BHAVE prototype

Draft of Version 1.0

Tomas Krilavičius

2006

http://fmt.cs.utwente.nl/tools/behave

University of Twente
Faculty of Electrical Engineering, Mathematics and Computer Science
Formal Methods and Tools Group
P.O. Box 217, 7500AE, Enschede, The Netherlands
http://fmt.cs.utwente.nl

Typeset by LaTeX.
Cover: no cover

Copyright © 2006 by T. Krilavičius, Enschede, The Netherlands.

Contents

1 What is Bhave prototype

1.1 Introduction

Bhave prototype is a prototype of *Behavioural Hybrid Process Calculus* [Brinksma and Krilavičius, 2005] simulation tool. BHPC is a process calculus for modelling and analysis of hybrid systems. It combines the behavioural approach for dynamical systems [Polderman and Willems, 1998] and classical process algebraic techniques [Hoare, 1978, Milner, 1980, Bolognesi and Brinksma, 1987, Baeten and Weijland, 1990].

The tool was developed to test BHPC simulation techniques [Krilavičius, 2006]. It is not intended for critical case studies, but to be used for further language and algorithms developments. It supports a restricted set of BHPC operators. Moreover, it should be used with BHPCC (BHPC Parser [van Putten, 2006] which, provided an BHPC specification, translates it to an internal format [van Putten, 2006]. The tool does not provide any visualisation facilities, however tools like Microsoft Excel can be employed to generate various plots form the simulation results.

1.2 Related tools

There are several tools that are related to Bhave prototype.

- *BHPCC* translates BHPC specification to an interal format. It is being developed as a part of master project van Putten [2006].
- Discrete Bhave is a tool for discrete simulation of BHPC. It is being developed as a part of master project Schonenberg [2006].

2 User manual

2.1 Input language

Input language for BHAVE prototype is a restricted BHPC transformed to an internal format (Section 2.2). The tool accepts a subset of Behavioural Hybrid Process Calculus operators:

- Stop process ($\mathbf{0}$).
- Action-prefix ($\mathsf{a} \,.\, B$).
- Symbolic trajectory-prefix ($[t_1, \ldots, t_m \mid \Phi \downarrow \mathcal{P}red \Downarrow \mathcal{P}red_{\text{exit}}] \,.\, B$). However, it is restricted to a certain class of continuous behaviours and conditions (limitations are mostly imposed by the specifics of MAPLE 9.5[1] that is used as an ODE solver)
  - Trajectories (Φ) are represented by ordinary differential equations (ODE) that are solvable by MAPLE. Moreover, on the right side of the differential equations declarations all variables should be decorated with (t) (current restriction of BHPCC and BHAVE prototype).
  - Conditions ($\mathcal{P}red$) are always true.
  - Exit conditions ($\mathcal{P}red_{\text{exit}}$) are represented by simple inequalities of type $x >\!\geqslant\!<\!\leqslant\!= e$, where x is a qualifier and c is an expression solvable by MAPLE. Moreover, more complex exit conditions can be given in form $\bigvee_i \text{cond}(x_i)$ where x_i is a particular qualifier. In other words, exit conditions can be given as a disjunction over the exit conditions on the different variables.
- Hiding ($\mathsf{new}\ w \,.\, B$) is ignored. However, user is informed about it, if it is encountered.
- Renaming ($B\,[\sigma]$) is ignored. However, user is informed about it, if it is encountered.
- Superposition ($\bigoplus_{i \in I} B_i$) is treated as an ordinary choice ($\sum_{i \in I} B_i$).
- Parallel composition ($B \parallel_A^H B$). Parallel simulation of trajectory-prefixes is limited in a following way:
  - All differential equations are collected from participating processes.
  - All exit conditions are collected, grouped by qualifiers and simplified. E.g., if we have trajectory-prefixes with exit conditions $\{h = 0\} \vee \{v > -10, v < 5\}$ and $\{h = 5\} \vee \{v > -5, v < 10\}$, correspondingly, then we get $\{v > -5, v < 5\}$.

[1] `http://www.maplesoft.com/`

- Initial values assignments are collected.
- Constructed system of differential equations with initial values and stop conditions is passed to Maple and simulated until any of exit conditions is satisfied or maximal simulation time is reached.
- All trajectory-prefixes stopped and simulation continues with successive processes, i.e., concatenation is not supported.

- Recursion (P), where re-initialisation parameters are given by simple expressions solvable by Maple.

2.1.1 ASCII language

ASCII language will be added later (from van Putten [2006]).

2.2 Internal format

As an input, Bhave prototype should be supplied correct BHPC specification in an *internal format*. Informally, the internal format is a XML based representation of the BHPC language (Section 2.1).

Specification of the internal format will be added later (from van Putten [2006]).

2.3 Output format

Output format is a tabulation separated text file. It writes simulation output in columns with first column representing time, the following columns representing qualifiers values and the last one for the actions. We illustrate it on output for the bouncing ball Example 2.5.1.

```
time      h         v         actions
0         12.       20.
0.05      12.988    19.510
0.1       13.951    19.020
0.15      14.890    18.530
0.2       15.804    18.040
0.25      16.694    17.550
0.3       17.559    17.060
          ...
4.25      8.494     −21.650
4.3       7.399     −22.140
4.35      6.280     −22.630
4.4       5.136     −23.120
4.45      3.968     −23.610
4.5       2.775     −24.100
4.55      1.558     −24.590
4.6       .316      −25.080
4.61257   0.316     −25.08    bounce
4.61257   .316      17.556
          ...
8.11257   1.737     −16.744
8.16257   .888      −17.234
8.21257   .136e−1   −17.724
8.21334   0.0136    −17.724   bounce
8.21334   .136e−1   12.407
          ...
13.6707   .725      −5.191
13.7207   .453      −5.681
13.7707   .157      −6.171
13.7957   0.157     −6.171    stopped
```

Remark 2.3.1. Software can be easily altered to produce output in a different format. However, in current version it is not controlled via command line, therefore changes in `SimulationOutput` class and re-compilation are necessary. □

2.4 Command line options

Command line options:

- `-cout Output to stdout is on.`
- `-i= Input file name.`
- `-o= Output file name (default = bhave_ouput.out).`
- `-showtime Show simulation time.`
- `-step= Simulation step size (default=0.05).`
- `-t= Simulation time (default=40).`
- `-? Show this help.`

2.5 Examples

In this section we present several examples to show Bhave prototype capabilities.

