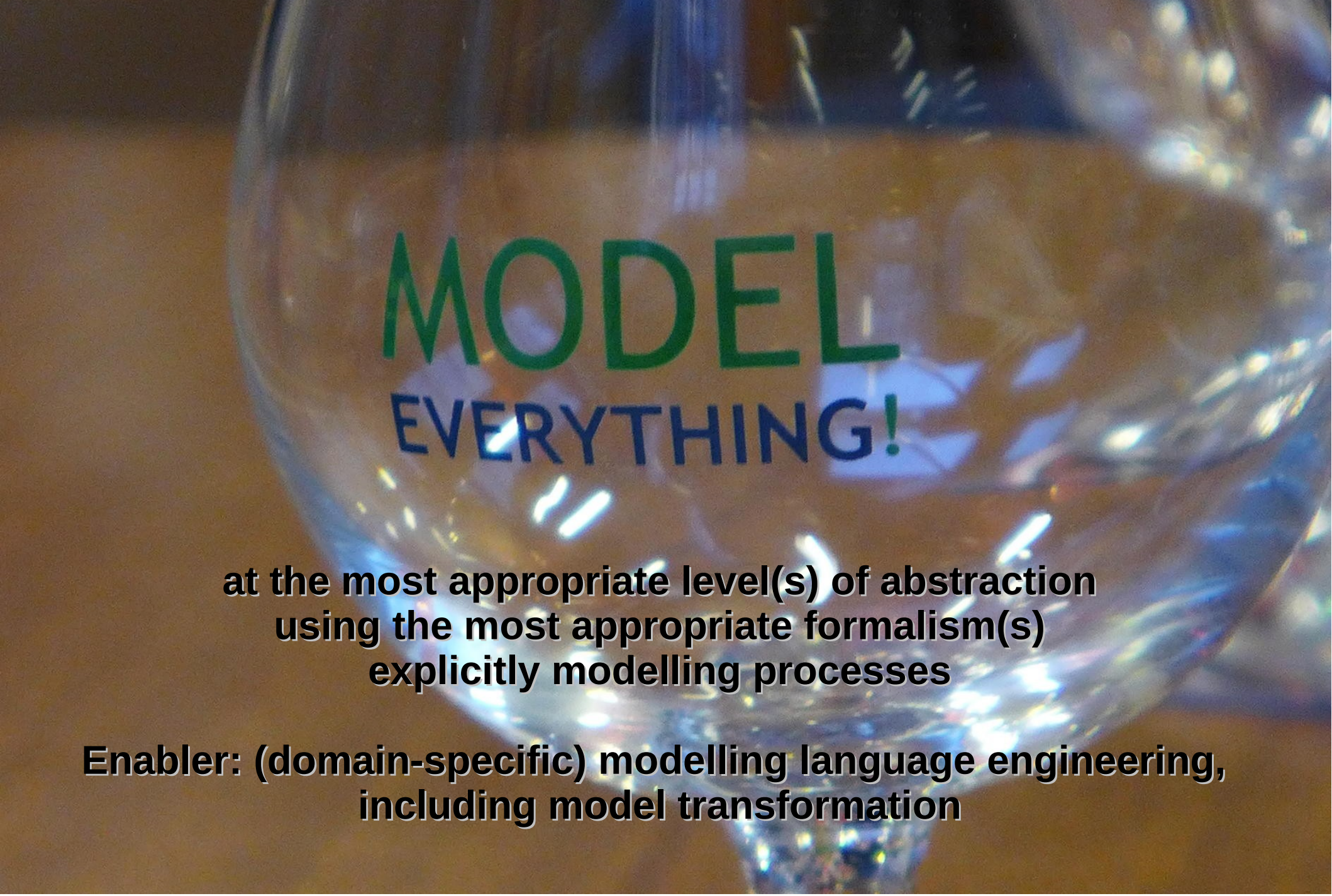
Modelica in Context

Hans Vangheluwe

Modelica tutorial @ MoDELS
Ottawa, Canada
27 September 2015



at the most appropriate level(s) of abstraction
using the most appropriate formalism(s)
explicitly modelling processes

Enabler: (domain-specific) modelling language engineering, including model transformation

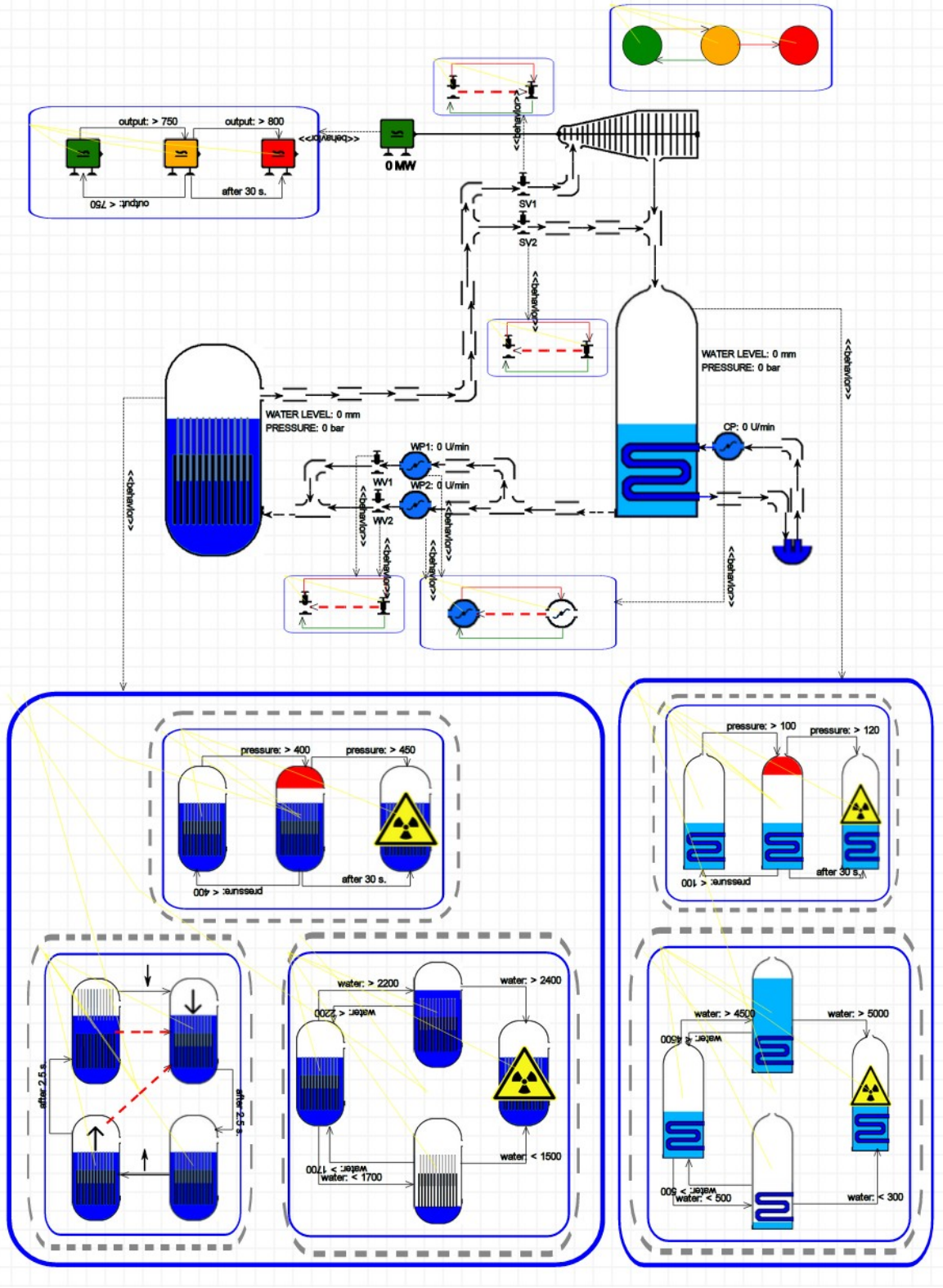
Pieter J. Mosterman and Hans Vangheluwe. Computer Automated Multi-Paradigm Modeling: An Introduction. Simulation: Transactions of the Society for Modeling and Simulation International , 80(9):433- 450, September 2004. Special Issue: Grand Challenges for Modeling and Simulation.

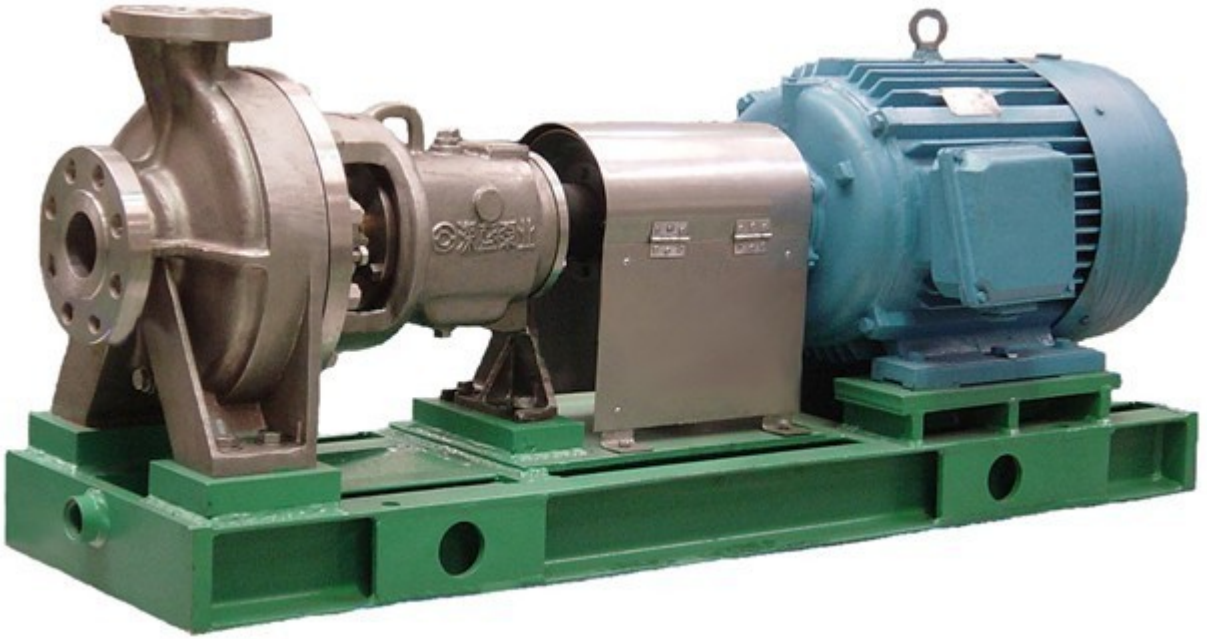
Modelica in Context

- Domain/Problem-Specific
- Laws of Physics
- Power Flow
- a-causal (Mathematical) ← **Modelica**
- Causal Block Diagrams
- Numerical (Discrete) Approximations
- Computer Numerical (Floating Point, Fixed Point)
- As-Fast-As-Possible vs. Real-time
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)

Modelica in Context

- Domain/Problem-Specific
- Laws of Physics
- Power Flow
- a-causal (Mathematical) ← **Modelica**
- Causal Block Diagrams
- Numerical (Discrete) Approximations
- Computer Numerical (Floating Point, Fixed Point)
- As-Fast-As-Possible vs. Real-time
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)


output: > 750
output: > 800
output: < 750
after 30 s.
<<behavior>>
0 MW
SV1
SV2
WATER LEVEL: 0 mm
PRESSURE: 0 bar
CP: 0 U/min
WP1: 0 U/min
WP2: 0 U/min
WV1
WV2
pressure: > 400
pressure: > 450
pressure: < 400
pressure: > 100
pressure: > 120
pressure: < 100
water: > 2200
water: > 2400
water: < 2200
water: < 1500
water: > 1700
water: < 1700
water: > 4500
water: > 5000
water: < 4500
water: < 300
water: > 500
water: < 500
after 2.5 s

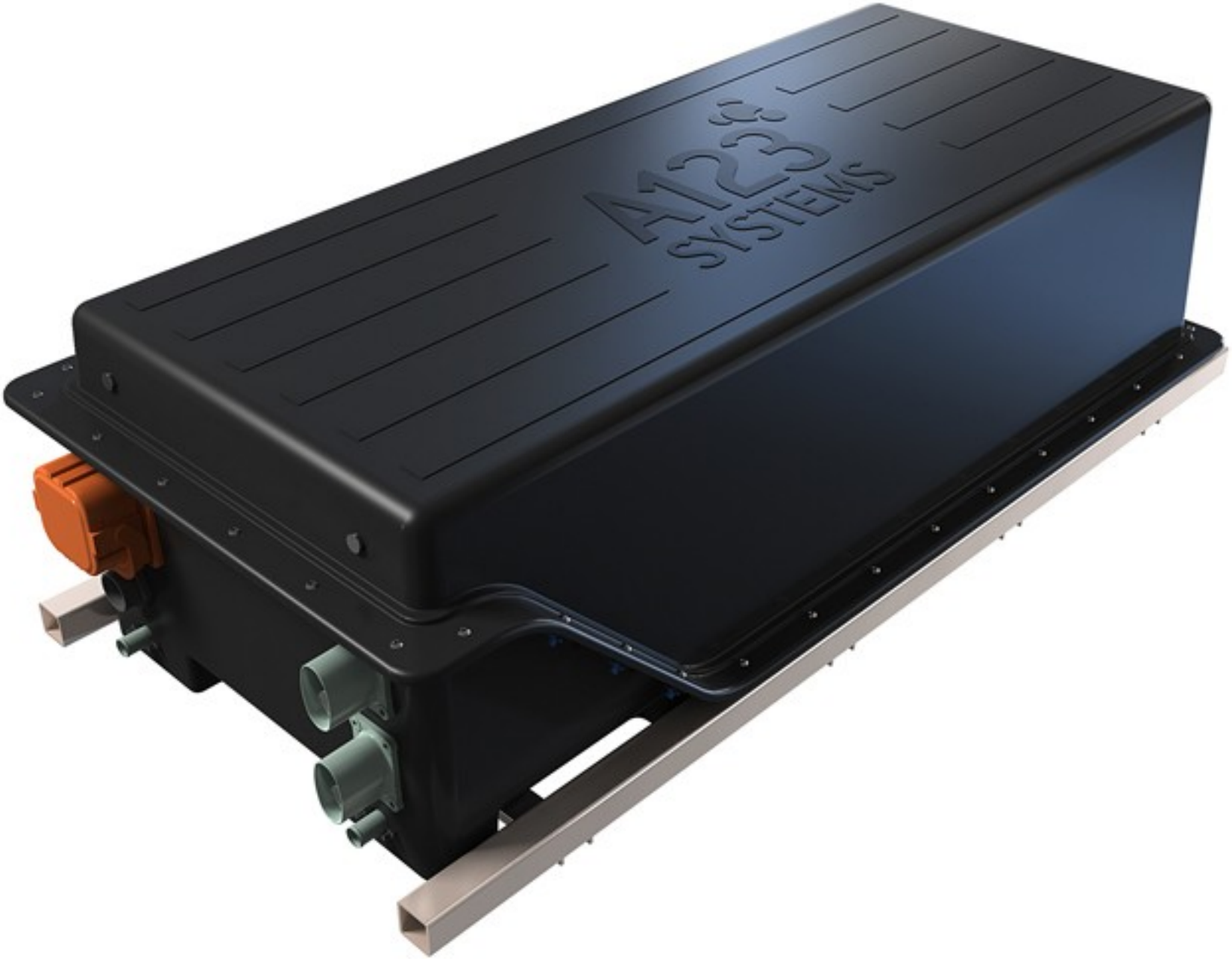


Boric Acid Transportation Pump

Product parameters

Design standards ： RCC-M
Flow ： 16.6m3/h
Head ： 85m
Temperature ： ~80℃
Pressure ： 1.6MPa

Used in 600MWe 、 900MWe 、 1000MWe PWR nuclear power plant boric acid transportation system.


A123
SYSTEMS

R = V*i

$$R = R_{ref}\,[1 + \alpha(T - T_{ref})]$$

Where,

R = Conductor resistance at temperature "T"

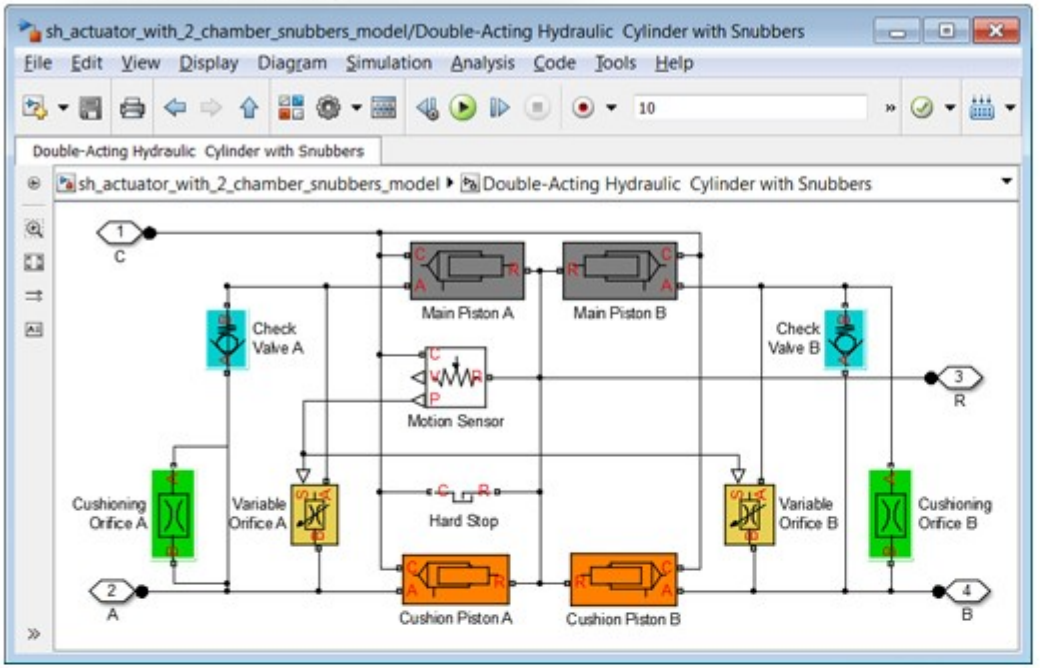
R_{ref} = Conductor resistance at reference temperature T_{ref}, usually 20° C, but sometimes 0° C.

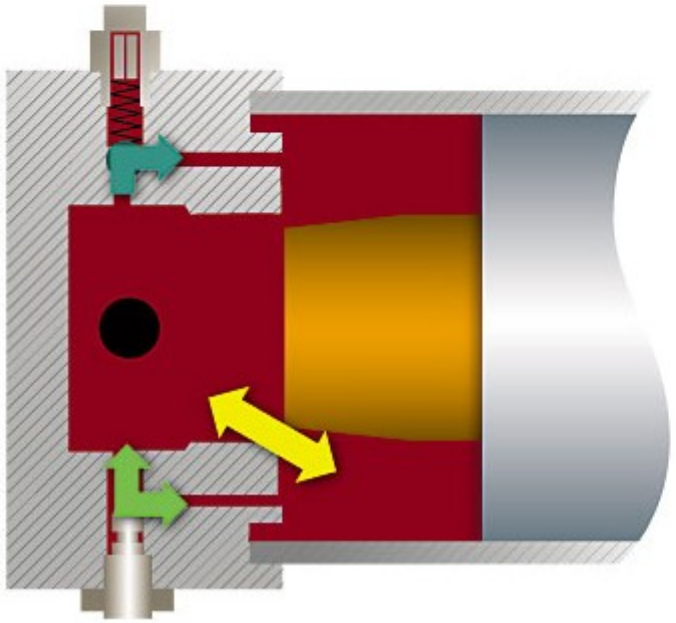
α = Temperature coefficient of resistance for the conductor material.

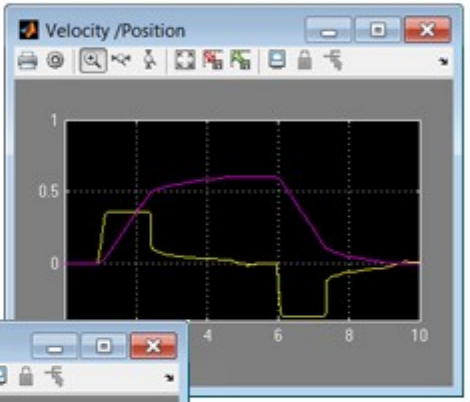
T = Conductor temperature in degrees Celcius.

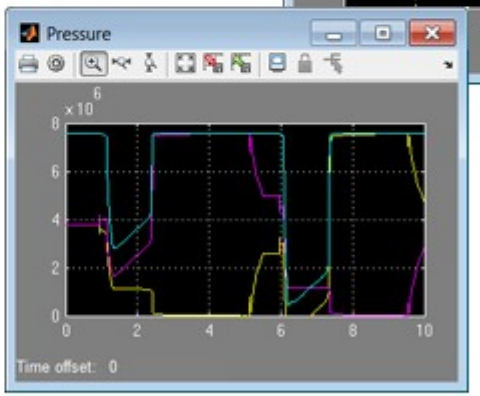
T_{ref} = Reference temperature that α is specified at for the conductor material.

SimHydraulics









Modelica in Context

- Domain/Problem-Specific
- Laws of Physics
- Power Flow
- a-causal (Mathematical) ← **Modelica**
- Causal Block Diagrams
- Numerical (Discrete) Approximations
- Computer Numerical (Floating Point, Fixed Point)
- As-Fast-As-Possible vs. Real-time
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)

* **Newton's Three Laws of Motion:**

Fundamental relationship between the acceleration of an object and the total forces acting upon it.

- First Law states that in order for the motion of an object to change, a force must act upon it, a concept generally called inertia.
- Second Law defines the relationship between acceleration, force, and mass.
- Third Law states that any time a force acts from one object to another, there is an equal force acting back on the original object.

* **Newton's Law of Gravity:**

Explains the attractive force between a pair of masses. In the twentieth century, it became clear that this is not the whole story, as Einstein's theory of general relativity has provided a more comprehensive explanation for the phenomenon of gravity.

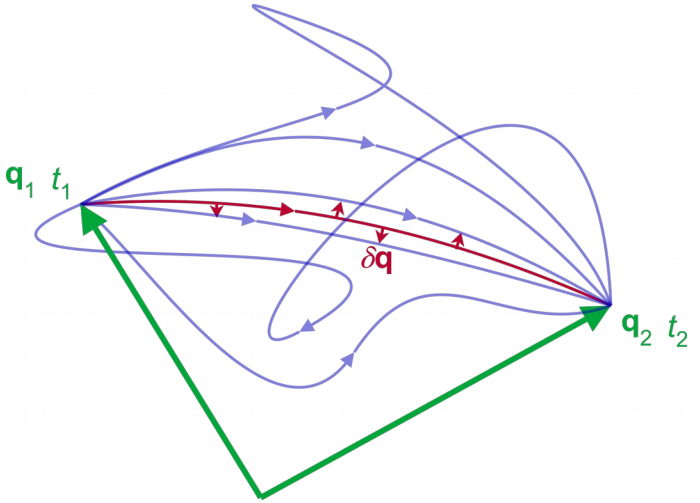
* **Conservation of Mass-Energy:**

The total energy in a closed or isolated system is constant, no matter what happens.

* **Conservation of Momentum:**

The total momentum in a closed or isolated system remains constant. An alternative of this is the law of conservation of angular momentum.

* …



In non-relativistic physics, the **principle of least action** – or, more accurately, the **principle of stationary action** – is a variational principle that, when applied to the action of a mechanical system, can be used to obtain the equations of motion for that system by stating a system follows the path where the average difference between the kinetic energy and potential energy is minimized or maximized over any time period. It is called stable if minimized. In relativity, a different average must be minimized or maximized. The principle can be used to derive Newtonian, Lagrangian, and Hamiltonian equations of motion.

The starting point is the *action*, denoted $\mathcal{S}$ (calligraphic S), of a physical system. It is defined as the integral of the Lagrangian L between two instants of time t_1 and t_2 - technically a functional of the N generalized coordinates $\mathbf{q} = (q_1, q_2 \ldots q_N)$ which define the configuration of the system:

$$\mathcal{S}[\mathbf{q}(t)] = \int_{t_1}^{t_2} L(\mathbf{q}(t), \dot{\mathbf{q}}(t), t)dt$$

where the dot denotes the time derivative, and t is time.

Mathematically the principle is[11][12][13]

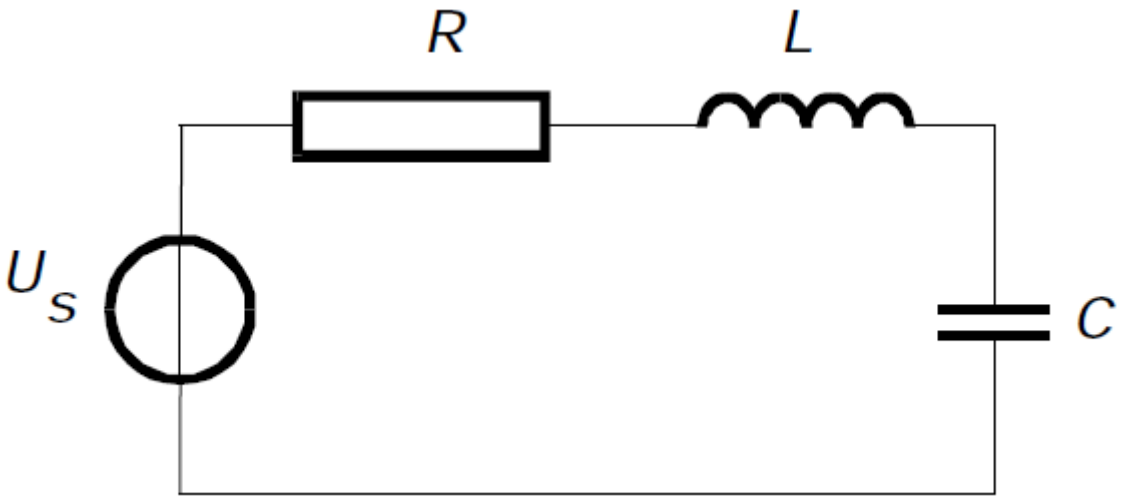
$$\delta\mathcal{S} = 0$$

where δ (Greek lowercase delta) means a *small* change. In words this reads:[10]

> *The path taken by the system between times t_1 and t_2 is the one for which the* ***action*** *is* ***stationary (no change)*** *to* ***first order***.

Modelica in Context

- Domain/Problem-Specific
- Laws of Physics
- Power Flow
- a-causal (Mathematical) ← **Modelica**
- Causal Block Diagrams
- Numerical (Discrete) Approximations
- Computer Numerical (Floating Point, Fixed Point)
- As-Fast-As-Possible vs. Real-time
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)

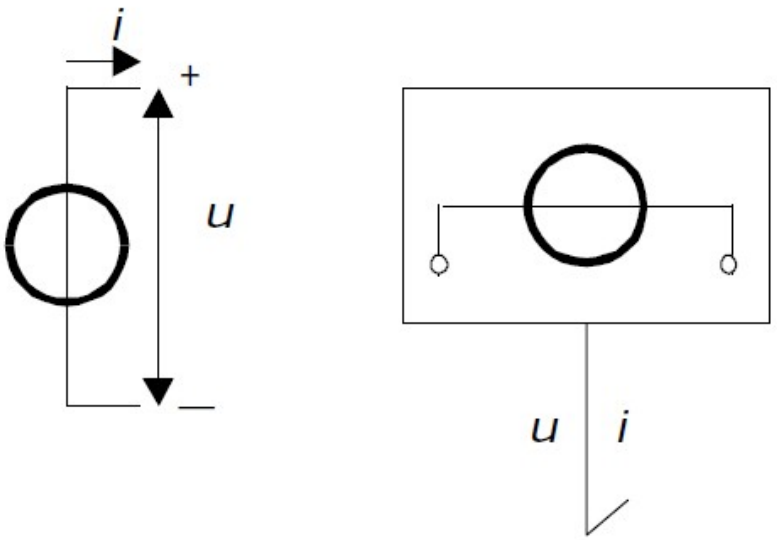


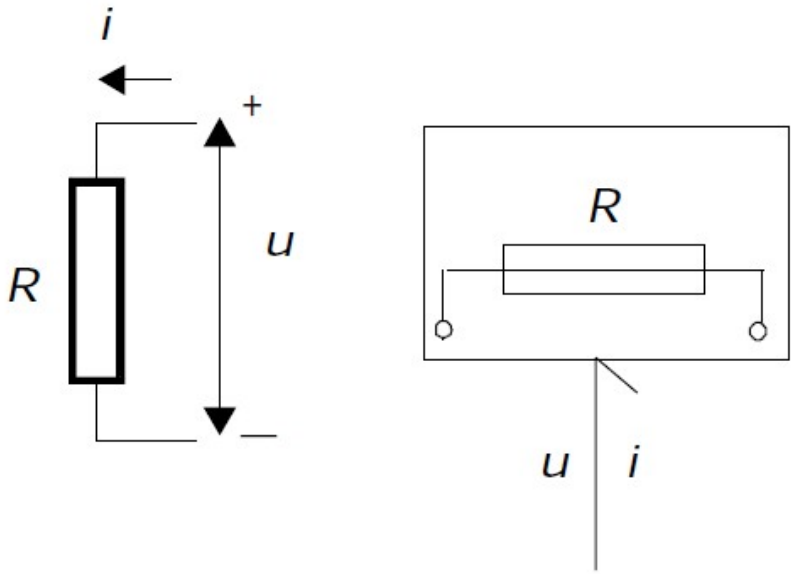
From Jan Broenink's Introduction to physical systems modelling with bond graphs

