Modeling Kernel Language (MKL)

A formal and extensible approach to equation-based modeling languages

Guest Talk, EECS, Chess, UC Berkeley

February 17, 2011

David Broman

Department of Computer and Information Science
Linköping University, Sweden
david.broman@ida.liu.se

Linköpings universitet

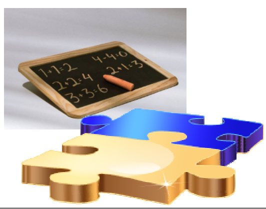
Agenda

Part I
What is an EOO Language?

$$\begin{aligned} J_1\dot{\omega}_1 &= M_v - M_1 \\ J_2\dot{\omega}_2 &= M_h - M_2 \\ \omega_1 &= -r\omega_2 \\ M_1 &= -r^{-1}M_2 \end{aligned}$$

Part II
Why MKL?

Part III
Expressiveness, Extensibility, and Formalization



Part I

What is an EOO language?

$$
\begin{aligned}
J_1\dot{\omega}_1 &= M_v - M_1 \\
J_2\dot{\omega}_2 &= M_h - M_2 \\
\omega_1 &= -r\omega_2 \\
M_1 &= -r^{-1}M_2
\end{aligned}
$$

What is Modeling and Simulation?

experiment on…

Simulation

$$
\begin{aligned}
J_1\dot{\omega}_1 &= M_v - M_1 \\
J_2\dot{\omega}_2 &= M_h - M_2 \\
\omega_1 &= -r\omega_2 \\
M_1 &= -r^{-1}M_2
\end{aligned}
$$

Mathematical Model
Differential-Algebraic Equations (DAEs)

Model

Modeling

answer questions about…

System

Equation-Based Object-Oriented (EOO) Languages

David Broman
david.broman@liu.se

Domain-Specific Language (DSL)

- **Primarily domain:** Modeling of physical systems
- **Multiple physical domains:** e.g., mechanical, electrical, hydraulic

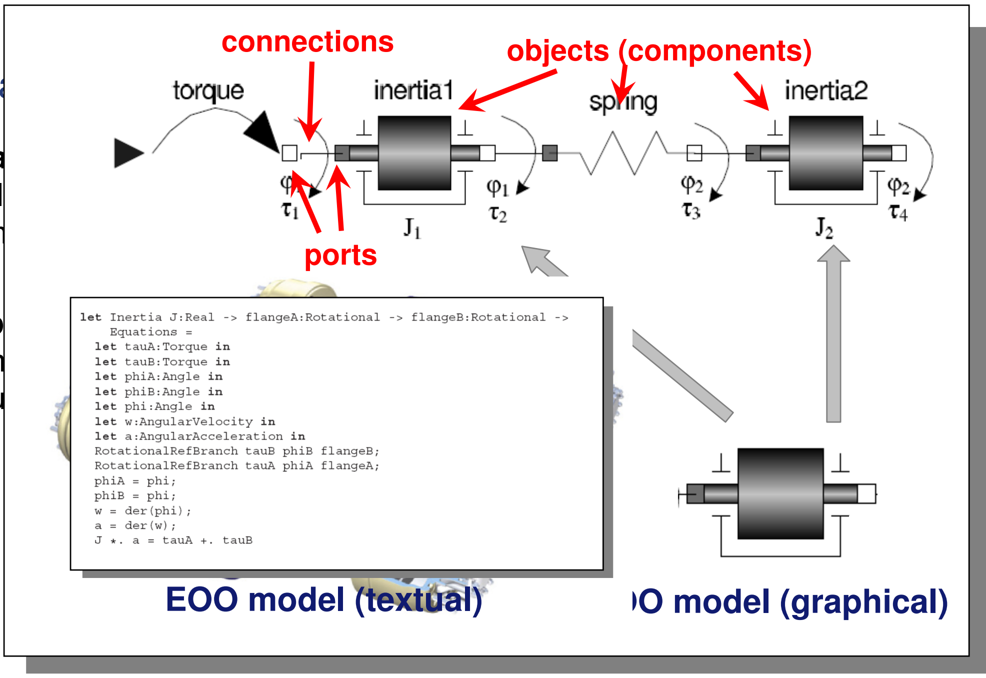
Equation-Based Object-Oriented (EOO)

Models and Objects

- **Object in e.g., Java, C++:** object = data + methods
- **Objects in EOO languages:** object = data + equations

Equation-Based Object-Oriented (EOO) Languages

David Broman
david.broman@liu.se



Equation-Based Object-Oriented (EOO) Languages

David Broman
david.broman@liu.se

Equation-Based Object-Oriented (EOO)

Domain-Specific Language (DSL)

- **Primarily domain:** Modeling of physical systems
- **Multiple physical domains:** e.g., mechanical, electrical, hydraulic

Models and Objects

- **Object in e.g., Java, C++:** object = data + methods
- **Objects in EOO languages:** object = data + equations

