CLASSICAL AND STATECHART – BASED MODELING OF STATE EVENTS AND OF STRUCTURAL CHANGES IN THE MODELICA – SIMULATOR MOSILAB

Günther Zauner[1, 2], Florian Judex[2], Peter Schwarz[3]

[1]"die Drahtwarenhandlung" Simulation Services, Neustiftgasse 57-59, 1070 Wien
[2]Vienna University of Technology, Institute for Analysis and Scientific Computing, 1040 Vienna, Wiedner Hauptstraße 8-10, Austria
[3]Fraunhofer Institute for Integrated Circuits, Dresden, Germany

Guenther.zauner@drahtwarenhandlung.at

Abstract

Mosilab (*MO*delling and *SI*mulation *LAB*oratory) a new simulation system developed by Fraunhofer understands Modelica, offers different modeling approaches, and supports structural dynamic systems. This will be discussed on the basis of a main example, the classical constrained pendulum. We show how the solution can be done using only standard Modelica components, where the benefits are and which kind of switching the states can be done. As we will see there is no possibility to define separate submodels with different state space dimensions and switch between these systems during one simulation run.

The next point of view lies on an extension of the Modelica framework. The most important new feature of this model description language is the definition of a statechart framework. With this construction the next three solutions of the constrained pendulum are done. The first approach is mathematically similar to the Modelica solution and defines poor parameter events within the statechart construct. This approach cannot handle events of higher order. The second approach for the model is done with two different submodels, one for the case that the rope of the pendulum is short and one for the case it is long. In the statechart the two models are then connected and disconnected to the main program and thereby switched between active and off. A third approaches with only one submodel but two instances of the system will conclude our model inspection.

We focus on how the numerical approaches are done in general and where are the benefits comparing to the other solutions. A final step is to look at the numerical quality of the output of the different approaches. This is done by validation with another example for which an analytical solution exists.

Keywords: Mosilab, state event, statechart, Modelica, constrained pendulum.

Presenting Author's biography

Günther Zauner. He has earned a degree in mathematics with specialization in "mathematical computer science". With an interdisciplinary background based on higher technical school he has experience in the application and the development of numerical methods and different modeling approaches. Current work focuses on simulation and modeling techniques and coupling of different approaches.

1 General

In the last decade a broad amount of knowledge in model description theory and modeling and simulation techniques, which could not be solved with the older systems, have their renaissance. Increasing power of computers and better algorithms lead to advanced modeling environments. One benefit are the customer friendly interfaces.

Nevertheless, these advanced modeling environments ask for well educated experts in the field of simulation. In nowadays definition of a project it is very often important to model a part of a system in detail, but when the system switches to another state the description is done imprecisely. Another often needed approach is that a state event makes restrictions to the actual model which leads to a change in the degree of freedom. Both here explained cases result in a change in the state space dimension or even a parameter change for the given system.

The new generation of simulation systems handles this challenge with different methods. One approach is to define a discrete class, where state event handling is done (e.g. ACSL), others restrict their system. They allow only parameter changing state events (e.g. Dymola/Modelica) or to blow up the whole system. A third class, which we will focus on in a selected example, is simulators with an implemented state machine. This group of simulators can handle state events of both sorts: the classical ones where the dimension of the state space remains the same and the hybrid switch between separate models.

2 The simulation environment

MOSILAB (*MO*deling and *SI*mulation *LAB*oratory) [1] is a simulator developed by Fraunhofer-Instituts FIRST, IIS/EAS, ISE, IBP, IWU and IPK within the research project GENSIM [2]. It is a generic simulation tool for modeling and simulation of complex multidisciplinary technical systems. The simulation environment supports the procedures modeling, simulation and post processing. The model description in MOSILAB is done in the Modelica [3] standard. Additional features are implemented to assure high flexibility during modeling the concept of structural dynamics. This is done by extending the Modelica standard with state charts, controlling dynamic models.