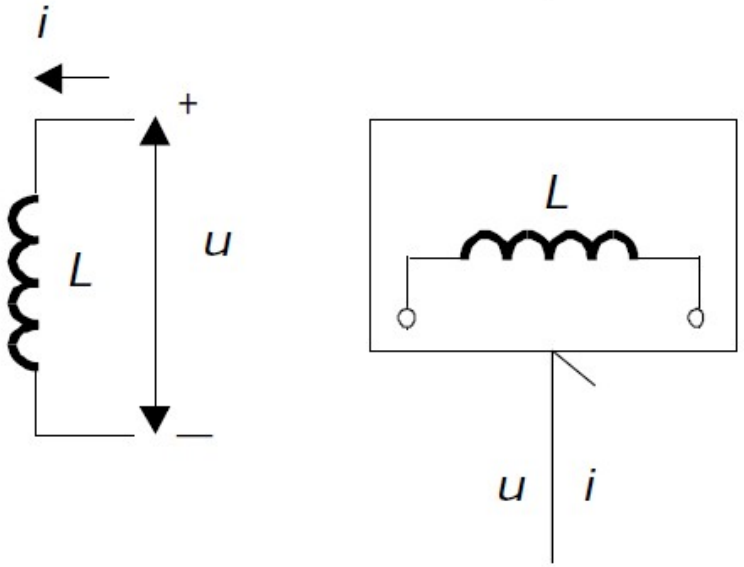
$$u_R = iR$$

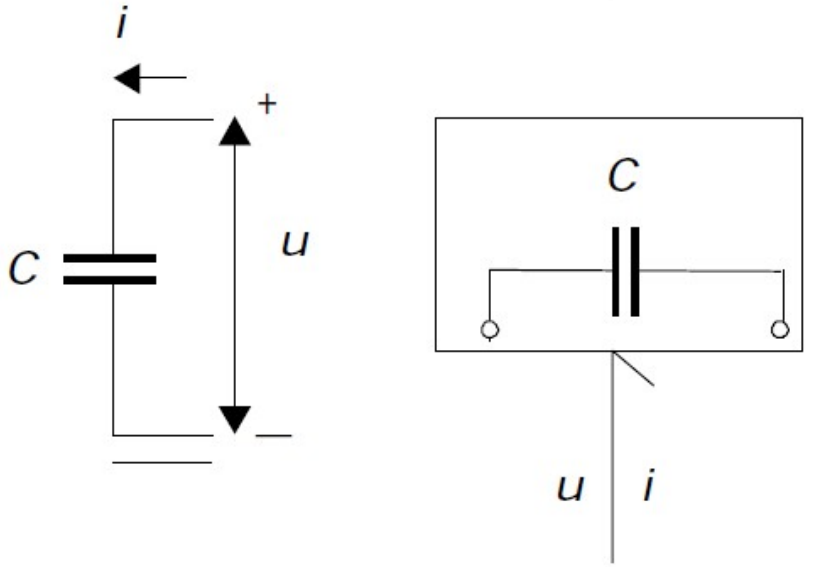
$$u_C = \frac{1}{C}\int i\mathrm{d}t$$

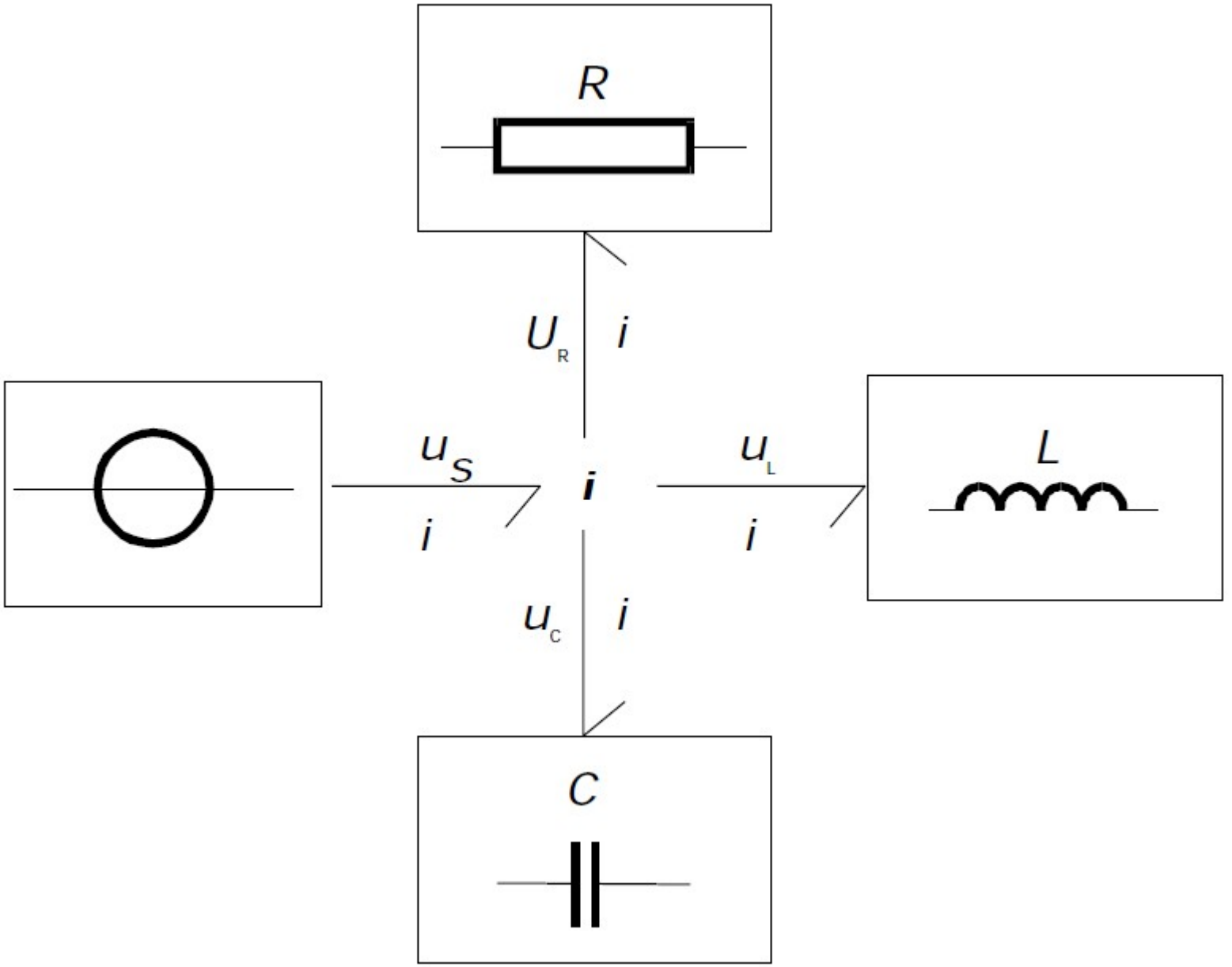
$$u_L = L\frac{di}{dt} \quad \text{or} \quad i_L = \frac{1}{L}\int u\mathrm{d}t$$


i
+
u
_
u
i

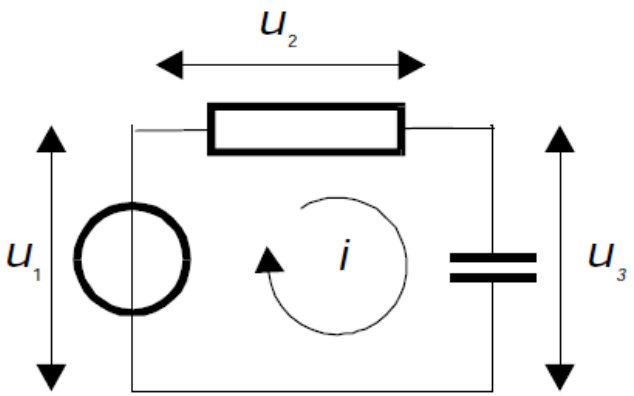

i
+
R
u
_
R
u
i


i
+
L
u
_
L
u
i

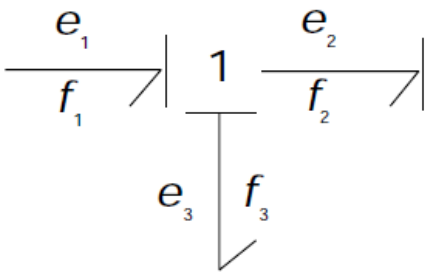

i
+
C
u
_
C
u
i


R
U_R
i
u_S
i
i
u_L
i
L
u_C
i
C

Domain-specific symbols



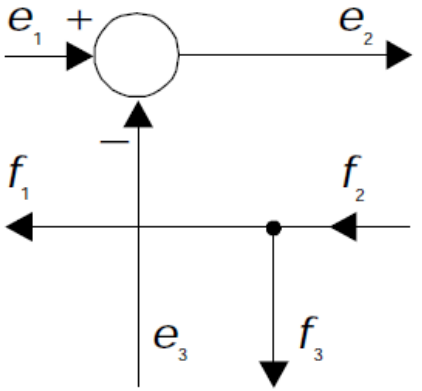
Bond-graph element



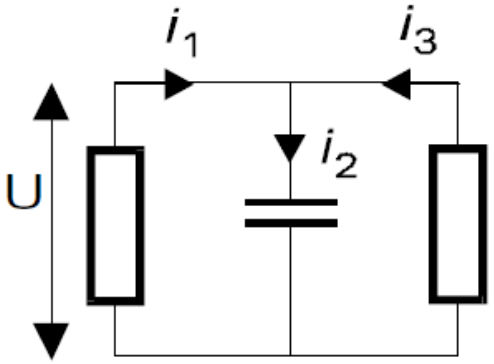
Equations

$$f_1 = f_2$$
$$f_3 = f_2$$
$$e_2 = e_1 - e_3$$

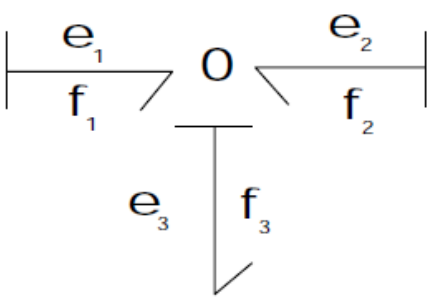
Block diagram expansion



Domain-specific symbols



Bond-graph element



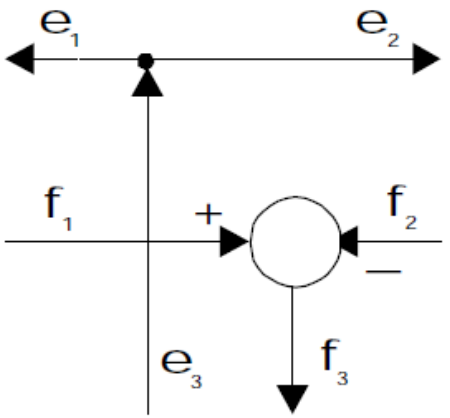
Equations

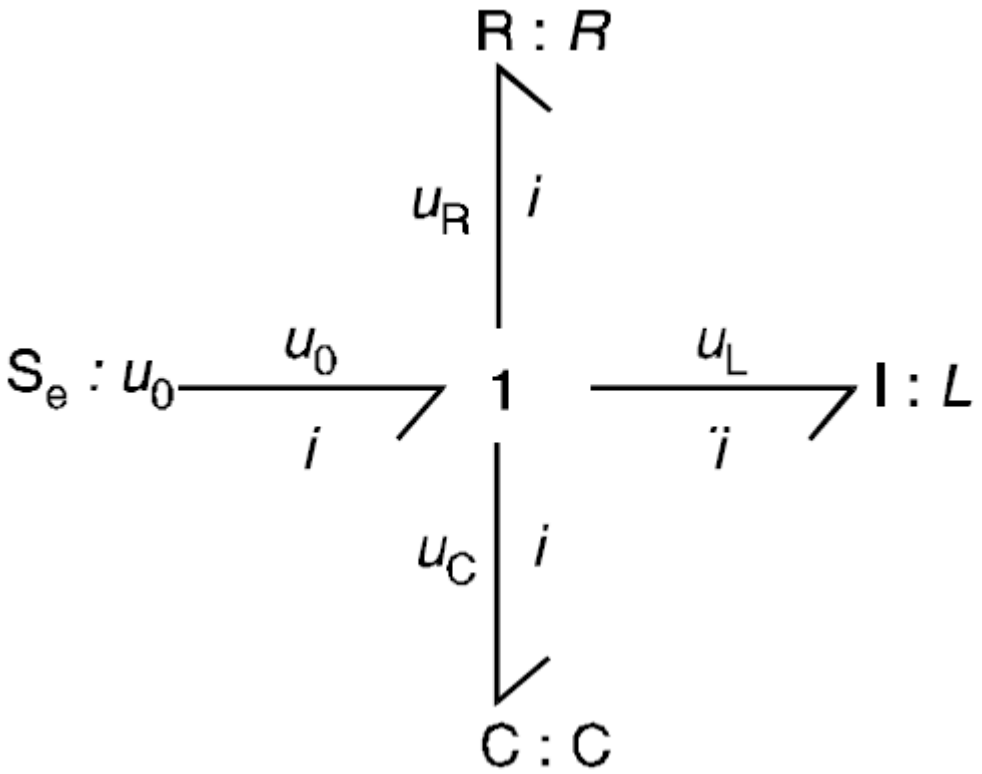
$$e_1 = e_3$$

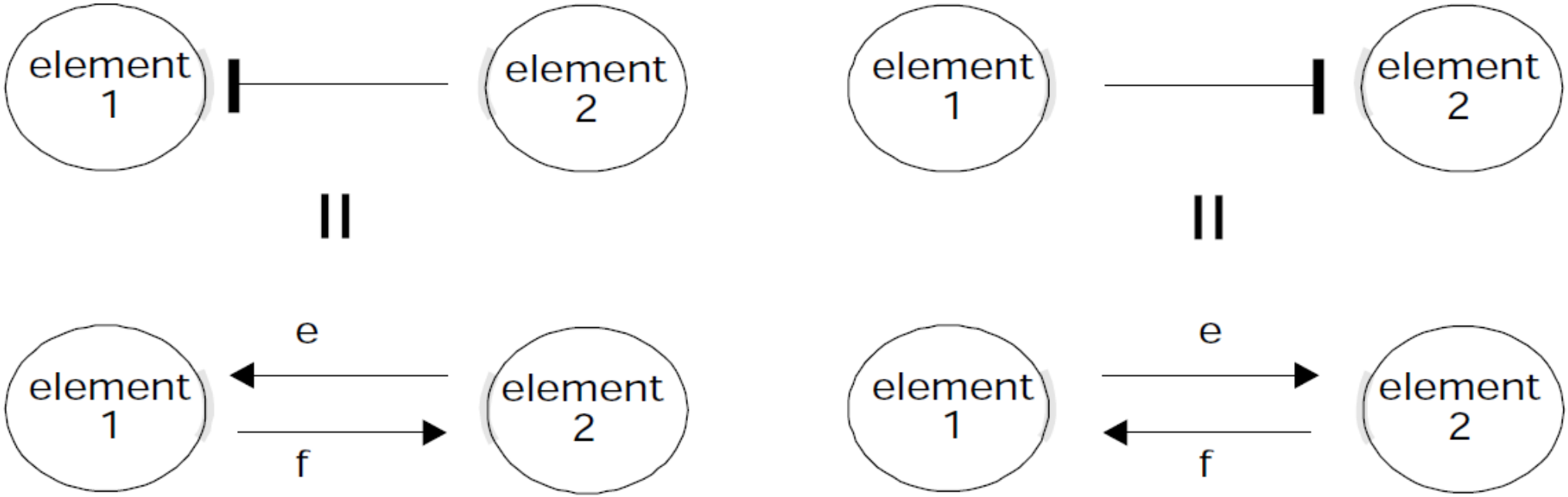
$$e_2 = e_3$$

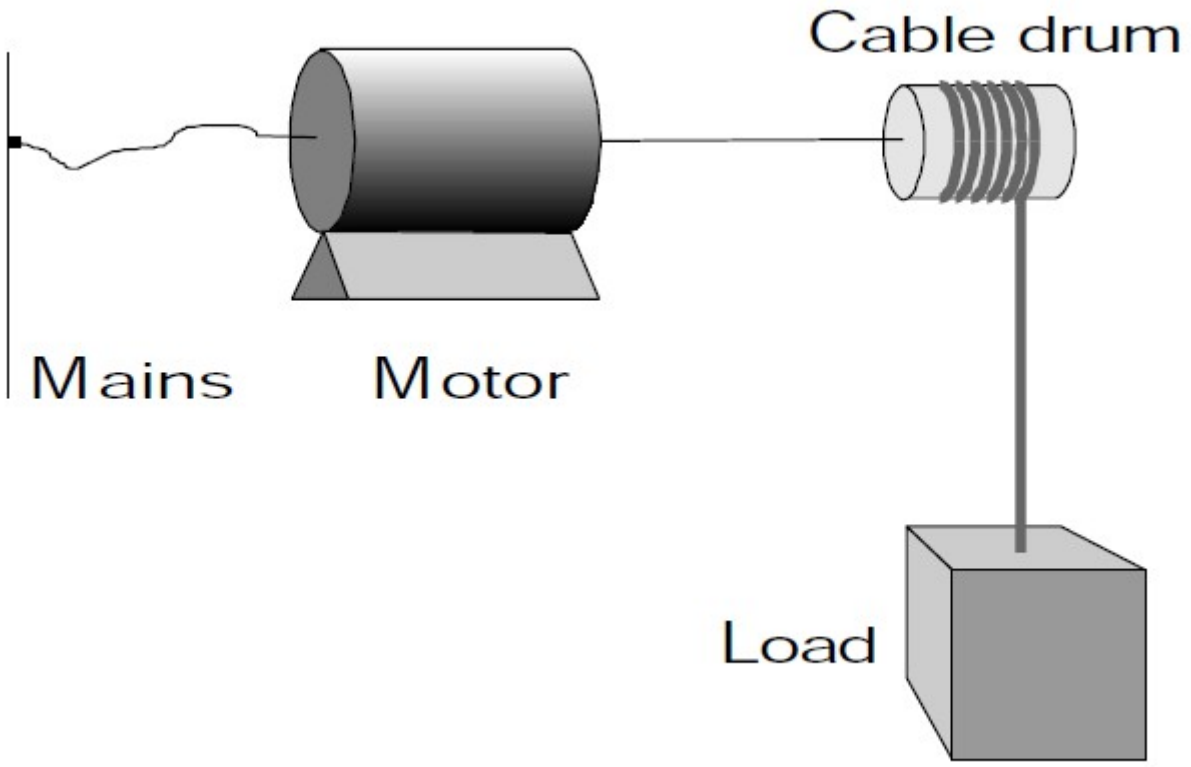
$$f_3 = f_1 - f_2$$

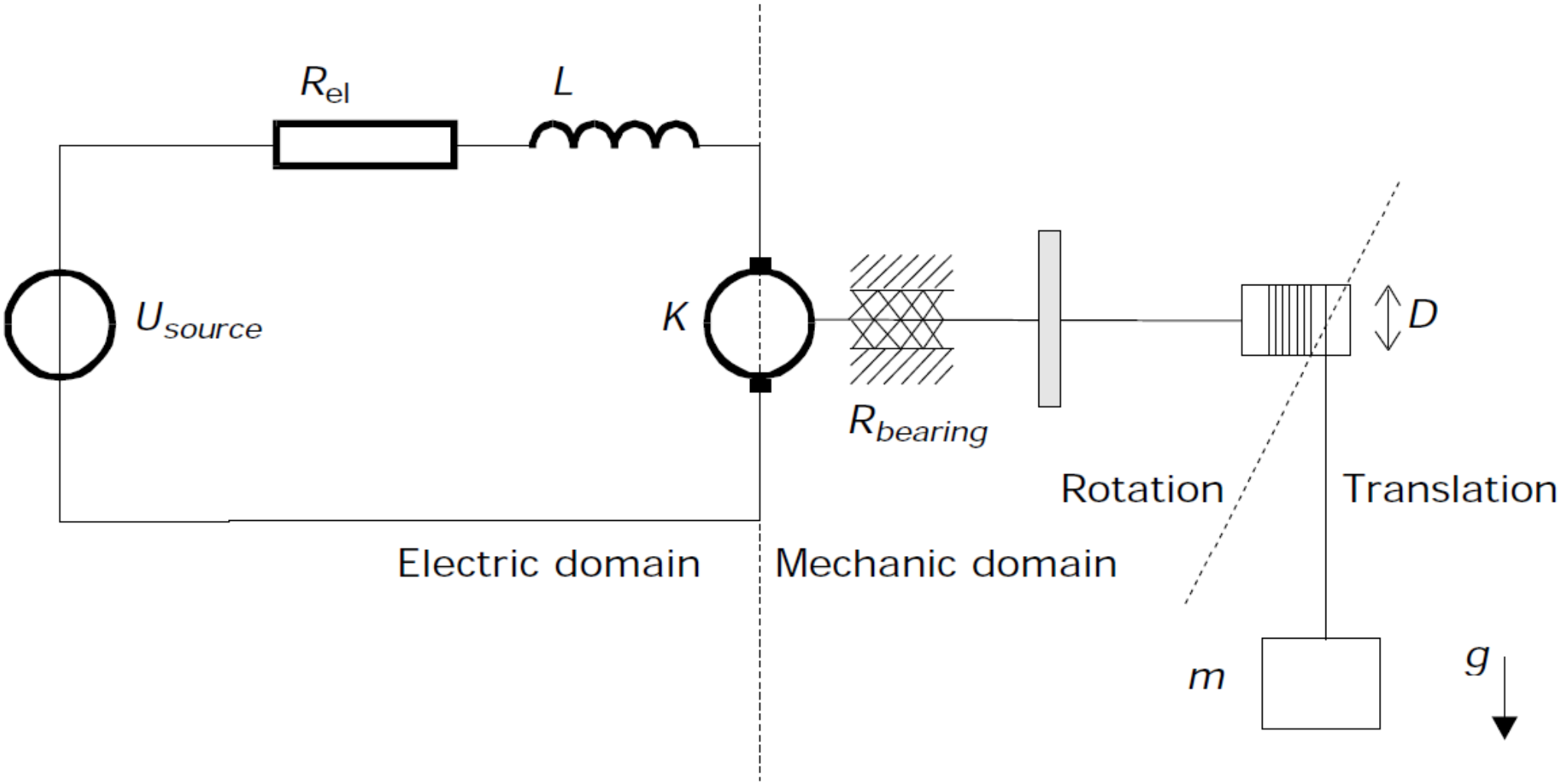
Block-diagram expansion

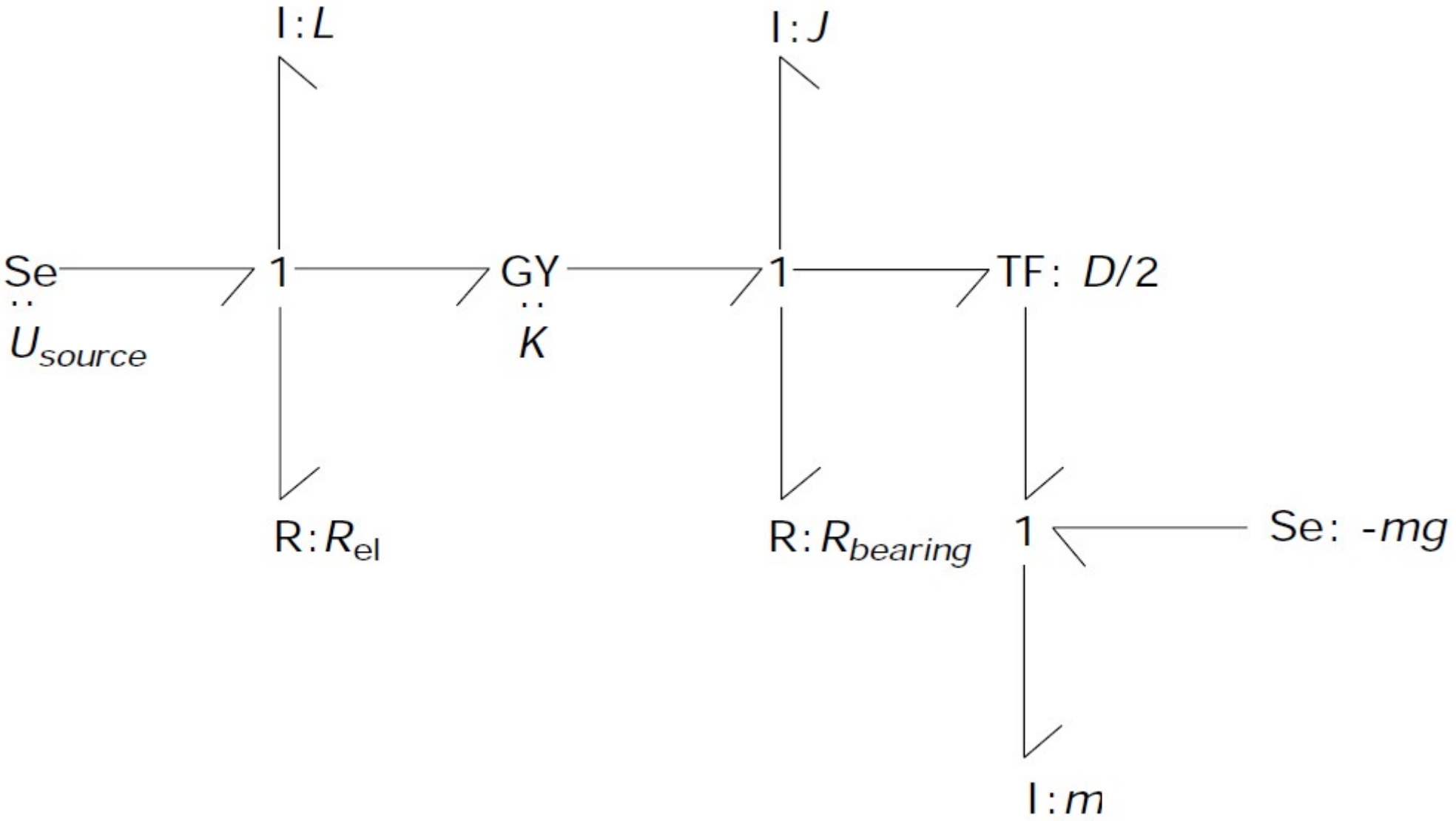


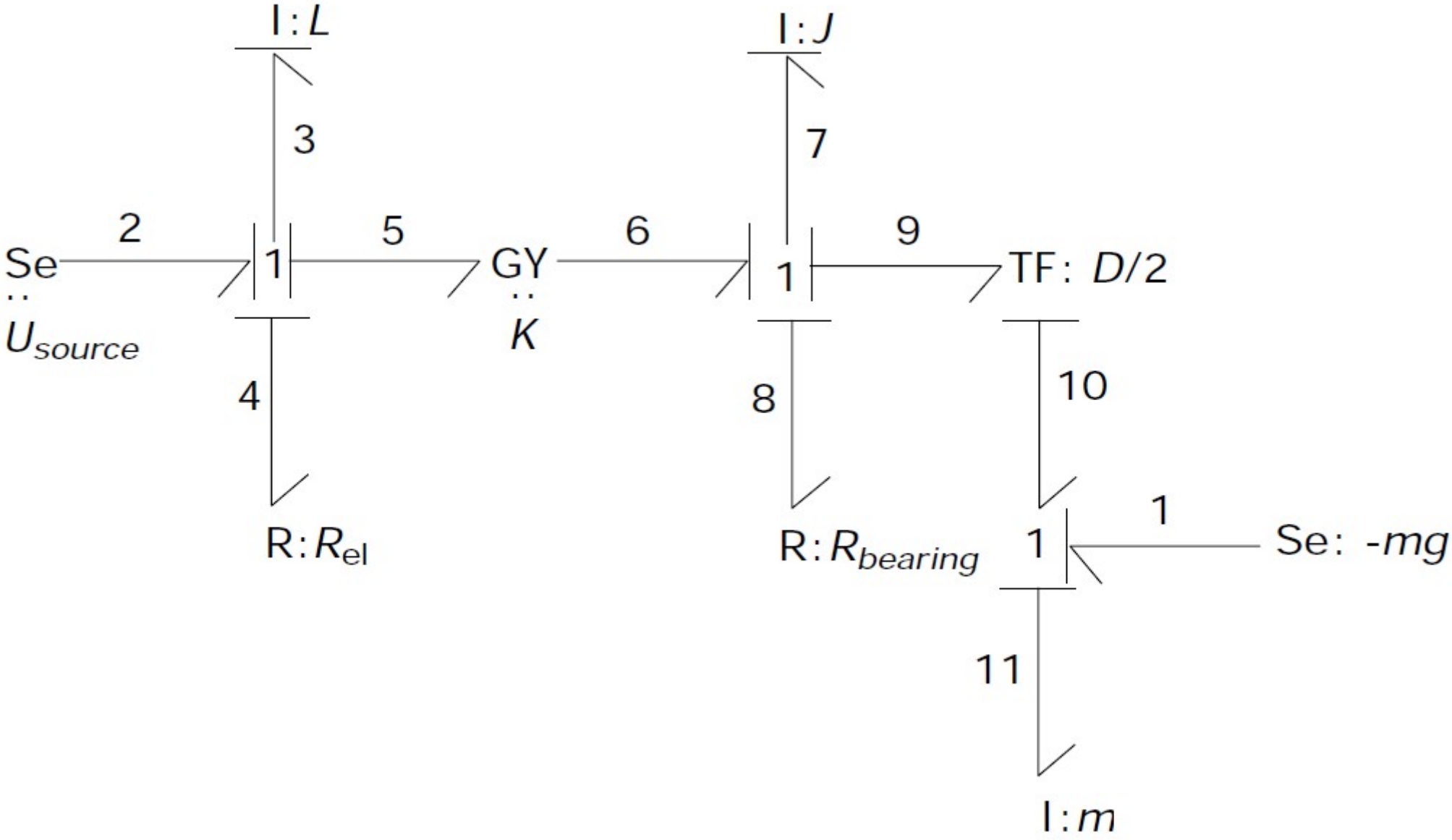

R : R
u_R
i
$S_e : u_0$
u_0
i
1
u_L
i
I : L
u_C
i
C : C

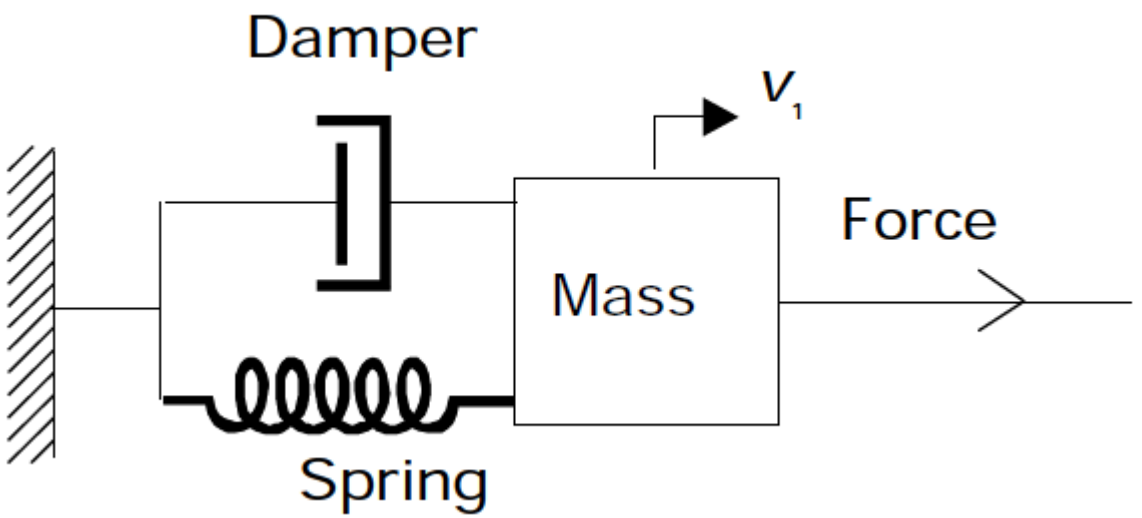

element 1
element 2
element 1
element 2
e
f
element 1
element 2
e
f
element 1
element 2

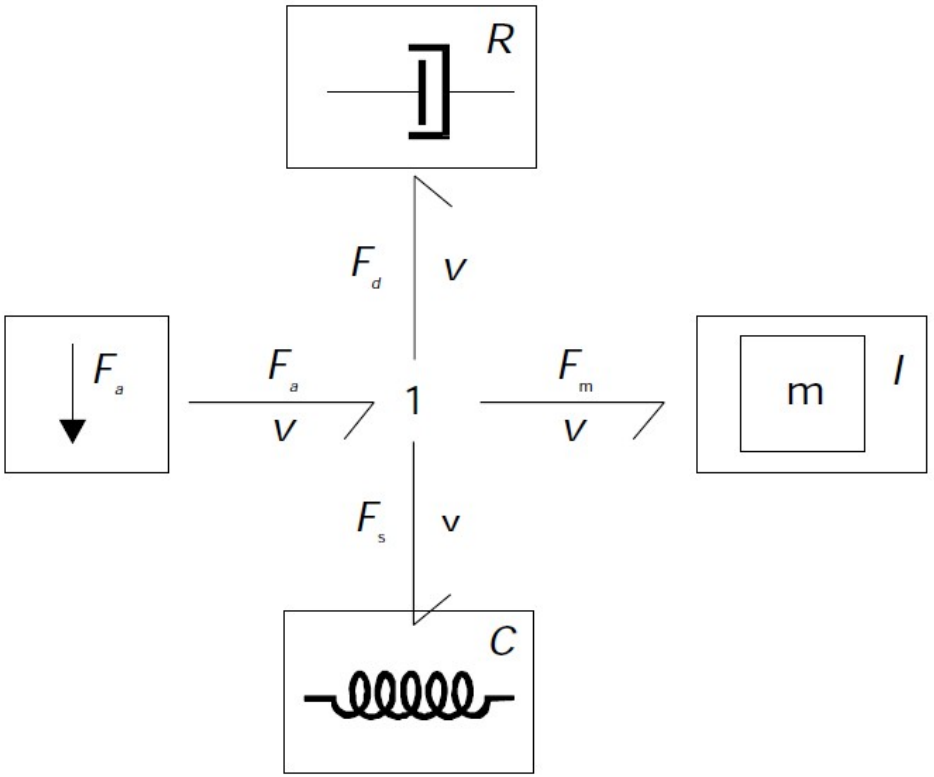

Cable drum
Mains
Motor
Load

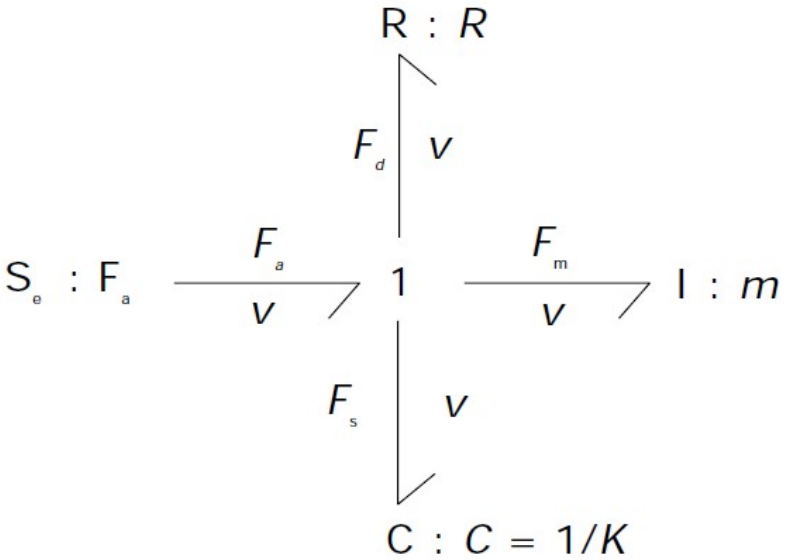

R_{el}
L
U_{source}
K
$R_{bearing}$
D
Rotation
Translation
Electric domain
Mechanic domain
m
g

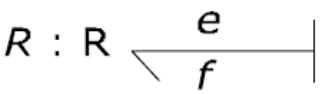
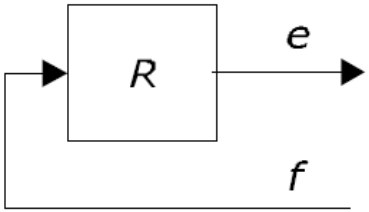
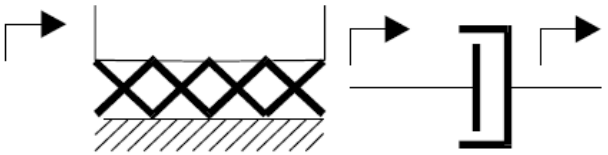
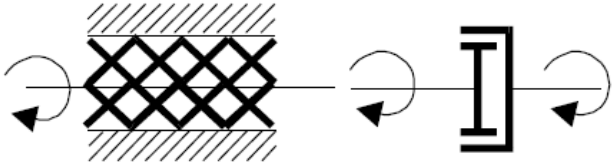
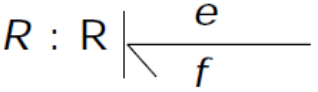
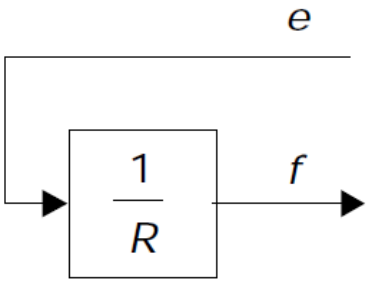

