A Domain Specific Visual Language for Modelica

Daniel Riegelhaupt - 20071521

What is Modelica ?

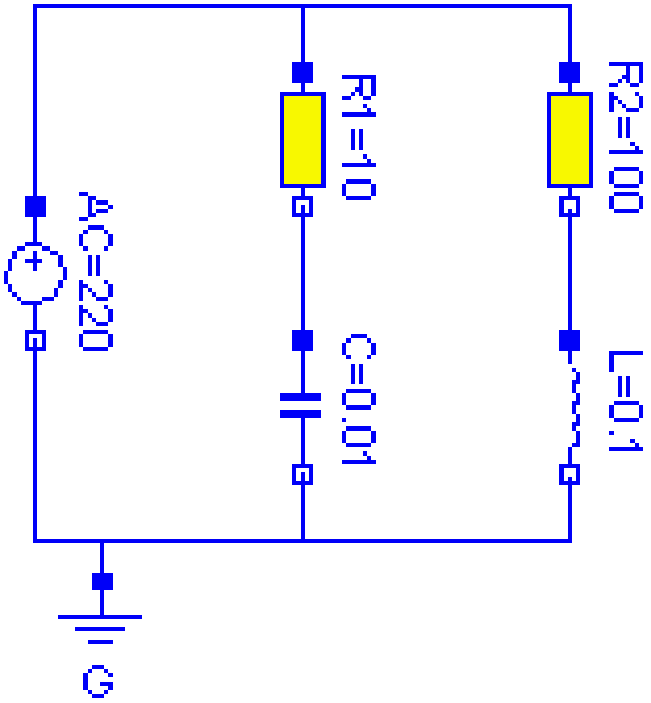
- Modelica is a freely available, dynamic (notion of time) declarative (mathematical equations) OO language for multi-domain modeling. [1]

  - OO for hierarchical purposes and inheritance purposes, not for message sending and such.

- Examples of domains are: mechatronic models in robotics, automotive and aerospace applications involving mechanical, electrical, hydraulic and control subsystems, distribution of electric power … [2]

[1] Fritzson, P., 2006. Introduction to object-oriented modeling and simulation with openmodelica.
[2] Modelica Association, December 2000. Modelica^{TM}- a unified objectoriented language for physical systems modeling tutorial. Version 1.4.

Small Introduction to Modelica [2]



That circuit becomes:

```
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); // Capacitor circuit
  connect (R1.n, C.p);
  connect (C.n, AC.n);
  connect (R1.p, R2.p); // Inductor circuit
  connect (R2.n, L.p);
  connect (L.n, C.n);
  connect (AC.n, G.p); // Ground
end circuit;
```

[2] Modelica Association, December 2000. Modelica^{TM}- a unified objectoriented language for physical systems modeling tutorial. Version 1.4.

Small Introduction to Modelica ^{2}

- Model is a class
- Resistor R1(R=10); etc … are declarations
- Equation is a keyword
- Connect is NOT a function it is an operator !

[2] Modelica Association, December 2000. Modelica^{TM}- a unified objectoriented language for physical systems modeling tutorial. Version 1.4.

Small Introduction to Modelica [2]

Simple example of connect:

```
connector Pin //connector is a class
        Voltage v; //type Voltage = Real(unit="V");
        flow Current i; //type Current = Real(unit="A");
end Pin;
```

Connect(pin1, pin2) will result in 2 equations:

1) pin1.v = pin2.v
2) pin1.i + pin2.i = 0 //generated by prefix flow

Notice pin1.i + pin2.i = 0 instead of pin1.i = -pin2.i !

[2] Modelica Association, December 2000. Modelica^{TM}- a unified objectoriented language for physical systems modeling tutorial. Version 1.4.

