OpenModelica Compiler Bootstrapping

Martin Sjölund, Linköping University
2011-02-07
3rd OpenModelica Workshop
Linköping, Sweden

Vision

- Build a modular and extensive Modelica compiler
- Compiler functionality resides in Modelica libraries
- Build toolchains using a Modelica editor using the components in the compiler libraries

Functionality in Libraries

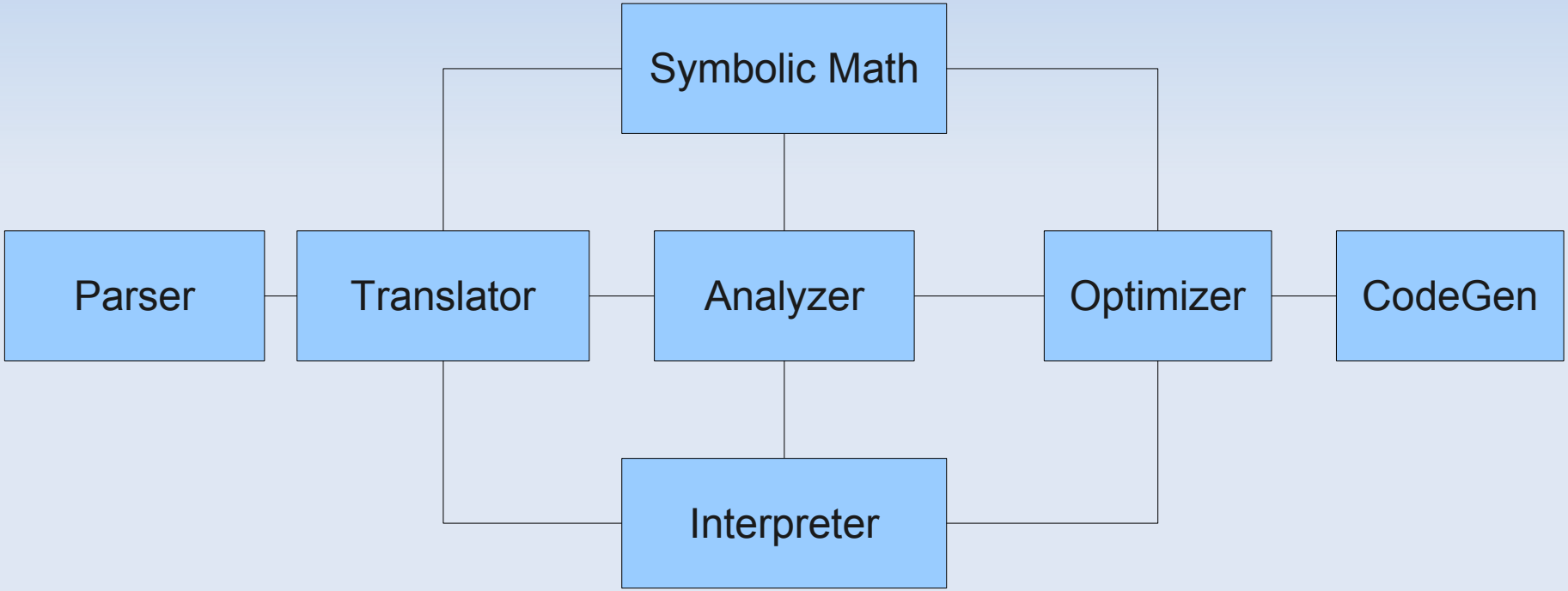
- Modelica has many operators with function syntax but special semantics
  - `initial, pre, sample, delay`
- But many primitive operations are missing
  - `str[n] or stringGet(str,n)`
  - `stringLength(str)`
  - If we had these, the MSL String package could be written in Modelica instead of external C

OpenModelica Script

- A mix of external ”builtin” functions and regular Modelica

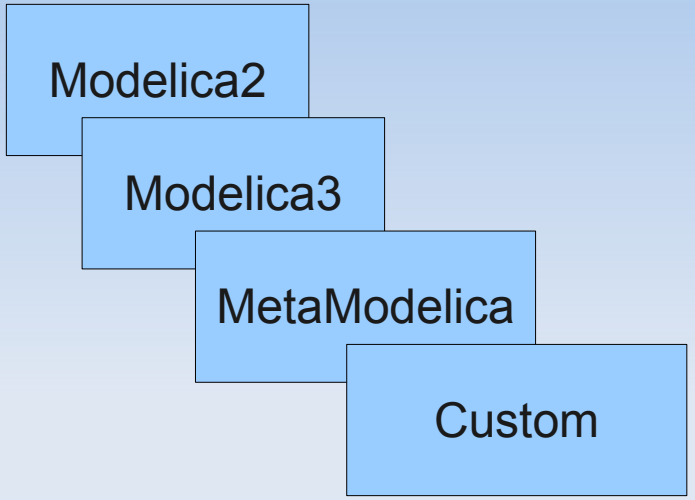
```
function readFileShowLineNumbers
  input String fileName; output String out;
protected String line; Integer num:=1;
algorithm
  out := "";
  for line in strtok(readFile(fileName),"\n") loop
    out := out + String(num) + ": " + line + "\n";
    num := num + 1;
  end for;
end readFileShowLineNumbers;
```

