Equation-Based Object-Oriented Languages for Acausal Modeling and Simulation

Lecture 12a in EECS 144/244

University of California, Berkeley
November 12, 2013

David Broman

broman@eecs.berkeley.edu

EECS Department
University of California, Berkeley, USA

and

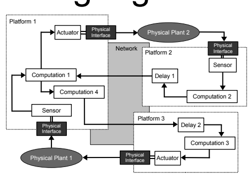
Linköping University, Sweden

Some of the slides are based OSMC tutorials and contributed by
Peter Fritzson (Based on book and lecture nodes), David Broman, Emma Larsdotter Nilsson, Peter Bunus, Adrian Pop, Jan Brugård, Mohsen Torabzadeh-Tari, and Adeel Asghar.

Agenda

Part I

EOO Languages for CPS



Part II

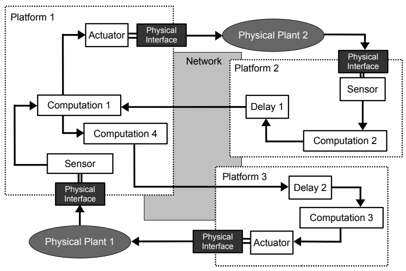
Modelica Overview

MODELICA

Part III

Modelyze – an extensible research language

Part I
EOO Languages for CPS



Cyber-Physical Systems (CPS)

