Abstract Syntax Can Make the Definition of Modelica Less Abstract

1st International Workshop on Equation-Based Object-Oriented Languages and Tools

Berlin

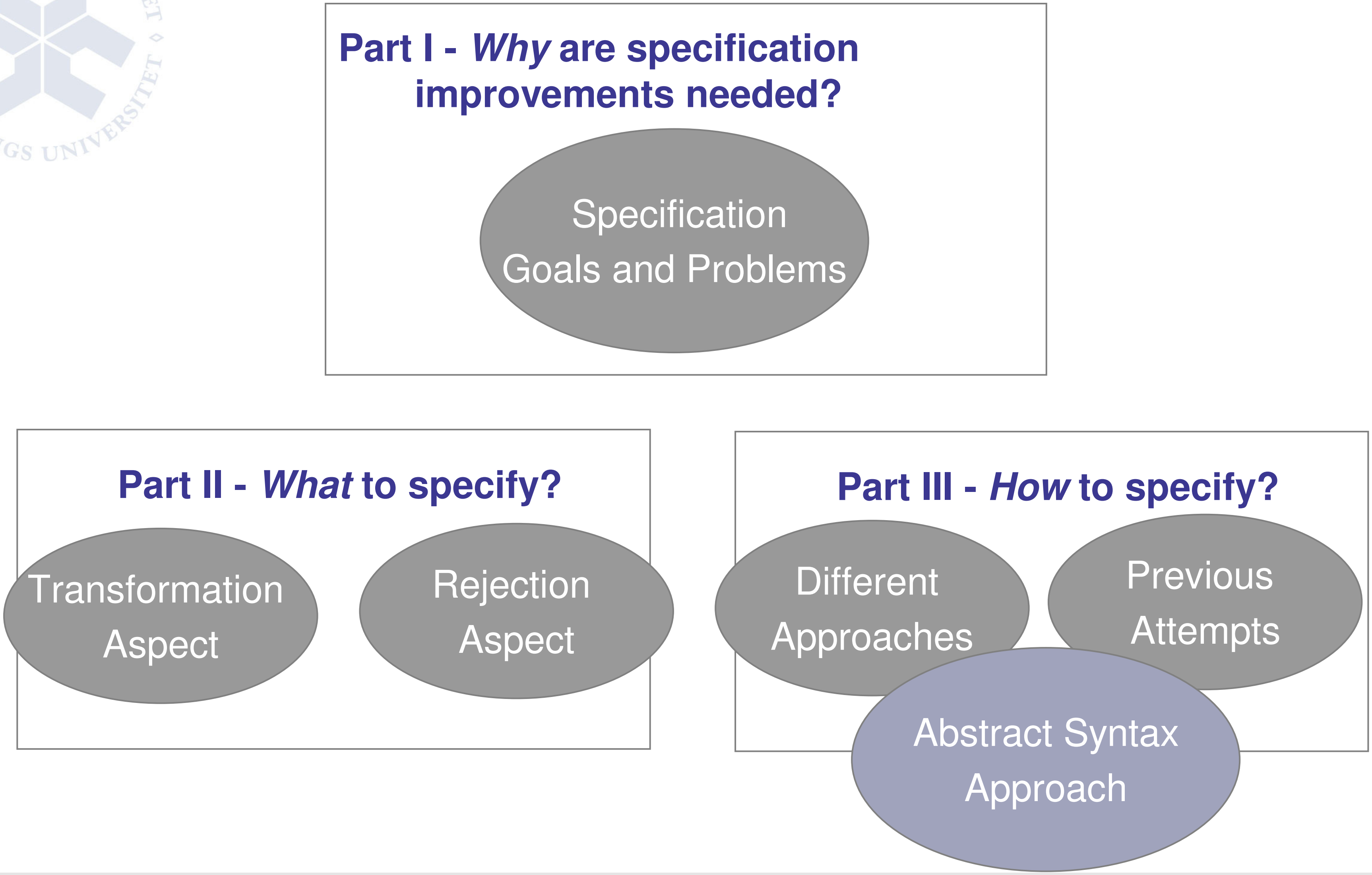
July 30, 2007

David Broman and Peter Fritzson

Linköping University, Sweden

{davbr,petfr}@ida.liu.se

Outline of the Talk



Part I - *Why* are specification improvements needed?