Acausality

- **At the equation-level**
  u = R * i
- **At the object connection level**

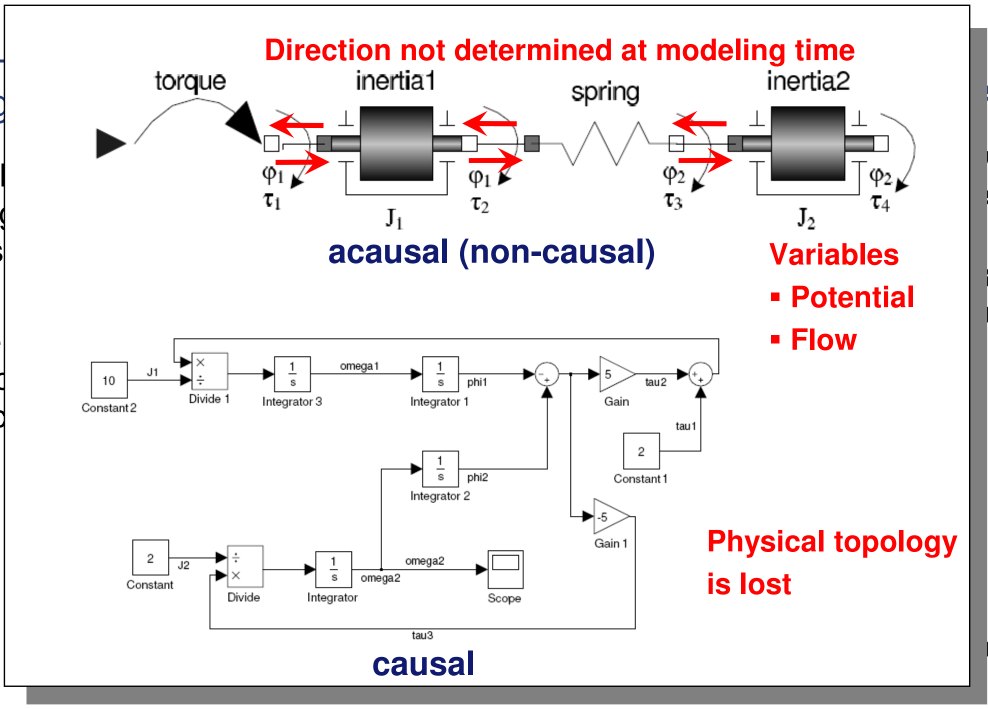
Equation-Based Object-Oriented (EOO) Languages

David Broman
david.broman@liu.se

Domain-
Languag

- **Primaril**
  Modeling
  systems
- **Multiple**
  e.g., mec
  hydraulic

cts
va, C++:
thods
anguages:
uations
evel



Equation-Based Object-Oriented (EOO) Languages

David Broman
david.broman@liu.se

Domain-
Languag

- Primaril
Modeling
systems

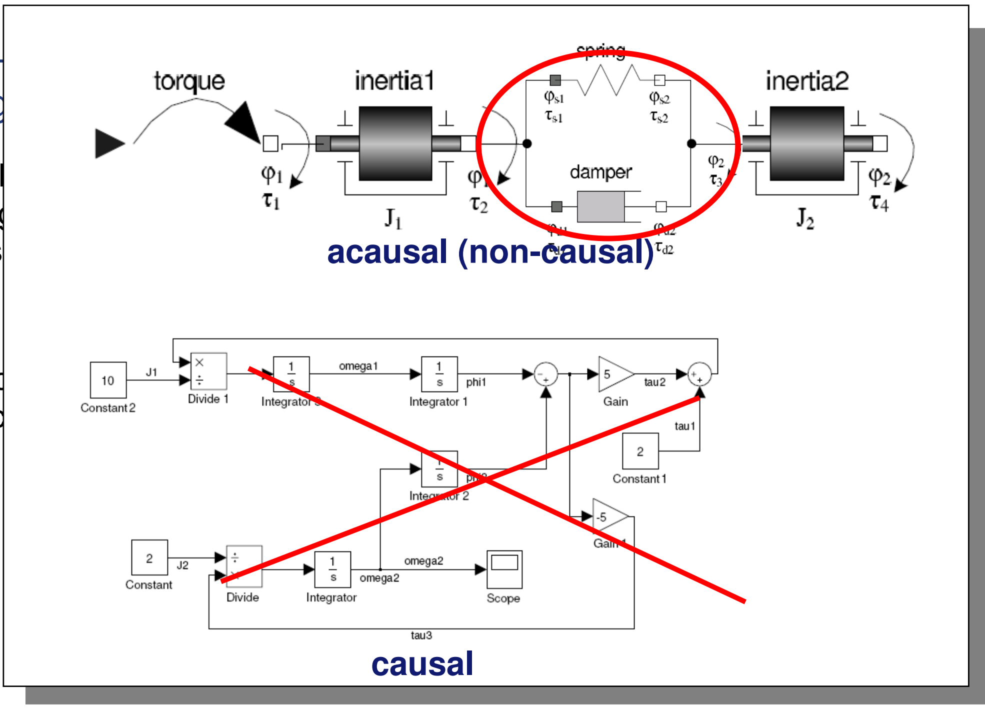
- Multiple
e.g., mec
hydraulic

ects

va, C++:
ethods

anguages:
uations

evel



Equation-Based Object-Oriented (EOO) Languages

David Broman
david.broman@liu.se

Domain-Specific Language (DSL)

- **Primarily domain:** Modeling of physical systems
- **Multiple physical domains:** e.g., mechanical, electrical, hydraulic

- **Modelica**
- **VHDL-AMS**
- **gPROMS**

Equation-Based Object-Oriented (EOO)

Models and Objects

- **Object in e.g., Java, C++:** object = data + methods
- **Objects in EOO languages:** object = data + equations

Acausality

- **At the equation-level**
  u = R * i
- **At the object connection level**

David Broman
david.broman@liu.se

Part II

Why MKL?