broman@eecs.berkeley.edu

Industrial Robots

Power Plants

Aircraft

Part I
EOO Languages
for CPS

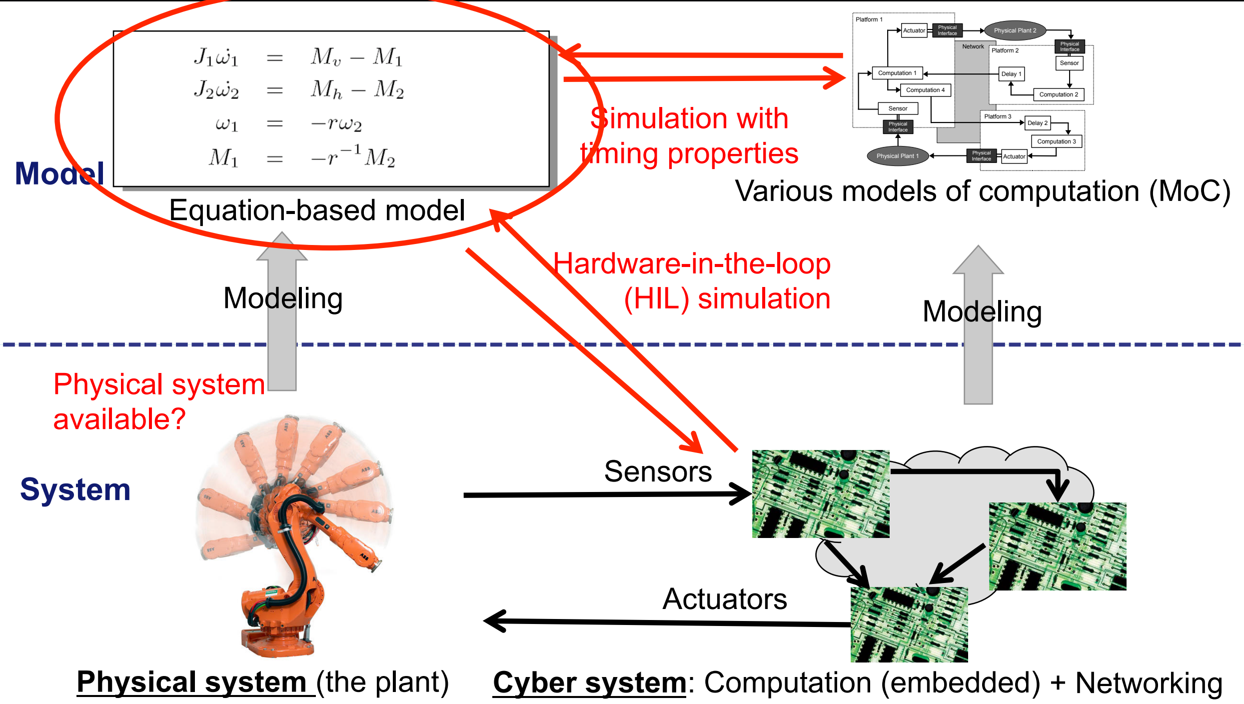
Part II
Modelica
Overview

Part III
Modelyze –
an Extensible Research Language

Linköpings universitet

Modeling and Simulating Cyber-Physical Systems

broman@eecs.berkeley.edu



Equation-Based Object-Oriented (EOO) Languages

broman@eecs.berkeley.edu

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

broman@eecs.berkeley.edu

Domai
Langu

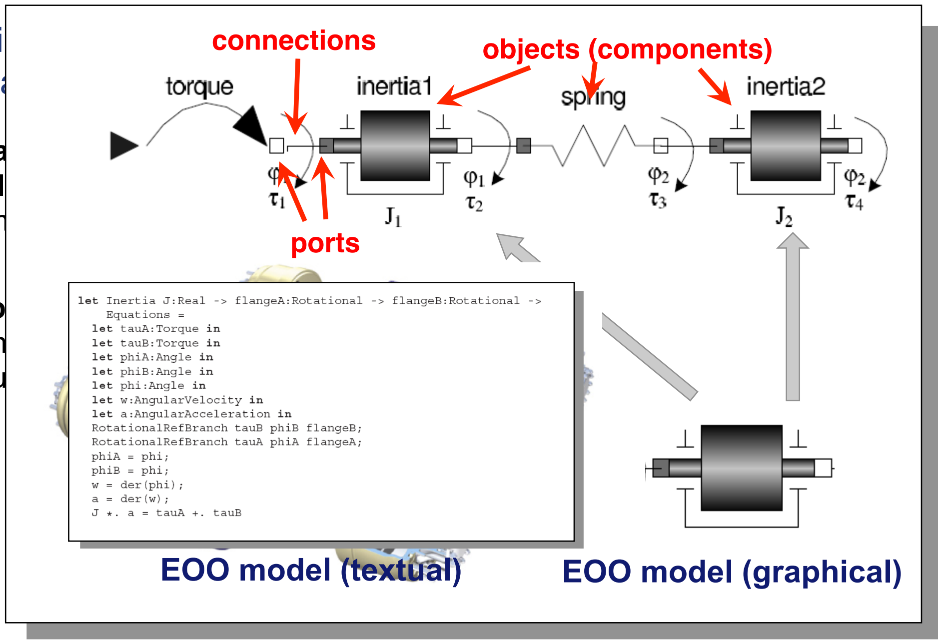
• Prima
Model
systen

• Multip
e.g., m
hydrau

jects

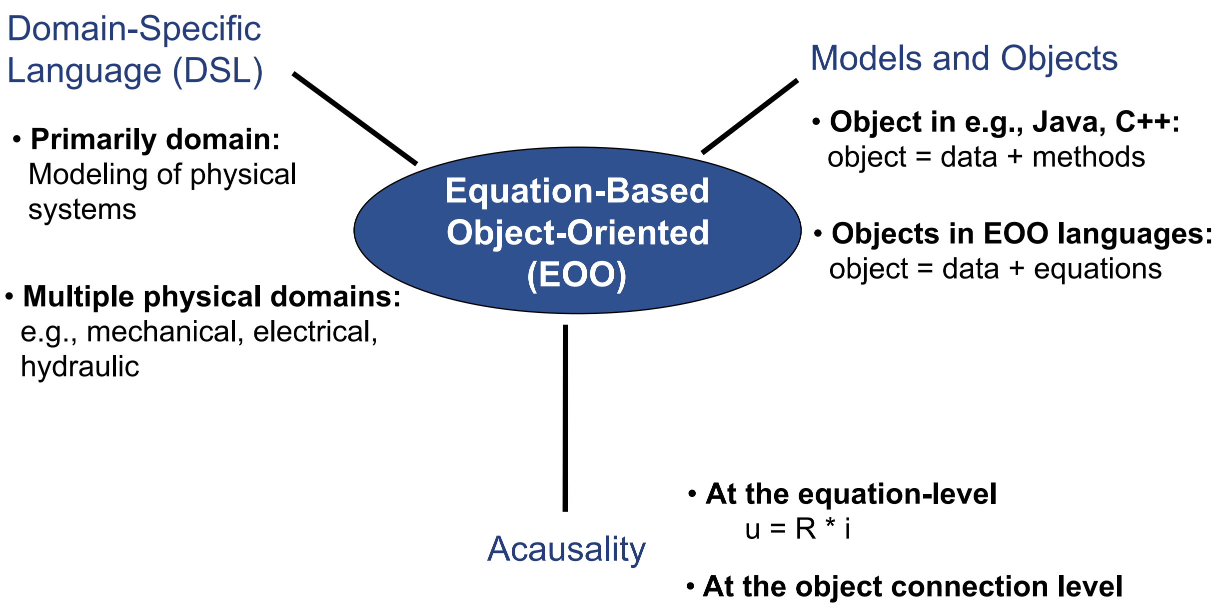
Java, C++:
methods

languages:
equations



Equation-Based Object-Oriented (EOO) Languages

broman@eecs.berkeley.edu



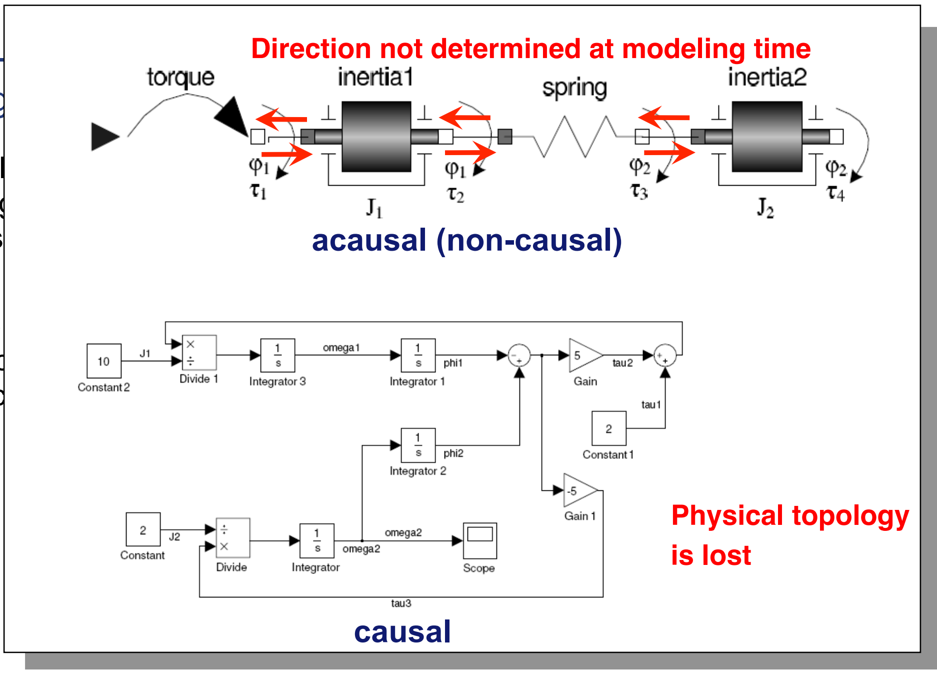
Equation-Based Object-Oriented (EOO) Languages

broman@eecs.berkeley.edu

Domain-
Languag

• **Primari**
Modelin
systems

• **Multiple**
e.g., mec
hydraulic



ects

va, C++:
thods

anguages:
uations

evel

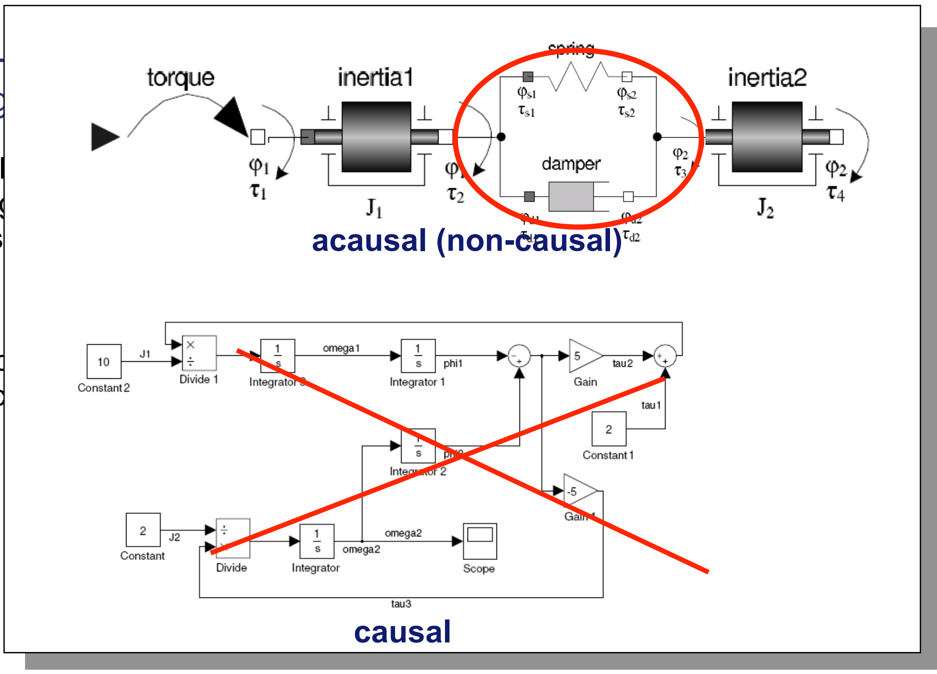
Equation-Based Object-Oriented (EOO) Languages

broman@eecs.berkeley.edu

Domain-
Languag

• **Primari**
Modelin
systems

• **Multiple**
e.g., mec
hydraulic



ects

va, C++:
thods

anguages:
uations

evel

Equation-Based Object-Oriented (EOO) Languages

broman@eecs.berkeley.edu

Domain-Specific Language (DSL)

