Discrete Hybrid Automata Approach to Structural Dynamic Modelling and Simulation

Gašper Mušič, Borut Zupančič, University of Ljubljana, Slovenia, *gasper.music@fe.uni-lj.si*

The paper presents the discrete hybrid automata (DHA) modelling formalism and related HYSDEL modelling language. The applicability of the framework in the context of modelling of structural-dynamic systems is discussed. High level and partially modular modelling capabilities of HYSDEL are presented and the possibility of modelling structural-dynamic systems is shown and illustrated by a simple example. To model structural dynamics, standard HYSDEL list structures are employed, and additional dynamic modes are introduced when state re-initializations are necessary at mode switching. For the derived DHA models an efficient simulation algorithm is presented. The main features of the framework are compared to characteristics of other modelling and simulation tools capable of capturing structural dynamics. Although DHA modelling framework only permits the simulation of a corresponding maximal state space model, and the simulation precision is limited, it offers other advantages, e.g. straightforward translation of the model to various optimization problems that can be solved by standard linear or quadratic programming solvers.

SNE 18/2, August 2008

Introduction

Hybrid systems were recognized as an emerging research area within the control community in the past decade. With improvements to the control equipment the complexity of modern computer-control systems increases. Various aspects of discrete-event operation, such as controller switching, changing operating modes, communication delays, and interactions between different control levels within the computercontrol systems are becoming increasingly important. Hybrid systems, defined as systems with interacting continuous and discrete-event dynamics, are the most appropriate theoretical framework to address these issues.

Mathematical models represent the basis of any system analysis and design such as simulation, control, verification, etc. The model should not be too complicated in order to efficiently define system behaviour and not too simple, otherwise it does not correspond to the real process and the behaviour of the model is inaccurate. Many modelling formalisms for hybrid systems were proposed in the engineering literature [1, 2, 3] and each class of models is usually appropriate only for solving a certain problem.

A common approach to analyse the behaviour of the developed model is to apply simulation and observe the response in the time domain. When hybrid models are dealt with, a number of problems must be resolved, such as detection of state-events, generated when a predefined boundary in the state-space is reached by the state trajectory, or a proper treatment of discontinuities, such as re-initialization of the state at the so-called state jumps, etc. A number of related simulation techniques and tools has been developed that deal successfully with these problems. One of the most challenging issues from the simulation viewpoint is a proper treatment of state dependent changes in the model structure during the simulation run. This means that in dependency of events, which are triggered from the state of the model or its environment, the number and types of equations can change during the simulation. These changes are often designated by a term model structural dynamics.

In the paper an approach is presented, where the system is modelled as a *discrete hybrid automaton* (DHA) using a HYSDEL (HYbrid System DEscription Language) modelling language [4, 5]. Using an appropriate compiler, a DHA model described by the HYSDEL modelling language can be translated to different modelling frameworks, such as *mixed logical dynamical* (MLD), *piecewise affine* (PWA), *linear complementarity* (LC), *extended linear complementarity* (ELC) or *max-min-plus-scaling* (MMPS) systems [6]. The system described as an MLD system [7] can be effectively simulated using an additional information from the HYSDEL compiler. The approach was applied to a simple example of a structural-dynamic system, which illustrates the applicability of the framework.

1 Discrete hybrid automata

According to [4] a *Discrete hybrid automaton* (DHA) is the interconnection of a *finite state machine* (FSM), which provides the discrete part of the hybrid system,

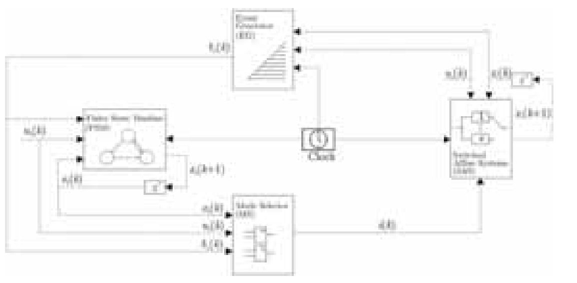

Figure 1. A discrete hybrid automation (DHA)

with a *switched affine system* (SAS) providing the continuous part of the hybrid dynamics. The interaction between the two is based on two connecting elements: the *event generator* (EG), which extracts logic signals from the continuous part, and *mode selector*, which defines the mode (continuous dynamics) of the SAS based on logic variables (states, inputs and events). The DHA system is shown on figure 1.

