**International Journal of Control**
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713393989

# Control of quantized systems based on discrete event models

P. P. H. H. Philips [a]; W. P. M. H. Heemels [b]; H. A. Preisig [b]; P. P. J. Van Den Bosch [b]
[a] Philips Centre for Manufacturing Techniques, Department of Mechatronics Research, P.P. Box 218, 5600 MD Eindhoven, The Netherlands. e-mail: patrick.philips@philips.com. [b] Eindhoven University of Technology, Department of Electrical Engineering, P.O. Box 513, 5600 MB Eindhoven, The Netherlands. e-mail: {h.a.preisig, p.p.j.v.d.bosch}@tue.nl.

**To cite this Article** Philips, P. P. H. H., Heemels, W. P. M. H., Preisig, H. A. and Bosch, P. P. J. Van Den(2003) 'Control of quantized systems based on discrete event models', International Journal of Control, 76: 3, 277 — 294
**To link to this Article: DOI:** 10.1080/0020717031000067402
**URL:** http://dx.doi.org/10.1080/0020717031000067402

Taylor & Francis
Taylor & Francis Group

# Control of quantized systems based on discrete event models

P. P. H. H. PHILIPS†, W. P. M. H. HEEMELS‡*, H. A. PREISIG‡ and P. P. J. VAN DEN BOSCH‡

This paper presents controller design methods for dynamical systems that are observed by discrete sensors. These sensors induce a partitioning of the state space and only this quantized information is available for the controller. The so-called 'quantized system' is modelled by a discrete-event model that serves as a basis for the controller design methods. However, instead of using solely the classical control methodologies for discrete-event systems as found in the literature, improvements are proposed by including additional information provided by the fact that the underlying plant is continuous by nature, such as continuity of the state trajectories and information on derivatives that holds for parts of the state space. The concept of discretely controlled invariant sets will play a crucial role in the development of control strategies and necessary and sufficient conditions for controlled invariance are presented. Also algorithms are included to compute the smallest and largest discretely controlled invariant sets with respect to a given set. All the techniques and computations are explicitly described by Boolean matrices and vectors and are ready for application. This is demonstrated for a benchmark problem of a two tank system.

## 1. Introduction

This paper is concerned with the control of a continuous plant that is only observed by discrete sensors and uses only a finite number of discrete inputs. Specifically, the continuous plant is described by a set of differential equations

$$\dot{x}(t) = f(x(t), u(t)) \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the state variable and $u(t)$ is the control input that only attains values from a discrete set, i.e. $u(t) \in \{\tilde{u}_1, \dots, \tilde{u}_k\} \subset \mathbb{R}^m$. For the case that $x(t) \in \mathbb{R}^2$, a trajectory evolves in a two-dimensional state space, see figure 1(*a*).

The discrete sensors are represented by a set of boundaries in the state space. For the $i$th component $x^i$ of the state vector $x$ the following values give the positions of the 'sensors'.

$$\beta_0^i < \beta_1^i < \cdots < \beta_{n_i}^i \qquad (n_i \geq 1)$$

These discrete sensors induce a partitioning of the state space into hypercubes, being rectangles in figure 1(*b*). The discrete measurements indicate at which boundaries and in which hypercube(s) the continuous state currently resides. This means that we only have coarse, discretized (quantized) information of the plant's state.

The reasons for studying dynamical systems observed by discrete sensors are twofold. The first motivation is the frequent occurrence of this type of sensors in practical situations, such as level sensors for measuring the height of a fluid in a vessel, encoders for determining the angular position of induction motors or (magnetic/optic) disc drives (Phillips and Tomizuka 1995, Heemels *et al.* 1995) and transportation systems, where the position of a vehicle is only known when certain markers have passed (De Bruin and van den Bosch 1998). A second reason comes from the situation in which a continuous plant is to be supervised by some programmable logical controller or computer program which uses discrete-state information. In such cases, the controller will often be a discrete-event system. Since a discrete-event controller cannot communicate with the system at a continuous level, an interface is required for the interconnection of the discrete-event controller and the continuous system, see figure 2(*a*). Note that it is not strictly necessary that discrete sensors are physically present; we can also have continuous sensors in our plant, but we only act on the basis of the (artificial) discrete information to realize the interface. In this figure, the discrete to analog conversion is denoted by 'DA', which is sometimes called the *injector* (Lunze *et al.* 1997). The block with 'AD' represents the analog to discrete conversion and is also called the *quantizer* (Lunze 1994). The interaction of discrete-event controller with a continuous system (figure 2(*a*)) results in a hybrid dynamical system (Antsaklis and Nerode 1998, Morse *et al.* 1999, Van der Schaft and Schumacher 2000) for which synthesis problems are extremely difficult to solve. The approach we will follow for dealing with this particular kind of hybrid systems is to model the continuous plant together with the interface as a discrete-event model as shown in figure 2(*b*). As a consequence, we have to study the interaction of two discrete-event models, which is less difficult than considering the overall hybrid system. In literature, these kind of systems are

Received June 2001. Revised October 2002.

* Author for correspondence. e-mail: w.p.m.h.heemels@tue.nl

† Philips Centre for Manufacturing Techniques, Department of Mechatronics Research, P.P. Box 218, 5600 MD Eindhoven, The Netherlands. e-mail: patrick.philips@philips.com

‡ Eindhoven University of Technology, Department of Electrical Engineering, P.O. Box 513, 5600 MB Eindhoven, The Netherlands. e-mail: {h.a.preisig, p.p.j.v.d.bosch}@tue.nl

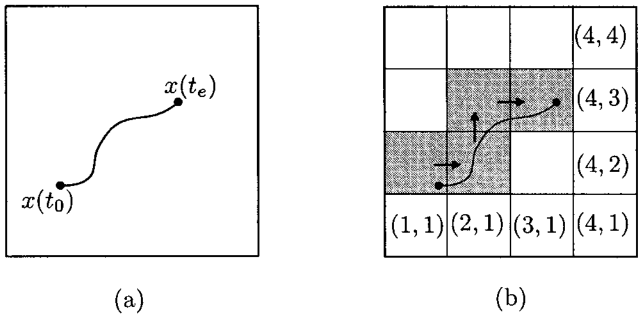

Figure 1. (*a*) A trajectory in two-dimensional state space; (*b*) discretization of the state space and the resulting discrete trajectory.

referred to as 'quantized systems' (Lunze 2000) and the resulting discrete-event models are known as 'discrete abstractions' (Lunze 1999) or 'qualitative models' (Lunze 1994).

The idea of abstracting discrete-event models from continuous systems is quite general and dates back to the seventies (see e.g. Kornoushenko 1975). Since that time various formalisms and methods have been proposed to solve the discrete abstraction problem. The differences between the various approaches are mainly caused by the description of the resulting discrete-event models (automata, timed automata, rectangular automata, Petri nets, (semi-)Markov processes or stochastic automata), the model of the continuous plant (continuous time or discrete time, linear or non-linear) and the choice of the discrete states as related to subsets of the continuous state space (rectangles, half spaces, or more general shapes). For instance, in Lunze (1994) a qualitative modelling approach using DE-automata for linear, discrete-time systems is presented and necessary and sufficient conditions are given for which the resulting automaton is deterministic. In Stursberg *et al.* (1997) two discretization methods are described which use discrete states based on rectangular sets and which result in timed automata. One method checks the derivatives in gridpoints and the other method uses forward/backward integration. In Raisch and O'Young (1998) it is explained how to obtain a discrete-event model from discrete-time systems by solving linear inequalities based on the system's inverse. This modelling method is only effective for affine models. In this paper the discrete-event modelling method examines the non-linear vector field of (1) on the boundaries separating discrete states to determine possible transitions. This technique has already been reported in Stiver and Antsaklis (1993) and Preisig (1996) and further developed in Preisig *et al.* (1997) for linear systems. The method has been adapted for the non-linear case in Bruinsma (1997) and Philips *et al.* (1997).

In spite of the fact that quantized systems are of interest from an industrial point of view and that the modelling, verification and fault detection aspects are well-established now (Kowalewski *et al.* 1999, Lunze 2000), control has received relatively little attention so far. One reason might be the complex control problems one has to tackle. Indeed, reachability and stabilization problems are highly non-trivial for non-linear control systems even in the case that no restrictions are present on the inputs and full information on the state of the system is available. In the setting of quantized systems similar problems have to be solved with the additional complication that only partial discrete state measurements and a finite number of control values are available. Another difficulty is caused by the fact that the resulting discrete-event representations are non-deterministic in the sense that from a discrete state with a given discrete input, several transitions are possible. As the result of a control action is not completely clear, the control design is far from easy.

Lunze studied the problem of qualitative feedback (Lunze 1995), where the control actions depend on the discrete state only. In Hsu (1985) an optimal control method is proposed on the basis of a deterministic 'cell-to-cell mapping' as an (incomplete) approximation

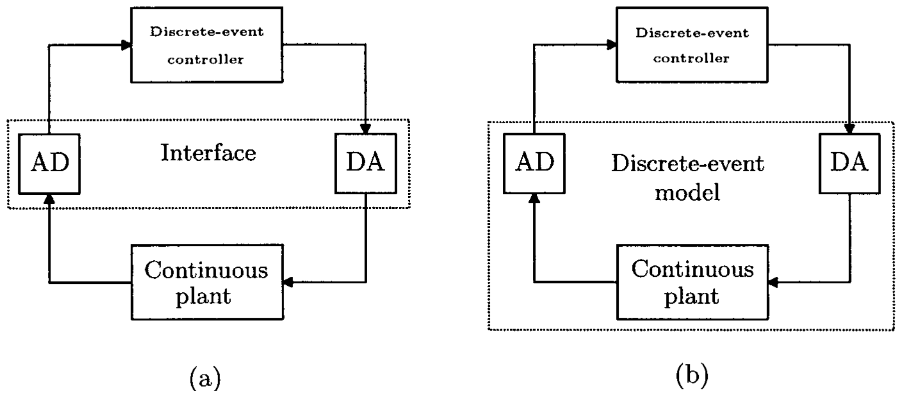

Figure 2. (*a*) Hybrid systems and (*b*) two discrete-event systems.

of the continuous dynamics (see also Papa *et al.* 1977). In addition to approaches like Cury *et al.* (1994) and Raisch and O'Young (1998) which design the supervisor only on the basis of the discrete approximation of the continuous plant (e.g. by using automata theory developed in Ramadge and Wonham (1987), we will also incorporate information provided by the continuous system in the proposed design methods. The fact that the underlying system is continuous allows control actions (e.g. preventing and moving inputs as will be seen below) that would not be possible for general discrete-event systems. As a consequence, the approach pursued here will have many advantageous features.

In this paper we consider the reachability problem (i.e. steering of the process from one place in the state space to another) and the stabilization problem (in addition to the reachability problem also keeping the process in the desired situation). Moreover, the methods are derived in such a manner that structured and computationally explicit controller design based on Boolean vector operations is possible. Moreover, no specialistic insight or knowledge of operators of the plant is needed, which is often the case with 'rule based' control strategies. Also, for complex systems, structured controller synthesis reduces the possibility that the controller is not prepared for all possible situations that can occur.

The paper is organized as follows. In § 2 it is briefly discussed how to obtain a discrete-event model from a continuous system described by differential equations. After a short introduction in Boolean representations and operators used in this paper, the problem statement is discussed in § 4. The key elements of the proposed controller design methods, two types of inputs and discretely controlled invariant sets, are introduced in § 5 and § 6, respectively. After this, two controller design methods are proposed in § 7 and § 8. The methods are illustrated by means of a two tank example. Next, it is shown how to adapt the proposed strategies for the case of delayed transition measurements in § 9. In § 10 we discuss the modifications of a certain standing assumption throughout the paper (made for explanatory reasons) in case it is violated. Finally, conclusions are drawn in § 11.

The following notations will be used throughout the paper. For a vector $x$ $x^i$ refers to the $i$th component of $x$. $A^{\mathrm{T}}$ denotes the transpose of the matrix $A$. For a set $V$ we denote the collection of all its subsets by $2^V$. The interior of a set $H \subseteq \mathbb{R}^n$ is denoted by $int(H)$. If $H$ is of the form $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_m, b_m]$ (possibly with $a_i = b_i$), then its relative interior $rint(H)$ is defined as $R_1 \times R_2 \times \cdots \times R_m$ with $R_i = \{a_i\}$ when $a_i = b_i$ and $R_i = (a_i, b_i)$ otherwise. For $T \subseteq \mathbb{R}$ the space $C^0(T, \mathbb{R}^m)$ denotes the collection of all continuous functions from $T$ and $\mathbb{R}^m$. Furthermore, $PC^0(T, \mathbb{R}^m)$ denotes the space of all piecewise continuous functions from $T$ to $\mathbb{R}^m$. If $T$ and $m$ are clear from the context, we sometimes write simply $PC^0$.

## 2. Discrete-event models of continuous systems

Consider the dynamical system described by the set of ordinary differential equations

$$\dot{x}(t) = f(x(t), u(t)) \tag{2}$$

where the input $u(t) \in \mathbb{R}^m$ takes values from a finite set† $\tilde{U} = \{\tilde{u}_1, \ldots, \tilde{u}_k\}$ and $x(t) \in \mathbb{R}^n$ is the state variable. It is assumed that (2) has a unique solution on $[0, \infty)$ for all initial conditions $x(0) = x_0$ and all piecewise continuous inputs $u \in PC^0$. Moreover, the function $f$ is continuous in the first argument $x$. Suppose that we have a number of sensors that detect a certain state variable $x^i$ reaching a certain value $\beta^i_j$. That means that, for each coordinate $x^i$, $(i = 1, \ldots, n)$, a set of boundaries is given

$$\beta^i_0 < \beta^i_1 < \cdots < \beta^i_{n_i} \qquad (n_i \geq 1) \tag{3}$$

such that, whenever $x^i(t) = \beta^i_j, j = 1, \ldots, n_i - 1$ a sensor will emit a signal indicating that the continuous state is at a specific boundary. The boundaries $\beta^i_0$ and $\beta^i_{n_i}$ are not detected but define the region of interest in the state space. The discretely observed and controlled system consisting of the continuous plant (2) and the boundaries (3) are called 'quantized systems' in the literature (see, for example, Lunze 2000). The boundaries (3) induce a natural partitioning of the state space into hypercubes (cells) (see figure 3).

