Control based on discrete-event models of continuous systems

CONTROL BASED ON DISCRETE-EVENT MODELS OF CONTINUOUS SYSTEMS

Patrick Philips, Martin Weiss, Heinz A. Preisig

Systems and Control Group, Faculty of Applied Physics
Eindhoven University of Technology,
P.O.Box 513, 5600 MB, Eindhoven, The Netherlands
fax : +31(0)402437170 and e-mail : p.p.h.h.philips@ctrl.phys.tue.nl

Keywords : Discrete-event models, Continuous system, Hybrid systems.

Abstract

A control stategy based on a possibly non-deterministic discrete-event model of a continuous system is proposed. Such a model can arise when a continuous plant is to be supervised by some computer program which uses discrete state information. The discrete-event representation of the continuous system will often be non-deterministic and therefore difficult to control. The contol method proposed exploits the fact that the original system is continuous. An example shows the possibilities of the control method.

1 Introduction

Discrete-event representations (models) of continuous systems are frequently arising in systems and control theory. For example, a system that is continuous by nature but is only observed by discrete sensors (i.e. sensors only detecting when a state crosses a certain boundary), can be represented by a discrete model. Also the analysis of some hybrid systems can be facilitated by transforming the continuous part into a discrete-event part such that only a discrete-event system has to be investigated which may be less difficult than studying the underlying hybrid system. A typical situation is when a (possibly controlled) continuous plant is to be supervised by some programmable logical controller or computer program which uses discrete state information. Since the discrete controller cannot communicate with the system at a continuous level it is necessary to use an interface. This corresponds to discretizing the continuous state space of the system into a finite set of symbols to be used by the controller. Various formalisms and methods have been proposed to describe these kind of systems and to solve the discretizing problem.

In [LUN94] a qualitative modelling approach for linear, discrete-time systems is presented and necessary and sufficient conditions are given for which the resulting automaton is deterministic. The discretizing method proposed in [STI93] is able to deal with non-linear systems and is based on a test which has similarities with the method for linear systems presented in [PRE96], which is extended for nonlinear systems in [PHI97]. The resulting discrete-event representation will in most cases be non-deterministic. This implies that, given a discrete state and a discrete input, more than one new state is possible. This makes it difficult to design a controller, since it is not clear what the result of a control action will be.

In this paper a control strategy based on a possibly non-deterministic discrete-event model of a continuous system is proposed. The method explicitly exploits the fact that the underlying system of the discrete-event model is continuous by nature. The paper is organized as follows. First, the continuous and discrete-event models will be presented. Then an alternative representation of the discrete-event model is given to enhance the computations necessary for the control strategy proposed in the next chapter. The control strategy is illustrated by means of a three tank example. Finally some conclusions are drawn.

2 The discrete-event model

The point of departure is a continuous system which is to be transformed into a discrete-event model. Suppose a dynamical system described by a set of ordinary differential equations in $\mathbb{R}^n$ is given:

$$\dot{x} = f(x, u), \qquad x \in \mathbb{R}^n \tag{1}$$

It is assumed that f is continuous and that the system (1) has a unique solution for every initial value in the region of interest.

The discrete-event model which has to result is given by the system [SON90]:

$$\Sigma = (\mathcal{T}, \mathcal{X}, \mathcal{U}, \phi)$$

where the time set $\mathcal{T} = \mathbb{Z}$, and the finite set of inputs $\mathcal{U}$, also called the input alphabet are given. Next, a finite set

of discrete states $\mathcal{X}$, and the possibly non-deterministic transition function $\phi : \mathcal{X} \times \mathcal{U} \to \mathcal{P}(\mathcal{X})$ have to be determined.

2.1 Discrete states

First, the state space is partitioned into regions. For this purpose for each coordinate x^i, $(i = 1...n)$ of x a set of boundaries is chosen:

$$\beta_0^i < \beta_1^i < \ldots < \beta_{n_i}^i \quad (n_i \geq 1). \tag{2}$$

A discrete state is given by an integer $\tilde{x} \in N_p = \{1, ..., p\}$, where $p = \prod_i n_i$. The discrete state $\tilde{x}$ is connected with the continuous state space due to its relation with a *discrete state region,* $R(\tilde{x})$ defined as the bounded region in $\mathbb{R}^n$

$$R(\tilde{x}) = \{x \in \mathbb{R}^n \,|\beta_{j_i-1}^i \leq x^i \leq \beta_{j_i}^i, \ \forall i\}. \tag{3}$$

where $j_i \in \{1, ..., n_i\}$ is chosen such that there are p different discrete state regions covering the region of interest.

