Master Theses in Development of Simulation, Optimization and Compiler Tools

Introduction

Simulation and optimization of dynamic systems is becoming a standard tool in several industrial branches. This trend is largely driven by the need to decrease product time-to-market. In order to increase productivity in the product design phase, model-based approaches are increasingly used. To meet the demand for model-based design methods and tools, domain specific languages have been developed. One such language is Modelica, which is a language targeted at modeling of complex heterogeneous physical systems. Modelica is currently used in a wide range of applications, including automotive systems, power plants, thermo-fluid systems, and robotics.

JModelica.org and OPTIMICA Compiler Toolkit

Most of the theses proposals in this document are related to JModelica.org (http://www.jmodelica.org), which is an open-source Modelica-based simulation and optimization environment, which also supports the Modelica language extension Optimica. A main part of the JModelica.org platform is the Modelica and Optimica compilers which are developed using the JastAdd framework. One product based on JModelica.org is OPTIMICA Compiler Toolkit (OCT). It is based on JModelica.org and extends it with several unique features. JModelica.org and OPTIMICA Compiler Toolkit is maintained and developed by Modelon AB (http://www.modelon.com).

Contact: Johan Åkesson, Modelon AB

Runtime instantiation in JModelica.org

Mostly, compilers perform runtime instantiation, allocating memory for instances during execution. In order to simplify the compilation process of Modelica the JModelica.org compiler performs instantiation during early compilation. We have identified a few cases where it can be benefitial to delay the instantiation until execution, or at least code generation, which would lead to reduced execution time and memory consumption for the compiler.

The aim of this thesis would be to implement the delayed instantiation for a few cases and give us an understanding of how it would work in more complex cases.

Student profile: Two skilled and motivated students, who has taken a course in compiler construction (preferably with good grades), and has interest in compiler development and Java programming.

Contact: Jonathan Kämpe, Modelon AB

Interactive HTML diagnostics in JModelica.org

Good compiler diagnostics is a powerful tool when debugging large models. JModelica.org currently have a limited debugging diagnostics in the form of a few static HTML pages that are generated during compilation. The goal of the proposed master's thesis project is to further improve compiler diagnostics reports and utilize the powers of HTML, JavaScript and CSS to offer the user with a more interactive diagnostics tool. The different parts of the project are:

- Determining the requirements by surveying and interact with model developers and determine their workflow.
- Improve existing HTML diagnostics by incorporating CSS and JavaScript to visualize the relationship between equations, variables and their computation order.

Student profile: One skilled and motivated student with interest in compilers, Java, HTML, JavaScript and CSS development, who has taken a course in compiler construction.

Contact: Jon Sten, Modelon AB

High-performance storage of simulation data

Modern tools for simulation of large-scale dynamic systems are becoming a common technology in a wide range of industrial domains, including avionics, automotive, power plants, electronics and robotics. State of the art tools are capable of simulating models with more than 100.000 equations, which in turn generates large amounts of simulation data that needs to be stored in an efficient manner. To meet this need, a new data storage format is under development, combining two industrial standards: the Functional Mock-up Interface (FMI) for exchange of compiled dynamic models, and HDF5, which is a general purpose storage format based on XML and binary data storage. The task in this project is to implement support for the new storage format in the open source Modelica platform JModelica.org. The project includes a number of challenges, e.g., development of a user-friendly software design, learning two industrial standards, and obtaining high performance. The implementation will be validated on industrial grade simulation models.

The project is done within the scope of the JModelica.org open source project.

Student profile: One skilled and motivated student with interest in numerical algorithms and programming. Prior knowledge of the C programming language, Modelica and XML is considered a merit, but is not mandatory.

Contact: Christian Andersson, Modelon AB

FMI Bindings for the Julia language

The Functional Mockup Interface (FMI) represents a common language for model exchange between many modelling and simulation environments. It describes how FMUs, models that have been compiled into binary code, can be used to simulate components within a larger system.

The aim of this thesis project is to implement initial support for working with FMUs in the open source Julia programming language. Julia is a high-level, high-performance dynamic programming language for technical computing, which is gaining increasing attention for its execution speed and expressiveness. The existing PyFMI Python package already provides similar functionality for working with FMUs, and can inform the design, but there will be a need to rethink the design in terms of Julia’s programming paradigm. The basis of the project will be to create a Julia package with Julia bindings (with the aid of a wrapper generator) to FMI Library, a C library for loading and low level access to FMUs. These bindings will then be used to implement FMU simulation functionality in Julia by interfacing to existing solvers. Developed functionality will be hosted as one or more open source Julia packages on github.com.

The project is done at the company Modelon AB located at IDEON, Lund.

Student profile: Two skilled and highly motivated students with interest in numerics, programming and new programming languages. Prior knowledge of C, Python, and Julia is considered a merit, but is not mandatory.

Contact: Jonathan Kämpe, Modelon AB

Equation scaling in steady state solver

OPTIMICA Compiler Toolkit (OCT) contains non-linear equation solvers which are used for steady-state simulations and for solving the algebraic equations as a part of dynamics simulations. Scaling of the equations and iteration variables is utilized to compensate for ill-conditioned systems. Moreover, the solver takes into account the bounds on the iteration variables that indicate either model validity range or physically relevant regions.

In this project different scaling and re-scaling strategies are to be investigated. The scaling strategy should be implemented in a modular way and as an extension of the Kinsol and Minpack solver codes where additional debug logging will be inserted to facilitate the analysis. The project will include creating a collection of relevant test cases to be used as benchmarks.

The project will be run within the open-source JModelica.ORG platform at Modelon AB in Gothenburg.

Student profile: One (or two) skilled and highly motivated student(s) with interest in numerical algorithms and programming. Prior knowledge of C and MATLAB is considered a merit, but is not mandatory.

Contact: Iakov Nakhimovski & Agnes Ramle, Modelon AB

Improved model testing using compiler API

Testing of software is an essential part of any workflow. Modelon have 14 Modelica libraries full of models that are all continuously developed. To ensure that the development is not breaking any tests it is best practice to run tests locally before committing to trunk, this is however very inefficient when the test suite takes hours to run.
This theses proposes to use the compiler API of the OPTIMICA Compiler Toolkit to make testing more efficient. Ideas include, but are not limited to:

- Given a model or component, run all associated tests.
- Given a line of code in a file, run all associated tests.
- Compare parameter settings between models.

Student profile: One skilled and motivated student with interest in compilers, Java and possibly Python, who has taken a course in compiler construction.

Contact: Johan Ylikiiskilä, Modelon AB