Synchronous Events in the OpenModelica Compiler with a Petri Net Library Application
Simulation hybrid systems in Modelica

Willi Braun, Bernhard Bachmann and Sabrina Pross

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

2010-10-03
Outline

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

Hybrid modeling

Hybrid

- Mixed systems with continuous and discrete components
- Simulation needs handling with events and discontinuities

Applications

- Switched electric circuits
- Controlled systems
- Petri Nets
Hybrid models

Hybrid Petri Net

Elements of a Petri Net

- Fundamental items are places and transitions
- Directed arcs connect items

P1

T1

P3

T3

P5

T2

P4

P2

1

2

5

1

1

2

3
Hybrid models

Hybrid Petri Net

Elements of a Petri Net

- Fundamental items are places and transitions
- Directed arcs connect items

Directed arcs connect items.
Hybrid models
Hybrid Petri Net

Elements of a Petri Net
- Fundamental items are places and transitions
- Directed arcs connect items

Modifications
- The fireing speed can be described by differential equations for continuous behaviour
- Transitions may have a stochastic delay
- Edges may have weightings, threshold and inhibition
Hybrid models

Hybrid Petri Net

Elements of a Petri Net
- Fundamental items are places and transitions
- Directed arcs connect items

Modifications
- The firing speed can be described by differential equations for continuous behaviour
- Transitions may have a stochastic delay
- Edges may have weightings, threshold and inhibition

⇒ Hybrid models
Hybrid models

Petri Nets in OpenModelica

Library in OpenModelica

- Continuous, discrete and stochastic places and transitions can be combined
- Combined Petri Nets are versatile applicable
  For example: production or biological processes

Problems in OpenModelica

- “when equations” were not treated synchronously

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 occurs the value is stored twice.
- In this example $y$ is the right limit and $\text{pre}(y)$ is the left limit.
Hybrid models
Hybrid Modelica DAE-System

Flatten Modelica model:

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

↓ matching and sorting algorithm transform to

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

//known Variable: u
when y2 > 2 then
  y3 = f1(y4);
end when;
y4 = f2(y5);
when y1 > 0 then
  y5 = f3(u);
y2 = f4(y4);
end when;
y1 = f5(u);

Incidence-Matrix

\[
\begin{pmatrix}
0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 \\
\end{pmatrix}
\]
Hybrid models
Synchronous Data-flow principle

Sorting of when equations

```modelica
// known Variable: u
y1 = f5(u);
when y1 > 0 then
    y5 = f2(u);
end when;
y4 = f3(y5);
when y1 > 0 then
    y2 = f4(y4);
end when;
when y2 > 2 then
    y3 = f1(y4);
end when;
```

Incidence-Matrix

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 \\
\end{bmatrix}
\]
Hybrid models
Synchronous Data-flow principle

//known Variable: u
y1 = f5(u);
when y1 > 0 then
  y5 = f2(u);
end when;
y4 = f3(y5);
when y1 > 0 then
  y2 = f4(y4);
end when;
when y2 > 2 then
  y3 = f1(y4);
end when;

Incidence-Matrix

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

Sorting equations

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

Flatten Modelica model:

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

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

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

Flatten Modelica model:

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

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

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

We get four blocks:

- **continuous** \( f \) and \( g \) \( \rightarrow \) derivative states and algebraic variables
- **discrete** \( h \) \( \rightarrow \) discrete algebraic variables
- **all** \( z \) \( \rightarrow \) all three block together

An additional block to manage the conditions for events:

\( c(t) \rightarrow \text{Zero Crossing functions} \)
Implementation in OpenModelica

Approach to simulate hybrid models

1. Start
2. Find consistent initial values
3. Check for initial events
4. Try a continues integration step
5. Check for event condition in current interval
6. Any events?
   - No: Accept step and emit values
     - No: End time?
       - Yes: End
       - No: New events?
         - Yes: Fire event and change values for event
11. Find event time in current interval
12. Calculate values before event and emit them
13. Fire event and change values for event
14. End
Implementation in OpenModelica

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
Hybrid models in Modelica Implementation in OpenModelica
Examples and results
Summary

Implementation in OpenModelica
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 in OpenModelica

Check for event conditions

- 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 in OpenModelica

Find event time

Root-finding method

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

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

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

$$y(t_e - \epsilon) = g(x(t_e - \epsilon), q(t_i), q_{\text{pre}}(t_i), c(t_e - \epsilon), t)$$
Implementation in OpenModelica

Handle event

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

$$\dot{x}(t_e + \epsilon) = f(x(t_e + \epsilon), q_{pre}(t_e), c(t_e), t)$$
$$y(t_e + \epsilon) = \frac{g(x(t_e + \epsilon), q_{pre}(t_e), c(t_e), t)}{h(x(t_e + \epsilon), q_{pre}(t_e), c(t_e), t)}$$
Examples

Example for correct sorting of when-equations

```model when_sorting
  Real x;
  Real y;
  Boolean w(start=true);
  Boolean v(start=true);
  Boolean z(start=true);

  equation
    when sample(0,1) then
      x = pre(x) + 1;
      y = pre(y) + 1;
    end when;
    when y > 2 and pre(z) then
      w = false;
    end when;
    when x > 2 then
      z = false;
    end when;
    when y > 2 and z then
      v = false;
    end when;
  end when_sorting;
```

Examples
Example for correct sorting of when-equations

```model when_sorting
  Real x;
  Real y;
  Boolean w(start=true);
  Boolean v(start=true);
  Boolean z(start=true);
  equation
    when sample(0,1) then
      x = pre(x) + 1;
      y = pre(y) + 1;
    end when;
    when y > 2 and pre(z) then
      w = false;
    end when;
    when x > 2 then
      z = false;
    end when;
    when y > 2 and z then
      v = false;
    end when;
  end when_sorting;
```

Incidence-Matrix

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>x</th>
<th>w</th>
<th>z</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>w</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>z</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>v</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Examples

Example for correct sorting of when-equations

```model when_sorting
  Real x;
  Real y;
  Boolean w(start=true);
  Boolean v(start=true);
  Boolean z(start=true);
  equation
    when sample(0,1) then
      x = pre(x) + 1;
      y = pre(y) + 1;
    end when;
    when y > 2 and pre(z) then
      w = false;
    end when;
    when x > 2 then
      z = false;
    end when;
    when y > 2 and z then
      v = false;
    end when;
  end when_sorting;
```

![Graph showing the behavior of x, y, w, v, and z over time](graph.png)
Examples

Petri Net example for a production process
Examples
Petri Net example for a production process
Examples

Petri Net example for a production process
Summary

- With this algorithmic approach we can simulate many synchronously appearing events.
- The Petri Net Library for OpenModelica has been optimized, so that now events can be specified with the aid of “when-equation”.

- The performance of the current implementation can be further improved:
  - by handling time events separately with a suitable step-size control.
  - by achieving more advanced root finding methods.