Modelica – An Object-Oriented Language for Physical System Modeling

by

Steven Xu

Feb 19, 2003

Overview

- The Modelica design was initiated by Hilding Elmqvist in Sept. 1996
- Has been designed by the developers of some OO modeling languages
- In Feb 2000, Modelica Association, a non-profit, non-governmental organization, was founded for further development, promotion and application of Modelica languages
- Website: http://www.modelica.org

Overview

- Modelica is a freely available, object-oriented language for modeling of large, complex, and heterogeneous physical system. [1]
- Suited for multi-domain modeling
- Models in Modelica are mathematically described by DAEs (Deferential Algebraic Equations)
- Supports non-causal, hybrid, and hierarchical modeling

Overview

- Library of basic models in different domains
- Supports both high-level modeling by composition and detailed library component modeling by equations

Basic Language Elements [2]

- Basic components: Real, Interger, Boolean and String
- Structured components
- Component arrays, to handle real matrices, arrays of sub-models etc
- Equations and/or algorithms(assignment statements)
- Connections
- Functions

Example: Electrical Types

```
type Time = Real(quantity="Time", unit="s");

type Voltage = Real(quantity="Voltage",
  unit="V");

type Current = Real(quantity="Current",
  unit="A");
```

Classes for reuse of knowledge

- In Modelica, the basic structuring element is a **class**; **model, type, connector, block, function, package, record,** etc are restricted classes
- Modelica supports “interface” for components that have common properties (**partial model**). This facility is similar to inheritance in other OO languages.

Example: a connector

```
connector Pin "pin of an electric component"
  Voltage v "Potential at the pin";
  flow Current i "Current flowing into the pin";
end Pin;
```

- A connection **connect**(`Pin1, Pin2`), connects the two pins such that they form one node
- This implies two equations:

```
    Pin1.v = Pin2.v
    Pin1.i + Pin2.i = 0
```

Example: a partial model

- An electrical port

```
partial model OnePort “Superclass of Components with
two electrical pins p and n"
    Voltage v "Voltage drop between p and n";
    Current i "Current flowing from p to n";
    Pin p;
    Pin n;
equation
    v = p.v - n.v;
    0 = p.i + n.i;
    i = p.i;
end OnePort;
```

Example: a resistor

```
model Resistor "Ideal linear electrical resistor"
  extends OnePort;
  parameter Real R(unit=“Ohm”)

equation
  R*i = v;  “Ohm’s Law”
end Resistor;
```

Example: a capacitor

```
model Capacitor "Ideal electrical Capacitor"
  extends OnePort;
  parameter real C(unit=“F”)

equation
  C*der(v) = i;
end Capacitor;
```

Example: a simple circuit

```
model circuit
   Resistor R1(R=10);
   Capacitor C(C=0.01);
   Resistor R2(R=100);
   Inductor L(L=0.1);
   VsourceAC AC;
   Ground G;
equation
   connect(AC.p, R1.p);
   connect(R1.n, C.p);
   connect(C.n, AC.n);
   connect(R1.p, R2.p);
   connect(R2.n, L.p);
   connect(L.n, C.n);
   connect(AC.n, G.p);
end circuit;
```

Hybrid Models

- Modelica can be use for mixed continuous and discrete models

- Discontinuous Models

  if-then-else expressions allow modeling of phenomena in different operating regions. It supports discontinuities.