Specification Goals and Problems

Part II - *What* to specify?

Transformation Aspect

Rejection Aspect

Part III - *How* to specify?

Different Approaches

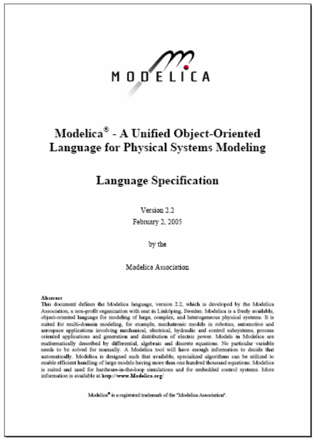
Previous Attempts

Abstract Syntax Approach

Part I

Why are specification improvements needed?

Problem: Interpreting the specification



MODELICA

Modelica® - A Unified Object-Oriented Language for Physical Systems Modeling

Language Specification

Version 2.2
February 2, 2005

by the

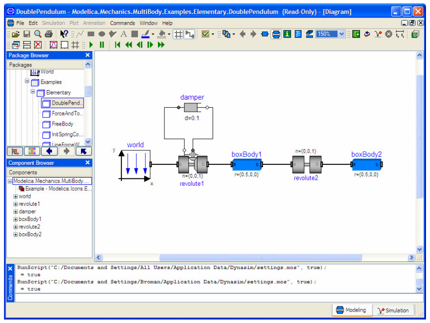
Modelica Association

Abstract
This document defines the Modelica language, version 2.2, which is developed by the Modelica Association, a non-profit organization with seat in Linköping, Sweden. Modelica is a freely available, object-oriented language for modeling of large, complex, and heterogeneous physical systems. It is suited for multi-domain modeling, for example, mechatronic models in robotics, automotive and aerospace applications involving mechanical, electrical, hydraulic and control subsystems, process oriented applications and generation and distribution of electric power. Models in Modelica are mathematically described by differential, algebraic and discrete equations. No particular variable needs to be solved for manually. A Modelica tool will have enough information to decide that automatically. Modelica is designed such that available, specialized algorithms can be utilized to enable efficient handling of large models having more than one hundred thousand equations. Modelica is suited and used for hardware-in-the-loop simulations and for embedded control systems. More information is available at **http://www.Modelica.org/**

Modelica® is a registered trademark of the "Modelica Association".

Modelica Language Specification

Interpretation of syntax and semantics



```
RunScript("C:/Documents and Settings/All Users/Application Data/Dynasim/settings.mos", true);
 = true
RunScript("C:/Documents and Settings/Broman/Application Data/Dynasim/settings.mos", true);
 = true
```

Simulation Tool

Problem

- The Modelica Specification is open for interpretation

- May result in incorrect and incompatible tools

- Dymola
- Modelica System Designer
- MOSILAB
- SimulationX
- OpenModelica

pelab
Linköping University
David Broman, davbr@ida.liu.se

Specification goals

Unambigious

"Can only be interpreted in exactly one way"

