**D3.4 (M24)**

**Validation oriented state machine translation (M24 prototype)**

<table>
<thead>
<tr>
<th>Access¹</th>
<th>PU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type²</td>
<td>Prototype</td>
</tr>
<tr>
<td>Version</td>
<td>1.1</td>
</tr>
<tr>
<td>Due Dates³</td>
<td>M24, M36</td>
</tr>
</tbody>
</table>

---

**Open CPS**

*Open Cyber-Physical System Model-Driven Certified Development*

---

**Executive summary⁴:**

This report accompanies the first incremental development prototype of a validation oriented approach for translating Modelica state machines. The prototype is part of the OpenModelica v1.12.0 distribution released on October 31, 2017, which can be downloaded from the OpenModelica website (https://www.openmodelica.org/). This report presents the state of the M24 prototype and briefly describes the remaining work planned for the final M36 prototype.

---

¹ Access classification as per definitions in PCA; PU = Public, CO = Confidential. Access classification per deliverable stated in FPP.
² Deliverable type according to FPP, note that all non-report deliverables must be accompanied by a deliverable report.
³ Due month(s) according to FPP.
⁴ It is mandatory to provide an executive summary for each deliverable.
Deliverable Contributors:

<table>
<thead>
<tr>
<th>Deliverable Leader&lt;sup&gt;6&lt;/sup&gt;</th>
<th>Name</th>
<th>Organisation</th>
<th>Primary role in project</th>
<th>Main Author(s)&lt;sup&gt;5&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bernhard Thiele</td>
<td>LIU</td>
<td>WP3 Leader</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Contributing Author(s)&lt;sup&gt;7&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Internal Reviewer(s)&lt;sup&gt;8&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena Buffoni</td>
</tr>
</tbody>
</table>

Document History:

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Reason for change</th>
<th>Status&lt;sup&gt;9&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>07/11/2017</td>
<td>First Draft</td>
<td>Draft</td>
</tr>
<tr>
<td>0.2</td>
<td>07/11/2017</td>
<td>Internal review version completed</td>
<td>In Review</td>
</tr>
<tr>
<td>0.9</td>
<td>15/11/2017</td>
<td>Integrated comments by first reviewer</td>
<td>In Review</td>
</tr>
<tr>
<td>1.0</td>
<td>16/11/2017</td>
<td>Integrated additional comments</td>
<td>In Review</td>
</tr>
<tr>
<td>1.1</td>
<td>17/11/2017</td>
<td>Changing status to released</td>
<td>Released</td>
</tr>
</tbody>
</table>

<sup>5</sup> Indicate Main Author(s) with an “X” in this column.

<sup>6</sup> Deliverable leader according to FPP, role definition in PCA.

<sup>7</sup> Person(s) from contributing partners for the deliverable, expected contributing partners stated in FPP.

<sup>8</sup> Typically person(s) with appropriate expertise to assess deliverable structure and quality.

<sup>9</sup> Status = “Draft”, “In Review”, “Released”. 
## Contents

1. **Introduction** ......................................................... 4  
2. **Features available in the M24 prototype** .................. 4  
3. **Implementation** .................................................... 5  
4. **Usage** ...................................................................... 6  
   4.1 Creating a New Modelica State Class ............. 6  
   4.2 Making Transitions ............................................. 7  
   4.3 State Machine Simulation ................................. 7  
5. **Industry benchmark problems** ............................... 8  
6. **Features planned for the final prototype at M36** .... 8  
7. **Conclusion** ............................................................. 8  
8. **References** ............................................................. 9
1 Introduction

This report accompanies the first incremental development prototype of the validation oriented Modelica state machines translator. Automata-based programming is an important paradigm in embedded-software development where the program (or a part of it) is modeled as a set of finite state machines. Indeed, high-level control applications typically consists of

1. data-flow parts, realized as block diagrams, and
2. system logic, realized as state machines.

In previous attempts, efforts have been spent on developing library-based solutions for providing a mechanism for graphical Modelica-based modeling of state-machines (e.g., STATEGRAPH and STATEGRAPH2 library [Ott+09]). However, these library-based attempts were not considered to be powerful enough, convenient, and safe to use.

Therefore, Modelica 3.3 [Mod14] introduced dedicated built-in language support for clocked state machines [Elm+12] that was inspired by semantics from Statechart [Har87] and mode automata formalisms [MR03], particularly the mode automata variant implemented in the Lucid Synchrone 3.0 language [Pou06].

The present deliverable is developed in T3.2 with the aim of integrating smoothly with the the data-flow based code-generation as described in T3.1 [TB16; TS16].

2 Features available in the M24 prototype

The M24 prototype supports hierarchic and parallel composition of states, immediate (strong) and delayed (weak) transitions, entering a state with reset or resume of internal state memory (enter by history).