Each hypercube (which is closed, i.e. the faces of the hypercube are included) can be labelled with a unique integer $\tilde{x} \in \tilde{X} = \{1, \ldots, p\}$, where $p = \prod_{i=1}^n n_i$. We refer to the integer $\tilde{x}$ as the discrete state associated with the hypercube $H_x(\tilde{x})$. Two discrete states $\tilde{x}_1$ and $\tilde{x}_2$ are called *adjacent*, if their corresponding hypercubes share an $(n-1)$-dimensional boundary plane $H_x(\tilde{x}_1) \cap H_x(\tilde{x}_2)$. The crossing of a boundary between

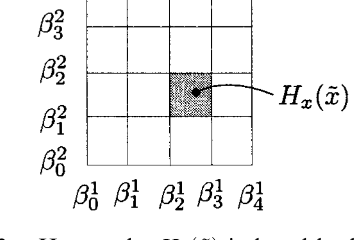

Figure 3. Hypercube $H_x(\tilde{x})$ induced by boundaries.

† For ease of explanation, $u(t)$ takes values from a discrete set. In case of a continuous input-space, discrete inputs can be obtained by a similar discretization as will be introduced for the state space (see also Philips 2001).

$\tilde{x}_1$ and $\tilde{x}_2$ is called a *discrete event* or *transition* and is denoted by $\tilde{x}_1 \to \tilde{x}_2$.

The continuous system (2) together with the set of boundaries (3) will be transformed into a discrete-event model given by the automaton (Sontag 1990) $\Sigma = (\tilde{X}, \tilde{U}, \phi)$ in which the finite set of states $\tilde{X}$ and the inputs $\tilde{U}$ are specified as above. The possibly non-deterministic transition function $\phi: \tilde{X} \times \tilde{U} \to 2^{\tilde{X}}$ has to be determined such that the set $\phi(\tilde{x}, \tilde{u}) \subseteq \tilde{X}$ contains all discrete states to which a transition from $\tilde{x}$ with input $\tilde{u}$ is possible. The loss of information is an intrinsic property of this operation (since the continuous state in the discrete abstraction is not exactly known). However, we require the discrete-event model to satisfy the following condition for all pairs of discrete states $\tilde{x}_1$ and $\tilde{x}_2$, and discrete input $\tilde{u} \in \tilde{U}$.

**Condition 1:** *$\tilde{x}_2 \in \phi(\tilde{x}_1, \tilde{u})$ (i.e. the transition $\tilde{x}_1 \to \tilde{x}_2$ is admissible under $\tilde{u}$) if and only if $\tilde{x}_1$ and $\tilde{x}_2$ are adjacent and there exists a solution $x$ of* (2) *with $u(t) = \tilde{u}$, starting from some initial condition $x(0) = x_0 \in H_x(\tilde{x}_1)$ such that $x(t') \in int(H_x(\tilde{x}_2))$ for some $t' > 0$ and $x(t) \in H_x(\tilde{x}_1) \cup H_x(\tilde{x}_2)$ for $0 \le t \le t'$.*

For explanatory reasons we restrict ourselves for the moment to transitions between states that are adjacent. The necessary modifications and extensions to include also transitions to non-adjacent discrete states will be treated in § 10.

Note that the 'if'-part of this condition complies with the notion of *completeness* (Lunze 1994), which is crucial as it allows the transfer of the results obtained for the discrete-event model to the underlying continuous system. To be precise, if a property holds for the discrete-event model, then completeness implies that it also holds for the continuous system. In the context of fault diagnosis this was shown in Förstner and Lunze (2001). The 'only if'-part guarantees that a certain type of spurious solution is not included in the model in the sense that each (single) transition of the discrete model can also be generated by the continuous system. This is related to the fundamental concept of *soundness* as is used in computer science. The following result has been proven in Philips (2001) and is based on a result by Nagumo (1942) (see also Blanchini (1999, Theorem 3.1)).

**Condition 2:** *The discrete states $\tilde{x}_1$ and $\tilde{x}_2$ are adjacent such that $\{x \in \mathbb{R}^n \mid x^r = \beta^r_j\}$ is the separating hyperplane and $x \in H_x(\tilde{x}_1) \Rightarrow x^r \le \beta^r_j$ and $x \in H_x(\tilde{x}_2) \Rightarrow x^r \ge \beta^r_j$.*

**Proposition 1** (Philips 2001)**:** *Consider two discrete states $\tilde{x}_1$ and $\tilde{x}_2$ satisfying Condition 2. Then $\tilde{x}_2 \in \phi(\tilde{x}_1, \tilde{u})$ if and only if*

$$\exists x_0 \in H_x(\tilde{x}_1) \cap H_x(\tilde{x}_2) \qquad \text{for which } f^r(x_0, \tilde{u}_0) > 0 \quad (4)$$

A computer program can automatically generate the transition function by checking (4) via the value of the maximization problem $\max_{x_0 \in H_x(\tilde{x}_1) \cap H_x(\tilde{x}_2)} f^r(x_0, \tilde{u})$ for all adjacent pairs of states and all relevant inputs. For more details on the discrete abstraction as described in this section, the interested reader is referred to Philips (2001).

## 3. Boolean representations and operators

For computational and explanatory reasons, it is convenient to represent the discrete state $\tilde{x} = i \in \{1, \ldots, p\}$ by the Boolean vector $\hat{x} = [0, \ldots, 0, 1, 0, \ldots, 0]^{\mathrm{T}} \in \{0, 1\}^p$, where the 1 is at the $i$th position, and a set of discrete states $\tilde{X}_1 \subseteq \tilde{X}$ by $\hat{x}_1$ defined by

$$\hat{x}_1^i = \begin{cases} 1 & \text{if } i \in \tilde{X}_1 \\ 0 & \text{if } i \notin \tilde{X}_1 \end{cases}$$

Various operations acting on sets can be computed easily by using the binary operations $\oplus$ ('or'), $\otimes$ ('and'), and $\ominus$ on $\{0, 1\}$ defined by $0 \oplus 0 = 0 \otimes 0 = 0 \otimes 1 = 1 \otimes 0 = 0 \ominus 0 = 0 \ominus 1 = 1 \ominus 1 = 0$, and $0 \oplus 1 = 1 \oplus 0 = 1 \oplus 1 = 1 \otimes 1 = 1 \ominus 0 = 1$.

**Union:** The union of two sets $\tilde{Z} = \tilde{X}_1 \cup \tilde{X}_2$ is computed by $\hat{z} = \hat{x}_1 \oplus \hat{x}_2$ in which $\oplus$ has to be interpreted componentwise, i.e. $\hat{z}^i = \hat{x}_1^i \oplus \hat{x}_2^i$.

**Intersection:** The intersection $\tilde{Z} = \tilde{X}_1 \cap \tilde{X}_2$ is computed by $\hat{z} = \hat{x}_1 \otimes \hat{x}_2$, with $\hat{z}^i = \hat{x}_1^i \otimes \hat{x}_2^i$. Furthermore, note that $\tilde{X}_1 \cap \tilde{X}_2 \neq \emptyset \Leftrightarrow \hat{x}_1^{\mathrm{T}} \hat{x}_2 \neq 0$.

**Difference:** The set difference $\tilde{Z} = \tilde{X}_1 \backslash \tilde{X}_2$ is computed by $\hat{z} = \hat{x}_1 \ominus \hat{x}_2$, with $\hat{z}^i = \hat{x}_1^i \ominus \hat{x}_2^i$.

Boolean matrices also appear to be useful in the context of automata $\Sigma$ as they can be visualized by a directed graph (digraph), which in turn can be represented by so-called adjacency matrices. Given the automation $\Sigma = (\tilde{X}, \tilde{U}, \phi)$ then for each input $\tilde{u} \in \tilde{U}$ the adjacency matrix $A_{\tilde{u}} \in \{0, 1\}^{p \times p}$ is a Boolean matrix with

$$(a_{\tilde{u}})_{ij} = \begin{cases} 1 & \text{if } i \in \phi(j, \tilde{u}) \\ 0 & \text{if } i \notin \phi(j, \tilde{u}) \end{cases}$$

The following operations on Boolean matrices $A$ and $B$ of appropriate dimensions turn out to be convenient in the sequel.

**Boolean matrix (or vector) multiplication:** $C = AB$ is defined as $c_{ij} = \bigoplus_{k=1}^{l} a_{ik} \otimes b_{kj} = (a_{i1} \otimes b_{1j}) \oplus (a_{i2} \otimes b_{2j} \oplus \cdots \oplus (a_{il} \otimes b_{lj})$.

**Boolean matrix 'or':** $C = A \oplus B$ is defined as $c_{ij} = a_{ij} \oplus b_{ij}$.

**Boolean matrix 'and':** $C = A \otimes B$ is defined as $c_{ij} = a_{ij} \otimes b_{ij}$.

Note that for the state $\tilde{x}_1$, represented in the Boolean vector domain as $\hat{x}_1$, and the discrete input $\tilde{u}$, the set of states $\tilde{X}_2$ to which transitions are possible from $\tilde{x}_1$ with input $\tilde{u}$ is computed by $\hat{x}_2 - A_{\tilde{u}}\hat{x}_1$. Furthermore, $\hat{x}_3 = A_{\tilde{u}}^{\mathrm{T}}\hat{x}_1$ is the set of states $\tilde{X}_3$ from which transitions are possible to $\tilde{x}_1$ using input $\tilde{u}$.

## 4. Control goals

The discrete-event model extracted from the continuous system will now be used for control purposes. Note that the only available information for the controller (see also figure 2) is the knowledge of the hypercube $H_x(\tilde{x})$ that contains the current continuous state and moreover, when the continuous state is at a boundary, the corresponding sensor emits a signal. The controller will apply a piecewise constant input to the system based on this information.

**Definition 1:** Let the system (2) with boundaries (3) and discrete inputs $\tilde{U}$ be given. A solution trajectory $x \in C^0[0, T]$ of (2) is generated with an input $u \in PC[0, T]$ satisfying $u(t) \in \tilde{U}$ for all $t \in [0, T]$ is called a *discretely controlled trajectory*.

The objective of the paper is to propose a systematic method to solve the following control problems.

**The reachability problem:** Given the system (2), a set of boundaries (3), the discrete initial state $\tilde{x}_0$, the set of target states $\tilde{X}_e$, and a set of discrete inputs $\tilde{U}$. Find a controller that realizes for any continuous initial state $x_0 \in H_x(\tilde{x}_0)$ a discretely controlled trajectory $x$ (with $x(0) = x_0$) such that $x(t) \in int(\bigcap_{\tilde{x}\in\tilde{X}_e} H_x(\tilde{x}))$ for some $t \geq 0$.

The reachability problem is the problem of controlling the trajectory from an initial discrete state to a desired final set of discrete states (more precisely, to the corresponding hypercubes in the state space).

**The stabilization problem:** Given the system (2), a set of boundaries (3), the discrete initial state $\tilde{x}_0$, the set of target states $\tilde{X}_e$, and a set of discrete inputs $\tilde{U}$. Find a controller that realizes for any continuous initial state $x_0 \in H_x(\tilde{x}_0)$ a discretely controlled trajectory $x$ ($x(0) = x_0$) with $x(t) \in int(\bigcup_{\tilde{x}\in\tilde{X}_e} H_x(\tilde{x}))$ for some $t \geq 0$ and that satisfies the following property: for all $t_e > 0$ with $x(t_e) \in int(\bigcup_{\tilde{x}\in\tilde{X}_e} H_x(\tilde{x}))$ it holds that $x(t) \in \bigcup_{\tilde{x}\in\tilde{X}_e} H_x(\tilde{x})$ for all $t \geq t_e$ as well.

The stabilization problem is the problem of controlling the state trajectory from an initial discrete state to a desired set of discrete states and preventing the trajectory from leaving the corresponding set of hypercubes after the state has entered the interior.

**Remark 1:** Note that this notion of stability implies the more natural stability for which only the existence of $t_e$ such that $x(t) \in \bigcup_{\tilde{x}\in\tilde{X}_e} H_x(\tilde{x})$ for all $t \geq t_e$ is required, as e.g. for qualitative stability (Lunz 1995). Realizing the latter definition of stability on the basis of the automaton is equivalent to the definition as stated here.

We will propose systematic controller design methodologies under an appropriate assumption for now. The necessary modifications needed in the case of violating this assumption are discussed at the end of the paper.

**Assumption 1:** *The generated discretely controlled trajectories x will never reach a point belonging to more than two hypercubes, i.e. for all $t \in \mathbb{R}_+$ it holds that $x(t) \notin \bigcap_{i\in I} H_x(\tilde{x}_i)$ for all index sets I with three or more elements.*

In practice, this means that the implemented control strategies are valid as long as the continuous state will not lie on the intersection of two different boundaries in (3). This is detected when two sensors will emit a signal simultaneously. The reason to adopt this assumption is for the sake of readability as the computation and main ideas that form the basis of the control strategies can be adjusted in case of violation of Assumption 1 as will be outlined at the end of the paper. This requires the inclusion of transitions between states that are not adjacent, which was excluded in the modelling of the quantized system is § 2.

The following concepts are used for specifying the control strategies. We introduce these notions first in a non-precise manner, after which we formalize them in the next section.

**Preventing input:** An input that prevents a transition from happening, see figure 4(*a*). A preventing input is applied immediately when a boundary is reached. This means that we assume that it is exactly measured when the state trajectory reaches a boundary between two hypercubes, and moreover, that instantaneous control action is possible.

**Moving input:** An input for which it is certain that the state trajectory moves towards a particular boundary plane.[†] Applying this input guarantees for all initial states $x_0$ in the hypercube $H_x(\tilde{x})$ that the trajectory (started in $x_0$) leaves $H_x(\tilde{x})$ in a point closer to the specified boundary plane than the initial state $x_0 \in H_x(\tilde{x})$, see figure 4(*b*).