Two discrete states are adjacent if the corresponding discrete state regions share an $(n-1)$-dimensional boundary. The crossing of a boundary, that is the transition from one discrete state to another is called a *discrete event.*

It will be convenient to represent the discrete state $\tilde{x}$ by means of a boolean vector $\hat{x} \in B^p = \{0, 1\}^p$. The transformation from the integer to the boolean vector representation $IB : N_p \to B^p$ follows immediately:

$$IB(\tilde{x}) = [0, 0, ..., 0, 1, 0, ..., 0, 0]^T$$

with the 1 at the $\tilde{x}$-*th* position. Consequently the transformation $BI : B^p \to N_p$ is defined by

$$BI(\hat{x}) = k \text{ if the } k\text{-}th \text{ element of } \hat{x} \text{ is } 1$$

In the boolean vector representation a vector is called a set if the integer representation of that vector is a set. The set $\{\tilde{x}_i\}$ then is given by

$$\hat{x} = IB(\{\tilde{x}_i\}) = \sum_{\tilde{x} \in \{\tilde{x}_i\}} IB(\tilde{x})$$

where the summation has to be taken as a boolean vector addition. In this way it is possible to call a vector $\hat{x}_1$ a subset of $\hat{x}_2$ if $\{\tilde{x}_i\}_1 \subseteq \{\tilde{x}_i\}_2$. In the boolean vector notation it is easy to check if $\hat{x}_1 \subseteq \hat{x}_2$ since

$$\hat{x}_1 \subseteq \hat{x}_2 \Longleftrightarrow \hat{x}_1 + \hat{x}_2 = \hat{x}_2$$

2.2 Transition function

A discrete event approximation of the continuous model has to satisfy the following property: let $\tilde{x}_1$ and $\tilde{x}_2$ be two adjacent states, then a transition $\tilde{x}_2 = \phi(\tilde{x}_1, u_i)$ is admissible iff there exists a solution $x(t)$ of (1) with $u = u_i$, starting with initial condition $x(t_1) = x_1 \in R(\tilde{x}_1)$ and crossing $x_2 \in R(\tilde{x}_2)$ at time t_2 such that $x(t) \in R(\tilde{x}_1) \cup R(\tilde{x}_2)$ for $t_1 \leq t \leq t_2$.

From this it can be seen that determining the transition function is equal to answering the question whether or not there exists a solution of (1) with constant $u_i \in \mathcal{U}$, that starts in $R(\tilde{x}_1)$ and reaches $R(\tilde{x}_2)$ where $\tilde{x}_1$ and $\tilde{x}_2$ are any two adjacent states. In [PRE96] a sufficient condition is presented to answer this question. The key-idea here is that if at the boundary between $\tilde{x}_1$ and $\tilde{x}_2$ the derivative of $x(t)$ given by (1) has a component in the direction of $R(\tilde{x}_2)$, then due to continuity, a transition from $\tilde{x}_1$ to $\tilde{x}_2$ is possible. A computer program checking this condition can automatically generate the transition function. An implementation of this method, described in [PHI97] results in *automaton tables.* An automaton table A is a table in which, given a discrete state, all possible new states can be found. Each discrete input $u_i \in \mathcal{U}$ requires the computation of an automaton table A_i. Then the transition function ϕ is characterized by $(\{u_1, ...u_k\}, \{A_1, ..., A_k\})$. In almost all cases the transition function ϕ will be non-deterministic, that is given u_i, more than one possible new discrete state will be possible.

2.3 Interconnection Matrices

The information an automaton table A contains can be represented by means of an interconnection matrix (adjacency matrix). An interconnection matrix $\hat{A} \in B^{p \times p} = \{0, 1\}^{p \times p}$ is a boolean matrix where

$$\hat{a}_{ij} = \begin{cases} 1 & \text{if } i = \phi(j, u) \\ 0 & \text{else} \end{cases}$$

In the boolean vector domain it is easy to compute the new set $\{\tilde{x}_i\}_2$ given an automaton table A and the initial set $\{\tilde{x}_i\}_1$, that is $\hat{x}_2 = \hat{A}\hat{x}_1$.

