Appl. Math. and Comp. Sci., 1995, vol.5, No.3, 511–521

ARITHMETIC MODELLING — A LINK BETWEEN DISCRETE-TIME SYSTEMS AND BINARY PROCESS CONTROL

DIETER FRANKE*

The paper addresses finite state machines which provide suitable mathematical models for discrete-event dynamical systems. Boolean automata are of particular interest. In contrast to the classical automata theory, an arithmetic representation of Boolean functions is used based on multilinear polynomials. By this technique, finite automata are embedded in the Euclidean vector space, which makes it possible to find a closer relationship between discrete-event systems and classical discrete-time systems. The problem of self-regulation of binary dynamic systems is interpreted in terms of feedback control structures with a focus on two special classes of systems. First, systems which are linear with respect to state and control are studied within the framework of the classical state space theory. Then systems which are linear in the control and multilinear in the state are considered. It is shown that multilinear state feedback can be utilized to globally linearize the overall system. The design equations turn out to be linear with respect to the controller parameters in this case.

1. Introduction

The state equations of a finite Boolean automaton (Booth, 1967; Bochmann, 1975),

$$\boldsymbol{x}(k+1) = \boldsymbol{f}\left[\boldsymbol{x}(k), \boldsymbol{u}(k)\right] \tag{1}$$

$$\boldsymbol{y}(k) = \boldsymbol{g}\left[\boldsymbol{x}(k), \boldsymbol{u}(k)\right] \tag{2}$$

with input $\boldsymbol{u} \in \mathcal{B}^p$, state $\boldsymbol{x} \in \mathcal{B}^n$ and output $\boldsymbol{y} \in \mathcal{B}^q$ resemble those of a classical discrete-time system. The main difference is the use of Boolean algebra in functions $\boldsymbol{f}$ and $\boldsymbol{g}$ rather than arithmetic operations, since the components of $\boldsymbol{u}, \boldsymbol{x}$ and $\boldsymbol{y}$ are logical variables taking only values 0 and 1.

However, an arithmetic representation of Boolean functions which has been used in Boolean reliability theory so far (Störmer, 1970; Reinschke, 1973) has been also employed by Franke (1992; 1993; 1994) for a novel approach to binary dynamic systems. Any completely specified Boolean function $y = f(\boldsymbol{x}) = f(x_1, ..., x_n)$ can be

* Universität der Bundeswehr Hamburg, Fachbereich Elektrotechnik, Holstenhofweg 85, D–22043 Hamburg, Germany

written uniquely as a multilinear polynomial

$$y = f(\boldsymbol{x}) = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{j=2}^{n} \sum_{i=1}^{j-1} a_{ij} x_i x_j$$

$$+ \sum_{k=3}^{n} \sum_{j=2}^{k-1} \sum_{i=1}^{j-1} a_{ijk} x_i x_j x_k + \ldots + a_{123\ldots n} x_1 x_2 x_3 \ldots x_n \tag{3}$$

It has the same structure as classical Shegalkin polynomials (Shegalkin, 1928) but uses *arithmetic* operations.

The number of coefficients appearing in this equation is $N = 2^n$ which agrees with the number of rows of the sequence table of $y = f(\boldsymbol{x})$. Therefore, given any *completely specified* sequence table, the coefficients are determined uniquely by solving a set of *linear* algebraic equations.

Consider a simple case when $n = 2$,

$$y = f(x_1, x_2) = a_0 + a_1 x_1 + a_2 x_2 + a_{12} x_1 x_2 \tag{4}$$

Let the switching Table 1 be given, with given binary values $y^{(1)}, \ldots, y^{(4)}$.