[†] Trajectories starting at the specified boundary plane will cross it.

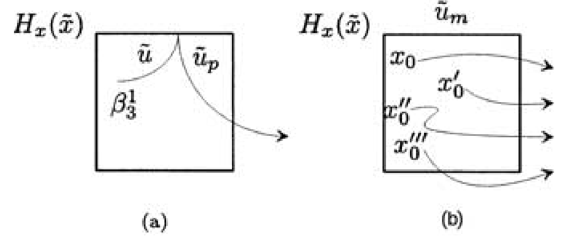

Figure 4. (*a*) Preventing input $\tilde{u}_p$. (*b*) Moving input $\tilde{u}_m$.

## 5. Preventing and moving inputs

### 5.1. *Preventing inputs*

Let the quantized system be given by the continuous plant (2), the set of boundaries (3), and the finite set of inputs $U$.

**Definition 2** (preventing input)**:** The input $\tilde{u} \in \tilde{U}$ is *preventing* for the transition $\tilde{x}_1 \to \tilde{x}_2$ if this transition is not possible with the specific input $\tilde{u}$, i.e. $\tilde{x}_2 \notin \phi(\tilde{x}_1, \tilde{u})$.

This means that $\tilde{u}$ can be used to prevent the state trajectory from going from $H_x(\tilde{x}_1)$ to $H_x(\tilde{x}_2)$.

Preventing inputs can be determined directly from the discrete-event model of the continuous system (2). By using the Boolean vector notation $\hat{x}$ for a discrete state $x$ and the set of adjacency matrices $\{A_{\tilde{u}}\}_{\tilde{u}\in\tilde{U}}$ for representing the discrete-event model, the computation of preventing inputs is straightforward.

**Proposition 2:** *Given a discrete state $\tilde{x}_1$, the input $\tilde{u} \in \tilde{U}$ is preventing for the transition $\tilde{x}_1 \to \tilde{x}_2$ if and only if $\hat{x}_2^{\mathrm{T}} A_{\tilde{u}} \hat{x}_1 = 0$.*

**Proof:** The set of states $\tilde{X}_3$ that can be reached from $\tilde{x}_1$ with input $\tilde{u}$ is represented by $\hat{x}_3 := A_{\tilde{u}}\hat{x}_1$. The discrete state $\tilde{x}_2$ is not an element of $\tilde{X}_3$ iff $\hat{x}_2^{\mathrm{T}}\hat{x}_3 = 0$. Hence, the relation $\hat{x}_2^{\mathrm{T}} A_{\tilde{u}} \hat{x}_1 = 0$ is equivalent to $\tilde{x}_2 \notin \phi(\tilde{x}_1, \tilde{u})$. □

There exists an input $\tilde{u} \in \tilde{U} = \{\tilde{u}_1, \ldots, \tilde{u}_k\}$ that is preventing for $\tilde{x}_1 \to \tilde{x}_2$ if and only if for at least one of the adjacency matrices $A_{\tilde{u}_l}$, $l = 1, \ldots, k$ the $ij$th element is zero, where $i$ and $j$ are the integer representations of $\tilde{x}_2$ and $\tilde{x}_1$, respectively. As a consequence, the Boolean matrix $S_{\tilde{U}} \in \{0,1\}^{p\times p}$ defined by

$$S_{\tilde{U}} = A_{\tilde{u}_1} \otimes A_{\tilde{u}_2} \otimes \cdots \otimes A_{\tilde{u}_k} \bigotimes_{\tilde{u}\in\tilde{U}} A_{\tilde{u}} \tag{5}$$

satisfies the property that $s_{ij} = 0$ is equivalent to $(a_{\tilde{u}})_{ij}$ for some $\tilde{u} \in \tilde{U}$. Hence, there exists at least one input $\tilde{u} \in \tilde{U}$ that is preventing for $\tilde{x}_1 \to \tilde{x}_2$ if and only if $\hat{x}_2^{\mathrm{T}} S_{\tilde{U}} \hat{x}_1 = 0$. Furthermore, $\hat{x}_2 = S_{\tilde{U}}\hat{x}_1$ contains all neighbouring states of $\tilde{x}_1$ to which a transition from $\tilde{x}_1$ *cannot be prevented by any input* of the set $\tilde{U}$.

### 5.2. *Moving inputs*

From the description of moving inputs it is clear that an input is moving for a transition $\tilde{x}_1 \to \tilde{x}_2$ if with this input for all trajectories $x$ starting in $H_x(\tilde{x}_1)$ the distance to the boundary plane $H_x(\tilde{x}_1) \cap H_x(\tilde{x}_2)$ is decreased when $x$ leaves $H_x(\tilde{x}_1)$. From figure 4(*b*) it can be seen that an input resulting in the trajectory starting in $x_0''$ is moving according to this description. However, to facilitate computations (specifically, to avoid the integration and to make automatic verification possible) the definition of moving inputs will be a somewhat stronger notion that will not include the inputs corresponding to trajectories like the one emanating from $x_0''$ in figure 4(*b*).

**Definition 3** (moving input)**:** Let two discrete states $\tilde{x}_1$ and $\tilde{x}_2$ satisfying Condition 2 be given. An input $\tilde{u} \in \tilde{U}$ is *moving* for the transition $\tilde{x}_1 \to \tilde{x}_2$ if

$$f^r(x, \tilde{u}) > 0 \qquad \forall x \in H_x(\tilde{x}_1) \tag{6}$$

This means that the $r$th coordinate of $x$ increases monotonically as $f^r(x, \tilde{u}) > 0$ and therefore moves towards the boundary between $H_x(\tilde{x}_1)$ and $H_x(\tilde{x}_2)$. As $H_x(\tilde{x}_1)$ is closed and $f$ continuous in $x$, there actually exists an $\varepsilon > 0$ such that $f^r(x, \tilde{u}) > \varepsilon$ for all $x \in H_x(\tilde{x}_1)$.

Fix input $\tilde{u} \in \tilde{U}$. All transitions for which $\tilde{u}$ is a moving input are represented by the *direction matrix* $D_{\tilde{u}} \in \{0,1\}^{p\times p}$ given by

$$(d_{\tilde{u}})_{ij} = \begin{cases} 1 & \text{if } \tilde{u} \text{ is moving for } j \to i \\ 0 & \text{else} \end{cases}$$

The computation of $D_{\tilde{u}}$ can be automated by using optimizations for checking (6). By using $\lambda_{\min} := \min_{x\in H_x(\tilde{x}_1)} f^r(x, \tilde{u})$ and $\lambda_{\max} := \max_{x\in H_x(\tilde{x}_1)} f^r(x, u)$ we have that $f^r(x, \tilde{u}) > 0$, $\forall x \in H_x(\tilde{x}_1) \Leftrightarrow \lambda_{\min} > 0$ and $f^r(x, \tilde{u}) < 0$, $\forall x \in H_x(\tilde{x}_1) \Leftrightarrow \lambda_{\max} < 0$. Note that $(d_{\tilde{u}})_{ij} \Rightarrow (d_{\tilde{u}})_{ji} = 0$. If the continuous system (2) is linear, the direction matrix can be computed efficiently, since all information needed can be extracted from the adjacency matrices $\{A_{\tilde{u}}\}$ (Philips 2001).

Since $\hat{x}_2 := D_{\tilde{u}}\hat{x}_1$ represents the set of all discrete states adjacent to $\tilde{x}_1$ for which the input $\tilde{x}$ is moving, it is clear that the input $\tilde{u}$ is moving for the transition $\tilde{x}_1 \to \tilde{x}_2$ if and only if $\hat{x}_2^{\mathrm{T}} D_{\tilde{u}} \hat{x}_1 \neq 0$. If $\hat{x}_3$ represents the set $\tilde{X}_3$ then $\hat{x}_3^{\mathrm{T}} D_{\tilde{u}} \hat{x}_1 \neq 0$ implies the existence of an $\tilde{x}_3' \in \tilde{X}_3$ such that the input $\tilde{u}$ is moving for $\tilde{x}_1 \to \tilde{x}_3'$. In this case we say that $\tilde{u}$ is moving for the set $\tilde{X}_3$ from $\tilde{x}_1$.

Furthermore, for a set of inputs $\tilde{U}$ the direction matrix $D$ is given by

$$D = \bigoplus_{\tilde{u}\in\tilde{U}} D_{\tilde{u}}$$

Obviously, there exists an input $\tilde{u} \in \tilde{U}$ which is moving for the transition $\tilde{x}_1 \to \tilde{x}_2$ if and only if $\hat{x}_2^{\mathrm{T}} D \hat{x}_1 \neq 0$.

## 6. Discretely controlled invariant sets

Since the proposed controller design methods will depend heavily on the construction of discretely controlled invariant sets, this notion is formalized first.

**Definition 4** (Discretely controlled invariant set)**:** The set $\Omega := \bigcup_{\tilde{x} \in \tilde{Z}} H_x(\tilde{x})$ (and the corresponding set of discrete states $\tilde{Z}$) is called a *discretely controlled invariant set*, if for each initial state $x_0 \in \Omega$, a discretely controlled trajectory $x$ exists, with $x(0) = x_0$ and $x(t) \in \Omega$ for all $t \geq 0$.†

The following result relies on Assumption 1. If this assumption does not hold, extensions are required as discussed in § 10. Note that the results on discretely controlled invariant sets in this section are also true in the context of automata.

**Proposition 3:** *The set $\tilde{Z} \subseteq \tilde{X}$ is discretely controlled invariant using the set of inputs $\tilde{V} \subseteq \tilde{U}$ iff (in Boolean vector notation) $S_{\tilde{V}} \hat{z} \ominus \hat{z} = 0$, where $S_{\tilde{V}} = \bigotimes_{\tilde{u} \in \tilde{V}} A_{\tilde{u}}$.*

**Proof:** Recall that $\hat{z}' := S_{\tilde{V}} \hat{z}$ represents the set of states to which transitions from states in the set $\tilde{Z}$ cannot be avoided. If $\tilde{Z}'$ is a subset of $\tilde{Z}$ (or equivalently $\hat{z}' \ominus \hat{z} = 0$), then transitions to states outside the set $\tilde{Z}$ can be prevented with inputs from the set $\tilde{V}$, hence any trajectory starting in $\bigcup_{\tilde{x} \in \tilde{Z}} H_x(\tilde{x})$ can be kept from leaving this set. If $S_{\tilde{V}} \hat{z} \ominus \hat{z} \neq 0$, then clearly transitions to states outside $\tilde{Z}$ cannot be prevented, and thus $\tilde{Z}$ is not discretely controlled invariant. □

**Definition 5:** Given the set $\tilde{Z}$, then $\tilde{X}_{inv}$ is the *largest discretely controlled invariant set* in $\tilde{Z}$ if

(i) $\tilde{X}_{inv} \subseteq \tilde{Z}$,
(ii) $\tilde{X}_{inv}$ is discretely controlled invariant,
(iii) If $\tilde{X}'_{inv}$ satisfies (i) and (ii) then $\tilde{X}'_{inv} \subseteq \tilde{X}_{inv}$.

To compute the largest controlled invariant set, a method can be used that is related to the one for linear control systems (see e.g. Wonham 1979). In the proof also the existence of the largest control invariant set is shown.

**Proposition 4:** *Given a set $\tilde{Z} \subseteq \tilde{X}$, the largest discretely controlled invariant set $\tilde{X}_{inv}$ in $\tilde{Z}$ using the collection of inputs $\tilde{V}$ exists and is computed from the sequence of sets given by $\tilde{Z} = \tilde{X}_0 \supset \tilde{X}_1 \supset \tilde{X}_2 \supset \cdots \supset \tilde{X}_j = X_{inv}$, with*

(1) $\hat{x}_0 := \hat{z}$,
(2) $\hat{x}_{k+1} = \hat{x}_k \ominus S_{\tilde{V}}^{\mathrm{T}}(\hat{x} \ominus \hat{x}_k)$

*and* $j = \min\{k \in \mathbb{N} \mid \hat{x}_{k+1} = \hat{x}_k\}$.

**Proof:** (i) Note that from step (2) we have that $\tilde{X}_{k+1} \subseteq \tilde{X}_k$, since only states are removed from $\tilde{X}_k$ to obtain $\tilde{X}_{k+1}$. Hence $\tilde{X}_k \subseteq \tilde{Z}$ and (i) is satisfied.

(ii) $\tilde{X}_j = \tilde{X}_{j+1}$ is equivalent to $\hat{x}_j = \hat{x}_j \ominus S_{\tilde{V}}^{\mathrm{T}}(\hat{x} \ominus \hat{x}_j)$ implying that $0 = \hat{x}_j^{\mathrm{T}} S_{\tilde{V}}^{\mathrm{T}}(\hat{x} \ominus \hat{x}_j) = (S_{\tilde{V}} \hat{x}_j)^{\mathrm{T}}(\hat{x} \ominus \hat{x}_j)$. This implies that the intersection of the sets represented by $S_{\tilde{V}} \hat{x}_j$ and $\hat{x} \ominus \hat{x}_j$ is empty. Hence $S_{\tilde{V}} \hat{x}_j \ominus \hat{x}_j = 0$, which means that $\tilde{X}_j$ is controlled invariant.