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

- Modelica
- VHDL-AMS
- gPROMS
- Modelyze
- …

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**

broman@eecs.berkeley.edu

Part II
Modelica Overview

MODELICA

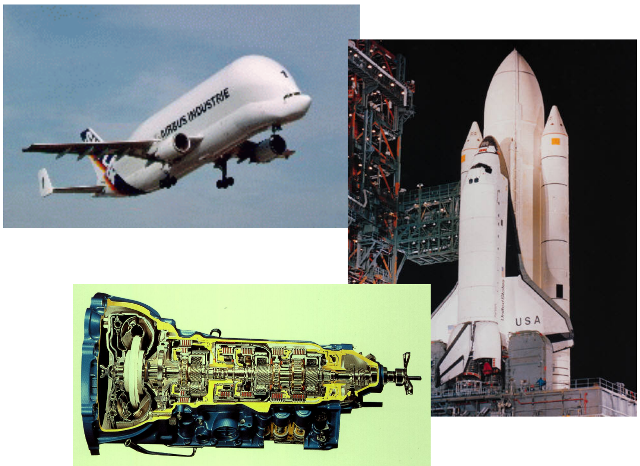
Part I EOO Languages for CPS
Part II Modelica Overview
Part III Modelyze – an Extensible Research Language
Linköpings universitet

What is Modelica?

broman@eecs.berkeley.edu

A language for modeling of complex physical systems

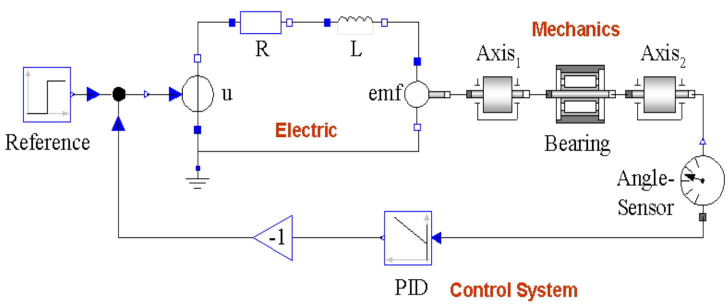
- Robotics
- Automotive
- Aircrafts
- Satellites
- Power plants
- Systems biology



What is Modelica?

broman@eecs.berkeley.edu

A language for modeling of complex physical systems



Primary designed for **simulation**, but there are also other usages of models, e.g. optimization.

What is Modelica?

A language for modeling of complex physical systems

i.e., Modelica is **not** a tool

Free, open language specification:


MODELICA

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

Language Specification

Available at: www.modelica.org

Modelica Tools

Commercial Environments

Dymola by Dassault Systemes

SimulationX by ITI GmbH

LMS Imagine.Lab AMESim by LMS

MapleSim by Maplesoft

MOSILAB by Fraunhofer FIRST

CyModelica by CyDesign Labs

OPTIMICA Studio by Modelon AB

MWorks by Suzhou Tongyuan

Wolfram SystemModeler by Wolfram

Free Environments

OpenModelica supported by OSMC

Jmodelica.org supported by Modelon

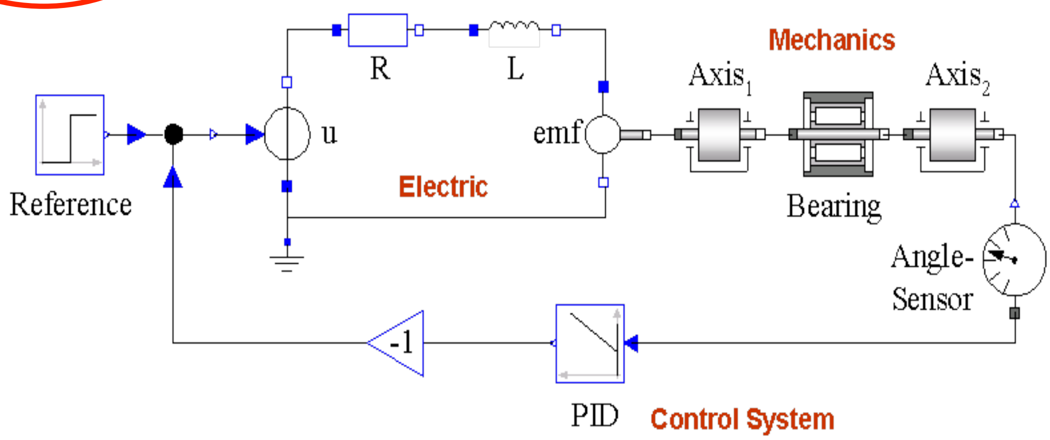
Modelicac (part of Scilab)

SimForge

What is special about Modelica?

broman@eecs.berkeley.edu

Multi-Domain Modeling



What is special about Modelica?

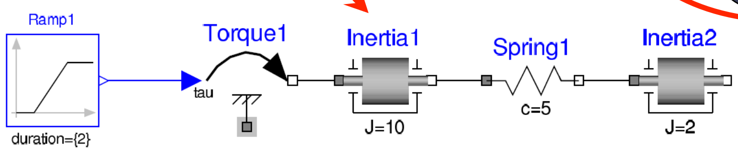
broman@eecs.berkeley.edu

Multi-Domain Modeling

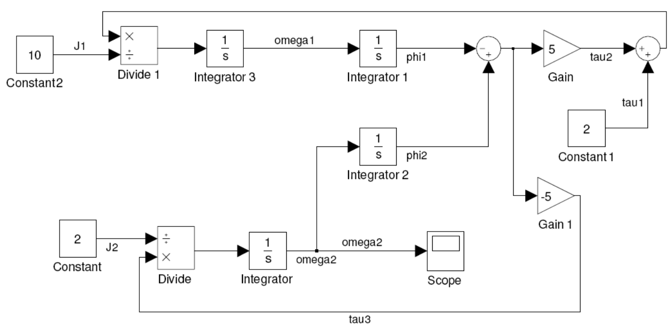
Keeps the physical structure

Visual Acausal Component Modeling

Acausal model (Modelica)



Causal block-based model (Simulink)