Tab. 1. Switching table for $y = f(x_1, x_2)$.

x_2	x_1	y
0	0	$y^{(1)}$
0	1	$y^{(2)}$
1	0	$y^{(3)}$
1	1	$y^{(4)}$

Then the corresponding coefficients must satisfy the equation

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_{12} \end{bmatrix} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ y^{(4)} \end{bmatrix} \tag{5}$$

which has the unique solution

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_{12} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ y^{(4)} \end{bmatrix} \tag{6}$$

In the past, this type of modelling Boolean functions has only been used in Boolean reliability theory. It does, however, also apply to functions $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ and $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u})$ appearing on the right-hand sides of eqns. (1) and (2), respectively. This makes finite-state machines much closer related to classical discrete-time systems than it was assumed in the past and provides a novel access to analysis and design of binary dynamical systems.

2. Arithmetically Linear Boolean Functions and Automata

There is an appealing class of special systems which have a simple structure even in higher dimensions: *arithmetically linear systems.* The state equations of an arithmetically linear automaton have obviously the general form

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{a}_0 \tag{7}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) + \boldsymbol{c}_0 \tag{8}$$

where $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$ are constant matrices of appropriate dimensions, and $\boldsymbol{a}_0, \boldsymbol{c}_0$ are n-dimensional constant vectors.

Consider as a simple practical example the Boolean switching table (Table 2) of a pneumatic positioning cylinder widely used in manufacturing processes.

Tab. 2. Boolean switching table for positioning cylinder.

$x(k)$	$u(k)$	$x(k+1)$
0	0	0
0	1	1
1	0	0
1	1	1

Here $x(k)$ is the Boolean state of the cylinder ($x = 0$ retracted, $x = 1$ extended) at step k, and $u(k)$ is the Boolean control applied at step k. Obviously, the Boolean state equation for this system is arithmetically linear,

$$x(k+1) = u(k)$$

The arithmetically linear automaton just introduced is *completely specified.* Its linearity results from the special feature that all the coefficients of multilinear terms in eqn. (3) vanish. There is, however, another important class of automata, namely *incompletely specified* automata, which sometimes can be made algebraically linear by setting certain coefficients equal to zero.

Incompletely specified Boolean functions arise whenever one or more rows of the sequence table can be excluded by knowledge or is forbidden to occur. As an example, consider Table 1 with the first row being excluded. Based on the notation used in eqn. (4), eqn. (6) does not apply directly, since $y^{(1)}$ is unspecified. However, the solution of the underdetermined set of equations can be made unique by setting $a_{12} = 0$. This makes, on the one hand, $y = f(x_1, x_2) = a_0 + a_1 x_1 + a_2 x_2$ strictly

linear, and, on the other hand, specifies $y^{(1)} = y^{(2)} + y^{(3)} - y^{(4)}$. The value of $y^{(1)}$ will not in general be binary. For $y^{(2)} = y^{(3)} = 1$, $y^{(4)} = 0$ we obtain $y^{(1)} = 2$ and hence $a_0 = 2$, $a_1 = a_2 = -1$. Therefore, $y = f(x_1, x_2) = 2 - x_1 - x_2$ in this example.

The automaton described by eqns. (1), (2) with dimensions n, p and q for the state $\boldsymbol{x}$, control $\boldsymbol{u}$ and output $\boldsymbol{y}$, respectively, is said to be incompletely specified, if only a subset of rows is admitted in the sequence table.

An important class of incompletely specified automata are those which allow only a subset of 2^n entries of $\boldsymbol{x}$. In such an automaton, the control $\boldsymbol{u}(k)$ is said to be *applicable* to the state $\boldsymbol{x}(k)$ if $\boldsymbol{x}(k+1) = \boldsymbol{f}[\boldsymbol{x}(k), \boldsymbol{u}(k)]$ also belongs to the allowed subset.

3. Arithmetically Linear Feedback Control

In many situations, a prescribed cyclic operation of a binary dynamical system is required, and therefore the problem of self-regulation of the process arises. If the required period of the cycle is given as

$$k_p < 2^n \tag{9}$$

where n is again the dimension of the state vector $\boldsymbol{x}$, the process to be designed is *incompletely specified.*

The problem to be considered is the following: given an arithmetically linear binary process described by eqns. (7), (8), does there exist a non-dynamic arithmetically linear state feedback

$$\boldsymbol{u}(k) = \boldsymbol{r}_0 + \boldsymbol{R}\boldsymbol{x}(k) \tag{10}$$

or a dynamic one,

$$\boldsymbol{u}(k) = \boldsymbol{r}_0 + \boldsymbol{R}\boldsymbol{x}(k) + \sum_{\nu=1}^{m} \boldsymbol{R}_\nu \boldsymbol{x}(k-\nu), \quad m < k_p \tag{11}$$

such that the preassigned cyclic behaviour with the period k_p of the closed-loop system will be achieved?