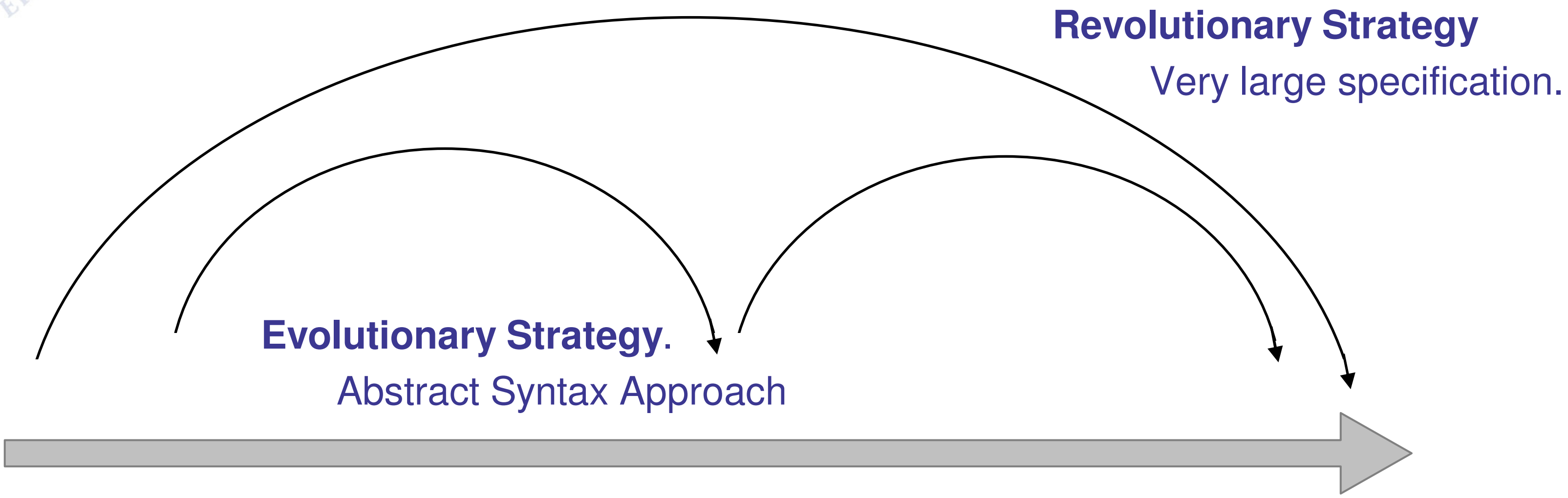
Understandable

"Easy to grasp with moderate computer science knowledge"

Expressive

"State the syntax and semantics in a compact manner"

An evolutionary "middle-way" strategy



Revolutionary Strategy

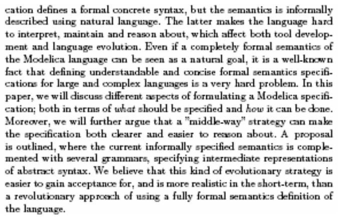
Very large specification.

Evolutionary Strategy.

Abstract Syntax Approach

Informal approach

e.g. natural language text



cation defines a formal concrete syntax, but the semantics is informally described using natural language. The latter makes the language hard to interpret, maintain and reason about, which affect both tool development and language evolution. Even if a completely formal semantics of the Modelica language can be seen as a natural goal, it is a well-known fact that defining understandable and concise formal semantics specifications for large and complex languages is a very hard problem. In this paper, we will discuss different aspects of formulating a Modelica specification; both in terms of *what* should be specified and *how* it can be done. Moreover, we will further argue that a "middle-way" strategy can make the specification both clearer and easier to reason about. A proposal is outlined, where the current informally specified semantics is complemented with several grammars, specifying intermediate representations of abstract syntax. We believe that this kind of evolutionary strategy is easier to gain acceptance for, and is more realistic in the short-term, than a revolutionary approach of using a fully formal semantics definition of the language.

Formal approach

e.g. operational semantics

$$\frac{\Gamma \vdash e_1 : \text{bool} \qquad \Gamma \vdash e_2 : T \qquad \Gamma \vdash e_3 : T}{\Gamma \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 : T}\ (\text{t-if})$$

Part II

What to specify?

What should be specified?

