INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTATIONSDAG 2016-06-03

EXAMENSARBETE Iterative symbol simplification - extending the Jmodelica.org compiler with a framework for symbolic simplification algorithms
STUDENT Johan Calvén, Zimon Kuhs
HANDLEDARE Jon Sten (Modelon AB), Jonathan Kämpe (Modelon AB), Niklas Fors (LTH)
EXAMINATOR Görel Hedin (LTH)

Extending the *JModelica.org* compiler with a framework for optimization algorithms

POPULÄRVETENSKAPLIG SAMMANFATTNING **Johan Calvén, Zimon Kuhs**

Compiler optimization algorithms serve to increase the efficacy of the resulting programs, but usually operate sequentially without revisiting the code post-change. This work has implemented a framework in the JModelica.org compiler that allows its optimization algorithms to re-investigate modified programs and search for additional improvements.

In order to increase the efficiency of programs many algorithms for improving programs have been developed to be utilized by compilers, e.g. by removing unnecessary methods or unused variables. Usually these algorithms are run sequentially in a set order even though the changes made by one algorithm in a later stage could put the program in a state where there is additional opportunity for improvements by an earlier algorithm. By using a framework for these algorithms, designed with re-visitation in mind, it is possible for the algorithms to take turns modifying the program. This allows algorithms to find even more improvements without running them against the entire program ad infinitum.

We have constructed the basis for such a framework for the JModelica.org Modelica compiler. It provides a structure where the developer of a new algorithm is not required to consider in which way it works in the context of other algorithms, but where it will still benefit from the changes made by them.

Modelica is a programming language used to model physical systems using sets of equations. The sets of equations describe how different parts of the system relate to each other, e.g. the set of equations in figure 1.

$$\begin{cases} \texttt{a = b + c} \\ \texttt{b = c + 1} \\ \texttt{c = 1} \end{cases} \tag{1}$$

The algorithm *variability propagation* (*VP*) looks for variables that are constants (e.g. $c = 1$) and replaces the uses of that variable with the constant. In the figure, it would find the last equation and replace the cs in the other two. This would mean that the second equation would become $b = 1 + 1 = 2$, meaning that *VP* could improve the system further. In the framework, the altered equation will be marked so that the other algorithms, *VP* included, will know that it has changed and it might be possible to perform further changes. *VP* will then find and replace the new constant $b = 2$.