Debugging Symbolic Transformations in Equation Systems

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

4th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools
September 2011, ETH Zürich, Switzerland
Debugging EOO Languages

- 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

...
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 $a$)
  - ...
Alias Elimination

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

- The alias relation \(a=b\) stored in variable \(a\)
- The equations are e.g. stored as \((\text{lhs, rhs, list<ops>})\)
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:
  \( y + \text{der}(x \times (\text{time} \times z)) \)
  =>
  \( y + \text{der}(x \times (\text{time} \times 1.0)) \)

(2) simplify:
  \( y + \text{der}(x \times (\text{time} \times 1.0)) \)
  =>
  \( y + \text{der}(x \times \text{time}) \)

(3) expand derivative (symbolic diff):
  \( y + \text{der}(x \times \text{time}) \)
  =>
  \( y + (x + \text{der}(x) \times \text{time}) \)

(4) solve:
  0.0 = y + (x + \text{der}(x) \times \text{time})
  =>
  \( \text{der}(x) = ((-y) - x) / \text{time} \)
## Trace of Dummy Derivatives Alg.

<table>
<thead>
<tr>
<th>Differentiation</th>
<th>Substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{d}{dt} L^2$</td>
<td>$2.0 \times (\text{der}(x) \times x + \text{der}(y) \times y)$</td>
</tr>
<tr>
<td>$\Rightarrow$</td>
<td>$\Rightarrow$</td>
</tr>
<tr>
<td>$0.0$</td>
<td>$2.0 \times (\text{DER}.x \times x + \text{DER}.y \times y)$</td>
</tr>
<tr>
<td>Differentiation</td>
<td>$\Rightarrow$</td>
</tr>
<tr>
<td>$\frac{d}{dt} x^2 + y^2$</td>
<td>$2.0 \times (u \times x + \text{DER}.y \times y)$</td>
</tr>
<tr>
<td>$\Rightarrow$</td>
<td>$\Rightarrow$</td>
</tr>
<tr>
<td>$2.0 \times (\text{der}(x) \times x + \text{der}(y) \times y)$</td>
<td>$2.0 \times (u \times x + v \times y)$</td>
</tr>
<tr>
<td>$\Rightarrow$</td>
<td>$\Rightarrow$</td>
</tr>
<tr>
<td></td>
<td>$2.0 \times (u \times xloc[1] + v \times xloc[0])$</td>
</tr>
</tbody>
</table>
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