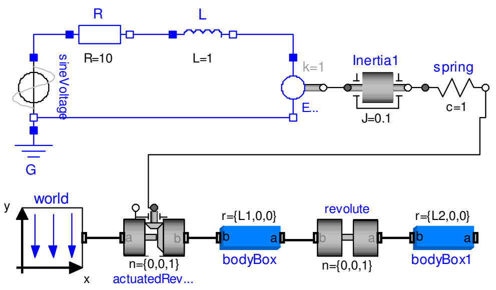
Syntax

- the structure

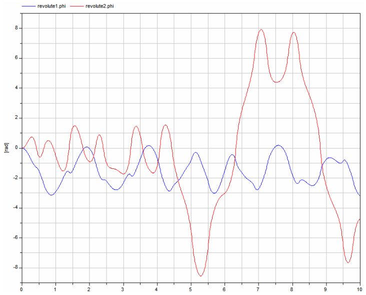
Semantics

- the meaning

1. Transformation Aspect

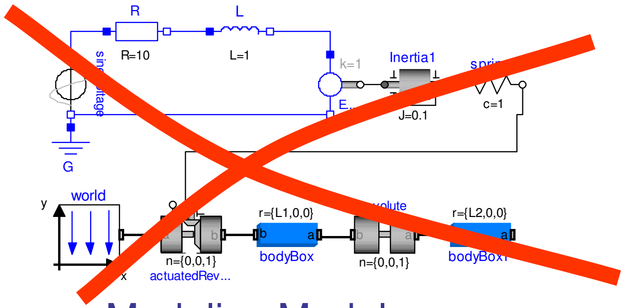


Modelica Model



Simulation Result

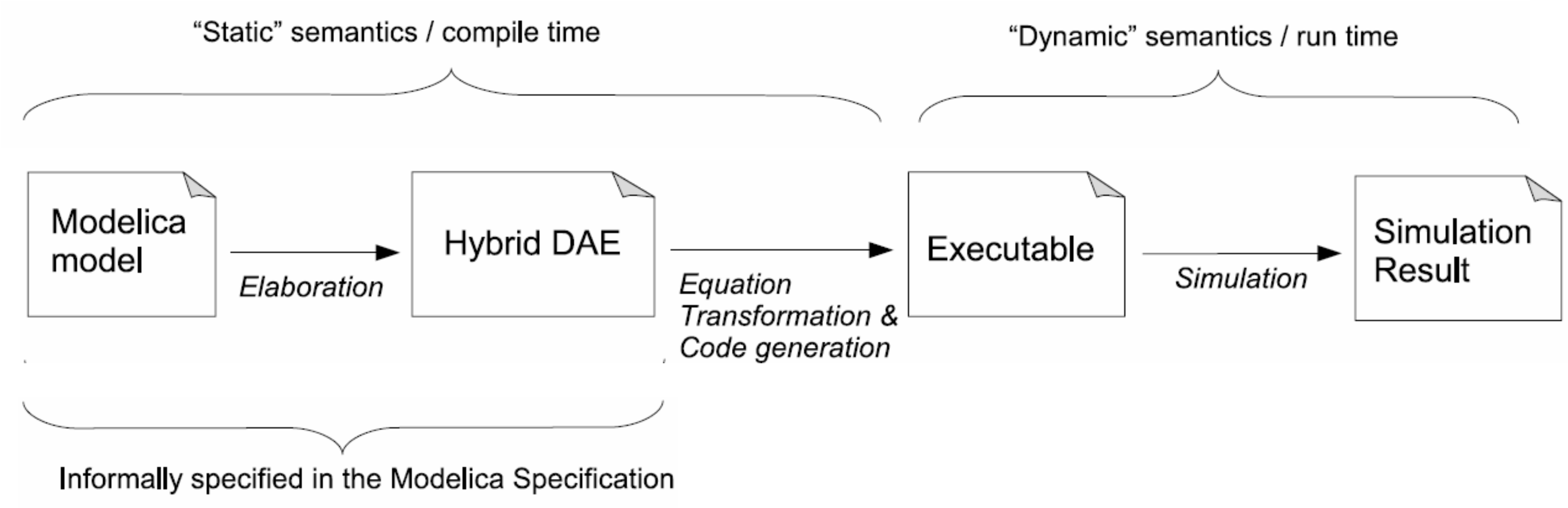
2. Rejection Aspect



Modelica Model

Transformation Aspect

What is actually the result of an execution?



"Static" semantics / compile time

"Dynamic" semantics / run time

Modelica model

Elaboration

Hybrid DAE

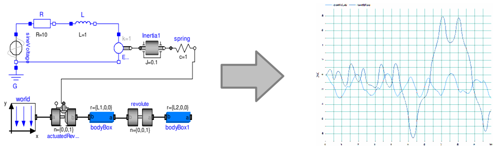
Equation Transformation & Code generation