I:L
I:J
Se
1
GY
1
TF: D/2
U_source
K
R:R_el
R:R_bearing
1
Se: -mg
I:m

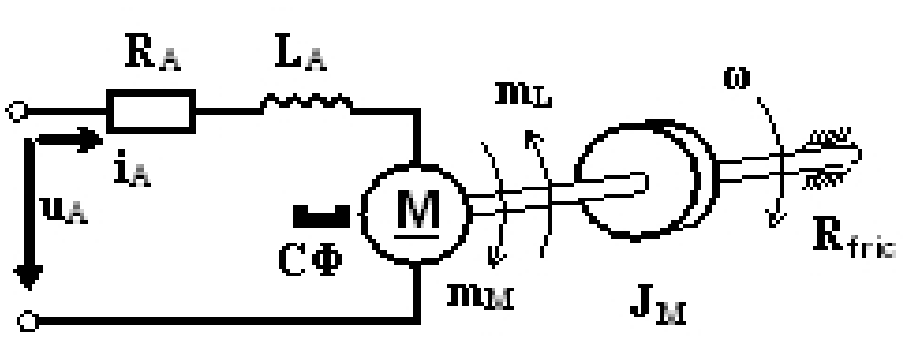

I : L
I : J
3
7
2
5
6
9
Se
1
GY
1
TF : D/2
U source
K
4
8
10
R : R el
R : R bearing
1
1
Se : -mg
11
I : m

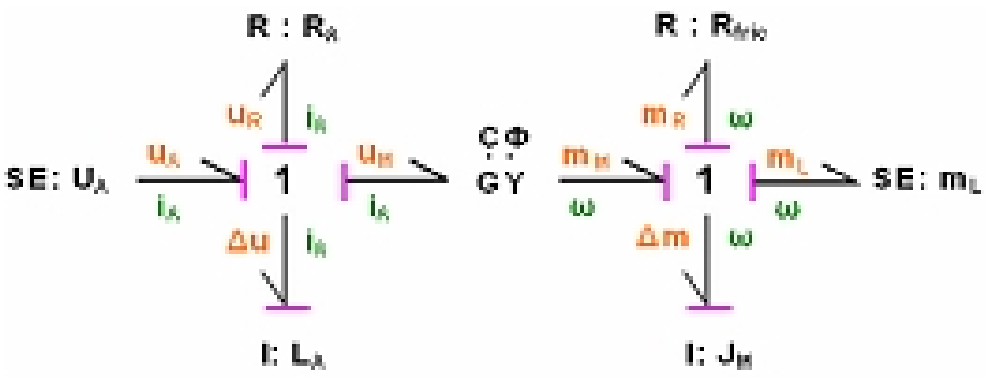

Damper
v_1
Force
Mass
Spring

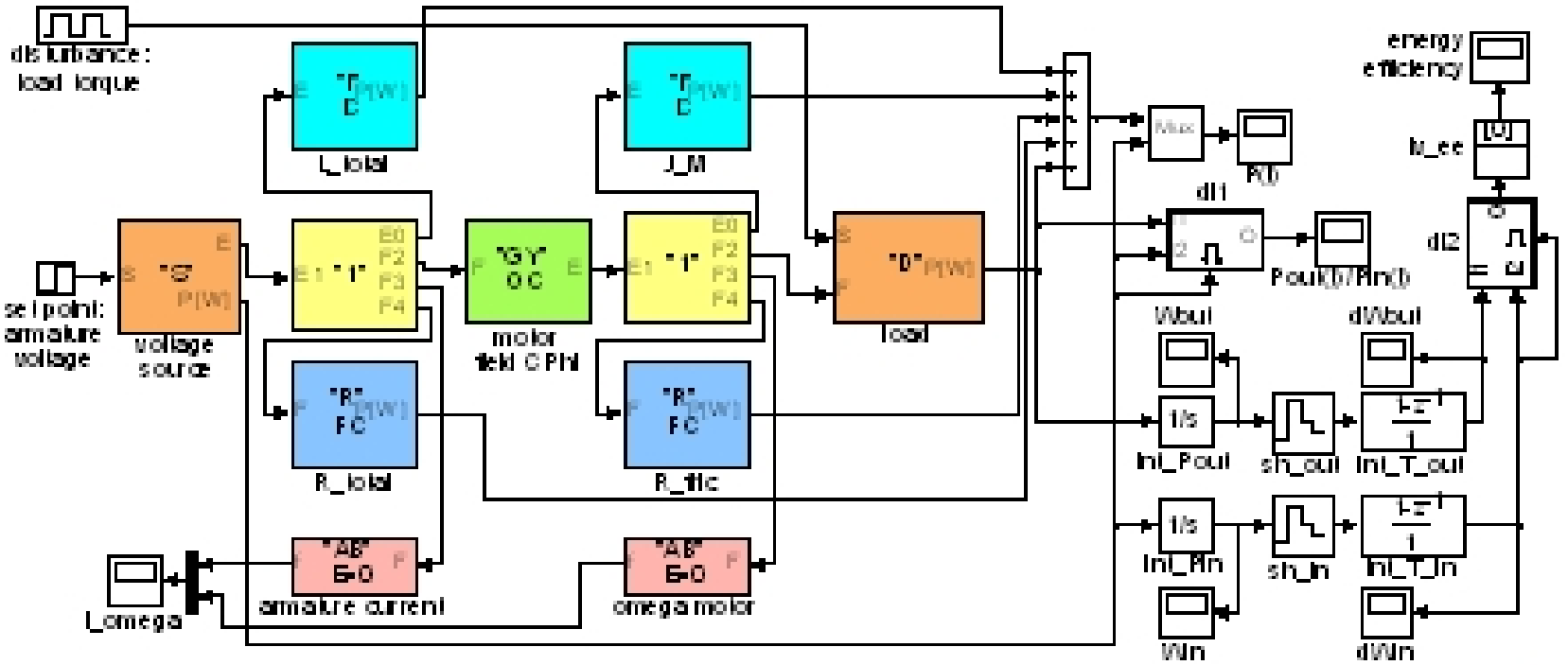

R
F_d
v
F_a
F_a
v
1
F_m
v
m
I
F_s
v
C

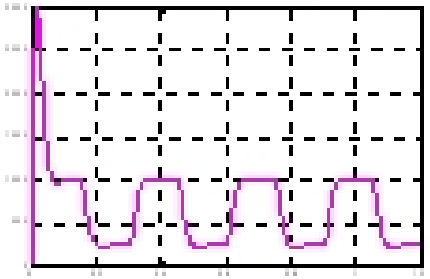

R : R
F_d
v
S_e : F_a
F_a
v
1
F_m
v
I : m
F_s
v
C : C = 1/K

Domain-specific symbols	Bond-graph element	Equations	Block diagram expansion
 Resistor		$e = Rf$	
 Friction Damper			
 Friction Damper		$f = \frac{1}{R} e$	

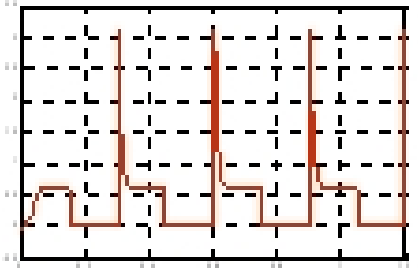




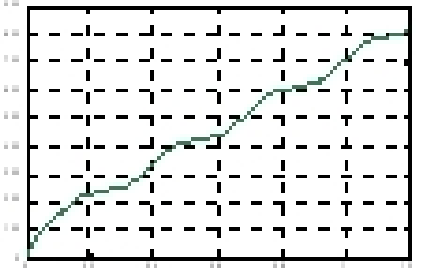




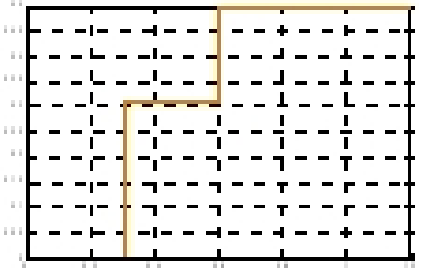
Power P(t)



Pout(t)/Pin(t)



Energy Win



Energy efficiency

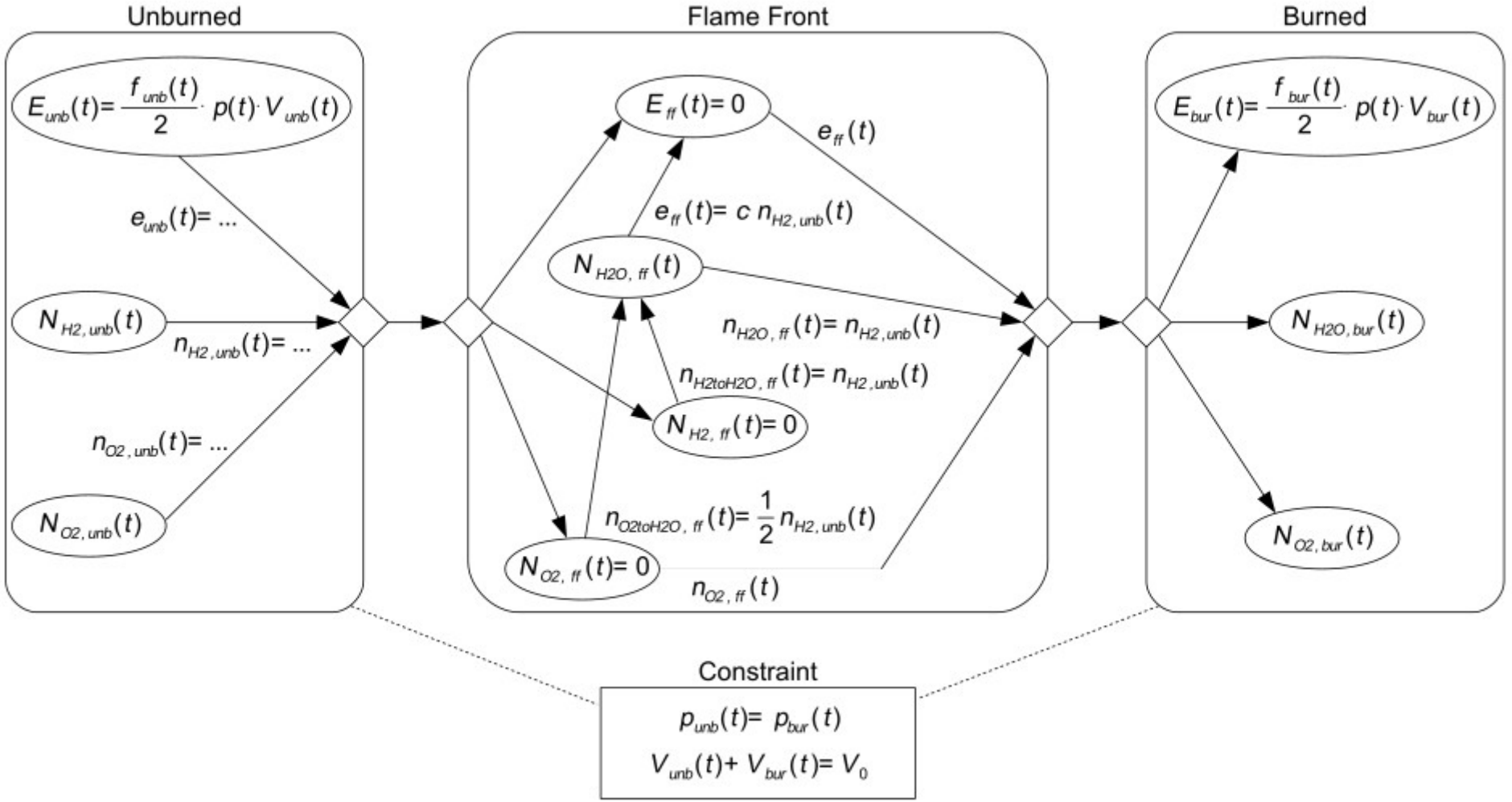
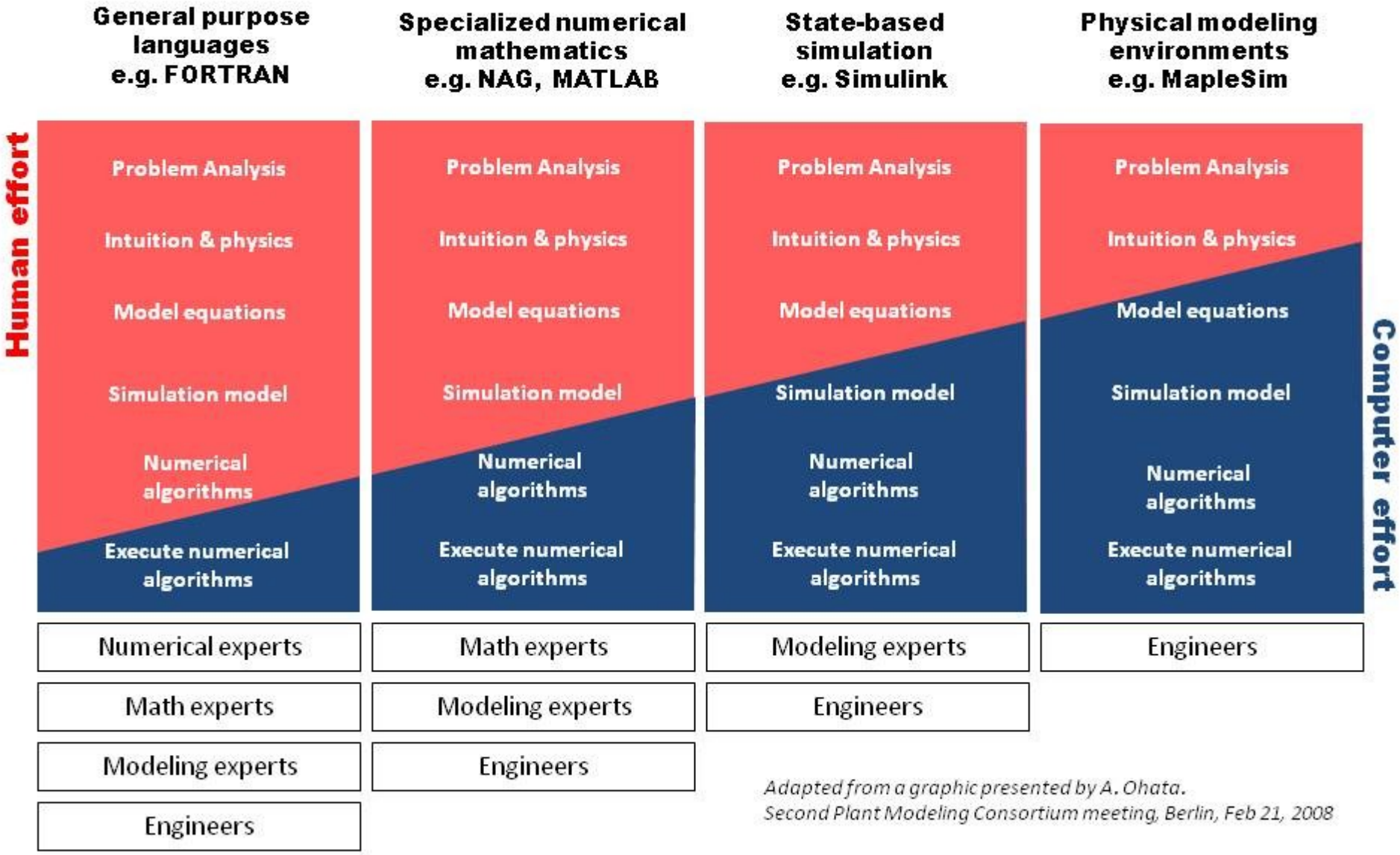


Figure 2. High Level Model Description (HLMD) example - hydrogen-oxygen combustion in a closed chamber.

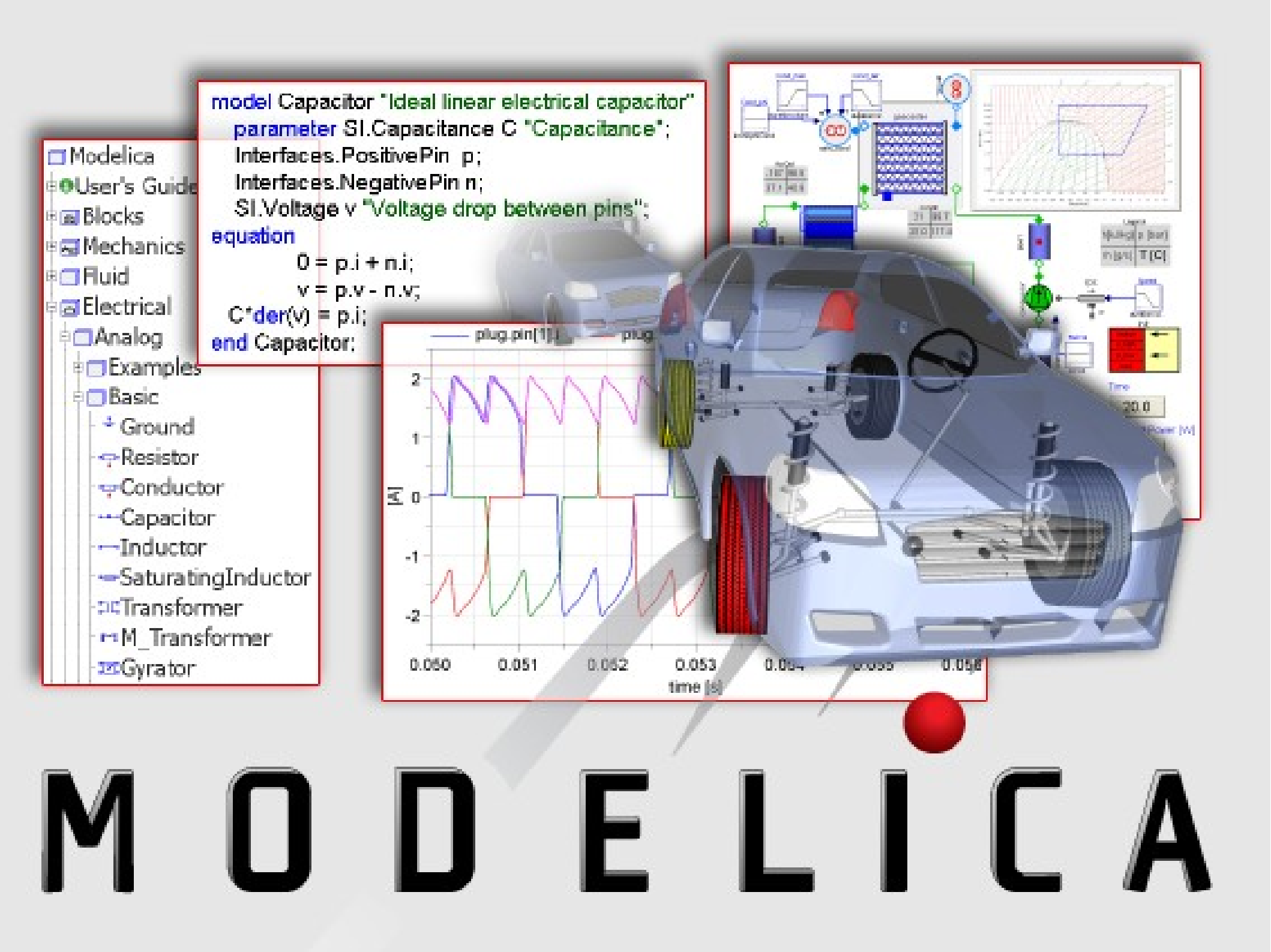
Akira Ohata @ Toyota



Adapted from a graphic presented by A. Ohata.
Second Plant Modeling Consortium meeting, Berlin, Feb 21, 2008

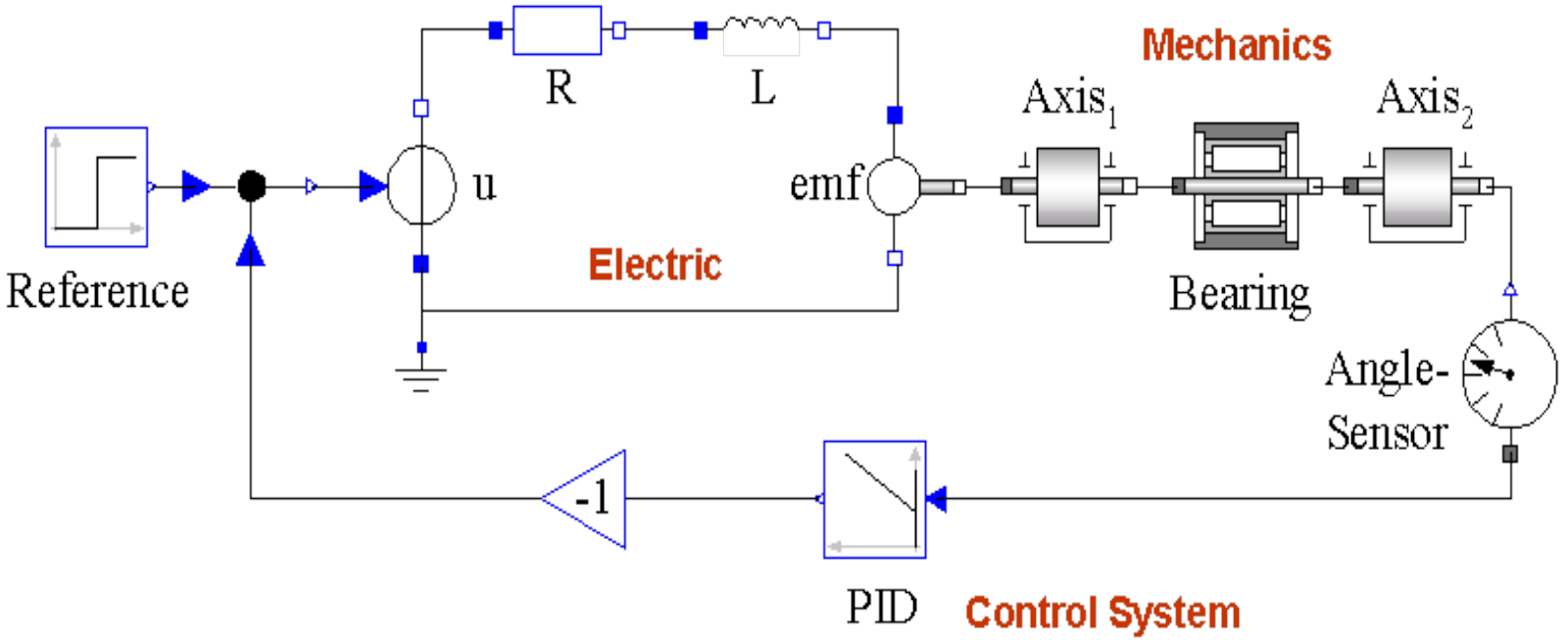
Modelica in Context

- Domain/Problem-Specific
- Laws of Physics
- Power Flow
- a-causal (Mathematical) ← **Modelica**
- Causal Block Diagrams
- Numerical (Discrete) Approximations
- Computer Numerical (Floating Point, Fixed Point)
- As-Fast-As-Possible vs. Real-time
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)


model Capacitor "Ideal linear electrical capacitor"
parameter SI.Capacitance C "Capacitance";
Interfaces.PositivePin p;
Interfaces.NegativePin n;
SI.Voltage v "Voltage drop between pins";
equation
0 = p.i + n.i;
v = p.v - n.v;
C*der(v) = p.i;
end Capacitor;
Modelica
User's Guide
Blocks
Mechanics
Fluid
Electrical
Analog
Examples
Basic
Ground
Resistor
Conductor
Capacitor
Inductor
SaturatingInductor
Transformer
M_Transformer
Gyrator
plug.pin[1].i
plug.pin[2].i
2
1
0
-1
-2
[A]
0.050
0.051
0.052
0.053
time [s]
MODELICA

Multi-Domain Modeling

http://www.modelica.org



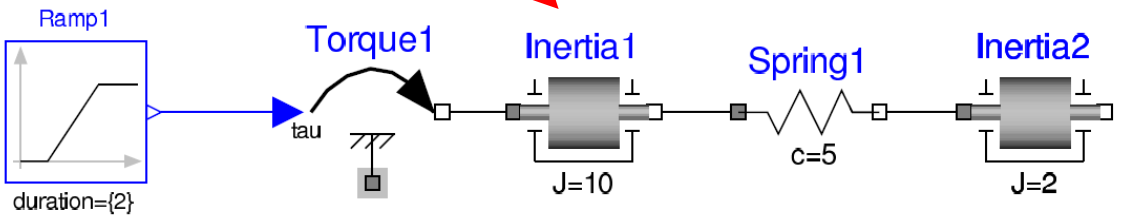
this slide from Peter Fritzson's Modelica tutorial

Multi-Domain Modeling

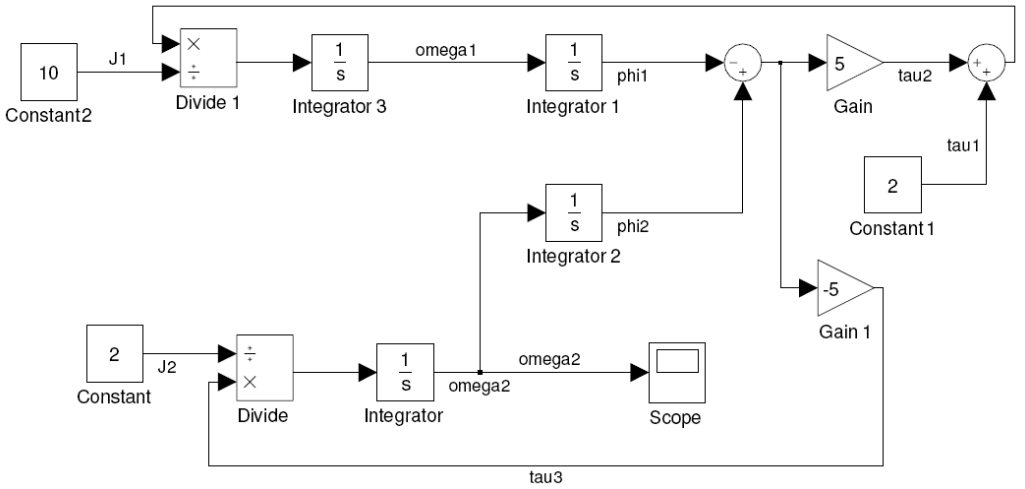
Visual Acausal Hierarchical Component Modeling

Keeps the physical structure

Acausal model (Modelica)



Causal block-based model (Simulink)



this slide from Peter Fritzson's Modelica tutorial

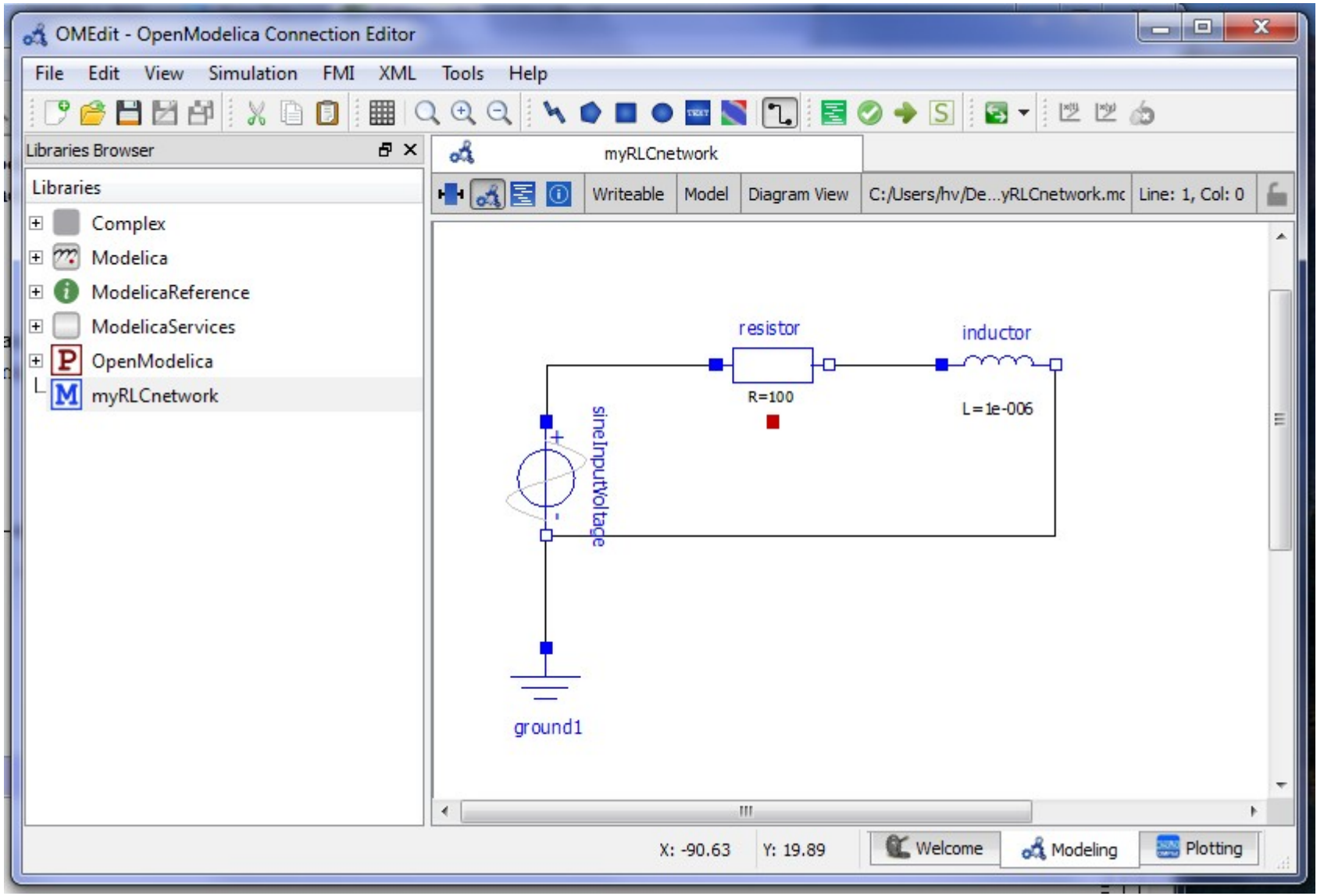
MODELICA

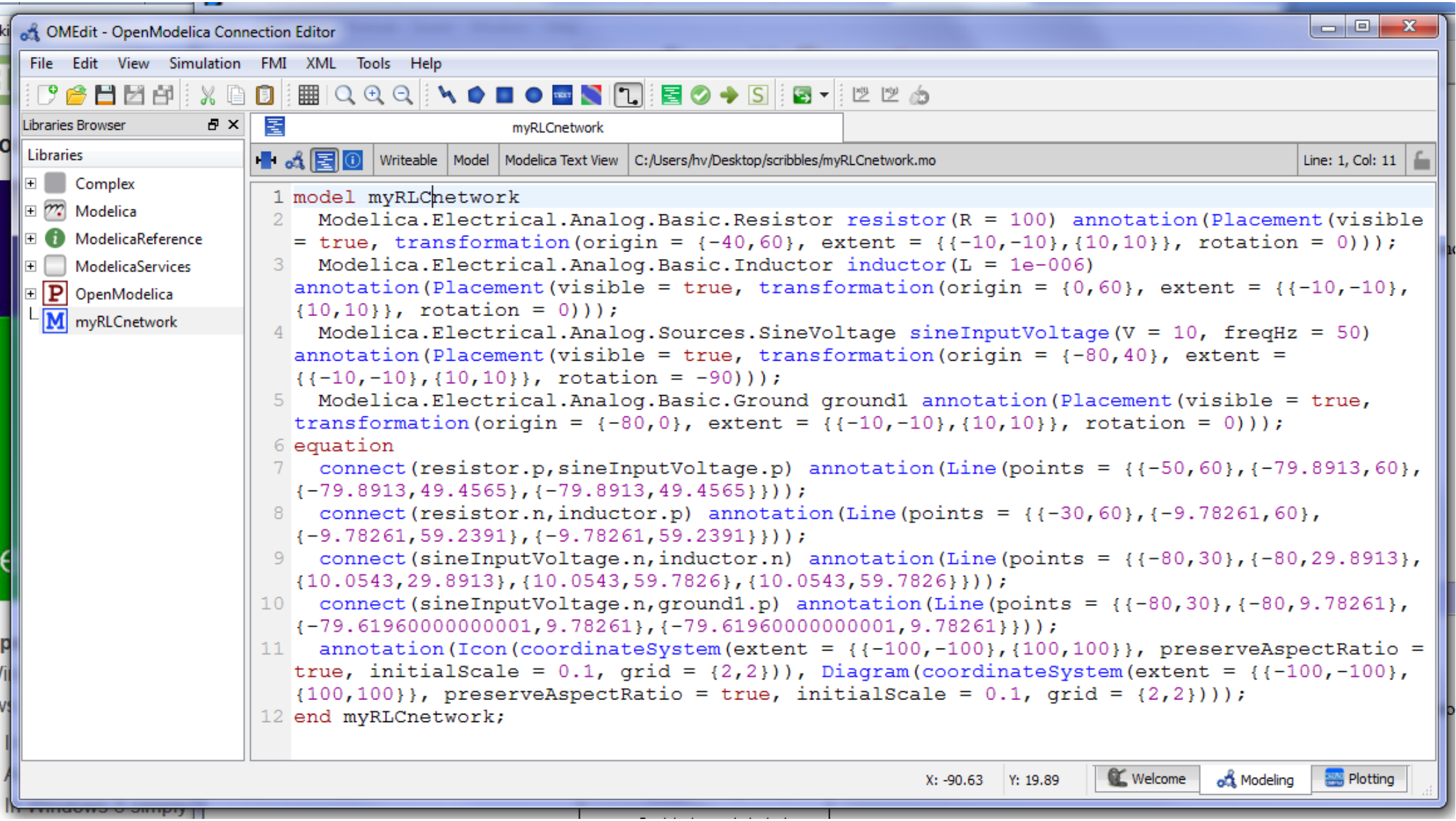
- Model exchange/re-use standard (Modelica Association)
- Modelica Standard Library (MSL)
- Object-oriented, hierarchical; semantics based on flattening
- Computationally a-causal modelling; semantics based on DAEs
- Originated in Hilding Elmquist's 1978 PhD thesis @ Lund
- Early 1990's: Modelica Design Team (started in SiE)