Small Introduction to Modelica [2]

We can use the pin to create more complex elements:

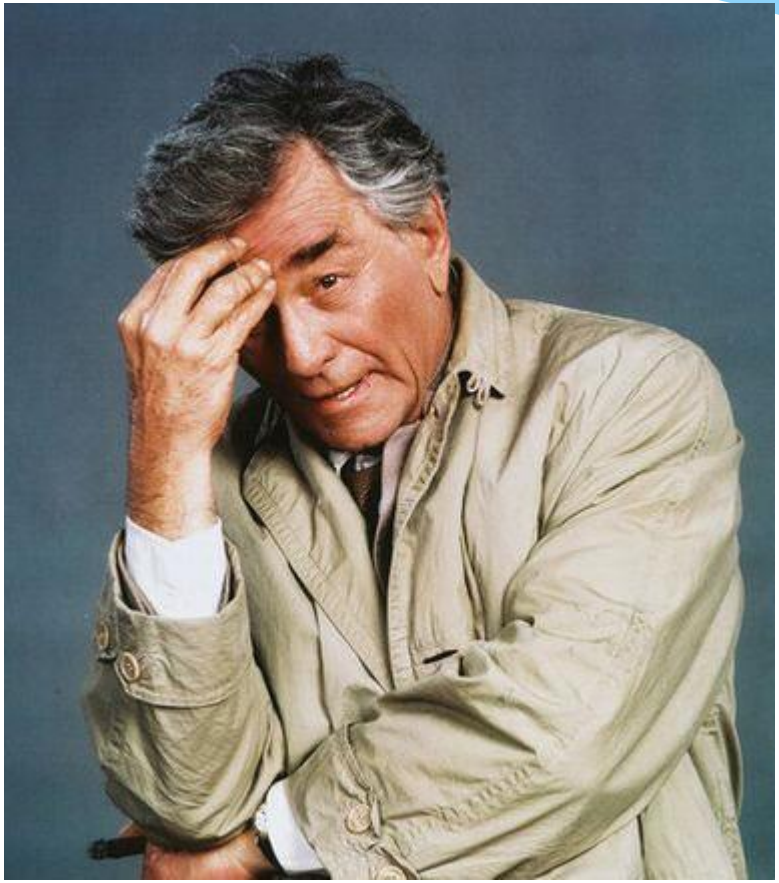
```
partial model OnePort // can’t use it by itself
"Superclass of elements with two electrical pins” //commentary
           Pin p, n;
           Voltage v;
           Current i;
equation
           v = p.v - n.v;
           0 = p.i + n.i;
           i = p.i;
end OnePort;

model Resistor "Ideal electrical resistor"
           extends OnePort;
           parameter Real R(unit="Ohm") "Resistance";
equation
           R*i = v; //law of Ohm
end Resistor;
```

Parameter indicates that it stay constants during simulation but can change inbetween runs

[2] Modelica Association, December 2000. Modelica^{TM}- a unified objectoriented language for physical systems modeling tutorial. Version 1.4.

Small Introduction to Modelica [2]