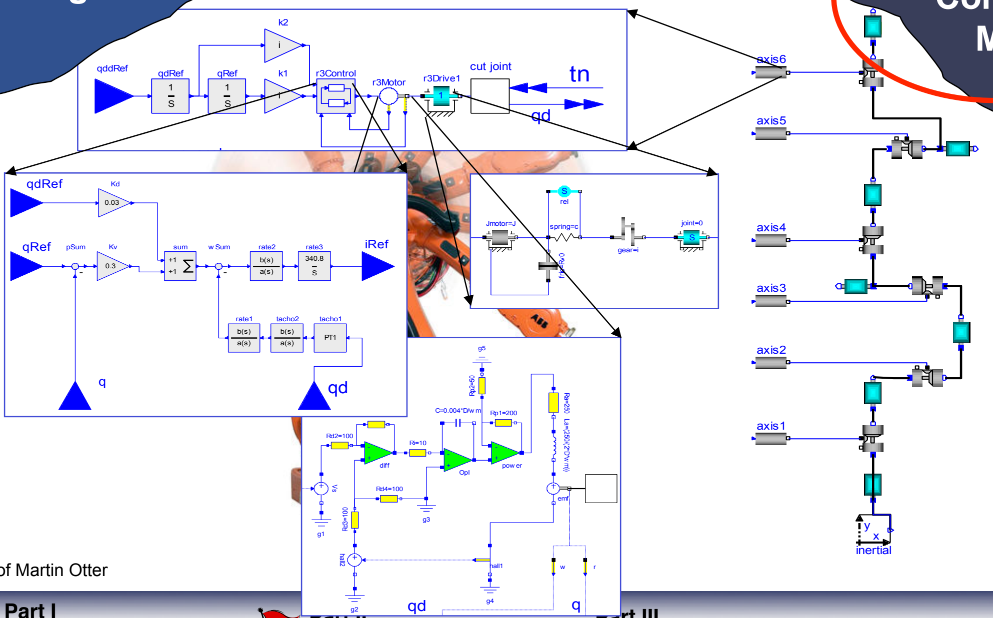
What is special about Modelica?

broman@eecs.berkeley.edu

Multi-Domain Modeling

Hierarchical system modeling

Visual Acausal Component Modeling



Courtesy of Martin Otter

What is special about Modelica?

broman@eecs.berkeley.edu

Multi-Domain Modeling

Visual Acausal Component Modeling

A textual *class-based* language
OO primary used for as a structuring concept

Behaviour described declaratively using

- Differential algebraic equations (DAE) (continuous-time)
- Event triggers (discrete-time)

Variable declarations

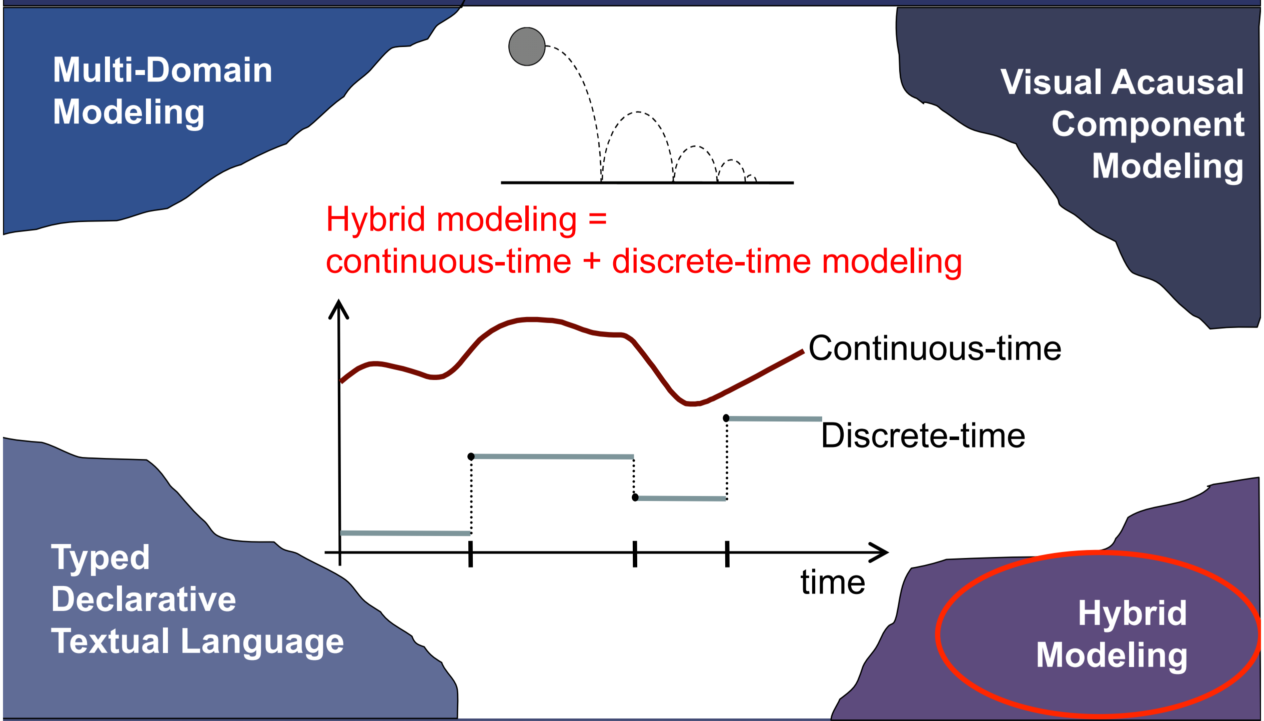
```
class VanDerPol  "Van der Pol oscillator model"
  Real x(start = 1)  "Descriptive string for x”;
  Real y(start = 1)  "y coordinate”;
  parameter Real lambda = 0.3;
equation
  der(x) = y;
  der(y) = -x + lambda*(1 - x*x)*y;
end VanDerPol;
```

Differential equations

Typed Declarative Textual Language

What is special about Modelica?