3.1. Non-dynamic State Feedback

In the case of a prescribed period $k_p < 2^n$, the required sequence of k_p cyclic states and the corresponding controls can be written as an incompletely specified Boolean switching table. Now, by using the trial constant state feedback law (10),

$$\boldsymbol{u}(k) = \boldsymbol{R}\boldsymbol{x}(k) + \boldsymbol{r}_0$$

a set of arithmetically linear equations is obtained for the coefficients of this control law. Therefore, whenever this set of equations has a solution, a non-dynamic state feedback described by eqn. (10) exists and can be computed from these equations.

Consider as an example three positioning cylinders described by Table 2, which are required to operate in a cycle with the period $k_p = 6$ according to Table 3.

Tab. 3. Cyclic operation of three positioning cylinders.

	$x_3(k)$	$x_2(k)$	$x_1(k)$	$u_3(k)$	$u_2(k)$	$u_1(k)$
1.	1	0	0	1	0	1
2.	1	0	1	0	0	1
3.	0	0	1	0	1	1
4.	0	1	1	0	1	0
5.	0	1	0	1	1	0
6.	1	1	0	1	0	0

In this example, a constant state feedback (10) exists, and the solution is

$$\boldsymbol{R} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{r}_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

3.2. Dynamic State Feedback

A constant state feedback law (10) may not exist due to failure of the underlying rank condition for the set of linear equations. In this case, a trial dynamic state feedback is proposed according to (11),

$$\boldsymbol{u}(k) = \boldsymbol{R}\boldsymbol{x}(k) + \sum_{\nu=1}^{m} \boldsymbol{R}_\nu \boldsymbol{x}(k-\nu) + \boldsymbol{r}_0, \quad m < k_p$$

using delayed values $\boldsymbol{x}(k-\nu)$ in addition to $\boldsymbol{x}(k)$. The switching table for cyclic operation is augmented by columns $\boldsymbol{x}(k-\nu)$, thus yielding an augmented set of arithmetically linear equations for the coefficients of the control law (11).

Whenever the rank condition for solvability of these equations can be satisfied for some m, a dynamic state feedback (11) exists and can be computed from these equations.

Table 4 outlines an example with two positioning cylinders which operate in a cycle with $k_p = 6$.

Tab. 4. Example for dynamic state feedback.

	$x_2(k-2)$	$x_1(k-2)$	$x_2(k)$	$x_1(k)$	$u_2(k)$	$u_1(k)$
1.	0	0	0	0	0	1
2.	1	0	0	1	1	1
3.	0	0	1	1	0	1
4.	0	1	0	1	0	0
5.	1	1	0	0	1	0
6.	0	1	1	0	0	0

A constant state feedback (10) does not exist in this example, since $k_p > 2^n$. However, by augmenting the table by the delayed state $\boldsymbol{x}(k-2)$, the rank condition is satisfied and the controller is readily obtained as.

$$\boldsymbol{u}(k) = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{x}(k-2) + \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

4. Arithmetically Multilinear Automata

In the general case, the right-hand sides of (1), (2) cannot be expected to be representable as arithmetically linear functions. This is especially true for completely specified machines to be considered in the subsequence. For the study of this type of systems the following notation of a completely specified Boolean function $y = f(\boldsymbol{x})$ proves to be convenient.