A switched affine system (SAS) represents a sampled continuous system that is described by the following set of linear affine equations:

$$x_r(k+1) = A_{i(k)}x_r(k) + B_{i(k)}u_r(k) + f_{i(k)} \tag{1a}$$

$$y_r(k) = C_{i(k)}x_r(k) + D_{i(k)}u_r(k) + g_{i(k)} \tag{1b}$$

where $k \in \mathbb{Z}_{\geq 0}$ represents the independent variable (time step) ($\mathbb{Z}_{\geq 0} \triangleq \{0,1,...\}$ is a set of nonnegative integers) $x_r \in \mathcal{X}_r \subseteq \mathbb{R}^{n_r}$ is the continuous state vector, $u_r \in \mathcal{U}_r \subseteq \mathbb{R}^{m_r}$ is the continuous input vector, $y_r \in \mathcal{Y}_r \subseteq \mathbb{R}^{p_r}$ is the continuous output vector, $\{A_i, B_i, f_i, C_i, D_i, g_i\}_{i\in\mathcal{I}}$ is a set of matrices of suitable dimensions, and $i(k) \in \mathcal{I}$ is a variable that selects the linear state update dynamics. A SAS of the form (1) changes the state update equation when the switch occurs, i.e. $i(k) \in \mathcal{I}$ changes. An SAS can be also rewritten as the combination of linear terms and *if-then-else* rules. The state-update function (1a) can also be written as:

$$z_1(k) = \begin{cases} A_1 x_r(k) + B_1 u_r(k) + f_1 & i(k) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2a}$$

$$\ldots$$

$$z_s(k) = \begin{cases} A_s x_r(k) + B_s u_r(k) + f_s & i(k) = s \\ 0 & \text{otherwise} \end{cases} \tag{2b}$$

$$x_r(k+1) = \sum_{i=1}^{s} z_i(k) \tag{2c}$$

An event generator (EG) generates a logic signal according to the satisfaction of linear affine constraints:

$$\delta_e(k) = f_H(x_r(k), u_r(k), k) \tag{3}$$

where $f_H : \mathbb{R}^{n_r} \times \mathbb{R}^{m_r} \times \mathbb{Z}_{\geq 0} \to \mathcal{D} \subseteq \{0,1\}^{n_e}$ is a vector of descriptive functions of a linear hyperplane. The relation f_H for time events is modeled as $[\delta_e^{\,i}(k) = 1] \leftrightarrow [kT_s \geq t_i]$, where T_s is the sampling time, while for threshold events is modeled as $[\delta_e^{\,i}(k) = 1] \leftrightarrow [a_i^T x_r(k) + b_i^T u_r(k) \leq c_i]$, where a_i, b_i, c_i represent the parameters of a linear hyperplane. $\delta_e^{\,i}$ denotes the i-th component of a vector $\delta_e(k)$.

A finite state machine (FSM) is a discrete dynamic process that evolves according to a logic state update function:

$$x_b(k+1) = f_B(x_b(k), u_b(k), \delta_e(k)) \tag{4}$$

where $x_b \in \mathcal{X}_b \subseteq \{0,1\}^{n_b}$ is the Boolean state, $u_b \in \mathcal{U}_b \subseteq \{0,1\}^{m_b}$ is the Boolean input, $\delta_e(k)$ is the input coming from the EG, and $f_B : \mathcal{X}_b \times \mathcal{U}_b \times \mathcal{D} \to \mathcal{X}_b$ is a deterministic logic function. An FSM may have also associated Boolean output:

$$y_b(k) = g_B(x_b(k), u_b(k), \delta_e(k)) \tag{5}$$

where $y_b \in \mathcal{Y}_b \subseteq \{0,1\}^{p_b}$.