broman@eecs.berkeley.edu



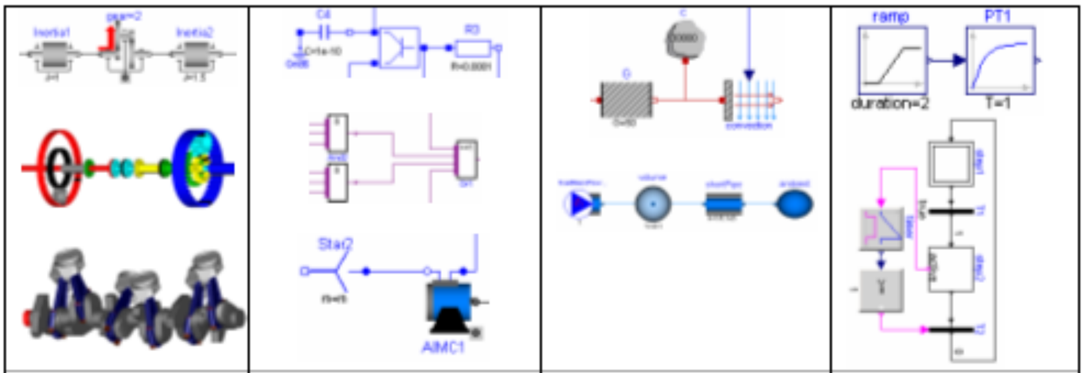
Some Domains

broman@eecs.berkeley.edu

Domain Type	Potential	Flow	Carrier	Modelica Library
Electrical	Voltage	Current	Charge	`Electrical. Analog`
Translational	Position	Force	Linear momentum	`Mechanical. Translational`
Rotational	Angle	Torque	Angular momentum	`Mechanical. Rotational`
Magnetic	Magnetic potential	Magnetic flux rate	Magnetic flux	
Hydraulic	Pressure	Volume flow	Volume	`HyLibLight`
Heat	Temperature	Heat flow	Heat	`HeatFlow1D`
Chemical	Chemical potential	Particle flow	Particles	Under construction
Pneumatic	Presure	Mass flow	Air	`PneuLibLight`

Modelica Standard Library

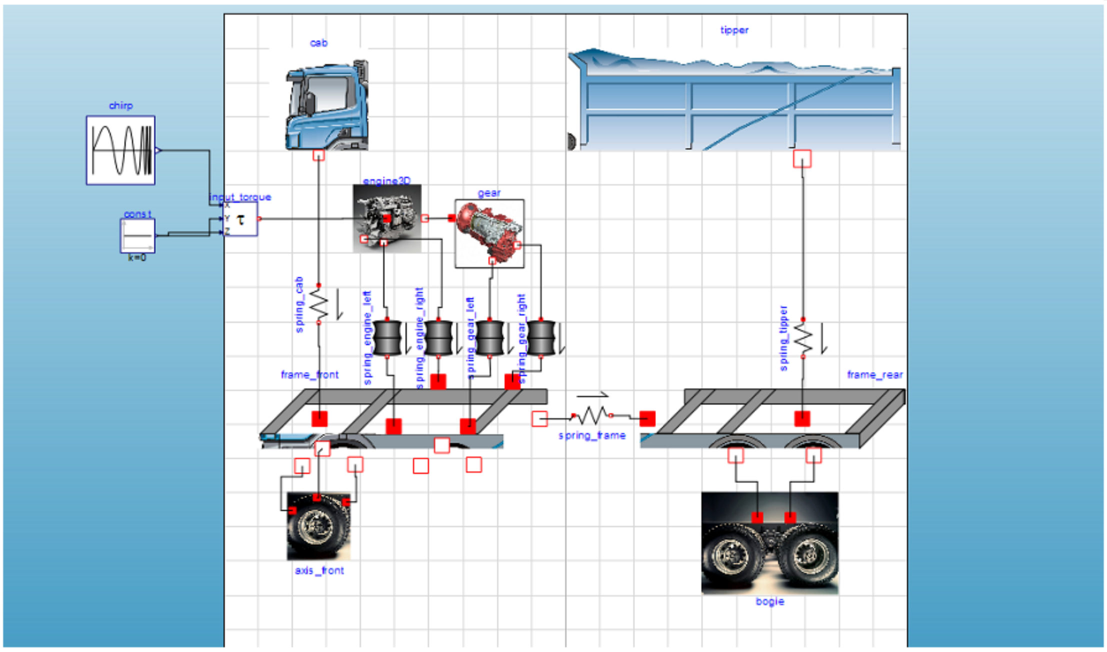
broman@eecs.berkeley.edu



1D and 3D mechanics	analog and digital electrical circuits, electrical machines	heat transfer, fluid systems	cont., discrete logical blocks, state machines