Since the polynomial is linear (in the arithmetic sense) with respect to its 2^n coefficients, $f(\boldsymbol{x})$ can be written as an inner product

$$y = f(\boldsymbol{x}) = \boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{a} \tag{12}$$

where the coefficients are assumed to be arranged in the vector $\boldsymbol{a}$ in the same order as they appear in (3). Therefore we obtain in the case $n = 1$:

$$\boldsymbol{p}^T(\boldsymbol{x}) = [1, x], \quad \boldsymbol{a}^T = [a_0, a_1]$$

in the case $n = 2$:

$$\boldsymbol{p}^T(\boldsymbol{x}) = [1, x_1, x_2, x_1x_2], \quad \boldsymbol{a}^T = [a_0, a_1, a_2, a_{12}]$$

in the case $n = 3$:

$$\boldsymbol{p}^T(\boldsymbol{x}) = [1, x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1x_2x_3]$$

$$\boldsymbol{a}^T = [a_0, a_1, a_2, a_3, a_{12}, a_{13}, a_{23}, a_{123}]$$

etc. It can be seen that the structure of the vector $\boldsymbol{p}(\boldsymbol{x})$ depends only on the number n, whereas the vector $\boldsymbol{a}$ depends only on the concrete Boolean function.

Consider two completely specified Boolean functions of $\boldsymbol{x} \in \mathcal{B}^n$,

$$y_1 = f_1(\boldsymbol{x}) = \boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{a}, \quad y_2 = f_2(\boldsymbol{x}) = \boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{b}$$

Since the arithmetic product $y = y_1y_2$ is obviously a completely specified Boolean function as well, it can be written as

$$y = y_1\, y_2 = \boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{c} = \boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{a}\boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{b} \tag{13}$$

Due to the idempotence property, $x_i^2 = x_i$, evaluation of (13) will always yield

$$\boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{a}\boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{b} = \boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{\beta}(\boldsymbol{a}, \boldsymbol{b}) \tag{14}$$

and hence

$$\boldsymbol{c} = \boldsymbol{\beta}(\boldsymbol{a}, \boldsymbol{b}) \tag{15}$$

where the components of $\boldsymbol{\beta}$ are bilinear in the components of $\boldsymbol{a}$ and $\boldsymbol{b}$. To be specific, one obtains in the case $n = 1$:

$$c_0 = a_0 b_0, \quad c_1 = a_0 b_1 + a_1(b_0 + b_1) \tag{16}$$

in the case $n = 2$:

$$\begin{aligned} &c_0 = a_0 b_0, \;\; c_1 = a_0 b_1 + a_1(b_0 + b_1), \;\; c_2 = a_0 b_2 + a_2(b_0 + b_2) \\ &c_{12} = a_0 b_{12} + a_1(b_2 + b_{12}) + a_2(b_1 + b_{12}) + a_{12}(b_0 + b_1 + b_2 + b_{12}) \end{aligned} \tag{17}$$

etc.

The arithmetic representation of Boolean functions described above can be introduced into the right-hand sides of (1), (2), which yields multilinear polynomials with respect to the components of $\boldsymbol{u}(k)$ and $\boldsymbol{x}(k)$. Since the concepts of *state* feedback will be considered below, discussion will be confined to eqn. (1) in the subsequence. Due to its multilinear structure this equation can obviously be written as

$$\begin{aligned} \boldsymbol{x}(k+1) \;=\; \Bigg[\boldsymbol{A}_0 + \sum_{\mu=1}^{p} \boldsymbol{A}_\mu u_\mu(k) + \sum_{\nu=2}^{p} \sum_{\mu=1}^{\nu-1} \boldsymbol{A}_{\mu\nu} u_\mu(k) u_\nu(k) + \ldots \\ + \; \boldsymbol{A}_{12\ldots p} u_1(k) u_2(k) \ldots u_p(k) \Bigg] \boldsymbol{p}[\boldsymbol{x}(k)] \end{aligned} \tag{18}$$

which means *separation* of $\boldsymbol{u}$ and $\boldsymbol{x}$. Again the vector $\boldsymbol{p}(\boldsymbol{x})$ depends only on the degree n of the system whereas the matrices $\boldsymbol{A}$ are specified by the concrete sequential machine.