Executable

Simulation

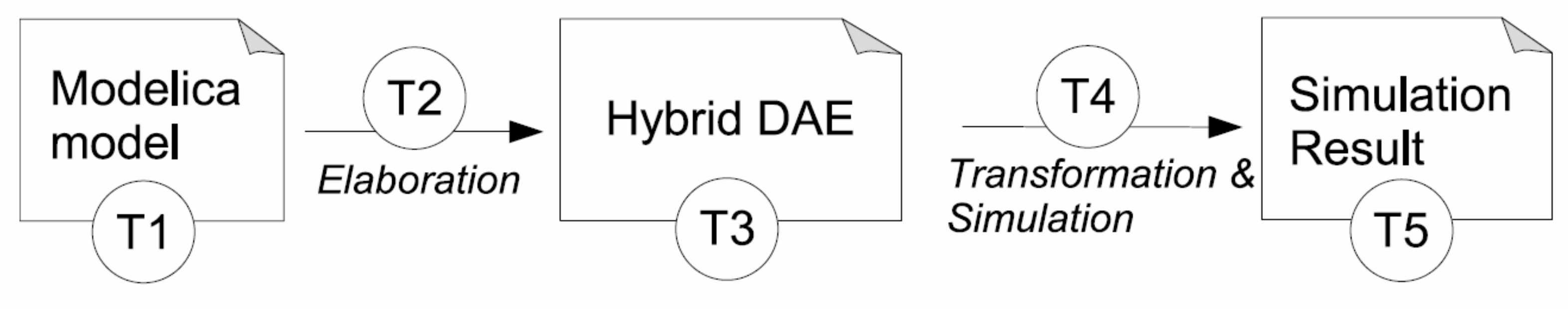
Simulation Result

Informally specified in the Modelica Specification



Rejection Aspect

What is actually a valid Modelica model?



Modelica model

T1

T2

Elaboration

Hybrid DAE

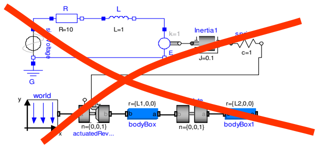
T3

T4

Transformation & Simulation

Simulation Result

T5



Part III

How to specify?

Previous Specification Attempts

Types in Modelica

(Broman, Fritzson & Furic, 2006)

- Clarify the type concept in Modelica
- Concerns only the rejection aspect

Instance Creation (Elaboration)

(Mauss, 2005)

- Only transformation aspect
- Subset of the language

Modelica Specification

(Modelica Association, 2005)

- Informal semantics, natural language
- Concrete syntax

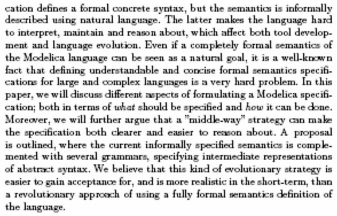
RML and Natural Semantics

(Kågedal & Fritzson, 1998)

- 1998, large subset of specfication
- hard to get an overview of - became very large
- now the code base for OpenModelica

Informal approach

e.g. natural language text



cation defines a formal concrete syntax, but the semantics is informally described using natural language. The latter makes the language hard to interpret, maintain and reason about, which affect both tool development and language evolution. Even if a completely formal semantics of the Modelica language can be seen as a natural goal, it is a well-known fact that defining understandable and concise formal semantics specifications for large and complex languages is a very hard problem. In this paper, we will discuss different aspects of formulating a Modelica specification; both in terms of *what* should be specified and *how* it can be done. Moreover, we will further argue that a "middle-way" strategy can make the specification both clearer and easier to reason about. A proposal is outlined, where the current informally specified semantics is complemented with several grammars, specifying intermediate representations of abstract syntax. We believe that this kind of evolutionary strategy is easier to gain acceptance for, and is more realistic in the short-term, than a revolutionary approach of using a fully formal semantics definition of the language.