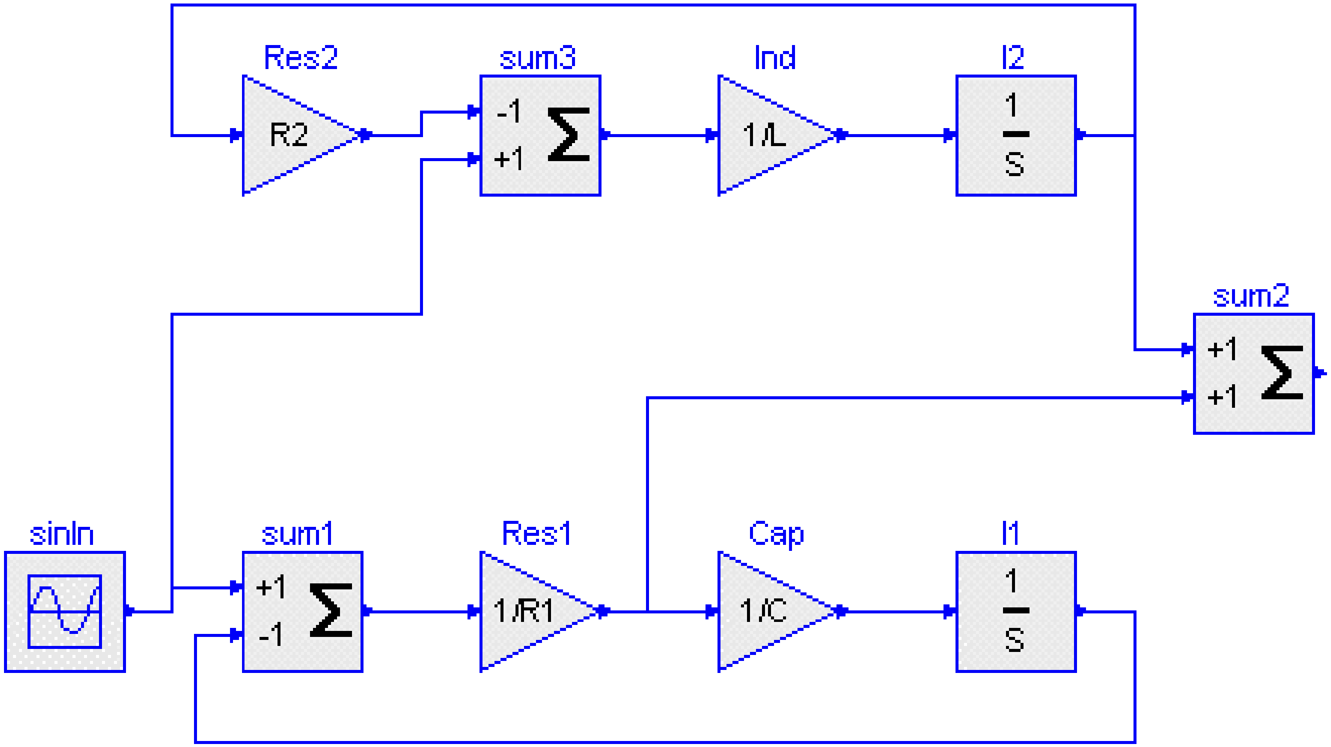
Question

Flow and equations …

What does that remind you of in the context of modeling ?

[2] Modelica Association, December 2000. Modelica^{TM}- a unified objectoriented language for physical systems modeling tutorial. Version 1.4.

Small Introduction to Modelica [2]



Modelica isn't a causal language but it can be transformed into causal block diagram

[2] Modelica Association, December 2000. Modelica^{TM}- a unified objectoriented language for physical systems modeling tutorial. Version 1.4.

Small Introduction to Modelica [2]

Multiple editors:

- Dymola (Commercial)
- OpenModelica (Free)
- Others: https://www.modelica.org/tools

[2] Modelica Association, December 2000. Modelica^{TM}- a unified objectoriented language for physical systems modeling tutorial. Version 1.4.

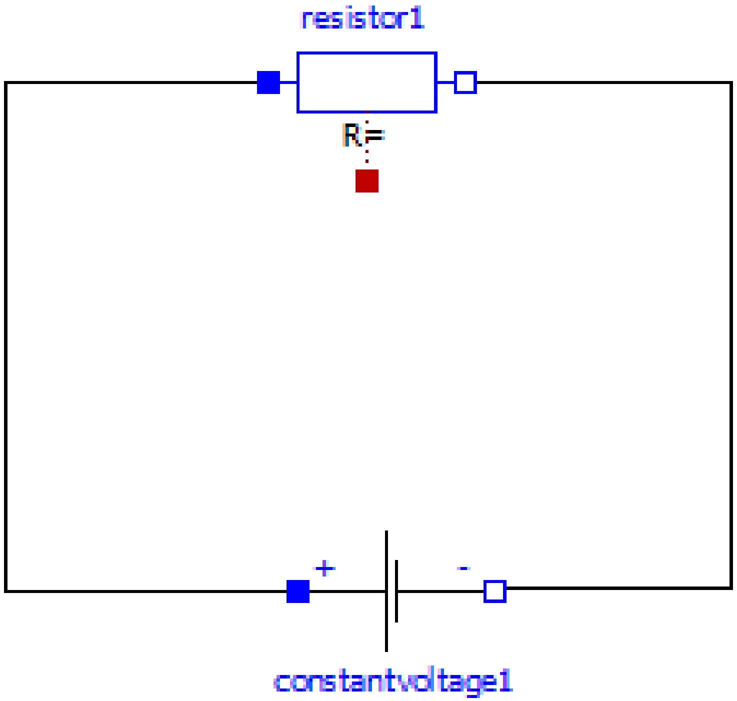
Want to Know More ?

This was just the tip of the iceberg …

Want to know more ?

- https://www.modelica.org/
- https://www.modelica.org/documents/ModelicaSpec32.pdf
- https://www.modelica.org/documents/ModelicaTutorial14.pdf

So… Why a DSVL ?