Expressiveness

David Broman
david.broman@liu.se

Expressiveness – ease and possibility of expressing complex models or tasks

Language versions: A, v1.0 → A, v1.1 → A, v2.0 → A, v2.2

Standard library versions: L, v1.0 → L, v1.1 → L, v2.0 → L, v2.2

Extensibility

David Broman
david.broman@liu.se

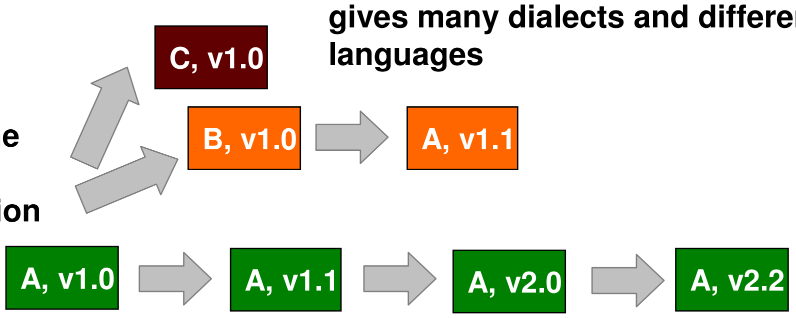
Extensibility – mechanisms to add new language features

Uses

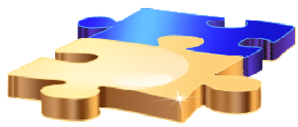
- **Simulation**
- **Optimization**
- **Code generation for real-time**
- **Model export**
- **Grey-box system identification**

etc.

gives many dialects and different languages



gives larger and more complex languages



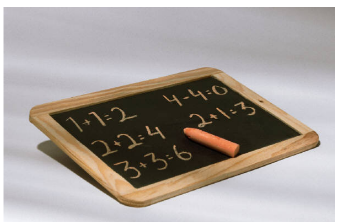
Formalization

David Broman
david.broman@liu.se

Formalization – precise semantics “meaning” of the language

Language Specifications of state-of-the-art are informally defined

- **hard to interpret unambiguously when developing compilers**
- **hard to reason about when extending the language**
- **hard to formalize e.g. Modelica due to size and complexity**



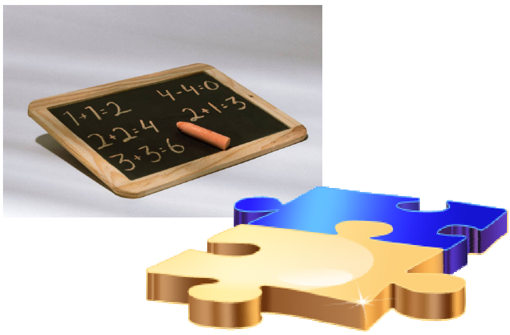
What is MKL?

Modeling Kernel Language (MKL)

- **Purpose: Research language – explore new concepts**
- **Bottom-up approach**
- **Small extensible language**
- **Precise formal semantics**
- **Base it on a proven foundation – the lambda calculus**
- **Statically typed functional language**
- **Platform for experimental equation-based DSLs (Continuous-time, hybrid, structural dynamic, and acusal models).**

Part III

Expressiveness, Extensibility, and Formaliztion



Expressiveness - HOAM

David Broman
david.broman@liu.se

Higher-Order Acusal Models (HOAM)

Higher-Order Functions
I.e. first class citizens, can be passed around as any value

+

Acausal Models
Models in EOO languages, composing DAEs and other interconnected models.

=

Higher-Order Acausal Models
I.e., first class acausal models.

Expressiveness - HOAM

David Broman
david.broman@liu.se

DEFINITION 3 (Higher-Order Acausal Model (HOAM)).
A higher-order acausal model is an acausal model, which can be

1. *parametrized with other HOAMs.*
2. *recursively composed to generate new HOAMs.*
3. *passed as argument to, or returned as result from functions.*

Replaces several of Modelica's constructs with one concept, e.g.,

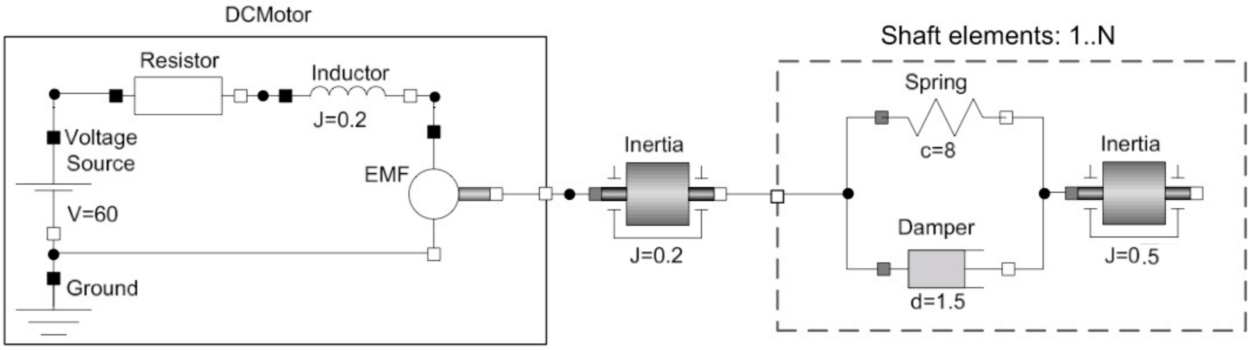
- **Conditional components**
- **For-equations**
- **Redeclare construct**

Part I
What is an EOO language?
Part II
Why MKL?
Part III
Expressiveness, Extensibility and Formalization
Linköpings universitet