To prove (iii) we first claim that for any set $Z' \subseteq \tilde{Z}$ it holds that $X'_k \subseteq \tilde{X}_k$ for all $k$ where $\{\tilde{X}'_k\}$ is the sequence of sets generated by the algorithm as given in the proposition with $\tilde{X}'_0 := \tilde{Z}'$. Starting with $\tilde{X}'_0 := \tilde{Z}'$ and $\tilde{X}_0 := \tilde{Z}$ we have that $\tilde{X}'_0 \subseteq \tilde{X}_0$, so the claim holds for $k = 0$. We proceed by induction, i.e. suppose $\tilde{X}'_k \subseteq \tilde{X}_k$ is true. Now we will prove that if $\tilde{x}_1 \in \tilde{X}'_k \cap \tilde{X}_k$ is removed from $\tilde{X}_k$ on the basis of the algorithm, then it is also removed from $\tilde{X}'_k$ implying that $\tilde{X}'_{k+1} \subseteq \tilde{X}_{k+1}$ holds as well. Suppose that $\tilde{x}_1 \in \tilde{X}'_k \cap \tilde{X}_k$ has to be removed from $\tilde{X}_k$, i.e. $\hat{x}_1^{\mathrm{T}} S_{\tilde{V}}^{\mathrm{T}}(\hat{x} \ominus \hat{x}_k) = 1$. Since $\tilde{X}'_k \subseteq \tilde{X}_k$ this implies that $\hat{x}_1^{\mathrm{T}} S_{\tilde{V}}^{\mathrm{T}}(\hat{x} \ominus \hat{x}'_k) = 1$. Consequently, $\tilde{x}_1$ will be removed from $\tilde{X}'_k$ to obtain $\tilde{X}'_{k+1}$. Hence, $\tilde{X}'_{k+1} \subseteq \tilde{X}_{k+1}$.

To prove (iii) take an arbitrary $\tilde{X}'_{inv}$ satisfying (i) and (ii). Put $\tilde{Z}'$ in the claim above equal to $\tilde{X}'_{inv}$ and observe that controlled invariance of $\tilde{X}'_{inv}$ yields that $\tilde{X}'_k = \tilde{X}'_{inv}$ for all $k$. Since $\tilde{X}'_{inv} \subseteq \tilde{X}_k$ for all $k$ (due to the claim) the result follows as $\tilde{X}_k$ is equal to $\tilde{X}_{inv}$ for $k = j$. □

Since $\tilde{X}_0$ has a finite number of elements and only states are removed, the algorithm terminates in a finite number of steps.

**Definition 6:** Given the set $\tilde{Z}$, then $\tilde{X}_{inv}$ is the smallest discretely controlled invariant set containing $\tilde{Z}$, if

(i) $\tilde{Z} \subseteq \tilde{X}_{inv}$,
(ii) $\tilde{X}_{inv}$ is discretely controlled invariant,
(iii) If $\tilde{X}'_{inv}$ satisfies (i) and (ii) then $\tilde{X}_{inv} \subseteq \tilde{X}'_{inv}$.

**Proposition 5:** *Given a set $\tilde{Z} \subseteq \tilde{X}$, the smallest discretely controlled invariant set $\tilde{X}_{inv}$ for the set of inputs $\tilde{V}$ containing $\tilde{Z}$ exists and is computed from the sequence of sets given by $\tilde{Z} = \tilde{X}_0 \subset \tilde{X}_1 \subset \tilde{X}_2 \subset \cdots \subset \tilde{X}_j \tilde{X}_{inv}$, with*

(1) $\hat{x}_0 := \hat{z}$,
(2) $\hat{x}_{k+1} = \hat{x}_k \oplus S_{\tilde{V}} \hat{x}_k$

*and* $j = \min\{k \in \mathbb{N} \mid \hat{x}_{k+1} = \hat{x}_k\}$.

† Strictly speaking, we should call the set 'positively invariant', since we are only concerned with $t \geq 0$.

**Proof:** (i) From step (1) we have $\tilde{X}_k \subseteq \tilde{X}_{k+1}$ as only states are added to obtain $\tilde{X}_{k+1}$ from $\tilde{X}_k$. Hence, $\tilde{Z} \subseteq \tilde{X}_k$ for all $k$ as $\tilde{X}_0 = \tilde{Z}$ and (i) is satisfied.

(ii) $\tilde{X}_j = \tilde{X}_{j+1}$ is equivalent to $\hat{x}_j = \hat{x}_j \oplus S_{\tilde{V}}\hat{x}_j$ and consequently also to $S_{\tilde{V}}\hat{x}_j \ominus \hat{x}_j = 0$ (since $S_{\tilde{V}}\hat{x}_j$ must represent a subset of $\tilde{X}_j$). This proves the controlled invariance of $\tilde{X}_j$.

(iii) We claim that for any set $\tilde{Z}' \supseteq \tilde{Z}$ it holds that $\tilde{X}'_k \supseteq \tilde{X}_k$ for all $k$, where $\{\tilde{X}'_k\}$ is generated by step (2) of the algorithm with $\tilde{X}'_0 = \tilde{Z}'$. Starting with $\tilde{X}'_0 := \tilde{Z}'$ and $\tilde{X}_0 := \tilde{Z}$ we have that $X'_0 \supseteq \tilde{X}_0$, proving that the claim holds for $k = 0$. To proceed by induction, assume that $\tilde{X}'_k \supseteq \tilde{X}_k$, which means that we can write $\hat{x}'_k = \hat{x}_k \oplus \Delta\hat{x}'_k$, with $\Delta\hat{x}'_k$ representing the set difference $X'_k \backslash \tilde{X}_k$. Now, it follows that $\hat{x}_{k+1} = \hat{x}_k \oplus S_{\tilde{V}}\hat{x}_k$ and $\hat{x}'_{k+1} = \hat{x}_k \oplus \Delta\hat{x}'_k \oplus S_{\tilde{V}}(\hat{x}_k \oplus \Delta\hat{x}'_k) = \hat{x}_k \oplus \Delta\hat{x}'_k \oplus S_{\tilde{V}}\hat{x}_k \oplus S_{\tilde{V}}\Delta\hat{x}'_k = \hat{x}_{k+1} \oplus \Delta\hat{x}'_k \oplus S_{\tilde{V}}\Delta\hat{x}'_k$. Hence, $\tilde{X}'_{k+1} \supseteq \tilde{X}_{k+1}$. Similar reasoning as for the proof of (iii) for Proposition 4 gives the desired result. □

Since, starting from $\tilde{X}_0$, only states are added and the total number of discrete states in $\tilde{X}$ is finite, the algorithm is finite as well.

**Remark 2:** Note that the smallest controlled invariant subspace containing a given subspace does not exist for linear time-invariant systems as the intersection of controlled invariant subspaces is not necessarily controlled invariant (Wonham 1979). For the quantized systems as studied here the existence of the smallest discretely controlled invariant set is guaranteed on the basis of the proof of Proposition 5.

For the sake of the reachability problem it is convenient to have an extension of the concept of controlled invariant set.

**Definition 7** *($\Gamma$-controlled invariant set)*: *Let the sets $\tilde{Z} \subseteq \tilde{X}$ and $\tilde{Z}_1 \subseteq \tilde{X}$ satisfying $\tilde{Z}_1 \subseteq \tilde{Z}$ be given with $\Omega := \bigcup_{\tilde{x}\in\tilde{Z}} H_x(\tilde{x})$ and $\Gamma := \bigcup_{\tilde{x}\in\tilde{Z}_1} H_x(\tilde{x})$ the corresponding regions in the (continuous) state space. $\Omega$ is called $\Gamma$-controlled invariant (and $\tilde{Z}$ is $\tilde{Z}_1$-controlled invariant), if for each initial state $x_0 \in \Omega$, a discretely controlled trajectory $x$ (with $x(0) = x_0$) exists, such that either $x(t) \in \Omega$ for all $t \geq 0$, or $x(t_e) \in \Gamma$, where $t_e = \inf\{t > 0 \,|\, x(t) \notin \Omega\}$.*

Loosely speaking, $\Omega$ is $\Gamma$-controlled invariant, if the continuous state trajectory leaves the set via $\Gamma \subseteq \Omega$ in case the state cannot be controlled to stay in $\Omega$ (see figure 5). Note that discretely controlled invariance is equivalent to $\emptyset$-controlled invariance.

**Proposition 6:** *Let $\tilde{Z}_1 \subseteq \tilde{Z} \subseteq \tilde{X}$ be given. $\tilde{Z}$ is $\tilde{Z}_1$-controlled invariant for the set of inputs $\tilde{V} \subseteq \tilde{U}$ iff (in Boolean vector notation) $S_{\tilde{V}}(\hat{z} \ominus \hat{z}_1) \ominus (\hat{z}) = 0$, where $S_{\tilde{V}} = \bigotimes_{\tilde{u}\in\tilde{V}} A_{\tilde{u}}$.*

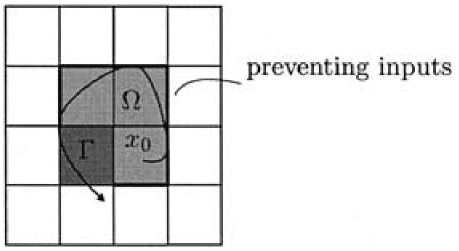

Figure 5. A $\Gamma$-controlled invariant set.

**Proof:** From all the discrete states in $\tilde{Z}$ that are not in $\tilde{Z}_1$ (i.e. $\tilde{Z}\backslash\tilde{Z}_1$ represented by $\hat{z} \ominus \hat{z}_1$) transitions cannot be prevented to the set $\hat{z}' = S_{\tilde{V}}(\hat{z} \ominus \hat{z}_1)$. If $\hat{z}' \ominus (\hat{z}) = 0$, then $\tilde{Z}' \subseteq \tilde{Z}$ and we can only leave $\tilde{Z}$ from its subset $\tilde{Z}_1$. If $S_{\tilde{V}}(\hat{z} \ominus \hat{z}_1) \ominus (\hat{z}) \neq 0$, then there exists discrete states in $\tilde{Z}\backslash\tilde{Z}_1$ from which a transition to a state outside $\tilde{Z}$ cannot be prevented and hence, $\tilde{Z}$ is not $\tilde{Z}_1$-controlled invariant. □

## 7. 'Forceable state-transition' control strategy

From the transition function $\phi$ it can be seen that a transition from one state to another is possible with a particular input. However, due to non-determinism, often also other transitions are possible with this input. As a consequence, it is not known in advance which of the possible transitions actually will happen, since this depends on the unknown continuous state. The idea of the first controller synthesis is to find a way to guarantee that a transition from one discrete state to another is possible and all other transitions can be ruled out.

**Definition 8** (forceable transition): For two adjacent discrete states $\tilde{x}_1$ and $\tilde{x}_2$ the transition $\tilde{x}_1 \to \tilde{x}_2$ is called forceable, if from each initial state $x_0 \in H_x(\tilde{x}_1)$ there exists a discretely controlled trajectory $x$ (with $(x(0) = x_0)$ for which there exists a $t_1 > 0$ such that $x(t) \in H_x(\tilde{x}_1) \cup H_x(\tilde{x}_2)$ for all $t \in [0, t_1]$ and $x(t_1) \in int(H_x(\tilde{x}_2))$.

The forceable transitions define a directed graph on the set of discrete states that we call the *forceability graph*. It is obvious that the solvability of the reachability follows easily from the forceability graph.

**Proposition 7:** *The reachability problem is solvable for $\tilde{x}_0$ to $\tilde{x}_e$, if the force-ability graph contains a path from $\tilde{x}_0$ to $\tilde{x}_e$.*

Unfortunately, the construction of the forceability graph is very difficult in general. In the next section we will provide a sufficient condition for forceability and we show how it can be systematically checked by using the direction matrices $D_{\tilde{u}}$ and the adjacency matrices $A_{\tilde{u}}$. This leads to a construction of a subgraph of the forceability graph. Hence, if there is a path from the initial discrete state to the target discrete state in the subgraph then we can conclude that the problem is solvable and a control strategy can be obtained that realizes the desired path. However, if such path does not exist the reachability problem may still have a solution.

### 7.1. *Sufficient condition for the forceability of a transition*

Using the concept of moving and preventing inputs, we can formulate the following proposition.

**Proposition 8:** *Let $\tilde{x}_1$ and $\tilde{x}_2$ be adjacent discrete states. The transition $\tilde{x}_1 \to \tilde{x}_2$ is* forceable, *if for each transition $\tilde{x}_1 \to \tilde{x}_3 \neq \tilde{x}_2$ to a state $\tilde{x}_3$ to $\tilde{x}_1$, there exists an input that is preventing for $\tilde{x}_1 \to \tilde{x}_3$ among the inputs that are moving for $\tilde{x}_1 \to \tilde{x}_2$.*

**Proof:** Without loss of generality we assume that Condition 2 holds. Suppose that the set of inputs that are moving for the transition $\tilde{x}_1 \to \tilde{x}_2$ is given by $\tilde{U}_m$. As $H_x(\tilde{x}_1)$ is closed and bounded and $\tilde{U}_m$ has a finite number of elements, it holds that $\min_{\tilde{u}\in\tilde{U}_m} \min_{\tilde{u}\in H_x(\tilde{x}_1)} f^r(x,\tilde{u}) \geq \varepsilon$ for some $\varepsilon > 0$. Hence, if we pick any $\tilde{u}_1 \to \tilde{U}_m$ it is guaranteed that the continuous state $x(t)$ moves towards the common boundary $H_x(\tilde{x}_1) \cap H_x(\tilde{x}_2)$. The only phenomenon that might obstruct $x(t)$ from crossing the boundary $H_x(\tilde{x}_1) \cap H_x(\tilde{x}_2)$, is that another boundary $H_x(\tilde{x}_1) \cap H_x(\tilde{x}_3)$, $x_3 \neq \tilde{x}_2$ is reached first. In this case the transition $\tilde{x}_1 \to \tilde{x}_3$ will be prevented by an input $\tilde{u}_2 \in \tilde{U}_m$. Since $\tilde{u}_2 \in \tilde{U}_m$, it is still guaranteed that $x(t)$ keeps on moving towards the desired boundary. By construction, it is certain that the transition $\tilde{x}_1 \to \tilde{x}_2$ will occur, and all other transitions $\tilde{x}_1 \to \tilde{x}_3$, $\tilde{x}_3 \neq \tilde{x}_2$, will not (and thus $\tilde{x}_1 \to \tilde{x}_2$ is forceable). □

### 7.2. *Forceability graph*

Since the condition in Proposition 8 is only sufficient, it will result in a subgraph of the forceability graph only. However, the construction can be obtained in an automated manner. Let for a set of discrete inputs $\tilde{U} = \{\tilde{u}_1, \ldots, \tilde{u}_k\}$, the transition matrices $\{A_{\tilde{u}}\}_{\tilde{u}\in\tilde{U}}$ and the direction matrices $\{D_{\tilde{u}}\}_{\tilde{u}\in\tilde{U}}$ be given. The conditions of Proposition 8 guaranteeing the transition $\tilde{x}_1 \to \tilde{x}_2$ to be forceable can be checked by performing the following steps:

(1) Determine all inputs for which $\tilde{x}_1$ is moving for $\tilde{x}_1 \to \tilde{x}_2$, i.e. compute

$$\tilde{U}_m = \{\tilde{u} \in \tilde{U} \mid \hat{x}_2^{\mathrm{T}} D_{\tilde{u}} \hat{x}_1 \neq 0\} \tag{7}$$

(2) Check if, transitions to discrete states other than $\tilde{x}_2$ can be prevented by some input $\tilde{u} \in \tilde{U}_m$, i.e. verify that

$$S_{\tilde{U}_m} \hat{x}_1 \ominus \hat{x}_2 = 0$$

with $S_{\tilde{U}_m} := \bigotimes_{\tilde{u}\in\tilde{U}_m} A_{\tilde{u}}$.