A mode selector (MS) selects the dynamic mode $i(k)$ of the SAS according to the Boolean state $x_b(k)$, the Boolean inputs $u_b(k)$ and the events $\delta_e(k)$ using the Boolean function $f_M : \mathcal{X}_b \times \mathcal{U}_b \times \mathcal{D} \to \mathcal{I}$. The output of this function

$$i(k) = f_M(x_b(k), u_b(k), \delta_e(k)) \tag{6}$$

is called the *active mode*.

2 HYSDEL modelling language

DHA models can be built by using the HYSDEL modelling language [4], which was designed particularly for this class of systems. The HYSDEL modelling language allows the description of hybrid dynamics in textual form. The HYSDEL description of hybrid systems represents an abstract modelling step. Once the system is modelled as DHA, i.e. described by HYSDEL language, the model can be translated into an MLD model using an associated HYSDEL compiler. At this point, we will give just a brief introduction into the structure of a HYSDEL list.

A HYSDEL list is composed of two parts: the `INTERFACE`, where all the variables and parameters are declared, and the `IMPLEMENTATION`, which consists of specialised sections, where the relations between the variables are defined.

The `AD` section allows the definition of Boolean variables and is based on the semantics of the *event generator* (EG), i.e. in the `AD` section the δ_e variables are defined. The `LOGIC` section allows the specification of arbitrary functions of Boolean variables. Since the *mode selector* is defined as a Boolean function, it can be defined in this section. The `DA` section defines the switching of the continuous variables according to *if-then-else* rules depending on Boolean variables, i.e. part of *switched affine system* (SAS), namely z_i variables (see Equation (2)) are defined. The `CONTINUOUS` section defines the linear dynamics expressed as difference equations, i.e. defines the remaining Equation (2c) of the SAS. The `LINEAR` section defines continuous variables as an affine function of continuous variables, which in combination with the `DA` and the `CONTINUOUS` section enables more flexibility when modelling SAS. The `AUTOMATA` section specifies the state transition equations of the *finite state machine* (FMS) as a Boolean function (4), i.e. defines Boolean variables x_b. The `MUST` section specifies constraints on continuous and Boolean variables, i.e. defines the sets $\mathcal{X}_r$, $\mathcal{X}_b$, $\mathcal{U}_r$ and $\mathcal{U}_b$.

For more detailed description on the functionality of the modelling language HYSDEL and the associated compiler (tool HYSDEL), the reader is referred to [4, 5].

3 Structural-dynamic systems and HYSDEL

In general, discontinuities are modelled in HYSDEL by the use of auxiliary variables. Two types of such variables exist: Boolean or discrete (δ) and continuous (z).

3.1 Modelling of discontinuities

Discrete auxiliary δ variables may be defined based on continuous variables in the `AD` section of the HYSDEL list, which has the following syntax:

`AD{` *ad-item*$_+$ `}`

and each *ad-item* is one of the following:

var `=` *affine-expr* `<=` *real-number* `;`
var `=` *affine-expr* `>=` *real-number* `;`

The affine expression is a linear affine combination of real variables

$$a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \tag{7}$$

where a_i is a function of parameters, and x_i are real (state, input, output, and auxiliary) variables [5]. The δ variables defined in such a way represent outputs of the event generator (EG) in Fig. 1.

Continuous auxiliary z variables are defined in the DA section of the HYSDEL list, which has the following syntax:

`DA{` *da-item*$_+$ `}`

and each *da-item* is one of the following:

var `= { IF` *Boolean-expr* `THEN` *affine-expr* `};`
var `= { IF` *Boolean-expr* `THEN` *affine-expr*
`ELSE` *affine-expr* `};`

if the `ELSE` part is omitted, it is assumed to be 0.