Question

What is wrong with this ?

So… Why a DSVL ?

Answer …

```
Translation of Unnamed:
DAE having 12 scalar unknowns and 12 scalar equations.
Error: The equations
         equation
         constantVoltage.p.i+resistor.p.i = 0;
         which was derived from
         constantVoltage.p.i+resistor.p.i = 0;
         0 = constantVoltage.p.i+constantVoltage.n.i;
         constantVoltage.i = constantVoltage.p.i;
         0 = resistor.p.i+resistor.n.i;
         resistor.i = resistor.p.i;
         constantVoltage.n.i+resistor.n.i = 0;

mean circular equalities for
constantVoltage.p.i, constantVoltage.n.i,
constantVoltage.i, resistor.p.i,
resistor.n.i, resistor.i
Translation aborted.
Translation aborted.
Translation aborted.
ERROR: 1 error was found
```

Clear as day no ?

So… Why a DSVL ?

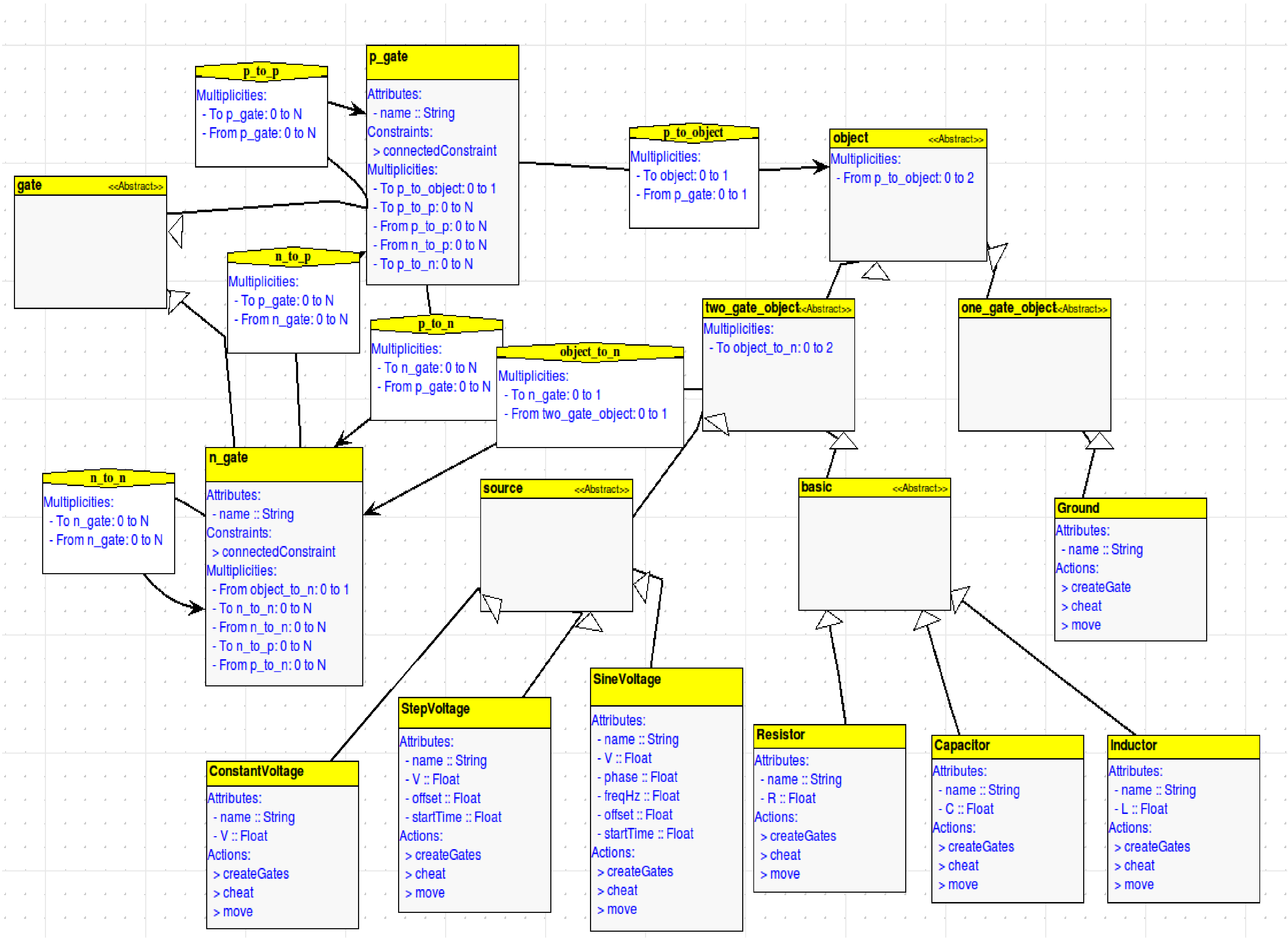
What it actually means:

there is no ground

A case study a small DSVL for Modelica

Make a DSVL for the following 7 pieces of the electrical circuit in ATOM^3:

- Constant voltage source
- Sine voltage source
- Step voltage source
- Ground
- Resistor
- Capacitor
- Inductor


p_to_p
Multiplicities:
- To p_gate: 0 to N
- From p_gate: 0 to N
p_gate
Attributes:
- name :: String
Constraints:
> connectedConstraint
Multiplicities:
- To p_to_object: 0 to 1
- To p_to_p: 0 to N
- From p_to_p: 0 to N
- From n_to_p: 0 to N
- To p_to_n: 0 to N
p_to_object
Multiplicities:
- To object: 0 to 1
- From p_gate: 0 to 1
object
<<Abstract>>
Multiplicities:
- From p_to_object: 0 to 2
gate
<<Abstract>>
n_to_p
Multiplicities:
- To p_gate: 0 to N
- From n_gate: 0 to N
p_to_n
Multiplicities:
- To n_gate: 0 to N
- From p_gate: 0 to N
object_to_n
Multiplicities:
- To n_gate: 0 to 1
- From two_gate_object: 0 to 1
two_gate_object
<<Abstract>>
Multiplicities:
- To object_to_n: 0 to 2
one_gate_object
<<Abstract>>
n_to_n
Multiplicities:
- To n_gate: 0 to N
- From n_gate: 0 to N
n_gate
Attributes:
- name :: String
Constraints:
> connectedConstraint
Multiplicities:
- From object_to_n: 0 to 1
- To n_to_n: 0 to N
- From n_to_n: 0 to N
- To n_to_p: 0 to N
- From p_to_n: 0 to N
source
<<Abstract>>
basic
<<Abstract>>
Ground
Attributes:
- name :: String
Actions:
> createGate
> cheat
> move
SineVoltage
Attributes:
- name :: String
- V :: Float
- phase :: Float
- freqHz :: Float
- offset :: Float
- startTime :: Float
Actions:
> createGates
> cheat
> move
StepVoltage
Attributes:
- name :: String
- V :: Float
- offset :: Float
- startTime :: Float
Actions:
> createGates
> cheat
> move
ConstantVoltage
Attributes:
- name :: String
- V :: Float
Actions:
> createGates
> cheat
> move
Resistor
Attributes:
- name :: String
- R :: Float
Actions:
> createGates
> cheat
> move
Capacitor
Attributes:
- name :: String
- C :: Float
Actions:
> createGates
> cheat
> move
Inductor
Attributes:
- name :: String
- L :: Float
Actions:
> createGates
> cheat
> move

A case study a small DSVL for Modelica

Basic idea:

- name the 7 elemtents exactly as they are called in Modelica
- give them the same attributes (including the inherited ones)
- add cardinalities
- Gates are only needed in the editor. in Modelica they are inherited and don’t need to be added separately
- Connection between gates are allowed in all directions (p -> p , n-> n , n<->p)

A case study a small DSVL for Modelica

Basic idea continued:

- Gate connection constraint: you can only connect g1 to g2 if:
  - g1 != g2
  - They aren’t already connected in any direction

