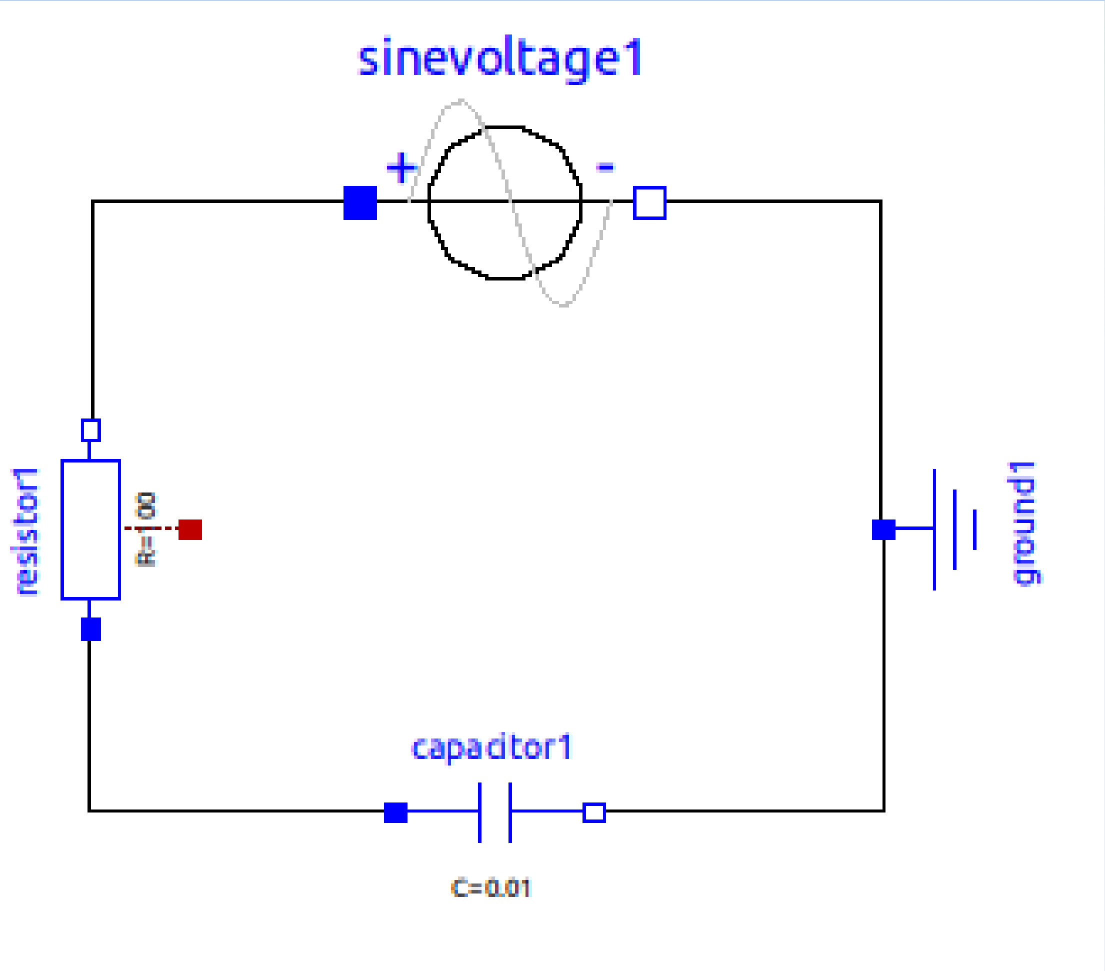
Modelica transformational Debugger and implementation in the OpenModelica Compiler

Martin Sjölund <martin.sjolund@liu.se>
Peter Fritzson <peter.fritzson@liu.se>
Linköping University, Sweden

OpenModelica Workshop
Feb 2012, Linköping University, Sweden

What Happens in a Modelica Compiler?

Example - RC Circuit (Diagram)



Example - RC Circuit (Code)

```
model RC
  Modelica.Electrical.Analog.Basic.Ground ground1;
  Modelica.Electrical.Analog.Basic.Resistor resistor1(R = 100);
  Modelica.Electrical.Analog.Basic.Capacitor capacitor1(C = 0.01);
  Modelica.Electrical.Analog.Sources.SineVoltage sinevoltage1(V = 240,
freqHz = 50);
equation
  connect(capacitor1.n,ground1.p);
  connect(sinevoltage1.n,ground1.p);
  connect(resistor1.n,sinevoltage1.p);
  connect(resistor1.p,capacitor1.p);
end RC;
```

Example - RC Circuit (Flat Code)

```
class RC // 24 equations and variables
 …
equation
  …
  ground1.p.v = 0.0;
  0.0 = resistor1.p.i + resistor1.n.i;
  resistor1.i = resistor1.p.i;
  resistor1.T_heatPort = resistor1.T;
  capacitor1.i = capacitor1.C * der(capacitor1.v);
  capacitor1.v = capacitor1.p.v - capacitor1.n.v;
  0.0 = capacitor1.p.i + capacitor1.n.i;
  capacitor1.i = capacitor1.p.i;
  …
end RC;
```