Example 2.5.1 (Bouncing ball). A bouncing ball is a simple example of hybrid systems. It is a simplified model of an elastic ball that is bouncing and losing a fraction of its energy with every bounce. The altitude of the ball is h, v is a vertical speed, and c is a coefficient for the lost energy. The ball moves according to the flow conditions

$$\dot{h} = v \qquad \dot{v} = -g$$

and at the bounce time the velocity is reassigned to $-cv$.

In BHPC it can be defined in the following way:

$$\begin{aligned} \mathrm{BB}(h_0, v_0) &\triangleq [h, v \mid \Phi(h_0, v_0) \Downarrow h = 0] \,.\, \mathrm{BB}(h, -c * v) \\ \Phi(h_0, v_0) &= \{h, v : (0, t] \to \mathbb{R} \mid \\ &\qquad h(0) = h_0, v(0) = v_0, \dot{h} = v, \dot{v} = -g, h \geqslant 0\} \end{aligned}$$

Symbolic trajectory-prefix $[h, v \mid \Phi(h_0, v_0) \Downarrow h = 0]$ defines the dynamics of the ball until the bounce, and then the process continues recursively calling itself with updated signals $\mathrm{BB}(0, -c * v)$.

The simulation results are depicted in Figure 2.1. Velocity is depicted by the dashed line and altitude is depicted by the solid line.

Specification in the internal format is presented in Listing 2.1. We will go over and explain the main parts of the specification.

- Line 5 starts initial process definition. Initial process can be used only to call a process with initial values.
  - Line 6 defines the name of process to start.
  - Line 7 provides an initial value for the first parameter.
- Line 10 starts a process definition.
  - Line 11 provides an identifier of the process.
  - Line 12 gives a parameter of the process.

```
<bhpc>
    <version>1.1 beta</version>
    <qualifier>h</qualifier>
    <qualifier>v</qualifier>
    <initialprocess>
        <processname>BB</processname>
        <parameter>12</parameter>
        <parameter>20</parameter>
    </initialprocess>
    <processdefinition>
        <processname>BB</processname>
        <parameter>h</parameter>
        <parameter>v</parameter>
        <trajectoryprefix>
            <qualifier>h</qualifier>
            <qualifier>v</qualifier>
            <signal>
                <name>phi</name>
                <parameter>h</parameter>
                <parameter>v</parameter>
            </signal>
      <conditions>h &gt;= 0</conditions>
      <exitconditions>h = 0</exitconditions>
    </trajectoryprefix>
    <actionprefix>bounce</actionprefix>
    <processcall>
      <name>BB</name>
      <parameter>h</parameter>
      <parameter>-0.7 * v</parameter>
        </processcall>
  </processdefinition>
    <trajectorydefinition>
        <name>phi</name>
        <parameter>h</parameter>
        <parameter>v</parameter>
        <qualifier>h</qualifier>
        <qualifier>v</qualifier>
        <assignment>
            <qualifier>h</qualifier><time>0</time><value>h</value>
        </assignment>
        <assignment>
            <qualifier>v</qualifier><time>0</time><value>v</value>
        </assignment>
        <differentialequation>
            <qualifier>h</qualifier><value>v(t)</value>
        </differentialequation>
        <differentialequation>
            <qualifier>v</qualifier><value>-9.8</value>
        </differentialequation>
        <expression>h >= 0</expression>
  </trajectorydefinition>
</bhpc>
```