  eg. `y = if Time > 100 then a else b`

Hybrid Models

- The actions to be performed at events are specified by a *when*-statement

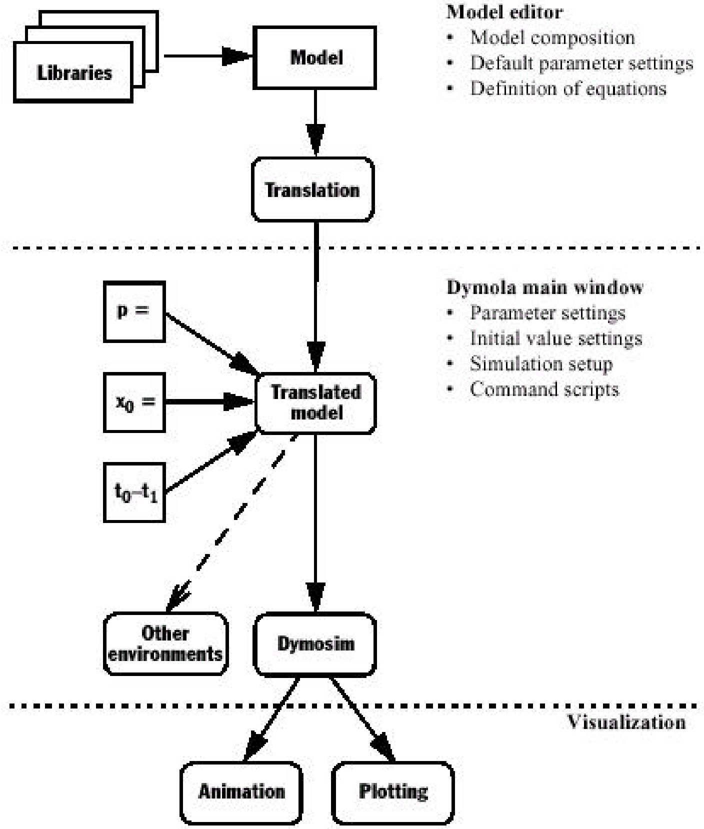
ie.
```
when condition then
     equations
end when
```

- The equations are activated instantaneously when the condition becomes true

Dymola: a commercial tool

- ***Dymola*** is a commercial tool developed by Dynasim AB in Sweden.
- ***Dymola*** has a Modelica translator which is able to perform all necessary symbolic transformations for large systems (> 100 000 equations) as well as for real time applications. A graphical editor for model editing and browsing, as well as a simulation environment are included

How Dymola works…[4]



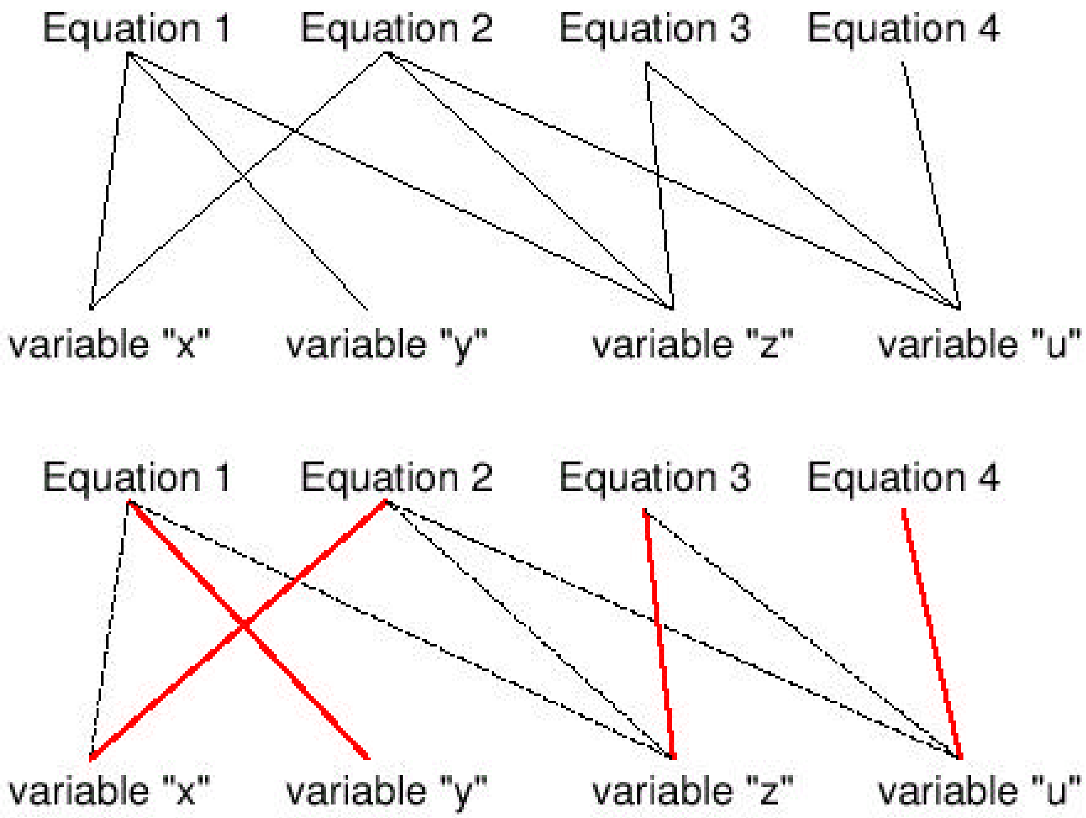
Non-causal models [6]