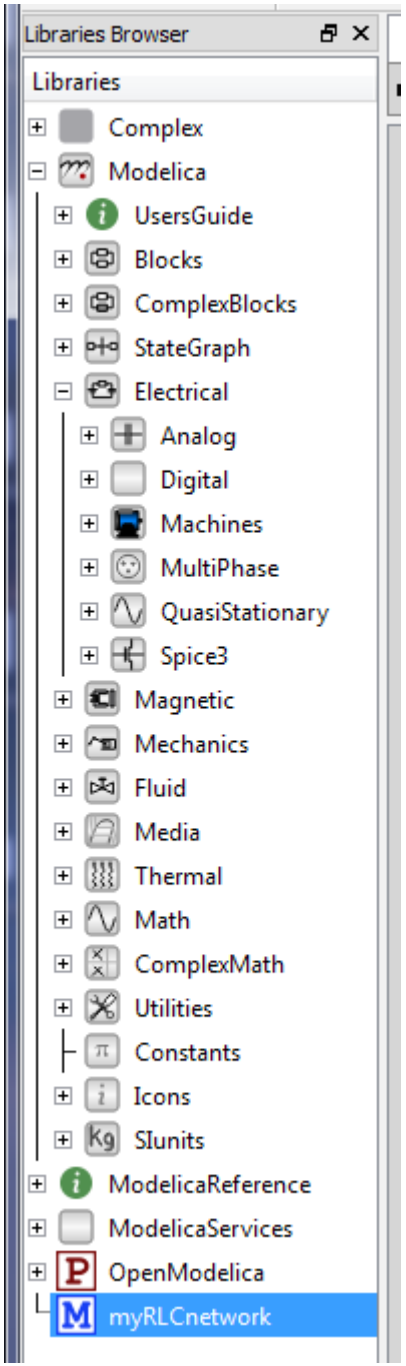

model Capacitor "Ideal linear electrical capacitor"
parameter SI.Capacitance C "Capacitance";
Interfaces.PositivePin p;
Interfaces.NegativePin n;
SI.Voltage v "Voltage drop between pins";
equation
0 = p.i + n.i;
v = p.v - n.v;
C*der(v) = p.i;
end Capacitor;
Modelica
User's Guide
Blocks
Mechanics
Fluid
Electrical
Analog
Examples
Basic
Ground
Resistor
Conductor
Capacitor
Inductor
SaturatingInductor
Transformer
M_Transformer
Gyrator
MODELICA

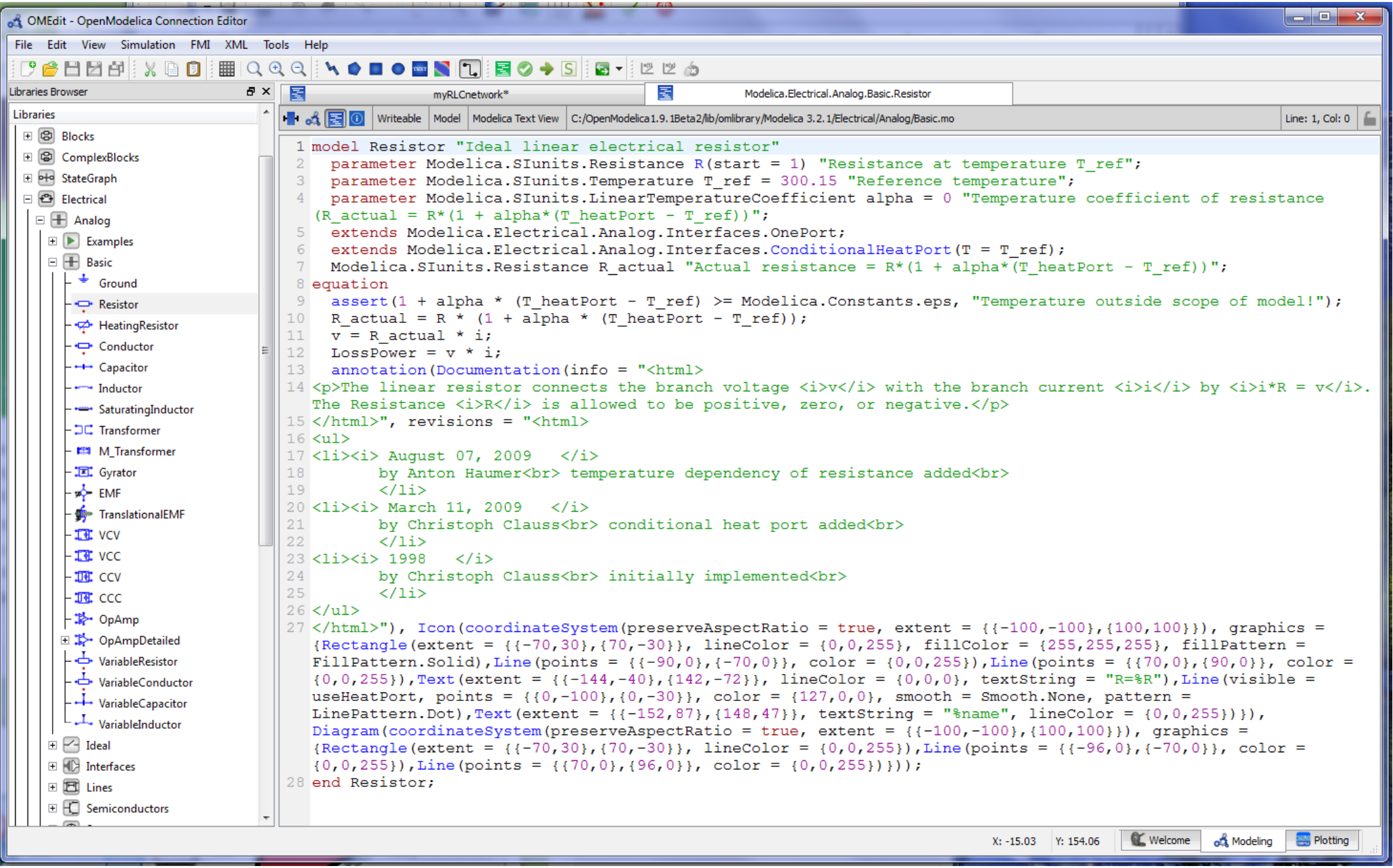
MODELICA

- hybrid (discrete-time/discrete-event) constructs (e.g., used to model network protocols based on TrueTime http://www.control.lth.se/truetime/)
- Limited support for Dynamic Structure models (i.e., no “agents”)
- Separate model from its (numerical) solution ...
- Generate Functional Mockup Interface (FMI) compliant simulation units
- Currently: many commercial and open (OpenModelica) tools
- Related: Mathworks Simscape, EcosimPro, NMF, gProms, ...


OMEdit - OpenModelica Connection Editor
File
Edit
View
Simulation
FMI
XML
Tools
Help
Libraries Browser
Libraries
Complex
Modelica
ModelicaReference
ModelicaServices
OpenModelica
myRLCnetwork
myRLCnetwork
Writeable
Model
Diagram View
C:/Users/hv/De...yRLCnetwork.mc
Line: 1, Col: 0
resistor
R=100
inductor
L=1e-006
sineInputVoltage
ground1
X: -90.63
Y: 19.89
Welcome
Modeling
Plotting


OMEdit - OpenModelica Connection Editor
File Edit View Simulation FMI XML Tools Help
Libraries Browser
Libraries
Complex
Modelica
ModelicaReference
ModelicaServices
OpenModelica
myRLCnetwork
myRLCnetwork
Writeable Model Modelica Text View C:/Users/hv/Desktop/scribbles/myRLCnetwork.mo Line: 1, Col: 11
1 model myRLCnetwork
2 Modelica.Electrical.Analog.Basic.Resistor resistor(R = 100) annotation(Placement(visible = true, transformation(origin = {-40,60}, extent = {{-10,-10},{10,10}}, rotation = 0)));
3 Modelica.Electrical.Analog.Basic.Inductor inductor(L = 1e-006) annotation(Placement(visible = true, transformation(origin = {0,60}, extent = {{-10,-10},{10,10}}, rotation = 0)));
4 Modelica.Electrical.Analog.Sources.SineVoltage sineInputVoltage(V = 10, freqHz = 50) annotation(Placement(visible = true, transformation(origin = {-80,40}, extent = {{-10,-10},{10,10}}, rotation = -90)));
5 Modelica.Electrical.Analog.Basic.Ground ground1 annotation(Placement(visible = true, transformation(origin = {-80,0}, extent = {{-10,-10},{10,10}}, rotation = 0)));
6 equation
7 connect(resistor.p,sineInputVoltage.p) annotation(Line(points = {{-50,60},{-79.8913,60},{-79.8913,49.4565},{-79.8913,49.4565}}));
8 connect(resistor.n,inductor.p) annotation(Line(points = {{-30,60},{-9.78261,60},{-9.78261,59.2391},{-9.78261,59.2391}}));
9 connect(sineInputVoltage.n,inductor.n) annotation(Line(points = {{-80,30},{-80,29.8913},{10.0543,29.8913},{10.0543,59.7826},{10.0543,59.7826}}));
10 connect(sineInputVoltage.n,ground1.p) annotation(Line(points = {{-80,30},{-80,9.78261},{-79.61960000000001,9.78261},{-79.61960000000001,9.78261}}));
11 annotation(Icon(coordinateSystem(extent = {{-100,-100},{100,100}}, preserveAspectRatio = true, initialScale = 0.1, grid = {2,2})), Diagram(coordinateSystem(extent = {{-100,-100},{100,100}}, preserveAspectRatio = true, initialScale = 0.1, grid = {2,2})));
12 end myRLCnetwork;
X: -90.63
Y: 19.89
Welcome
Modeling
Plotting


Libraries Browser
Libraries
Complex
Modelica
UsersGuide
Blocks
ComplexBlocks
StateGraph
Electrical
Analog
Digital
Machines
MultiPhase
QuasiStationary
Spice3
Magnetic
Mechanics
Fluid
Media
Thermal
Math
ComplexMath
Utilities
Constants
Icons
SIunits
ModelicaReference
ModelicaServices
OpenModelica
myRLCnetwork

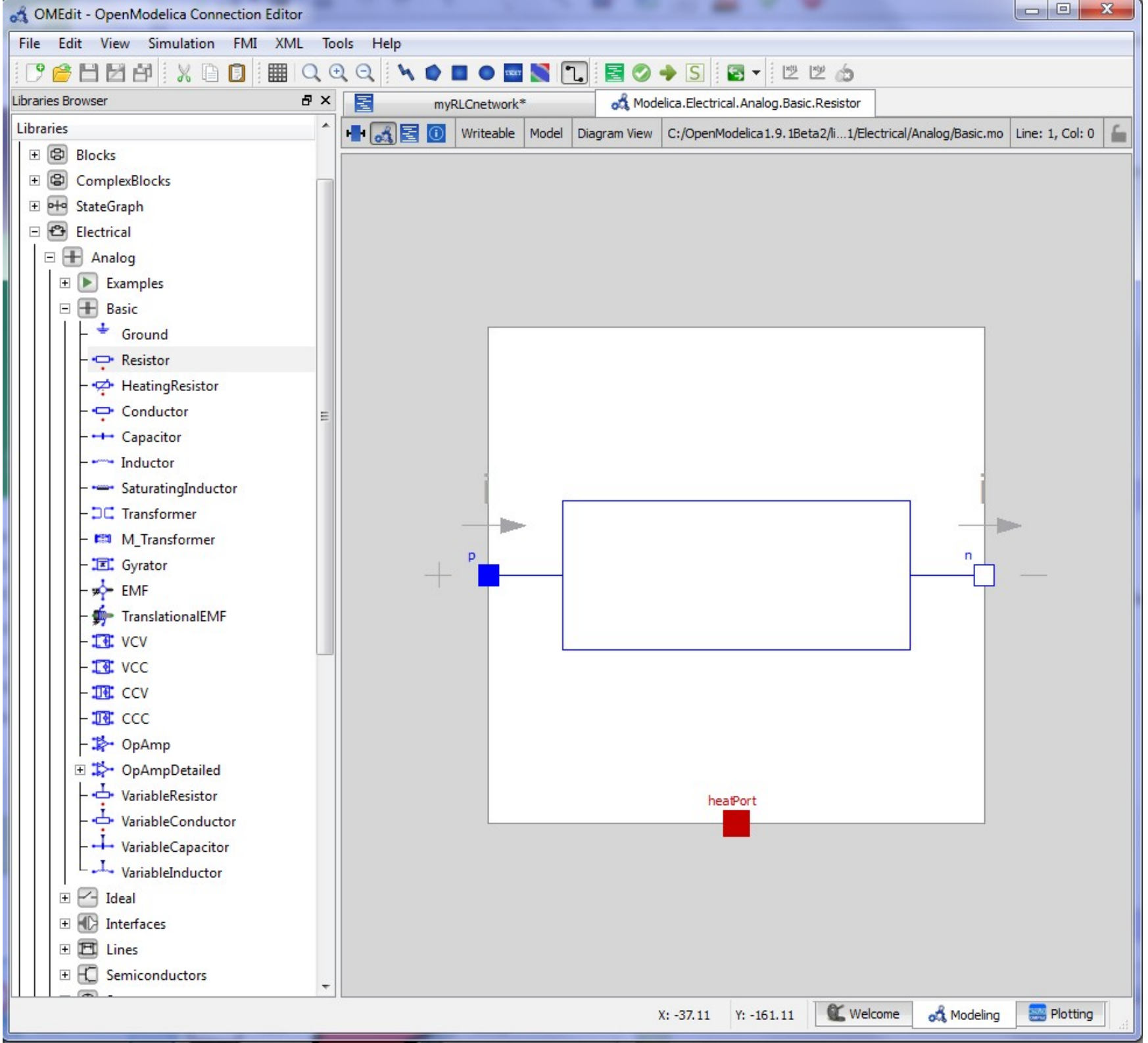

OMEdit - OpenModelica Connection Editor
File Edit View Simulation FMI XML Tools Help
Libraries Browser
Libraries
Blocks
ComplexBlocks
StateGraph
Electrical
Analog
Examples
Basic
Ground
Resistor
HeatingResistor
Conductor
Capacitor
Inductor
SaturatingInductor
Transformer
M_Transformer
Gyrator
EMF
TranslationalEMF
VCV
VCC
CCV
CCC
OpAmp
OpAmpDetailed
VariableResistor
VariableConductor
VariableCapacitor
VariableInductor
Ideal
Interfaces
Lines
Semiconductors
myRLCnetwork*
Modelica.Electrical.Analog.Basic.Resistor
Writeable
Model
Modelica Text View
C:/OpenModelica1.9.1Beta2/lib/omlibrary/Modelica 3.2.1/Electrical/Analog/Basic.mo
Line: 1, Col: 0
model Resistor "Ideal linear electrical resistor"
parameter Modelica.SIunits.Resistance R(start = 1) "Resistance at temperature T_ref";
parameter Modelica.SIunits.Temperature T_ref = 300.15 "Reference temperature";
parameter Modelica.SIunits.LinearTemperatureCoefficient alpha = 0 "Temperature coefficient of resistance (R_actual = R*(1 + alpha*(T_heatPort - T_ref))";
extends Modelica.Electrical.Analog.Interfaces.OnePort;
extends Modelica.Electrical.Analog.Interfaces.ConditionalHeatPort(T = T_ref);
Modelica.SIunits.Resistance R_actual "Actual resistance = R*(1 + alpha*(T_heatPort - T_ref))";
equation
assert(1 + alpha * (T_heatPort - T_ref) >= Modelica.Constants.eps, "Temperature outside scope of model!");
R_actual = R * (1 + alpha * (T_heatPort - T_ref));
v = R_actual * i;
LossPower = v * i;
annotation(Documentation(info = "<html>
The linear resistor connects the branch voltage v with the branch current i by i*R = v. The Resistance R is allowed to be positive, zero, or negative.
</html>", revisions = "<html>

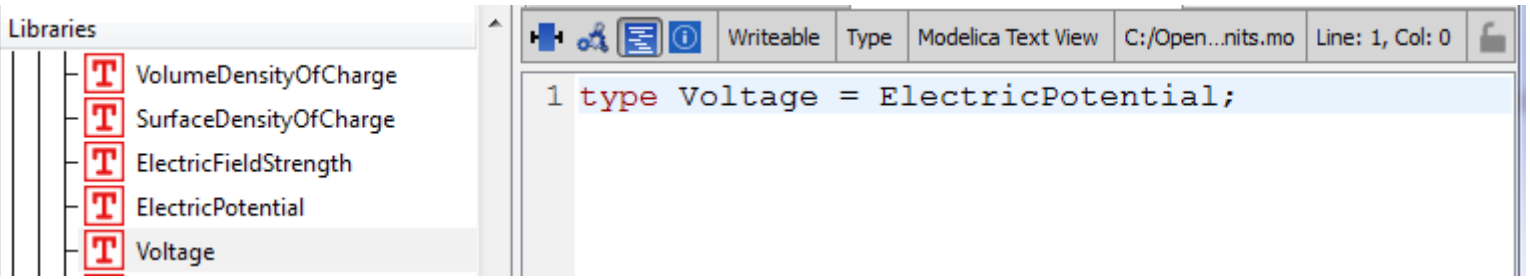
 August 07, 2009
by Anton Haumer
 temperature dependency of resistance added

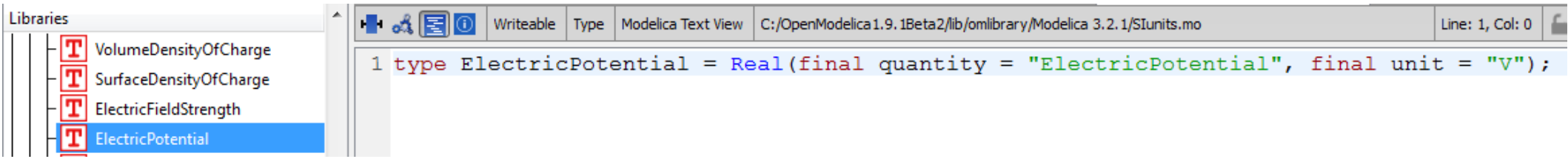
 March 11, 2009
by Christoph Clauss
 conditional heat port added

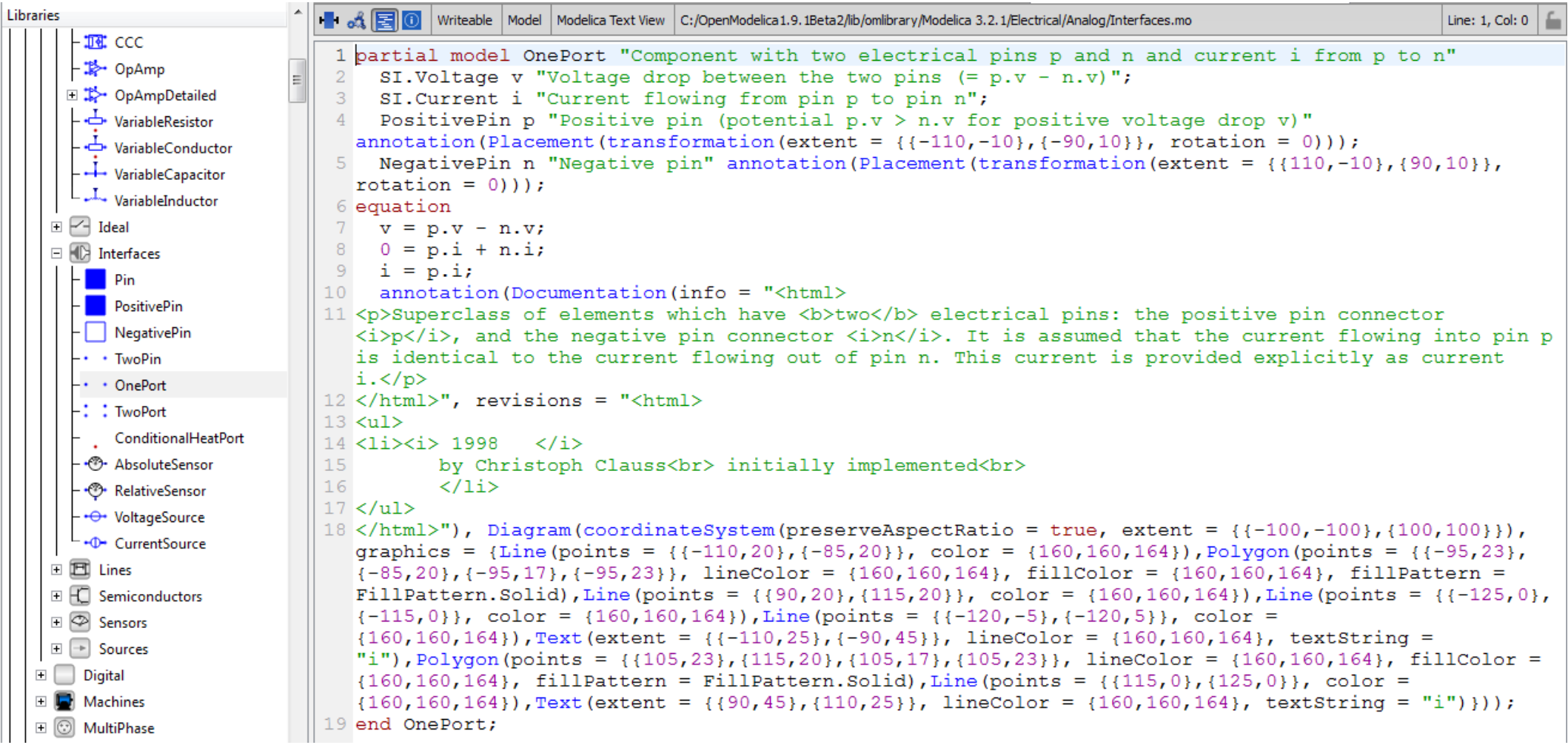
 1998
by Christoph Clauss
 initially implemented

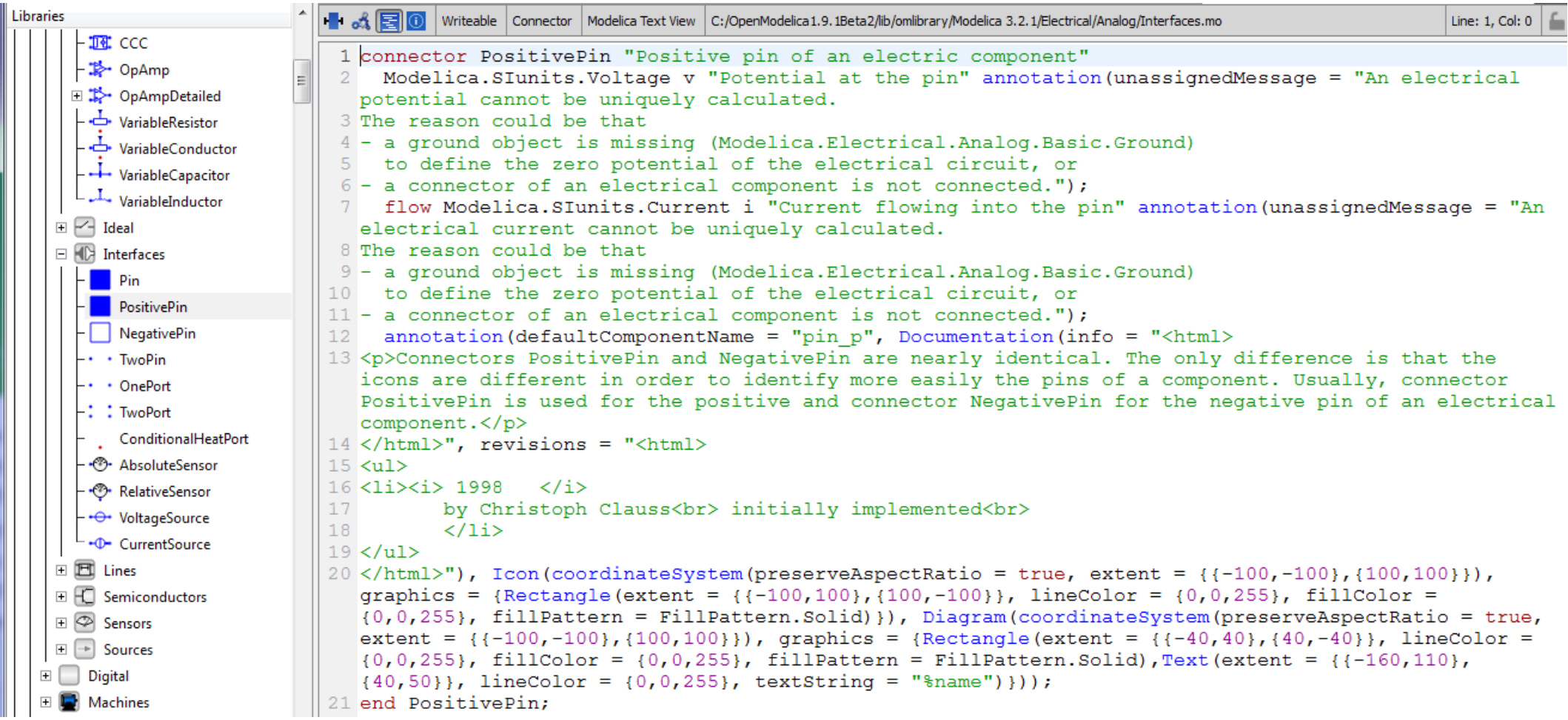
</html>"), Icon(coordinateSystem(preserveAspectRatio = true, extent = {{-100,-100},{100,100}}), graphics = {Rectangle(extent = {{-70,30},{70,-30}}, lineColor = {0,0,255}, fillColor = {255,255,255}, fillPattern = FillPattern.Solid),Line(points = {{-90,0},{-70,0}}, color = {0,0,255}),Line(points = {{70,0},{90,0}}, color = {0,0,255}),Text(extent = {{-144,-40},{142,-72}}, lineColor = {0,0,0}, textString = "R=%R"),Line(visible = useHeatPort, points = {{0,-100},{0,-30}}, color = {127,0,0}, smooth = Smooth.None, pattern = LinePattern.Dot),Text(extent = {{-152,87},{148,47}}, textString = "%name", lineColor = {0,0,255})}),
Diagram(coordinateSystem(preserveAspectRatio = true, extent = {{-100,-100},{100,100}}), graphics = {Rectangle(extent = {{-70,30},{70,-30}}, lineColor = {0,0,255}),Line(points = {{-96,0},{-70,0}}, color = {0,0,255}),Line(points = {{70,0},{96,0}}, color = {0,0,255})}));
end Resistor;
X: -15.03
Y: 154.06
Welcome
Modeling
Plotting


OMEdit - OpenModelica Connection Editor
File Edit View Simulation FMI XML Tools Help
Libraries Browser
Libraries
Blocks
ComplexBlocks
StateGraph
Electrical
Analog
Examples
Basic
Ground
Resistor
HeatingResistor
Conductor
Capacitor
Inductor
SaturatingInductor
Transformer
M_Transformer
Gyrator
EMF
TranslationalEMF
VCV
VCC
CCV
CCC
OpAmp
OpAmpDetailed
VariableResistor
VariableConductor
VariableCapacitor
VariableInductor
Ideal
Interfaces
Lines
Semiconductors
myRLCnetwork*
Modelica.Electrical.Analog.Basic.Resistor
Writeable
Model
Diagram View
C:/OpenModelica1.9.1Beta2/li...1/Electrical/Analog/Basic.mo
Line: 1, Col: 0
p
n
heatPort
X: -37.11
Y: -161.11
Welcome
Modeling
Plotting


Libraries
VolumeDensityOfCharge
SurfaceDensityOfCharge
ElectricFieldStrength
ElectricPotential
Voltage
Writeable
Type
Modelica Text View
C:/Open...nits.mo
Line: 1, Col: 0
1 type Voltage = ElectricPotential;

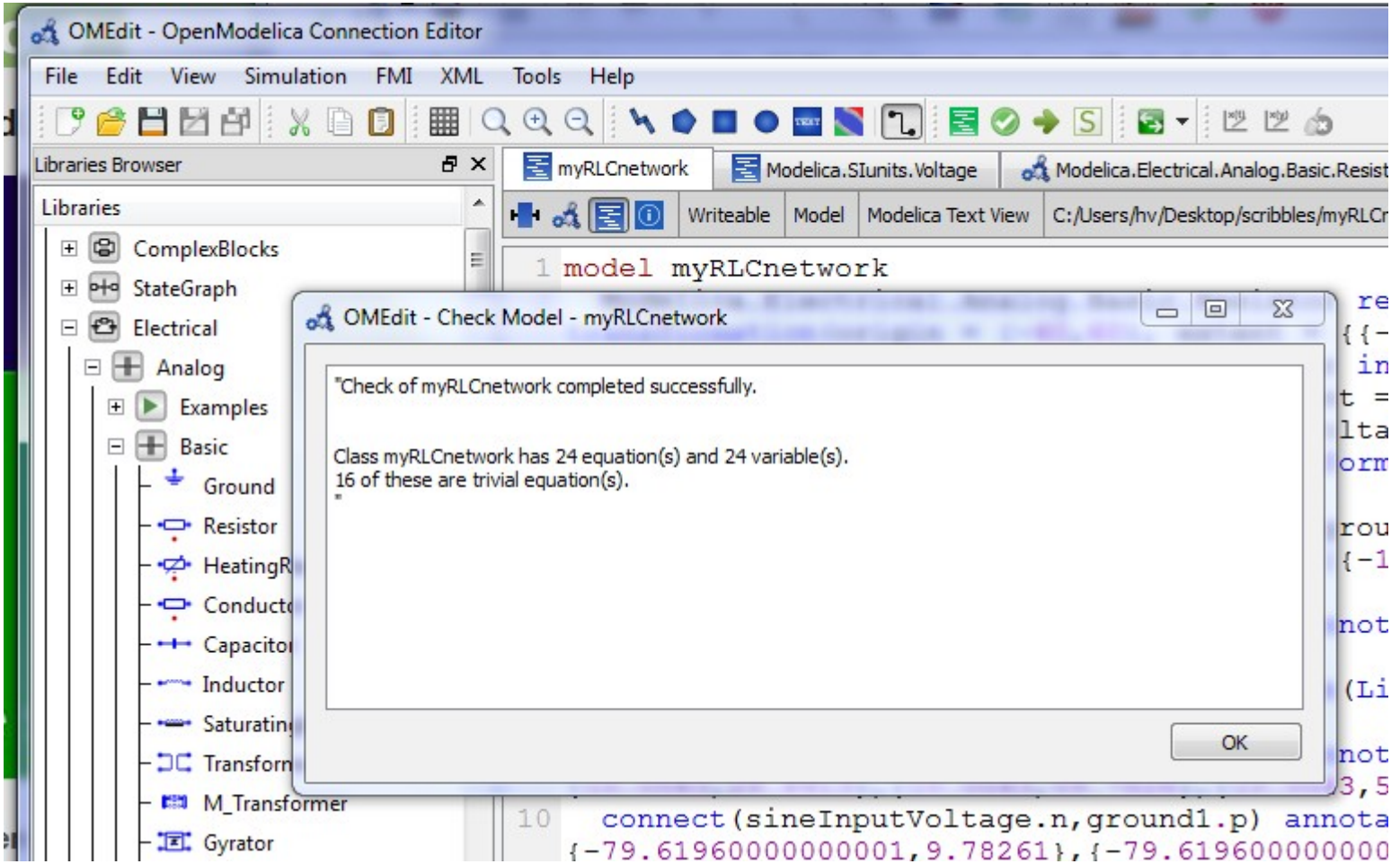

Libraries
VolumeDensityOfCharge
SurfaceDensityOfCharge
ElectricFieldStrength
ElectricPotential
Writeable
Type
Modelica Text View
C:/OpenModelica1.9.1Beta2/lib/omlibrary/Modelica 3.2.1/SIunits.mo
Line: 1, Col: 0
1 type ElectricPotential = Real(final quantity = "ElectricPotential", final unit = "V");


Libraries
CCC
OpAmp
OpAmpDetailed
VariableResistor
VariableConductor
VariableCapacitor
VariableInductor
Ideal
Interfaces
Pin
PositivePin
NegativePin
TwoPin
OnePort
TwoPort
ConditionalHeatPort
AbsoluteSensor
RelativeSensor
VoltageSource
CurrentSource
Lines
Semiconductors
Sensors
Sources
Digital
Machines
MultiPhase
Writeable
Model
Modelica Text View
C:/OpenModelica1.9.1Beta2/lib/omlibrary/Modelica 3.2.1/Electrical/Analog/Interfaces.mo
Line: 1, Col: 0
1 partial model OnePort "Component with two electrical pins p and n and current i from p to n"
2 SI.Voltage v "Voltage drop between the two pins (= p.v - n.v)";
3 SI.Current i "Current flowing from pin p to pin n";
4 PositivePin p "Positive pin (potential p.v > n.v for positive voltage drop v)"
annotation(Placement(transformation(extent = {{-110,-10},{-90,10}}, rotation = 0)));
5 NegativePin n "Negative pin" annotation(Placement(transformation(extent = {{110,-10},{90,10}},
rotation = 0)));
6 equation
7 v = p.v - n.v;
8 0 = p.i + n.i;
9 i = p.i;
10 annotation(Documentation(info = "<html>
11 Superclass of elements which have two electrical pins: the positive pin connector
p, and the negative pin connector n. It is assumed that the current flowing into pin p
is identical to the current flowing out of pin n. This current is provided explicitly as current
i.
12 </html>", revisions = "<html>
13
14 <li> 1998
15 by Christoph Clauss
 initially implemented
16
17
18 </html>"), Diagram(coordinateSystem(preserveAspectRatio = true, extent = {{-100,-100},{100,100}}),
graphics = {Line(points = {{-110,20},{-85,20}}, color = {160,160,164}),Polygon(points = {{-95,23},
{-85,20},{-95,17},{-95,23}}, lineColor = {160,160,164}, fillColor = {160,160,164}, fillPattern =
FillPattern.Solid),Line(points = {{90,20},{115,20}}, color = {160,160,164}),Line(points = {{-125,0},
{-115,0}}, color = {160,160,164}),Line(points = {{-120,-5},{-120,5}}, color =
{160,160,164}),Text(extent = {{-110,25},{-90,45}}, lineColor = {160,160,164}, textString =
"i"),Polygon(points = {{105,23},{115,20},{105,17},{105,23}}, lineColor = {160,160,164}, fillColor =
{160,160,164}, fillPattern = FillPattern.Solid),Line(points = {{115,0},{125,0}}, color =
{160,160,164}),Text(extent = {{90,45},{110,25}}, lineColor = {160,160,164}, textString = "i")}));
19 end OnePort;