HOAM – Example

David Broman
david.broman@liu.se

Example of a mechatronic system with a DC motor and a flexible shaft




```
let MechSys =
  let r1:Rotational in
  let r2:Rotational in
  let r3:Rotational in
  DCMotor r1;
  Inertia 0.2 r1 r2;
  FlexibleShaft 120 r2 r3
```

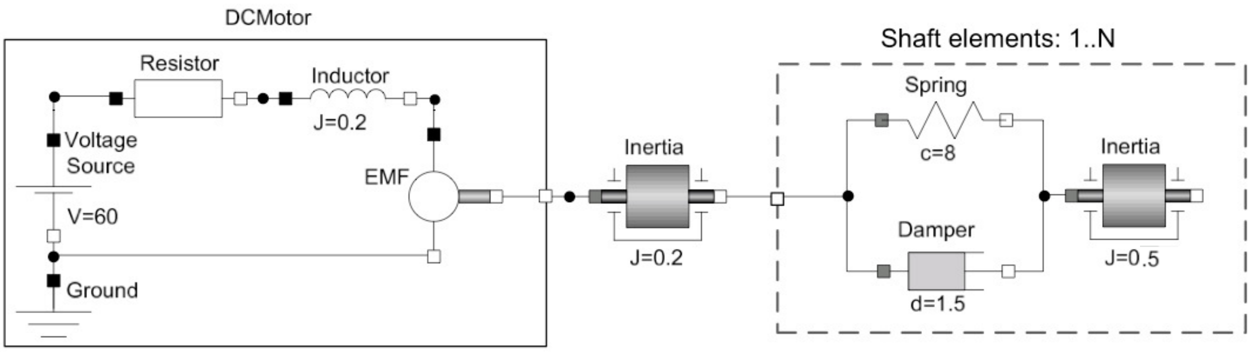
Creates a flexible shaft with 120 shaft elements.

How is this model defined?

HOAM – Example

David Broman
david.broman@liu.se

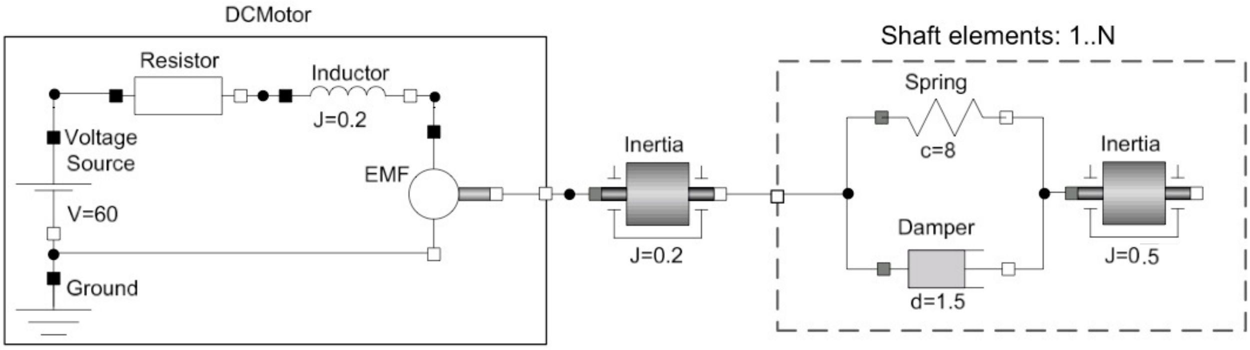
Example of a mechatronic system with a DC motor and a flexible shaft




```
let Inductor L:Real -> p:Electrical -> n:Electrical -> Equations=
  let i:Current in
  let v:Voltage in
  ElectricalBranch i v p n;
  L *. (der i) = v
```

HOAM – Example

Example of a mechatronic system with a DC motor and a flexible shaft

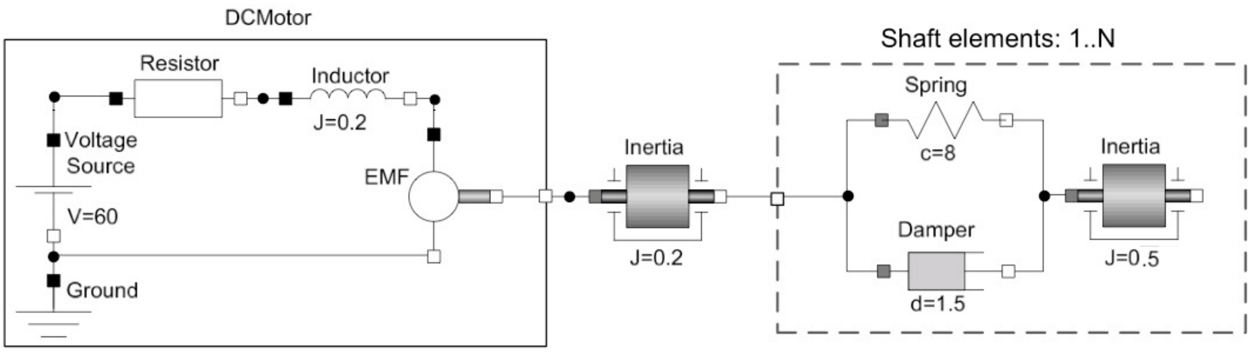



```
let ShaftElement flangeA:Rotational -> flangeB:Rotational ->
                 Equations =
   let r1:Rotational in
   Spring 8. flangeA r1;
   Damper 1.5 flangeA r1;
   Inertia 0.5 r1 flangeB
```

One shaft element is created by standard components.