3 Control

Using the information provided by the interconnection matrices to control the system will be very difficult due to its non-deterministic nature. The control of non-deterministic discrete-event systems is still in development (see *e.g.*[HEY98]). There is however a difference between a discrete-event system resulting from a continuous system and a regular discrete-event system. The difference is that in the first situation it may be possible to let a control action respond to an event (measurement) as to 'undo' this event. That is, instead of trying to make sure transitions cannot occur at all (which is very difficult due to the non-determinism), sometimes a transition can be corrected. The motivation for this is that if the continuous system crosses a boundary from one discrete state to another, and it is certain that there exists an input for which at that particular boundary the first derivative of x at each point at the boundary has a component in the opposite direction, then due to continuity close enough

just after that boundary also the first derivatives of x has a component in the opposite direction. This will be exploited to construct a control strategy.

3.1 The Control Algorithm

The control problem is to find a sequence of control inputs such that, starting from the initial discrete state, $\hat{x}_0$ the system will evolve towards the target discrete state, $\hat{x}_e$. The control strategy proposed is based on the following two operations:

1. InvSet($\hat{x}$): Compute the smallest set, say $\hat{x}_{inv}$ containing the set $\hat{x}$ for which there exists inputs preventing the system leaving this set $\hat{x}_{inv}$ (so it is a controlled invariant set)

2. NeighSet($\hat{x}$): Compute the set of neighbors of $\hat{x}$, i.e. the discrete states sharing boundaries with $\hat{x}$, that can force the transition to $\hat{x}$ and create a new set, say $\hat{x}_{ext}$ from the union of this neighbors with $\hat{x}$ (so $\hat{x}$ is extended with its neighbors from which a transition to $\hat{x}$ can be forced)

With 'force a transition' it is meant that if the continuous state is on the boundary of a discrete state, then there exists an input such that a transition to the adjacent discrete state is possible, whereas the opposite transition is not possible (with that same input). Notice that this is a much weaker condition than the requirement that given a discrete state there must exist an input such that a transition to a certain adjacent state is the *only* possible transition.

The operations InvSet($\hat{x}$) and NeighSet($\hat{x}$) are easily expressed in the boolean vector domain. First define the Neighbor interconnection matrix, $\hat{N} \in B^{p\times p}$ in the following manner:

$$\hat{n}_{ij} = \begin{cases} 1 & \text{if } i \text{ is a neighbor of } j \\ 0 & \text{else} \end{cases}$$

With $\hat{N}$ it is clear that $\hat{x}_2 = \hat{N}\hat{x}_1$ contains all (adjacent) neighbor states of $\hat{x}_1$.

Next, define the Smallest interconnection matrix, $\hat{S}$ and the Largest interconnection matrix, $\hat{L}$ as

$$\hat{S} = \hat{A}_1 \& \hat{A}_2 \& ... \& \hat{A}_k$$
$$\hat{L} = \hat{A}_1 + \hat{A}_2 + ... + \hat{A}_k$$

with $C = A\&B$ defined as $c_{ij} = a_{ij} \cdot b_{ij}$ and $C = A + B$ is defined as $c_{ij} = a_{ij} + b_{ij}$.

The interpretation of $\hat{S}$ is that $\hat{x}_2 = \hat{S}\hat{x}_1$ contains all neighbor states of $\hat{x}_1$ that *cannot be avoided by any input*. The interpretation of $\hat{L}$ is that $\hat{x}_2 = \hat{L}\hat{x}_1$ contains all neighbor states of $\hat{x}_1$ that *can be reached by some input*.

With this computation of InvSet($\hat{x}$) and NeighSet($\hat{x}$) become

Algorithm 1 InvSet($\hat{x}$)

$$\hat{x}_{inv} = (\sum_{i=0}^{q-1} \hat{S}^i)\hat{x}$$

with

$$q = \min\{l \mid (\sum_{i=0}^{l} \hat{S}^i)\hat{x} = (\sum_{i=0}^{l-1} \hat{S}^i)\hat{x}\}$$

Algorithm 2 NeighSet($\hat{x}$)

$$\hat{x}_{ext} = \hat{N}^{for}\hat{x}$$

with $\hat{N}^{for} \in B^{p\times p}$ *defined as*

$$\hat{N}^{for} = (\hat{N} - \hat{A}_1)\&A_1^T + (\hat{N} - \hat{A}_2)\&\hat{A}_2^T + ...$$
$$... + (\hat{N} - \hat{A}_k)\&\hat{A}_k^T$$

To facilitate the computation of InvSet($\hat{x}$) it is assumed that it is measured when the continuous state is at a boundary. This allows the controller to respond when a boundary actually is hit instead of when a boundary already is crossed.

