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

Lecture 12a in EECS 144/244
University of California, Berkeley
April 17, 2013

David Broman
broman@eecs.berkeley.edu

EECS Department
University of California, Berkeley, USA
and
Linköping University, Sweden

Some of the slides are based on 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
OpenModelica Demo

Part IV
Modelyze – a research language
Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
OpenModelica Demo

Part IV
Modelyze – a Research language

Cyber-Physical Systems (CPS)
Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
OpenModelica Demo

Part IV
Modelyze – a Research language

broman@eecs.berkeley.edu

Modeling and Simulating Cyber-Physical Systems

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

Various models of computation (MoC)

Simulation with timing properties

Hardware-in-the-loop (HIL) simulation

Modeling

Physical system available?

System

Sensors

Actuators

Equation-Based Object-Oriented (EOO) Languages

broman@eecs.berkeley.edu

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

Various models of computation (MoC)
Equation-Based Object-Oriented (EOO) Languages

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

Acausality
- At the equation-level: \( u = R \times i \)
- At the object connection level
Equation-Based Object-Oriented (EOO) Languages

Domain-Specific Languages

- 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: acausal (non-causal) ↔ causal

Physical topology is lost
Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
OpenModelica Demo

Part IV
Modelyze – a Research language

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)

Equation-Based Object-Oriented (EOO)

Models and Objects

- At the equation-level
  \[ u = R \cdot i \]
- At the object connection level

Part II
Modelica Overview

Acausality

Part II
Modelica Overview

Part II
Modelica Overview
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
- Modelica (part of Scilab)
- SimForge
What is special about Modelica?

Multi-Domain Modeling

Keeps the physical structure

Acausal model (Modelica)

Causal block-based model (Simulink)

Visual Acausal Component Modeling
What is special about Modelica?

**Multi-Domain Modeling**

A textual *class-based* language

**Visual Acausal Component Modeling**

- **Behaviour described declaratively using**
  - Differential algebraic equations (DAE) (continuous-time)
  - Event triggers (discrete-time)

```modelica
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**

Hybrid modeling = continuous-time + discrete-time modeling

** typed Declarative Textual Language**

Continuous-time

Discrete-time

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>

broman@eecs.berkeley.edu
### Modelica Standard Library

<table>
<thead>
<tr>
<th>Part I</th>
<th>Part II</th>
<th>Part III</th>
<th>Part IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOO Languages for CPS</td>
<td>Modelica Overview</td>
<td>OpenModelica Demo</td>
<td>Modelyze – a Research language</td>
</tr>
</tbody>
</table>

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

---

### Modelica in Automotive Industry

![Modelica in Automotive Industry Diagram]

---

<table>
<thead>
<tr>
<th>Part I</th>
<th>Part II</th>
<th>Part III</th>
<th>Part IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOO Languages for CPS</td>
<td>Modelica Overview</td>
<td>OpenModelica Demo</td>
<td>Modelyze – a Research language</td>
</tr>
</tbody>
</table>
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** → **Elaboration** → **Hybrid DAE** → **Executable** → **Simulation** → **Simulation Result**

- “Static” semantics / compile time
- “Dynamic” semantics / run time

**Elaboration**
- Equation Transformation & Code generation

broman@eecs.berkeley.edu
Acausal Modeling

The order of computations is not decided at modeling time

<table>
<thead>
<tr>
<th>Acausal</th>
<th>Causal</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Visual Component Level</strong></td>
<td></td>
</tr>
<tr>
<td>![Visual Component Diagram]</td>
<td></td>
</tr>
<tr>
<td><strong>Equation Level</strong></td>
<td></td>
</tr>
<tr>
<td>A resistor equation:</td>
<td></td>
</tr>
<tr>
<td>( R \times i = v );</td>
<td>( i := v/R; ) | ( v := R \times i; ) | ( R := v/i; )</td>
</tr>
</tbody>
</table>

Simple model - Hello World!

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

Name of model

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

Initial condition

Simulation in OpenModelica environment

```
simulate(HelloWorld, stopTime = 2)
plot(x)
```
General representation of DAEs:

\[
0 = \mathbf{f}(t, \mathbf{\dot{x}}, \mathbf{x}, u(t), p)
\]

- \( t \) time
- \( \mathbf{\dot{x}}(t) \) vector of differentiated state variables
- \( \mathbf{x}(t) \) vector of state variables
- \( y(t) \) vector of algebraic variables
- \( u(t) \) vector of input variables
- \( p \) vector of parameters and/or constants

Typically, the compiler transforms the DAE to an ODE before simulation, sometimes using an index reduction algorithms (more about this in lecture 12b).

**Textual and Graphical Models**

- **Modelica**
- **OpenModelica**
- **Modelyze – a Research language**

**Using Modelica**