Libraries
CCC
OpAmp
OpAmpDetailed
VariableResistor
VariableConductor
VariableCapacitor
VariableInductor
Ideal
Interfaces
Pin
PositivePin
NegativePin
TwoPin
OnePort
TwoPort
ConditionalHeatPort
AbsoluteSensor
RelativeSensor
VoltageSource
CurrentSource
Lines
Semiconductors
Sensors
Sources
Digital
Machines
Writeable
Connector
Modelica Text View
C:/OpenModelica1.9.1Beta2/lib/omlibrary/Modelica 3.2.1/Electrical/Analog/Interfaces.mo
Line: 1, Col: 0
connector PositivePin "Positive pin of an electric component"
Modelica.SIunits.Voltage v "Potential at the pin" annotation(unassignedMessage = "An electrical potential cannot be uniquely calculated.
The reason could be that
- a ground object is missing (Modelica.Electrical.Analog.Basic.Ground)
to define the zero potential of the electrical circuit, or
- a connector of an electrical component is not connected.");
flow Modelica.SIunits.Current i "Current flowing into the pin" annotation(unassignedMessage = "An electrical current cannot be uniquely calculated.
The reason could be that
- a ground object is missing (Modelica.Electrical.Analog.Basic.Ground)
to define the zero potential of the electrical circuit, or
- a connector of an electrical component is not connected.");
annotation(defaultComponentName = "pin_p", Documentation(info = "<html>
Connectors PositivePin and NegativePin are nearly identical. The only difference is that the icons are different in order to identify more easily the pins of a component. Usually, connector PositivePin is used for the positive and connector NegativePin for the negative pin of an electrical component.
</html>", revisions = "<html>

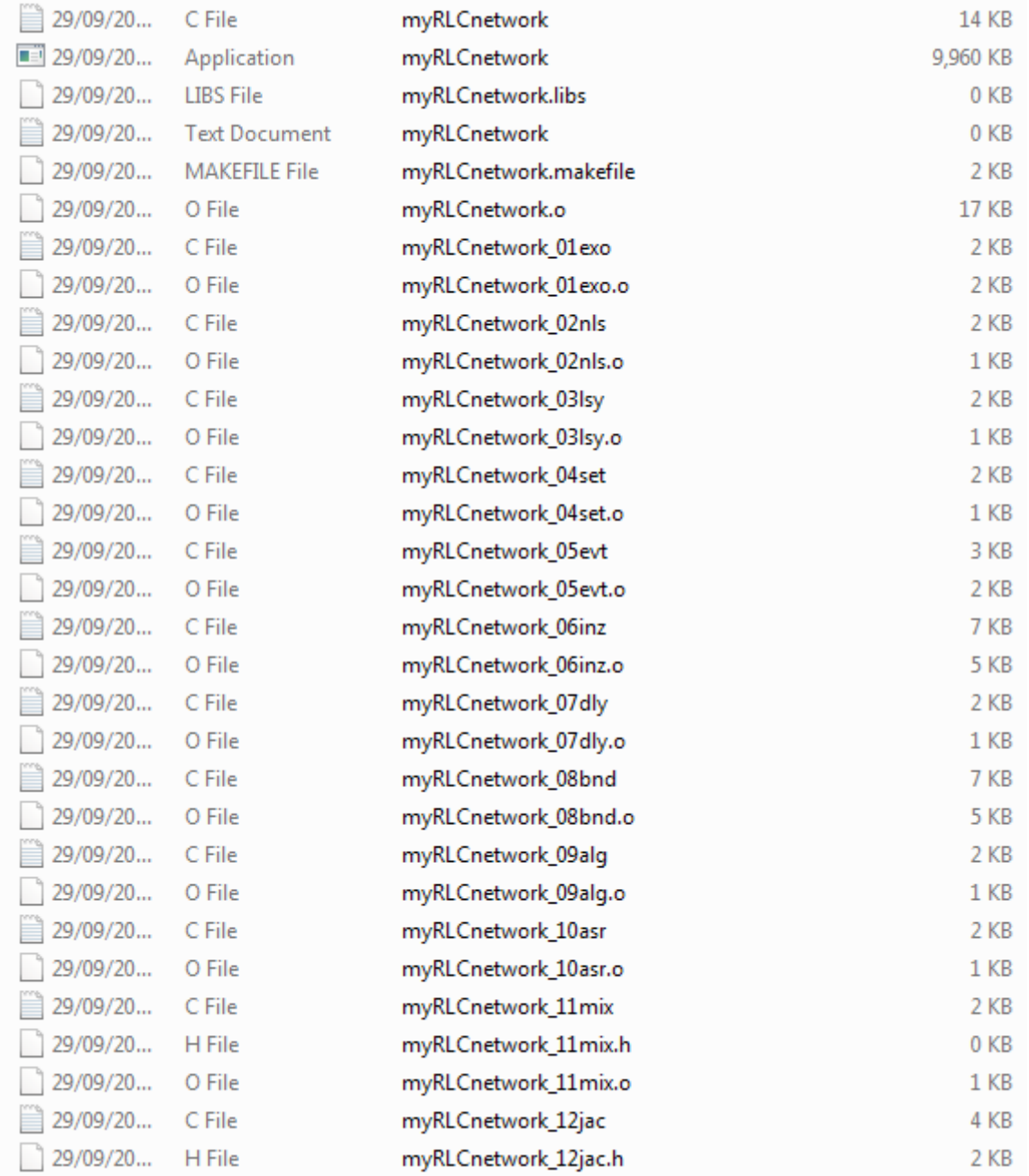
 1998
by Christoph Clauss
 initially implemented

</html>"), Icon(coordinateSystem(preserveAspectRatio = true, extent = {{-100,-100},{100,100}}), graphics = {Rectangle(extent = {{-100,100},{100,-100}}, lineColor = {0,0,255}, fillColor = {0,0,255}, fillPattern = FillPattern.Solid)}), Diagram(coordinateSystem(preserveAspectRatio = true, extent = {{-100,-100},{100,100}}), graphics = {Rectangle(extent = {{-40,40},{40,-40}}, lineColor = {0,0,255}, fillColor = {0,0,255}, fillPattern = FillPattern.Solid),Text(extent = {{-160,110},{40,50}}, lineColor = {0,0,255}, textString = "%name")}));
end PositivePin;

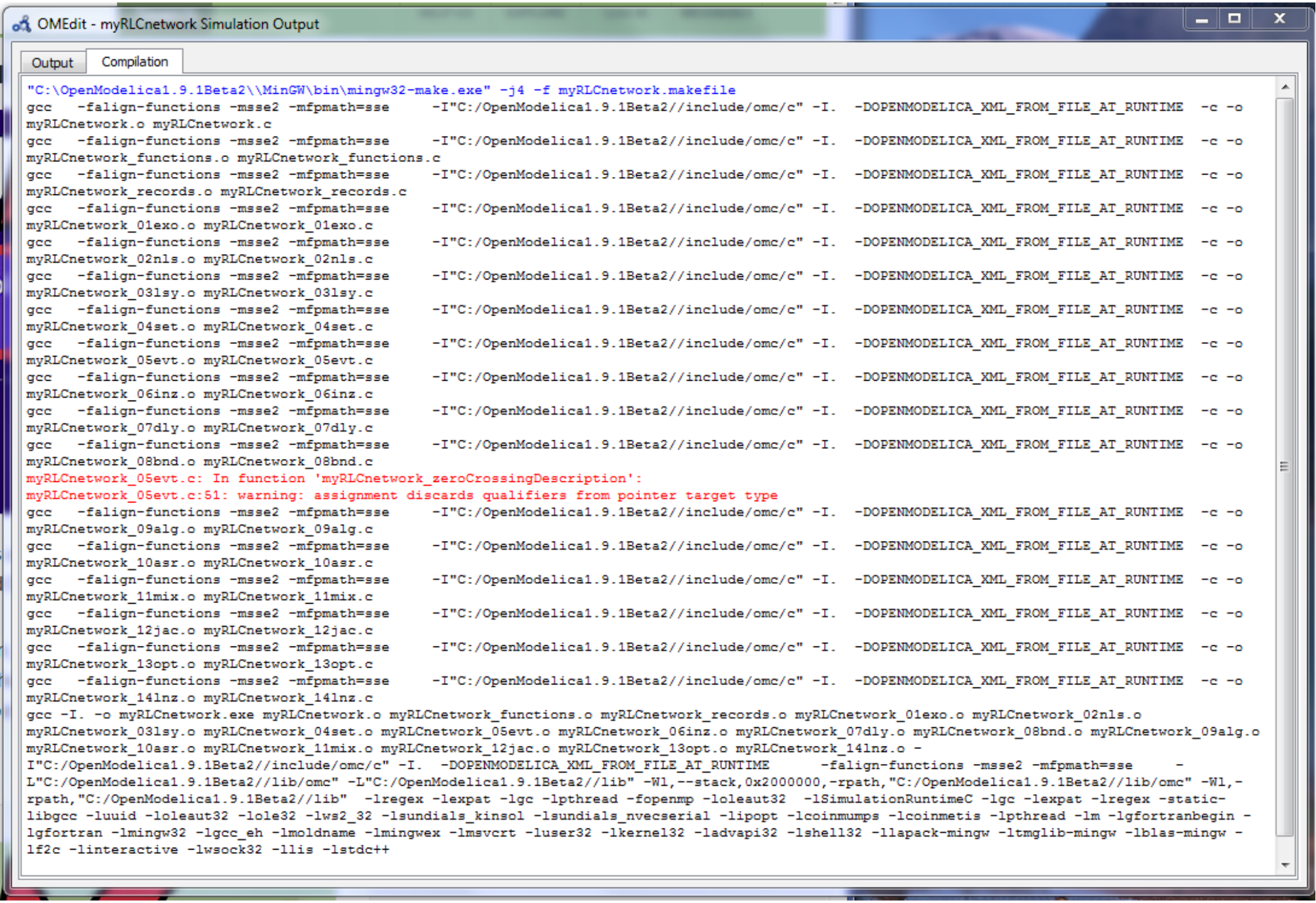

OMEdit - OpenModelica Connection Editor
File Edit View Simulation FMI XML Tools Help
Libraries Browser
Libraries
ComplexBlocks
StateGraph
Electrical
Analog
Examples
Basic
Ground
Resistor
Inductor
M_Transformer
Gyrator
myRLCnetwork
Modelica.SIunits.Voltage
Writeable
Model
Modelica Text View
1 model myRLCnetwork
10 connect(sineInputVoltage.n,ground1.p) annota
{-79.61960000000001,9.78261},{-79.619600000000
OMEdit - Check Model - myRLCnetwork
"Check of myRLCnetwork completed successfully.
Class myRLCnetwork has 24 equation(s) and 24 variable(s).
16 of these are trivial equation(s).
"
OK

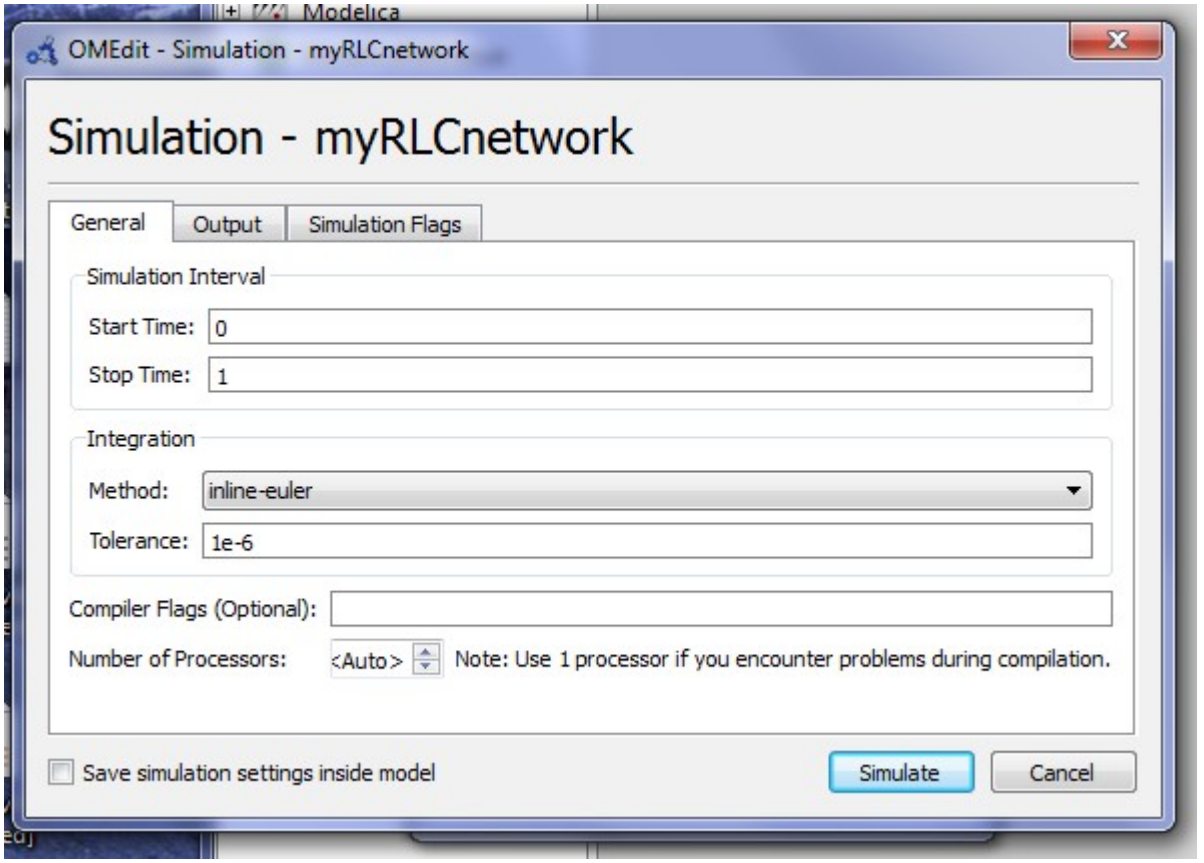
```
class myRLCnetwork
  Real resistor.v(quantity = "ElectricPotential", unit = "V") "Voltage drop between the two pins (= p.v - n.v)";
  Real resistor.i(quantity = "ElectricCurrent", unit = "A") "Current flowing from pin p to pin n";
  Real resistor.p.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real resistor.p.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  Real resistor.n.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real resistor.n.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  parameter Boolean resistor.useHeatPort = false "=true, if HeatPort is enabled";
  Real resistor.LossPower(quantity = "Power", unit = "W") "Loss power leaving component via HeatPort";
  Real resistor.T_heatPort(quantity = "ThermodynamicTemperature", unit = "K", displayUnit = "degC", min = 0.0, start = 288.15, nominal =
300.0) "Temperature of HeatPort";
  parameter Real resistor.R(quantity = "Resistance", unit = "Ohm", start = 1.0) = 100.0 "Resistance at temperature T_ref";
  parameter Real resistor.T_ref(quantity = "ThermodynamicTemperature", unit = "K", displayUnit = "degC", min = 0.0, start = 288.15,
nominal = 300.0) = 300.15 "Reference temperature";
  parameter Real resistor.alpha(quantity = "LinearTemperatureCoefficient", unit = "1/K") = 0.0 "Temperature coefficient of resistance
(R_actual = R*(1 + alpha*(T_heatPort - T_ref))";
  Real resistor.R_actual(quantity = "Resistance", unit = "Ohm") "Actual resistance = R*(1 + alpha*(T_heatPort - T_ref))";
  parameter Real resistor.T(quantity = "ThermodynamicTemperature", unit = "K", displayUnit = "degC", min = 0.0, start = 288.15, nominal =
300.0) = resistor.T_ref "Fixed device temperature if useHeatPort = false";
  Real inductor.v(quantity = "ElectricPotential", unit = "V") "Voltage drop between the two pins (= p.v - n.v)";
  Real inductor.i(quantity = "ElectricCurrent", unit = "A", start = 0.0) "Current flowing from pin p to pin n";
  Real inductor.p.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real inductor.p.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  Real inductor.n.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real inductor.n.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  parameter Real inductor.L(quantity = "Inductance", unit = "H", start = 1.0) = 1e-006 "Inductance";
  Real sineInputVoltage.v(quantity = "ElectricPotential", unit = "V") "Voltage drop between the two pins (= p.v - n.v)";
  Real sineInputVoltage.i(quantity = "ElectricCurrent", unit = "A") "Current flowing from pin p to pin n";
  Real sineInputVoltage.p.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real sineInputVoltage.p.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  Real sineInputVoltage.n.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real sineInputVoltage.n.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  parameter Real sineInputVoltage.offset(quantity = "ElectricPotential", unit = "V") = 0.0 "Voltage offset";
  parameter Real sineInputVoltage.startTime(quantity = "Time", unit = "s") = 0.0 "Time offset";
  parameter Real sineInputVoltage.V(quantity = "ElectricPotential", unit = "V", start = 1.0) = 10.0 "Amplitude of sine wave";
  parameter Real sineInputVoltage.phase(quantity = "Angle", unit = "rad", displayUnit = "deg") = 0.0 "Phase of sine wave";
  parameter Real sineInputVoltage.freqHz(quantity = "Frequency", unit = "Hz", start = 1.0) = 50.0 "Frequency of sine wave";
  output Real sineInputVoltage.signalSource.y "Connector of Real output signal";
  parameter Real sineInputVoltage.signalSource.amplitude = sineInputVoltage.V "Amplitude of sine wave";
  parameter Real sineInputVoltage.signalSource.freqHz(quantity = "Frequency", unit = "Hz", start = 1.0) = sineInputVoltage.freqHz
"Frequency of sine wave";
  parameter Real sineInputVoltage.signalSource.phase(quantity = "Angle", unit = "rad", displayUnit = "deg") = sineInputVoltage.phase
"Phase of sine wave";
  parameter Real sineInputVoltage.signalSource.offset = sineInputVoltage.offset "Offset of output signal";
  parameter Real sineInputVoltage.signalSource.startTime(quantity = "Time", unit = "s") = sineInputVoltage.startTime "Output = offset for
time < startTime";
  protected constant Real sineInputVoltage.signalSource.pi = 3.141592653589793;
  Real ground1.p.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real ground1.p.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
```

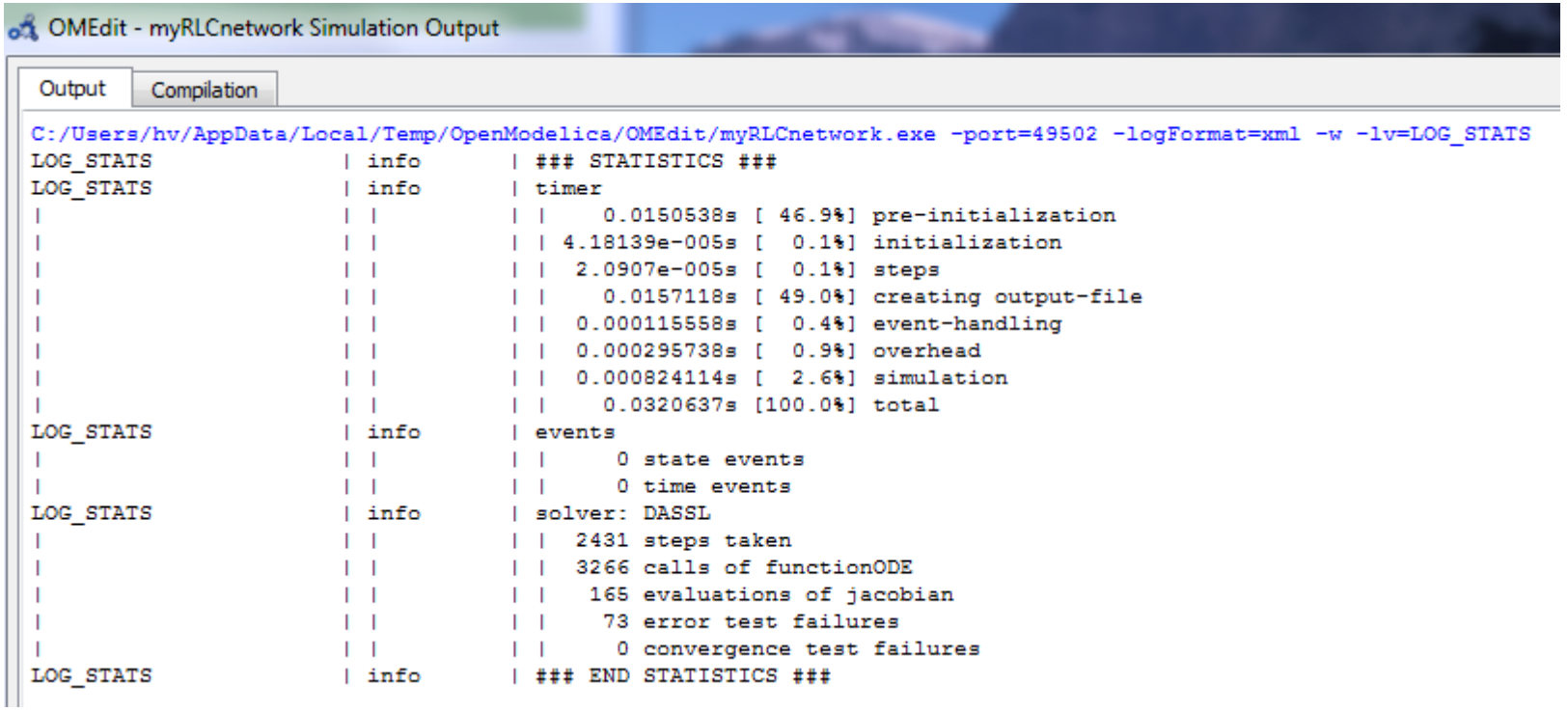
```
equation
  assert(1.0 + resistor.alpha * (resistor.T_heatPort - resistor.T_ref) >= 1e-015,"Temperature outside scope of model!");
  resistor.R_actual = resistor.R * (1.0 + resistor.alpha * (resistor.T_heatPort - resistor.T_ref));
  resistor.v = resistor.R_actual * resistor.i;
  resistor.LossPower = resistor.v * resistor.i;
  resistor.v = resistor.p.v - resistor.n.v;
  0.0 = resistor.p.i + resistor.n.i;
  resistor.i = resistor.p.i;
  resistor.T_heatPort = resistor.T;
  inductor.L * der(inductor.i) = inductor.v;
  inductor.v = inductor.p.v - inductor.n.v;
  0.0 = inductor.p.i + inductor.n.i;
  inductor.i = inductor.p.i;
  sineInputVoltage.signalSource.y = sineInputVoltage.signalSource.offset + (if time <
sineInputVoltage.signalSource.startTime then 0.0 else sineInputVoltage.signalSource.amplitude * sin(6.283185307179586 *
sineInputVoltage.signalSource.freqHz * (time - sineInputVoltage.signalSource.startTime) +
sineInputVoltage.signalSource.phase));
  sineInputVoltage.v = sineInputVoltage.signalSource.y;
  sineInputVoltage.v = sineInputVoltage.p.v - sineInputVoltage.n.v;
  0.0 = sineInputVoltage.p.i + sineInputVoltage.n.i;
  sineInputVoltage.i = sineInputVoltage.p.i;
  ground1.p.v = 0.0;
  resistor.p.i + sineInputVoltage.p.i = 0.0;
  resistor.n.i + inductor.p.i = 0.0;
  inductor.n.i + sineInputVoltage.n.i + ground1.p.i = 0.0;
  resistor.p.v = sineInputVoltage.p.v;
  inductor.p.v = resistor.n.v;
  ground1.p.v = inductor.n.v;
  ground1.p.v = sineInputVoltage.n.v;
end myRLCnetwork;
```

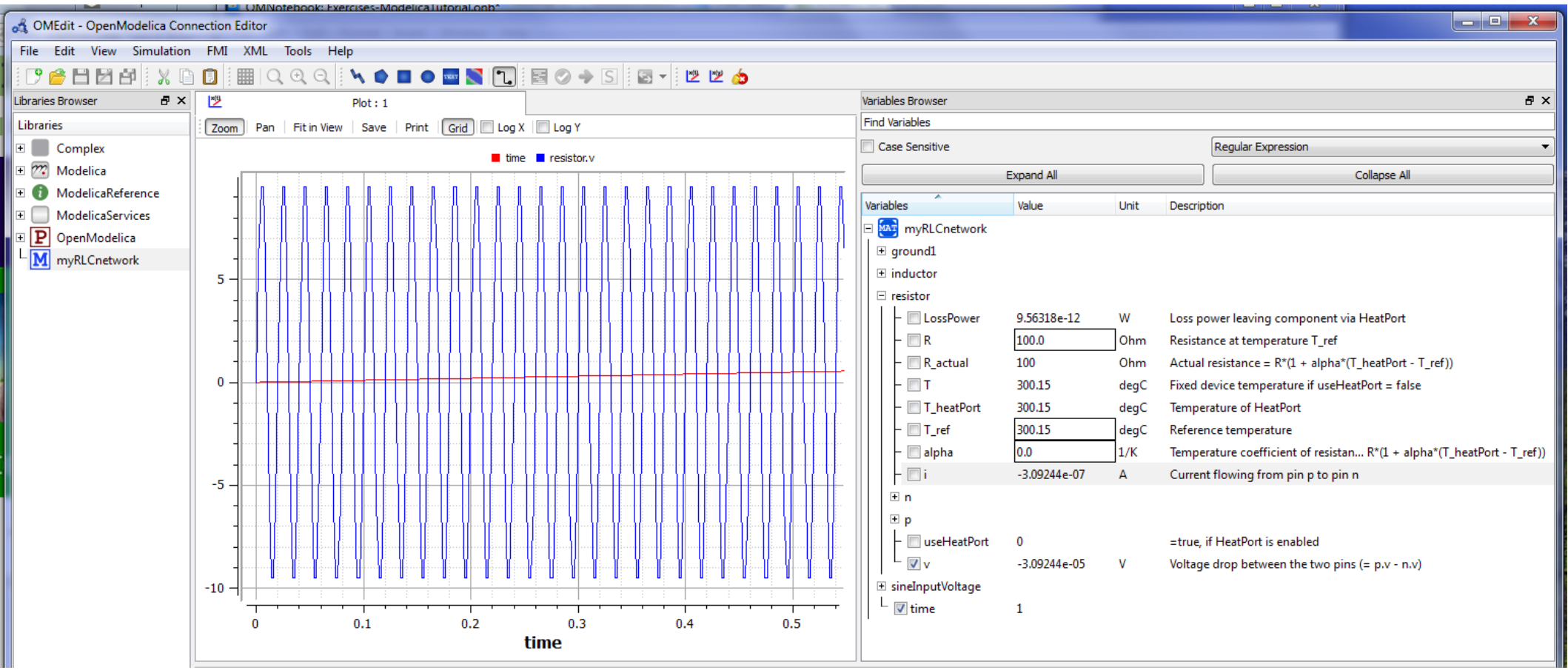


29/09/20...	C File	myRLCnetwork	14 KB
29/09/20...	Application	myRLCnetwork	9,960 KB
29/09/20...	LIBS File	myRLCnetwork.libs	0 KB
29/09/20...	Text Document	myRLCnetwork	0 KB
29/09/20...	MAKEFILE File	myRLCnetwork.makefile	2 KB
29/09/20...	O File	myRLCnetwork.o	17 KB
29/09/20...	C File	myRLCnetwork_01exo	2 KB
29/09/20...	O File	myRLCnetwork_01exo.o	2 KB
29/09/20...	C File	myRLCnetwork_02nls	2 KB
29/09/20...	O File	myRLCnetwork_02nls.o	1 KB
29/09/20...	C File	myRLCnetwork_03lsy	2 KB
29/09/20...	O File	myRLCnetwork_03lsy.o	1 KB
29/09/20...	C File	myRLCnetwork_04set	2 KB
29/09/20...	O File	myRLCnetwork_04set.o	1 KB
29/09/20...	C File	myRLCnetwork_05evt	3 KB
29/09/20...	O File	myRLCnetwork_05evt.o	2 KB
29/09/20...	C File	myRLCnetwork_06inz	7 KB
29/09/20...	O File	myRLCnetwork_06inz.o	5 KB
29/09/20...	C File	myRLCnetwork_07dly	2 KB
29/09/20...	O File	myRLCnetwork_07dly.o	1 KB
29/09/20...	C File	myRLCnetwork_08bnd	7 KB
29/09/20...	O File	myRLCnetwork_08bnd.o	5 KB
29/09/20...	C File	myRLCnetwork_09alg	2 KB
29/09/20...	O File	myRLCnetwork_09alg.o	1 KB
29/09/20...	C File	myRLCnetwork_10asr	2 KB
29/09/20...	O File	myRLCnetwork_10asr.o	1 KB
29/09/20...	C File	myRLCnetwork_11mix	2 KB
29/09/20...	H File	myRLCnetwork_11mix.h	0 KB
29/09/20...	O File	myRLCnetwork_11mix.o	1 KB
29/09/20...	C File	myRLCnetwork_12jac	4 KB
29/09/20...	H File	myRLCnetwork_12jac.h	2 KB


OMEdit - myRLCnetwork Simulation Output
Output
Compilation
"C:\OpenModelica1.9.1Beta2\\MinGW\bin\mingw32-make.exe" -j4 -f myRLCnetwork.makefile
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork.o myRLCnetwork.c
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_functions.o myRLCnetwork_functions.c
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_records.o myRLCnetwork_records.c
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_01exo.o myRLCnetwork_01exo.c
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_02nls.o myRLCnetwork_02nls.c
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_03lsy.o myRLCnetwork_03lsy.c
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_04set.o myRLCnetwork_04set.c
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_05evt.o myRLCnetwork_05evt.c
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_06inz.o myRLCnetwork_06inz.c
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_07dly.o myRLCnetwork_07dly.c
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_08bnd.o myRLCnetwork_08bnd.c
myRLCnetwork_05evt.c: In function 'myRLCnetwork_zeroCrossingDescription':
myRLCnetwork_05evt.c:51: warning: assignment discards qualifiers from pointer target type
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_09alg.o myRLCnetwork_09alg.c
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_10asr.o myRLCnetwork_10asr.c
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_11mix.o myRLCnetwork_11mix.c
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_12jac.o myRLCnetwork_12jac.c
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_13opt.o myRLCnetwork_13opt.c
gcc -falign-functions -msse2 -mfpmath=sse -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -c -o myRLCnetwork_14lnz.o myRLCnetwork_14lnz.c
gcc -I. -o myRLCnetwork.exe myRLCnetwork.o myRLCnetwork_functions.o myRLCnetwork_records.o myRLCnetwork_01exo.o myRLCnetwork_02nls.o myRLCnetwork_03lsy.o myRLCnetwork_04set.o myRLCnetwork_05evt.o myRLCnetwork_06inz.o myRLCnetwork_07dly.o myRLCnetwork_08bnd.o myRLCnetwork_09alg.o myRLCnetwork_10asr.o myRLCnetwork_11mix.o myRLCnetwork_12jac.o myRLCnetwork_13opt.o myRLCnetwork_14lnz.o -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -falign-functions -msse2 -mfpmath=sse -L"C:/OpenModelica1.9.1Beta2//lib/omc" -L"C:/OpenModelica1.9.1Beta2//lib" -Wl,--stack,0x2000000,-rpath,"C:/OpenModelica1.9.1Beta2//lib/omc" -Wl,-rpath,"C:/OpenModelica1.9.1Beta2//lib" -lregex -lexpat -lgc -lpthread -fopenmp -loleaut32 -lSimulationRuntimeC -lgc -lexpat -lregex -static-libgcc -luuid -loleaut32 -lole32 -lws2_32 -lsundials_kinsol -lsundials_nvecserial -lipopt -lcoinmumps -lcoinmetis -lpthread -lm -lgfortranbegin -lgfortran -lmingw32 -lgcc_eh -lmoldname -lmingwex -lmsvcrt -luser32 -lkernel32 -ladvapi32 -lshell32 -llapack-mingw -ltmglib-mingw -lblas-mingw -lf2c -linteractive -lwsock32 -llis -lstdc++


Modelica
OMEdit - Simulation - myRLCnetwork
Simulation - myRLCnetwork
General
Output
Simulation Flags
Simulation Interval
Start Time: 0
Stop Time: 1
Integration
Method: inline-euler
Tolerance: 1e-6
Compiler Flags (Optional):
Number of Processors: <Auto>
Note: Use 1 processor if you encounter problems during compilation.
Save simulation settings inside model
Simulate
Cancel


OMEdit - myRLCnetwork Simulation Output
Output
Compilation
C:/Users/hv/AppData/Local/Temp/OpenModelica/OMEdit/myRLCnetwork.exe -port=49502 -logFormat=xml -w -lv=LOG_STATS
LOG_STATS | info | ### STATISTICS ###
LOG_STATS | info | timer
| | | | 0.0150538s [46.9%] pre-initialization
| | | | 4.18139e-005s [0.1%] initialization
| | | | 2.0907e-005s [0.1%] steps
| | | | 0.0157118s [49.0%] creating output-file
| | | | 0.000115558s [0.4%] event-handling
| | | | 0.000295738s [0.9%] overhead
| | | | 0.000824114s [2.6%] simulation
| | | | 0.0320637s [100.0%] total
LOG_STATS | info | events
| | | | 0 state events
| | | | 0 time events
LOG_STATS | info | solver: DASSL
| | | | 2431 steps taken
| | | | 3266 calls of functionODE
| | | | 165 evaluations of jacobian
| | | | 73 error test failures
| | | | 0 convergence test failures
LOG_STATS | info | ### END STATISTICS ###

