Tool Demonstration: OpenModelica Graphical Editor and Debugger

Adeel Asghar and Peter Fritzson

EOOLT Workshop
April 19, 2013, Nottingham
OpenModelica OMEdit Graphic Editor

GUI Improvements in coming 1.9.0 final release (now available in nightly builds)

• Better Package support
• Better and simpler access to parameters
• Better model documentation display
OpenModelica OMEDit Graphic Editor & GUI
Plot Example

Show Demo Movie
Static vs Dynamic Debugging

- **Static Debugging**
  - Analyze the model/program at compile-time
  - Explain inconsistencies and errors, trace error dependencies
  - Example: Underconstrained/overconstrained systems of equations
  - Example: errors in symbolic transformations of models

- **Dynamic Debugging**
  - Find sources of errors at run-time, for a particular execution
  - **Declarative dynamic debugging** – compare the execution with a specification and semi-automatically find the location of the error
  - **Traditional dynamic debugging** – interactively step through the program, set breakpoints, display and modify data structures, trace, stack inspection

- **Goal: Integrated Static and Dynamic Debugging**
Dynamic Debugging

Large Modelica Algorithmic Code Models
Tool Architecture and Communication

Modelica Model

OpenModelica Compiler

Gnu Compiler

C Code

Executable

Modelica source code positions are mapped to C source code positions

Debugger Graphical User Interface

GDB-MI
Example Mapping Modelica Positions to C Code

Convert Modelica code to C source code by adding Modelica line number references.
Debugger Integrated in Eclipse OpenModelica MDT Environment

- Eclipse plugin MDT (Modelica Development Tooling) is the integrated development environment
- Debugger is a debug plug-in within MDT
Static Debugging

Transformational Debugging of Equation-Based Models
Debugging Equation Systems

Modelica Compiler Backend
- Complex mathematical transformations
- Hidden to users
- Users want to access this information
- Not intuitive
  - No explicit control flow
  - Numerical solvers
  - Linear/Non-linear blocks
  - Optimization
  - Events
Tracing Symbolic Transformations

- Simple Idea
  - Store transformations as equation metadata

- Works best for operations on single equations
  - Alias Elimination \((a = b)\)
  - Equation solving \((f_1(a,b) = f_2(a,b), \text{solve for } a)\)

- Multiple equations require special handling
  - Gaussian Elimination (linear systems, several equations)
  - ...
Tracing Overhead?

- OpenModelica compiler implementation is so fast that tracing is enabled by default
  - 1 extra comparison and/or cons operation per optimization
  - Not noticeable during normal compilation
- No overhead unless you output the trace
The alias relation $a=b$ stored in variable $a$

The equations are e.g. stored as $(lhs,rhs,list<ops>)$

```
\begin{align*}
a &= b \\
c &= a + b \\
d &= a - b \\
c &= a + b \text{ (subst } a=b) \Rightarrow \\
c &= b + b \text{ (simplify) } \Rightarrow \\
c &= 2 \times b \\
d &= a - b \text{ (subst } a=b) \Rightarrow \\
d &= b - b \text{ (simplify) } \Rightarrow \\
d &= 0.0
\end{align*}
```
Trace Example (1)

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

(1) substitution:
  y + der(x * (time * z))
  =>
  y + der(x * (time * 1.0))

(2) simplify:
  y + der(x * (time * 1.0))
  =>
  y + der(x * time)

z = 1.0;

(3) expand derivative
(sympollic diff):
  y + der(x * time)
  =>
  y + (x + der(x) * time)

(4) solve:
  0.0 = y + (x + der(x) * time)
  =>
  der(x) = ((-y) - x) / time