- Connections in Modelica implies a set of equations which are in a non-causal form

eg.

```
x + y + z = 0          Equation 1
x + 3z + u = 0         Equation 2
z – u - 16 = 0         Equation 3
u - 5 = 0              Equation 4
```

Causality assignment [6]



Causality assignment [6]

```
x + y + z = 0          Equation 1
x + 3z + u = 0         Equation 2
z – u - 16 = 0         Equation 3
u - 5 = 0              Equation 4
```

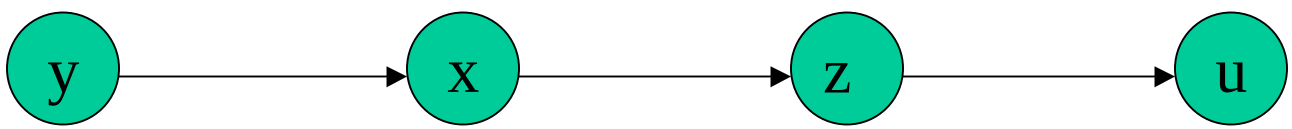
Thus can be rewritten as the following causal form

```
y = -x – z             Equation 1
x = -3z – u            Equation 2
z = u + 16             Equation 3
u = 5                  Equation 4
```

Sorting equations

- Equations need be sorted to allow an algorithm to calculate the variables sequentially.
- Dependency graph of the previous example

  eg. to solve 'z', 'u' has to be calculated first



Sorting result

```
u = 5                  Equation 4
z = u + 16             Equation 3
x = -3z – u            Equation 2
y = -x – z             Equation 1
```

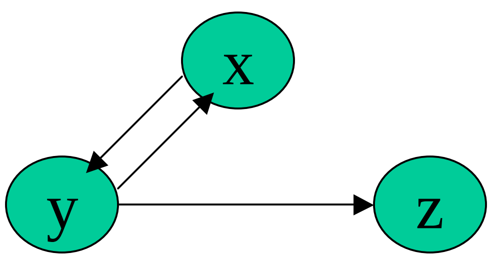
- Now this set of equations can be solved sequentially
- Equations can be sorted using graph algorithms

Equations can not be sorted

- If variables are interdependent, then equations can not be sorted, ie. the dependency graph is no longer acyclic
- For example

```
x = y + 10
y = x + z
z = 5
```

the dependency graph is cyclic

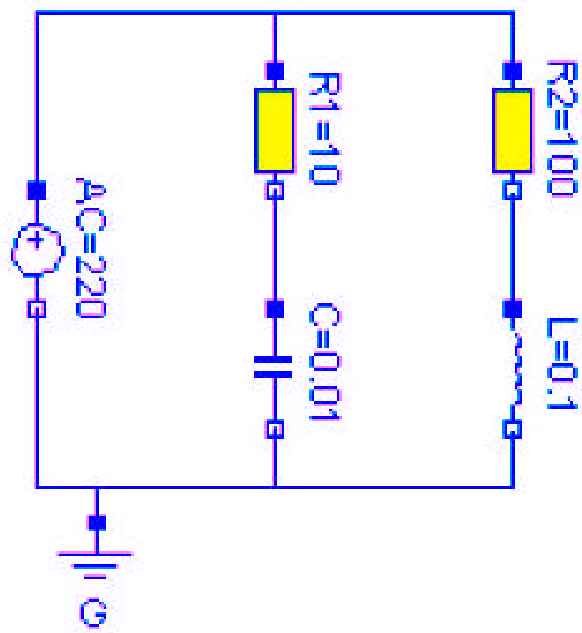


May be solved in another way…

- In some physical systems, many of the equations that appear are constant coefficient linear equations.
- In some cases, this set of equations may be solved, either in numerical or symbolic form
- Cramer’s Rule…

Demo…

- This is the system to be modeled



References

[1] Overview article of Modelica. Available at: http://www.modelica.org/

[2] Modelica Tutorial, version 1.4. Available at: http://www.modelica.org/documents.shtml

[3] EcosimPro Mathematical Algorithms

[4] Dymola User Manual.

[5] Introduction to Physical Modeling with Modelica. Michael Tiller. 2001

[6] Object-Oriented Modeling and Simulation of Physical System. Hans Vangheluwe. 2001