Towards an unified OpenModelica Simulation Interface - OMSI
Current development status

Andreas Heuermann\textsuperscript{1} \hspace{1cm} Willi Braun\textsuperscript{1} \hspace{1cm} Niklas Worschech\textsuperscript{2} \hspace{1cm} Bernhard Bachmann\textsuperscript{1}

\textsuperscript{1} FH Bielefeld University of Applied Sciences \hspace{1cm} Bosch Rexroth \textsuperscript{2}
Bielefeld, Germany \hspace{1cm} Lohr, Germany

February 2, 2019
1 Motivation

2 OMSI structure
   - New C structure
   - SimCode and Templates
   - Call structure for OMSIC and OMSICpp
   - Solver interface
   - OMSI simulation runtime

3 Current development status
   - Implemented features
   - Modelica Standard Library Coverage

4 Summary
Motivation

Current issues

- SimCode data structure got bloated over time
- SimCode not independent of target language
- No clear separation between model and runtime data
Motivation

Current issues

C runtime

ModelicaTest_3.2.2 test using OpenModelica

New features require development for each runtime
⇒ Different performance of runtimes
Motivation

**Goal: Make developer life more simple**

- Unify data structures in back-end and simulation runtimes
- Share code generation for C and C++
- Clear interface and shared functionalities
  - Base: Use common base functionalities and solvers
  - C-Runtime: Simulation runtime in ANSI C for e.g. realtime applications
  - Cpp-Runtime: Simulation runtime for e.g desktop applications
Motivation

Goal: Make developer life more simple

- Unify data structures in back-end and simulation runtimes
- Share code generation for C and C++
- Clear interface and shared functionalities
  - Base: Use common base functionalities and solvers
  - C-Runtime: Simulation runtime in ANSI C for e.g. realtime applications
  - Cpp-Runtime: Simulation runtime for e.g desktop applications
Motivation

**Goal: Make developer life more simple**

- Unify data structures in back-end and simulation runtimes
- Share code generation for C and C++
- Clear interface and shared functionalities
  - Base: Use common base functionalities and solvers
  - C-Runtime: Simulation runtime in ANSI C for e.g. realtime applications
  - Cpp-Runtime: Simulation runtime for e.g desktop applications
Motivation

Goal: Make developer life more simple
- Unify data structures in back-end and simulation runtimes
- Share code generation for C and C++
- Clear interface and shared functionalities
  - Base: Use common base functionalities and solvers
  - C-Runtime: Simulation runtime in ANSI C for e.g. realtime applications
  - Cpp-Runtime: Simulation runtime for e.g desktop applications

Free extra
- Better FMI support
OpenModelica Simulation Interface (OMSI)
Outline

1. Motivation
2. OMSI structure
   - New C structure
   - SimCode and Templates
   - Call structure for OMSIC and OMSICpp
   - Solver interface
   - OMSI simulation runtime
3. Current development status
   - Implemented features
   - Modelica Standard Library Coverage
4. Summary
Overview

New C structure

Targets
- Separate data to smaller segments
- Store data thread-safe
- Forward only necessary information
- Don’t copy to much memory
Include new OMSIData in SimCode

```haskell
uniontype OMSIData =
  "contains data for code generation for OMSI"

record OMSI_DATA =
  "OMSIFunction initialization: "contains equations and variables for initialization problem"
  "OMSIFunction simulation: contains equations and variables for simulation problem"
end OMSI_DATA;
end OMSIData;

uniontype OMSIFunction =
  "contains equations and variables for initialization or simulation problem"

record OMSI_FUNCTION =
  list<SimEqSystem> equations = "list of single equations and systems of equations"
  list<SimCodeVar> SimVar> inputVars = "list of simcode variables determining input variables for equation(s)"
  list<SimCodeVar> SimVar> outputVars = "list of simcode variables determining output variables for equation(s)"
  list<SimCodeVar> SimVar> innerVars = "list of simcode variables determining inner variables for equation(s), e.g. $DER(x)$"
  Integer nAILVars = "number of input, inner and output vars"
  SimCodeFunction.Context context = "contains cref SimVar hash table for lookup function in templates"
  Integer nAlgebraicSystems = "number of linear and non-linear algebraic systems in OMSI_FUNCTION.equations"
end OMSI_FUNCTION;
end OMSIFunction;
```

Shared functions for code generation
Overview
Redesign of SimCode structure

- Include new OMSIData in SimCode

```haskell
uniontype OMSIData
  "contains data for code generation for OMSI"
end OMSIData;

uniontype OMSIFunction
  "contains equations and variables for initialization or simulation problem"
end OMSIFunction;
```