HOAM – Example

Example of a mechatronic system with a DC motor and a flexible shaft




```
let FlexibleShaft n:Int -> flangeA:Rotational ->
                  flangeB:Rotational -> Equations =
  if n == 1 then
    ShaftElement flangeA flangeB
  else
    let r1:Rotational in
    ShaftElement flangeA r1;
    FlexibleShaft (n-1) r1 flangeB
```

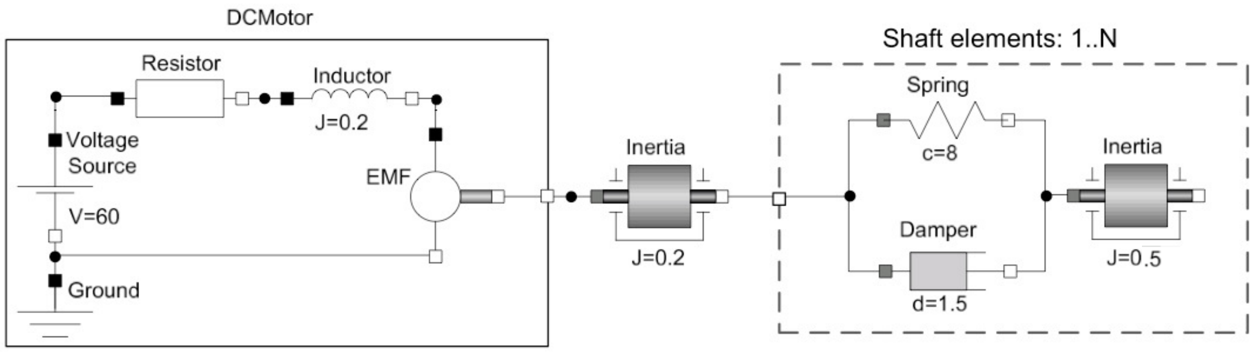
The flexible shaft is recursively defined by creating ShaftElements.

The recursion terminates after n steps (in the example 120 steps)

HOAM – Example

David Broman
david.broman@liu.se

Example of a mechatronic system with a DC motor and a flexible shaft



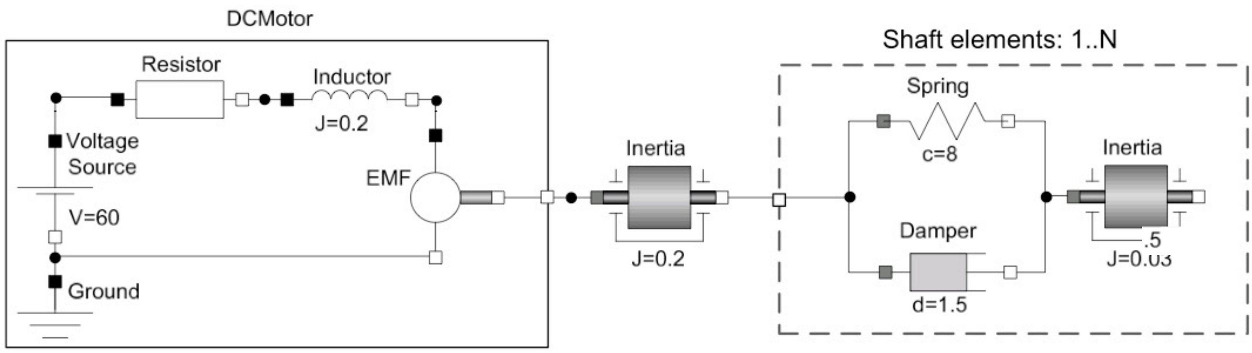

```
let MechSys =
  let r1:Rotational in
  let r2:Rotational in
  let r3:Rotational in
  DCMotor r1;
  Inertia 0.2 r1 r2;
  FlexibleShaft 120 r2 r3
```

Do we always need a special recursive model?

HOAM – Example

David Broman
david.broman@liu.se

Example of a mechatronic system with a DC motor and a flexible shaft

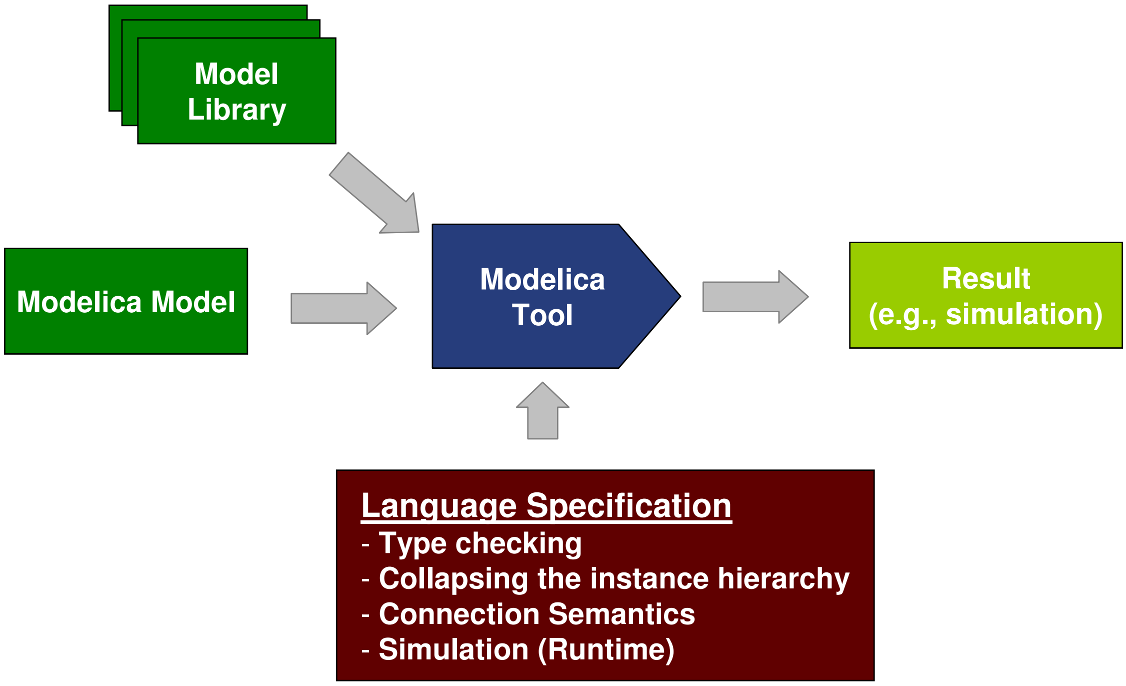



```
let MechSys =
  let r1:Rotational in
  let r2:Rotational in
  let r3:Rotational in
  DCMotor r1;
  Inertia 0.2 r1 r2;
 (serializeRotational 120 ShaftElement) r2 r3
```