By executing these computations for all transitions $\tilde{x}_1 \to \tilde{x}_2$, the forceability matrix $P$ is constructed

$$\hat{x}_2 P \hat{x}_1 = p_{ij} = \begin{cases} 1, & \text{if } S_{\tilde{U}_m}\hat{x}_1 \ominus \hat{x}_2 = 0, \\ & \quad \tilde{U}_m = \{\tilde{u} \in \tilde{U} \mid \hat{x}_2^{\mathrm{T}} D_{\tilde{u}} \hat{x}_1 \neq 0\} \\ 0, & \text{else} \end{cases}$$

where $j$ and $i$ are the integer presentations of $\tilde{x}_1$ and $\tilde{x}_2$, respectively.

The Boolean vector $\hat{x}_2 := P\hat{x}_1$ represents a set that contains discrete states to which a transition from $\tilde{x}_1$ is forceable. The forceability matrix $P$ is the adjacency matrix of a subgraph of the forceability graph. If we label each edge in the forceability graph with the set of inputs $\tilde{U}_m$ as defined in (7) then we have the *labelled forceability graph*. The labels indicate which inputs can be chosen to prevent undesirable transitions and still guarantee movement towards a desired transition.

### 7.3. *Control strategy*

The reachability problem requires to discretely control the state from the initial discrete state $\tilde{x}_0$ to the target set of discrete states $\tilde{X}_e$ $(\tilde{x}_0 \notin \tilde{X}_e)$. For this we have to search for a path in the forceability subgraph from $\tilde{x}_0$ to $\tilde{X}_e$. The following algorithm provides all these paths with minimal length (if any) by first propagating backward (from $\tilde{X}_e$) and then forward (from $\tilde{x}_0$).

*Step* 1. Set $\hat{x}_1 := \hat{x}_e$ and iterate

$$\hat{x}_{i+1} = P^{\mathrm{T}} \hat{x}_i$$

until $\tilde{x}_0 \in \tilde{X}_{i+1}$, i.e. $\hat{x}_0^{\mathrm{T}} \hat{x}_{i+1} = 1$. Set $r = i + 1$. Now we have computed all forceable paths of length $r - 1$ that lead to $\tilde{X}_e$, and at least one of these paths starts from $\tilde{x}_0$, see figure 6(*a*).

*Step* 2. Set $\hat{z}_1 = \hat{x}_0$ and for $i = 1, \ldots, r - 1$ compute

$$\hat{z}_{i+1} = P\hat{z}_i \otimes \hat{x}_{r-i}$$

So, the set $\tilde{Z}_i$ (represented by the Boolean vector $\hat{z}_i$) is the intersection of the forceable paths

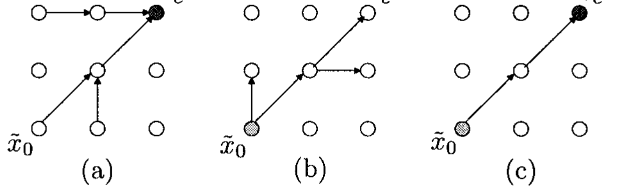

Figure 6. (*a*) Forceable paths of length 2 ending in $\tilde{x}_e$, (*b*) forceable paths starting in $\tilde{x}_0$, and (*c*) forceable paths starting in $\tilde{x}_0$ and ending in $\tilde{x}_e$.

starting from $\tilde{x}_0$ of length $i-1$, see figure 6(*b*) and the forceable paths that are $r-i-1$ steps away from $\tilde{X}_e$, see figure 6(*a*). The resulting collection $\{\tilde{Z}_1, \tilde{Z}_2, \ldots, \tilde{Z}_r\}$ contains the discrete states on forceable paths of length $r-1$ originating from $\tilde{x}_0$ and ending in $\tilde{X}_e$, see figure 6(*c*). It is always possible to go from $\tilde{Z}_i$ to $\tilde{Z}_{i+1}$ such that, starting from $\tilde{x}_0$ we reach $\tilde{X}_e$.

The actual control algorithm consists of choosing control inputs in such a way that the continuous state trajectory evolves from $\tilde{Z}_1$ to $\tilde{Z}_2$, then from $\tilde{Z}_2$ to $\tilde{Z}_3$, etc. until you reach the target set $\tilde{Z}_r$. This is done by selecting moving inputs $\tilde{u} \in \tilde{U}_m$ from the labelled forceability graph such that a transition from $\tilde{x}_1 \in \tilde{Z}_i$ to $\tilde{x}_2 \in \tilde{Z}_{i+1}$ is forced (initially one starts with $i = 1$ and $\tilde{x}_1 = \tilde{x}_0$). Each time a sensor detects that the continuous state reaches a boundary $H_x(\tilde{x}_1) \cap H_x(\tilde{x}_3)$ with $x_3 \notin \tilde{Z}_{i+1}$ the particular transition is prevented by choosing a moving input $\tilde{u} \in \tilde{U}_m$ such that $\tilde{x}_3 \notin \phi(\tilde{x}_1, \tilde{u})$, i.e. $\hat{x}_3^{\mathrm{T}} A_{\tilde{u}} \hat{x}_1 = 0$. Whenever a sensor detects that the continuous state reaches the boundary $H_x(\tilde{x}_1) \cap H_x(\tilde{x}_2)$ with the discrete state $\tilde{x}_2 \in \tilde{Z}_{i+1}$, an input is chosen that is preventing for $\tilde{x}_2 \to \tilde{x}_1$ and moving for a discrete state $\tilde{x}_4$ in $\tilde{Z}_{i+2}$ such that $\tilde{x}_2 \to \tilde{x}_4$ is forceable (note that such an input must exist by construction). Note that the input only changes whenever transitions have to be prevented and hence the controllers are *synchronous* with the discrete events (see e.g. Lunze *et al.* 2001).

**Remark 3:** The above control strategy works under the validity of Assumption 1. The following modification is allowed to reduce the possibility of this assumption being violated as will be explained by the example in § 7.4. The modification require a parameter $\tau > 0$ representing a delay time in changing the input in some cases.

A moving input $\tilde{u}$ has been chosen from $\tilde{x}_1 \in \tilde{Z}_i$ to $\tilde{x}_2 \in \tilde{Z}_{i+1}$. Subsequently, one of the following control action is taken:

- Each time a sensor detects that the continuous state reaches a boundary $H_x(\tilde{x}_1) \cap H_x(\tilde{x}_3)$ with $\tilde{x}_3 \notin \tilde{Z}_{i+1}$ the particular transition is prevented by choosing a moving input $\tilde{u} \in \tilde{U}_m$ such that $\tilde{x}_3 \notin \phi(\tilde{x}_1, \tilde{u})$, i.e. $\hat{x}_3^{\mathrm{T}} A_{\tilde{u}} \hat{x}_1 = 0$.
- Whenever a sensor detects that the continuous state reaches the boundary $H_x(\tilde{x}_1) \cap H_x(\tilde{x}_2)$ with the discrete state $\tilde{x}_2 \in \tilde{Z}_{i+1}$, the current moving input is maintained for $\tau > 0$ time units to make sure that the boundary actually is crossed. After $\tau$ time units, an input is chosen that is moving for a discrete state $\tilde{x}_4$ in $\tilde{Z}_{i+2}$.

If in the last situation a boundary of an undesired discrete state $\tilde{x}_5 \notin \tilde{Z}_{i+2}$ is reached before $\tau$ time units have expired then a preventing input for $\tilde{x}_2 \to \tilde{x}_5$, which is moving for $\tilde{x}_2 \to \tilde{x}_4$ is directly applied.

Note that $\tau = 0$ complies with the control strategy described earlier. For $\tau > 0$ the controllers are not synchronous with the discrete events any more. Taking $\tau > 0$ has the advantage that the system's state will not run over the boundary $H_x(\tilde{x}_1) \cap H_x(\tilde{x}_2)$ thereby reducing the probability that Assumption 1 is violated. Indeed, if $\tau = 0$ a preventing input is chosen that might keep the state at the boundary. If in this situation a new discrete event occurs, Assumption 1 is violated and the controller scheme as outlined here is not guaranteed to work and necessary modifications have to be taken as described in § 10. A typical example where this situation might occur is the closing of a valve that keeps the level in a tank at a certain level exactly equal to the sensor level, see the example below.

**Remark 4:** The usage of the forceability subgraph is exploited for solving the reachability problem. To solve the stabilization problem one should solve the reachability problem for the largest discretely controlled invariant set contained in $X_e$. If this set is empty, then the stabilization problem cannot be solved. In this case, the 'best' one can do is solving the reachability problem for the smallest discretely controlled invariant set containing $\tilde{X}_e$.

### 7.4. *Example: two tank system*

Consider the two tank system depicted in figure 7, which is often used as a kind of benchmark problem in the context of quantized systems (Lunze *et al.* 1997, 2001).

Here, it consists of two communicating tanks, which are connected through a small pipe. Both tanks can be filled by controlling the valves in the respective feed pipes ($s_1, s_2$). The pipe between the tubes can be blocked by means of switch $s_3$. Only the second tank has a drain $s_4$. The input $\tilde{u} = (s_1, s_2, s_3, s_4)$ consists of the vector of switch positions (open or closed) for the four valves controlling the flows in the system. The input set $\tilde{U}$

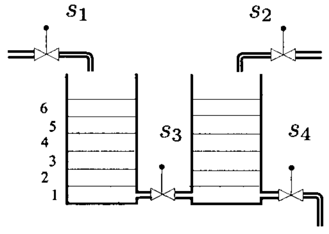

Figure 7. The two tank system.

has $2^4 = 16$ elements. The state vector $x = [x^1, x^2]^{\mathrm{T}}$ is given by the water levels in each tank and is governed by

$$\dot{x}^1 = s_1 \frac{4\hat{m}_1}{\pi \rho D^2} - s_3 \frac{d^2}{D^2}\sqrt{2g}\Psi(x^1 - x^2)$$

$$\dot{x}^2 = s_3 \frac{d^2}{D^2}\sqrt{2g}\Psi(x^1 - x^2) + s_1 \frac{4\hat{m}_2}{\pi \rho D^2} - s_4 \frac{d^2}{D^2}\sqrt{2g}\Psi(x^2)$$

with $\Psi(x) := \operatorname{sign}(x)\sqrt{|x|}$ and where $\hat{m}_1, \hat{m}_2$ are the mass flows of the feed pipes, $\rho$ is the density of water, $g$ is the earth's gravitational acceleration and $D, d$ are the diameters of the tanks and pipes, respectively. Each tank is divided into six parts (states) defined by the levels $(\beta^i_j)$: 0, 0.01, 0.1, 0.2, 0.3, 0.4, and 0.5 $[m]$.

After computing the transition and direction matrices $\{A_{\tilde{u}}\}$ and $\{D_{\tilde{u}}\}$ we obtain the forceability graph as depicted in figure 8, where the discrete states are represented by their so-called tuple representation for convenience. This means that a discrete state is described by a tuple $\tilde{x} = (\tilde{x}^1, \tilde{x}^2)$ with the components $\tilde{x}^1, \tilde{x}^2$ representing the discrete state corresponding to the components of $x^1$ and $x^2$, respectively.

The goal is to steer the system from initial state $\tilde{x}_0 = (6, 6)$ to $\tilde{x}_e = (4, 6)$. From the forceability (sub)-graph it can be seen that a possible path is given by the sequence $(6,6) \rightarrow (6,5) \rightarrow (6,4) \rightarrow (5,4) \rightarrow (5,3) \rightarrow (4,3) \rightarrow (4,4) \rightarrow (4,5) \rightarrow (4,6)$ and consequently, a solution exists to this reachability problem. To implement the control strategy, $\tau$ is chosen equal to 0.1 [s] (see Remark 3). A resulting time trajectory from $\tilde{x}_0 = (6, 6)$ to $\tilde{x}_e = (4, 6)$ is shown in figure 9.

The vertical dotted lines in figure 9 indicate the time instants at which the input changes. Starting with $s_4$ open and all other switches closed, after approximately 21 s the boundary $\beta^2_4$ is reached and $\tau = 0.1$ s later, at $t_1$ the controller opens $s_3$ (and closes $s_4$) in order to lower the level in the first tank. As a result, the level of the second tank starts to rise, which would cause $x_2$ to cross boundary $\beta^2_4$. This is prevented immediately by opening $s_4$ at time $t_2$. At $t_3 - \tau$ discrete state $(5, 4)$ is reached and $s_3$ is closed ($s_4$ is still open) at time $t_3$ such that the level

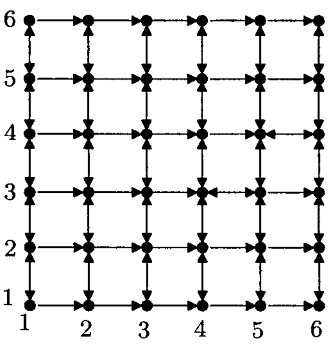

Figure 8. Forceability graph for the two tank system.