Listing 2.1: Bouncing ball in the internal format

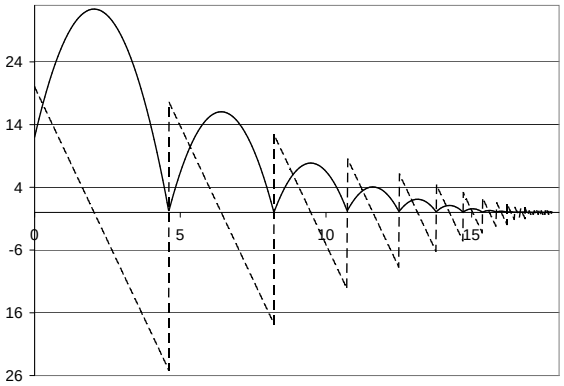

Figure 2.1: Altitude and velocity of the ball

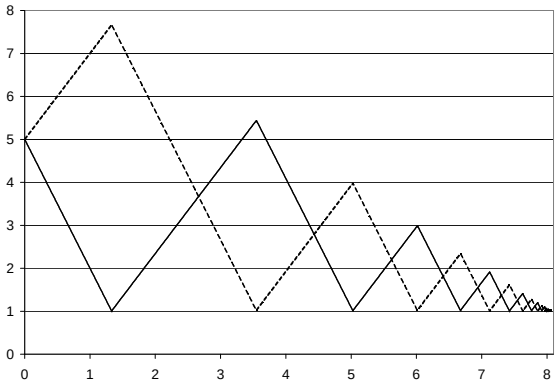

Figure 2.2: Fluid level changes of two tanks

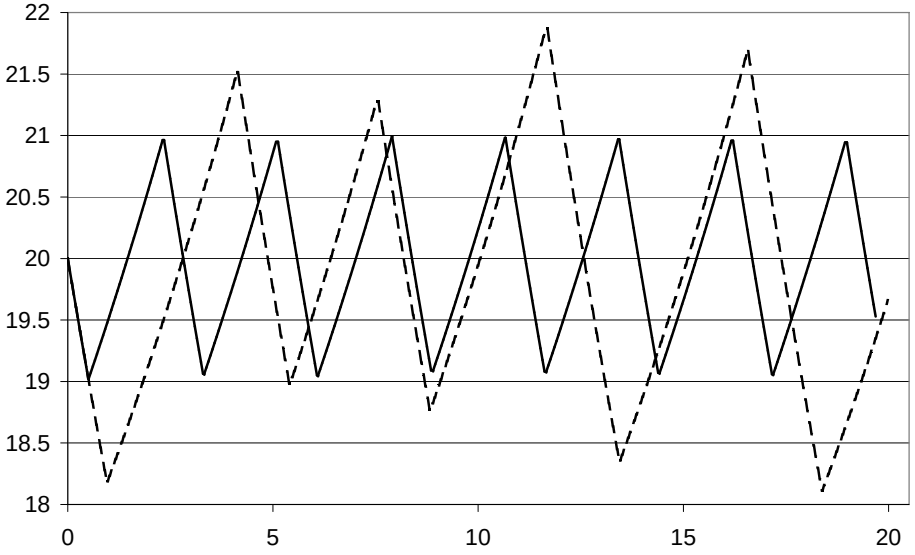

Figure 2.3: Temperature changes for simple and upgraded thermostat

- Line 14 starts a definition of trajectory-prefix that is the first operator in this process.
  - Line 15 defines a qualifier.
  - Line 17 starts a definition of signal, i.e, a reference to the definition of trajectory-prefix dynamics.
  - Line 18 provides a signal name.
  - Line 19 provides the first signal parameter.
  - Line 22 give conditions for the trajectory prefix.
  - Line 23 give exit conditions for the trajectory prefix.
- Line 25 defines a trajectory-prefix, the second operator in this process definition.
- Line 26 defines a recursive call, the third operator in this process declaration.
  - Line 27 gives the new process name.
  - Line 28 gives the new value.
- Line 32 starts a trajectory definition.
  - Line 33 provides a name of the trajectory definition.
  - Line 34 provides a parameter of the trajectory definition.

- Line 38 provides an assignment of initial values.
- Line 44 declares a differential equation defining an evolution of h.
- Line 50 declares a condition (not used in the current version)

□

Example 2.5.2 (Simple and controlled thermostat). A thermostat is one of the best-known introductory examples of hybrid systems. The room temperature is controlled by a thermostat, which continuously senses the temperature and switches a heater on and off. The temperature changes are defined by the ordinary differential equations. When the heater is off, the temperature decreases according to the exponential function $l(t) = \theta e^{Kt}$, where t is time, l is the temperature in the room, θ is the initial temperature, and K is a constant determined by the room. When the heater is on, the temperature increases according to the function $l(t) = \theta e^{-Kt} + h(1 - e^{-Kt})$, where h is a constant that depends on the power of the heater. The temperature should be maintained between $temp_{\min}$ and $temp_{\max}$. Temperatures $temp_{\text{on}}$ and $temp_{\text{off}}$ are the minimal and maximal thresholds, when the heater can be turned on and off, respectively.

The model of thermostat in BHPC:

$$\begin{aligned}
\text{ThOff}(l_0) &\triangleq \Big[l \;\Big|\; \Phi_{\text{Off}}(l_0) \Downarrow \mathit{tempOn} \geqslant l \geqslant \mathit{tempMin} \Big] \,.\, \mathsf{on} \,.\, \text{ThOn}(l) \\
\text{ThOn}(l_0) &\triangleq \Big[l \;\Big|\; \Phi_{\text{On}}(l_0) \Downarrow \mathit{tempOff} \leqslant l \leqslant \mathit{tempMax} \Big] \,.\, \mathsf{off} \,.\, \text{ThOff}(l) \\
\Phi_{\text{Off}}(l_0) &= \{ l : (0,t] \to \mathbb{R} \mid l(0) = l_0, \dot{l} = -Kl \} \\
\Phi_{\text{On}}(l_0) &= \{ l : (0,t] \to \mathbb{R} \mid l(0) = l_0, \dot{l} = K(h - l) \}
\end{aligned}$$

It consists of two process.

- In process ThOff the heater is off and the symbolic trajectory-prefix defines the temperature fall. When the temperature reaches the interval $[\mathit{tempOn}, \mathit{tempMin}]$, the process can perform action on and switch to the process ThOn.
- Process ThOn analogously defines the period of heating.

However, it is possible to upgrade such thermostat without changing the specification itself. Let us add a controller that observes temperature and forces the thermostat to switch on and off at exactly 19 and 21, correspondingly:

$$\begin{aligned}
\text{Control}(l) \triangleq \Big[l \;\Big|\; \text{any}(l) \Downarrow l = 19 \Big] \,.\, \mathsf{on}. \\
\Big[l \;\Big|\; \text{any}(l) \Downarrow l = 21 \Big] \,.\, \mathsf{off} \,.\, \text{Control}(l) \\
\text{UpgradedThermostat}(l) \triangleq \text{ThOff}(l) \,\|^{l}_{\mathsf{on},\mathsf{off}}\, \text{Control}(l)
\end{aligned}$$

where any(l) is a special function that models an observer, i.e., it accepts any behaviour for l. It works only in parallel composition. Technically it just adds exit conditions to the parallel composition of trajectory-prefixes.