From Unsorted DAE to Sorted ODE

```
class RC // 24 equations and variables
 …
equation
  …
  ground1.p.v = 0.0;
  0.0 = resistor1.p.i + resistor1.n.i;
  resistor1.i = resistor1.p.i;
  resistor1.T_heatPort = resistor1.T;
  capacitor1.i = capacitor1.C * der(capacitor1.v);
  capacitor1.v = capacitor1.p.v - capacitor1.n.v;
  0.0 = capacitor1.p.i + capacitor1.n.i;
  capacitor1.i = capacitor1.p.i;
  ...
end RC;
```

```
class RC // 5 equations and variables
 …
 // 14 alias variables 5 constants
equation
  sinevoltage1.signalSource.y =
sinevoltage1.signalSource.offset + (if time <
sinevoltage1.signalSource.startTime then 0.0 else
sinevoltage1.signalSource.amplitude *
sin(6.28318530717959 *
(sinevoltage1.signalSource.freqHz * (time -
sinevoltage1.signalSource.startTime)) +
sinevoltage1.signalSource.phase));
  resistor1.v = capacitor1.v -
sinevoltage1.signalSource.y;
  capacitor1.i = -resistor1.v / resistor1.R_actual;
  resistor1.LossPower = -resistor1.v * capacitor1.i;
  der(capacitor1.v) = capacitor1.i / capacitor1.C;
end RC;
```

Debugging Equation Systems

- Modelica involves a lot of magic
  - Lots of math
  - Hidden to users
  - Users want to access this information
  - Some algorithms work better for certain input
  - Not intuitive
    - No explicit control flow
    - Numerical solvers
    - Linear/Non-linear blocks
    - Optimization
    - Events

Typical OMC Error Message

Error solving nonlinear system 132

time = 0.002

residual[0] = 0.288956

x[0] = 1.105149

residual[1] = 17.000400

x[1] = 1.248448

...

Better Message (Post-Mortem)

Error solving nonlinear system 132 <more info>

- time = 0.002
- residual[0] = 0.288956
- x[0] = 1.105149
- residual[1] = 17.000400
- x[1] = 1.248448
- ...

Origin

- Several Levels
  - (Graphical Representation)
  - Source Code
  - Flat Equation-System
  - Optimized Equation-System
  - Translated Code (typically C)
- It should always be possible to go backwards
  - Simple for flattened equation system to source
  - Harder for optimized code

Symbolic Transformations

- From source code to flat equations
  - Most of the structure remains
  - Few symbolic manipulations (mostly simplification/evaluation)
- Equation System Optimization
  - Changes structure
  - Strong connected components
  - Variable replacements
  - … and more

Tracing Transformations

- Simple Idea
  - Store transformations as equation metadata
  - Works best for operations on single equations
- Each kind of transformation is different
  - Alias Elimination (a = b)
  - Gaussian Elimination (linear systems, several equations)
  - Equation solving ($f_1(a,b) = f_2(a,b)$, solve for a)
  - ...

OpenModelica Implementation (1)

- Equation source has an extra field for transformations
- Optimization modules add information to this field
  - Some operations now need to keep track of any changes made
  - Expression simplification changed to fix-point algorithm

Before:

```
e2 = simplify(e1);
```

Now:

```
(e2,b) = simplify(e1);
source = addSymTSimplify
(b, source, e1, e2);
```

OpenModelica Implementation (2)

- Overhead?
  - It is so fast we enable tracing by default (1 extra comparison and/or cons operation per optimization)
  - No overhead unless you print the trace
    - +simCodeTarget=Dump

Alias Elimination

a = b

c = a + b

d = a - b

c = a + b (subst a=b) =>

c = b + b (simplify) =>

c = 2 * b

d = a - b (subst a=b) =>

d = b - b (simplify) =>

d = 0.0

- The alias relation a=b stored in variable a
- The equations are e.g. stored as (lhs,rhs,list<ops>)

Debugging Using the Trace

- Text-file
  - Initial implementation
  - Verify performance and correctness of the trace
- Database (SQL/XML queries)
  - Graphical debugging
  - Cross-referencing equations (dependents/parents)
  - Ability to see why a variable is solved in a particular way
  - Requires a schema

Trace Example

0 = y + der(x * time * z); z = 1.0;

(1) subst:

y + der(x * (time * z))

=>

y + der(x * (time * 1.0))

(2) simplify:

y + der(x * (time * 1.0))

=>

y + der(x * time)

(3) expand derivative (symbolic diff):

y + der(x * time)

=>

y + (x + der(x) * time)

(4) solve:

0.0 = y + (x + der(x) * time)

=>

der(x) = ((-y) - x) / time
time <> 0

Trace of Dummy Derivatives Alg.

differentiation:

d/dtime L ^ 2.0

=>

0.0

differentiation:

d/dtime x ^ 2.0 + y ^ 2.0

=>

2.0 * (der(x) * x + der(y) * y)

subst:

2.0 * (der(x) * x + der(y) * y)

=>

2.0 * ($DER.x * x + $DER.y * y)

=>

2.0 * (u * x + $DER.y * y)

=>

2.0 * (u * x + v * y)

=>

2.0 * (u * xloc[1] + v * xloc[0])

Readability of Trace

- Most equations have very few transformations on them
- Most of the interesting equations have a few
  - Still rather readable

MSL 3.1 MultiBody DoublePendulum

# Ops	Frequency	Comment
0	457	Parameters
1	89	Dummy eq & know var
2	720	Alias vars
3	479	Alias vars
4	124	Alias after simplify
5	25	Alias after simplify
6	99	Alias after simplify
7	55	Scalar eq
8	37	...
9	110	...
10	72	...
11	12	...
12	25	...
13	35	...
14	3	Known constant after many replacements
21	27	World object (3x3 matrix with many occurances of aliased vars)

Future Work

- Create database instead of text-file
- Graphical debugger
- Simulation runtime uses database
- Tracing in algorithmic code
- More operations recorded
  - Dead code elimination
  - Control flow and events
  - Forgotten optimization modules