The z variables defined this way can be used to implement switching dynamic part (SAS) of the HDA in figure 1. Using this approach, also the changes in the model structure can be easily implemented.

The actual continuous dynamic of the system is implemented in discretized form in the `CONTINUOUS` section, which has the following syntax:

`CONTINUOUS{` *cont-item*$_+$ `}`

and each *cont-item* is one of the following:

var `=` *affine-expr* `;`

Typically, a list of such *cont-item*s looks like:

```
x1 = z11 + z12 + ... + z1m ;
x2 = z21 + z22 + ... + z2m ;
...
xn = zn1 + zn2 + ... + znm ;
```

where n is the number of states and m the number of dynamical nodes. Auxiliary variables z_{11} to z_{nm} are defined within the `DA` section.

When the mode is not active the z_{ij} variables can be zero or may be held at any other value, depending on the problem.

The conditions related to reset of the state at switching or other similar conditions can be easily taken into account if a new mode is defined, which is active only at a single sampling instant.

With regard to structural changes, it is obvious that the states can not be created or deleted during the simulation run but can only be held 'inactive' when they are not required. Therefore the simulation runs by employing a corresponding maximal state space model.

3.2 Example

To illustrate the HYSDEL modelling of structural-dynamic systems a simple example will be shown. A system under consideration has two dynamic modes, the first one being active when the system output is below 0.5 and the second one when the output is above 0.5.

In the first mode the system dynamics is described by

$$\dot{y} + 0.5y = 0.5u \tag{8}$$

where u is the input to the system and y is the system output.

The second mode is described by

$$\ddot{y} + 2\dot{y} + y = u \tag{9}$$

The system is written in the state space form by assigning the state variables $x_1 = y$ and $x_2 = \dot{y}$, and discretized at the sampling time $T_s = 0.1\text{s}$. Then the first mode is described by:

$$\begin{aligned} x_1(k+1) &= a_{11}^1 x_1(k) + b_{11}^1 u_1(k) \\ y(k) &= x_1(k) \end{aligned} \tag{10}$$

and the second mode by

$$\begin{aligned} \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} &= \begin{bmatrix} a_{11}^2 & a_{12}^2 \\ a_{21}^2 & a_{22}^2 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} b_{11}^2 \\ b_{21}^2 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} \end{aligned} \tag{11}$$

Equations (10) and (11) are coded in the `IMPLEMENTATION` part of HYSDEL list as follows:

```
IMPLEMENTATION {
AUX {
  BOOL d, df;
  REAL z1, z21, z22; }
AD {
  d = x1 <= limit;
  df = a11_1*x1 + b11_1*u >= limit; }
DA {
  z1 = {IF d THEN
          a11_1*x1 + b11_1*u
        ELSE
          a11_2*x1 + a12_2*x2 + b11_2*u};
  z21 = {IF ~d THEN
          a21_2*x1 + a22_2*x2 + b21_2*u};
  z22 = {IF d & df THEN
          (a11_1 - a11_2)/a12_2*x1 +
          (b11_1 - b11_2)/a12_2*u}; }
CONTINUOUS {
      x1 = z1;
      x2 = z21 + z22; }
OUTPUT {
      y = x1; }
}
```

The *limit* parameter is set to 0.5, while other parameters are obtained by discretization procedure. It can be observed that an additional dynamic mode is introduced, which is active when x_1 is below the *limit* but the value of x_1 in the next time step exceeds the *limit*. This way the time of mode switching is predicted and x_2 is set to the value which causes a smooth transition to the new mode (both y and $\dot{y}$ are continuous). Both z_{21} and z_{22} are forced to zero when corresponding modes are inactive.