There are quite interesting special cases contained in (18), e.g. systems which are *linear with respect to the controls*:

$$\boldsymbol{x}(k+1) = \left[\boldsymbol{A}_0 + \sum_{\mu=1}^{p} \boldsymbol{A}_\mu u_\mu(k) \right] \boldsymbol{p}[\boldsymbol{x}(k)] \tag{19}$$

In what follows the class of systems (19) will be discussed subject to feedback control.

5. Multilinear State Feedback Control

In the classical control theory, it is quite common to achieve a desired plant behaviour via feedback control. Especially, the concepts of state feedback offer a large variety of design objectives and design procedures. Therefore the concept of non-dynamical state feedback will be applied to the automaton (1) using arithmetic notations for both the binary process and the binary controller. By means of the inner product notation introduced in (12) the Boolean state feedback law can always be written as

$$\boldsymbol{u}(k) = \boldsymbol{R}\, \boldsymbol{p}[\boldsymbol{x}(k)] \tag{20}$$

where $\boldsymbol{R}$ is a constant matrix of appropriate dimensions. The matrix $\boldsymbol{R}$ is, of course, not yet specified, whereas the matrices $\boldsymbol{A}$ of the binary process (18) are assumed to be known. The binary control law (20) can be viewed as *a rule basis* for the process (18).

The components of the control law (20) are

$$u_\mu(k) = \boldsymbol{r}_\mu^T \boldsymbol{p}[\boldsymbol{x}(k)] = \boldsymbol{p}^T[\boldsymbol{x}(k)]\boldsymbol{r}_\mu, \quad \mu = 1, ..., p \tag{21}$$

where $\boldsymbol{r}_\mu^T$ is the μ-th row of the matrix $\boldsymbol{R}$. By inserting (21) into the plant equation (18) it can be seen that the closed-loop system is an autonomous automaton whose state equation in view of (14), (15) can be written as

$$\boldsymbol{x}(k+1) = \widetilde{\boldsymbol{A}}\, \boldsymbol{p}[\boldsymbol{x}(k)] \tag{22}$$

where $\widetilde{\boldsymbol{A}}$ is a constant $n \times 2^n$ matrix whose elements are multilinear functions of the controller parameters. However, in the special case of a plant which is linear with respect to the controls according to (19), the elements of $\widetilde{\boldsymbol{A}}$ are obviously *linear* with respect to the controller parameters. Only this special case will be considered in the sequel.

6. Global Linearization

In the special case of a *scalar* control $u(k)$, eqn. (18) is always linear with respect to u. In (Franke, 1993) it is shown in an exemplary discussion that, in this case, multilinear state feedback can be designed in such a way that all the multilinear terms are cancelled out in the closed-loop system. Hence this is a *global linearization* technique. The resulting *linear state equations* can be treated like a classical discrete-time system. Free controller parameters can be utilized to achieve a specified cyclic behaviour, the notion of *eigenvalues* playing a central role.

The present approach is a more systematic extension to the multi-input case according to eqn. (19). By inserting the control law (21) into (19) one obtains the overall system

$$\boldsymbol{x}(k+1) = \left[\boldsymbol{A}_0 + \sum_{\mu=1}^{p} \boldsymbol{A}_\mu\, \boldsymbol{p}^T[\boldsymbol{x}(k)]\boldsymbol{r}_\mu\right] \boldsymbol{p}[\boldsymbol{x}(k)] \tag{23}$$

whose i-th component in view of (14), (15) reads

$$\begin{aligned} x_i(k+1) &= \left[\boldsymbol{a}_{i0}^T + \sum_{\mu=1}^{p} \boldsymbol{a}_{i\mu}^T\, \boldsymbol{p}^T[\boldsymbol{x}(k)]\boldsymbol{r}_\mu\right] \boldsymbol{p}[\boldsymbol{x}(k)] \\ &= \boldsymbol{p}^T[\boldsymbol{x}(k)] \left[\boldsymbol{a}_{i0} + \sum_{\mu=1}^{p} \boldsymbol{\beta}(\boldsymbol{r}_\mu, \boldsymbol{a}_{i\mu})\right], \quad i = 1, ..., n \end{aligned} \tag{24}$$