The resulting model description language is called MOSILA [4]. The textual editor in which the model setup is done is expanded by the component diagram as in other Modelica simulators. But there exits another graphical layer which supports state chart definition by using UML diagrams (*U*nified *M*arkup *L*anguage). This is one main benefit compared to some other tools, because the event handling can be done intuitively and the thereby defined program code can be modified and extended in the textual layer as well. Moreover, simulator coupling with standard tools (e.g. MATLAB/Simulink, FEMLAB) is realized. Features for coupling a new simulator with MATLAB are in general used for optimization. The included MATLAB algorithms can be used and so, the system runs in co – simulation, whereby the model is defined intuitively in Modelica standard with additional states and the optimization routine is started in MATLAB/Simulink.

Mosilab offers a list of explicit and implicit integration methods for solving the defined system of DAEs (*D*ifferential *A*lgebraic *E*quations). The default method is the IDA Dassl routine. This method is capable to handle stiff systems. The other implemented methods are Explicit Euler, Implicit Euler, Implicit Trapeze and Explicit Trapeze.

3 Modeling

In this section three different models will be explained in detail. The first one, the constrained pendulum, is used to show the high flexibility of Mosilab and to represent the different ways of implementing a state event. The second is a linear model [6] for which an analytical solution exists and which is used to show the mathematical correctness of the implemented solution algorithms.

The third one gives an overview about advanced modeling and simulation with Mosilab/Modelica, it is a model of the free pendulum. Out of the given model definition we will see that a pure Modelica solution is not possible any more, because the dimension of the state space changes. This happens when a statechart is inevitable in Mosilab.

3.1 Constrained pendulum

The constrained pendulum is a classical nonlinear model in simulation techniques. To make the problem easier than it is in real life, we assume the mass m is large enough so that, as an approximation, we state that all the mass is contained at the bob of the pendulum (that is the mass of the rigid shaft of the pendulum is assumed negligible). This model has been presented in the definition of ARGESIM comparison C7 [5]. There is no exact analytical solution to this problem. Therefore, the results have to be obtained by numerical methods. In this section a description of the model will be given.

$$ml\,\ddot{\varphi} = -mg\,\sin(\varphi) - dl\,\dot{\varphi} \qquad (1)$$

Hereby φ denotes the angle in radiant measured in counter clockwise direction from the vertical position. The parameters in the model are the mass *m* and the length of the rope *l*. The damping is realized with the constant *d*. In Mosilab it is an important difference, if the modeler is using *constant* or *parameter*!

As it is a constrained pendulum a pin is fixed at a certain position. This position is given by the angle

angle φ_p and the length l_p. Every time when the rope of the pendulum hits the pin the length of the pendulum has to be shortened. In this case the pendulum swings on with the position of the pin as the point of rotation and the shortened length

$$l_s = l - l_p . \tag{2}$$

We will focus on the first example defined in the ARGESIM comparison C7, where the following parameters, constants, and initial values are defined:

$$\begin{aligned} & m = 1.02, l = 1, l_p = 0.7, g = 9.81 \\ & \varphi_{start} = \frac{\pi}{6}, \dot{\varphi}_{start} = 0, d = 0.2 \\ & \varphi_p = -\frac{\pi}{12} \end{aligned} \tag{3}$$

3.2 Two state model

The here defined model is based on the definition of the ARGESIM comparison C5 [6]. This is a system with two coupled differential equations with a classical parameter state event. The reason why we chose this more or less simple example is, that in contrast to the system defined in 3.1 this system can be solved analytically and therefore we can compare the solution generated in Mosilab with the original analytical solution. Furthermore the different model approaches can be compared pertaining to the solution quality.

This example tests the ability of the simulator to handle discontinuities of the aforementioned type in a satisfactory way. The problem is as follows

$$\begin{aligned} \dot{y}_1 &= c_1 (y_2 + c_2 - y_1) \\ \dot{y}_2 &= c_3 (c_4 - y_2). \end{aligned} \tag{4}$$

This ordinary differential equation (ODE) system is essentially a simple linear stiff problem with exponential decays as analytical solution. One of these is a very rapid transient, and the stationary solution of the slow decay varies from the two states of the model. This actually "drives" the model (and the discontinuity).

The parameter c_1 and c_3 stay unchanged during simulation. The parameter c_2 is 0.4 and c_4 is 5.5 when the model is in state 1 (also the initial state). The initial values are $y_1(0) = 4.2$ and $y_2(0) = 0.3$. The model remains in state 1 as long as $y_1 < 5.8$. The choice of c_2 and c_4 ensures that y_1 will grow past 5.8.

