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. Copyright © Open Source Modelica Consortium.

Agenda

Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
Modelyze – an extensible research language
Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
Modelyze – an Extensible Research Language

Cyber-Physical Systems (CPS)
Modeling and Simulating Cyber-Physical Systems

Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
Modelyze – an Extensible Research Language

Physical system (the plant)  Cyber system: Computation (embedded) + Networking

Physical system available?

Equation-Based Object-Oriented (EOO) Languages

Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
Modelyze – an Extensible Research Language

Equation-Based Object-Oriented (EOO) Languages

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
Equation-Based Object-Oriented (EOO) Languages

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: \( u = R \cdot i \)
- At the object connection level

Part I
EOO Languages
for CPS

Part II
Modelica
Overview

Part III
Modelyze – an Extensible Research Language
Domain-Specific Languages

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

Equation-Based Object-Oriented (EOO) Languages

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

Acausality

- At the equation-level: \( u = R \times i \)
- At the object connection level: acausal (non-causal)

Physical topology is lost
Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
Modelyze – an Extensible Research Language
What is Modelica?

A language for modeling of complex physical systems

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

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:

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?

**Multi-Domain Modeling**

![Multi-Domain Modeling Diagram]

**What is special about Modelica?**

- **Multi-Domain Modeling**
- **Keeps the physical structure**
- **Visual Acausal Component Modeling**

**Acausal model** (Modelica)

**Causal block-based model** (Simulink)
What is special about Modelica?

Multi-Domain 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)

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;

Typed
Declarative
Textual Language

Variable
declarations

Differential equations
What is special about Modelica?

Multi-Domain Modeling

Visual Acausal Component Modeling

Typed Declarative Textual Language

Hybrid modeling = continuous-time + discrete-time modeling

Continuous-time

Discrete-time

time

Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
Modelyze – an Extensible Research Language

broman@eecs.berkeley.edu

What is special about Modelica?

Multi-Domain Modeling

Visual Acausal Component Modeling

Typed Declarative Textual Language

Hybrid modeling = continuous-time + discrete-time modeling

Continuous-time

Discrete-time

time

Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
Modelyze – an Extensible Research Language

broman@eecs.berkeley.edu

Some Domains

<table>
<thead>
<tr>
<th>Domain Type</th>
<th>Potential</th>
<th>Flow</th>
<th>Carrier</th>
<th>Modelica Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrical</td>
<td>Voltage</td>
<td>Current</td>
<td>Charge</td>
<td>Electrical. Analog</td>
</tr>
<tr>
<td>Translational</td>
<td>Position</td>
<td>Force</td>
<td>Linear momentum</td>
<td>Mechanical. Translational</td>
</tr>
<tr>
<td>Rotational</td>
<td>Angle</td>
<td>Torque</td>
<td>Angular momentum</td>
<td>Mechanical. Rotational</td>
</tr>
<tr>
<td>Magnetic</td>
<td>Magnetic potential</td>
<td>Magnetic flux rate</td>
<td>Magnetic flux</td>
<td></td>
</tr>
<tr>
<td>Hydraulic</td>
<td>Pressure</td>
<td>Volume flow</td>
<td>Volume</td>
<td>HyLibLight</td>
</tr>
<tr>
<td>Heat</td>
<td>Temperature</td>
<td>Heat flow</td>
<td>Heat</td>
<td>HeatFlow1D</td>
</tr>
<tr>
<td>Chemical</td>
<td>Chemical potential</td>
<td>Particle flow</td>
<td>Particles</td>
<td>Under construction</td>
</tr>
<tr>
<td>Pneumatic</td>
<td>Pressure</td>
<td>Mass flow</td>
<td>Air</td>
<td>PneuLibLight</td>
</tr>
</tbody>
</table>
Modelica in Avionics

Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
Modelyze – an Extensible Research Language

broman@eecs.berkeley.edu

Modelica in Power Generation
GTX Gas Turbine

Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
Modelyze – an Extensible Research Language

broman@eecs.berkeley.edu

Hello

developed by MathCore for Siemens

Courtesy of Siemens Industrial Turbomachinery AB
**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**

**Modelica Association established 2000**
- Open, non-profit organization

**Modelica Conferences**
- 9 international conferences (2000-2012)

---

**Typical Simulation Process**

```
Modelica model    Hybrid DAE    Executable    Simulation Result
```

- **“Static” semantics / compile time**
  - Elaboration
  - Equation Transformation & Code generation
- **“Dynamic” semantics / run time**
  - Simulation
Simple model - Hello World!

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

Name of model
Initial condition
Continuous-time variable
Parameter, constant during simulation
Differential equation

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

Simulation in OpenModelica environment

Simulation in OpenModelica environment

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

Textual and Graphical Models

Model: Circuit

```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;
```

Used model (defined elsewhere)
Named component = model instance
Modification of parameter value
Connect equations
Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
Modelyze – an Extensible Research Language

Equations and Inheritance

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.

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

Inherits equations and components from TwoPin

Differential equation

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

Connectors (Ports)

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.
Connections and Flow Variables

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 from potential variables:
\[ \text{L.n.v} = \text{AC.n.v} \]
\[ \text{AC.n.v} = \text{G.p.v} \]

Equation from flow variables:
\[ \text{L.n.i + AC.n.i + G.p.i} = 0 \]

Fundamental concept making acausal modeling work (simplified)

The resulting equation system is an DAE. (differential-algebraic equations)

Hybrid Modeling

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

Continuous-time

Discrete changes in time

Events

Real \( x; \)
Voltage \( v; \)
Current \( i; \)

Discrete Real \( x; \)
Integer \( i; \)
Boolean \( b; \)
Event creation – when

**when-equations**

```plaintext
when <conditions> then
  <equations>
end when;
```

---

**Time event**

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

Only dependent on time, can be scheduled in advance

---

**State event**

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

Related to a state. Check for zero-crossing

---

Reinit - discontinuous changes

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

```plaintext
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;
```

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...
What is Modelyze?

**Purpose**: Research language – address the expressiveness and analyzability problem by making the language extensible

- 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

**Language Specification**
- Type checking
- Collapsing the instance hierarchy
- Connection Semantics
- Simulation (Runtime)

---

Modelica Environment
References and Further Reading

- 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

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

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.