Higher-order function that can compose any mechanical component in series

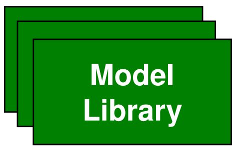
Modelica Environment

David Broman
david.broman@liu.se



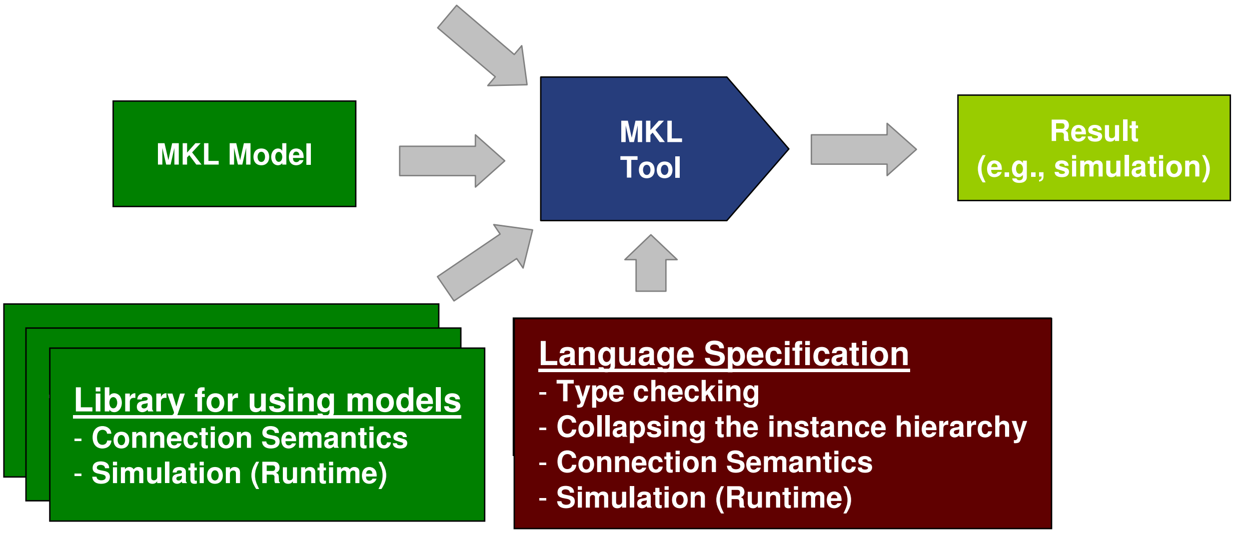
MKL Environment

David Broman
david.broman@liu.se



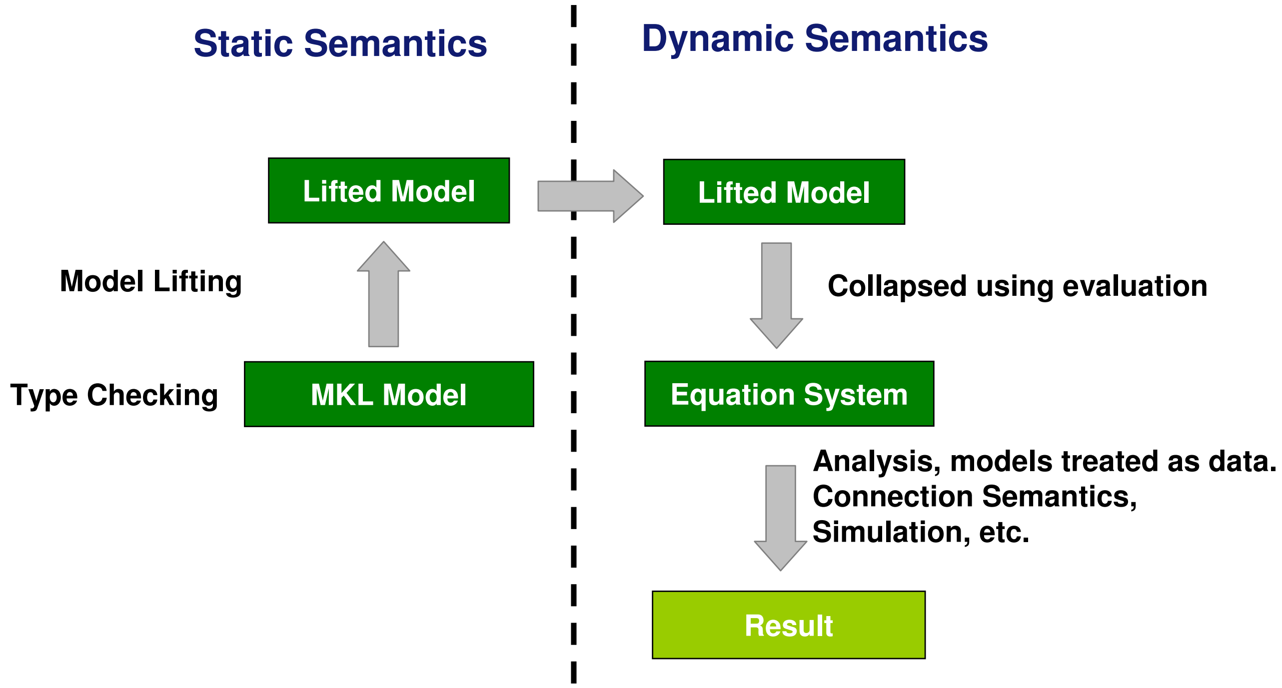
Benefits

- Tool vendors – no need to update tool after lib ext.
- Library developer - less dependent on tool vendors
- A model behaves the same way in different tools



Intensional Analysis and Model Lifting

