Advancements of the OpenModelica Compiler toward a full implementation of event handling
Simulation hybrid systems

Willi Braun, Bernhard Bachmann, Sabrina Pross, Melanie Krems

Department of Applied Mathematics
University of Applied Sciences Bielefeld
33609 Bielefeld, Germany

2010-02-08
Outline

1. Hybrid models
2. Implementation in OpenModelica
3. Examples and results
Hybrid models
Hybrid modeling

Hybrid

- Mixed systems with continuous and discrete components
- Modeling of discontinuous systems is a strength of Modelica
- Simulation needs handling with events and discontinuities

Applications

- Switched electric circuits
- Controlled systems
- PetriNets
Hybrid models
Hybrid modeling

**Hybrid**
- Mixed systems with continuous and discrete components
- Modeling of discontinuous systems is a strength of Modelica
- Simulation needs handling with events and discontinuities

**Applications**
- Switched electric circuits
- Controlled systems
- PetriNets
Hybrid models

Hybrid modeling

**Hybrid**
- Mixed systems with continuous and discrete components
- Modeling of discontinuous systems is a strength of Modelica
- Simulation needs handling with events and discontinuities

**Applications**
- Switched electric circuits
- Controlled systems
- PetriNets
Hybrid models
Petri-Nets

Elements of Petri-Nets
- Fundamental items are places and transitions
- Directed edges connect items

Modifications
- Differential equations describe firing speed for continuous behavior
- Edges may have weightings, threshold and inhibition
- Stochastic delay

⇒ Hybrid models
Hybrid models

Petri-Nets

Elements of Petri-Nets
- Fundamental items are places and transitions
- Directed edges connect items

Modifications
- Differential equations describe firing speed for continuous behavior
- Edges may have weightings, threshold and inhibition
- Stochastic delay

⇒ Hybrid models
Hybrid models
Petri-Nets in OpenModelica

Library in OpenModelica
- Continuous, discrete and stochastic places and transitions can be combined
- Combined PetriNets are versatile applicable
  For example: production or biological processes

Problems in OpenModelica
- “when equations” are not treated synchronously
- Some minor bugs in the use of arrays and functions

Figure: Elements of the PetriNet-Library

Figure: Example network
Hybrid models
Modelling events with Modelica

```model example_if
    input Real u;
    Real y;
    equation
        y = if u > 0 then 1 else -1;
    end example_if;
```

- Conditional expressions like \( u > 0 \) trigger events.
- If events occur, the value is stored twice.
- In this example \( y \) is the right limit and \( \text{pre}(y) \) is the left limit.
Hybrid models
Problems of simulate hybrid models

Why are there problems while integrating of discontinuous systems?

- Numerical integration calls for continuous differential equations
- Since all integrators approximate solutions with polynomials
Hybrid models
Problems of simulate hybrid models

Why are there problems while integrating of discontinuous systems?

- Numerical integration calls for continuous differential equations
- Since all integrators approximate solutions with polynomials

Solution

1. Numerical integration stops at an event
2. Make all discontinues changes
3. Numerical integration starts again
Flatten Modelica model:

\[
0 = F(\dot{x}(t), x(t), y(t), u(t), p, q(t_e), q_{pre}(t_e), c(t_e), t)
\]

\[
\downarrow \text{matching and sorting algorithm transform to}
\]

\[
z = \begin{pmatrix}
\dot{x}(t) \\
y(t) \\
q(t_e)
\end{pmatrix}
= \begin{pmatrix}
f(x(t), u(t), p, q_{pre}(t_e), c(t_e), t) \\
g(x(t), u(t), p, q_{pre}(t_e), c(t_e), t) \\
h(x(t), u(t), p, q_{pre}(t_e), c(t_e), t)
\end{pmatrix}
\]
Hybrid models
Synchronous equations

model when_example
  Real x(start=0.1), y;
  discrete Real a(start=1);
  equation
    y = sin(time*2);
    der(x) = a*y;
    when {y<-0.5, x>0.2} then
      a = x-pre(a);
  end when;
end when_example;
Hybrid models
Synchronous equations

**model** when_example
    Real x(start=0.1), y;
    discrete Real a(start=1);
**equation**
    y = sin(time*2);
    der(x) = a*y;
    when {y<−0.5,x>0.2} then
        a = x−pre(a);
    end when;
end when_example;

**Incidence-Matrix**

\[
\begin{pmatrix}
1 & 0 & 0 \\
1 & 1 & 0 \\
1 & 1 & 1
\end{pmatrix}
\]

\[
y = \sin(time)
\]
\[
a = x - \text{pre}(a)
\]
\[
der(x) = a \times y
\]
Hybrid models

Synchronous equations

```model when_example
  Real x(start=0.1), y;
  discrete Real a(start=1);
  equation
    y = sin(time*2);
    der(x) = a*y;
    when {y<-0.5, x>0.2} then
      a = x-pre(a);
    end when;
end when_example;
```