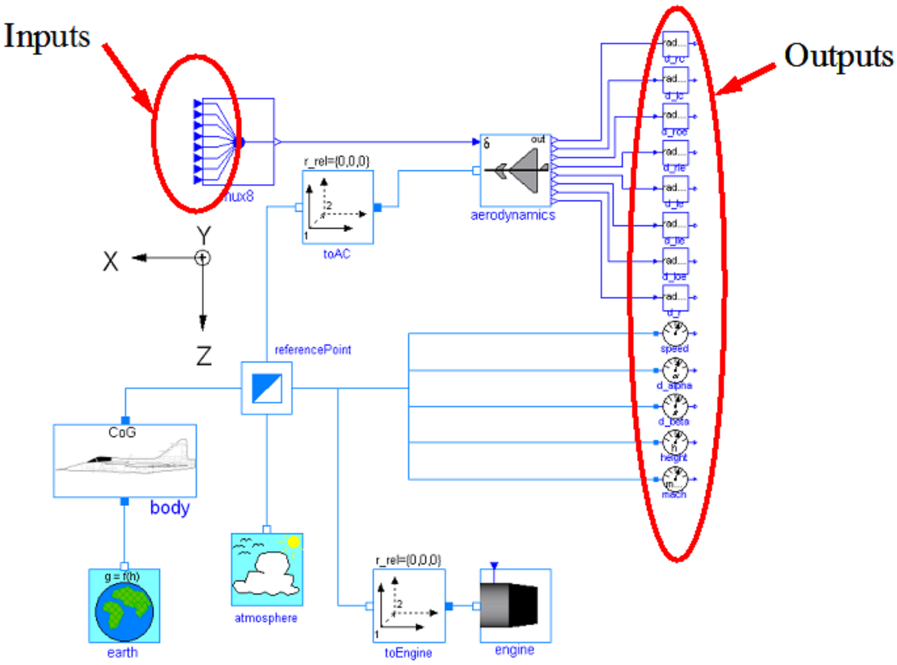
Modelica in Autmotive Industry

broman@eecs.berkeley.edu



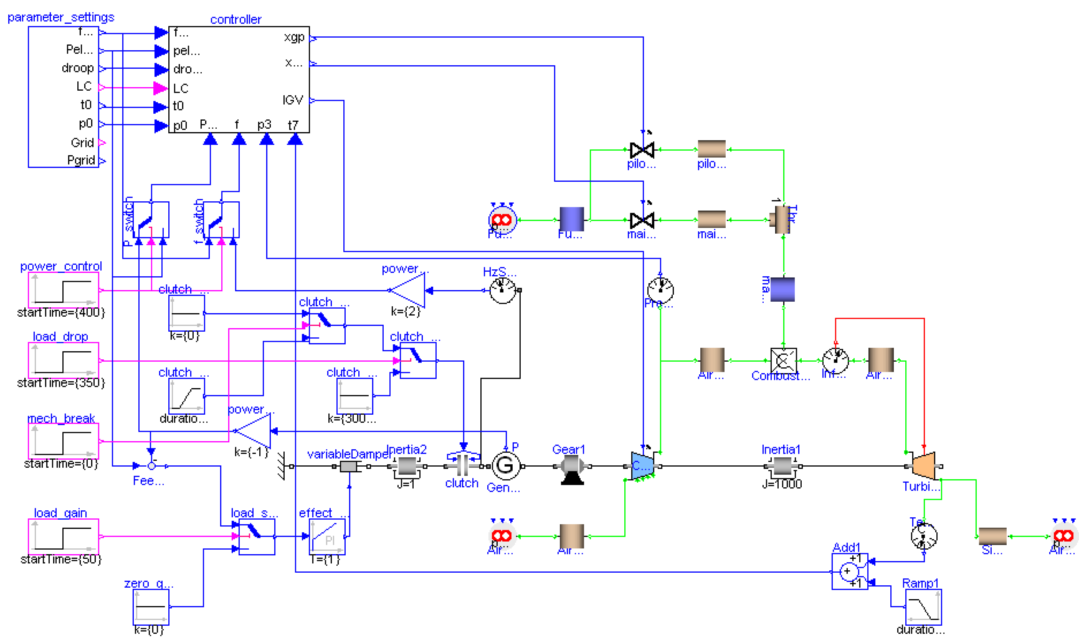
Modelica in Avionics

broman@eecs.berkeley.edu



Modelica in Power Generation GTX Gas Turbine

broman@eecs.berkeley.edu



Courtesy of Siemens Industrial Turbomachinery AB

Developed by MathCore for Siemens

Brief Modelica History

Modelica design group meetings

- First meeting in fall 1996
- International group of people with expert knowledge in both language design and physical modeling
- Industry and academia

Modelica Language Versions

- v1.0 (1997), v2.0 (2002) v.2.2 (2005) v.3.0 (2007) 3.1 (2009) 3.2 (2010), 3.2 revision 1 (2012)

Modelica Association established 2000

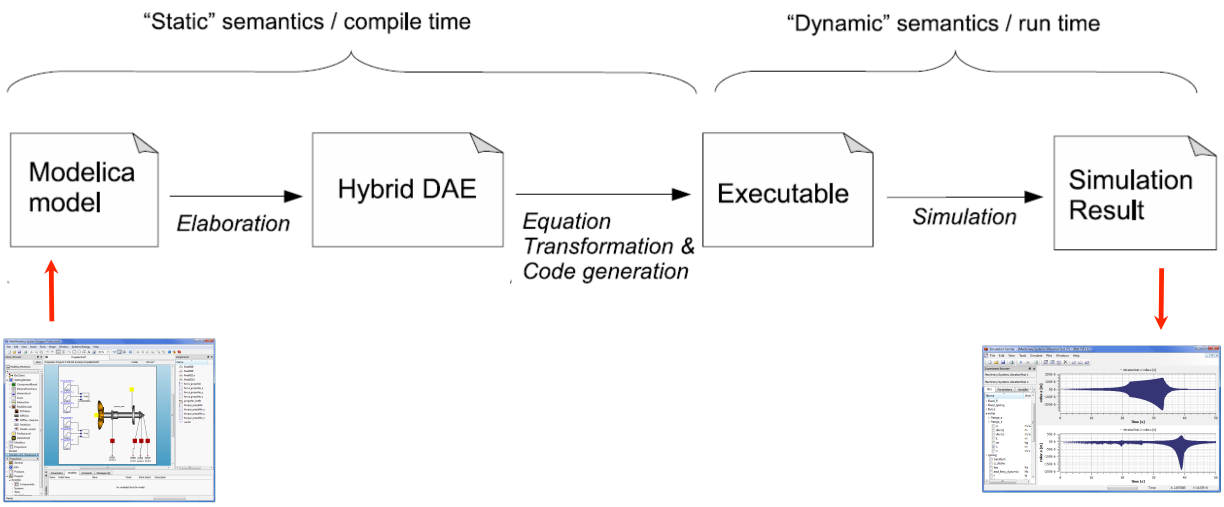
- Open, non-profit organization

Modelica Conferences

- 9 international conferences (2000-2012)

MODELICA

Typical Simulation Process



Simple model - Hello World!

Equation: x' = - x
Initial condition: x(0) = 1

Name of model

Initial condition

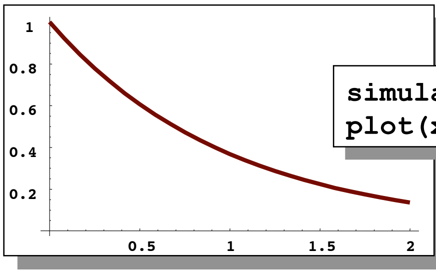
Continuous-time variable

Parameter, constant during simulation

```
model HelloWorld "A simple equation"
  Real x(start=1);
  parameter Real a = -1;
equation
  der(x)= a*x;
end HelloWorld;
```

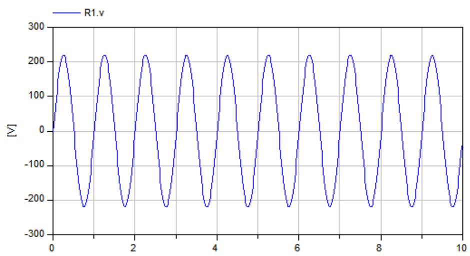
Differential equation

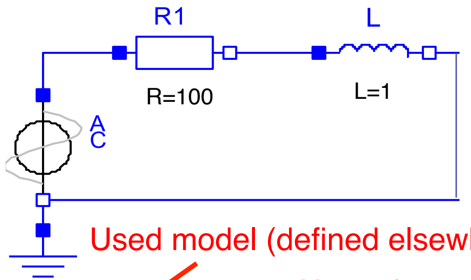
Simulation in OpenModelica environment




```
simulate(HelloWorld, stopTime = 2)
plot(x)
```


Textual and Graphical Models





Used model (defined elsewhere)

Named component = model instance

Modification of parameter value

Connect equations

```
model Circuit
protected
  replaceable Resistor R1(R=10);
  replaceable Inductor L(L=0.1);
  VsourceAC AC;
  Ground G;
equation
  connect(AC.p, R1.p);
  connect(R1.n, L.p);
  connect(L.n, AC.n);
  connect(AC.n, G.p);
end Circuit;
```

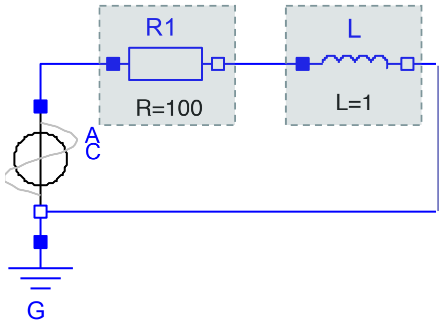
Equations and Inheritance

broman@eecs.berkeley.edu

```
model TwoPin
  Pin p;
  Pin n;
  Real v;
  Real i;
equation
  v = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
end TwoPin;
```

Pin p, n and Reals v and i are copied to the subclass

Equations are copied as well.