Corresponding specifications in XML format are presented in Listings 2.2 and 2.3.

Let $\mathit{tempOff} = 21$, $\mathit{tempMax} = 22$, $\mathit{tempOn} = 19$ and $\mathit{tempMin} = 18$. Then the results of simulation of the simple and controlled thermostat are depicted in Figure 2.3. Dashed line depicts evolution of the simple thermostat and solid line depicts evolution of the coupled version. □

Example 2.5.3 (Two tanks). Consider the two tanks model [Lygeros and Sastry, 1999, Schutter and Heemels, 2004], where both tanks have one common fluid source that provides fluid at the rate of l_{in} units per second. Through a pipe, the fluid source can be directed either to the left tank or to the right tank. Both tanks have openings at the bottom, and from the tanks water drains at the rates of d_{left} and d_{right} units per second, respectively. Initially, the tanks contain ll_{init} and lr_{init} units of fluid, respectively. The pipe can switch between the tanks instantaneously. The objective is to keep the fluid volumes in the interval $(l_{\min}, l_{\max})$.

```
<processdefinition>
    <processname>ThOn</processname>
    <parameter>l</parameter>
    <trajectoryprefix>
        <qualifier>l</qualifier>
        <signal>
            <name>PhiThOn</name> <parameter>l</parameter>
        </signal>
        <exitconditions>l &gt;= 21</exitconditions>
        <exitconditions>l &lt;= 22</exitconditions>
    </trajectoryprefix>
    <actionprefix>thermOff</actionprefix>
    <processcall>
        <name>ThOff</name> <parameter>l</parameter>
    </processcall>
</processdefinition>

<processdefinition>
    <processname>ThOff</processname>
    <parameter>l</parameter>
    <trajectoryprefix>
        <qualifier>l</qualifier>
        <signal>
            <name>PhiThOff</name> <parameter>l</parameter>
        </signal>
        <exitconditions>l &gt;= 18</exitconditions>
        <exitconditions>l &lt;= 19</exitconditions>
    </trajectoryprefix>
    <actionprefix>thermOn</actionprefix>
    <processcall>
        <name>ThOn</name> <parameter>l</parameter>
    </processcall>
</processdefinition>

<trajectorydefinition>
    <name>PhiThOn</name>
    <parameter>l</parameter>
    <qualifier>l</qualifier>
    <assignment>
        <qualifier>l</qualifier><time>0</time><value>l</value>
    </assignment>
    <differentialequation>
        <qualifier>l</qualifier><value>-0.1*(9.2-l(t))</value>
    </differentialequation>
</trajectorydefinition>

<trajectorydefinition>
    <name>PhiThOff</name>
    <parameter>l</parameter>
    <qualifier>l</qualifier>
    <assignment>
        <qualifier>l</qualifier><time>0</time><value>l</value>
    </assignment>
    <differentialequation>
        <qualifier>l</qualifier><value>-0.1*l(t)</value>
    </differentialequation>
</trajectorydefinition>
```

Listing 2.2: Simple Thermostat

```
<processdefinition>
    <processname>Thermostat</processname>
    <parameter>l</parameter>
    <parallelcomposition>
        <case>
            <processcall>
                <name>ThOff</name>
                <parameter>l</parameter>
            </processcall>
        </case>
        <qualifier>l , thermOn , thermOff</qualifier>%

        <case>
            <processcall>
                <name>ThC</name>
                <parameter>l</parameter>
            </processcall>
        </case>
    </parallelcomposition>
</processdefinition>

<processdefinition>
    <processname>ThC</processname>
    <parameter>l</parameter>
    <trajectoryprefix>
        <qualifier>l</qualifier>
        <signal>
            <name>PhiThC</name>
            <parameter>l</parameter>
        </signal>
        <exitconditions>l = 19</exitconditions>
    </trajectoryprefix>
    <actionprefix>thermOn</actionprefix>
    <trajectoryprefix>
        <qualifier>l</qualifier>
        <signal>
            <name>PhiThC</name>
            <parameter>l</parameter>
        </signal>
        <exitconditions>l = 21</exitconditions>
    </trajectoryprefix>
    <actionprefix>thermOff</actionprefix>
    <processcall>
        <name>ThC</name>
        <parameter>l</parameter>
    </processcall>
</processdefinition>

<trajectorydefinition>
    <name>PhiThC</name>
    <parameter>l</parameter>
    <qualifier>l</qualifier>
</trajectorydefinition>
```

Listing 2.3: Thermostat controller

```
<initialprocess>
    <processname>TwoTanks</processname>
    <parameter>5</parameter>
    <parameter>5</parameter>
</initialprocess>

<processdefinition>
    <processname>TwoTanks</processname>
  <parameter>l1</parameter>
  <parameter>l2</parameter>
  <trajectoryprefix>
    <qualifier>l1</qualifier><qualifier>l2</qualifier>
    <signal>
        <name>Filling1</name>
        <parameter>l1</parameter>
        <parameter>l2</parameter>
        </signal>
    <exitconditions>l1 = 10</exitconditions>
    <exitconditions>l2 = 1</exitconditions>
   </trajectoryprefix>

   <actionprefix>FillTwo</actionprefix>

   <trajectoryprefix>
    <qualifier>l1</qualifier>
    <qualifier>l2</qualifier>
        <signal>
            <name>Filling2</name>
            <parameter>l1</parameter>
            <parameter>l2</parameter>
        </signal>
        <exitconditions>l1 = 1</exitconditions>
     <exitconditions>l2 = 10</exitconditions>
   </trajectoryprefix>

   <actionprefix>FillOne</actionprefix>

   <processcall>
    <name>TwoVessels</name>
    <parameter>l1</parameter>
    <parameter>l2</parameter>
   </processcall>

</processdefinition>
```