A case study a small DSVL for Modelica

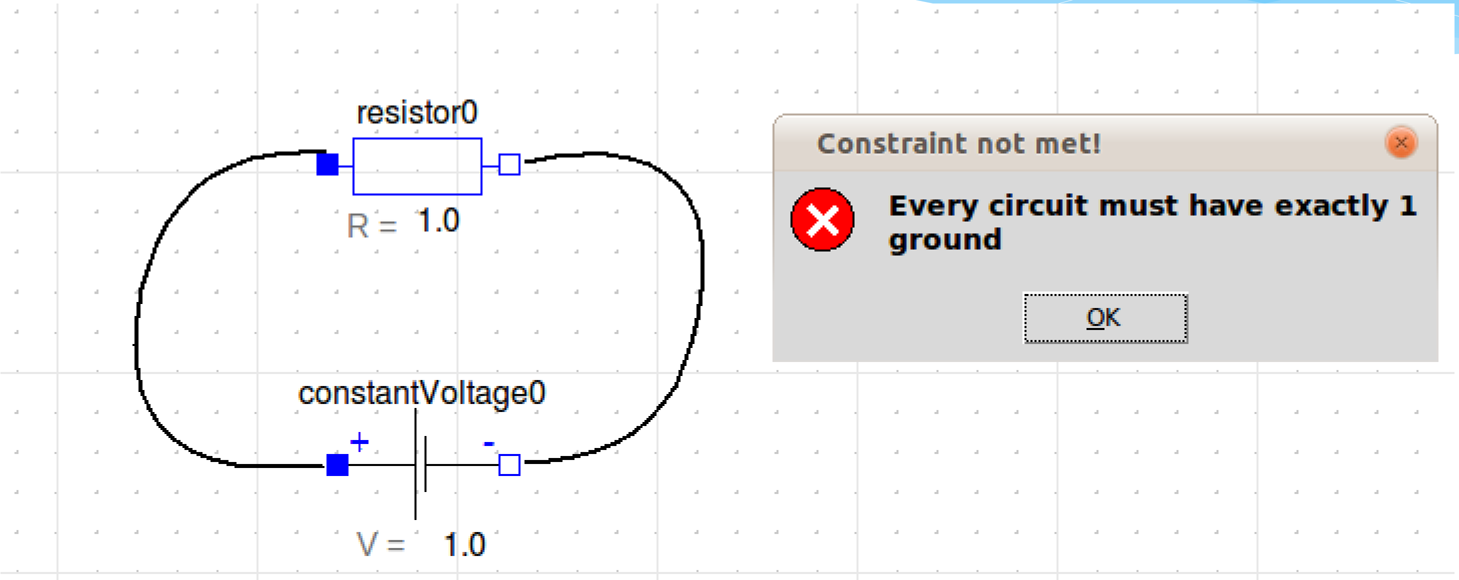
The domain constraints (checked in python):

- Each circuit must be closed
- Each circuit must have at least one source
- Each circuit must have exactly one ground

Generation (also in python) pretty straight forward:

- Iterate once to declare
- Iterate a second time to connect

A case study a small DSVL for Modelica

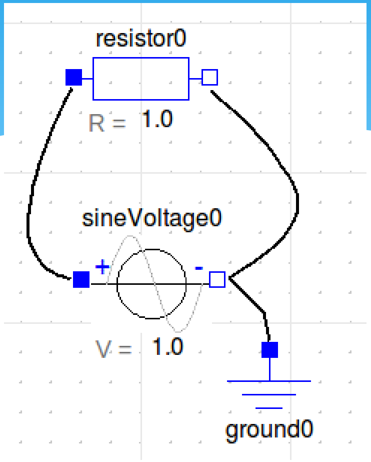


Constraint not met!

Every circuit must have exactly 1 ground

Better No ?