When the model switches to state 2, parameters c_2 and c_4 change to c_2 = -0.3 and c_4 = 2.73. The model remains in state 2 as long as $y_1 > 2.5$. When passing this instance the model switches back to state 1; the choice of c_2 and c_4 ensures that this will happen.

Analytical solution values can be found. We are focusing on a simulation period starting at time point 0 and ending at time point 5. For comparison we state that the last discontinuity occurs at time 4.999999646 and the $y_1(5.0)$ value should be approximately 5.369.

3.3 Free pendulum on a string

Until now the definitions of systems of interest have been looking on models where the state space dimension does not change during simulation. The state events can all be interpreted as simple parameter events. Now a system is given where the state space dimension has to be changed for real.

This example is a little bit more complicated. Let us again consider a pendulum. The massive bob of the pendulum is fixed on a string. The general structure of the system is depicted in Fig. 1 [5].

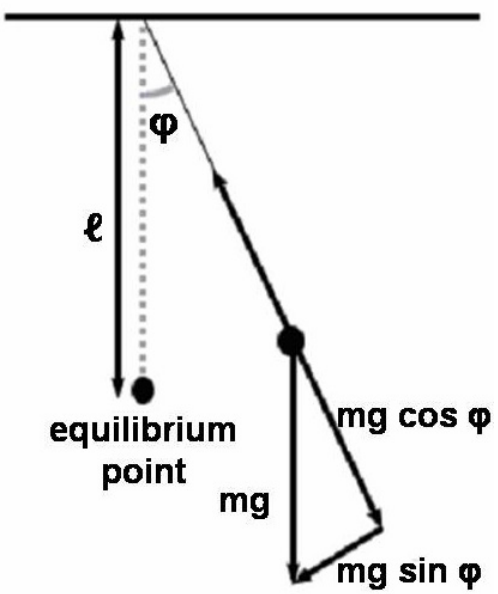

Fig. 1 Force diagram of the model

In case of a rollover of the pendulum it can start to fall freely until the constraints of the string apply again. This can happen if the pendulum swings higher than $\pm\pi/2$ and the centrifugal force is smaller than the gravitational force.

Accordingly, the so defined model has two different states:

- The normal pendulum movement

  and

- the free fall case.

The movement of the pendulum is given in equation (1). We have to define the equations for the free fall case. They are given by

$$\begin{aligned} \dot{v}_y &= -g \\ \dot{v}_x &= 0 \end{aligned} \tag{5}$$

For our model we have an additional constraint, which is based on the fixed length l of the pendulum:

$$x^2 + y^2 \leq l \qquad (6)$$

This model cannot be solved using simple parameter state events and is defined to give an example that problems in simulation of technical systems as well as in biology, genetics, etc. occur not only in very sophisticated systems. As seen here the need for state space switching in nowadays modeling and simulation techniques is quite common.

After the definition of the main tasks and the extra example we will have a closer look on the implementation approaches of the constrained pendulum and test the simulator by solving different solution of the two state model and comparing them with the analytical solution.

4 Solutions of the constrained pendulum

In this chapter the most important different solution approaches in Modelica of the classical constrained pendulum are discussed. Benefits and restrictions of the different implementations are listed. In the implementations of the constrained pendulum the tangential velocity is used instead of angular velocity. This has the benefit that only the length of the pendulum has a discrete change in case of hitting or leaving the pin.

4.1 Standard Modelica approach

In this approach only standard Modelica code is used. It is defined in the Mosilab equation layer, which is part of the model editor. The model can be formulated as implicit law, which means that it is not necessary to transform the equations to an explicit form:

```
equation  /*pendulum*/
v = l1*der(phi); vdot = der(v);
mass*vdot/l1+mass*g*sin(phi)+damping*v=0
```

The state event, which appears every time when the rope of the pendulum hits the pin or looses the connection to it, is modeled in an *algorithm* section with *if* (or *when*) – conditions:

```
algorithm
if (phi<=phipin) then length:=ls;
end if;
if (phi>phipin) then length:=l1;
end if;
```

This section defines the length of the rope depending on the actual state of the constrained pendulum. Mosilab handles the *if* – command by means of a state event finder. This is important to find the time point of the state event in a given time slice. The solution of the so defined system is depicted in Fig. 2.