- Shared functions for code generation
Overview

Redesign of SimCode structure

- Include new OMSIData in SimCode

```haskell
uniontype OMSIData =
  "contains data for code generation for OMSI"

record OMSI_DATA =
  OMSIFunction_initialization = "contains equations and variables for initialization problem"
  OMSIFunction_simulation = "contains equations and variables for simulation problem"
end OMSI_DATA;
end OMSIData;
```

- Shared functions for code generation
Generate some code

- Use common functions in templates
- Files for omsi_function and omsi_alg_systems
- Generate comparable code for equations
Generate some code

- Use common functions in templates
- Files for omsi_function and omsi_alg_systems
- Generate comparable code for equations
Generate some code:
- Use common functions in templates
- Files for omsi_function and omsi_alg_systems
- Generate comparable code for equations
Overview

Templates

Generate some code

- Use common functions in templates
- Files for omsi_function and omsi_alg_system
- Generate comparable code for equations
Generate some code

- Use common functions in templates
- Files for omsi_function and omsi_alg_systems
- Generate comparable code for equations
Generate some code

- Use common functions in templates
- Files for omsi_function and omsi_alg_systems
- Generate comparable code for equations
Overview

Call structure for OMSIC and OMSICpp
Overview

Solver

OMSISolver

<<Interface>>

Solver Api

Attributes
+ solver_callbacks

Operations
+ allocate ()
+ free ()
+ prepare_specific_data ()
+ set_start_vector ()
+ set_solver ()
+ solve_system ()
+ getXXX ()
+ setXXX ()

Solver

Attributes
+ name: string
+ is_linear: bool
- specific_solver_data

Operations
+ solve_system ()
# solver_specific_functions ()

LAPACK Solver

Attributes
# name = "solver_lapack"
# is_linear = true
- data: LAPACK_DATA

Operations
#

Homotopy Solver

Attributes
# name = "solver_homotopy"
# is_linear = false
- data: HOMOTOPY_DATA

Operations
Overview
OMSU simulation with OMSICpp

OMSU Cpp simulation runtime
- Simulate OMSU or FMI 2.0 Model Exchange FMU
- Optional arguments passed to simulation executable, e.g. experiment settings
- Aim to use OMSI ODE solver
Outline

1 Motivation

2 OMSI structure
   - New C structure
   - SimCode and Templates
   - Call structure for OMSIC and OMSICpp
   - Solver interface
   - OMSI simulation runtime

3 Current development status
   - Implemented features
   - Modelica Standard Library Coverage

4 Summary
Current development status

Implemented features

What’s working at the moment

- Code generation equations and equation systems
- Solving of linear and non-linear algebraic loops
- Event handling
- Exporting of Model Exchange FMU’s
Modelica Standard Library - 3.2.2

- Build OMSIC FMU and import with OpenModelica to generate simulation executable
- Tested on Ubuntu Bionic (18.04)
- First models are building and simulating
- Just started, great improvements are to be expected
Current development status

Next steps

- Complete code generation for all equations and equation systems
- Validate results automatically
- Test OMSICpp
- Merge first version into Modelica master
- Increase number of usable solvers on OMSISolver library
Outline

1 Motivation

2 OMSI structure
   - New C structure
   - SimCode and Templates
   - Call structure for OMSIC and OMSICpp
   - Solver interface
   - OMSI simulation runtime

3 Current development status
   - Implemented features
   - Modelica Standard Library Coverage

4 Summary
Summary

- Unified data structures in SimCode, Templates and runtimes
- Reduction of maintenance work
- Shared solver library
- Support of FMI 2.0 (ModelExchange)
- First working simulations

Open tasks

- Complete SimCode and Templates
- Reach high MSL coverage
- Switch from ”old” C and Cpp runtime to OMSI runtime
- Exploit parallelism
Summary

- Unified data structures in SimCode, Templates and runtimes
- Reduction of maintenance work
- Shared solver library
- Support of FMI 2.0 (ModelExchange)
- First working simulations

Open tasks

- Complete SimCode and Templates
- Reach high MSL coverage
- Switch from "old" C and Cpp runtime to OMSI runtime
- Exploit parallelism

Thank you for your attention!
Summary

- Unified data structures in SimCode, Templates and runtimes
- Reduction of maintenance work
- Shared solver library
- Support of FMI 2.0 (ModelExchange)
- First working simulations

Open tasks

- Complete SimCode and Templates
- Reach high MSL coverage
- Switch from "old" C and Cpp runtime to OMSI runtime
- Exploit parallelism

Questions?