3.1.1 Controlled Invariant Sets

With these operations the controlled invariant sets can be computed. Set $\hat{x} = \hat{x}_e$ and $k = 1$ and do

step 1 InvSet($\hat{x}$), for the resulting $\hat{x}_{inv}$, label it $\hat{x}_{inv}^k$ and do

step 2 NeighSet($\hat{x}_{inv}$). If $\hat{x}_{inv}$ does not contain $\hat{x}_0$ set $\hat{x} = \hat{x}_{ext}$ and $k = k + 1$ and do step1, else stop.

The result is a set of controlled invariants sets $\{\hat{x}_{inv}^k\}$ satisfying $\hat{x}_{inv}^k \subset \hat{x}_{inv}^{k+1}$. The control strategy consists of two phases:

Correction Phase If a transition took place which resulted in a discrete state that is in a larger controlled invariant set than the previous discrete state was, then this transition has to be corrected.

Improvement Phase If no transition has to be corrected then there is the freedom to use the inputs for trying to let the state evolve to a smaller invariant set, that is if the state is in $\hat{x}_{inv}^k$ then try to go to $\hat{x}_{inv}^{k-1}$ until the state is arrived in $\hat{x}_{inv}^1$.

3.1.2 Correction Phase

In a control situation it is kept track of the discrete state the continuous state is in and a number r is assigned to it:

$$r = \min\{l | \hat{x} \in \{\hat{x}_{inv}^k\}\}$$

If after a transition from state $\hat{x}_1$ to $\hat{x}_2$ it appears that $r_2 > r_1$ which means that the state has entered a larger control invariant subset, then a correction is necessary. Now an input must be chosen that forces the opposite transition. From the construction of the controlled invariant sets it is clear that such input exists if some care is taken of the choice of the inputs in the improvement phase, as will be explained in the next.

3.1.3 Improvement Phase

If it is not necessary to correct a transition then the inputs can be used to let the state evolve to a smaller controlled invariant set. To do this, first it must be recognized that, depending on the discrete state the continuous state is in, there may exist inputs that have to be avoided. For characterizing all illegal inputs the following observations must be formalized.

Condition 3 *A transition from $\hat{x}^k_{inv}$ to $\hat{x}^{k-1}_{inv}$ is only possible if the continuous state is in a subset of $\hat{x}^k_{inv}$ whose elements are neighbors of $\hat{x}^{k-1}_{inv}$ and for which there exists an input such that a transition to $\hat{x}^{k-1}_{inv}$ is possible. Furthermore only inputs are allowed for which no uncorrectable transition to a specific set $(\hat{x}^k)^{ill}$ to be defined is possible. The subset of $\hat{x}^k_{inv}$ satisfying these conditions will be denoted $(\hat{x}^k_{inv})^{trans}$.*

Condition 4 *If a discrete state $\hat{x} \in \hat{x}^k_{inv} \setminus (\hat{x}^k_{inv})^{trans}$ has a path to $(\hat{x}^k_{inv})^{trans}$ then this path is legal only if it is realizable with inputs for which no uncorrectable transitions to a specific set $(\hat{x}^k)^{ill}$ to be defined is possible. The subset of $\hat{x}^k_{inv}$ satisfying this condition (which includes $(\hat{x}^k_{inv})^{trans}$) will be denoted $(\hat{x}^k_{inv})^{legal}$.*

Condition 5 *There is a legal path from $\hat{x}^{k+1}_{inv}$ to $\hat{x}^{k-1}_{inv}$ if there is path from $\hat{x}^{k+1}_{inv}$ to $(\hat{x}^k_{inv})^{legal}$ which is realizable with inputs for which no uncorrectable transitions to $(\hat{x}^{k+1})^{ill}$ are possible.*

The first two conditions lead to the construction of a new set of controlled invariant subsets $\{(\hat{x}^k_{inv})^{legal}\}$ with $\hat{x}^k_{inv} \supseteq (\hat{x}^k_{inv})^{legal}$. The third condition states that in order to prevent that no further improvement is possible, the state has to be navigated from $(\hat{x}^k_{inv})^{legal}$ to $(\hat{x}^{k-1}_{inv})^{legal}$, that is, it must be taken care of that the state is always in a legal set.