Figure 1 shows a Modelica state machine with hierarchical and parallel composition of states which is inspired by a mode-automaton example described by Maraninchi and Rémond [MR03]. This and other examples are available in OpenModelica’s test suite. The depicted graphical rendering is from the Dymola\textsuperscript{10} tool, since the graphical editor of OpenModelica v1.12.0 does not yet support transparency when rendering hierarchical compositions. State $a$ is composed by the two parallel state machines ($c$-$d$ and $e$-$f$). The initial states ($a$, $c$, $e$) are marked by an attached filled circle. The state machine has three “global” variables ($x$, $y$, $z$) which can be accessed in the inner state instances. For allowing that implicit access it is necessary to use Modelica inner/outer constructs as depicted in the picture.

The state machine in Figure 1 has the two inputs $i$ and $j$, which are instances of the standard Boolean directed data-flow connectors\textsuperscript{11} from the Modelica Standard Library (MSL). Modelica state machines are based on the clocked synchronous language elements extension and are activated whenever the associated clock ticks. If no clock is explicitly specified, a default clock with a periodicity of 1s is selected. Figure 2 shows the result plot from simulating the example model in OpenModelica for 30s while feeding a constant input stream of $i=\text{true}$ and $j=\text{true}$.

\textsuperscript{10}Dassault Systèmes, \url{www.dymola.com}.  
\textsuperscript{11}Hence, instances of class Modelica.Blocks.Interfaces.BooleanInput.
3 Implementation

The approach of translating a state machine is briefly outlined in Figure 3. The state machine constructs are instantiated and transformed into a symbolic intermediate representation which has been described in [TPF15]. This intermediate representation is further elaborated, instance hierarchies are collapsed (flattened) and transformed into clocked synchronous data-flow equations. At this point state machines are in the same representation as clocked data-flow equations allowing the complete reuse of the compiler processing modules for synchronous data-flow equations.
4 Usage

The state machine support is part of the OpenModelica v1.12.0 distribution. Modeling with state machines is possible by writing textual Modelica code, or by using OpenModelica’s graphical editor OMEdit. The following section describe the basic support by the graphical editor. The available graphical support for creating and debugging state machines will be extended in future versions (task T4.1). The following sections are adapted from the OpenModelica User’s Guide.

4.1 Creating a New Modelica State Class

This involves the same steps as creating a new class. Additionally the State checkbox needs to be ticked.
4.2 Making Transitions

In order to make a transition from one state to another the user needs to make sure that the transition mode in the toolbar is enabled (enabled by default, toolbar icon shown below).

Move the mouse over the state. The mouse cursor will change from arrow cursor to cross cursor. To start the transition press left button and move while keeping the button pressed. Now release the left button. Move towards the end state and click when cursor changes to cross cursor.

A Create Transition dialog box will appear which allows you to set the transition attributes. Cancelling the dialog will cancel the transition.

Double click the transition or right click and choose Edit Transition to modify the transition attributes.

4.3 State Machine Simulation

Support for Modelica state machines was added in the Modelica Language Specification v3.3. A subtle problem can occur if Modelica v3.2 libraries are loaded, e.g., the Modelica Standard Library v3.2.2, because in this case the OpenModelica Compiler (OMC) automatically switches into Modelica v3.2 compatibility mode. Trying to simulate a state machine in Modelica v3.2 compatibility mode results in an error. It is possible to use the OMC flag --std=latest in order to enforce (at least) Modelica v3.3 support. In OMEdit this can be achieved by setting that flag in the Tools→Options→Simulation dialog.
5 Industry benchmark problems

RTE has provided first (preliminary) test cases for the state machine translation which are available in the project’s internal subversion repository. The goal is to use state machines for describing mode switches in electrical power systems. At present these test cases fail in OpenModelica v1.12.0 even if they are simplified up to the point where no state machine is used at all. The problem is that OpenModelica fails in solving some of the nonlinear systems of equations resulting from the (physical) electrical power systems model. Solving this issue and supporting the indicated use-case is a task for the final project year.

6 Features planned for the final prototype at M36

For the final prototype the OpenModelica embedded code-generator developed in T3.3 needs to be extended so that it supports state machines. The unified symbolic intermediate representation of state machines and clocked synchronous data-flow as indicated in Section 3 will facilitate this task. Notice that the M24 OpenModelica embedded code-generator prototype (D3.6) is described in report [TS17]. Further, the present prototype needs to be extended and improved so that it provides traceability information, better error diagnostics, and improved debugging capabilities (interconnected with task T4.1 and T4.2).

7 Conclusion

The prototype scheduled for M24 has been developed and is part of the OpenModelica v1.12.0 distribution released on October 31, 2017, which can be downloaded from the OpenModelica website (https://www.openmodelica.org/). Development of the remaining features for the final prototype at M36 as well as other improvements to the M24 prototype will start next year.
References