Listing 2.4: Process TwoTanks

```
<trajectorydefinition>
    <name>Filling1</name>
    <parameter>l1</parameter>
    <parameter>l2</parameter>
    <qualifier>l1</qualifier>
    <qualifier>l2</qualifier>
    <assignment>
        <qualifier>l1</qualifier>
        <time>0</time>
        <value>l1</value>
    </assignment>
    <assignment>
        <qualifier>l2</qualifier>
        <time>0</time>
        <value>l2</value>
    </assignment>
    <differentialequation>
        <qualifier>l1</qualifier>
        <value>2</value>
    </differentialequation>
    <differentialequation>
        <qualifier>l2</qualifier>
        <value>-3</value>
    </differentialequation>
</trajectorydefinition>

<trajectorydefinition>
    <name>Filling2</name>
    <parameter>l1</parameter>
    <parameter>l2</parameter>
    <qualifier>l1</qualifier>
    <qualifier>l2</qualifier>
    <assignment>
        <qualifier>l1</qualifier>
        <time>0</time>
        <value>l1</value>
    </assignment>
    <assignment>
        <qualifier>l2</qualifier>
        <time>0</time>
        <value>l2</value>
    </assignment>
    <differentialequation>
        <qualifier>l1</qualifier>
        <value>-3</value>
    </differentialequation>
    <differentialequation>
        <qualifier>l2</qualifier>
        <value>2</value>
    </differentialequation>
</trajectorydefinition>
```

Listing 2.5: TwoTanks Trajectories

```
<processdefinition>
    <processname>TankLOff</processname>
        <parameter>l1</parameter>
        <trajectoryprefix>
            <qualifier>l1</qualifier>
            <signal>
                <name>LLOff</name>
                <parameter>l1</parameter>
            </signal>
        </trajectoryprefix>
        <actionprefix>onLL</actionprefix>
        <processcall>
            <name>TankLOn</name> <parameter>l1</parameter>
        </processcall>
</processdefinition>

<trajectorydefinition>
    <name>LLOff</name>
    <parameter>l1</parameter>
    <qualifier>l1</qualifier>
    <assignment>
        <qualifier>l1</qualifier> <time>0</time> <value>l1</value>
    </assignment>
    <differentialequation>
        <qualifier>l1</qualifier> <value>-3</value>
    </differentialequation>
</trajectorydefinition>

<processdefinition>
    <processname>TankLOn</processname>
    <parameter>l1</parameter>
    <trajectoryprefix>
        <qualifier>l1</qualifier>
        <signal>
            <name>LLOn</name>
            <parameter>l1</parameter>
        </signal>
    </trajectoryprefix>
    <actionprefix>offLL</actionprefix>
    <processcall>
        <name>TankLOff</name> <parameter>l1</parameter>
    </processcall>
</processdefinition>

<trajectorydefinition>
    <name>LLOn</name>
    <parameter>l1</parameter>
    <qualifier>l1</qualifier>
    <assignment>
        <qualifier>l1</qualifier> <time>0</time> <value>l1</value>
    </assignment>
    <differentialequation>
        <qualifier>l1</qualifier> <value>2</value>
    </differentialequation>
</trajectorydefinition>
```

Listing 2.6: Modular TwoTanks: Right Tank

```
<processdefinition>
    <processname>TankROff</processname>
        <parameter>lr</parameter>
        <trajectoryprefix>
            <qualifier>lr</qualifier>
            <signal>
                <name>LROff</name>
                <parameter>lr</parameter>
            </signal>
        </trajectoryprefix>
        <actionprefix>onLR</actionprefix>
        <processcalr>
            <name>TankROn</name> <parameter>lr</parameter>
        </processcalr>
</processdefinition>

<trajectorydefinition>
    <name>LROff</name>
    <parameter>lr</parameter>
    <qualifier>lr</qualifier>
    <assignment>
        <qualifier>lr</qualifier> <time>0</time> <value>lr</value>
    </assignment>
    <differentialequation>
        <qualifier>lr</qualifier> <value>-3</value>
    </differentialequation>
</trajectorydefinition>

<processdefinition>
    <processname>TankROn</processname>
    <parameter>lr</parameter>
    <trajectoryprefix>
        <qualifier>lr</qualifier>
        <signal>
            <name>LROn</name>
            <parameter>lr</parameter>
        </signal>
    </trajectoryprefix>
    <actionprefix>offLR</actionprefix>
    <processcalr>
        <name>TankROff</name> <parameter>lr</parameter>
    </processcalr>
</processdefinition>

<trajectorydefinition>
    <name>LROn</name>
    <parameter>lr</parameter>
    <qualifier>lr</qualifier>
    <assignment>
        <qualifier>lr</qualifier> <time>0</time> <value>lr</value>
    </assignment>
    <differentialequation>
        <qualifier>lr</qualifier> <value>2</value>
    </differentialequation>
</trajectorydefinition>
```