4 Simulation

Once a DHA system is modelled by the HYSDEL modelling language, the companion HYSDEL compiler generates the equivalent MLD model [4]. In [7] the authors proposed discrete-time hybrid systems denoted as *mixed-logical dynamical* (MLD) systems:

$$x(k+1) = Ax(k) + B_1 u(k) + B_2 \delta(k) + B_3 z(k) \tag{12a}$$

$$y(k) = Cx(k) + D_1 u(k) + D_2 \delta(k) + D_3 z(k) \tag{12b}$$

$$E_2 \delta(k) + E_3 z(k) \le E_1 u(k) + E_4 x(k) + E_5 \tag{12c}$$

where $x = [x_r, x_b]' \in \mathbb{R}^{n_r} \times \{0,1\}^{n_b}$ is a vector of continuous and logic states, $u = [u_r, u_b]' \in \mathbb{R}^{m_r} \times \{0,1\}^{m_b}$ are continuous and logic inputs, $y = [y_r, y_b]' \in \mathbb{R}^{p_r} \times \{0,1\}^{p_b}$ are continuous and logic outputs, $\delta \in \{0,1\}^{r_b}$, $z \in \mathbb{R}^{r_r}$ auxiliary logic and continuous variables, respectively, and A, B_1, B_2, B_3, C, D_1, D_2, D_3, E_1, …, E_5 are matrices of suitable dimensions. Inequalities (12c) can also contain additional constraints on the variables (states, inputs and auxiliary variables). This permits the inclusion of additional constraints and the incorporation of heuristic rules into the model.

4.1 The structure of an MLD form

Hybrid systems consist of continuous and logic variables. Relations between latter can be described by propositional calculus [7]. Propositional calculus enable statements to be combined in compound statements by means of connectives: "$\wedge$" (and), "$\vee$" (or), "$\neg$" (not), etc. Each compound statement can be translated into a conjunctive normal form (CNF) or product of sums (POS) of the following form

$$\wedge_{j=1}^{m} (\vee_{i \in P_j} X \ \vee_{i \in N_j} \neg X) \tag{13}$$

where P_j and N_j are sets of indices of literals X_i and inverted literals $\neg X_i$. By associating logical (binary) variables $\delta_i \in \{0,1\}$ with each propositional variable X_i then the compound statement (13) can be equivalently translated into a following set of integer linear inequalities:

$$\begin{aligned} 1 &\le \sum_{i\in P_1}\delta_i + \sum_{i\in N_1}(1-\delta_i) \\ &\vdots \\ 1 &\le \sum_{i\in P_m}\delta_i + \sum_{i\in N_m}(1-\delta_i) \end{aligned} \tag{14}$$

This translation technique can be adopted to model logic parts of processes, logic constraints of the plant and heuristic knowledge about plant operation, as integer linear inequalities.

A/D interface: Propositional variable X, defined by statement $X \triangleq [f(x_r) \le 0]$, i.e. $[f(x_r \le 0)] \leftrightarrow [\delta = 1]$, can be can be translated into the following set of mixed-integer inequalities

$$\begin{aligned} f(x_r) &\le M(1-\delta) \\ f(x_r) &\ge \varepsilon + (m-\varepsilon)\delta \end{aligned} \tag{15}$$

where ε is a small positive real number and M and m are constants defined by $M \triangleq \max f(x_r)$ and $M \triangleq \min f(x_r)$.

D/A interface: In this case the results of logical events define values of continuous variables. The most common D/A interface is the `IF-THEN-ELSE` construct, `IF` X `THEN` $z = f_1(x_r)$ `ELSE` $z = f_2(x_r)$, which can be translated into the following set of mixed-integer inequalities:

$$\begin{aligned} (m_2 - M_1)\delta + z &\le f_2(x_r) \\ (m_1 - M_2)\delta - z &\le -f_2(x_r) \\ (m_1 - M_2)(1-\delta) + z &\le f_1(x_r) \\ (m_2 - M_1)(1-\delta) - z &\le -f_1(x_r) \end{aligned} \tag{16}$$

where z is an auxiliary continuous variable defined by auxiliary logical variable δ associated to literal X. M_i and m_i are defined as in Equation (15).

