Experimentation with a Prototype OpenModelica Compiler in Julia

John Tinnerholm, Adrian Pop, and Martin Sjölund

LINKÖPING UNIVERSITY

Motivation

Motivation

- Integration with the Julia ecosystem
- Provide a standard-compliant Modelica environment in Julia
  - Larger OpenSource community
  - Supporting VSS/Multi-mode DAE models within standard Modelica
- Outsourcing implementation language development
- Enable separate use of OpenModelica Compiler packages
  - Utilizing Julia package manager
  - Tool development without a monolithic compiler

Julia

- A new programming language by Jeff Bezanson, Stefan Karpinski and Viral B. Shah
- A language for Numerical and Symbolic Computation
- Many libraries for Linear Algebra, Differential Equations, Fast Fourier Transforms etc.
- H. Wilkinson Prize for Numerical Software in 2019
- Already in use for equation-based-modelling, Modia.jl
  - Support for Multi-Mode DAE Models, however non standard-compliant
- OMJulia for interoperability with OpenModelica

The MetaModelica to Julia translator

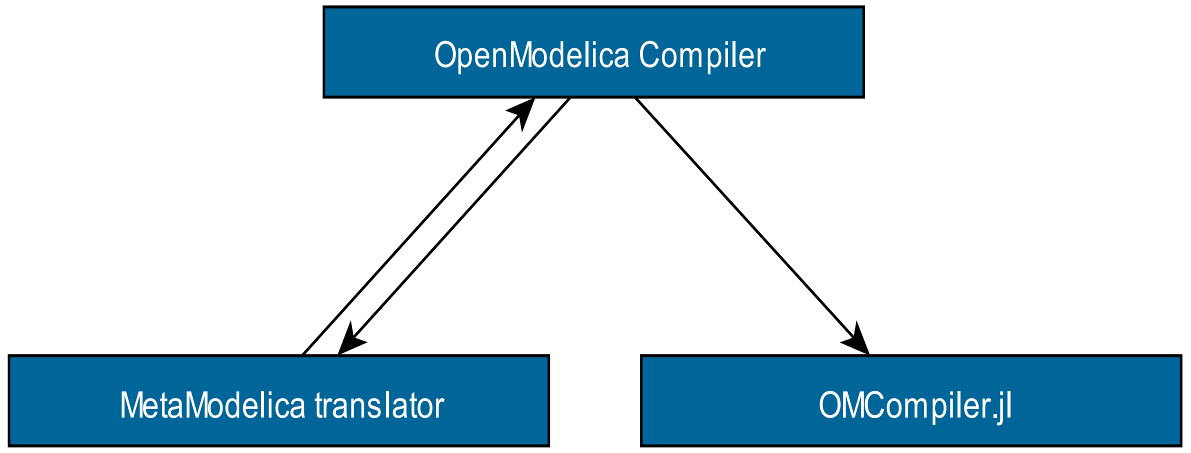
The MetaModelica to Julia translator

- Translated from the old OpenModelica frontend
- Maps MetaModelica packages to Julia modules
- Difficulties with certain syntactical constructs
  - Circular packages
  - Uniontypes with static methods

```
uniontype Equation
  record EQ_IF
    Exp ifExp;
    list<EquationItem> equationTrueItems;
    list<tuple<Exp,
    list<EquationItem>>> elseIfBranches;
    list<EquationItem> equationElseItems;
  end EQ_IF;

  record EQ_EQUALS
    Exp leftSide;
    Exp rightSide;
  end EQ_EQUALS;
  // ...
```

The MetaModelica to Julia translator

- Continuous integration of features
- Supporting parallel development of the existing compiler



MetaModelica via Metaprogramming

- Julia's AST macros makes language extensions easy
- Structural elements such as inheritance, uniontypes match and matchcontinue are provided

```
@Uniontype Equation begin
 @Record EQ_IF begin
  ifExp::Exp
  equationTrueItems::List{EquationItem}
  elseIfBranches::List{
    Tuple{Exp,List{EquationItem}}}
  equationElseItems::List{EquationItem}
 end

 @Record EQ_EQUALS begin
  leftSide::Exp
  rightSide::Exp
 end
 # ...
end
```

```
uniontype Equation
  record EQ_IF
    Exp ifExp;
    list<EquationItem> equationTrueItems;
    list<tuple<Exp,
    list<EquationItem>>> elseIfBranches;
    list<EquationItem> equationElseItems;
  end EQ_IF;

  record EQ_EQUALS
    Exp leftSide;
    Exp rightSide;
  end EQ_EQUALS;
  // ...
end
```

MetaModelica.jl

- Compiler runtime
  - MetaModelica.jl
  - Pattern-matching
  - Immutable List
  - Uniontypes
- Possible to use as a standalone package