Listing 2.7: Modular TwoTanks: Left Tank

```
<processdefinition>
    <processname>Controller</processname>
    <parameter>ll</parameter> <parameter>lr</parameter>
    <trajectoryprefix>
        <qualifier>ll</qualifier> <qualifier>lr</qualifier>
        <signal>
            <name>ControlLIn</name>
            <parameter>ll</parameter> <parameter>lr</parameter>
        </signal>
        <exitconditions>ll = 10</exitconditions>
        <exitconditions>lr = 1</exitconditions>
    </trajectoryprefix>
    <actionprefix>offLL</actionprefix>
    <actionprefix>onLR</actionprefix>
    <trajectoryprefix>
        <qualifier>ll</qualifier> <qualifier>lr</qualifier>
        <signal>
            <name>ControlRIn</name>
            <parameter>ll</parameter> <parameter>lr</parameter>
        </signal>
        <exitconditions>ll = 1</exitconditions>
        <exitconditions>lr = 10</exitconditions>
    </trajectoryprefix>
    <actionprefix>offLR</actionprefix>
    <actionprefix>onLL</actionprefix>
    <processcall>
        <name>Controller</name>
        <parameter>ll</parameter> <parameter>lr</parameter>
    </processcall>
</processdefinition>

<trajectorydefinition>
    <name>ControlLIn</name>
    <parameter>ll</parameter> <parameter>lr</parameter>
    <qualifier>ll</qualifier> <qualifier>lr</qualifier>
    <assignment>
        <qualifier>ll</qualifier> <time>0</time> <value>ll</value>
    </assignment>
    <assignment>
        <qualifier>lr</qualifier> <time>0</time> <value>lr</value>
    </assignment>
</trajectorydefinition>

<trajectorydefinition>
    <name>ControlRIn</name>
    <parameter>ll</parameter> <parameter>lr</parameter>
    <qualifier>ll</qualifier> <qualifier>lr</qualifier>
    <assignment>
        <qualifier>ll</qualifier> <time>0</time> <value>ll</value>
    </assignment>
    <assignment>
        <qualifier>lr</qualifier> <time>0</time> <value>lr</value>
    </assignment>
</trajectorydefinition>
```

Listing 2.8: Modular TwoTanks: Controller

```
<initialprocess>
    <processname>TwoVessels</processname>
    <parameter>5</parameter> <parameter>5</parameter>
</initialprocess>

<processdefinition>
    <processname>TwoVessels</processname>
    <parameter>ll</parameter> <parameter>lr</parameter>

    <parallelcomposition>
        <case>
            <processcall>
                <name>Controller</name>
                <parameter>ll</parameter> <parameter>lr</parameter>
            </processcall>
        </case>
        <qualifier>offLL, onLL, offLR, onLR, ll, lr</qualifier>
        <case>
            <parallelcomposition>
                <case>
                    <processcall>
                        <name>TankLOn</name> <parameter>ll</parameter>
                    </processcall>
                </case>
                <qualifier></qualifier>
                <case>
                    <processcall>
                        <name>TankROff</name> <parameter>lr</parameter>
                    </processcall>
                </case>
            </parallelcomposition>
        </case>
    </parallelcomposition>
</processdefinition>
```

Listing 2.9: Modular TwoTanks: System

Let l_{left} and l_{right} are volumes in the left and right tanks, correspondingly.

$$
\begin{aligned}
&\text{TwoTanks}(l_{\text{left}}, l_{\text{right}}) \triangleq \\
&\quad \left[l_{\text{left}}, l_{\text{right}} \;\middle|\; \dot{l}_{\text{left}} = d_{\text{left}} + l_{\text{in}}, \dot{l}_{\text{right}} = d_{\text{right}} \Downarrow l_{\text{left}} = l_{\max} \vee l_{\text{right}} = l_{\min}\right]. \\
&\quad \mathsf{FillRight}. \\
&\quad \left[l_{\text{left}}, l_{\text{right}} \;\middle|\; \dot{l}_{\text{left}} = d_{\text{left}}, \dot{l}_{\text{right}} = d_{\text{right}} + l_{\text{in}} \Downarrow l_{\text{left}} = l_{\min} \vee l_{\text{right}} = l_{\max}\right]. \\
&\quad \mathsf{FillLeft}. \\
&\quad \text{TwoTanks}(l_{\text{left}}, l_{\text{right}})
\end{aligned}
$$

Corresponding specifications in XML format are presented in Listings 2.4 and 2.5.

Let $l_{\text{in}} = 5, d_{\text{left}} = d_{\text{right}} = -3, l_{\min} = 1$ and $l_{\max} = 10$. Then simulation results are depicted in Figure 2.2.

This system is of interest, because by ignoring physical reality one can devise a controller that keeps both water levels within the required bounds: whenever l_{left} falls to $l_{\min}$, direct the pipe to the left tank, and whenever l_{right} falls to $l_{\min}$, direct the pipe to the right tank (or, corresponding switching at the $l_{\max}$). However, such a controller cannot be realised physically, because it would cause the pipe to switch back and forth infinitely often within a finite amount of time, if $d_{\text{left}} + d_{\text{right}} \neq l_{\text{in}}$.

To demonstrate compositional modelling advantages we propose a slightly different version of the same system. In this specification l is water level in the tank, d_{out} is a drain rate and l_{in} is an inflow rate.

$$
\begin{aligned}
\text{TankIn}(l, d_{\text{out}}, l_{\text{in}}) &\triangleq \left[l \;\middle|\; \dot{l} = d_{\text{out}} \Downarrow \texttt{true}\right]. \mathsf{off}. \text{TankOut}(l, d_{\text{out}}, l_{\text{in}}) \\
\text{TankOut}(l, d_{\text{out}}, l_{\text{in}}) &\triangleq \left[l \;\middle|\; \dot{l} = d_{\text{out}} + l_{\text{in}} \Downarrow \texttt{true}\right]. \mathsf{off}. \text{TankIn}(l, d_{\text{out}}, l_{\text{in}}) \\
\text{Controller}(l_{\text{left}}, l_{\text{right}}) &\triangleq \left[l_{\text{left}}, l_{\text{right}} \;\middle|\; \text{any}(t) \Downarrow l_{\text{left}} = l_{\max} \vee l_{\text{right}} = l_{\min}\right]. \mathsf{fill}_{\text{right}}. \\
&\qquad \left[l_{\text{left}}, l_{\text{right}} \;\middle|\; \text{any}(t) \Downarrow l_{\text{left}} = l_{\min} \vee l_{\text{right}} = l_{\max}\right]. \mathsf{fill}_{\text{left}}. \\
&\qquad \text{Controller}(l_{\text{left}}, l_{\text{right}})
\end{aligned}
$$