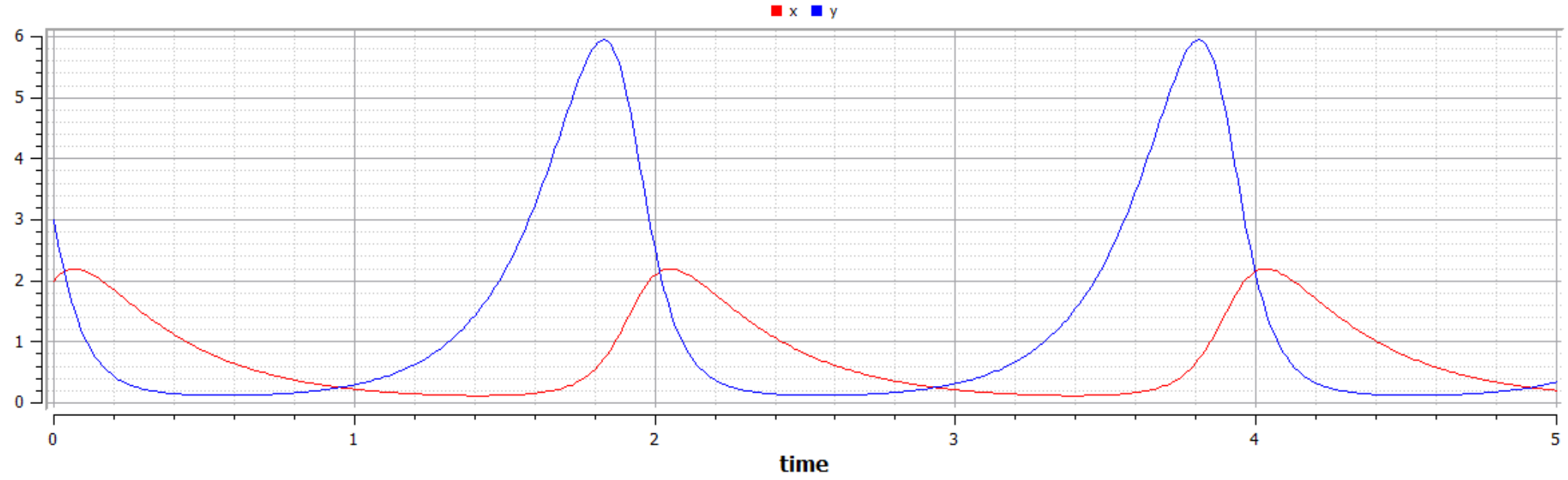

OMEdit - OpenModelica Connection Editor
File Edit View Simulation FMI XML Tools Help
Libraries Browser
Libraries
Complex
Modelica
ModelicaReference
ModelicaServices
OpenModelica
myRLCnetwork
Plot : 1
Zoom Pan Fit in View Save Print Grid Log X Log Y
time resistor.v
5
0
-5
-10
0 0.1 0.2 0.3 0.4 0.5
time
Variables Browser
Find Variables
Case Sensitive
Regular Expression
Expand All
Collapse All
Variables Value Unit Description
myRLCnetwork
ground1
inductor
resistor
LossPower 9.56318e-12 W Loss power leaving component via HeatPort
R 100.0 Ohm Resistance at temperature T_ref
R_actual 100 Ohm Actual resistance = R*(1 + alpha*(T_heatPort - T_ref))
T 300.15 degC Fixed device temperature if useHeatPort = false
T_heatPort 300.15 degC Temperature of HeatPort
T_ref 300.15 degC Reference temperature
alpha 0.0 1/K Temperature coefficient of resistan... R*(1 + alpha*(T_heatPort - T_ref))
i -3.09244e-07 A Current flowing from pin p to pin n
n
p
useHeatPort 0 =true, if HeatPort is enabled
v -3.09244e-05 V Voltage drop between the two pins (= p.v - n.v)
sineInputVoltage
time 1

```
model mySimpleEqnSet "simple equation set"
  Real x(start=2, fixed=true);
  Real y(start=3, fixed=true);
equation
  der(x) = 2*x*y-3*x;
  der(y) = 5*y-7*x*y;
end mySimpleEqnSet;
```

```
plot({x,y})
```

[done]

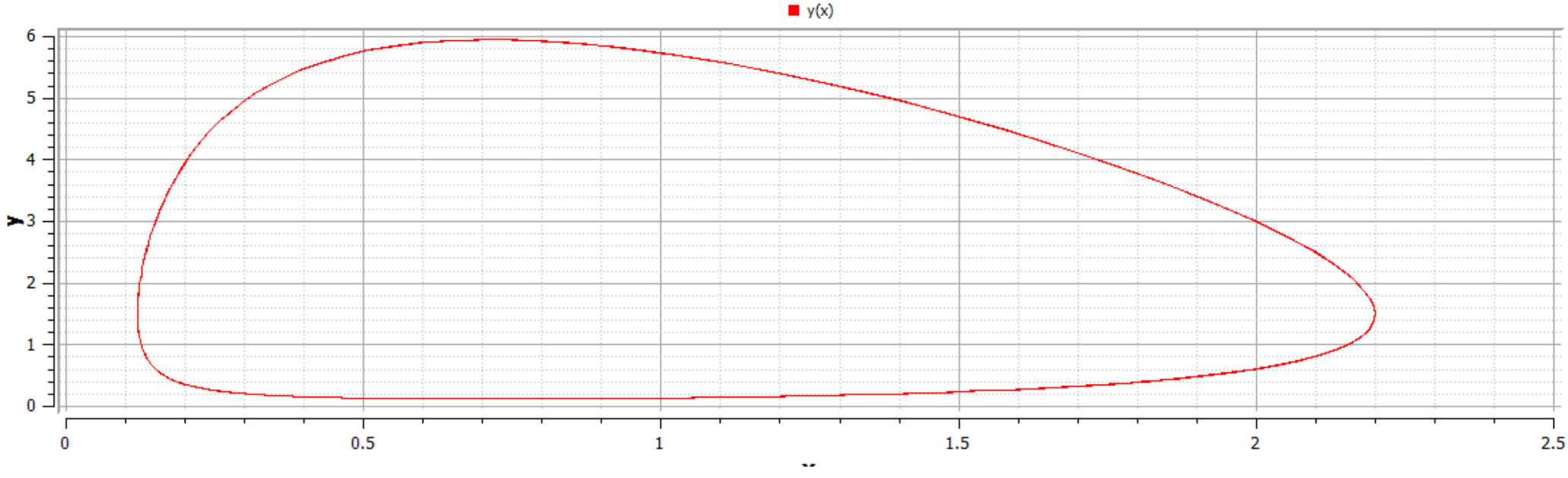
Zoom | Pan | Fit in View | Save | Print | Grid | Log X | Log Y




```
plotParametric(x,y)
```

[done]

Zoom | Pan | Fit in View | Save | Print | Grid | Log X | Log Y



MODELICA

Electrical Types

```
type Time = Real (final quantity="Time", final unit="s");
type ElectricPotential = Real (final quantity="ElectricPotential",
                               final unit="V");
type Voltage = ElectricPotential;
type ElectricCurrent = Real (final quantity="ElectricCurrent",
                             final unit="A");
type Current = ElectricCurrent;
```

Beware: variables are **signals** (functions of **time**)!

MODELICA

Electrical Pin Interface

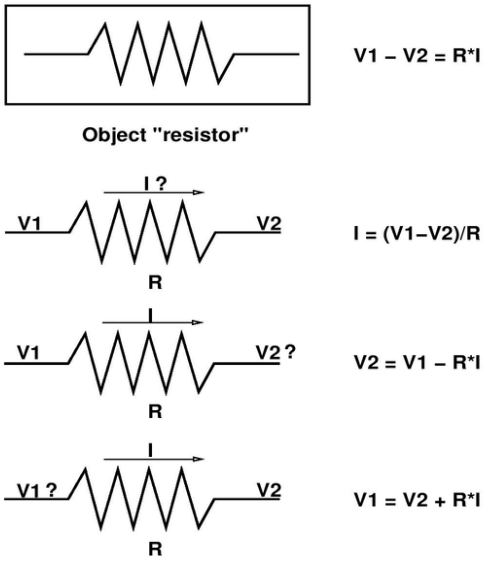
```
connector PositivePin "Positive pin of an electric component"
       Voltage v "Potential at the pin";
  flow Current i "Current flowing into the pin";
end PositivePin;
```

MODELICA

Electrical Port

```
partial model OnePort
  "Component with two electrical pins p and n
   and current i from p to n"
  Voltage v "Voltage drop between the two pins (= p.v - n.v)";
  Current i "Current flowing from pin p to pin n";
  PositivePin p;
  NegativePin n;
equation
  v = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
end OnePort;
```

Object-oriented re-use and causality

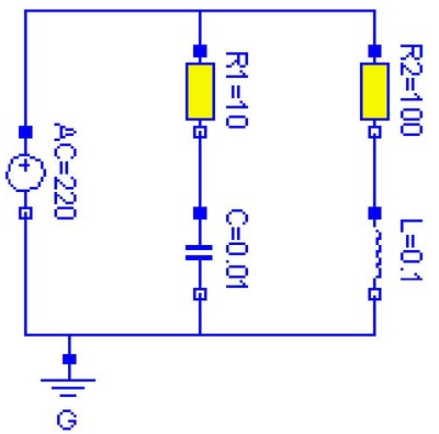


Electrical Resistor

```
model Resistor "Ideal linear electrical resistor"
  extends OnePort;
  parameter Resistance R=1 "Resistance";
  equation
    R*i = v;
end Resistor;
```