Inherits equations and components from **TwoPin**

```
model Resistor
  extends TwoPin;
  Real R = 100;
equation
  R*i = v;
end Resistor;
```

Differential equation

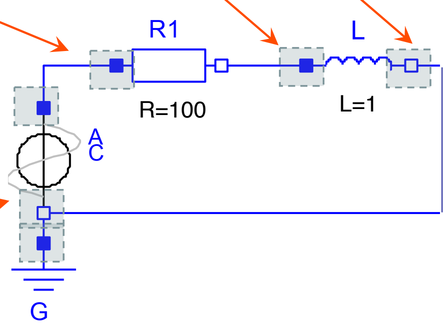
Algebraic equation

```
model Inductor
  extends TwoPin;
  Real L = 1;
equation
  L*der(i) = v;
end Inductor;
```


Connectors (Ports)

broman@eecs.berkeley.edu

```
model TwoPin
  Pin p;
  Pin n;
  Real v;
  Real i;
equation
  v = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
end TwoPin;
```



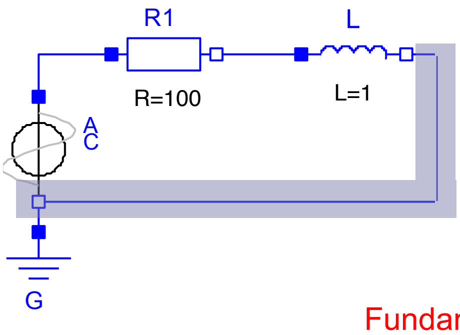
Connectors are instances of a connector class.

```
connector Pin
  Real v;
  flow Real i;
end Pin;
```

Connections and Flow Variables

broman@eecs.berkeley.edu

```
model Circuit
protected
  replaceable Resistor R1(R=10);
  replaceable Inductor L(L=0.1);
  VsourceAC AC;
  Ground G;
equation
  connect(AC.p, R1.p);
  connect(R1.n, L.p);
  connect(L.n, AC.n);
  connect(AC.n, G.p);
end Circuit;
```




```
connector Pin
  Real v;
  flow Real i;
end Pin;
```

Equations from potential variables:

```
L.n.v = AC.n.v
AC.n.v = G.p.v
```

Equation from flow variables:

```
L.n.i + AC.n.i + G.p.i = 0
```

Fundamental concept making acausal modeling work (simplified)

The resulting equation system is an DAE.

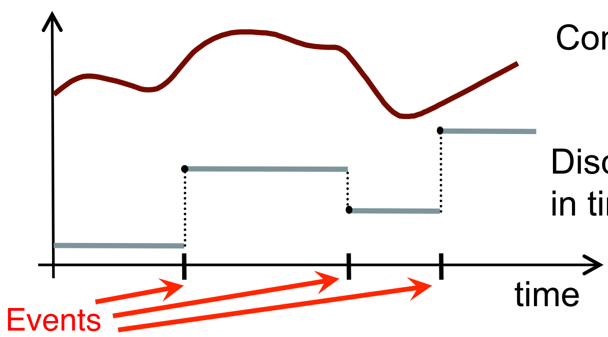
(differential-algebraic equations).

Hybrid Modeling

broman@eecs.berkeley.edu

Hybrid modeling = continuous-time + discrete changes (events)

(Using Modelica terminology)



Continuous-time

```
Real x;
Voltage v;
Current i;
```

Discrete changes in time

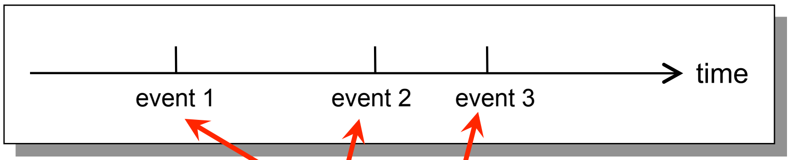
```
discrete Real x;
Integer i;
Boolean b;
```

Event creation – when

broman@eecs.berkeley.edu

when-equations

when <*conditions*> **then**
<*equations*>
end when;



Equations only active at event times

Time event

```
when time >= 10.0 then
  ...
end when;
```

Only dependent on time, can be scheduled in advance

State event

```
when sin(x) > 0.5 then
  ...
end when;
```

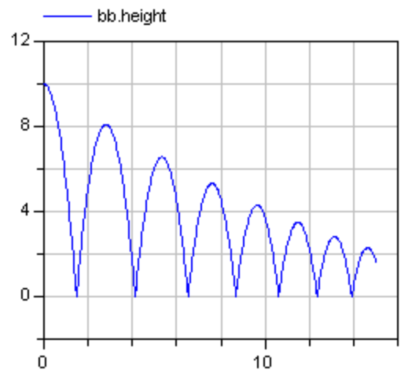
Related to a state. Check for zero-crossing

Reinit - discontinuous changes

broman@eecs.berkeley.edu

The value of a *continuous-time* state variable can be instantaneously changed by a `reinit`-equation within a `when`-equation

```
model BouncingBall "the bouncing ball model"
  parameter Real g=9.81;   //gravitational acc.
  parameter Real c=0.90;   //elasticity constant
  Real height(start=10),velocity(start=0);
equation
  der(height) = velocity;
  der(velocity)=-g;
  when height<0 then
    reinit(velocity, -c*velocity);
  end when;
end BouncingBall;
```



Initial conditions

Reinit "assigns" continuous-time variable `velocity` a new value

Modelica – large and complex

We have just “scratched on the surface of the language”

Examples of the features which has not been covered

- Functions and algorithm sections
- Arrays and matrices
- Inner / outer variables (lookup in instance hierarchy)
- Annotations
- Loop constructs
- Partial classes
- Packages, blocks...

And much more...

