Efficient Recursion to Loop Conversion for OMC Compiler Bootstrapping

Contact: Peter Fritzson (petfr@ida.liu.se, tel: 0708-281484)
or Martin Sjölund (martin.sjolund@liu.se)
PELAB – Programming Environment Lab, Institutionen för Datavetenskap
www.openmodelica.org

At PELAB, together with the Open Source Modelica Consortium (an international open source effort supported by 30 organizations, see www.openmodelica.org) the OpenModelica environment including the OpenModelica Compiler (OMC) of the Modelica language including MetaModelica extensions is developed. The development is open source under the OSMC-PL and GNU V3 licenses, and supported by an Eclipse plug-in MDT (Modelica Development Tooling), also including a debugger, and a template language used for code generators to C and C#. OMC is written in the MetaModelica language with heavy use of recursion.

The goal of this master thesis project is to significantly improve the compilation of recursive calls in MetaModelica to efficient C, by using the so-called tail-recursion optimization that converts recursion into iteration. This makes the code much faster and consuming much less memory. A condition for tail-recursive code is in fig 1 below, the last operation before the function returns must be a call to the function itself. It that cases it is possible for standard C compilers (GNU, Visual Studio) to convert the tail-recursive call into iteration. However, the currently generated code (Fig 2) is not tail-recursive since it contains extra code for exception handling and local intermediate storage. However, with some extra analysis of the source code, it should be possible to optimize away the exception handling and local storage code, thereby enabling the tail recursion optimization. This optimization should be used on a big test case, the 150 000 line OMC compiler, to enable it to compile itself efficiently, i.e., compiler bootstrapping.

The master thesis project requires some knowledge of compiler construction, as well as experience and interest in advanced programming.

1. Tail-recursive C code. Last operation is a function call.

```
#include <stdio.h>

long fac(long i, long acc) {
   if (i>1)
     return fac(i-1,i+acc);
   else
     return acc;
}
```

References:

[1] Peter Fritzson. Principles of Object-Oriented Modeling and Simulation with Modelica 2.1, Wiley-IEEE Press, 2004.
[2] Tail recursion. http://en.wikipedia.org/wiki/Tail_recursion
[3] Pop, Stavåker, Fritzson. Exception Handling for Modelica. Modelica'2008 conf. http://www.modelica.org/events/modelica2008/Proceedings/html/sessions.html
[4] Peter Fritzson. MetaModelica Users Guide.http://www.openmodelica.org/index.php/developer/devdocumentation

2.Currently OMC generated C code from MetaModelica:

```
Recursive_facMetaModelica_rettype
_Recursive_facMetaModelica(modelica_integer i, mode-
lica_integer acc) {
   Recursive_facMetaModelica_rettype tmp1;
   state tmp2;
   modelica_integer out;
   modelica_boolean tmp8;
   tmp2 = get_memory_state();
   {
     modelica_boolean tmp5;
     modelica_boolean tmp7;
     modelica_integer tmp3; /* loop index */
     modelica_integer tmp4; /* switch done? */
     tmp4 = 0;
     for (tmp3=0; 0==tmp4 && tmp3<2;tmp3++) {
       try {
         switch (tmp3) {
           case 0: {
             {
               if (((modelica_integer)i != 0)) {
                 break;
               }
               out = (modelica_integer)acc;
               tmp5 = (1);
             }
             tmp4 = 1;
             break;
           };
           case 1: {
             {
               Recursive_facMetaModelica_rettype tmp6;
               i = (modelica_integer)i;
               tmp6 = _Recursive_facMetaModelica(( (mode-
lica_integer)i - 1), ((modelica_integer)acc * (mode-
lica_integer)i));
```