A Portable Debugger for Algorithmic Modelica Code

Adrian Pop, Peter Fritzson
Programming Environments Laboratory (PELAB)
Department of Computer and Information Science (IDA)
in collaboration with Department of Mechanical Engineering (IKP)
Linköping University (LiU)

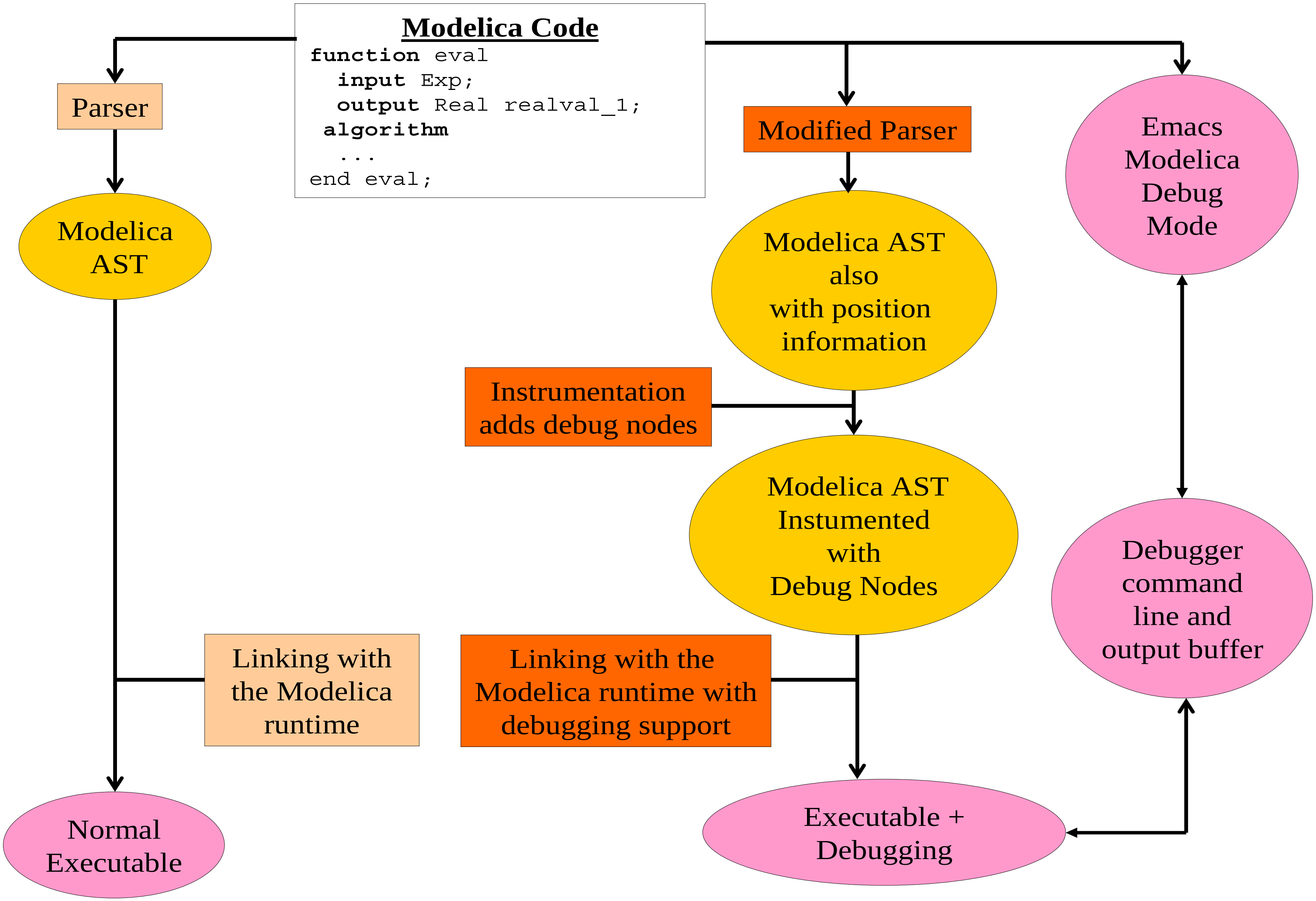
Outline

- Introduction
  - Why there is a need for such a debugger?
- Debugger implementation
  - Overview
  - Source code instrumentation
  - Functionality
- Java Browser for Modelica Data Structures
- Conclusions & Future Work
- Debugger Demo

Introduction

- Why do we need such a debugger?
  - debugging of equation-sections exists
  - debugging of algorithmic code
- Modelica
  - large algorithmic sections in functions
  - scripting
- Modelica+
  - implementation of the OpenModelica compiler
  - 43 packages, 57083 lines of code, 4054 functions, 132 data structures

Debugger Implementation - Overview



Debugger Implementation - Instrumentation

```
function bubbleSort
  input Real [:] unordElem;
  output Real [size(unordElem, 1)] ordElem;
  protected
    Real tempVal;
    Boolean isOver = false;
  algorithm
    ordElem := unordElem;
    while not isOver loop
      isOver := true;
      for i in 1:size(ordElem, 1)-1 loop
       if ordElem[i] > ordElem[i+1]
       then
          tempVal      := ordElem[i];
          ordElem[i]   := ordElem[i+1];
          ordElem[i+1] := tempVal;
          isOver := false;
       end if;
      end for;
    end while;
 end bubbleSort;
```

```
function bubbleSort
  input Real [:] unordElem;
  output Real [size(unordElem, 1)] ordElem;
  protected
    Real tempVal;
    Boolean isOver = false;
  algorithm
    Debug.register_in("unordElem",unordElem);
    Debug.step(...);
    ordElem := unordElem;
    Debug.register_out("ordElem", ordElem);
    Debug.register_in("isOver", isOver);
    Debug.step(...);
    while not isOver loop
     isOver := true;
     Debug.register_out("isOver", isOver);
     Debug.register_in("ordElem",ordElem);
     Debug.step(...);
     for i in 1:size(ordElem, 1)-1 loop
       Debug.register_out("i", i);
       Debug.register_in("i", i);
       Debug.register_in("ordElem[i]",
                           ordElem[i]);
       Debug.register_in("ordElem[i+1]",
                           ordElem[i+1]);
       Debug.step(...);
   ...
end bubbleSort;
```