$$
\begin{aligned}
&\text{System}(l_{\text{left}}, l_{\text{right}}, dl_{\text{out}}, dr_{\text{out}}, ll_{\text{in}}, lr_{\text{in}}) \triangleq \\
&\quad \Big(\text{TankOn}(l_{\text{left}}, dl_{\text{out}}, ll_{\text{in}}) \left[l_{\text{left}}/l, dl_{\text{out}}/d_{\text{out}}, ll_{\text{in}}/l_{\text{in}}, \mathsf{fill}_{\text{left}}/\mathsf{on}, \mathsf{fill}_{\text{right}}/\mathsf{off}\right] \parallel \\
&\quad \text{TankOn}(l_{\text{right}}, dr_{\text{out}}, lr_{\text{in}}) \left[l_{\text{right}}/l, dr_{\text{out}}/d_{\text{out}}, lr_{\text{in}}/l_{\text{in}}, \mathsf{fill}_{\text{right}}/\mathsf{on}, \mathsf{fill}_{\text{left}}/\mathsf{off}\right]\Big) \\
&\quad \parallel^{l_{\text{left}}, l_{\text{right}}}_{\mathsf{fill}_{\text{left}}, \mathsf{fill}_{\text{right}}} \text{Controller}(l_{\text{left}}, l_{\text{right}})
\end{aligned}
$$

- The left and right tanks are presented in Listings 2.7 and 2.6, respectively. Unfortunatelly, renaming is not implemented yet, therefore it was done by hands.
- Controller is presented in Listing 2.8.
- Top level specification is presented in Listing 2.9.

□

3 Source documentation

3.1 Technical details

We give some technical implementation details.

Software Language C++ (standard).

Compiler Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.xx.xxxx.

Additional libraries Xerces-C++ 2.7.0 (a validating XML parser written in a portable subset of C++); maple.h (Open Maple).

Code statistics ~ 9klines (~ 5klines of pure code).

Documentation Doxygen 1.4.5.

Operating systems Windows XP (however C++ code and Xerces-C should be portable to other operating systems, Open Maple portability was not investigated).

Solver MAPLE 9.5 is used as differential equations, expressions, inequalities solver.

Input format BHPC in the internal format [van Putten, 2006] provided by BHPCC (BHPC compiler) [van Putten, 2006].

Output format text file, with tabulation symbol separated columns containing time, values and actions.

Visualisation plots were produced using Microsoft (R) Excel 2003 XY(Scatter) routine of Chart Wizard.

3.2 Build

Building instructions:

- Apache XERCES-C++ library is should be available and XERCESCROOT environment variable set.
- MAPLE should be available and MAPLE environment variable set.
- Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.xx.xxxx is required. It was not tested with other versions or compilers.
- Microsoft Visual C++ v6.0 and Code::Blocks v1.0 projects and workspaces are provided and can be used to build the tool.

3.3 Architecture specifics

- References to qualifiers set are carried as qid attributes in XML. It is assigned either to case or processexpression tags.
- Process and Engine classes operate on processexpression XML documents that are incomplete process XML documents.

3.4 Detailed documentation

Detailed documentation is provided in HTML generated by Doxygen.

3.5 Bugs

- Symbolic trajectory-prefix $\left[l \;\middle|\; l(0) = c, \dot{l} = f(l) \Downarrow l = \right]$ would not work, because Maple stops solving ODE as soon as l becomes equal to c. In BHPC semantics $l(0$ is not a part of the trajectory-prefix (because left-open right-closed intervals are adopted). E.g., it occurs in the bouncing ball example (Example 2.5.1), if the process is defined as follows

  $$\mathrm{BB}(h_0, v_0) \triangleq [h, v \mid \Phi(h_0, v_0) \Downarrow h = 0] \,.\, \mathrm{BB}(0, -c * v)$$

  i.e., in the recursive call the altitute is set to 0.

  One of potential solutions is to make a first small step without detecting exit conditions.

Moreover, some items (or non-existence of such) discussed in Section 4.1 can be considered as a bugs.

4 Future plans

4.1 Short term future plans

In this section we discuss our short term future plans. It does not mean that all items will be implemented soon, but just that these items are important and not very complicated to add.

- Concatentation support. Currently all parallel trajectory-prefix finish evolution at the same time. That is quite unconvenient. However implementation of concatenation is quite involved technically:
  - To support concatention of cyclic trajectories additional information is necessary, e.g., values of the first derivates. Some problems were encountered feeding such to information to ODE solving facilities of Maple.
  - Currently exit conditions of all parallel trajectory-prefixes are put together. A procedure to check exit conditions for every process should be implemented.
- Conditions over whole evolutions of trajectory-prefix are not supported in the current version. The main problem is feeding such information do ODE solving facilities of Maple.
- Setting precision for different qualifiers. The first step could be setting the same precision for all qualifiers.
- Setting precision for differential equations solver.
- Batch mode simulation.
- Constatns support.
- Optimisation of trajectory-prefix simulation. In the simulation of trajectory-prefix communication with Maple is very inefficient, because only simple data types are used to extract values. Hovewer it should be possible to access more complex Maple data types and in such a way improve efficiency.

4.2 Long term future plans

It this section we disscuss long term plans for BHPC hybrid simulation tool. Mostly these are the ideas how to improve the tool.