With regard to the notation of the vector $\boldsymbol{p}(\boldsymbol{x})$ introduced in (12), it is obvious that $\boldsymbol{p}^{(1)}$ with the first $n+1$ components of $\boldsymbol{p}$ is linear with respect to $\boldsymbol{x}$, and $\boldsymbol{p}^{(2)}$ with the remaining components is multilinear. Therefore, by using partitioning,

$$\boldsymbol{p}^T(\boldsymbol{x}) = \left[\boldsymbol{p}^{(1)T}(\boldsymbol{x}), \boldsymbol{p}^{(2)T}(\boldsymbol{x})\right], \quad \boldsymbol{a}_{i0}^T = \left[\boldsymbol{a}_{i0}^{(1)T}, \boldsymbol{a}_{i0}^{(2)T}\right], \quad \boldsymbol{\beta}^T = \left[\boldsymbol{\beta}^{(1)T}, \boldsymbol{\beta}^{(2)T}\right]$$

the conditions for global linearization are

$$a_{i0}^{(2)} + \sum_{\mu=1}^{p} \beta^{(2)}(r_\mu, a_{i\mu}) = 0, \quad i = 1, ..., n \tag{25}$$

and the resulting globally linear system is

$$x_i(k+1) = p^{(1)T}[x(k)] \left[a_{i0}^{(1)} + \sum_{\mu=1}^{p} \beta^{(1)}(r_\mu, a_{i\mu}) \right], \quad i = 1, ..., n \tag{26}$$

Equations (25) are a set of $n(2^n - n - 1)$ scalar *linear* equations for $p\,2^n$ controller parameters. It will provide solutions whenever

$$p \geq \frac{n(2^n - n - 1)}{2^n} \tag{27}$$

If the set of equations is underdetermined, the degrees of freedom can be utilized so as to achieve different cyclic behaviours of the autonomous automaton (26).

Example. Consider the discrete-event system with $x \in \mathcal{B}^2, u \in \mathcal{B}^2$, given by its completely specified switching Table 5. The equivalent arithmetic representation turns out to be linear with respect to controls u_1 and u_2.

Tab. 5. Switching table for a binary process with $x \in \mathcal{B}^2, u \in \mathcal{B}^2$.

$u_2(k)$	$u_1(k)$	$x_2(k)$	$x_1(k)$	$x_2(k+1)$	$x_1(k+1)$
0	0	0	0	0	1
0	0	0	1	1	1
0	0	1	0	1	1
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	1

Linearity conditions (25) for the overall system take here the form

$$-1 + 2r_{1,0} + r_{1,1} + r_{1,2} + r_{2,1} + r_{2,2} + r_{2,12} = 0$$

$$-1 + 2r_{1,0} + r_{1,1} + r_{1,2} - r_{2,1} - r_{2,2} - r_{2,12} = 0$$

and the resulting globally linear system according to (26) is

$$\boldsymbol{x}(k+1) = \begin{bmatrix} -r_{1,0} - r_{1,1} + r_{2,0} & -r_{1,0} - r_{1,2} + r_{2,0} \\ 1 - r_{1,0} - r_{1,1} - r_{2,0} & 1 - r_{1,0} - r_{1,2} - r_{2,0} \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} 1 - r_{2,0} \\ r_{2,0} \end{bmatrix}$$

It allows the following solutions for completely specified Boolean automata:

a) $\boldsymbol{x}(k+1) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ b) $\boldsymbol{x}(k+1) = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

c) $\boldsymbol{x}(k+1) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ d) $\boldsymbol{x}(k+1) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

For system (a) the controller equations are $u_1(k) = x_2(k), u_2(k) = 0$, and the periodic sequence of the states is

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \to \begin{bmatrix} 1 \\ 0 \end{bmatrix} \to \begin{bmatrix} 1 \\ 1 \end{bmatrix} \to \begin{bmatrix} 0 \\ 1 \end{bmatrix} \to \begin{bmatrix} 0 \\ 0 \end{bmatrix} \cdots$$

The period $k_p = 4$ reflects the conjugate complex pair of the eigenvalues $\widetilde{\lambda}_{1/2} = \pm j$ of the system.