It is clear that $(\hat{x}^k)^{ill}$ are those states that are disadvantageous and therefore should be avoided. These states are 1) those who are in a larger controlled invariant set than the one the continuous state is at the current moment and 2) those for which no further improvement of the state evolution is possible within the proposed control strategy. This are the states for which no legal path is possible to states for which a transition to a smaller controlled invariant subset is possible. With this $(\hat{x}^k)^{ill}$ becomes:

$$(\hat{x}^k)^{ill} = (\hat{x}^l_{inv} \setminus \hat{x}^k_{inv}) \cup (\hat{x}^k_{inv} \setminus (\hat{x}^k_{inv})^{legal}) \cup ... \\ ... \cup (\hat{x}^1_{inv} \setminus (\hat{x}^1_{inv})^{legal})$$

where l denotes the number of sets in $\{x^k_{inv}\}$.

From condition 3 and condition 4 it is clear that $\{(\hat{x}^k_{inv})^{legal}\}$ has to be computed in an iterative scheme, since $(\hat{x}^k_{inv})^{trans}$ and $(\hat{x}^k_{inv})^{legal}$ depend on $(\hat{x}^k_{inv})^{ill}$ which in turn depends on $(\hat{x}^k_{inv})^{trans}$ and $(\hat{x}^k_{inv})^{legal}$. Notice that $(\hat{x}^1_{inv})^{legal} = \hat{x}^1_{inv}$. Now define

$$\hat{S}^{for} = L \;\&\; (A_1 + (N - A_1^T)) \;\&\; (A_2 + (N - A_2^T)) \;\&... \\ ...\&\; (A_k + (N - A_k^T))$$

and notice that $\hat{x}_2 = \hat{S}^{for}\hat{x}_1$ are those states for which a transition from $\hat{x}_1$ cannot be corrected.

Algorithm 6 *Set $(\hat{x}^1_{inv})^{legal} = \hat{x}^1_{inv}$, $(\hat{x}^k_{inv})^{trans}_0 = \hat{x}^k_{inv}$, $\hat{x}^{l+1}_{inv} = \hat{N}\hat{x}^l_{inv}$, and $(\hat{x}^k_{inv})^{ill}_0 = \hat{x}^{l+1}_{inv} \setminus \hat{x}^k_{inv}$. For $j = 2, ..., l$ do*

At the *r-th* iteration, do

Step 1 for all $\hat{A}_i$, $i = 1, ..., k$ compute

$$(\hat{p}^j_r)^i = \hat{A}_i^T (\hat{x}^{j-1}_{inv})^{legal} \cup (\hat{x}^{j-1}_{inv})^{legal} \cap (\hat{x}^j_{inv})^{trans}_{r-1}$$
$$(\hat{q}^j_r)^i = (\hat{A}_i^T (\hat{x}^j)^{ill}_{r-1} \cap (\hat{S}^{for})^T (\hat{x}^j)^{ill}_{r-1}) \cap (\hat{x}^j_{inv})^{trans}_{r-1}$$
$$(\hat{x}^j_{inv})^{trans,i}_r = (\hat{p}^j_r)^i \setminus (\hat{q}^j_r)^i$$

Then set

$$(\hat{x}^j_{inv})^{trans}_r = ((\hat{p}^j_r)^1 \setminus (\hat{q}^j_r)^1) \cup ((\hat{p}^j_r)^2 \setminus (\hat{q}^j_r)^2) \cup ... \\ ... \cup ((\hat{p}^j_r)^k \setminus (\hat{q}^j_r)^k)$$

Step 2 Set $(\hat{x}^j_{inv})^{legal}_r = (\hat{x}^j_{inv})^{trans}_r$ and do

Step 2a for all $\hat{A}_i$, $i = 1, ..., k$ compute

$$(\hat{p}^j_r)^i = \hat{A}_i^T (\hat{x}^j_{inv})^{legal}_r \cap (x^j_{inv} \setminus (\hat{x}^j_{inv})^{legal}_r)$$
$$(\hat{q}^j_r)^i = (\hat{A}_i^T (\hat{x}^j)^{ill}_{r-1} \cap (\hat{S}^{for})^T (\hat{x}^j)^{ill}_{r-1}) \cap \\ \cap (x^j_{inv} \setminus (\hat{x}^j_{inv})^{legal}_r)$$
$$(\hat{x}^j_{inv})^{legal,new,i}_r = (\hat{p}^j_r)^i \setminus (\hat{q}^j_r)^i$$

