Separate Compilation of Causalized Equations
EOOLT 2011

Christoph Höger
Technische Universität Berlin
Scope of this talk

- This talk focuses on compilation of Modelica.
- Should be applicable to any other equation based modeling language.
- The research is currently ongoing, not everything is solved yet.
Currently no available tool *compiles* Modelica

Instead, a complete model is *interpreted*

(The resulting system of equations is afterwards compiled)

Even the Specification insists on that method
(Separate) compilation of Modelica is a (solvable) Challenge.

- Advantages: early checking, distribution of reusable fragments, space saving, compilation time
- But what about simulation code efficiency?
Compiling Equations

\[ f(x) = g(y) \]

- Naive approach: generate residuals:
  \[ r(x, y) = f(x) - g(y) = 0 \]
  - One function per equation
  - Very inefficient compared to decent interpreters

- Necessary: causalized equation:
  \[ x = f^{-1}(g(y)) \]
Compiling Equations

\[ f(x) = g(y) \]

- Naive approach: generate residuals:
  \[ r(x, y) = f(x) - g(y) = 0 \]

- One function per equation
- Very inefficient compared to decent interpreters

- Necessary: causalized equation:
  \[ x = f^{-1}(g(y)) \]
Wrong order?

But where do we know that $x$ shall be solved for?

Turns out: we cannot!
Causalization as a Graph Problem

\[ f(x) = g(y) \]

Topological order on a perfect match

\[ x = f^{-1}(g(y)) \]

\[ y = g^{-1}(f(x)) \]
At compile-time it is undecidable which solution will be used

Create one block for every variable in an equation.

Every block contains an unknown (to solve for) and an expression

Generate code for each block

Collect blocks at runtime

Run causalization on blocks

\[ y := g^{-1}(f(x)) \]

\[ x := f^{-1}(g(y)) \]
Performance

Prototype implementation, scaled on number of model instances

![Graph showing runtime in seconds vs. instances of test2 for omc and TinyModelica]
Results

What has been achieved?

- Block-generation allows separate compilation of equations.
- Thus it allows compilation of partial models.
- Model libraries might be redistributed.
- Generated code can be (nearly) as fast as usual.
- Generated code can be *much* smaller.
Future Work

What needs to be done?

- Index reduction.
- Further optimizations.
- Discrete expressions that yield variables e.g. array access
- True structural dynamics.
Thank you for your attention!

Any Questions?