Abstract Syntax Approach

Formal approach

e.g. operational semantics

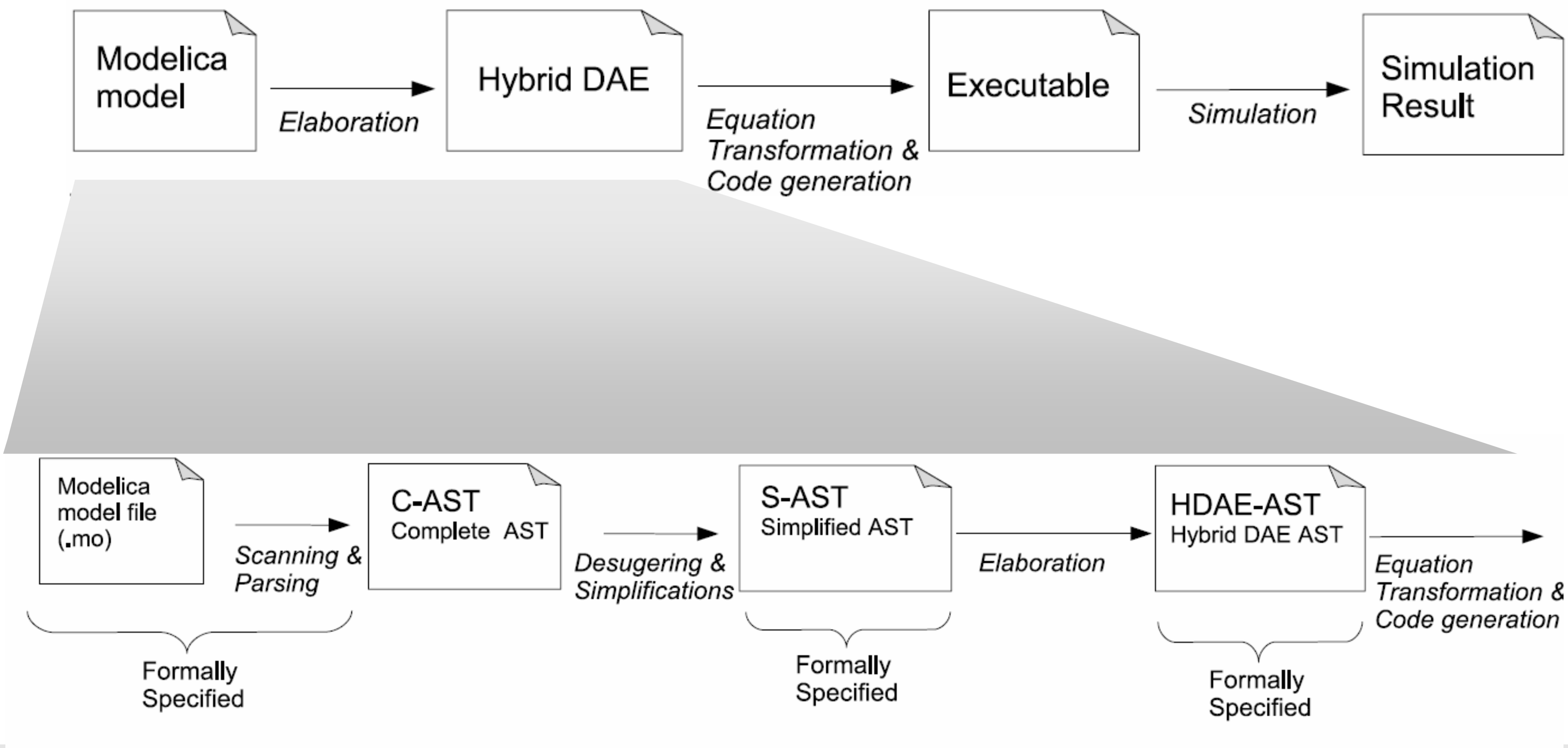
$$\frac{\Gamma \vdash e_1 : \mathrm{bool} \qquad \Gamma \vdash e_2 : T \qquad \Gamma \vdash e_3 : T}{\Gamma \vdash \mathrm{if}\ e_1\ \mathrm{then}\ e_2\ \mathrm{else}\ e_3 : T}\ (\text{t-if})$$

Abstract Syntax as a Middle-Way Strategy

Scanning, Parsing => Abstract Syntax Tree (AST)

Elaboration (transformation and rejection aspects)

- Input: Implicitly specified using concrete syntax
- Output: Not specified
- Transformation: Informally specified using natural language