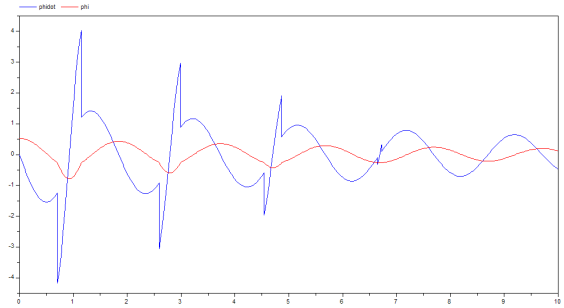

Fig. 2 Solution of the task defined in section 3.1, the red curve represents the angle, the blue curve depicts the angular velocity

In compare with the solutions done in another Modelica simulator (Dymola, in SNE [6]) and the reference solution, this outcome seems reasonable.

4.2 Mosilab state chart approaches

These approaches make use of an additional feature of Mosilab, namely modeling of discrete elements by state charts.

4.2.1 Parameter event solution

The state chart is used instead of the algorithm section and therefore instead of the *if*- or *when*- construct. This has the benefit of much higher flexibility and readability in case of complex conditions. Boolean variables define the status of the system and are managed by the state chart. This can be solved as follows:

```
event Boolean lengthen(start=false),
                shorten(start=false);
equation
lengthen=(phi>phipin);
shorten=(phi<=phipin);
.. here /*pendulum*/ -equations
statechart
state LengthSwitch extends State;
State
Short,Long,Initial(isInitial=true);
transition Initial -> Long
end transition;
transitionLong->Shortevent shorten
action
      length := ls;
end transition;
transitionShort->Longeventlengthen
action
      length := l1;
end transition;
end LengthSwitch;
```

From the modeling point of view, this is equivalent to the description with *if* – clauses. The Mosilab translator generates an implementation with different internal equations. Mosilab performs a simulation by handling the state event within the integration over the simulation period.

ISBN 978-3-901608-32-2

4.2.2 Model switching solutions

As already explained Mosilab's state chart engine is not only an alternative to the Modelica *if* – or *when* – construct, it is much more powerful.

This system allows any kind of hybrid model composition with models of different state spaces and also of different types. For the constrained pendulum we decompose the system into two different models:

- SHORT, for the case that the rope has contact to the pin

  and

- LONG, for the standard damped pendulum.

These two models are than controlled by a state chart, defined in a similar way as shown in the UML – diagram in Fig. 3.

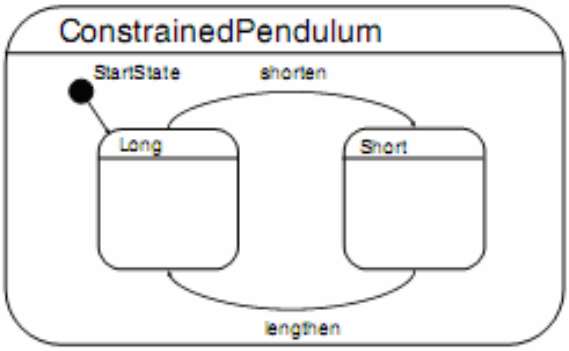

Fig. 3 UML – diagram of the statechart solution of the constrained pendulum. The main model controls the two submodels. In this example the LONG mode is the initial state.

As seen, the new model description comprehends now three parts, the main program which also consists of the state chart and two submodels. These two submodels can be defined separately, or because of the special structure, can be instances of one defined class.

