MASTER THESES IN DEVELOPMENT OF SIMULATION, OPTIMIZATION AND COMPILER TOOLS

Contents

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.

OPTIMICA Compiler Toolkit

Most of the theses proposals in this document are related to the OPTIMICA Compiler Toolkit (OCT), which is an Modelica-based simulation and optimization environment, which also supports the Modelica language extension Optimica. A main part of the toolkit is the Modelica and Optimica compilers which are developed using the JastAdd framework. OPTIMICA Compiler Toolkit is maintained and developed by Modelon AB.

Contact: Johan Åkesson, Modelon AB

Runtime instantiation in OCT

Mostly, compilers perform runtime instantiation, allocating memory for instances during execution. In order to simplify the compilation process of Modelica the OCT compiler performs instantiation during early compilation. We have identified a few cases where it can be beneficial 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

Good compiler diagnostics is a powerful tool when debugging large models. Our simulation platform currently have 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: Johan Åkesson, 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 needed in OCT. The task in this project is to investigate and evaluate existing storage format and, if relevant, propose a new format. The format is then to be implemented in OCT. The project includes several challenges, e.g., development of a user-friendly software design, learning industrial standards, and obtaining high performance. The implementation will be validated on industrial grade simulation models.

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 Winther, Modelon AB

Improve JastAdd generated code

The Modelica compiler inside OCT is built using the JastAdd framework. Using JastAdd to build compilers comes with many benefits and OCT would not be where it is today without JastAdd.

Using JastAdd however, means also that we are dependent on how JastAdd generates code and for performance, how efficient the code generated is.

This master thesis focuses on the generated JastAdd code in the context of OCT. The goal of the master thesis is to investigate the performance of the generated JastAdd code and propose changes to JastAdd that will increase the performance of OCT.

Student profile: Two skilled and motivated students, who has taken a course in compiler construction, and has interest in compiler development and Java programming.

Contact: Christian Winther, Modelon AB

Index reduction with arrays

In most Modelica compilers, there are algorithms that transforms a general Modelica model, defined by differential algebraic equations into a model defined by ordinary differential equations. This process is performed (simplified) using index reduction.

Current state of the art is that it is performed on scalar equations and scalar variables. This is troublesome for several reasons – the primary reason is that if large arrays or matrices are used, these must be split into single variables which is inefficient. Both in terms of memory and performance.

Recent advancements have been made to the index reduction algorithm by mapping it to array equations. This master thesis will investigate the new method and implement as well as evaluate the method inside OCT.

Student profile: Two skilled and motivated students, who has taken a course in compiler construction, and has interest in compiler development and Java programming.

Contact: Christian Winther, Modelon AB

Dynamic support for mixing FMUs of different bitset

The Functional Mock-up Interface (FMI) is a standard designed to provide a unified model execution interface for dynamic system models between modelling and simulation tools. The standard has become highly successful with numerous tools, both commercial and open-source, supporting it. A foundation for many importing tools supporting the standard is the FMI Library, an open-source (C based) implementation that provides most of the low-level functionality necessary for working with models following the standard. The problem that many tools face is that the models are either 32bit or 64bit. This pose a problem when trying to integrate the models into a larger system model. This master thesis proposal aims at investigating and implement support in FMI Library to handle mixing of models with different bitset, potentially using a client/server setup.

Student profile: One (or two) skilled and highly motivated student(s) with interests in programming. Prior knowledge of C is mandatory.

Contact: Christian Winther, Modelon AB

Python model diagnostics package

Debugging models that are not initializing or simulating can take a lot of time and be very hard without useful debugging tools. The aim of this thesis is to create a Python package based on the output from the solvers in OCT and diagnostics provided by the PyFMI package which gives the modeler clues about what can be changed in the model to reach convergence.

The challenge of the project is to find a bridge between the numerical issues in the different kinds of solvers that are part of OCT FMUs and the original model. It requires deep understanding of the underlying solvers and what can cause problems. Creating smaller (problematic) example models on which the package is applied will be part of the project.

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

Contact: Agnes Ramle, Modelon AB

System modeling and simulation platform of the future

Modelon is building a system modeling and simulation platform of the future with the goal of creating a robust commercial platform available in the cloud. The solution needs highly interactive user interface for system modeling, massive parallelization of numerical simulations, post processing of large data sets and visualization in 2D and 3D. Collaborative system design preferred by modern engineers is a key capability of the platform, as well as high standards for IT security to keep sensitive customer product data safe.

As a master's thesis student, you will interact with experts within Modelon in a range of fields, including physical modeling, numerical algorithms, systems design and compiler technology, all disciplines needed to create a great system design platform. You will work closely with the development team and gain experiences with agile software development practices, including Scrum, code review and pair programming, all of which are key elements of Modelon's software development process.

We offer master's thesis projects in a **broad range of areas**, including:

- 3D visualization of system simulation, including vehicles
- 3D editing of system models
- Customized client web apps to support design workflows in model-based design
- Enhanced system modeling capabilities, e.g., icon editing and documentation generation, and composition of compiled models (FMUs)
- Advanced and configurable visualization of computational results (3D plots, scatter plots etc.)

Student profile: One or two skilled and highly motivated student(s) with interest in numerical algorithms and UI programming. Prior knowledge of Python, JavaScript and 3D applications is considered a merit.

Contact: Johan Åkesson, Modelon AB