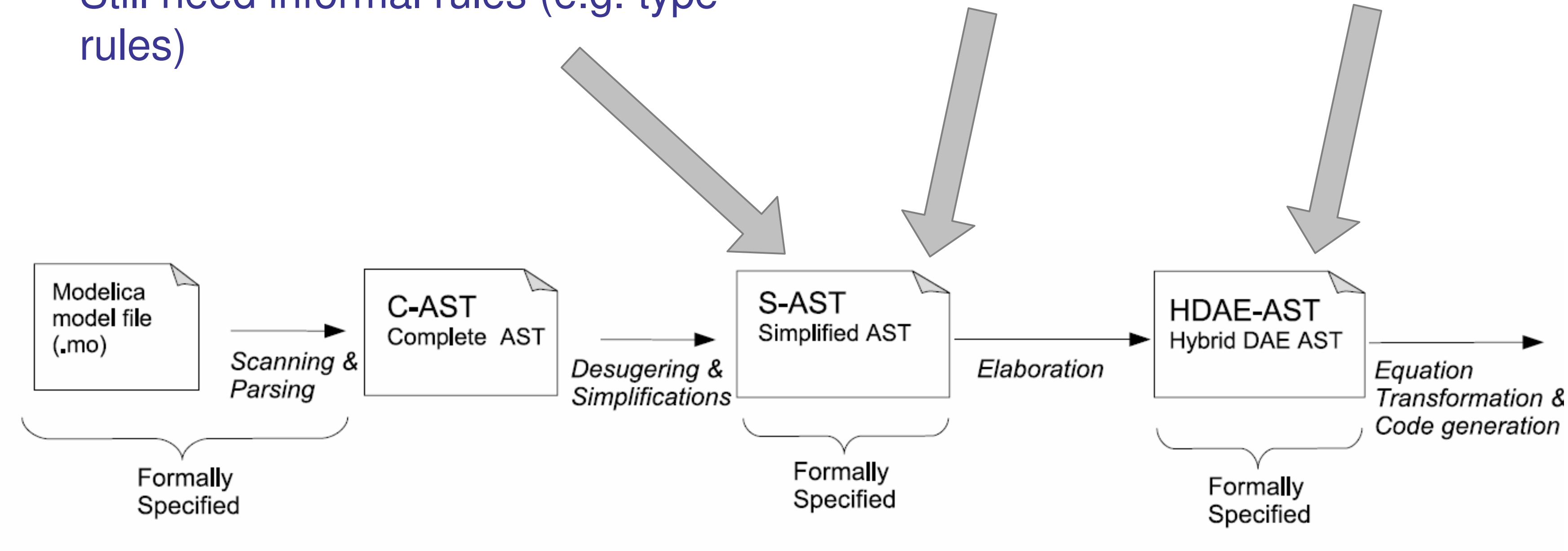
Transformation and Rejection Aspects

Rejection Aspect

- More restrictive than the concrete syntax grammar
- Include context sensitive information
- Still need informal rules (e.g. type rules)

Transformation Aspect

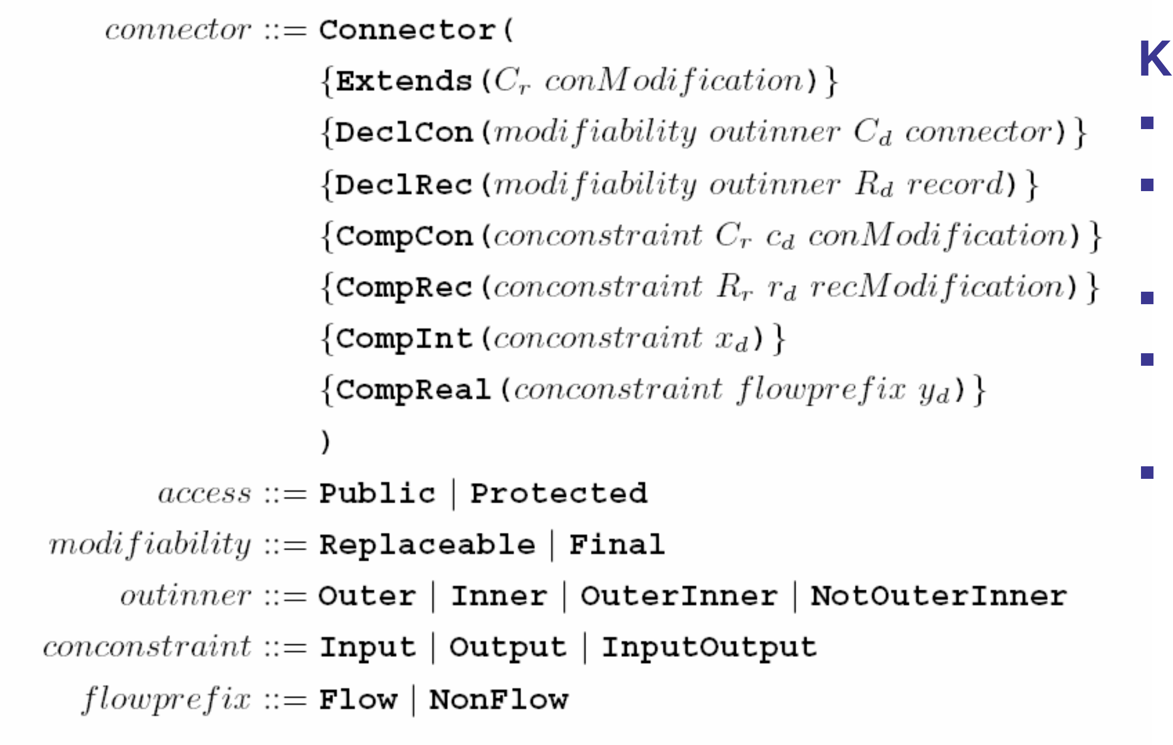
- Precise specification of input and output (grammar)
- Transformation still informal