```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;
```

**Enables later modification of component**

**Modification of parameter value**

**Named component = model instance**

**Used model (defined elsewhere)**

**Connect equations**
Equations and Inheritance

**Part I**
EOO Languages for CPS

**Part II**
Modelica Overview

**Part III**
OpenModelica Demo

**Part IV**
Modelyze – a Research language

---

**Connectors (Ports)**

**Part I**
EOO Languages for CPS

**Part II**
Modelica Overview

**Part III**
OpenModelica Demo

**Part IV**
Modelyze – a Research language
Connections and Flow Variables

Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
OpenModelica Demo

Part IV
Modelyze – a Research language

```
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:
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)

Models and Equation Generation

```
Positive port
Negative port
Each port in electrical domain:
i – flow variable (current)
v – potential variable (voltage)

Modelica connect-equations
connect(VS.n,G.p)
connect(R.n,G.p)

Port set for a3
{G.p,R.n,VS.n}

Potential equations (generated)
R.n.v = G.p.v
VS.n.v = G.p.v

Sum-to-zero equation (generated)
G.p.i + R.n.i + VS.n.i = 0

Inductor
Resistor

L * der(i) = v
R * i = v
```

```
(a)

VS
C
a1
L
a2
R
a3
G

Behavior equations
```
Abstraction and Composition

(a) and (c) describes the same circuit

(a) and (c) describes the same circuit

Same port (e.g. SC.n) can be seen as:
outside port
inside port

Abstraction and Composition

Hybrid Modeling

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

- A point in time that is instantaneous, i.e., has zero duration
- An event condition so that the event can take place
- A set of variables that are associated with the event
- Some behavior associated with the event, e.g. conditional equations that become active or are deactivated at the event
**Event creation – when**

**when-equations**

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

- **Time event**

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

- **State event**

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

Equations only active at event times

- **Time event**

  Only dependent on time, can be scheduled in advance

- **State event**

  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

- **Initial conditions**

  ![Graph](image)
Modelica – large and complex

broman@eecs.berkeley.edu

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 IV

Open Modelica Demo
Open Source Modelica Consortium (OSMC)
broman@eecs.berkeley.edu

OSMC
• Founded in 2007 in Linköping, Sweden
• Seven founding organizations
• Non-profit, non-governmental organization
• Aim of developing and promoting the OpenModelica project
• Non-commercial and commercial usage under the OSMC public license

Part IV
Modelyze – a Research Language
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)
- Novelty: Typed symbolic expressions
  - 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

**Modelica Environment**

- **Model Library**
- **Modelica Model**
- **Modelica Tool**
- **Result (e.g., simulation)**

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

Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
OpenModelica Demo

Part IV
Modelyze – a Research language

Modelyze DSL Model

Model Library

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

Library for using models
- Connection Semantics
- Simulation (Runtime)

Example DSL embedded in Modelyze

Tension in the string

Pendulum (y position)
String breaks (3.2 s)
Ball is flying freely
Ball bounces on the floor
Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
OpenModelica Demo

Part IV
Modelyze – a Research language

Example DSL embedded in Modelyze

```python
def BreakingPendulum(m:Real, l:Real, angle:Real) = {
    def x,y:Position;
    def Pendulum, BouncingBall:Mode;
    init x (l*sin(angle));
    init y (-l*cos(angle));
    probe("y") = y;

    hybridchart initmode Pendulum {
        mode Pendulum {
            def T:Force;
            probe("T") = T;
            PendulumExt(m, l, T, x, y);
            transition BouncingBall
                when (time >= 3.2) action nothing;
        }
        mode BouncingBall {
            x" = 0;
            -m*g = m*y";
            transition BouncingBall
                when (y <= -4) action (y" <= y" * -0.7);
        }
    }
}
```

Part I
EOO Languages for CPS

Part II
Modelica Overview

Part III
OpenModelica Demo

Part IV
Modelyze – a Research language

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.

OpenModelica is a free Modelica environment that includes both compiler and a graphical modeling IDE.

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

Thank you for listening!
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.