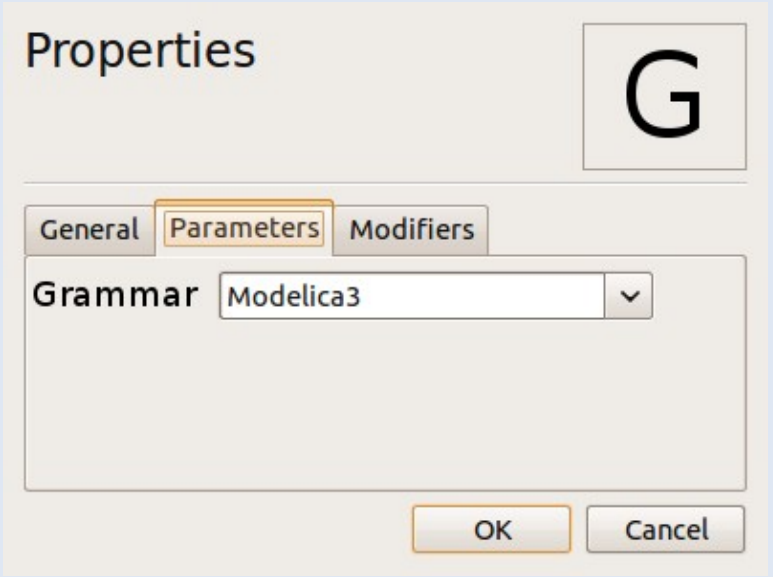
Static Approach to a Modelica Compiler



Modular Approach: Parser

- More choices
- And customizability: Parse your own language into OpenModelica abstract syntax





MetaModelica

- To realize our vision, we need to have the compiler in the same language as Modelica with some extensions
- MetaModelica created 2005
- OpenModelica translated to MetaModelica
- Bootstrapping effort started

Why Bootstrapping?

- MMC, the old MetaModelica Compiler
  - Written in RML+SML
  - Hard to maintain
  - Hard to extend
- OMC, the Modelica+MetaModelica Compiler
  - We get language features for free
  - Easy to extend
  - Easy to port MetaModelica extensions to Modelica
  - Debugging, Profiling, Testing

What's missing in Modelica?

- Implementing a Parser or Symbolic Math Library in Modelica
  - ”Impossible”
  - Modelica only has flat data structures
  - Expressions are recursive data structures

Introducing the Union Type

```
uniontype Expression
  record REAL "A real constant"
    Real r;
  end REAL;
  record ADD "lhs + rhs"
    Expression lhs, rhs;
  end ADD;
  record SUB "lhs - rhs"
    Expression lhs, rhs;
  end SUB;
end Expression;
```

Lists

```
uniontype RealList
  record NIL end NIL;
  record CONS
    Real head;
    RealList tail;
  end CONS;
end RealList;
```

```
RealList myReals =
CONS(1,CONS(2, NIL));
```

```
List<Real> myReals = 1::2::{};
```

- The list is a common data type
- Defining a new uniontype for each kind of list is not desirable
- So we introduce a `List` type

Options, Tuples

- `Option type`
  - `NONE()`
  - `SOME(value)`
- `Tuples`: Anonymous records
  - `(1,1.5,"abc",true)`

Polymorphism

- Reusable functions
  - Boxed data types

```
function listLength
   input List<TypeA> lst;
   output Integer length;
   replaceable type TypeA
subtypeof Any;
external "builtin";
end listLength;
// Works for any list
```