For system (b) the controller equations are $u_1(k) = x_1(k), u_2(k) = 0$, and the periodic sequence of the states is

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \to \begin{bmatrix} 1 \\ 0 \end{bmatrix} \to \begin{bmatrix} 0 \\ 0 \end{bmatrix} \cdots$$

whose period $k_p = 2$ corresponds to the eigenvalue $\widetilde{\lambda}_1 = -1$.

For system (c) the controller equations are $u_1(k) = x_2(k), u_2(k) \equiv 1$, and the periodic sequence of the states is

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \to \begin{bmatrix} 0 \\ 1 \end{bmatrix} \to \begin{bmatrix} 0 \\ 0 \end{bmatrix} \cdots$$

whose period $k_p = 2$ corresponds to the eigenvalue $\widetilde{\lambda}_2 = -1$.

For system (d) the controller equations are $u_1(k) = x_1(k), u_2(k) \equiv 1$, and the periodic sequence of the states is

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \to \begin{bmatrix} 0 \\ 1 \end{bmatrix} \to \begin{bmatrix} 1 \\ 1 \end{bmatrix} \to \begin{bmatrix} 1 \\ 0 \end{bmatrix} \to \begin{bmatrix} 0 \\ 0 \end{bmatrix} \cdots$$

again with the period $k_p = 4$ and the eigenvalues $\widetilde{\lambda}_{1/2} = \pm j$.

It is emphasized that there is no need to construct the reachability graph in order to apply the method proposed above. On the other hand, the reachability graph allows verification of the results. Figure 1 shows the reachability graph for the above example. It exhibits six possible cycles, four of which have been found above. The remaining two cycles cannot be modelled by arithmetically *linear* automata.

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \underset{u_2=0}{\overset{u_1=1}{\rightleftarrows}} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$u_2 = 1 \downarrow\uparrow u_1 = 1 \qquad u_1 = 0 \downarrow\uparrow u_2 = 1$$

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} \underset{u_2=0}{\overset{u_1=0}{\rightleftarrows}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Fig. 1. Reachability graph.

7. Conclusions

An approach to sequential finite state machines has been proposed based on arithmetic representations of Boolean functions. This allows bridging the gap between the discrete-event dynamic systems and the classical discrete-time systems since the same algebra is used. Rule-based control of finite automata has been formulated within the framework of the classical control theory. As a special aspect, global linearization via feedback control has been considered for binary dynamic systems whose arithmetic state equations are linear with respect to the controls.

References

Bochmann D. (1975): *Einführung in die strukturelle Automatentheorie.* — Berlin: VEB Verlag Technik.

Booth T.L. (1967): *Sequential Machines and Automata Theory.* — New York: J. Wiley and Sons.

Franke D. (1992): *A novel approach to finite automata with control application to transport systems.* — Proc. 1st Int. Conf. *Intelligent Systems Engineering*, Edinburgh, UK, IEE Conf. Publ., No.36, pp.71–76.

Franke D. (1993): *Global linearization of finite state machines via feedback control.* — Prep. 12th IFAC World Congress, Sydney, Australia, v.4, pp.157–160.

Franke D. (1994): *Sequentielle Systeme-Binäre und Fuzzy Automatisierung mit arithmetischen Polynomen.* — Wiesbaden: Vieweg-Verlag.

Reinschke K. (1973): *Zuverlässigkeit von Systemen*, Band 1: *Systeme mit endlich vielen Zuständen.* — Berlin: VEB Verlag Technik.

Shegalkin I.I. (1928): Matemat. sbornik v.35, pp.311–377.

Störmer H. (1970): *Mathematische Theorie der Zuverlässigkeit.* — Berlin: Akademie-Verlag.

Received: November 27, 1994