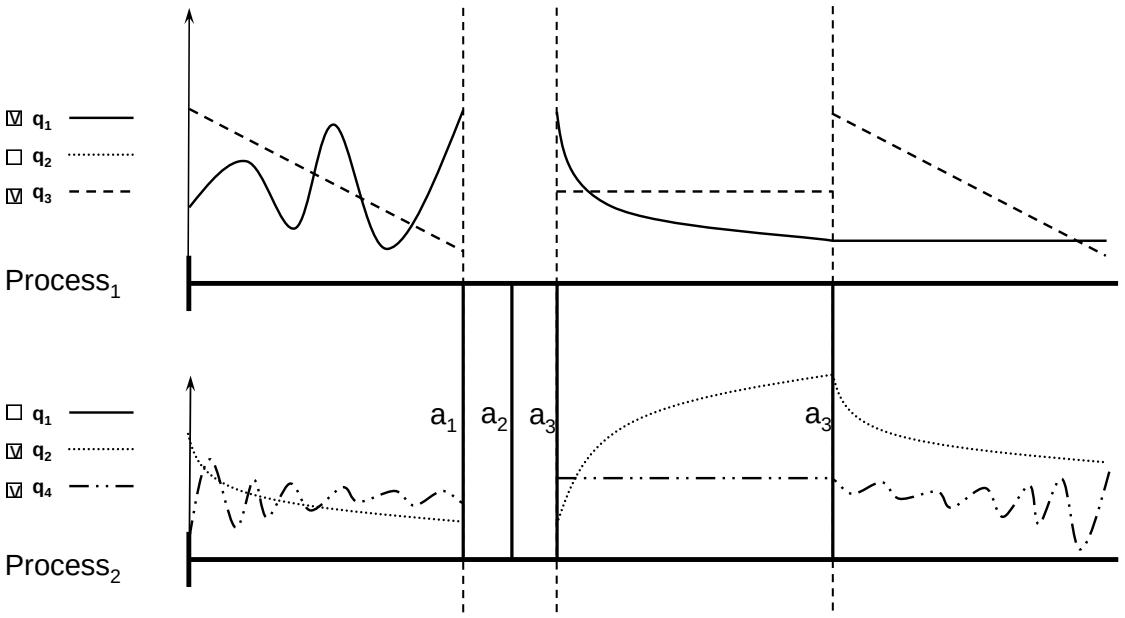

Figure 4.1: Visualisation of hybrid simulation

- Visualisation of results. Currently the only output is text-only (Section 2.3). In future it would be nice to visualise ouput. We propose to use MSCplots or MSP (*message sequence plots*) [Krilavičius and Schonenberg, 2005, Schonenberg, 2006, Krilavičius, 2006]. We illustrate it in Figure 4.1. However, an implementation of visualisation unit would require a lot of effort. It would require changes in the existing software, because it is necessary to carry additional information about action-prefixes, i.e., synchronising process. We anticipate that it should be done using techniques similar to relating qualifiers values and subprocesses, i.e., by carrying pointers to the corresponding information as attribute in XML.
- Visualisation of model. Currently specifications are not visualised at all. Some visualisation techniques are discussed in *Visualisation of models* section in Krilavičius [2006]. We anticipate that *object diagrams* can be suitable way to visualise BHPC models. Object diagrams represent object oriented approach to modelling of dynamic systems. In contrast to block diagrams, physical objects are represented by mnemonically shaped icons. Objects can be interconnected, consist or be integral parts of other objects. Considerable advantages of this approach are *encapsulation* and *inheritance*. Informally, encapsulation provides means to hide internal object details from the outside world. Inheritance allows the specific objects to inherit properties of generic ones. Object diagrams are gaining a lot of popularity lately, and are used in tools, e.g., Modelica.
- Hiding support.
- Renaming support.

Bibliography

J. C. M. Baeten and W. P. Weijland. *Process algebra*. Cambridge University Press, New York, NY, USA, 1990. ISBN 0-521-40043-0.

T. Bolognesi and H. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks*, 14:25–59, 1987.

E. Brinksma and T. Krilavičius. Behavioural hybrid process calculus. Technical Report TR-CTIT-05.45, CTIT, University of Twente, 2005. URL `http://www.cs.utwente.nl/~krilaviciust/publications/BHPC_techrep.pdf`.

C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, 1978. ISSN 0001-0782.

T. Krilavičius. *Hybrid Systems: Where Computer Science and Control Theory Meet*. PhD thesis, University of Twente, 2006. Draft.

T. Krilavičius and H. Schonenberg. Discrete simulation of behavioural hybrid process calculus. In P. M. E. Bra and J. J. van Wijk, editors, *IFM2005 Doctoral Symposium on Integrated Formal Methods*, pages 33–38, Eindhoven, Netherlands, November 2005. Technical University of Eindhoven, Department of Mathematics and Computer Science.

J. Lygeros and S. Sastry. Hybrid systems: Modeling, analysis & control, ee291, 1999. URL `http://paleale.eecs.berkeley.edu/~lygeros/Teaching/ee291E.html`.

R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer, 1980.

J.W. Polderman and J. C. Willems. *Introduction to Mathematical Systems Theory: a behavioral approach*. Springer, 1998.

M. H. Schonenberg. Discrete simulation of behavioural hybrid process algebra. Technical report, University of Twente, 2006. Draft of master thesis.

B. De Schutter and W.P.M.H. Heemels. *Modeling and Control of Hybrid Systems*. DISC, September 2004. Lecture Notes of the DISC Course.

A.E. van Putten. Behavioural hybrid process calculus parser and translator to modelica. Technical report, University of Twente, 2006. Draft of master thesis.

Index