Step 2b Set

$$(\hat{x}^j_{inv})^{legal,new}_r = ((\hat{p}^j_r)^1 \setminus (\hat{q}^j_r)^1) \cap \\ \cap ((\hat{p}^j_r)^2 \setminus (\hat{q}^j_r)^2) \cap ... \cap ((\hat{p}^j_r)^k \setminus (\hat{q}^j_r)^k)$$
$$(\hat{x}^j_{inv})^{legal}_r = (\hat{x}^j_{inv})^{legal,new}_r \cup (\hat{x}^j_{inv})^{legal}_r$$

and go to Step 2a until $(\hat{x}^j_{inv})^{legal}_r$ does not change any more. Set

$$(\hat{x}^j)^{ill}_r = (\hat{x}^l_{inv} \setminus \hat{x}^k_{inv}) \cup (\hat{x}^j_{inv} \setminus (\hat{x}^j_{inv})^{legal}_r) \cup ... \\ ... \cup (\hat{x}^1_{inv} \setminus (\hat{x}^1_{inv})^{legal})$$

Do the $r+1$-*th* iteration or stop if nothing changes anymore, in which case we have determined $(\hat{x}^j_{inv})^{legal}$ and $(\hat{x}^j)^{ill}$ and we can continue with computing the $j+1$-*th* sets.

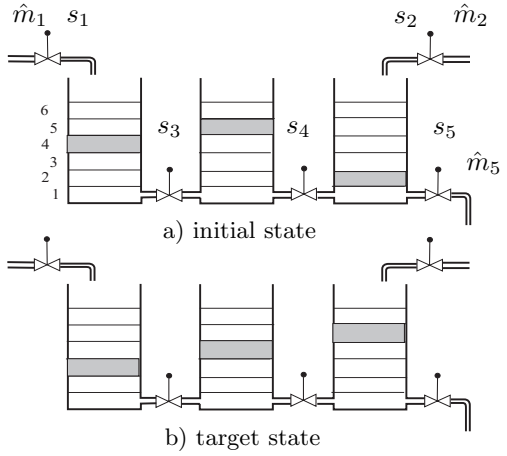

Figure 1: The three tank system with initial and target discrete states marked

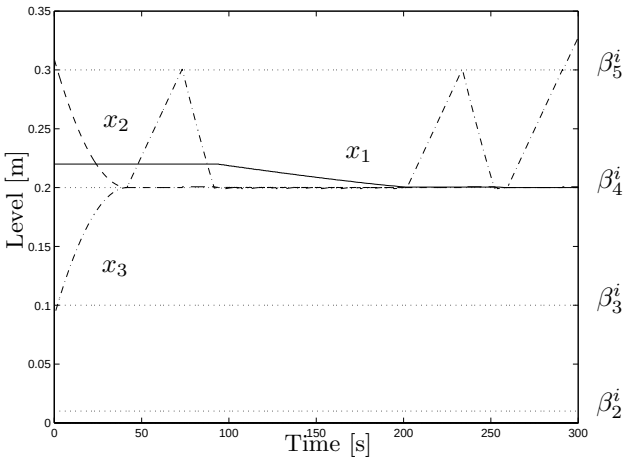

Figure 2: Time response for initial state $\hat{x}_{init} = (4, 5, 2)$ and target state $\hat{x}_{tar} = (3, 4, 5)$

3.2 Limitations

If the construction of the controlled invariant sets is not possible, that is if there is just one controlled invariant set covering the whole region of interest, then the method will fail. This situation can occur when a coordinate x^i cannot directly be influenced by any discrete input. Furthermore, in the former it is assumed that all sequences $\hat{x}_k = L^k \hat{x}_1$ are in fact possible. If this is not true then there may exists more illegal states. Also no improvement is possible if transitions to a smaller legal set just do not happen and the state evolves in a legal set without leaving it. These controllability related issues are the topics for future research.

4 Example: Three Tank Systems

The three tank system has the configuration depicted in Fig 1. It consists of three communicating tanks which are connected through small pipes. The input $u = (s_1, s_2, s_3, s_4, s_5)$ of the system are the on/off switches, $s_1, \ldots, s_5$ controlling the valves, where $s_i \in \{0, 1\}$. The state vector $x = [L_1, L_2, L_3]^T$is given by the water levels in each tank. Each tank is divided into six parts (discrete states) that are defined by the levels: 0, 0.01, 0.1, 0.2, 0.3, 0.4, and 0.5 $[m]$. It is only observed if a level is reached.