Accessing Data Structures

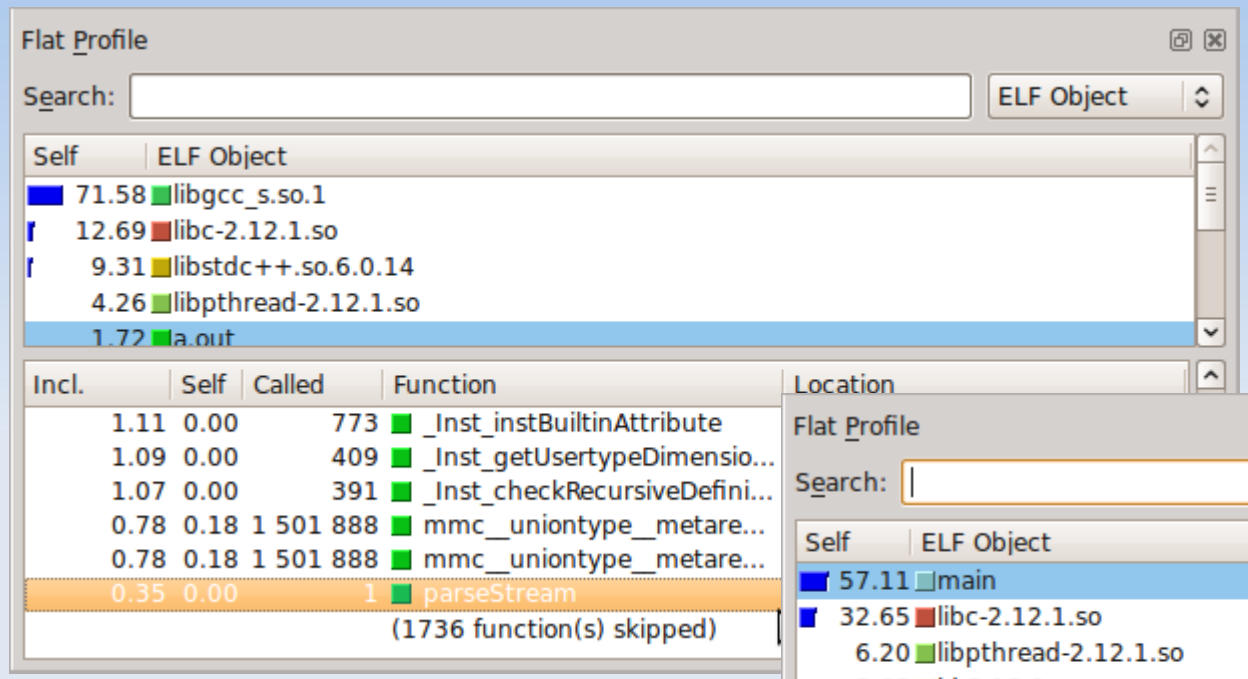
- Accessor functions
  - `j := if not listEmpty(lst) then 2*listGet(lst, 1) else 3;`
- Introducing pattern-matching in Modelica

```
j := match lst
  case (i :: _) then 2*i;
  else 3;
end match;
```

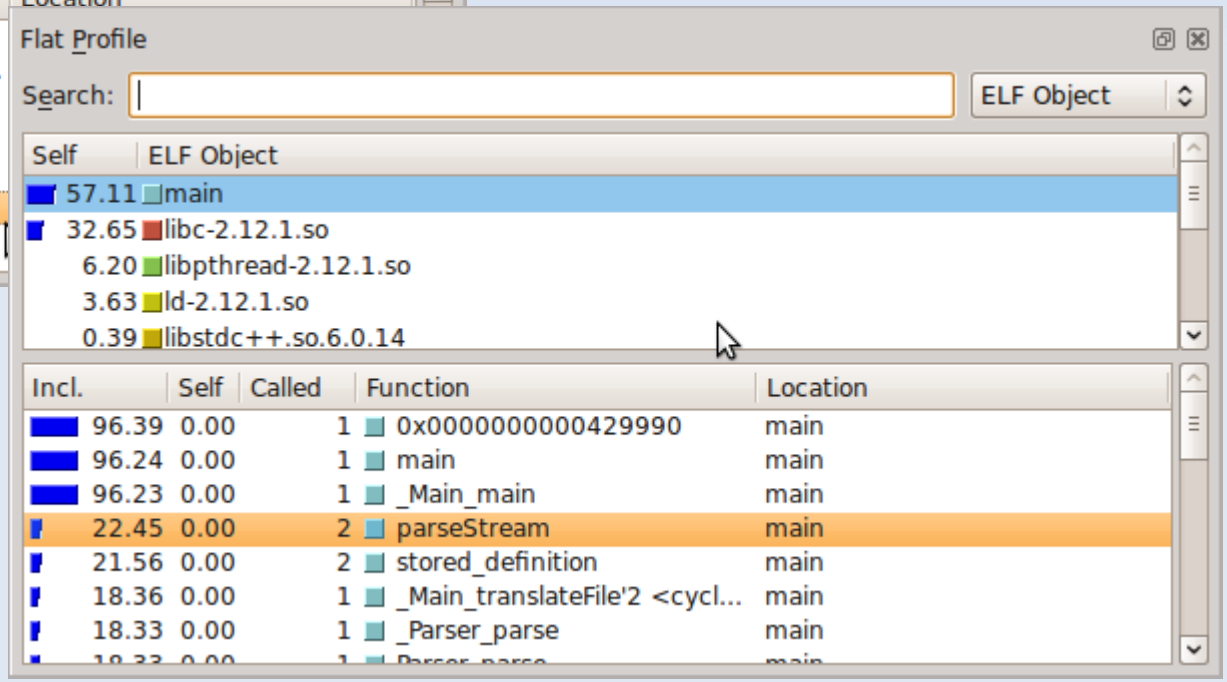
The Bootstrapped Compiler

- First version, Nov-Dec 2010
  - 10-100x slower than MMC
  - Slow compilation speed (hours)
  - Most tests failed due to lack of memory
- Current version, Jan-Feb 2011
  - Speed similar to MMC
  - Faster compilation than MMC
  - Most tests succeed despite lack of garbage collection

PEXPipe.mo before and after optimizations



2010-11-24



2011-02-05

Outlook

- Spring 2011
  - Adding garbage collector
  - Testing the implementation on all platforms (Linux, OMDEV, OSX and Visual Studio)
- Fall 2011
  - Replacing MMC with OMC
  - Rewriting compiler sources using new language extensions
  - Add more optimizations