Example of a grammar fragment (S-AST)

Modelica specification about connectors:

"No equations are allowed in the definition or in any of its components"



$$
\begin{aligned}
connector &::= \texttt{Connector(} \\
&\quad \{\texttt{Extends(}C_r \; conModification\texttt{)}\} \\
&\quad \{\texttt{DeclCon(}modifiability \; outinner \; C_d \; connector\texttt{)}\} \\
&\quad \{\texttt{DeclRec(}modifiability \; outinner \; R_d \; record\texttt{)}\} \\
&\quad \{\texttt{CompCon(}conconstraint \; C_r \; c_d \; conModification\texttt{)}\} \\
&\quad \{\texttt{CompRec(}conconstraint \; R_r \; r_d \; recModification\texttt{)}\} \\
&\quad \{\texttt{CompInt(}conconstraint \; x_d\texttt{)}\} \\
&\quad \{\texttt{CompReal(}conconstraint \; flowprefix \; y_d\texttt{)}\} \\
&\quad \texttt{)} \\
access &::= \texttt{Public} \mid \texttt{Protected} \\
modifiability &::= \texttt{Replaceable} \mid \texttt{Final} \\
outinner &::= \texttt{Outer} \mid \texttt{Inner} \mid \texttt{OuterInner} \mid \texttt{NotOuterInner} \\
conconstraint &::= \texttt{Input} \mid \texttt{Output} \mid \texttt{InputOutput} \\
flowprefix &::= \texttt{Flow} \mid \texttt{NonFlow}
\end{aligned}
$$

Key points

- Extended BNF style
- States allowed local class types and component types
- States allowed prefixes
- Meta-variables, declarative context-sensitive information
- Verbose - intended for specification, not implementation

Concluding Summary

Specification Goals

- Unambiguous
- Understandable
- Expressive

What to specify

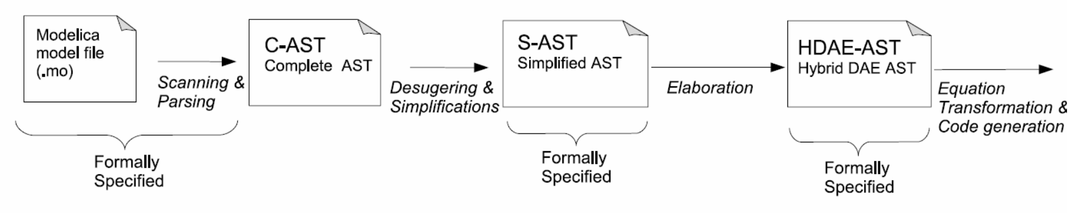
- Transformation aspects
- Rejection aspects

How to specify

- Current specification is informal, e.g. open for interpretation
- Evolutionary "middle-way" approach

Abstract Syntax Approach

- Specify INPUT and OUTPUT using grammars
- Complements the informal semantics


Modelica model file (.mo)
Scanning & Parsing
C-AST Complete AST
Desugering & Simplifications
S-AST Simplified AST
Elaboration
HDAE-AST Hybrid DAE AST
Equation Transformation & Code generation
Formally Specified
Formally Specified
Formally Specified

Thanks for listening!

Next Step

- Needs to be accepted by Modelica Association as the fundamental approach.
- Yet another "specification" of a subset is less meaningful
- There can only be one specification...