David Broman
david.broman@liu.se



Intensional Analysis – an Example

David Broman
david.broman@liu.se

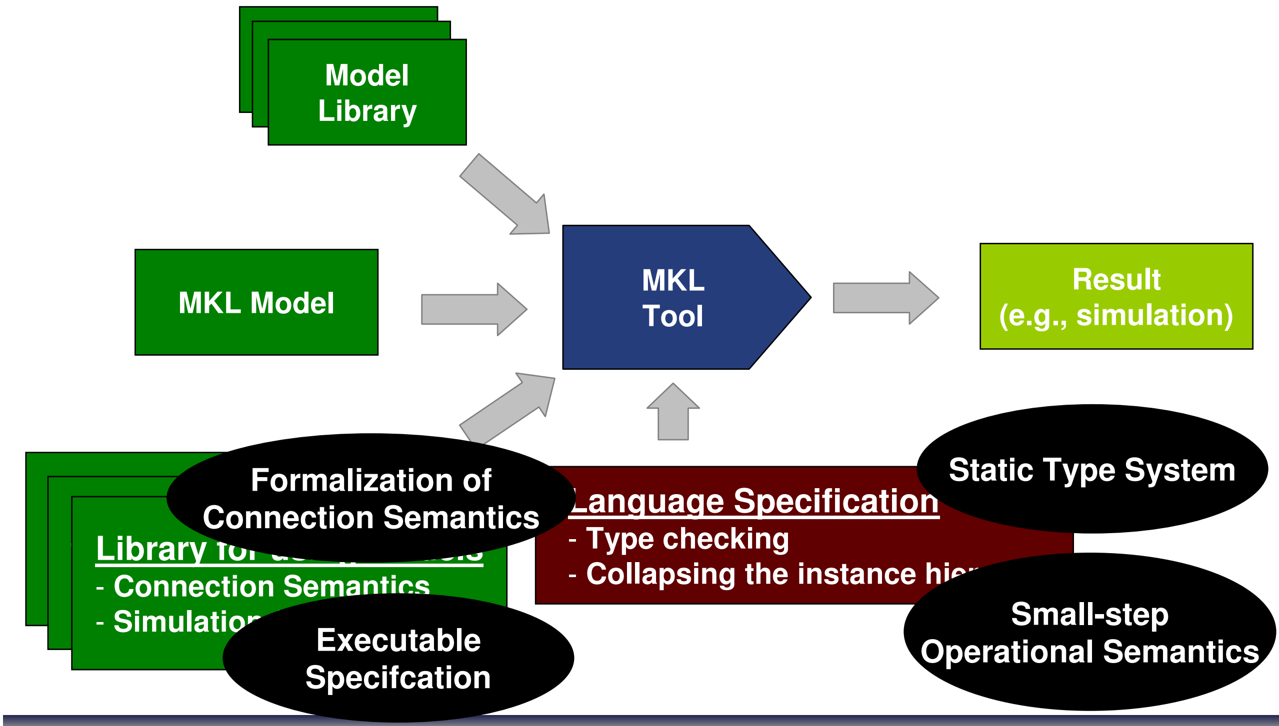
```
type InitValMap = (<Real> => Real)
```

Computing the mapping from unknowns to initial values

```
let initValues eqs:Equations -> InitValMap =
  let get eqs:Equations -> acc:InitValMap -> InitValMap =
    match eqs with
    | e1 ; e2 -> get e2 (get e1 acc)
    | Init x (val v:Real) -> Map.add x v acc
    | _ -> acc
  in get eqs (Map.empty)
```