Linear part enables to define linear relations as a system of inequalities and is defined as

$$\begin{aligned} z &\le f(x_r) \\ -z &\le -f(x_r) \end{aligned} \tag{17}$$

Continuous dynamical part is described by linear difference equations (discrete time domain) as follows

$$\begin{aligned} x_r(k+1) &= A_r x_r(k) + B_r u_r(k) \\ y_r(k) &= C_r x_r(k) + D_r u_r(k) \end{aligned} \tag{18}$$

By considering Equations (14,15,16,17,18) the mixed logical dynamical (MLD) system is derived and is presented by Equation (12). For more detailed description of the MLD structure the reader is referred to [4, 7].

4.2 Simulation of an MLD system

Using the current state $x(k)$ and input $u(k)$, the time evolution of (12) is determined by solving $\delta(k)$ and $z(k)$ from inequalities (12c), and then updating $x(k+1)$ and $y(k+1)$ from equalities (12a) and (12b), respectively. The MLD system (12) is assumed to be completely well-posed, i.e. for a given state $x(k)$ and input $u(k)$ the inequalities (12c) have a unique solution for $\delta(k)$ and $z(k)$. Obtaining the values of the auxiliary variables $\delta(k)$ and $z(k)$ presents a bottleneck in a simulation of a hybrid system modelled as an MLD system.

The variables $\delta(k)$ and $z(k)$ are defined by the system of inequalities (12c) and can be computed by defining and solving a mixed integer problem. It has to be pointed out that in this case the optimization is only used to find a feasible solution. Because the system is *well posed* the solution is unique and only one solution to the system of inequalities exists, which does not depend on the cost function. The disadvantage of this approach is the usage of the mixed integer optimization algorithms, which can be time consuming or even not able to find a feasible solution because of numerical sensitivity.

One of the reasons why the optimization approach is time consuming lies in the branch and bound nature of the underlying algorithm and in the fact that, once that the delta variables have been fixed, the inequalities (16) are actually equalities, i.e. $z = f_1(x_r)$ or $z = f_2(x_r)$.

To overcome the problem, which appears when using optimization approach, a special algorithm was developed. It is based on the knowledge of transformation procedure from DHA into MLD form and is able to compute values of $\delta(k)$ and $z(k)$ “explicitly”, i.e. without iterations. Such approach is of course much faster. The algorithm involves a direct E_1, …, E_5 matrix manipulation.

The algorithm abstracts the inequalities (12c) into sets based on an origin of a certain row. The result are six sets: `AD` set containing the inequalities from AD part of a system, `LOGIC` set containing the inequalities of logical relations, `LINEAR` set containing the linear relations, `DA` set containing inequalities from DA part of a system, `LOGIC MUST` set containing all logical constraints and `CONTINUOUS MUST` set containing all continuous constraints. The following algorithm exploits the definition of the variables $\delta(k)$ and $z(k)$ to define them:

1. Given $x(k)$ and $u(k)$.
2. **Repeat**
   a. Define $\delta_{AD}(k)$ variables for which all right hand side variables are defined.
   b. Define $\delta_{LO}(k)$ variables for which all right hand side variables are defined.
   c. Define $z_{LI}(k)$ variables for which all right hand side variables are defined.
   d. Define $z_{DA}(k)$ variables for which all right hand side variables are defined.
3. **Until** all $\delta(k) = [\delta_{AD}(k) \quad \delta_{LO}(k)]^T$ and $z(k) = [z_{LI}(k) \quad z_{DA}(k)]^T$ are defined.
4. Check logical constraints
5. Check continuous constraints
6. **If** all containts are fulfilled define $\delta(k)$, $z(k)$, new state $x(k+1)$ and output $y(k)$ **else** return error.