For each possible input combination (that is, for each discrete input) the corresponding automaton table has to be computed. So having five on/off switches gives $2^5 = 32$ different input combinations. With these 32 automaton tables, one is able to predict, given any initial state, the next possible state(s) for input u. Now the controller invariant sets $\{(\hat{x}_{inv}^k)^{legal}\}$ and the set of illegal state $\{(\hat{x}_k)^{ill}\}$ can be computed, following the procedure described in Section 3.1.

The aim is to end at $x_{tar} = [0.12, 0.24, 0.34]^T$, that is the discrete state $\hat{x}_{tar} = (3, 4, 5)$ from the initial state

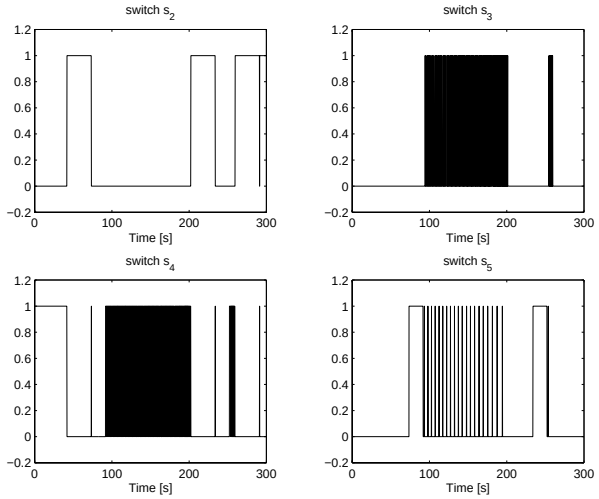

Figure 3: The inputs s_2, s_3, s_4 and s_5

$x_{init} = [0.22, 0.31, 0.09]^T$, that is $\hat{x}_{init} = (4, 5, 2)$, see Fig. 1. Using the proposed control strategy, a possible state-evolution is shown in Fig. 2. From Fig. 2 it can be seen that the control objective (in terms of discrete states) is achieved. For this, the level of the second tank first is adjusted such that the first tank is in a position to have an outflow. Then the level of the second tank rises again, although this is difficult to see from Fig. 2. The corresponding input is depicted in Fig. 3. Only the positions of the switches s_2, s_3, s_4, and s_5 are shown; $s_1 = 0$ during the simulation. Fig. 3 shows that the correction of transitions can result in scattering of the inputs, which may be undesirable in practical situations. This scattering possibly can be reduced or even prevented by changing the strategy for choosing an input from the set of legal inputs in the improvement phase. This will be a topic for future research.

5 Conclusions

A control strategy for a possibly non-deterministic discrete-event model of a continuous system is proposed. The strategy explicitly uses the fact that the discrete event model is based on a continuous system. The control strategy is applied to a three tank example.

References

[HEY98] Michael Heymann and Feng Lin. "Discrete-event control of nondeterministic systems" *IEEE Transactions on Automatic Control*, **43(1)**:3–17, January (1998).

[LUN94] J. Lunze. "Qualitative modelling of linear dynamical systems with quantized state measurements" *Automatica*, **30(3)**:417–431, (1994).

[PHI97] Patrick Philips, Udo Bruinsma, Martin Weiss, and Heinz A. Preisig. "A mathematical approach to discrete-event dynamic modelling of hybrid systems" In *Proceedings IFAC Symposium on AI in Real-Time Control*, Kuala Lumpur, Malaysia, September (1997).

[PRE96] Heinz A. Preisig. "A mathematical approach to discrete-event dynamic modelling of hybrid systems" *Computers Chemical Engeneering*, **20**:S1301–S1306, (1996).

[SON90] Eduardo D. Sontag. "Mathematical Control Theory: Deterministic Finite Dimensional Systems", volume 6 of *Texts in Applied Mathematics*. Springer-Verlag, New York, (1990).

[STI93] J. A. Stiver and Panos J. Antsaklis. "Extracting discrete event system models from hybrid control systems" In *Proceedings of the 1993 International Symposium on Intelligent Control*, pages 298–301, Chicago, Illinois, USA, August (1993).