The following source code is using the first method for implementation and first defines the two separate models and afterwards the main program.

```
model ConstrainedPendulum
model  Long
equation
       mass*vdot/l1+mass*g*sin(phi)+
       damping*v=0;
end  Long;
model  Short
equation
       mass*vdot/ls+mass*g*sin(phi)+
       damping*v=0;
end  Short;
event  discrete  Boolean
lengthen(start=true),
shorten(start  =  false);
equation
       lengthen=(phi>phipin);
       shorten=(phi<=phipin);
statechart
state  ChangePendulum  extends  State;
State
Short,Long,startState(isInitial=true);
transition  startState  ->  Long  action
       L:=new  Long();  K:=new  Short();
       add(L);
end  transition;
transition  Long->Short  event  shorten
action
       disconnect  ….;  remove(L);
       add(K);  connect  …
end  transition;
transition  Short  ->Longeventlengthen
action
       disconnect  …;  remove(K);
       add(L);  connect  ……
end  transition;
end  ChangePendulum;
end  ConstrainedPendulum;
```

The transitions organize the switching between the pendulums (*remove, add*). The *connect* statements are used for mapping local states to global state variables.

4.2.3 Summing up the results

In center of interest is also the difference in time behavior of the different solution methods. As this is a nonlinear model we can only calculate the numerical solutions and compare, for example, the time points where the last state event appears. This is the moment when the rope of the pendulum looses the connection to the pin the last time. In the model under investigation, this happens after the fourth time shortening the pendulum, which means after eight state events all together.

The solutions are calculated with the default simulation method, if possible. With this approach we try to test the simulation environments from the user's point of view. Many programmers and modelers do not care a lot about the implemented integration methods. For this reason the standard method has to produce reliable results in an appropriate calculation time.

Tab. 1 Comparison of the results

Simulation method	Time point of last event	Solution method
Pure Modelica	6.7199	Impl. Trapez Min. step 1e-6 Max. step 1e-4
Switch models	6.7204	IDA Dassl Min. step 1e-6 Max. step 0.08

As depicted in Tab. 1, the solutions are quite close but not identically the same. An explanation therefore is

that the standard Modelica solution cannot be done with the standard integration method. This could be examined making further tests with different choice of the minimal and maximal step size in each solution method.

5 Two state model

As this is another model where only parameters are changed in the case of the arrival of a state event, this model can be solved in four different ways as explained for the constrained pendulum in chapter 4.

The main differences in compare to the pendulum model are:

- An analytical solution exists for the model.
- Only the first derivative has to be calculated.
- The system is stiff.

The first solution is done again in standard Modelica notation. The most interesting part of the source code is implemented as follows:

```
algorithm
when (y1>=5.8) then
  c2:=-0.3;c4:=2.73;
end when;
when (y1<=2.5) then
  c2:=0.4;c4:=5.5;
end when;
equation
  der(y1)=c1*(y2+c2-y1);
  der(y2)=c3*(c4-y2);
```

The second way of solving this stiff system is to define a state chart in which only the parameters *c2* and *c4* are changed when an event occurs. We have two cases for this parameter state event. The one when the value of the variable *y1* gets higher than 5.8 and the second when this value falls below 2.5. For this model approach we only need one model in which the parameter rules are defined on a higher level.

```
event Boolean e(start=false),f(start =
false);
equation
…
statechart  state SCSol extends State;
State ax,bx,ix(isInitial=true);
     transition ix->ax end transition;

      transition ax->bx event e action
          c2:=-0.3;   //-1.25;
          c4:=2.73;  //4.33;
       end transition;

       transition bx->ax event f action
          c2:=0.4;
          c4:=5.5;
       end transition;
     end SCSol;
```

The source code above shows the easy way of implementation of this task.