All the computation is based on direct E_1, …, E_5 matrix manipulation and does not rely on any mixed integer solver but relies on additional information provided by the HYSDEL tool, such as row origin information (AD, DA...). A similar algorithm is implemented in the HYSDEL tool, version 2.0.5 [4, 5].

4.3 Example

The described simulation algorithm is applied to the example model introduced in section 3.2. A periodic pulse signal with the amplitude 1 is defined as an input to the system and simulation results are shown in Fig. 2 and 3.

It can be observed that dynamics is changed when the system output crosses the boundary at 0.5. The change in the dynamics can be seen if the shapes of the rising and falling responses are compared. It can also be observed that x_2 state is initialized to the appropriate value whenever the second dynamic mode is entered. This value guarantees a smooth transition to the new mode. On the contrary, x_2 is switched to zero when the second mode is exited, because it is not needed anymore.

5 Comparison to other tools

A number of simulation techniques and tools has been developed in recent years that deal successfully with hybrid phenomena. Structural-dynamics as an important part of hybrid dynamics can be seen in one of the two distinct ways. In one way, state events can be seen as a mechanism that switches on and off algebraic conditions, which freeze certain states for certain periods. In another way, local model with fixed state spaces are controlled by a global model. Following this, two different approaches for simulating structural-dynamic systems are developed: the maximal state space approach and the hybrid decomposition approach [8].

Most currently available simulation tools follow the maximal state space model approach, e.g. Modelica, VHLD-AMS, Dymola. Matlab incorporates a simulation tool Simulink, which also works with a maximal state space. Simulink supports triggered sub-models, which can be executed only at event occurrences. Recent versions also include support for Statechart-

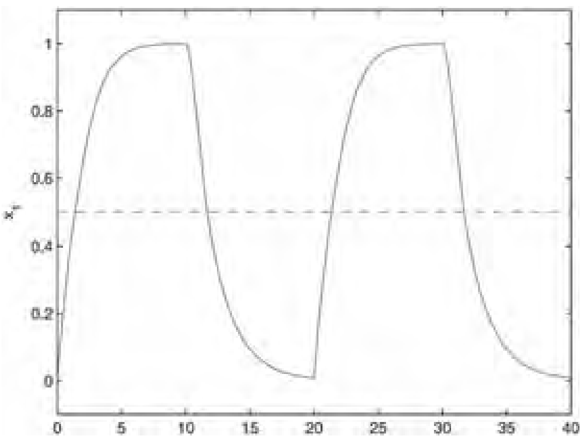

Figure 2. Simulated response ($x_1 = y$)

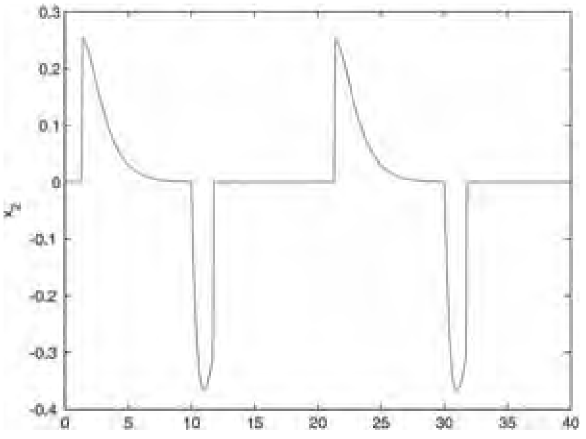

Figure 3. Simulated response (x_2)

based description of state-machines and discrete-event simulation by SimEvents Blockset, based on the entity concept. On the other hand, Simulink can not deal with Differential-Algebraic Equations (DAEs).