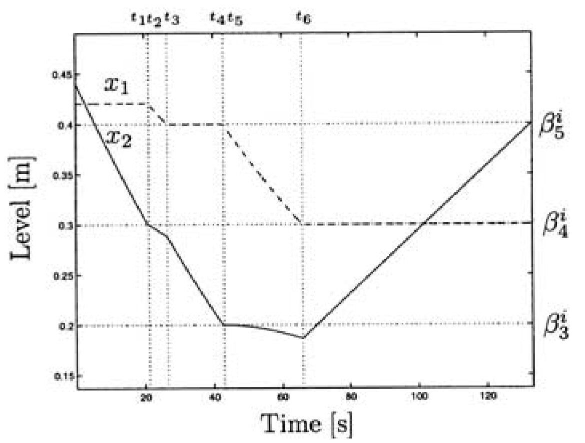

Figure 9. Trajectory for $\tilde{x}_0 = (6, 6)$ and $\tilde{x}_e = (4, 6)$.

in the second tank can drop in order to reach $(5, 3)$. Note that the delay of $\tau$ seconds prevents that the state stays at the boundary as discussed in Remark 3. Then $s_4$ is switched off and $s_3$ is switched on at $t_4$, allowing the level in the first tank to drop again. However, since the level in the second tank starts to rise, the transition $(5, 3) \rightarrow (5, 4)$ has to be prevented by opening $s_4$ again at $t_5$. Finally, at $t_6$ the valves $s_3$ and $s_4$ are turned off and $s_2$ is turned on such that the level in the second tank rises until the desired discrete state is reached. In summary, the following sequence is generated: $(6,6) \xrightarrow{0001} (6,5) \xrightarrow{0001} (6,4) \underset{0011}{\xrightarrow{0010}} (6,5) \xrightarrow{0011} (5,4) \xrightarrow{0001} (5,3) \xrightarrow{0010} (5,4) \xrightarrow{0011} (4,3) \xrightarrow{0100} (4,4) \xrightarrow{0100} (4,5) \xrightarrow{0100} (4,6)$. Here $(6,6) \xrightarrow{0001} (6,5)$ denotes the transition $(6,6) \rightarrow (6,5)$ with input $\tilde{u} = (0,0,0,1)$, and $(6,4) \overset{0010}{\longmapsto} (6,5) \xrightarrow{0011} (5,4)$ indicates that the transition $(6,4) \longrightarrow (5,4)$ is realized after preventing $(6,4) \xrightarrow{0010} (6,5)$ with input $\tilde{u}(0,0,1,1)$. From the simulation it follows that Assumption 1 is not violated.

## 8. Forceable 'set-transition' control strategy

The previous controller design method is restrictive in the sense that for the existence of a forceable transition it is necessary that *all* undesired transitions have to be preventable. The controller design method discussed in this section aims at relaxing this requirement. Even if a transition to a certain discrete state cannot be prevented, this will not be a problem when from the new state a transition to a different design state is possible. Instead of forcing transitions between states, we now aim at forcing transitions between sets. The key idea is to compute a finite sequence $\{\tilde{Z}_1, \ldots, \tilde{Z}_r\}$ of nested $\tilde{Z}_{i-1}$-controlled invariant sets, where $\tilde{Z}_1$ is the target set. Each set contains its predecessor, i.e. $\tilde{Z}_{i-1} \subset \tilde{Z}_i$. The last set $\tilde{Z}_r$ contains all relevant initial states (one can also proceed until the final set contains the whole discrete space $\tilde{X}$, such that a global controller can be realized). Furthermore, as these sets are controlled invariant with respect to its predecessor, the state trajectory can only move to the predecessor of the current set in which it is contained. This predecessor is smaller and 'closer' to the target set.

For a given set of relevant initial states $\tilde{X}_0$ and the set of target states $\tilde{X}_e$, the following procedure is performed to obtain the nested sequence of sets.

1. **(Initialization)** For the reachability problem, set $\hat{z}_1 := \hat{x}_a := \hat{x}_e$. For the stabilization problem we compute the largest controlled invariant set $\tilde{X}_{inv}$ for the set of inputs $\tilde{U}$ contained in $\tilde{X}_e$. If there is no non-trivial solution, then the stabilizing problem cannot be solved. The 'best' alternative is to compute the smallest controlled invariant set $\tilde{X}_{inv}$ for the set of inputs $\tilde{U}$ containing the target set of discrete states $\tilde{X}_e$. We set $\hat{z}_1 := \hat{x}_a := \hat{x}_{inv}$ ($\tilde{Z}_1 := \tilde{X}_a := \tilde{X}_{inv}$). For both the reachability and the stabilization problem, take $k := 1$.
2. **(Moving neighbours)** Next, the set $\tilde{X}_b$ is computed containing all discrete states $\tilde{x}'_b \notin \tilde{X}_a$ for which there exist moving inputs for some transition $\tilde{x}'_b \to \tilde{x}'_a \in \tilde{X}_a$, i.e.
$$\hat{x}_b = (D^{\mathrm{T}}\hat{x}_a) \ominus \hat{x}_a$$
3. **(Check controlled invariance)** Check for each single discrete state $\tilde{x}_b \in \tilde{X}_b$ whether transitions to states $\tilde{x} \notin \tilde{X}_a \cup \tilde{X}_b$ can be prevented by at least one of the moving inputs for some $\tilde{x}'_b \to \tilde{x}'_a \in \tilde{X}_a$. That is compute
$$\hat{x}_c = S_{\tilde{U}_m}\hat{x}'_b \ominus (\hat{x}'_b \oplus \hat{x}_a)$$
with $\tilde{U}_m = \{\tilde{u} \in \tilde{U} \mid \hat{x}_a^{\mathrm{T}} D_{\tilde{u}} \hat{x}'_b \neq 0\}$. Consider the following two cases.
   (i) **(Is controlled invariant)** If $\hat{x}_c = 0$ for all $\tilde{x}'_b \in \tilde{X}_b$, then we set $\tilde{Z}_{k+1} := \tilde{X}_a \cup \tilde{X}_b$, which is $\tilde{X}_a$-controlled invariant by Proposition 6. Note that this implies by induction that it is actually $\tilde{Z}_1$-controlled invariant.† If the set of initial states $\tilde{X}_0 \subseteq \tilde{Z}_{k+1}$, then the algorithm has successfully finished. Otherwise, rename $\hat{x}_a := \hat{x}_a \oplus \hat{x}_b$, set $k := k+1$ and execute Step 2 again.
   (ii) **(Make controlled invariant)** If $\hat{x}_c \neq 0$, then this implies that we cannot prevent some transition $\tilde{x}'_b \to \tilde{x}'_c \in \tilde{X}_c$ by any input $\tilde{u} \in \tilde{U}_m$. The discrete state $\tilde{x}'_b$ is removed from $\tilde{X}_b$, i.e. put $\hat{x}_b := \hat{x}_b \ominus \hat{x}'_b$. If $\hat{x}_b = \hat{x}_a$, then no further improvement is possible and the strategy fails. Otherwise, perform Step 3 again for the updated $\tilde{X}_b$.

† In case $\tilde{Z}_1$ is controlled invariant (as, for example, in the stabilization problem), then a $\tilde{Z}_1$-controlled invariant set containing $\tilde{Z}_1$ is controlled invariant as well.

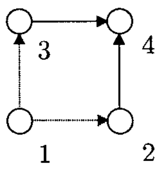

Figure 10. Forceable states versus forceable sets.

If finished successfully, the procedure results in a sequence of nested, $\tilde{Z}_1$-controlled invariant sets, $\tilde{x}_0 \in \tilde{Z}_r \supset \tilde{Z}_{r-1} \supset \cdots \supset \tilde{Z}_2 \supset \tilde{Z}_1$ (even with $\tilde{Z}_i$ being $\tilde{Z}_{i-1}$-controlled invariant).

The connection with the first controller design method can be explained by using the directed graph presented in figure 10.

In this figure, the solid arrows indicate forceable transitions whereas the dotted arrows represent transitions that cannot be forced but for which moving inputs exist. As can be seen, there is no forceable path from discrete state 1 to state 4. Hence, the first controller design method would fail to solve the reachability problem (from 1 to 4) for this situation. However, if we are certain that a transition from discrete state 1 to one of the states in the set $\{2, 3\}$ will happen, then the transition between the sets $\{1\}$ and $\{2, 3\}$ is forceable. From the set $\{2, 3\}$ we then finally can force a transition to the discrete state 4. Consequently, the methodology discussed in this section might yield a solution in this case.

### 8.1. *Control strategy*

The aim of the controller strategy is to let the continuous state $x(t)$ evolve from one $\tilde{Z}_1$-controlled invariant set $\tilde{Z}_i$ to the smaller set $\tilde{Z}_{i-1}$ which is 'closer' to the target set $\tilde{Z}_1$. This is repeated until the set of target discrete states $\tilde{Z}_1$ (equal to $\tilde{X}_e$ in the reachability problem) is entered. Initially, take $\tilde{x}_1 := \tilde{x}_0 \in \tilde{Z}_r$ and take $i = r$. Now one has to compute the set of inputs $\tilde{U}_m$ which are moving to the set $\tilde{Z}_{i-1}$. This set is given by $\tilde{U}_m = \{\tilde{u} \in \tilde{U} \mid \hat{z}_{i-1}^{\mathrm{T}} D_{\tilde{u}} \hat{x}_0 \neq 0\}$. One of the following actions has to be taken by the controller.

- Whenever it is observed that a transition is about to occur to a discrete state $\tilde{x}_2 \notin \tilde{Z}_i$ (i.e. the situation would be worse after the transition), then an input is chosen from $\tilde{U}_m$ which is preventing for the transition $\tilde{x}_1 \to \tilde{x}_2$. That is choose $\tilde{u} \in \tilde{U}_m$ for which $\hat{x}_2^{\mathrm{T}} A_{\tilde{u}} \hat{x}_1 = 0$.
- If $\tilde{x}_2 \in \tilde{Z}_i \backslash \tilde{Z}_{i-1}$ (i.e. the 'distance' to the target set stays 'equal'), then the current input is maintained for $\tau$ time units to ensure that the transition actu-

ally occurs. After $\tau$ time units an input is chosen that is moving for some transition $\tilde{x}_2 \to x_3 \in \tilde{Z}_{i-1}$.

- In case $\tilde{x}_2 \in \tilde{Z}_{i-1}$, the current moving input will also be maintained for $\tau$ time units to guarantee that the set $\tilde{Z}_{i-1}$ is entered, which is 'closer' to the target set $\tilde{Z}_1$. Next, the strategy aims at moving from $\tilde{Z}_{i-1}$ to $\tilde{Z}_{i-2}$ by switching to an input that is moving for $\tilde{x}_2 \to \tilde{Z}_{i-2}$ (i.e. take $i := i-1$ and $\tilde{x}_1 := \tilde{x}_2$ and repeat the above procedure again).

If in the last two situations a boundary of an undesired discrete state $\tilde{x}_4$ (i.e. outside $\tilde{Z}_i$ or outside $\tilde{Z}_{i-1}$, respectively) is reached before $\tau$ time units have elapsed, then a preventing input for $\tilde{x}_2 \to \tilde{x}_4$ is directly applied which is also moving for $\tilde{x}_2 \to \tilde{Z}_{i-1}$ or $\tilde{x}_2 \to \tilde{Z}_{i-2}$, respectively. The parameter $\tau > 0$, can be chosen freely. In case $\tau = \infty$, the input only changes whenever transitions have to be prevented.

By these control actions, we ensure that the trajectory can only leave the set $\tilde{Z}_i$ to a set $\tilde{Z}_k$ with $k < i$. This strategy is performed repeatedly until $\tilde{Z}_1$ is reached.

**Remark 5:** If a control action triggered by a time event (the elapse of a time period of $\tau$ units) is not included ($\tau = 0$ or $\tau = \infty$), the actions would be synchronous with the discrete transitions. However, in this case the following problems might occur. Suppose we are moving with input $\tilde{u}$ from $\tilde{x}_1$ to $\tilde{x}_2$ and the boundary $H_x(\tilde{x}_1) \cap H_x(\tilde{x}_2)$ is reached. If $\tau = \infty$ we enter the interior of $H_x(\tilde{x}_2)$ and the controller waits until the system arrives at a boundary $H_x(\tilde{x}_2) \cap H_x(\tilde{x}_3)$ for some $\tilde{x}_3$. However, it might happen that we never reach such a boundary. For instance, if the system has an equilibrium $x_{eq} \in H_x(\tilde{x}_2)$ for input $\tilde{u}\,(f(x_{eq}, \tilde{u}) = 0$ is possible as $\tilde{u}$ is not necessarily a moving input for $\tilde{x}_2$), the continuous state may never leave $\tilde{x}_2$ and no further improvement will be realized. Second, if $\tau = 0$ (meaning that we directly change the input when we reach the boundary $H_x(\tilde{x}_1) \cap H_x(\tilde{x}_2)$), then the newly selected moving input can steer the trajectory immediately back to $\tilde{x}_1$, because this input is not necessarily preventing for the transition $\tilde{x}_2 \in \tilde{x}_1$. These phenomena are prevented by selecting $0 < \tau < \infty$ as can be easily seen.

## 8.2. *Troublesome situation*

In general, it is not possible to guarantee that the state will evolve to a smaller set. Although moving inputs are used to try to let the state evolve in the direction of the desired state, it is not certain that the state will actually reach a smaller controlled invariant set as can be shown by looking at the following example.

