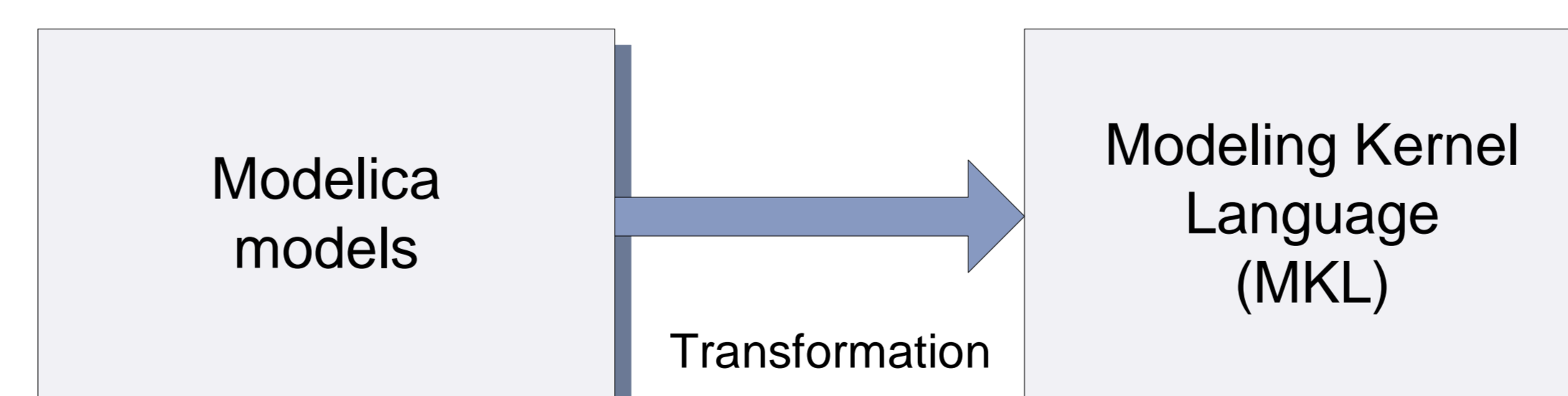


Figure 4. Illustrates our approach of developing a precise formal kernel language to which models constructed in for example the Modelica language can be transformed.

Acknowledgement

This research work is funded by CUGS (the Swedish National Graduate School in Computer Science), by SSF under the VISIMOD project, and by Vinnova under the NETPROG Safe and Secure Modeling and Simulation on the GRID project.

Why is EEO Important?

Acausal Modeling

Modeling is primarily based on equations, rather than statements.

Hybrid Modeling

Both continuous- and discrete-time behavior of a system can be described.

Multi-Domain

Can be used for modeling complex physical systems in multiple domains, e.g., electrical, mechanical, and hydraulic.

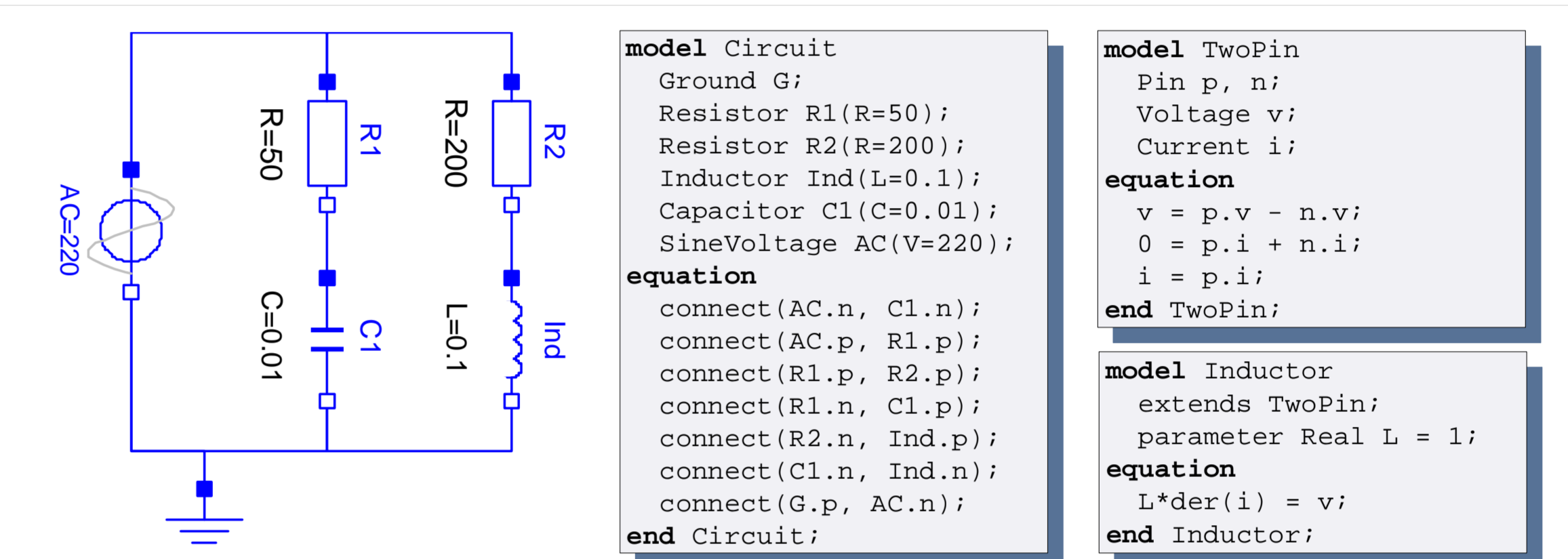


Figure 3. Simple electrical circuit viewed both as graphical components and textual source code.

Validation

Correctness

- Formal operational semantics including type system.
- Type soundness proofs.

Relevance and Usefulness

- Transform real Modelica models collected from industry to MKL.
- Perform experiments on a prototype implementation of MKL.

References

- [1] Modelica Association. *Modelica - A Unified Object-Oriented Language for Physical Systems Modeling – Language Specification Version 2.2*, February 2005. Available from: <http://www.modelica.org>
- [2] M. Oh and C. C. Pantelides. *A Modelling and Simulation Language for Combined Lumped and Distributed Parameter Systems*. *Computers and Chemical Engineering*, 20(6-7):611-633, 1996.
- [3] Ernst Christen and Kenneth Bakalar. *VHDL-AMS – A Hardware Description Language for Analog and Mixed-Signal Applications*. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 46(10):1263-1272, 1999.



Further Information

Please contact David Broman davbr@ida.liu.se or Prof. Peter Fritzson petfr@ida.liu.se for more information. An electronic version of this poster can be downloaded at www.ida.liu.se/~davbr.