At the moment, the developments of hybrid decomposition approach are mainly focused on various extensions of Modelica. One of such extensions, which closely follows all basic principles of the hybrid decomposition approach, is a modelling description language Mosila (Modelling and Simulation Language). In Mosila, dynamical object structures are introduced to represent variable models. Objects represent state attributes and behaviour in a form of equations, and the equation system may be changed when a structural change is triggered. A corresponding simulator MOSILAB has been successfully applied to simulation of a number of case studies [9].

The DHA modelling and simulation approach presented in this paper belongs to the group of maximal state space model approaches. Since it operates in discrete-time, it is less elaborated from the simulation viewpoint. State and time events may be detected with a limited precision that is mainly influenced by the chosen sampling-time. On the other hand, the description has a sound theoretical framework and models can be converted to other formal descriptions of hybrid systems. This enables analytical exploration of important system properties, e.g. stability. Furthermore, the models converted to the MLD form can be used for defining various optimization problems that can be solved by standard linear or quadratic programming solvers.

6 Conclusions

The discrete hybrid automata (DHA) modelling formalism and related HYSDEL modelling language can be applied also to modelling and simulation of structuraldynamic systems. The modelling is simple and requires only the description of the switching boundaries in the state space and a discretization of the corresponding dynamics. The coding into HYSDEL list is straightforward and could also be automated based on a higher level description of the model. The simulation is fast and relatively simple. Compared to other tools the accuracy of simulation is limited, but on the other hand, the underlying DHA description can be easily transformed to other descriptions of hybrid systems and also used as a basis for analysis and optimization.

References

[1] P. J. Antsaklis. *A brief introduction to the theory and applications of hybrid systems*. Proceedings of the IEEE, 88(7):879–887, 2000.

[2] S. Engell, G. Frehse, E. Schnieder. *Modelling, Analysis and Design of Hybrid Systems*. Lecture Notes in Control and Information Sciences.

[3] G. Labinaz, M.M. Bayoumi, K. Rudie. *A survey of modeling and control of hybrid systems*. Annual Reviews in Control, 21:79–92, 1997.

[4] F. D. Torrisi, A. Bemporad. *Hysdel - a tool for generating computational hybrid models for analysis and synthesis problems.* IEEE Transactions on Control Systems Technology, 12:235–249, 2004.

[5] F. D. Torrisi, A. Bemporad, G. Bertini, P. Hertach, D. Jost, D. Mignone. *HYSDEL 2.0.5 - User Manual*. Automatic Control Laboratory, ETH, Zürich, Switzerland, 2002.

[6] W. P. M. H. Heemels, B. De Schutter, A. Bemporad. *Equivalence of hybrid dynamical models*. Automatica, 37(7):1085–1091, 2001.

[7] Bemporad, M. Morari. *Control of systems integrating logic, dynamic, and constraints*. Automatica, 35(3):407–427, 1999.

[8] F. Breitenecker, N. Popper. *Structure of simulation systems for structural-dynamic systems*. In Proceedings of the First Asia International Conference on Modelling & Simulation (AMS'07), pages 574–579, 2007.

[9] Nytsch-Geusen, T. Ernst, A. Nordwig, P. Schneider, P. Schwarz, M. Vetter, C. Wittwer, A. Holm, T. Nouidui, J. Leopold, G. Schmidt, U Doll, A Mattes. Mosilab: *Development of a modelica based generic simulation tool supporting model structural dynamic*. In Proceedings of the 4th International Modelica Conference, pages 527–535, 2005. Proc. EUROSIM 2007 (B. Zupančič, R. Karba, S. Blažič) 9-13 Sept. 2007, Ljubljana, Slovenia ISBN

Corresponding author: Gašper Mušič
University of Ljubljana
Faculty of Electrical Engineering
1000 Ljubljana, Trzaška 25, Slovenia
gasper.music@fe.uni-lj.si

Accepted	EUROSIM 2007, June 2007
Received:	September 20, 2007
Revised:	June 12, 2007
Accepted:	July 10, 2007