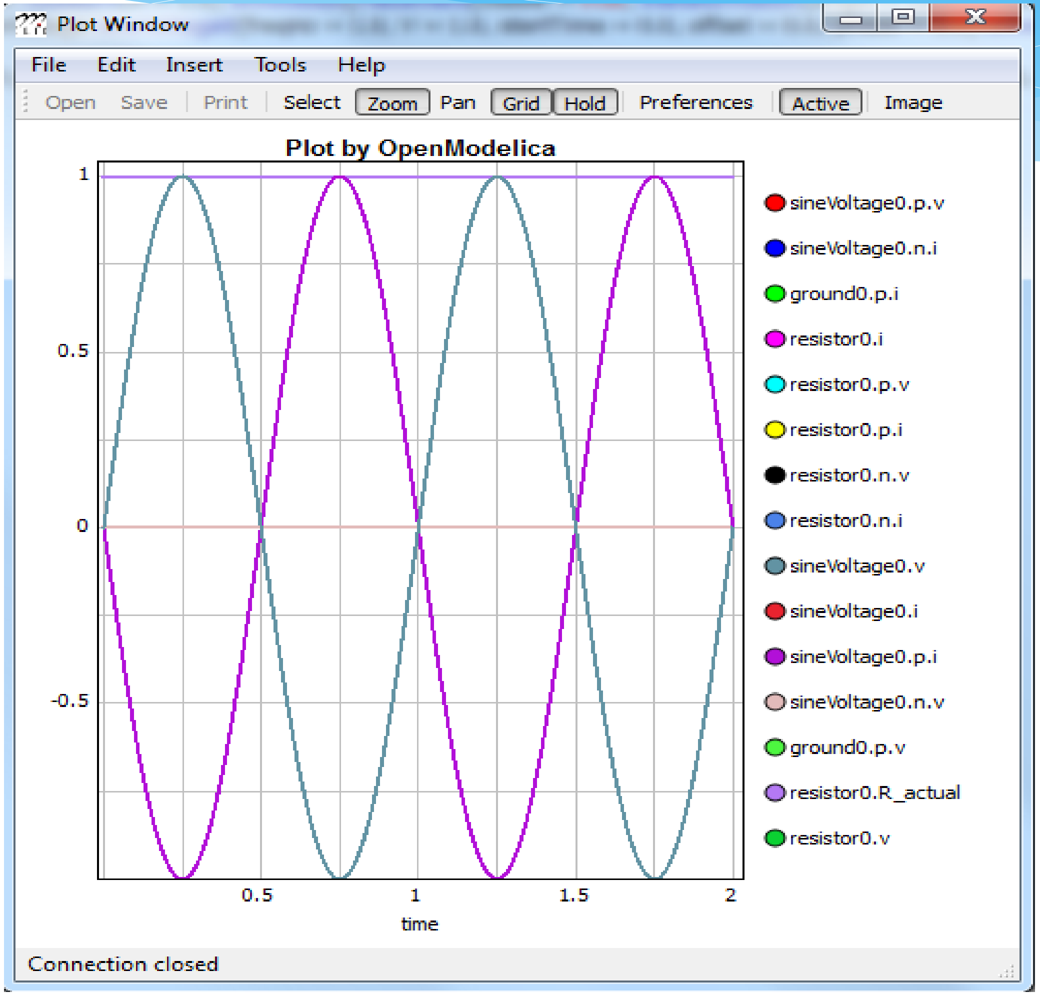
A case study a small DSVL for Modelica



Generates to:

```
model SimpleExample
          Modelica.Electrical.Analog.Basic.Resistor resistor0(R = 1.0);
          Modelica.Electrical.Analog.Sources.SineVoltage sineVoltage0(freqHz =
1.0, V = 1.0, startTime = 0.0, offset = 0.0, phase = 0.0);
          Modelica.Electrical.Analog.Basic.Ground ground0;
equation
          connect(resistor0.n, sineVoltage0.n);
          connect(sineVoltage0.n, ground0.p);
          connect(sineVoltage0.p, resistor0.p);
end SimpleExample;
```

A case study a small DSVL for Modelica



A case study a small DSVL for Modelica

Observations:

- Things that are possible in the physical world are not possible when simulating (but accepted by the DVSL) example: parallel sources
- Due to my lack of physics knowledge i tried some scenarios that aren't that good Like a conenction between a source capacitor and ground.
    - a domain expert is very handy in these cases
    - Not all simulators react the same

A case study a small DSVL for Modelica

Conclusions:

*the DSVL build is more general then Modelica (good thing can be reused)

*for it to be more Modelica specific the limitations of Modelica in that domain should be known (unfortunatly these are mathematical and part of the language not the domain)

*a domain expert is needed when doings something like this

Future Work

For big DSVL it could be useful to extract the annotated diagrams and convert them to for example ATOM^3 graph files.

- Basic idea: parse annotation in ANTLR and use string templates to output file.
- Problem: no annotations EBNF available in the specification , annotations can be vendor specific and hierachicle.

You can wake up now it is finished !

Any questions ?

Thank you for your time.