**Incidence-Matrix**

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>a</th>
<th>der(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>der(x)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Sorting equations**

Sorting is based on all equations in due to the correct order of evaluation at all time points.
**Implementation**  
**Hybrid Modelica DAE-System**

Flatten Modelica model:

\[
0 = F(\dot{x}(t), x(t), y(t), u(t), p, q(t_e), q_{pre}(t_e), c(t_e), t)
\]

\[
\downarrow \text{matching and sorting algorithm transform to}
\]

\[
z = \begin{pmatrix}
\dot{x}(t) \\
y(t) \\
q(t_e)
\end{pmatrix} = \begin{pmatrix}
f(x(t), u(t), p, q_{pre}(t_e), c(t_e), t) \\
g(x(t), u(t), p, q_{pre}(t_e), c(t_e), t) \\
h(x(t), u(t), p, q_{pre}(t_e), c(t_e), t)
\end{pmatrix}
\]

We get four blocks for code generation:

- **continuous** \( f \) and \( g \) \rightarrow derivative states and algebraic variables

- **discrete** \( h \) \rightarrow discrete algebraic variables

- **all** \( z \) \rightarrow all blocks together, in the right order

An additional block to manage the conditions for events:

\[
c(t) \rightarrow \text{Zero Crossing functions}
\]
Implementation
Approach to simulate hybrid models
Implementation
Approach to simulate hybrid models

Initial Step
- Initial-value problem is solved by a simplex-method
- Initial Zero-crossing functions and check for initial events
Implementation
Continuous integration

Integration step
- Integration method: $x_{i+1} = \Phi(x_i)$
- Calculate for the next step $t_{i+1}$ the new state vector $x(t_{i+1})$
- Evaluate continuous blocks $f$ and $g$

$$\dot{x}(t_{i+1}) = f(x(t_{i+1}), q(t_i), q_{pre}(t_i), c(t_{i+1}), t)$$
$$y(t_{i+1}) = g(x(t_{i+1}), q(t_i), q_{pre}(t_i), c(t_{i+1}), t)$$
**Implementation**

**Check for event conditions**

- Find consistent initial values
- Check for initial events
- Try a continues integration step
- **Check for event condition in current interval**

**Check for zero-crossing**

- Conditions are converted into zero-crossing functions
- \( x < 2 \) changes from false to true when \( x - 2 \) crosses zero
- If any zero-crossing becomes true an event is fired
Implementation

Find event time

Root-finding method

- Find root in interval \([t_i; t_{i+1}]\) as event time \(t_e\)
- Bisection is a very simple and robust method, but it is also relatively slow
- All methods approximate the root by setting limits on each side of \(t_e\)
- Additional we have \(t_e - \epsilon\) and \(t_e + \epsilon\)
**Hybrid models Implementation in OpenModelica**

**Examples and results**

**Implementation**

Handle event

1. Determine states at $t_e - \epsilon$ with interpolation
2. Determine continuous blocks by using functions $f$ and $g$
3. Save all variables as values for $\text{pre}()$ and emit them to result file

\[
\dot{x}(t_e - \epsilon) = f(x(t_e - \epsilon), q(t_i), \text{pre}(t_i), c(t_e - \epsilon), t)
\]

\[
y(t_e - \epsilon) = g(x(t_e - \epsilon), q(t_i), \text{pre}(t_i), c(t_e - \epsilon), t)
\]
Implementation

Handle event

1. Determine states at $t_e + \epsilon$ with interpolation
2. Evaluate all blocks by using function $\dot{z}$
3. Check for changes of discrete variables
4. Event Iteration

\[
\begin{align*}
\dot{x}(t_e + \epsilon) &= f(x(t_e + \epsilon), q_{pre}(t_e), c(t_e), t) \\
y(t_e + \epsilon) &= g(x(t_e + \epsilon), q_{pre}(t_e), c(t_e), t) \\
q(t_e + \epsilon) &= h(x(t_e + \epsilon), q_{pre}(t_e), c(t_e), t)
\end{align*}
\]
model EventIteration
  Real x(start=1), dx;
  discrete Real a(start=1);
  Boolean y(start=false);
  Boolean z(start=false);
  Boolean h1, h2;

equation
  der(x) = dx;
  dx = a * x;
  h1 = x >= 2; h2 = dx >= 4;
  when h1 then
    y = true;
  end when;
  when y then
    a = 2.0;
  end when;
  when h2 then
    z = true;
  end when;
end EventIteration;
Examples
Petri-Net Example
Summary

- With this approach we can manage many synchronously appearing events.
- We are on the way to an optimal formulation of the PetriNet library in OpenModelica.

- Work that remains to be done
  - Integrate further integrations methods and root-finding methods.
  - Adjust the code generation.