Debugger Implementation – Functionality (1)

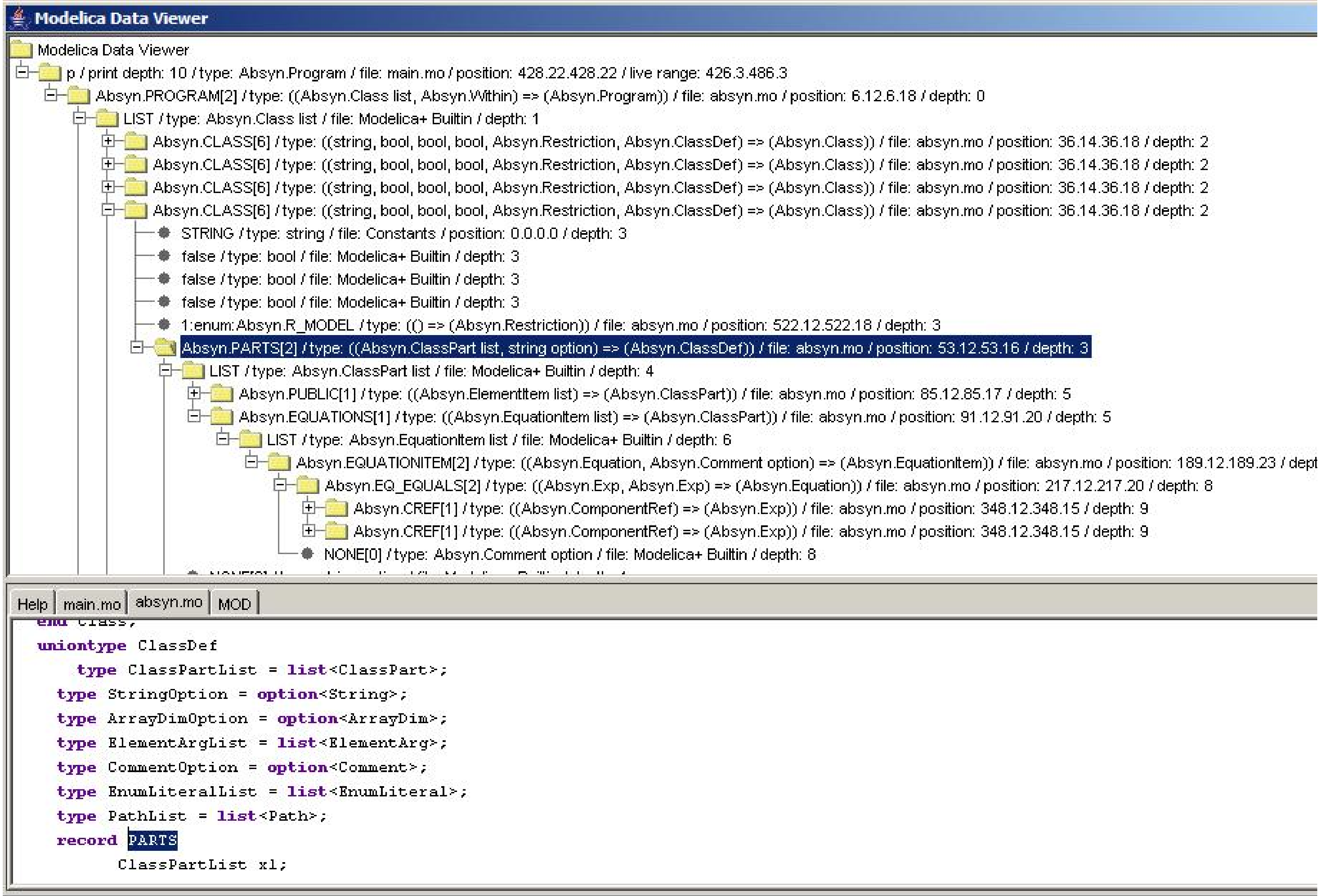
- Breakpoints
  - can be placed on lines or function/package names
  - can be deleted (selectively or all)

- Stepping and Running
  - step mode (step into) – stops before each statement
  - run mode – stops only at breakpoints
  - next mode (step over) – stops at next function call

Debugger Implementation – Functionality (2)

- Examining data
  - printing variables
  - setting the depth of printing
  - sending variables to an external browser
- Additional functionality
  - viewing status information
  - printing backtrace information (stack trace)
  - printing call chain
  - setting debugger defaults
  - getting help

Java Browser for Modelica+ Data Structures



Conclusions and Future Work

- Releasing the debugger together with OpenModelica compiler (implemented in Extended Modelica)
- Supporting all Modelica algorithmic constructs
- Better integration with other Modelica tools
  - model editor
  - equation debugger
  - Eclipse plugin for Modelica

Debugger Demo

- Debugger demo on an expression evaluator

Thank you!

Questions?