**Example 1:** Consider the discrete states $\tilde{x}_1$, $\tilde{x}_2$, $\tilde{x}_3$, and $\tilde{x}_4$ and their corresponding hypercubes, depicted

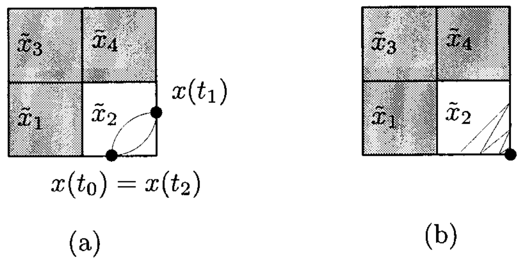

Figure 11. (*a*) A cyclic trajectory; (*b*) trajectory evolving to cornerpoint.

in figure 11. The objective is to reach the set $\{\tilde{x}_1, \tilde{x}_3, \tilde{x}_4\}$ from $\tilde{x}_2$. Suppose we start in the continuous state $x(t_0) \in H_x(\tilde{x}_2)$ with an input $\tilde{u}$ that is moving for $\tilde{x}_2 \to \tilde{x}_4$. However, in order to prevent the state from leaving $H_x(\tilde{x}_2)$, at time $t_1$ a preventing input is applied that is moving for $\tilde{x}_2 \to \tilde{x}_1$. It might be the case that at time $t_2$ the point $x(t_0)$ is reached again and the input $\tilde{u}$ is applied again, since it is preventing and moving. Thus, a cyclic trajectory arises that prevents us from reaching the set $\{x_1, \tilde{x}_3, \tilde{x}_4\}$, see figure 11(*a*). In a similar way it might be the case that the trajectory ends up in the lower right corner of $H_x(\tilde{x}_2)$, see figure 11(*b*). Note that in this case Assumption 1 is violated and the method needs to be extended as discussed at the end of the paper.

As a consequence of trying to solve the non-linear control problem in its full generality and the lack of knowledge due to the discrete sensors, no rigorous proof can be given that the continuous trajectory will actually reach a smaller controlled invariant set. However, it is guaranteed that the situation will never become worse. A possible (heuristic) solution for overcoming a problem as in figure 11(*a*) is to keep on moving towards the same state in the desired set (e.g. $\tilde{x}_4$ in the example) as long as possible. Note that both phenomena discussed in Example 1 can be observed from the discrete measurement and precautions can be taken (for instance, by choosing other preventing inputs). However, for the first control method based on forceable transitions a guarantee was given.

## 8.3. *Example: two tank system*

Again, we consider the two tank system discussed in §7.4. For this example the aim is to reach the discrete state $\tilde{x}_e = (4, 6)$ from the initial state $\tilde{x}_0 = (6, 6)$. If we compute the $\{\tilde{x}_e\}$-controlled invariant sets by exploiting the procedure discussed in this section then this results in 9 sets (including the target state $\tilde{x}_e$), $\{\tilde{Z}_k\}_{k=1,\ldots,9}$. It turns out that all these sets are actually controlled invariant because $\{\tilde{x}_e\}$ is controlled invariant. In figure 12 the sets $\tilde{Z}_{k+1} \backslash \tilde{Z}_k$ are depicted. Each such set has the same colour or shading in the figure and consist of discrete

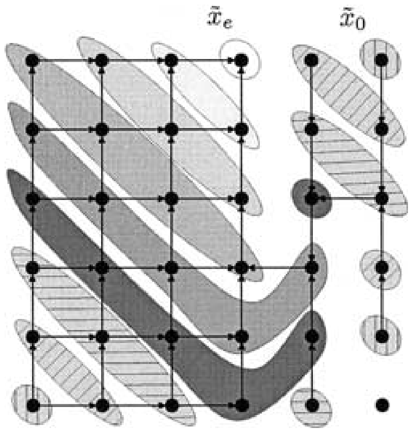

Figure 12. Digraph of all paths for which moving inputs exist ending in $x_e$.

states which are $k$ steps remote from the target state $\tilde{x}_e$, where $k = 1, \ldots, 8$. In figure 12 also all paths are shown for which moving inputs exist and that will lead to the target state $\tilde{x}_e = (4, 6)$ within 8 steps. Starting in the initial discrete state $\tilde{x}_0 = (6, 6) \in \tilde{Z}_9$ which is 8 transitions remote from the target state $\tilde{x}_e$, it is certain (for this example) that we can move to one of the states in the invariant set $\tilde{Z}_8$ consisting of states being 7 transitions remote from the target state. This is repeated until the target state $\tilde{x}_e$ is reached. If this control strategy is applied to the two tank system with $\tau = 0.1$ [s], the time-trajectory is obtained as for the example in § 7.4.

## 9. Delayed transition measurements

Previously it was assumed that it is exactly measured whenever the continuous state trajectory $x$ hits a boundary plane $H_x(\tilde{x}_1) \cap H_x(\tilde{x}_2)$ and that immediate control action is possible. This allows us to prevent undesired transitions. However, if the sensor emits a signal (just) after the continuous state has already crossed the boundary plane (and consequently is not on the boundary plane any more), or if immediate control action is not possible, then preventing inputs can no longer be used. In such cases, an observed transition has to be corrected instead of prevented, as formalized in the following definition.

**Definition 9:** Given two discrete states $\tilde{x}_1$ and $\tilde{x}_2$ satisfying Condition 2. An input $\tilde{u} \in \tilde{U}$ is *correcting* for the transition $\tilde{x}_1 \rightarrow \tilde{x}_2$ if

$$f^r(x, u) < 0, \qquad \in H_x(\tilde{x}_1) \cap H_x(\tilde{x}_2) \tag{8}$$

This means that $\tilde{u}$ can be used to correct the changing of the continuous state from $H_x(\tilde{x}_1)$ to $H_x(\tilde{x}_2)$ as long as it is sufficiently close to the boundary plane as is proven in Philips (2001).

By replacing the preventing inputs by correcting inputs, the controller design methods as discussed in this paper also can be used for the case of transition measurements. For more details, we refer to Philips (2001).

## 10. Modifications when Assumption 1 is not met

For the sake of a transparent presentation of the controller design methods Assumption 1 was adopted. In this section we will discuss the required modifications of the proposed control strategies in case the assumption does not hold. If this assumption is violated, then an input $\tilde{u}$ which is preventing for a transition $\tilde{x}_1 \rightarrow \tilde{x}_2$ no longer guarantees that the system remains in $\tilde{H}_x(\tilde{x}_1)$, when the continuous state lies at the intersection of three or more hypercubes. Indeed, it is still guaranteed that the trajectory $x$ will not enter $H_x(\tilde{x}_2)$, but it is not clear that $x$ will stay in (or 'return' to) $H_x(\tilde{x}_1)$, because other hypercubes might be reached.

**Example 2:** Consider the linear time-invariant system

$$\dot{x}^1 = -u^1 + u^2$$

$$\dot{x}^2 = u^1 - \tfrac{1}{2}u^2$$

with $\tilde{U} = \{(1, 0)^{\mathrm{T}}, (0, 1)^{\mathrm{T}}\}$. Note that the superscripts indicate the components of the vectors. Suppose there are only two sensors situated at the $x^i$-axes. This means $\beta_1^1 = \beta_1^2 = 0$ (we take $\beta_0^i = -\infty$ and $\beta_2^i = \infty$, $i = 1, 2$) and the discrete states are formed by the four quadrants denoted by $\Omega_i$, e.g. the third quadrant $\Omega_3$ is given by the hypercube $H_x(\tilde{x}_3) = \{x \in \mathbb{R}^2 \mid x^1 \leq 0, x^2 \leq 0\}$. On the basis of Proposition 3 it can be shown that $\Omega_3$ is discretely controlled invariant *provided* Assumption 1 holds. A feedback controller doing the job *under Assumption 1* is given by

$$u(t) = \begin{cases} (1, 0)^{\mathrm{T}}, & \text{if the last sensor signal was given by } x^1 = 0 \\ (0, 1)^{\mathrm{T}}, & \text{if the last sensor signal was given by } x^2 = 0 \end{cases}$$

It is easily verified that the trajectories of the closed-loop system starting in $\Omega_3$ will reach the origin within a finite time after an infinite number of control switches. Hence, Assumption 1 is violated by *all* discretely controlled trajectories starting in the third quadrant in spite of the fact that for *almost all* discretely controlled trajectories (open-loop) the origin is not reached. Due to the violation of the assumption, $\Omega_3$ is not discretely controlled invariant. Indeed, once arrived in the origin, there is no escape; we will leave $\Omega_3$ as the control value $(0, 1)^{\mathrm{T}}$ indeed prevents $\Omega_3 \rightarrow \Omega_4$, but steers the state trajectory into $\Omega_2$, while $(0, 1)^{\mathrm{T}}$ prevents $\Omega_3 \rightarrow \Omega_2$, but this choice steers the state into $\Omega_4$.

The phenomenon of an infinite number of discrete actions in a finite length interval is called Zenoness and forms a major problem in hybrid systems theory

(Johansson *et al.* 1999). Also in this context it obscures the control design as is demonstrated by the example above.

Note that for the first quadrant $\Omega_1$ discretely controlled invariance (except for starting in the origin, so for almost all trajectories generated by the controller) is true by applying the feedback controller

$$u(t) = \begin{cases} (1,0)^{\mathrm{T}}, & \text{if the last sensor signal was given by } x^2 = 0 \\ (0,1)^{\mathrm{T}}, & \text{if the last sensor signal was given by } x^1 = 0 \end{cases}$$

Unfortunately, it is not possible to guarantee *a priori* that Assumption 1 will not be violated as it depends on the designed control strategy. When the controller has been implemented in practice, the violation of Assumption 1 is mainly caused by the switching between inputs that are preventing (or correcting) different undesired transitions but move the trajectory $x$ to the same 'corner' like in Example 2. The Zeno behaviour can be detected on-line by noticing the fast subsequent switching signals ('chattering' of sensors) and precautions might be taken by, e.g. selection of other moving and preventing inputs.

However, a (better) alternative is modifying the proposed design methods to obtain a controller that guarantees that a transition can indeed be prevented (or corrected) even if Assumption 1 is not true. The discretely controlled invariance (and consequently, Assumption 1) is obscured on the lower dimensional boundaries between the hypercubes (e.g. $\{0\}$ in Example 2). For controlled invariance of a set (e.g. $\Omega_3$ in Example 2) we need to have a discrete control value that steers the state on the lower dimensional boundaries of the particular set towards the interior. For Example 2 this means that we have to find a control value that will be applied in the origin that steers the vector field into $\Omega_3$. Of course, for the example at hand this is not possible, but for the following two examples such a discrete control value can be found.

**Example 3:** Consider again Example 2 with the modified discrete input set $\tilde{U} = \{(0,1)^{\mathrm{T}}, (0,1)^{\mathrm{T}}, (-4,-6)^{\mathrm{T}}\}$. Note that the vector field is constant for discrete input $(-4,-6)^{\mathrm{T}}$ and equal to $(-2,-1)^{\mathrm{T}}$ pointing inside the third quadrant $\Omega_3$ for *all* points on the boundary. Note that it can be inferred from the adjacency matrices that $(-4,-6)^{\mathrm{T}}$ is preventing for $\Omega_3$ as this input is preventing for both $\Omega_3 \to \Omega_4$ *and* (!) $\Omega_3 \to \Omega_2$. Hence, we just find one control value that is preventing for all undesired transitions. This additional check has to be performed to determine the controlled invariance of $\Omega_3$ in this situation.

Note that in this case we can use the information available in the adjacency matrices to find such a control value and no additional computations are necessary. The vector field corresponding to the control value $(-4,-6)^{\mathrm{T}}$ actually points inside the interior of $\Omega_3$ for all its boundary points. So, this control is preventing for *all* possible transitions. It might be clear that is too conservative in general as is also demonstrated by the following example.

**Example 4:** Consider the modified linear time-invariant system of Example 2 given by

$$\dot{x}^1 = -u^1 + u^2 - (x^1 + x^2 + 1)u^3$$
$$\dot{x}^2 = u^1 - \tfrac{1}{2}u^2 - (2x^1 + x^2 + 1)u^3$$

with $\tilde{U} = \{(1,0,0)^{\mathrm{T}}, (0,1,0)^{\mathrm{T}}, (0,0,1)^{\mathrm{T}}\}$. Note that $\Omega_3$ can be made discretely-controlled invariant by applying

$$u(t) = \begin{cases} (1,0,0)^{\mathrm{T}} & \text{if the last sensor signal was given by only } x^1 = 0 \\ (0,1,0)^{\mathrm{T}} & \text{if the last sensor signal was given by only } x^2 = 0 \\ (0,0,1)^{\mathrm{T}} & \text{if the last sensor signal was given by both } x^2 = 0 \text{ and } x^1 = 0 \end{cases}$$

as the vector field in the origin is equal to $(-1,-1)^{\mathrm{T}}$ for control input $(0,0,1)^{\mathrm{T}}$. However, note that $(0,0,1)^{\mathrm{T}}$ is not a preventing input for $\Omega_3 \to \Omega_4$ or $\Omega_3 \to \Omega_2$.

In this example a different discrete control value is selected on a lower dimensional boundary (in this case the origin) between three hypersurfaces. Note that this can *in principle* not be inferred from the adjacency matrices and should be computed separately. This extension of the methodologies can be made by including also information on the transitions between *non-adjacent* discrete states, in particular between states that share *lower dimensional* boundaries. For all discrete inputs $\tilde{u}$ and all pairs of (non-equal) discrete states $\tilde{x}_1 = i$ and $\tilde{x}_2 = j$ for which the intersection of the corresponding hypercubes is non-empty, we have to decide if for all $x \in rint(H_x(i) \cap H_x(j))$† $f(x, \tilde{u})$ points inside $intH_x(j)$. Denote the answer to this question by $(c_{\tilde{u}})_{ij}$, which is equal to 1 if the statement holds, and 0 otherwise. Note that $(c_{\tilde{u}})_{ii} = 0$. The Boolean matrices obtained in this way are denoted by $C_{\tilde{u}}$ and are called the correcting matrices in the spirit of § 9. Indeed, one could say that if $(c_{\tilde{u}})_{ij} = 1$, that $\tilde{u}$ 'corrects' the transition $j \to i$. The computation of this matrix simply requires a

† Note that no other sensors will emit signals as long as the state lies in the relative interior. On the boundary of this intersection other sensor signals may be received, which provide additional information of the whereabouts of the continuous state, and a different discrete control value can be used on this boundary with an even lower dimension.

number of optimizations. To illustrate this, consider two adjacent discrete states that satisfy Condition 2. Then $(c_{\tilde{u}})_{ij} = 1$ if and only if $\max_{x \in rint(H_x(i) \cap H_x(j))} f^r(x, \tilde{u}) > 0$.† If the two states share an $(n-k)$ dimensional boundary, then the value of $k$ optimal values have to be verified on their sign. In Example 4 this means for $\Omega_3 \rightarrow \Omega_1$ that $(c_{\tilde{u}})_{13} = 1$ iff $\min_{x \in \Omega_3 \cap \Omega_1} f^1(x, \tilde{u}) < 0$ and $\min_{x \in \Omega_3 \cap \Omega_1} f^1(x, \tilde{u}) < 0$. Note that $\Omega_3 \cap \Omega_1 = \{0\}$ and hence only $\tilde{u} = (0, 0, 1)^{\mathrm{T}}$ satisfies the latter conditions.