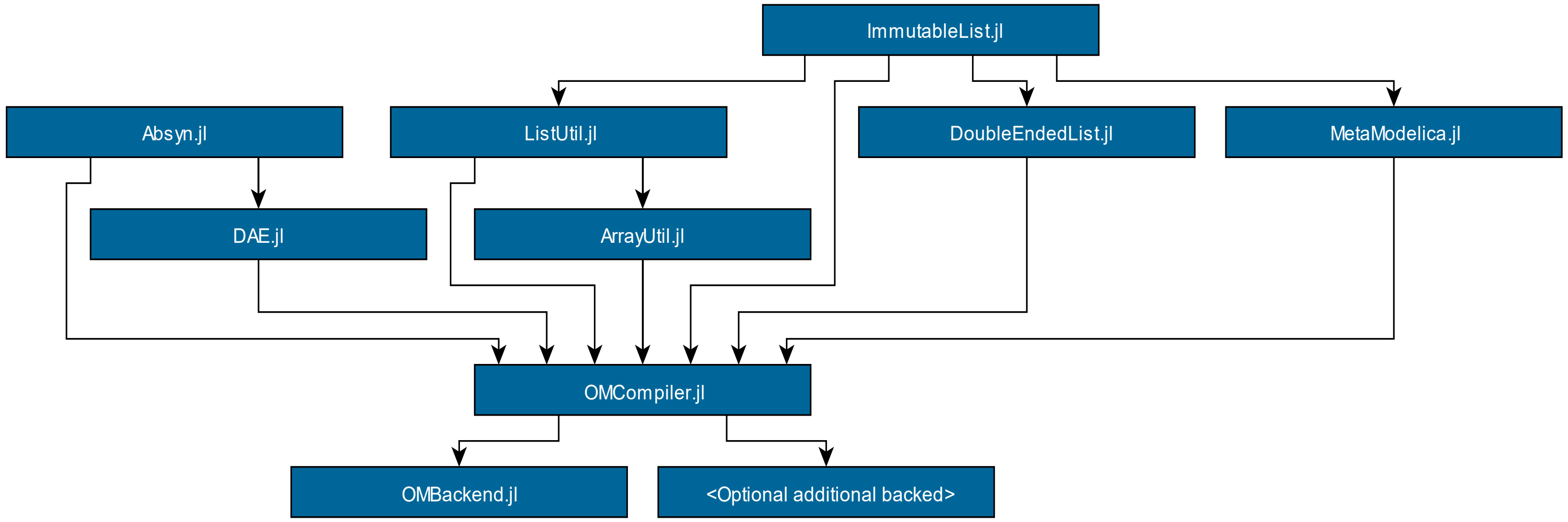
```
@Uniontype Equation begin
 @Record EQ_IF begin
  ifExp::Exp
  equationTrueItems::List{EquationItem}
  elseIfBranches::List{
    Tuple{Exp,List{EquationItem}}}
  equationElseItems::List{EquationItem}
 end

 @Record EQ_EQUALS begin
  leftSide::Exp
  rightSide::Exp
 end
 # ...
end
```

OMParser.jl

- The AST is constructed in ANTLR using C the foreign function interface of Julia
  - Capable of parsing any Modelica library (tested with MSL 3.2.3)
- Somewhat smaller memory footprint
  - Better data sharing

OMCompiler.jl

- Translated from the current frontend
- Able to parse and translate:
  - Absyn IR
  - SCode IR
  - DAE IR



OMCompiler.jl

- Generated Julia DAE IR compatible with existing MetaModelica DAE IR
- Somewhat smaller memory footprint

```
dae = DAE.DAE_LIST(Cons {
  DAE.Element
}(DAE.COMP("HelloWorld", Cons {
  DAE.Element
}(DAE.VAR(DAE.CREF_IDENT("x", DAE.T_REAL(Nil {
  Any
}()), Nil {
  Any
}()), DAE.VARIABLE(), DAE.BIDIR(), DAE.NON_PARALLEL(), DAE.PUBLIC(), DAE.T_REAL(Nil {
  Any
}()), nothing, Nil {
  Any
...
  Any
}()), SOME {
  SCode.Comment
} (SCode.COMMENT(nothing, nothing))), Nil {
Any
}()))
```

Future work

Future work

- Translating the new high performance frontend
- A Simulation runtime
- OMBackend.jl
- Investigate possible integration with:
  - Modia.jl
  - DifferentialEquations.jl

About the dragon?

Future work

- Dragon
  - LLVM
- Cogwheels
  - Modelica
- Colors?
  - Julia

Questions?

www.liu.se

LINKÖPING
UNIVERSITY

References

Elmqvist, H., Henningsson, T., & Otter, M. (2017). Innovations for Future Modelica. *Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017*, *132*, 693–702. https://doi.org/10.3384/ecp17132693

Zimmer, D. (2010). *Equation-based modeling of variable-structure systems*. (18924), 219. https://doi.org/10.3929/ethz-a-006053740

References

Lie, B., Palanisamy, A., Mengist, A., Buffoni, L., Sjölund, M., Asghar, A., … Fritzson, P. (2019). OMJulia: An OpenModelica API for Julia-Modelica Interaction. *Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4–6, 2019*, *157*, 699–708. https://doi.org/10.3384/ecp19157699

References

Casella, F. (2015). Simulation of Large-Scale Models in Modelica: State of the Art and Future Perspectives. *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*, *118*, 459–468. https://doi.org/10.3384/ecp15118459

Modeling, V., & Höger, C. (n.d.). *Elaborate Control*.

VADIM I. UTKIN. (1977). Variable Structure Systems with Sliding Modes. *IEEE Transactions on Automatic Control*, *22*(2), 212–222.

References

Benveniste, A., Ghorbal, K., Caillaud, B., Otter, M., Elmqvist, H., & Pouzet, M. (2017). Structural analysis of multi-mode DAE systems. *HSCC 2017 - Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control (Part of CPS Week)*, 253–263. https://doi.org/10.1145/3049797.3049806

References

Benveniste, A., Ghorbal, K., Caillaud, B., Otter, M., Elmqvist, H., & Pouzet, M. (2017). Structural analysis of multi-mode DAE systems. *HSCC 2017 - Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control (Part of CPS Week)*, 253–263. https://doi.org/10.1145/3049797.3049806

Lattner, C., & Adve, V. (2004, March). LLVM: A compilation framework for lifelong program analysis & transformation. In *International Symposium on Code Generation and Optimization, 2004. CGO 2004.* (pp. 75-86). IEE

References

https://github.com/OpenModelica/MetaModelica.jl

https://github.com/JKRT/OMCompiler.jl

https://github.com/OpenModelica/Absyn.jl

…