The 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;
```



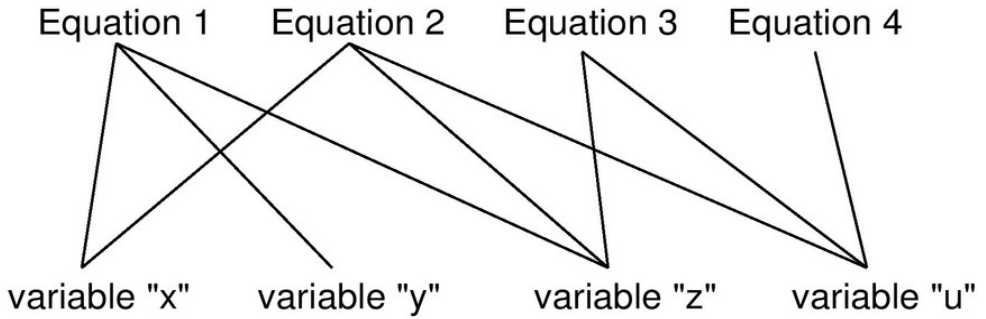
Meaning: set of Differential Algebraic Equations (DAEs) obtained by

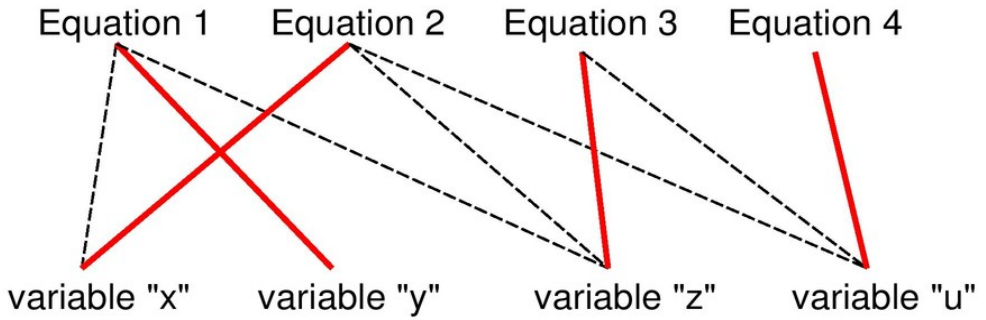
1. expanding inheritance/instantiation
2. flattening hierarchy, unique names
3. expanding connect() into equations (across vs. flow)

Non-causal model
(*e.g.*, from physical conservation laws)

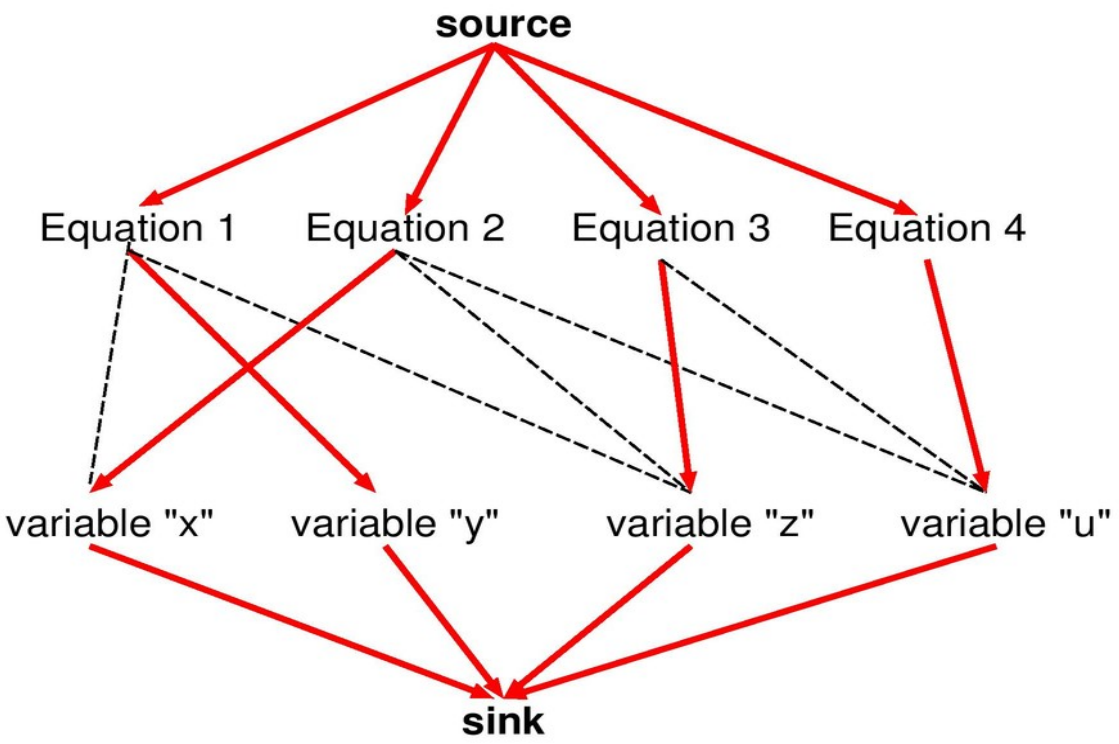
$$\begin{cases} x + y + z &= 0 \quad \text{Equation 1} \\ x + 3z + u^2 &= 0 \quad \text{Equation 2} \\ z - u - 16 &= 0 \quad \text{Equation 3} \\ u - 5 &= 0 \quad \text{Equation 4} \end{cases}$$

Causality assignment: bipartite graph, maximum cardinality matching





Causality assignment: network flow



+ weights for "bad inverses"

Causality assigned

$$\begin{cases} x + \underline{y} + z = 0 & \text{Equation 1} \\ \underline{x} + 3z + u^2 = 0 & \text{Equation 2} \\ \underline{z} - u - 16 = 0 & \text{Equation 3} \\ \underline{u} - 5 = 0 & \text{Equation 4} \end{cases}$$

re-write in causal form

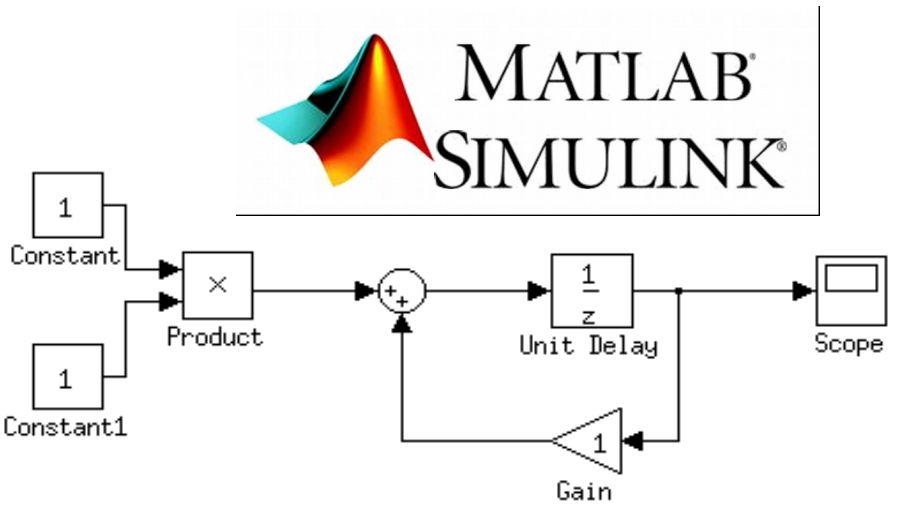
$$\begin{cases} \underline{y} = -x - z \\ \underline{x} = -3z - u^2 \\ \underline{z} = u + 16 \\ \underline{u} = 5 \end{cases}$$

Set of Algebraic Eqns (no cyclic dependencies)

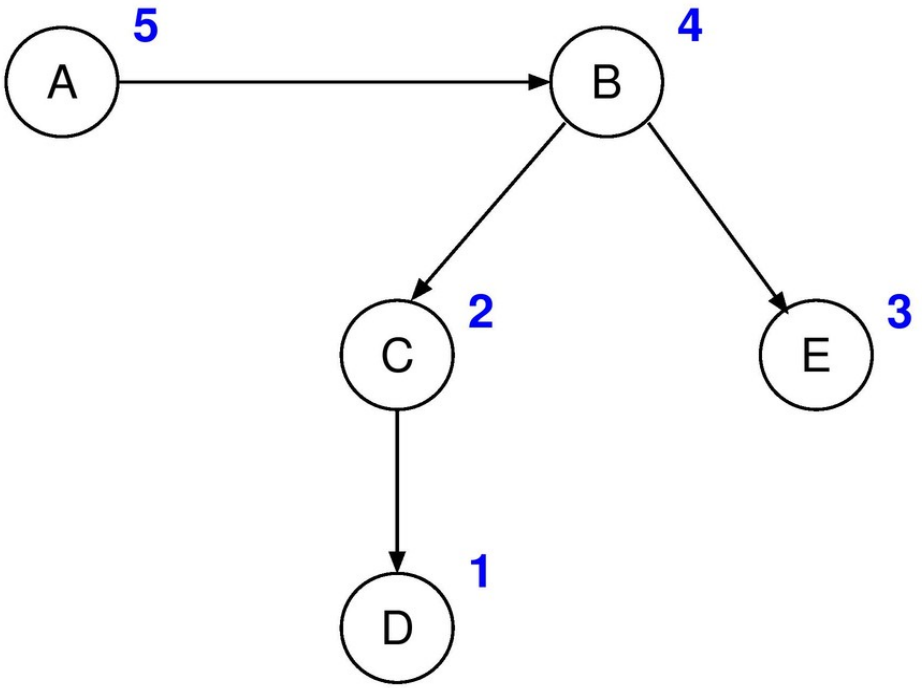
$$\begin{cases} a &= b^2 + 3 \\ b &= sin(c \times e) \\ c &= \sqrt{d - 4.5} \\ d &= \pi/2 \\ e &= u() \end{cases}$$

WRONG:

$$\begin{bmatrix} a &= b^2 + 3 = 3 \\ b &= sin(c \times e) = 0 \\ c &= \sqrt{d - 4.5} = error \\ d &= \pi/2 \\ e &= u() \end{bmatrix}$$



Sorting (no cyclic dependencies)
DFS, postorder numbering of dependency graph



Dependency Cycle (aka Algebraic Loop)

$$\begin{cases} x &= y + 16 \\ y &= -x - z \\ z &= 5 \end{cases}$$

Can *never* be sorted
due to a dependency *cycle* aka *strong component*
(every vertex in the component is reachable from every other)

$$x \to y \to x$$

May be solved implicitly

$$\left[\begin{array}{l} z = 5 \\ \left\{\begin{array}{rcl} x - y & = & -6 \\ x + y & = & -z \end{array}\right. \end{array}\right.$$

Implicit set of n equations in n unknowns.

- non-linear $\rightarrow$ non-linear solver.
- linear $\rightarrow$ numerical or symbolic solution.

Linear: may be solved symbolically (Cramer)

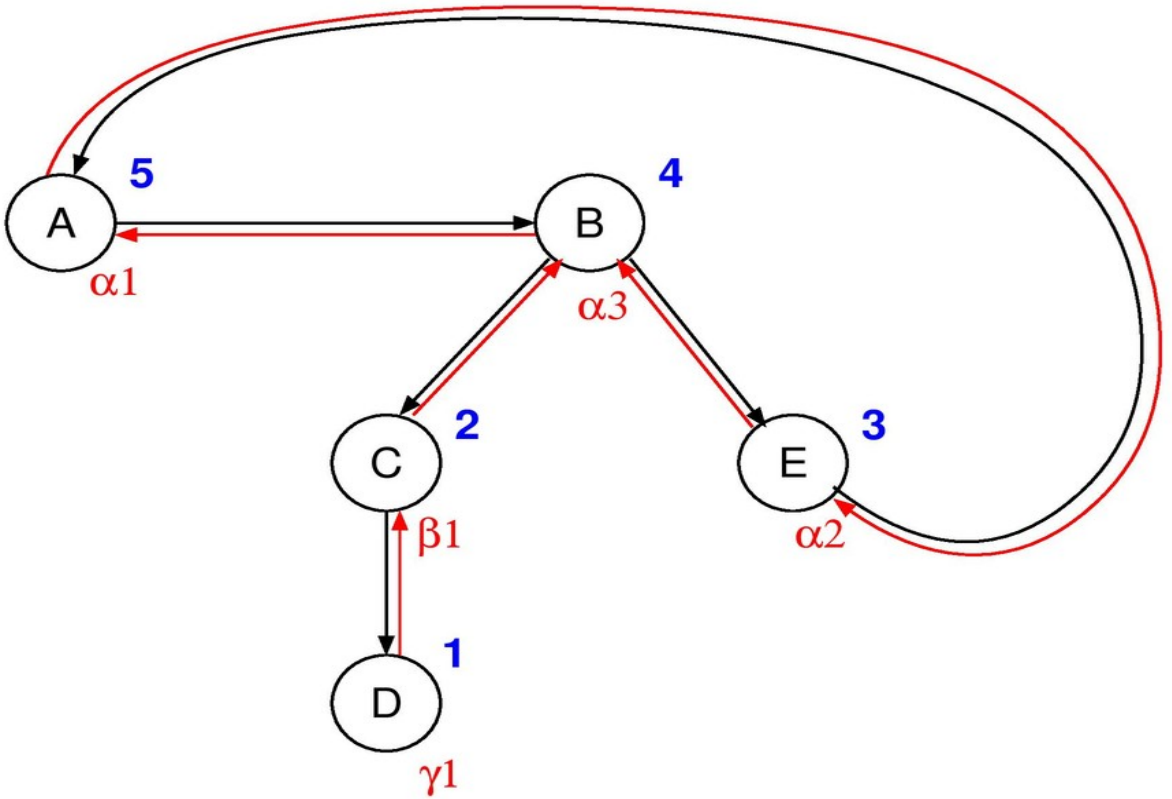
$$x = \frac{\begin{vmatrix} -6 & -1 \\ -z & 1 \end{vmatrix}}{\begin{vmatrix} 1 & -1 \\ 1 & 1 \end{vmatrix}} = \frac{-6-z}{2} \; ; \; y = \frac{\begin{vmatrix} 1 & -6 \\ 1 & -z \end{vmatrix}}{\begin{vmatrix} 1 & -1 \\ 1 & 1 \end{vmatrix}} = \frac{6-z}{2}$$

$$\left[\begin{array}{lcl} z & = & 5 \\ x & = & \frac{-6-z}{2} \\ y & = & \frac{6-z}{2} \end{array} \right.$$

Tarjan's algorithm for Cycle Detection

$$\begin{cases} a &= b^2 + 3 \\ b &= sin(c \times e) \\ c &= \sqrt{d - 4.5} \\ d &= \pi/2 \\ e &= a^2 + u() \end{cases}$$

Algebraic Loop (Cycle) Detection



Algebraic Loop (Cycle) Detection Result

$$\left[\begin{array}{rcl} d & = & \pi/2 \\ c & = & \sqrt{d-4.5} \\ \left\{\begin{array}{rcl} b & = & sin(c \times e) \\ a & = & b^2+3 \\ e & = & a^2+u() \end{array}\right. & & \end{array}\right] ; \left[\begin{array}{rcl} d & = & \pi/2 \\ c & = & \sqrt{d-4.5} \\ \left\{\begin{array}{rcl} b & -sin(c \times e) & = 0 \\ a & -b^2 \quad -3 & = 0 \\ a^2 & -e \quad +u() & = 0 \end{array}\right. & & \end{array}\right]$$

Modelica in Context

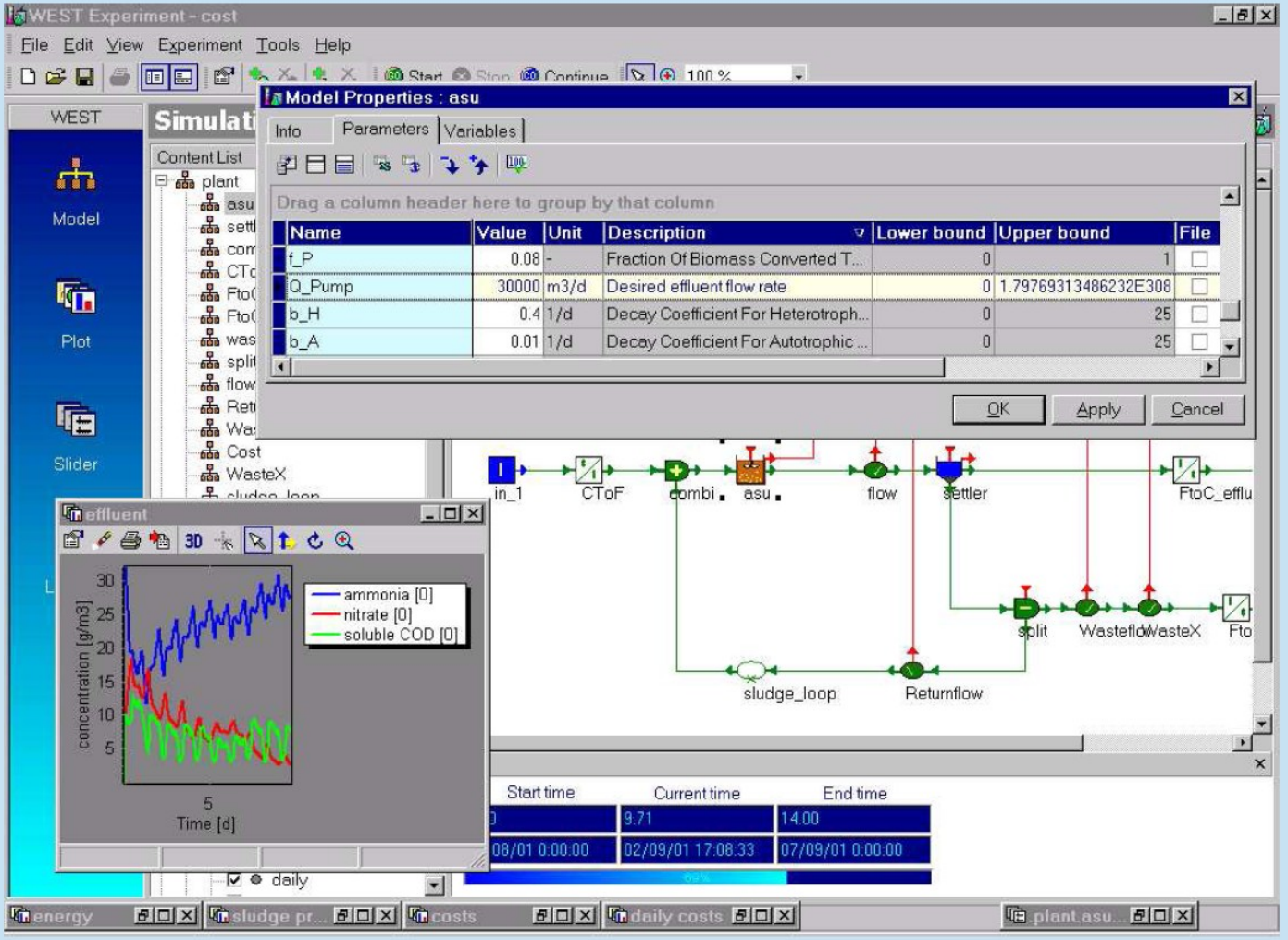
- Domain/Problem-Specific
- Laws of Physics
- Power Flow
- a-causal (Mathematical) ← **Modelica**
- Causal Block Diagrams
- Numerical (Discrete) Approximations
- Computer Numerical (Floating Point, Fixed Point)
- As-Fast-As-Possible vs. Real-time
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)

Operational vs. Denotational (Translational) semantics

NATO's Sarajevo Waste Water Treatment Plant

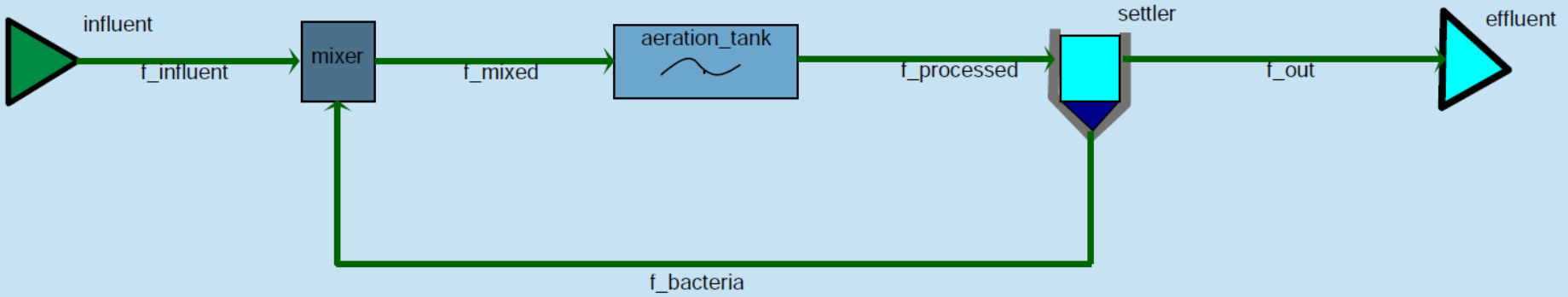
www.nato.int/sfor/cimic/env-pro/waterpla.htm

DS(V)M Environment

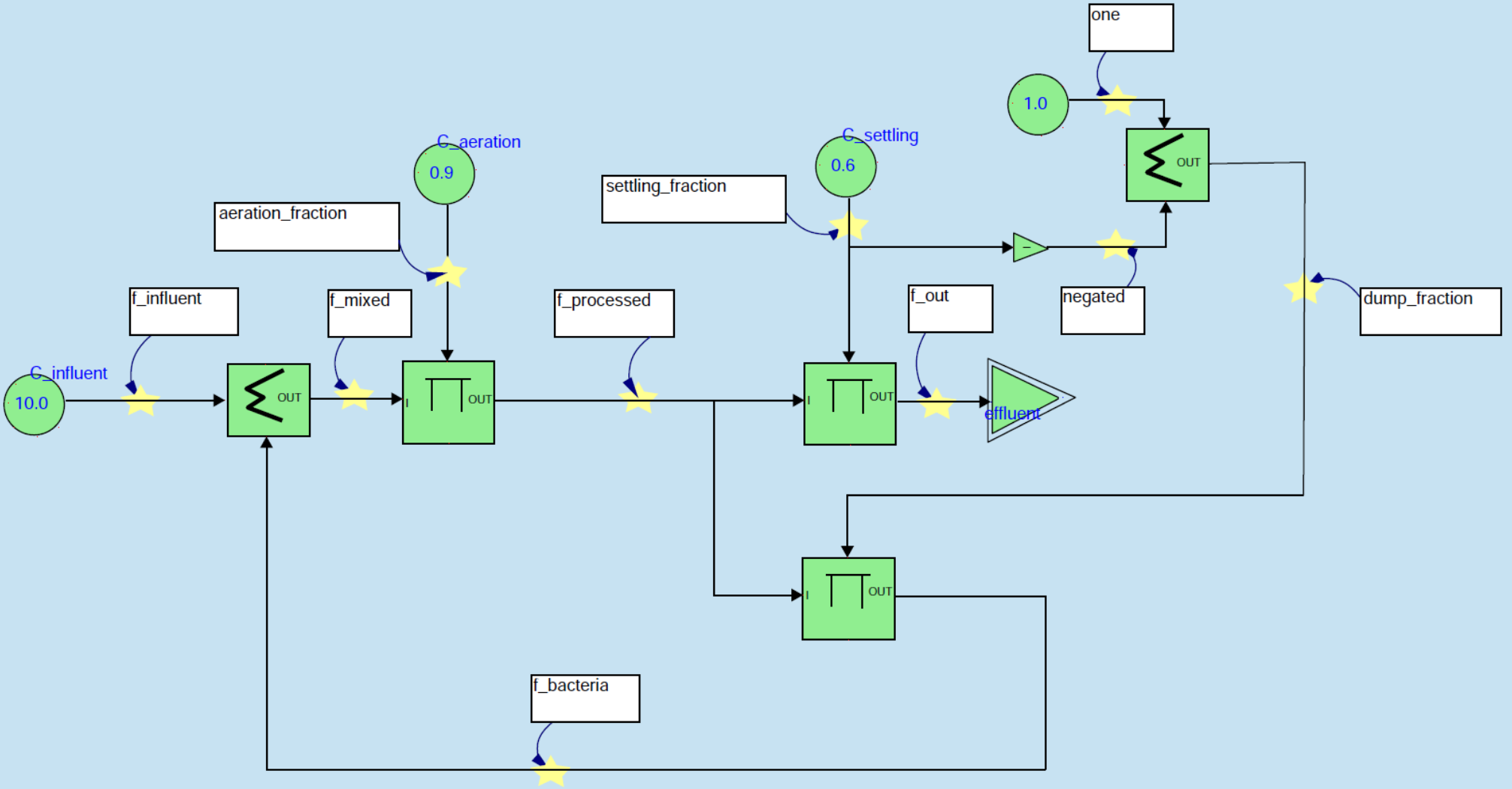


www.hemmis.com/products/west/

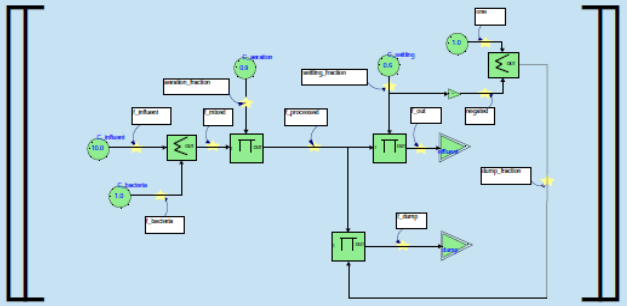
What does this WWTP model mean?



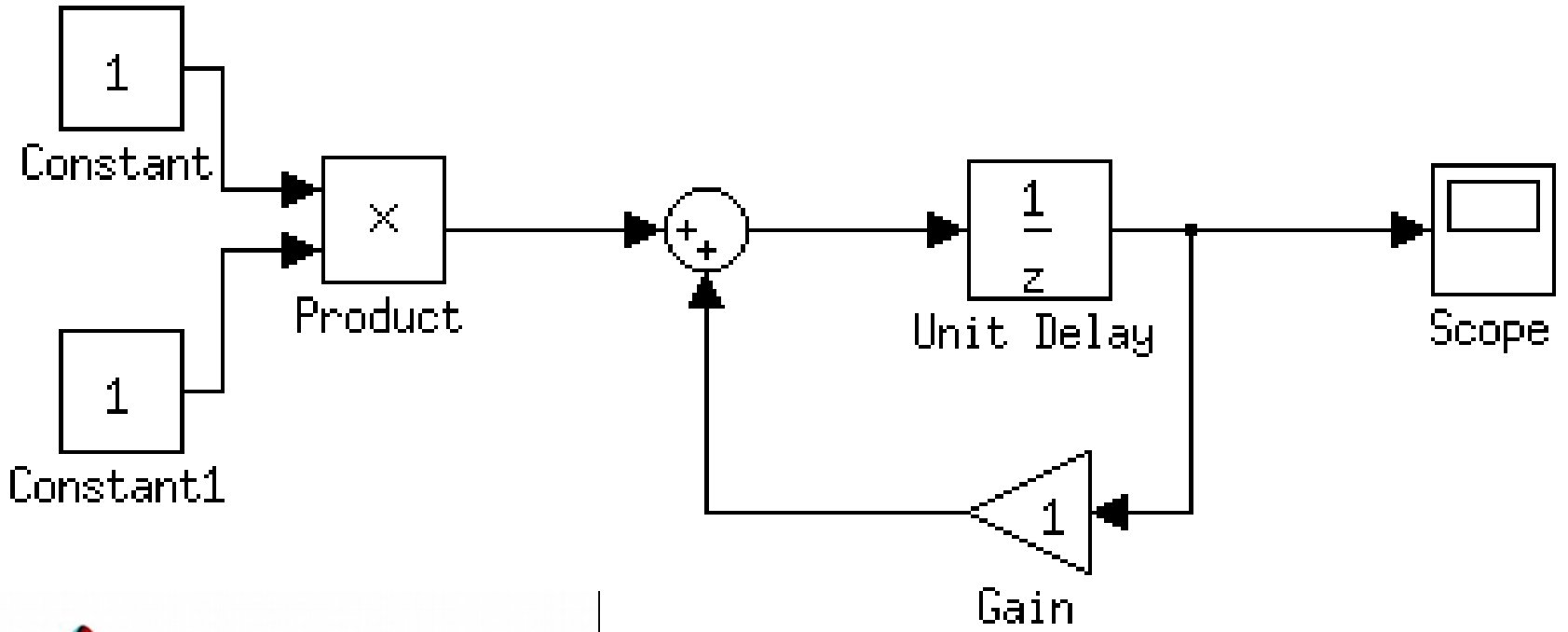
. . . its meaning (steady-state abstraction): Causal Block Diagram (CBD)



Meaning of the CBD . . . semantic mapping onto algEqns



$$
\left\{
\begin{array}{rcl}
f\_influent & = & C\_influent \\
f\_bacteria & = & C\_bacteria \\
f\_mixed & = & f\_influent + f\_bacteria \\
aeration\_fraction & = & C\_aeration \\
f\_processed & = & aeration\_fraction * f\_mixed \\
settling\_fraction & = & C\_settling \\
negated & = & -settling\_fraction \\
one & = & 1 \\
dump\_fraction & = & one + negated \\
f\_dump & = & f\_processed * dump\_fraction \\
f\_out & = & settling\_fraction * f\_processed
\end{array}
\right.
$$



MATLAB®
SIMULINK®

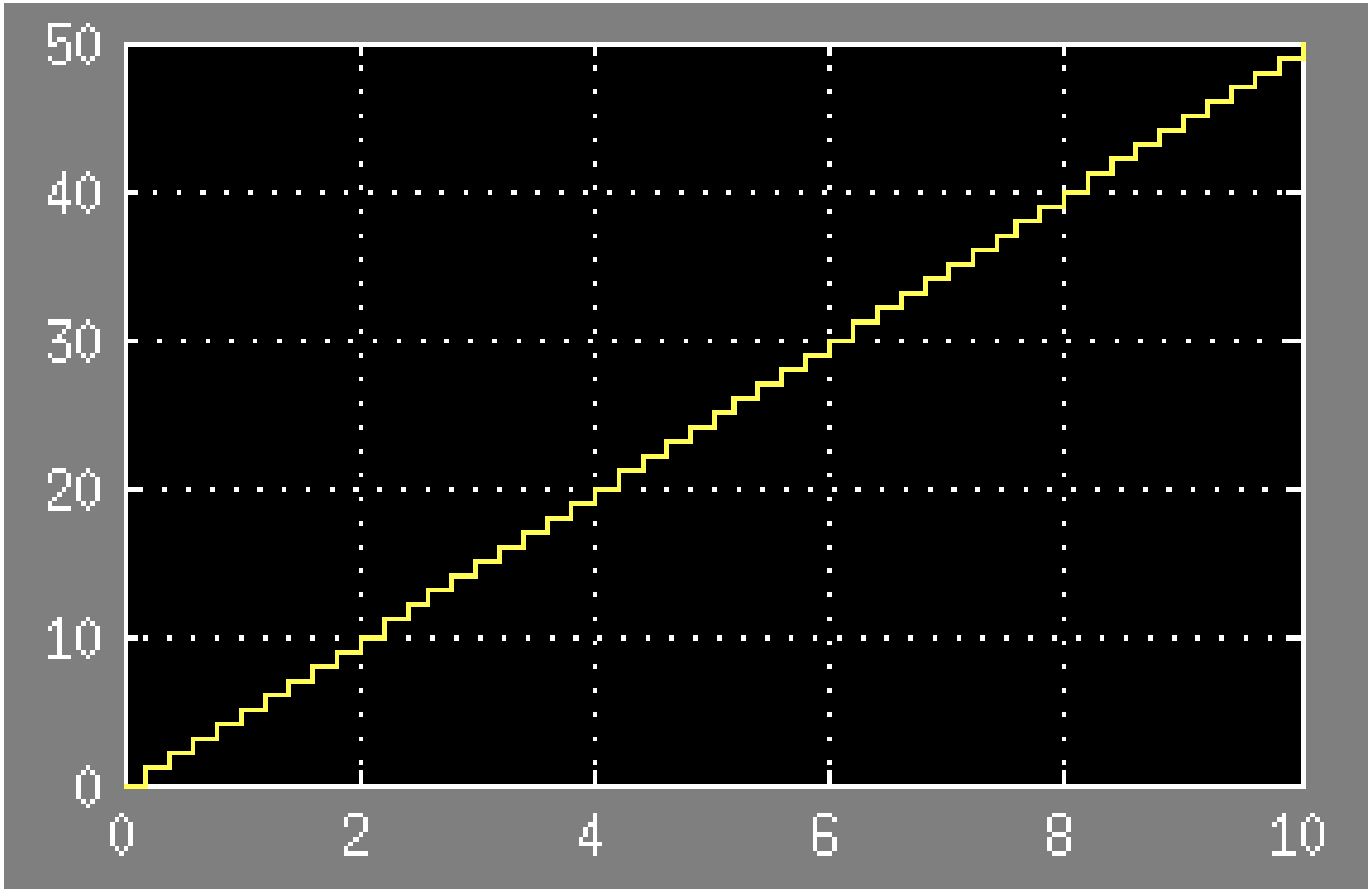
```
1: time_step ← 0
2: while not end_condition do
3:     schedule ← LOOPDETECT(DEPGRAPH(cbd))
4:     for gblock in schedule do
5:         COMPUTE(gblock)
6:     end for
7:     time_step ← time_step + 1
8: end while
```

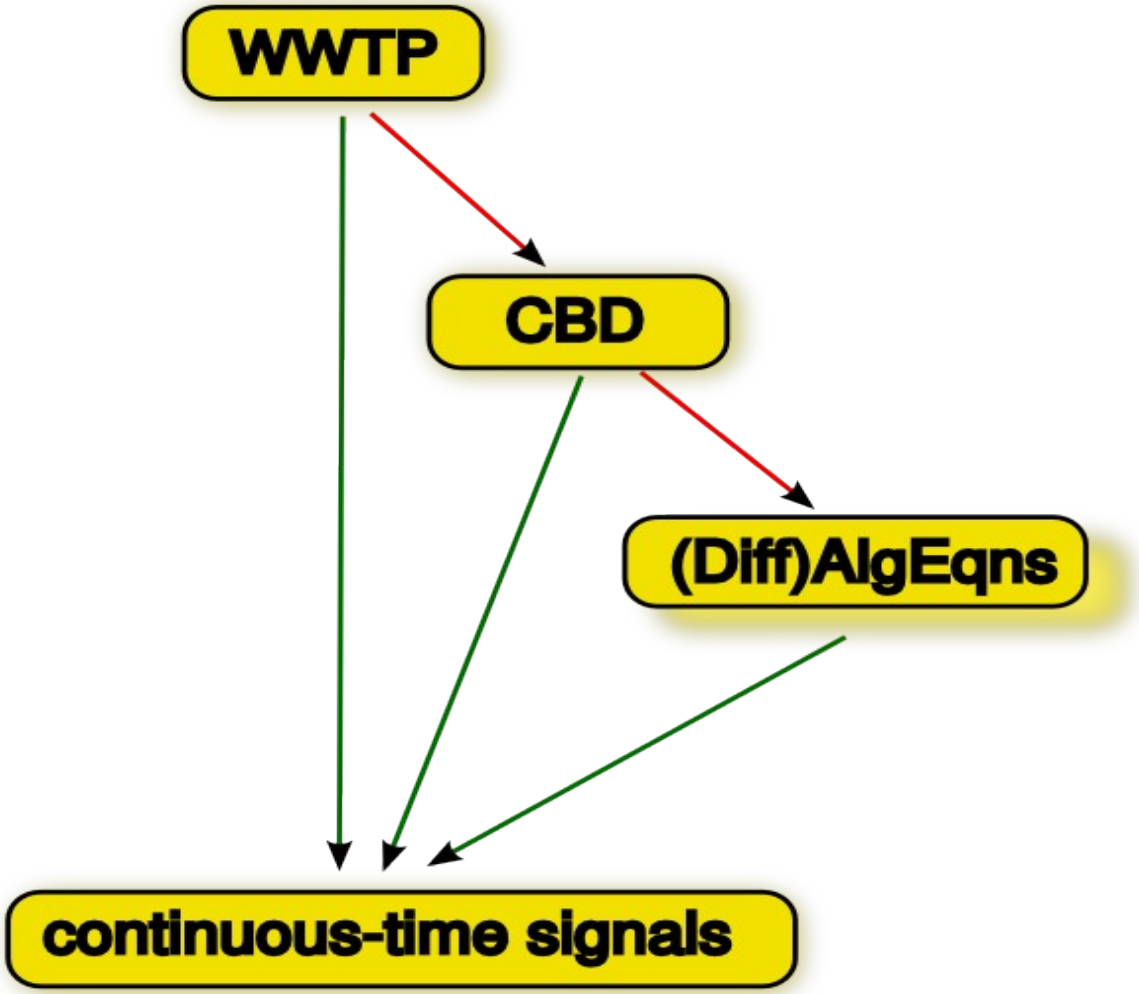
Simulation Algorithm

Small Steps

Algorithm 4 The CBD simulator's "main loop".

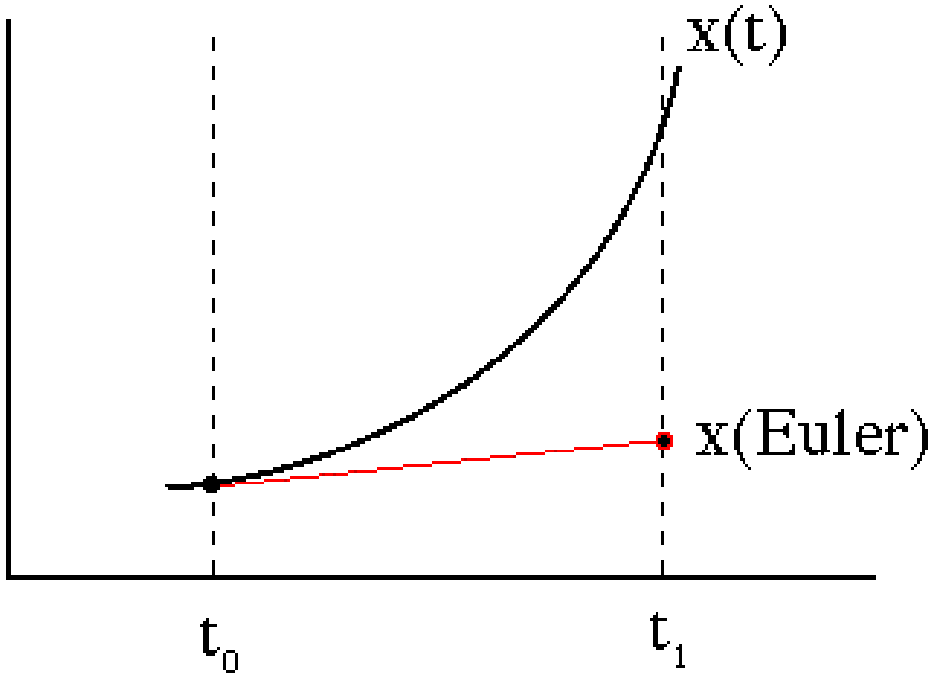
1: $time \leftarrow 0$
2: **while** not end_condition **do**
3: $schedule \leftarrow LOOPDETECT(DEPGRAPH(cbd))$
4: **for** gblock **in** schedule **do**
5: $COMPUTE(gblock)$
6: **end for**
7: $time \leftarrow time + \delta_t$
8: **end while**


50
40
30
20
10
0
0
2
4
6
8
10


WWTP
CBD
(Diff)AlgEqns
continuous-time signals

Modelica in Context

- Domain/Problem-Specific
- Laws of Physics
- Power Flow
- a-causal (Mathematical) ← **Modelica**
- Causal Block Diagrams
- Numerical (Discrete) Approximations
- Computer Numerical (Floating Point, Fixed Point)
- As-Fast-As-Possible vs. Real-time
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)

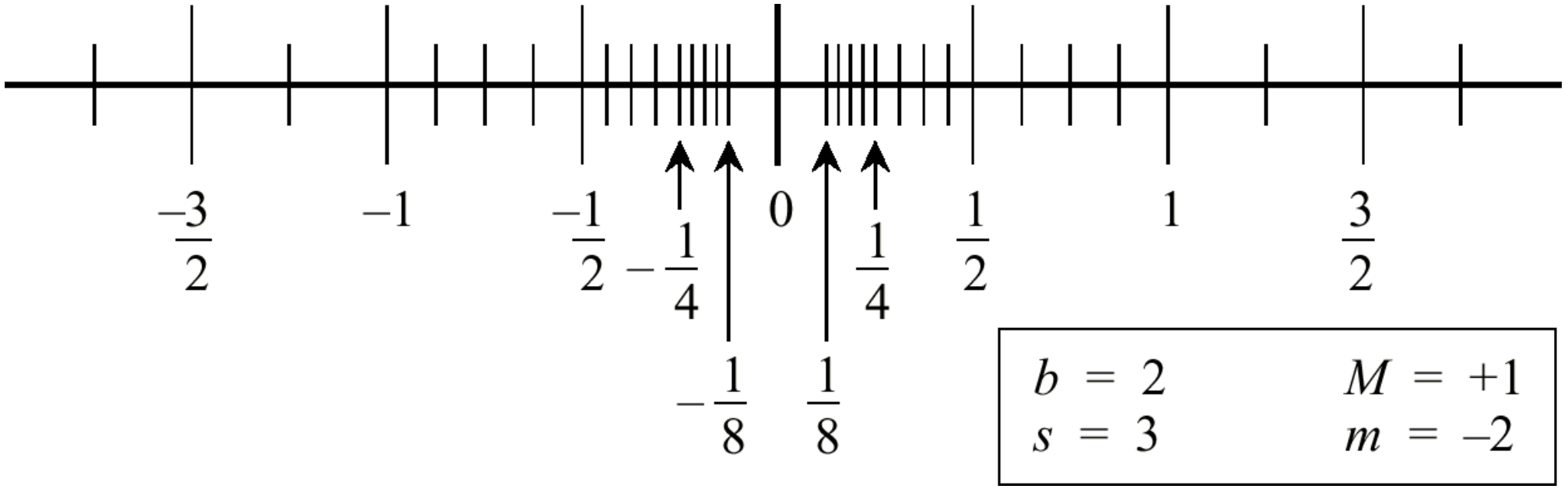


$$
\begin{aligned}
v_{x,n+1} &= v_{x,n} + \Delta t\, a_x(x_n, y_n, t) \\
x_{n+1} &= x_n + \Delta t\, v_{x,n}
\end{aligned}
$$

Modelica in Context

- Domain/Problem-Specific
- Laws of Physics
- Power Flow
- a-causal (Mathematical) ← **Modelica**
- Causal Block Diagrams
- Numerical (Discrete) Approximations
- Computer Numerical (Floating Point, Fixed Point)
- As-Fast-As-Possible vs. Real-time
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)

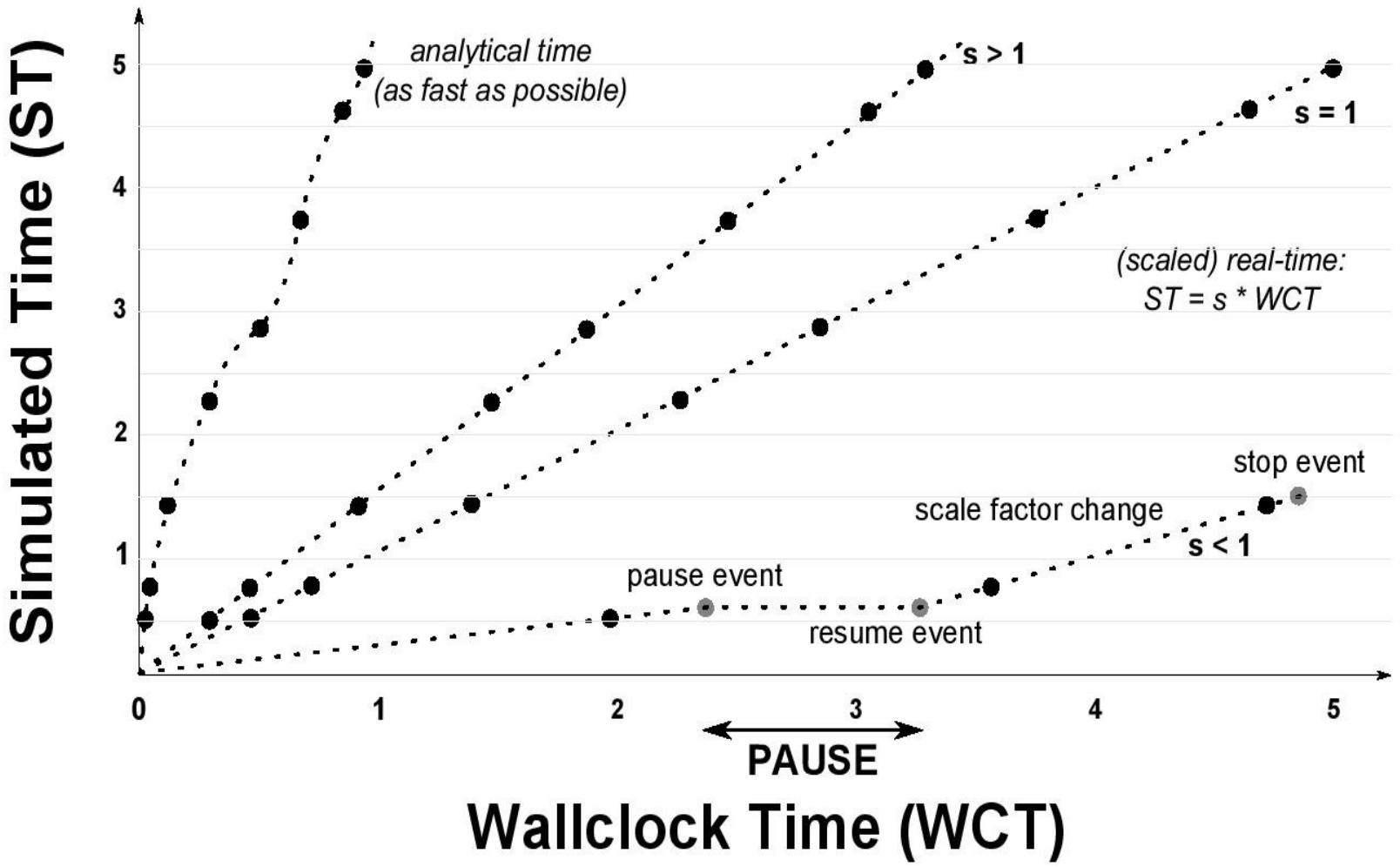
Example Floating Point Format



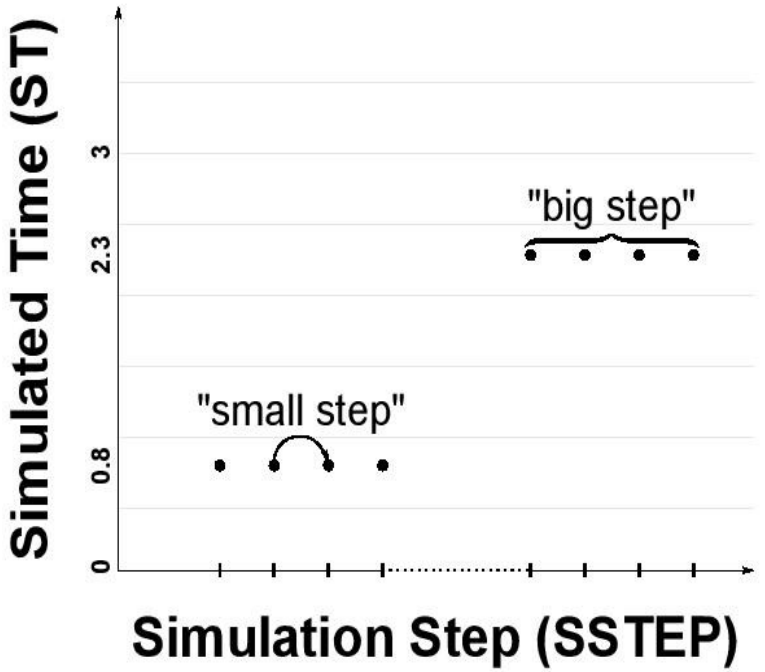
- Smallest non-zero positive number = $b^m \times b^{-1} = 1/8$
- Largest non-zero positive number = $b^M \times (1 - b^{-s}) = 7/4$
- Smallest gap = $b^m \times b^{-s} = 1/32$
- Largest gap = $b^M \times b^{-s} = 1/4$
- Number of representable numbers = $2\times((M-m)+1)\times(b-1)\times b^{s-1}+1 = 33$
  ... fits into available bits? Optimal number of bits?
- Note: fill the gap around 0: **de-normalized**

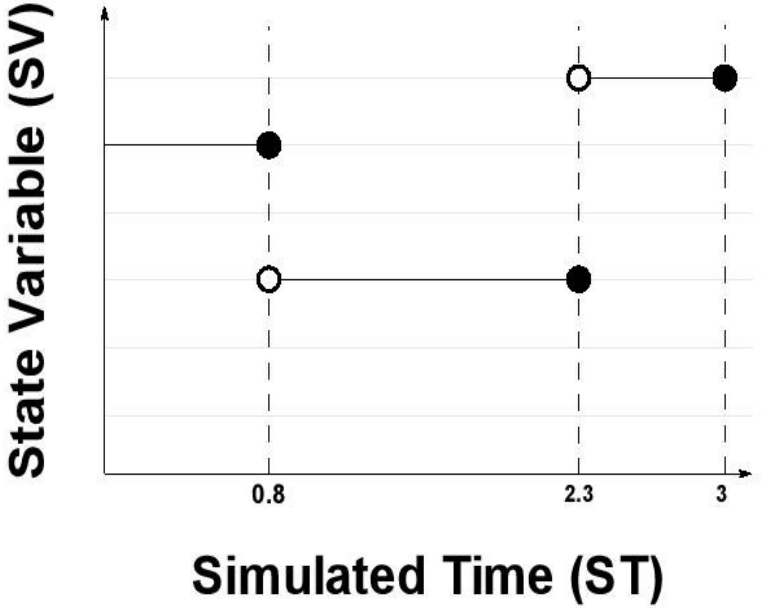
Modelica in Context

- Domain/Problem-Specific
- Laws of Physics
- Power Flow
- a-causal (Mathematical) ← **Modelica**
- Causal Block Diagrams
- Numerical (Discrete) Approximations
- Computer Numerical (Floating Point, Fixed Point)
- As-Fast-As-Possible vs. Real-time
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)

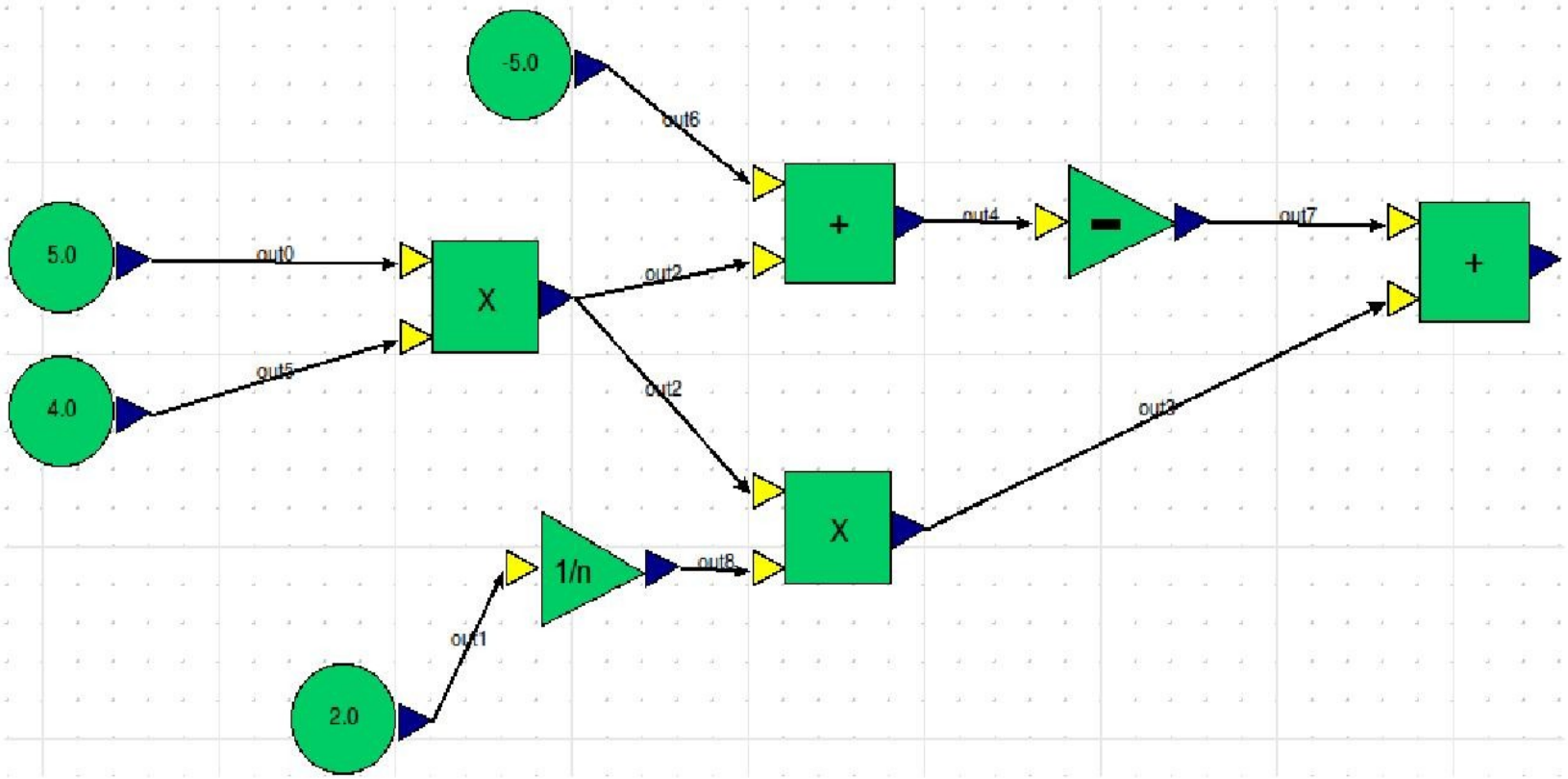

Simulated Time (ST)
Wallclock Time (WCT)
analytical time
(as fast as possible)
s > 1
s = 1
(scaled) real-time:
ST = s * WCT
stop event
scale factor change
s < 1
pause event
resume event
PAUSE
0
1
2
3
4
5

A "Generic" Simulator





A Simple Causal Block Diagram (CBD/SDF)



Ernesto Posse, Juan de Lara, and Hans Vangheluwe.
Processing causal block diagrams with graph-grammars in AToM^3.
Applied Graph Transformation (AGT) at ETAPS, pp. 23 – 34, April 2002.

Simulation Algorithm

Top-level time-stepping

Algorithm 2 The CBD simulator's "main loop".