The third way of implementation we will focus on concerning this example is to define two separate models which will then toggle between the states. This is the safest way for general implementation of systems with different states. On the one hand this cannot be done with the main part of simulators, on the other hand Mosilab is able to use this structure in different ways: the first approach is to define a submodel and switch between different instances of one and the same class (this would be enough in our case). The second solution is the general switching of active submodels to solve the system. This is done in the implemented solution.

```
model Zustand1 when (y1>=5.8) then
 constant Real c1=2.7*10^6;
  constant Real c3=3.5651205;
  constant Real c2=0.4;
  constant Real c4=5.5;
  Real y1;
  Real y2;
 equation
    der(y1)=c1*(y2+c2-y1);
    der(y2)=c3*(c4-y2);
end Zustand1;
model Zustand2   …    end Zustand2
event Boolean e(start=false),f(start =
false);
  Real y1(start=4.2);
  Real y2(start=0.3);
  dynamic Zustand1 Z1;
  dynamic Zustand2 Z2;

equation
e = (y1>5.8) or (y1 == 5.8);
f= (y1<2.5) or (y1 == 2.5);


statechart
state Zustandswechsel extends ANDState;
      State ax,bx,ix(isInitial=true);

   transition ix->ax action
    Z1:= new Zustand1();
    Z2:= new Zustand2();
    add (Z1);
    Z1.y1:=y1;
    Z1.y2:=y2;
    connect(Z1.y1,y1);
    connect(Z1.y2,y2);
   end transition;
   transition ax->bx event e action
    disconnect(Z1.y1,y1);
    disconnect(Z1.y2,y2);
    remove(Z1);
    Z2.y1:=y1;
    Z2.y2:=y2;
    add(Z2);
    connect(Z2.y1,y1);
    connect(Z2.y2,y2);
   end transition;
   transition bx->ax event f action
    disconnect(Z2.y1,y1);
    disconnect(Z2.y2,y2);
```



```
    remove(Z2);
    Z1.y1:=y1;
    Z1.y2:=y2;
    add(Z1);
    connect(Z1.y1,y1);
    connect(Z1.y2,y2);
   end transition;
end Zustandswechsel;
```

The compendium of the code above shows the basic structure of the problem solution. The next part represents the output part of the system.

5.1 The three solutions in compare

After defining the source code, the main interest of the user will focus on the quality of different implementations. From mathematical point of view the implemented solutions are equivalent.

The solutions are all calculated with standard solution method *IDADassl*. Two different step sizes are defined for the experiment. The first with

maxStep = 1e-6, minStep = 0.08,

the second experiment with

maxStep = 1e-12, minStep = 0.0008.

The other settings are all chosen by their default values. No changes are made. The results are all read out of the graphical interface. For more detailed output information the user can take a look at the generated data files to get more digits in the representation.

The results are depicted in the following table:

Tab. 2 Calculated values

	Settings 1	Settings 2
Event	5.0169	5.0000
Value at time point 5.0	5.7935	5.800–5.0998

As we can see in table 1, the values are exactly the same in all three approaches. This is very good for model reliability and necessary for further development. In compare with the exact solution of this problem (last event at time point 4.999999646 and the value $y_1(5.0)$ should be approximately 5.369.), we see that our simulation method works in an acceptable quality range. The imprecision of the output occurs also because the user gets only four digits after semicolon for the calculated value. Of course this is normally enough for standard technical system solution, but in our case, namely, for comparing with the exact analytical solution, it is not good enough. The value of the function at the time point 5 is in the allowed range.

6 Summary and Outlook

As pointed out in chapters 4 and 5 Mosilab is capable to handle as well nonlinear as linear stiff systems. The Modelica extension for state event handling is a strong tool for advanced modeling concepts. Nevertheless it is important to develop more features and work on the compatibility with the Modelica syntax, so that model exchange can be carried out.

The state chart extension of the Modelica notation is a very useful feature for modeling complex hybrid systems. Because of the state space switching ability it can be used to minimize the simulation time. Furthermore, the models get simpler and the number of equations, that have to be solved are the minimal number during computation.

The possibility to couple the simulation environment with Matlab/Simulink is another important feature of Modelica. As Matlab is very wide spread, a combination of both tools, especially in combination with modern simulation system development and optimization can be done efficiently.

7 References

[1] http://www.mosilab.de/

[2] http://mosilab.de/forschungsprojekt-gensim

[3] http://www.modelica.org/

[4] Thilo Ernst, André Nordwig, Christoph Nytsch-Geusen, Christoph Claus, André Schneider: MOSILA Modellbeschreibungssprache, Spezifikation, Version 2.0, from webpage: www.mosilab.de/downloads/dokumentation

[5] http://www.sparknotes.com/physics/oscillations/applicationsofharmonicmotion/section1.html

[6] http://www.argesim.org/comparisons/index.html

[7] F. Breitenecker, H .Ecker and I. Bausch-Gall. Simulation mit ACSL : eine Einführung in die Modellbildung, numerischen Methoden und Simulation . Braunschweig : Vieweg, 1993. - XI, 399 S