Part III

Modelyze –
an extensible research language

What is Modelyze?

broman@eecs.berkeley.edu

Modelyze
(MODEL and analYZE)

(Broman & Siek, 2012)

Purpose: Research language – address the expressiveness and analyzability problem by making the language <u>extensible</u>

Small, simple, host language for embedding domain-specific languages (DSL) of different models of computation (MoC)

Key aspect: Both the DSL and models in the DSL are defined in Modelyze

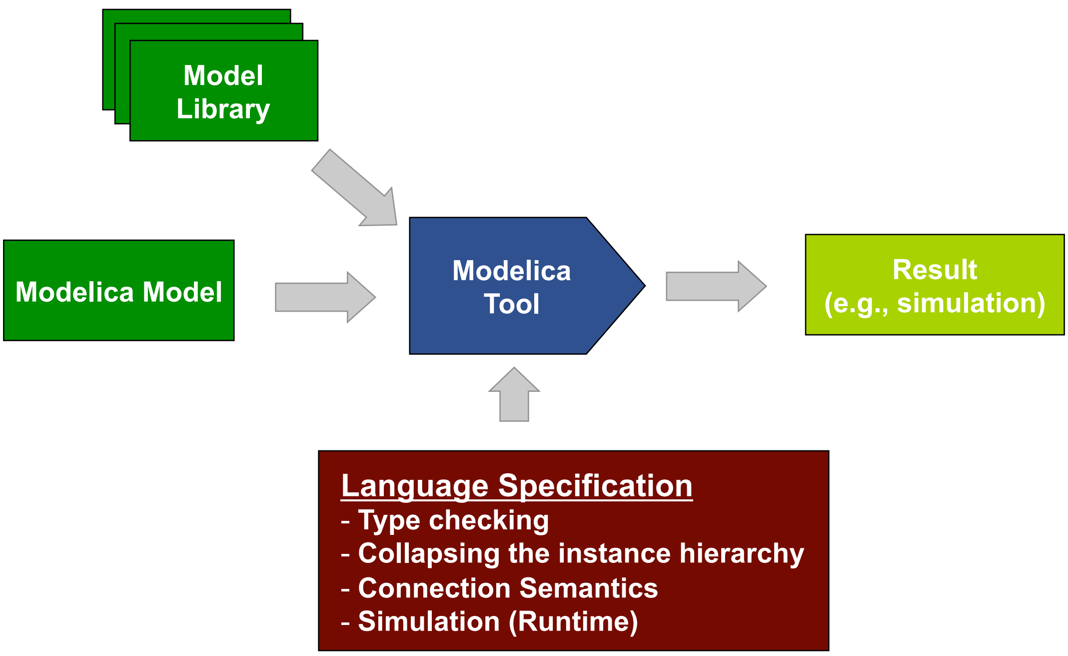
Gradually typed functional language (call-by-value)

Formal semantics for a core of the language. Proven type soundness for the core.

Prototype implementation (interpreter). Evaluated for series of equation-based DSLs.

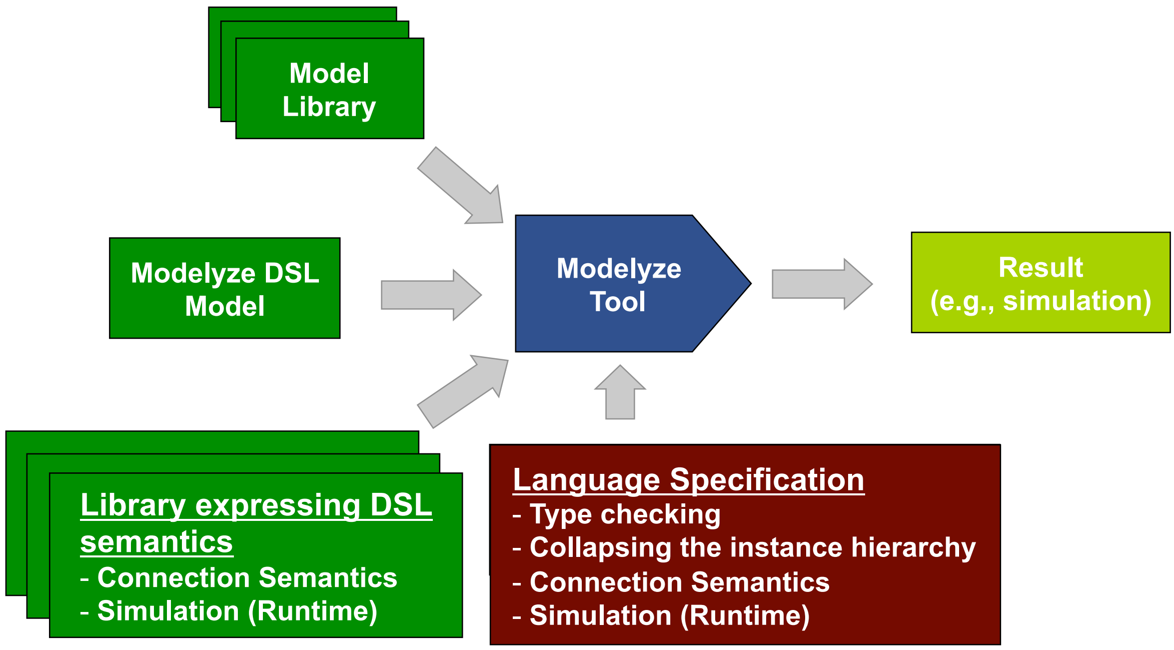
Modelica Environment

broman@eecs.berkeley.edu



Modelyze Environment

broman@eecs.berkeley.edu



References and Further Reading

broman@eecs.berkeley.edu

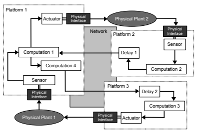
- Sven Erik Mattsson, Hilding Elmqvist, and Martin Otter. Physical System Modeling with Modelica. Control Engineering Practice 6. Pages 501-510, 1998.
- Peter Fritzson. Principles of Object-Oriented Modeling and Simulation with Modelica 2.1. Wiley-IEEE Press, New York, 2004.
- Peter Fritzson, Peter Aronsson, Håkan Lundvall, Kaj Nyström, Adrian Pop, Levon Saldamli, and David Broman The OpenModelica Modeling, Simulation, and Software Development Environment. Simulation News Europe. Issue 44, Pages 8-16, ARGESIM, 2005
- David Broman, Peter Fritzson, and Sébastien Furic. Types in the Modelica Language. In Proceedings of the Fifth International Modelica Conference, pages 303-315, Vienna, Austria, 2006.
- David Broman and Jeremy G. Siek. Modelyze: a gradually typed host language for embedding equation-based modeling languages. Technical Report UCB/EECS-2012-173, EECS Department, University of California, Berkeley, June 2012.

See http://www.modelica.org for more information on Modelica, including the latest language specification.

Part I EOO Languages for CPS | **Part II** Modelica Overview | **Part III** Modelyze – an Extensible Research Language
Linköpings universitet

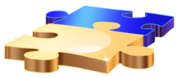
Conclusions

broman@eecs.berkeley.edu



EOO Languages are particularly good for physical modeling because of their acausal capability

Modelica is the current state-of-the-art EOO language. The fundamental formalism is DAEs.



Modelyze is an extensible research language for embedding equation-based languages.