```
1: time ← 0
2: while not end_condition do
3:     schedule ← LOOPDETECT(DEPGRAPH(cbd))
4:     for gblock in schedule do
5:         COMPUTE(gblock)
6:     end for
7:     time ← time + δ_t
8: end while
```

Simulation Algorithm

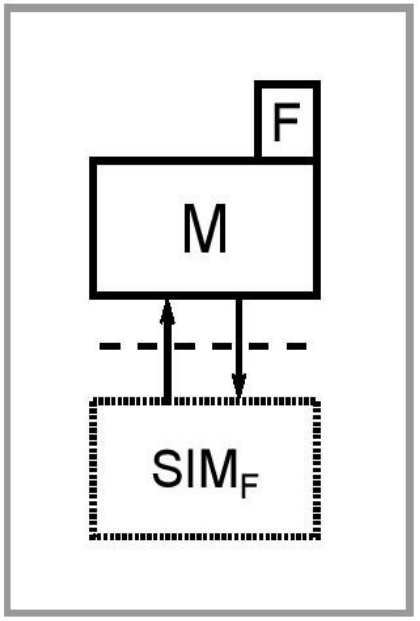
Big Steps

Algorithm 3 The CBD simulator's "main loop".

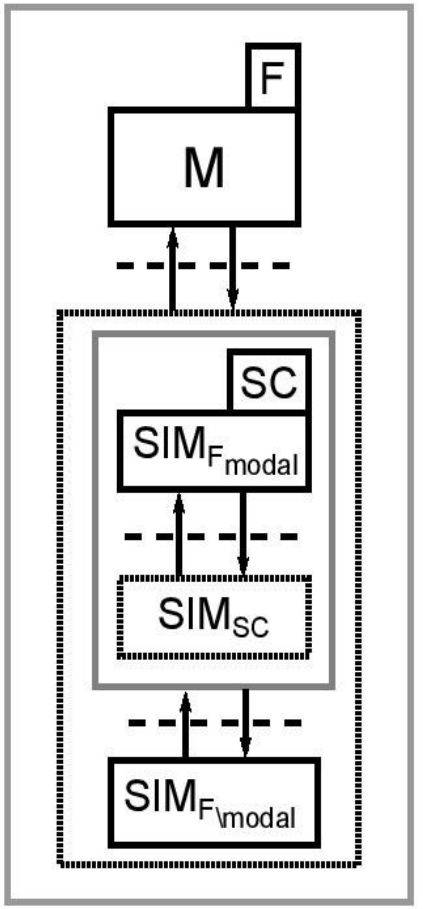
```
1: time ← 0
2: while not end_condition do
3:     schedule ← LOOPDETECT(DEPGRAPH(cbd))
4:     for gblock in schedule do
5:         COMPUTE(gblock)
6:     end for
7:     time ← time + δ_t
8: end while
```

De/Reconstructing the Simulator

Extracting the Modal Part



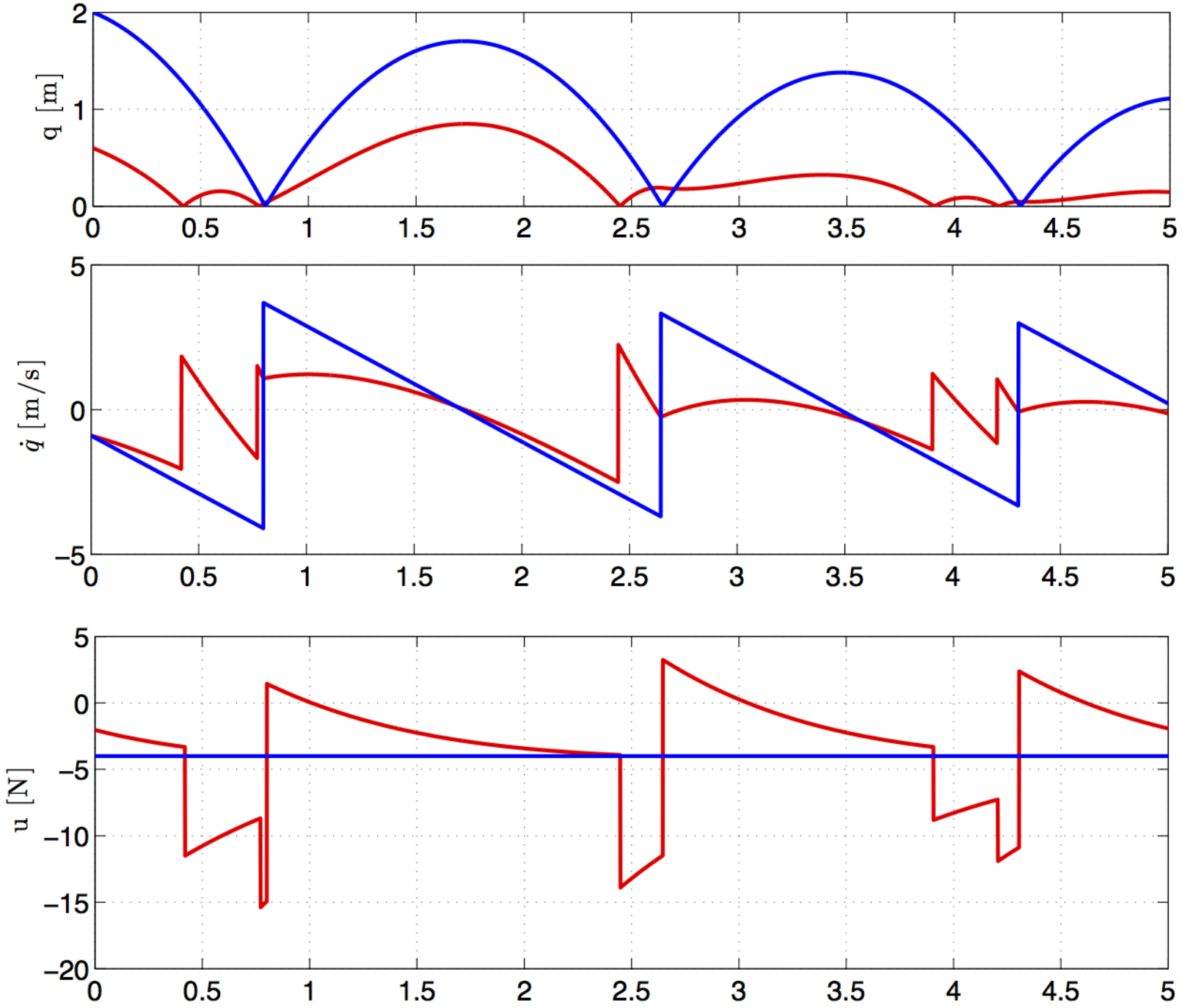
(a) A model M in formalism F and a simulation kernel SIM_F for F.



(b) De-/Re-constructing the simulator.

Modelica in Context

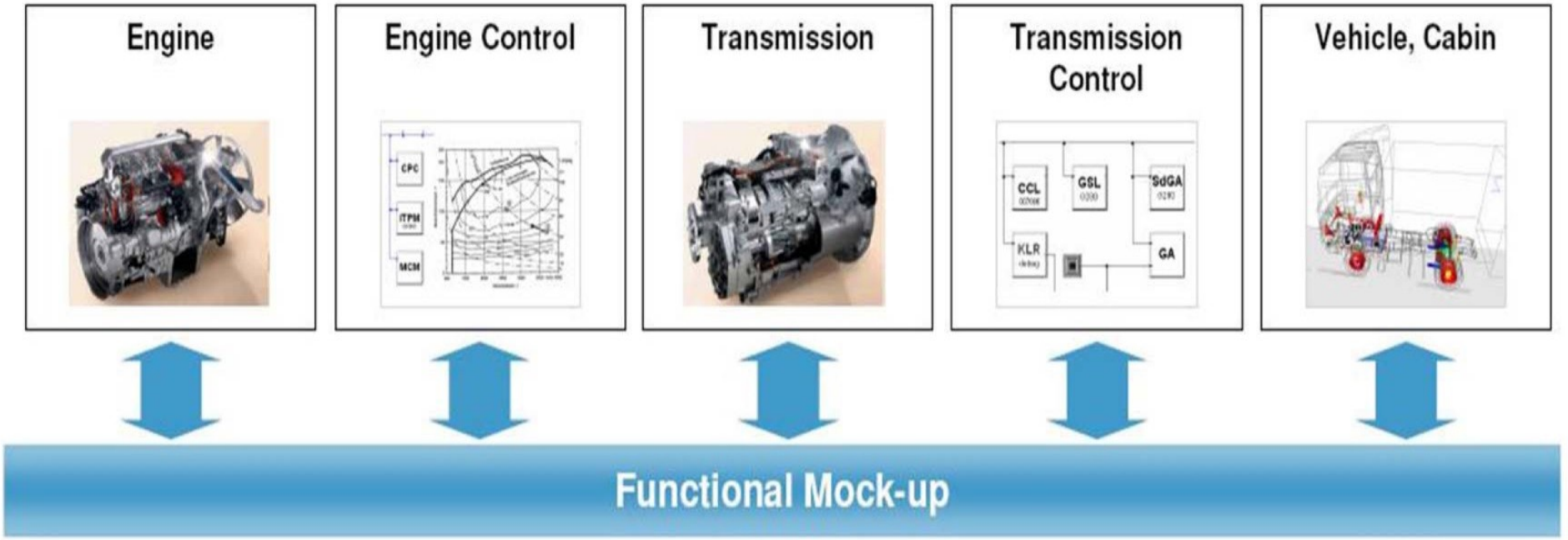
- Domain/Problem-Specific
- Laws of Physics
- Power Flow
- a-causal (Mathematical) ← **Modelica**
- Causal Block Diagrams
- Numerical (Discrete) Approximations
- Computer Numerical (Floating Point, Fixed Point)
- As-Fast-As-Possible vs. Real-time
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)


q [m]
$\dot{q}$ [m/s]
u [N]
2
1
0
5
0
−5
−10
−15
−20
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Modelica in Context

- Domain/Problem-Specific
- Laws of Physics
- Power Flow
- a-causal (Mathematical) ← **Modelica**
- Causal Block Diagrams
- Numerical (Discrete) Approximations
- Computer Numerical (Floating Point, Fixed Point)
- As-Fast-As-Possible vs. Real-time
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)

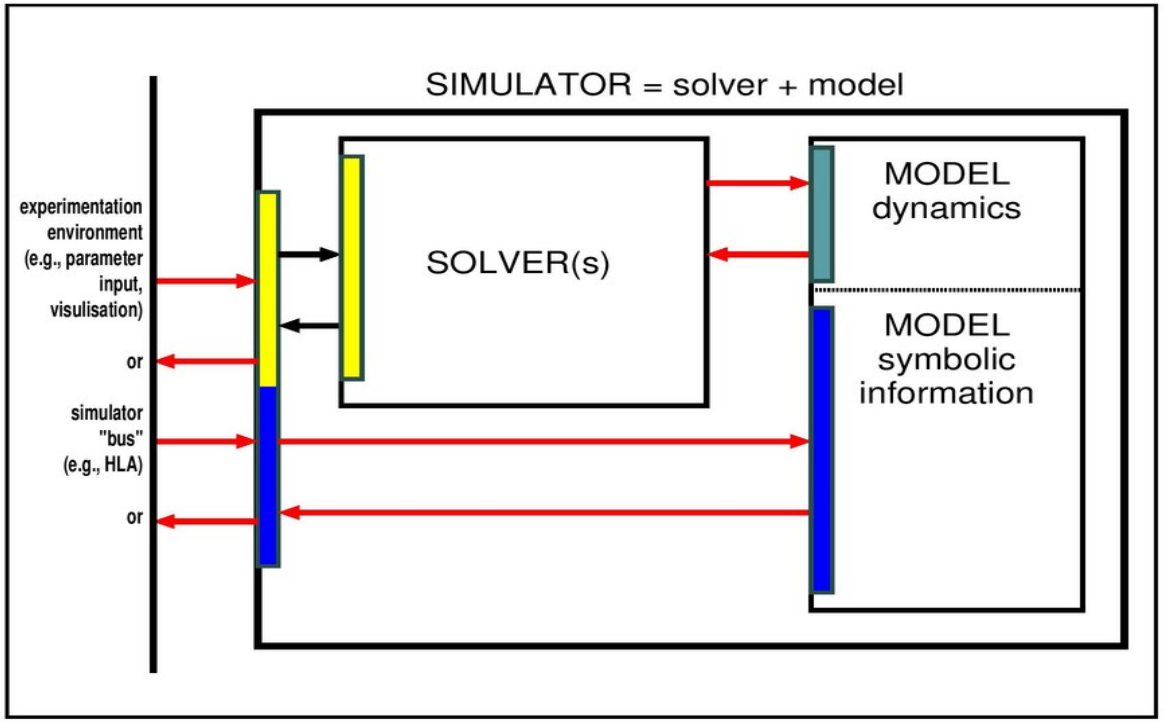
Functional Mockup Units (FMUs)

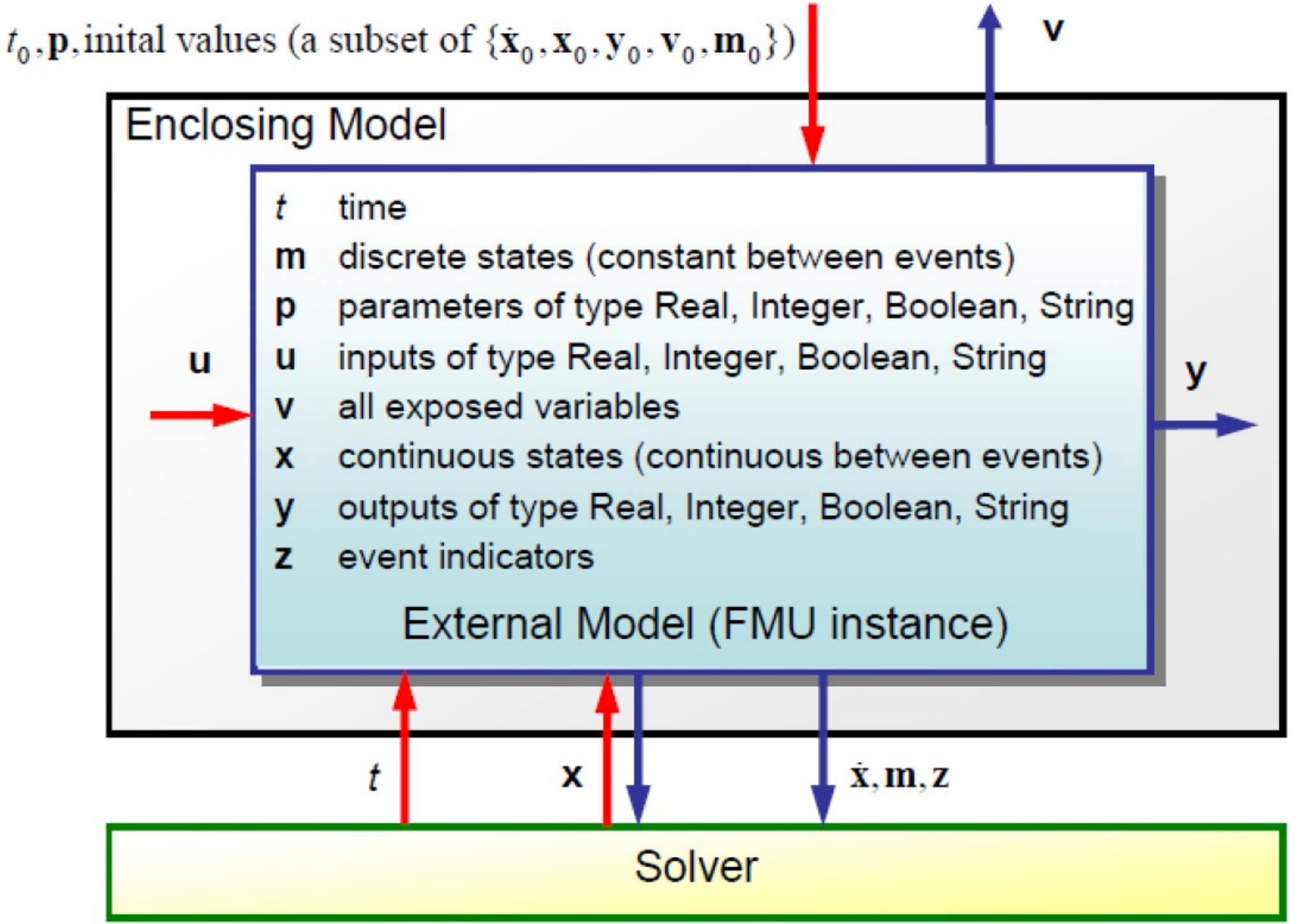


www.fmi-standard.org

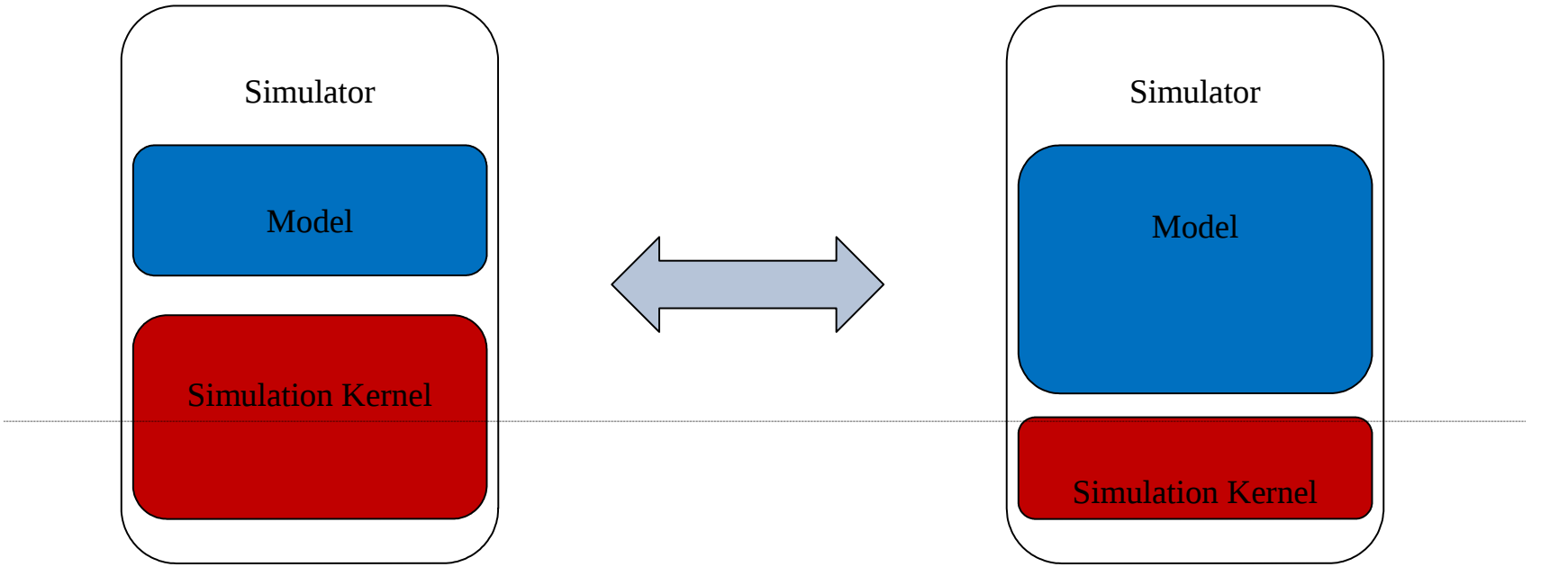
Bart Pussig, Bert Van Acker, Claudio Gomes

Model-Solver Interface
Simulator-Environment Interface




$t_0, \mathbf{p}$, inital values (a subset of $\{\dot{\mathbf{x}}_0, \mathbf{x}_0, \mathbf{y}_0, \mathbf{v}_0, \mathbf{m}_0\}$)
v
Enclosing Model
t time
m discrete states (constant between events)
p parameters of type Real, Integer, Boolean, String
u inputs of type Real, Integer, Boolean, String
v all exposed variables
x continuous states (continuous between events)
y outputs of type Real, Integer, Boolean, String
z event indicators
External Model (FMU instance)
u
y
t
x
$\dot{\mathbf{x}}, \mathbf{m}, \mathbf{z}$
Solver

meaningful operational semantics (Models of Computation)

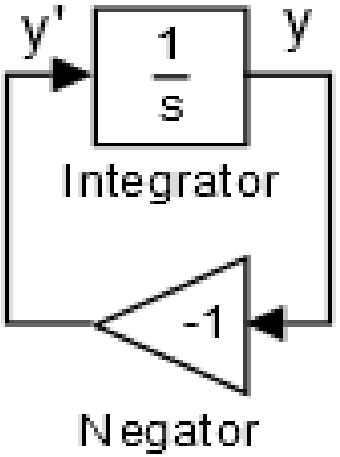


Traditional Simulator

Explicit Computational Semantics

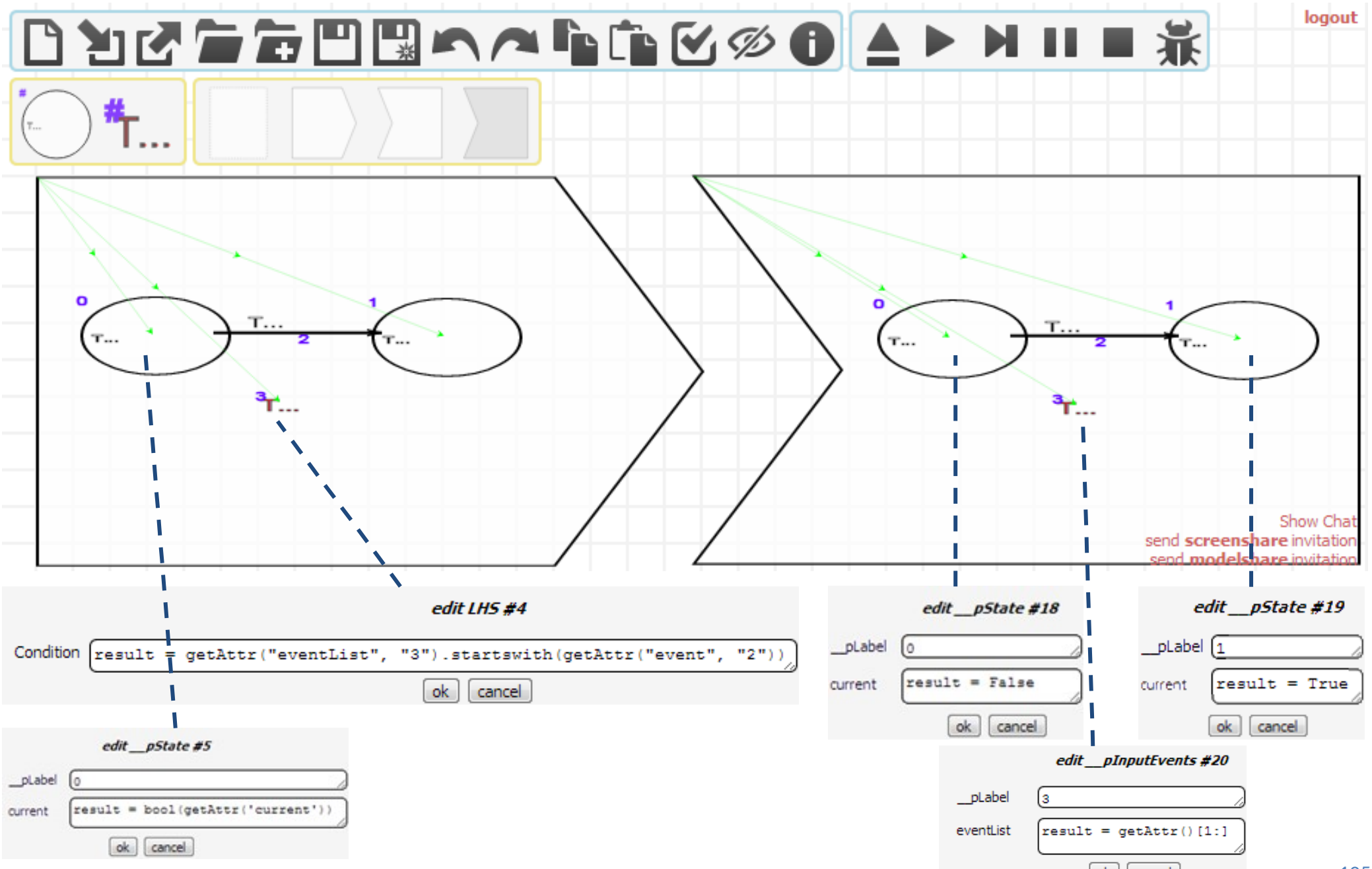
Models of Computation
Model Explicitly!

Euler discretization



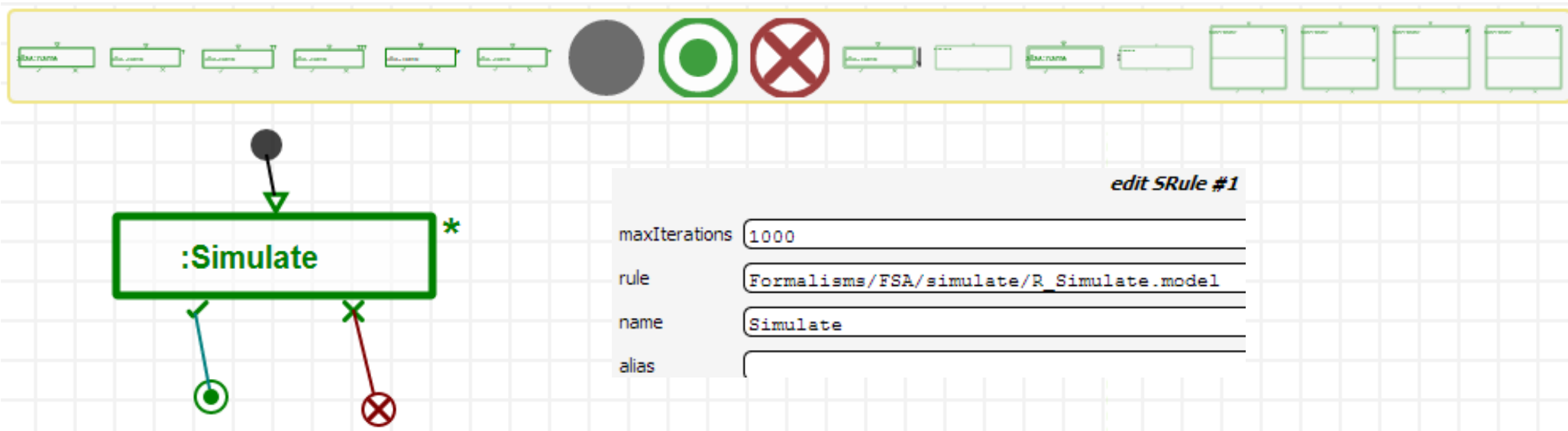
$$y[k] = y[k-1] + h * y'[k]$$

A GRAPH TRANSFORMATION RULE

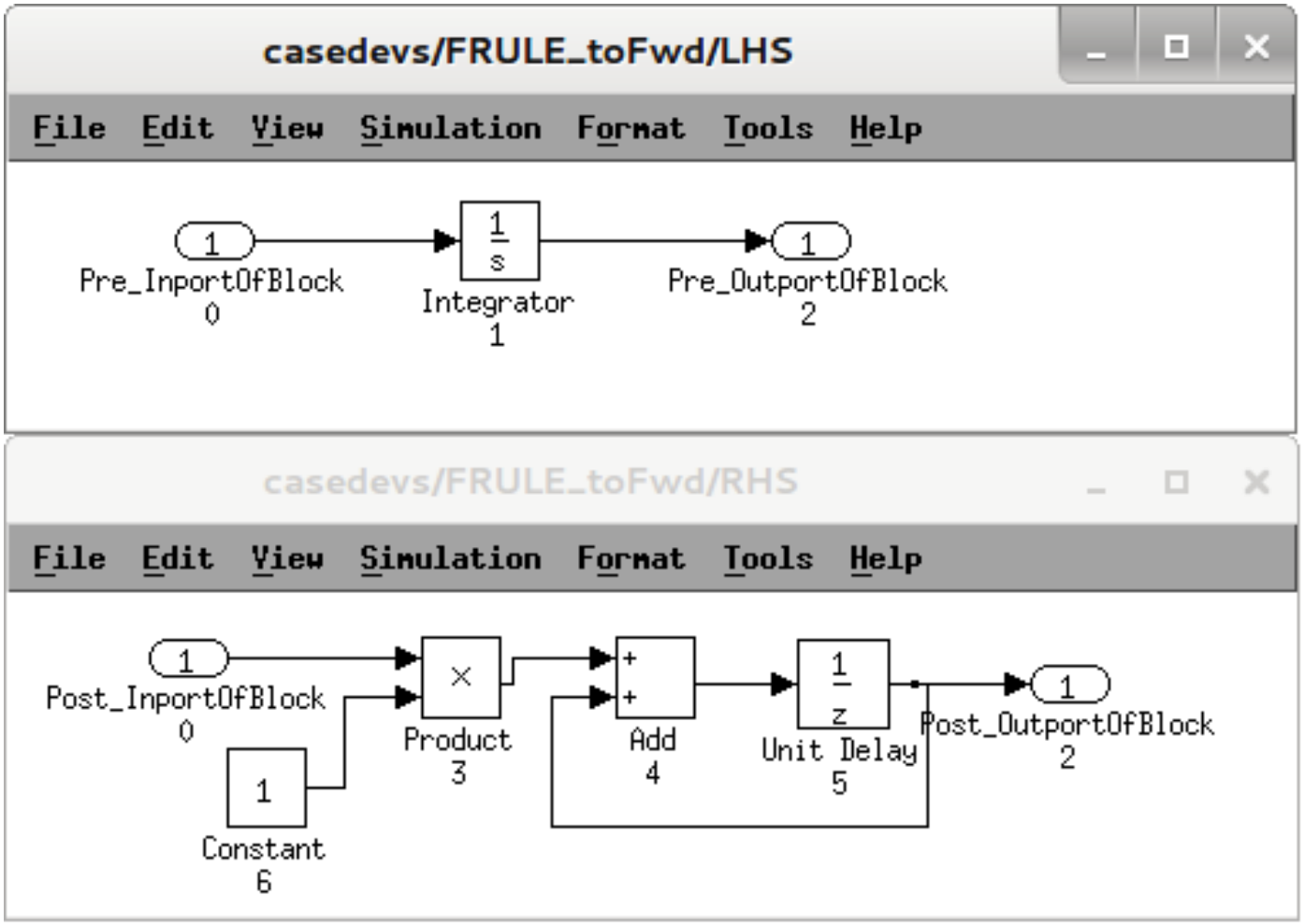


RULE SCHEDULING

- **Using the MoTif language, we can execute our “simulate” rule as an SRule**
- **SRule: Apply the rule recursively as long as possible**
  - **Find a match, rewrite the model, then re-match, rewrite the model, etc.**

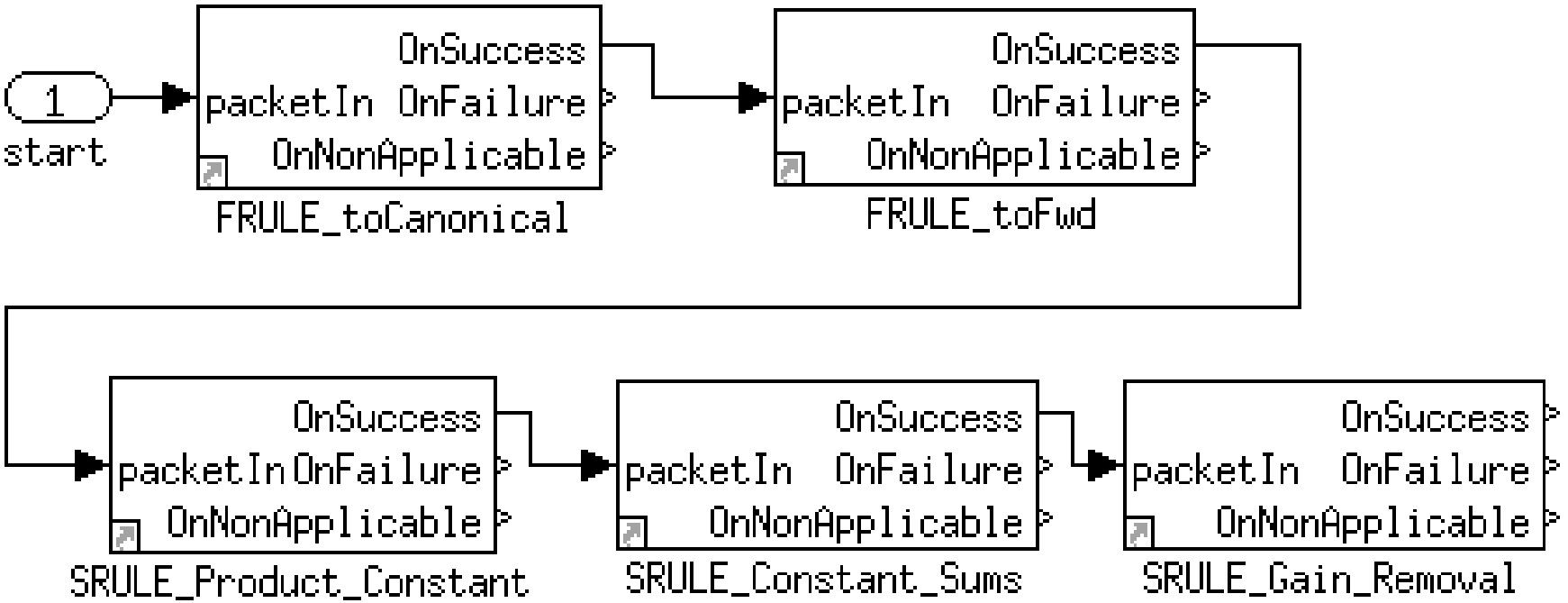


A Rule

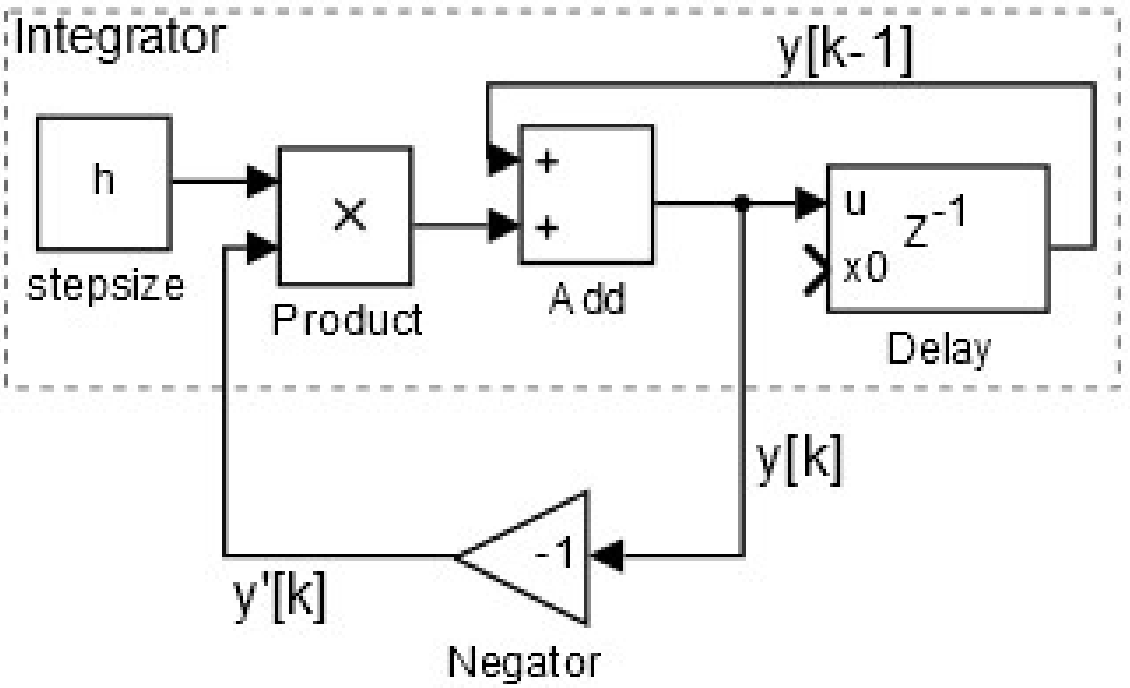


Joachim Denil's research

Transformation Schedule



Result of Transformation



- Explicit semantics
- Basis for analysis
- Give meaning to multi-formalism models
- Potential for global optimization
- Basis for complex Experimentation/Debugging environments