However, in certain cases no additional computations are necessary and the adjacency matrices $A_{\tilde{u}}$ suffice to construct the correcting matrices $C_{\tilde{u}}$. One such condition is the linearity of the vector field of (1). In this case the so-called *component equilibria surfaces*, which are defined by $\mathcal{S}_{r,\tilde{u}} := \{x \in \mathbb{R}^n \,|\, f^r(x, \tilde{u}) = 0\}$ for a given $\tilde{u}$ and component $r \in \{1, \ldots, n\}$ are hyperplanes. If for a pair of adjacent discrete states $\tilde{x}_1, \tilde{x}_2$ satisfying Condition 2 and given input $\tilde{u}$ the transitions in both directions are possible (i.e. $\hat{x}_1^{\mathrm{T}} A_{\tilde{u}} \hat{x}_2 = \hat{x}_2^{\mathrm{T}} A_{\tilde{u}} \hat{x}_1 = 1$), we know that the hyperplane $\mathcal{S}_{r,\tilde{u}}$ crosses the boundary plane $H_x(\tilde{x}_1) \cap H_x(\tilde{x}_2)$ between the corresponding hypercubes. If we determine all pairs of discrete states that satisfy this property we know exactly through which hypercubes and boundaries the plane is running. As $f^r(x, \tilde{u})$ is positive on one side and negative on the other side of the plane, we know this directional information for all points except the ones in the *interior* of the hypercubes through which the component equilibria surfaces are running. This is illustrated in figure 13. The dotted line indicates the component equilibria surface $\mathcal{S}_{2,\tilde{u}}$ for some $\tilde{u}$ and we observe that there are 25 discrete states. The arrows give the direction of the vector field, so we see that on the upper right area of the figure the direction is positive in the $x_1$-component and negative in the bottom left area. This information can be found directly in the adjacency matrices and we can easily detect through which discrete states $\mathcal{S}_{2,\tilde{u}}$ is running. Indeed, for instance, $(a_{\tilde{u}})_{11,16} = 1$ and $(a_{\tilde{u}})_{16,11} = 0$ mean that $\mathcal{S}_{2,\tilde{u}}$ does not intersect $H_x(16) \cap H_x(11)$, but $(a_{\tilde{u}})_{13,18} = 1$ and $(a_{\tilde{u}})_{18,13} = 1$ mean that $H_x(13) \cap H_x(18) \cap \mathcal{S}_{2,\tilde{u}}$ is not empty. Knowing the orientation of the discrete states w.r.t. equilibria surface means that we can easily deduce from $A_{\tilde{u}}$ that $f^2(a, \tilde{u}) < 0$, $f^2(b, \tilde{u}) < 0$ and $f^2(c, \tilde{u}) > 0$ ($a$, $b$, $c$ being the corner points in figure 13) in an automated manner.

The linearity of the vector field can actually be exploited to facilitate the computation of the adjacency matrices $\{A_{\tilde{u}} \,|\, \tilde{u} \in \tilde{U}\}$ as well (see Chapter 3 of Philips (2001), which makes the linear case very efficient to handle both the situation, where Assumption 1 is valid, and

† Observe the strictness of this inequality. In case of preventing inputs it would suffice to have a (non-strict) inequality due to Proposition 1.

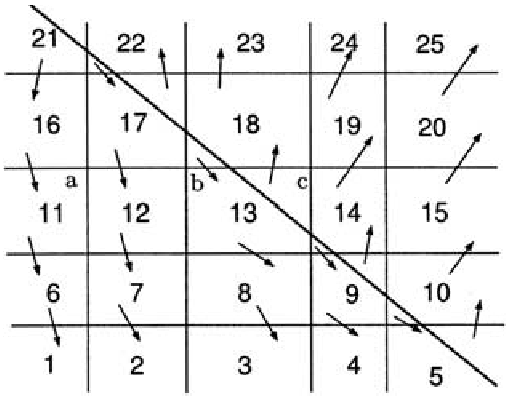

Figure 13. An affine component equilibria surface.

the situation, where it is violated. The above reasoning for linear plants can be extended to the case where component equilibria surfaces satisfy certain monotonicity properties.

Based on these modifications, we can also extend the invariance properties as derived in §6. Indeed, the matrix $S_{\tilde{V}}$ in Proposition 3 has to be replaced by the matrix $N \ominus \bigoplus_{\tilde{u} \in \tilde{V}} C_{\tilde{u}}$, where $N$ denotes the so-called neighbour matrix with $n_{ij} = 1$, if $i \neq j$ and $H_x(i) \cap H_x(j) \neq \emptyset$ and $n_{ij} = 0$ otherwise. However, $S_{\tilde{V}} \hat{z} \ominus \hat{z} = 0$ is only a sufficient condition for the invariance of the set $\tilde{Z} \subseteq \tilde{X}$ and not necessary in general. The reason for this is the difference between preventing and correcting inputs, which lies in the strictness of the value of the optimization problem (see the final footnote). The algorithms for computing the smallest and largest discretely controlled sets as outlined in §6 are still valid with the understanding that the controlled invariance should be replaced by satisfying the condition $S_{\tilde{V}} \hat{z}_{inv} \ominus \hat{z}_{inv} = 0$. Note that forceable transitions and moving inputs can only be used for *adjacent* discrete states (i.e. sharing an $(n-1)$-dimensional boundary) in spite of the extensions as discussed in this section.

With the above-mentioned modifications the proposed control design methods are valid even if Assumption 1 is violated.

## 11. Conclusions

In this paper controller design methods are proposed for quantized systems, i.e. dynamical systems specified by differential equations that are observed through discrete measurements. The control strategies are based on the usage of discrete-event abstractions of the quantized system. The corresponding automata are obtained via derivative tests on the boundaries between the discrete states. The discrete states comply to certain regions in the continuous state space. Instead of applying solely the classical control methodologies on the resulting discrete-event systems, improvements are proposed by including additional information provided by the continuous plant, such as continuity of the state trajectories and information on derivatives that holds for parts of the state space.

The basic building blocks for the controller synthesis methods are preventing (or correcting) inputs that guarantee that certain undesired transitions can be prevented (or corrected) and moving inputs that ensure that the continuous state evolves towards a desired direction. Based on these types of inputs we can derive sufficient conditions for enforcing specific transitions between discrete states, which can be used to solve the reachability and stabilization problem. This route is taken in the first controller synthesis method. The second method is based on constructing a nested sequence of discretely controlled invariant sets and steering the system through this sequence until the desired final set is reached. Necessary and sufficient conditions for controlled invariance are presented. Also algorithms are included to compute the smallest and largest discretely controlled invariant sets with respect to a given set. The second method allows a modification in which the nested controlled invariant sets are constructed by including discrete states in Step 3(ii) of the algorithm instead of removing them. This method is described in Chapter 5 of Philips (2001). These type of (preventing and moving) control actions are not included in the classical discrete-event control design methods as in, e.g. Ramadge and Wonham (1987) as they are only based on the adjacency matrices. The inclusion of the additional information on preventing, correcting and moving control inputs therefore extensively broadens the range of the control problems that can be tackled successfully.

All the techniques and computations are explicitly described by Boolean matrices and vectors and are ready for direct application. This is demonstrated for a benchmark problem of a two tank system.

At the end of the paper we provided a discussion to resolve problems that might arise from the fact that several boundaries are reached simultaneously. We described a method to guarantee the controlled invariance of the nested sets as used in the control design by adopting so-called correcting matrices. This requires additional off-line computations mostly in the form of optimizations. Under certain conditions (e.g. linearity of the underlying continuous plant) the computational burden can be reduced by obtaining this information directly from the adjacency matrices. We are currently looking for other more general conditions related to the monotonicity of the component equilibria surfaces that makes this computational reduction possible. The required systematic tools for performing these additional computations forms another point of future research, next to the further implementation and verification of the techniques on simulation and practical examples.

## References

Antsaklis, P. J., and Nerode, A. (guest Eds), 1998, Special issue on hybrid control systems. *IEEE Transactions on Automatic Control*, **43**.

Blanchini, F., 1999, Set invariance in control. *Automatica*, **35**, 1747–1767.

Bruinsma, U. B. D. M. R., 1997, State-event discrete modelling of non-linear batch plants. Master's thesis, Eindhoven University of Technology, August, NR-1294.

Cury, J. E. R., Kroch, B. H., and Niinomi, T., 1998, Synthesis of supervisory controllers for hybrid systems based on approximating automata. *IEEE Transactions on Automatic Control*, **43**, 564–568.

De Bruin, D., and van den Bosch, P. P. J., 1998, Measurement of the lateral vehicle position with permanent magnets. In *Proceedings of the IFAC Workshop on Intelligent Components for Vehicles (IVC'98)*, Seville, Spain, pp. 9–14.

Förstner, D., and Lunze, J., 2001, Discrete-event models of quantised systems for diagnosis. *International Journal of Control*, **74**, 690–700.

Heemels, W. P. M. H., Gorter, R. J. A., van Zijl, A., van den Bosch, P. P. J., Weiland, S., Hendrix, W. H. A., and Vonder, M. R., 1999, Asynchronous measurement and control: A case study on motor synchronization. *Control Engineering Practice*, **7**, 1467–1482.

Hsu, C. S., 1985, A discrete method of optimal control based upon the cell state space concept. *Journal of Optimization Theory and Applications*, **64**, 547–569.

Johansson, K. H., Lygeros, J., Sastry, S., and Egerstedt, M., 1999, Simulation of Zeno hybrid automata. In *38th IEEE Conference on Decision and Control*, Phoenix, USA, pp. 3538–3543.

Kornoushenko, E. K., 1975, Finite-automaton approximation to the behaviour of continuous plants. *Automation and Remote Control*, pp. 2068–2074.

Kowalewski, S., Engell, S., Preußig, J., and Stursberg, O., 1999, Verification of logic controllers for continuous plants using timed condition/event-system models. *Automatica*, **35**, 505–518.

Lunze, J., 1994, Qualitative modelling of linear dynamical systems with quantized state measurements. *Automatica*, **30**, 417–431.

Lunze, J., 1995, Stabilization of nonlinear systems by qualitative feedback controllers. *International Journal of Control*, **62**, 109–128.

Lunze, J., 1999, A timed discrete-event abstraction of continuous-variable systems. *International Journal of Control*, **72**, 1147–1164.

Lunze, J., 2000, Diagnosis of quantised systems by means of timed discrete-event representations. In N. Lynch and B. Krogh (Eds), *Hybrid Systems: Computation and Control* (Berlin: Springer-Verlag), pp. 258–271.

Lunze, J., Nixdorf, B., and Richter, H., 1997, Hybrid modelling of continuous-variable systems with application to supervisory control. In *Proceedings of the European Control Conference ECC'97*, Brussels, Belgium, 1–4 July.

Lunze, J., Nixdorf, B., and Richter, H., 2001, Process supervision by means of a hybrid model. *Journal of Process Control*, **11**, 89–104.

Morse, A. S., Pantelides, S. S., Sastry, S., and Schumacher, J. M. (guest Eds), 1999, A special issue on hybrid systems. *Automatica*, **35**.

Nagumo, M., 1942, Über die lage der integralkurven gewöhnlicher differential-gleichungen. *Proceedings of the Physico-Mathematical Society of Japan*, **24**, 551–559.

Papa, M., Tai, H.-M., and Shenoi, S., 1997, Cell mapping for controller design and evaluation. *IEEE Control Systems Magazine*, **17**, 52–65.

Philips, P., 2001, Modelling, control and fault detection of discretely-observed systems. PhD thesis, Eindhoven University of Technology, March.

Philips, P., Bruinsma, U., Weiss, M., and Preisig, H. A., 1997, 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.

Phillips, A. M., and Tomizuka, M., 1995, Multirate estimation and control under time-varying data sampling with applications to information storage devices. In *Proceedings of the 1995 American Control Conference*, Seattle, Washington, June, pp. 4151–4155.

Preisig, H. A., 1996, A mathematical approach to discrete-event dynamic modelling of hybrid systems. *Computers and Chemical Engineering*, **20**, S1301–S1306.

Preisig, H. A., Pijpers, M. J. H., and Weiss, M., 1997, A discrete modelling procedure for continuous processes based on state-discretisation. In *Proceedings of the 2nd Symposium on Mathematical Modelling*, Vienna, February, pp. 189–194.

Raisch, J., and O'Young, S. D., 1998, Discrete approximation and supervisory control of continuous systems. *IEEE Transactions on Automatic Control*, **43**, 569–573.

Ramadge, P. J. G., and Wonham, M. W., 1987, Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, **25**, 206–230.

Stiver, J. A., and Antsaklis, P. J., 1993, Extracting discrete event system models from hybrid control systems. In *Proceedings of the 1993 International Synmposium on Intelligent Control*, Chicago, Illinois, USA, pp. 298–301.

Stursberg, O., Kowalewski, S., and Engell, S., 1997, Generating timed discrete models of continuous systems. In *Proceedings of the 2nd IMACS Symposium on Mathematical Modelling of Systems (MATHMOD)*, Vienna, Austria, February 5–7, pp. 203–210.

Van der Schaft, A. J., and Schumacher, J. M., 2000, *An Introduction to Hybrid Dynamical Systems*, volume 251 of *Lecture Notes in Control and Information Sciences* (London: Springer).

Wonham, W. M., 1979, *Linear Multivariable Control: A Geometric Approach* (Berlin: Springer-Verlag).