Generation of Functional Mock-up Units from Causal Block Diagrams Reading Report

Bavo Vander Henst

University of Antwerp
Model Driven Engineering
Bavo.VanderHenst@student.uantwerpen.be

Abstract

In this report we will investigate related work of our project. Because there are not a lot of papers about Causal Block Diagrams we will mainly focus on the Functional Mockup Interface (FMI). Our work is closely related to that of Bart Pussig, so we will also examine his paper.

Keywords: Causal Block Diagrams, Optimazation, Functional Mockup Units, Co-simulation

1. Introduction

Models are increasingly used within software engineering to simulate the behaviour of a system. These models can be represented in different formalisms which makes co-simulation impossible. The Functional Mockup Interface [1] tries to solve this problem by defining an interface for co-simulation. In this project we will try to generate Functional Mockup Units (FMUs) from Causal Block Diagrams implemented in Python.

An additional part of this project is to try to optimize the CBD before the translation to the FMU to gain computation time while running the FMU.

2. FMI

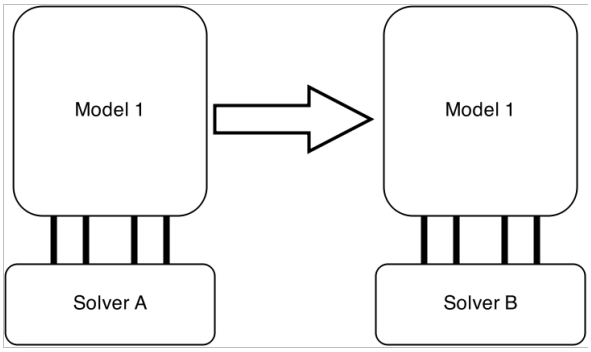
We start by looking at The Functional Mockup Interface. The FMI consists of two main parts.

1. *FMI for Model Exchange*
2. *FMI for Co-Simulation*

The main idea of both interfaces is to create a standard for files so they can be exchanged. We will look at the two parts individually.

2.1. Model Exchange

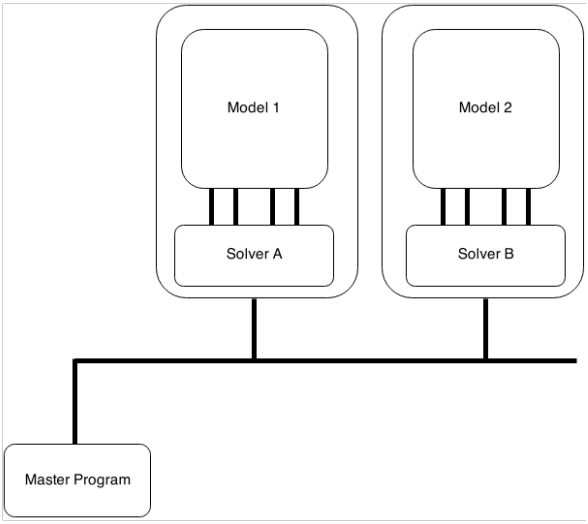
The interface of model exchange consists of some functions that can be called by the solver. These functions can be used to request the internal state of the model. Because of this *Black Box* that is created, we are able to exchange our model with someone else. We can also make a new solver without remaking our model.



Because the models inside can be compiled, which also helps to protect intellectual property, the model can be used, but not viewed.

2.2. Co-Simulation

The second interface is to couple two or more simulation tools (solvers) with each other. This can be needed if you don't want to recompile multiple models together or if you don't have the source code of the models. Here the interface isn't located between the model and the solver, but between the solver and a master program. The master program is responsible to syncronize all the simulations and to pass data among them.



As we can see in the image, all the simulation kernels (model + solver) are connected to a bus. The master program as well is connected to the same bus. Now we create a co-simulation enviroment where parts of the whole simulation can be swapped with new, better or faster algoritms.

2.3. Project

For the project we will create a co-simulation Functional Mockup Unit. This FMU will consist of 2 parts:

1. *C-code*
2. *XML-file*

The C-code is used to represent the data and contains algoritms to solve the used formalism. The XML-document holds all the extra data about every part of the used model. To generate an FMU the C-code is compiled and compressed with the XML-file.

3. Causal Block Diagrams to FMU

We will now look at ways of how to generate an FMU from Causal block diagrams. As [2] suggests, we will use an output trace of our CBD implementation to collect data for our FMU. We will try to go further into the optimization of the CBD before it is compiled into an FMU. Therefore we will first have to flatten the hierarchical CBD to one, non hierachical CBD.

4. The project

The project will consist of multiple steps from a CBD to a FMU:

1. Replace possible integrator and derivative blocks by their hierarchical block. This needs to be done because the integrator and derivative can't be computed directly. We must therefor use an approximation,
2. Flatten the hierarchical CBD making sure the tracebility is assured,
3. Optimize the CBD by constant folding and other optimization techniques,
4. Genereate C-code and XML data,
5. Compile to an FMU.

To investigate our optimization we will use the MathWorks ®/Simulink ®F14 case study. This is a model of the vertical axis movement of an F14 fighter jet moddeled in simulink.

References

[1] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauss, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, et al., The functional mockup interface for tool independent exchange of simulation models, in: Modelica'2011 Conference, March, 2011, pp. 20–22.

[2] B. Pussig, J. Denil, P. De Meulenaere, H. Vangheluwe, Generation of co-simulation compliant functional mock-up units from simulink® , using explicit computational semantics (2013).