Formalization of Semantics

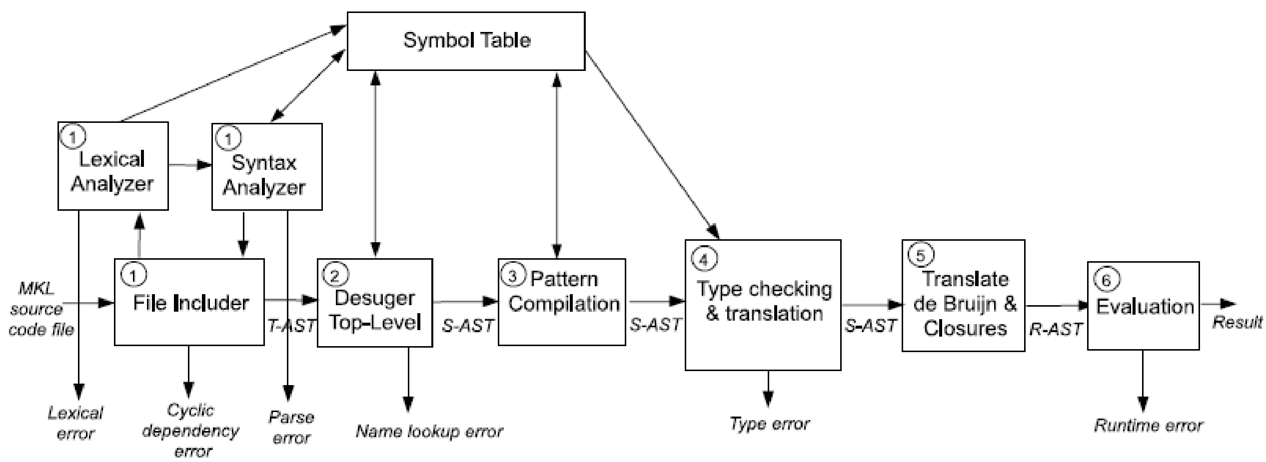
David Broman
david.broman@liu.se



How do we verify our solution?

David Broman
david.broman@liu.se

Prototype Implementation



Syntax and Dynamic Semantics

David Broman
david.broman@liu.se

Abstract Syntax (core of MKL)

Variables	$x, y \in \mathbb{X}$		
Unknowns	$u \in \mathbb{U}$		
Constants	$c \in \mathbb{C}$		
Expressions	e	$::=$	$x \mid \lambda x{:}\tau.e \mid e\ e \mid c \mid$ $u{:}\tau \mid \nu(\tau) \mid e \,@\, e \mid \texttt{val}\ e{:}\tau \mid \texttt{decon}(e, d, e, e)$
Deconstruct patterns	d	$::=$	$\texttt{uk}{:}\tau \mid x \,@\, x \mid \texttt{val}\ x{:}\tau$
Values	v	$::=$	$\lambda x{:}\tau.e \mid c \mid u{:}\tau \mid v \,@\, v \mid \texttt{val}\ v{:}\tau$
Ground Types	$\gamma \in \mathbb{G}$		
Types	τ	$::=$	$\gamma \mid \tau \to \tau \mid \texttt{<}\tau\texttt{>} \mid \texttt{<>}$

Big-Step Semanitcs (selected rule)

$$\frac{\begin{array}{c} e_1 \mid U_1 \;\Rightarrow\; \lambda x{:}\tau.e_3 \mid U_2 \\ e_2 \mid U_2 \;\Rightarrow\; v_1 \mid U_3 \qquad [x \mapsto v_1]e_3 \mid U_3 \;\Rightarrow\; v_2 \mid U_4 \end{array}}{e_1\ e_2 \mid U_1 \;\Rightarrow\; v_2 \mid U_4} \ \text{(BS-APPABS)}$$

Small-Step and Type System

David Broman
david.broman@liu.se

Small-Step Semantics (selected rules)

Computation Rules $\boxed{e \mid U \longrightarrow e' \mid U'}$

$$(\lambda x{:}\tau_1.e_1)v_1 \mid U \longrightarrow [x \mapsto v_1]e_1 \mid U \ \text{(E-APPABS)} \qquad c_1\ v_1 \mid U \longrightarrow \delta(c_1, v_1) \mid U \ \text{(E-DELTA)}$$

$$\frac{u \notin U}{\nu(\tau_1) \mid U \longrightarrow u{:}\texttt{<}\tau_1\texttt{>} \mid U \cup \{u\}} \ \text{(E-NEWUK)}$$

Congruence Rules $\boxed{e \mid U \longrightarrow e' \mid U'}$

$$\frac{e_1 \mid U \longrightarrow e_1' \mid U'}{e_1\ e_2 \mid U \longrightarrow e_1'\ e_2 \mid U'} \ \text{(E-APP1)} \qquad \frac{e_2 \mid U \longrightarrow e_2' \mid U'}{v_1\ e_2 \mid U \longrightarrow v_1\ e_2' \mid U'} \ \text{(E-APP2)}$$

Type System (selected rule)

$\boxed{\Gamma \vdash_L e \rightsquigarrow e' : \tau}$

$$\frac{\Gamma \vdash_L e_1 \rightsquigarrow e_1' : \tau_{11} \to \tau_{12} \qquad \Gamma \vdash_L e_2 \rightsquigarrow e_2' : \tau_2 \qquad \tau_{11} \sim \tau_2}{\Gamma \vdash_L e_1\ e_2 \rightsquigarrow e_1'\ e_2' : \tau_{12}} \ \text{(L-APP)}$$

Type Safety Proof

David Broman
david.broman@liu.se

Main Lemmas

Lemma 10.5 (Progress)
If $\vdash e : \tau$ *then* $e \in Values$ *or for all* U *there exists* U' *and* e' *such that* $e \mid U \longrightarrow e' \mid U'$.

Lemma 10.8 (Preservation)
If $\Gamma \vdash e : \tau$ *and* $e \mid U \longrightarrow e' \mid U'$ *then* $\Gamma \vdash e' : \tau$.

Conclusions

David Broman
david.broman@liu.se

Extensibility
(Library Approach)

Expressivness
(HOAM)

Formalization
(Operational Semantics)

Modeling Kernel Language (MKL)

Thanks for listening!