MODELING AND ANALYZING
FINITE STATE AUTOMATA IN THE FINITE FIELD $\mathbb{F}_2$

J. Reger and K. Schmidt, University of Erlangen-Nuremberg, Germany
Corresponding Author: J. Reger
Institute of Automatic Control
Department of Electrical, Electronic and Communication Engineering
Friedrich-Alexander-Universität Erlangen-Nürnberg
Cauerstraße 7, 91058 Erlangen, Germany
Phone: +49 9131-8527134, Fax: +49 9131-8528715
email: reger@ieee.org

Abstract. A method for determining multilinear state space models for general finite state automata is presented. The obtained model resides on $\mathbb{F}_2$, the finite field of characteristic 2 with the operations addition and multiplication, both carried out modulo 2. It is functionally complete in the sense that it is capable of describing all finite state automata, including non-deterministic and partially defined automata. For those cases in which the model over $\mathbb{F}_2$ is linear, means for a complete analysis of the cyclic behavior of these automata are recalled. With respect to these linear models, the cyclic structure of the state space is shown to be determined only by the periods of the elementary divisor polynomials of the system dynamics. An example illustrates the analysis procedure.

Keywords. Finite State Automata, Linear Modular Systems, Finite Fields, Feedback Shift Registers

1 Introduction

The investigation of the cyclic transition behavior is considered a major task within the analysis of finite state automata. Reminiscent of eigenvalues in linear continuous systems, linear state space models using *arithmetical polynomials* have been established for deterministic finite state automata in order to ascertain the cyclic structure of the state space by setting the eigenvalues of the closed loop system dynamics [3]. The main drawback of this approach is the lack of sufficient and efficient criteria for pointing out the existence of particular cyclic subspaces. Even for linear systems this approach provides just necessary criteria such that for a fixed set of eigenvalues some cyclic structures may be derived that only occur virtually. Another drawback is that the multiplicity of eigenvalues has no significance within this framework, and thus has not been investigated. Getting away with the latter problems a new method employing *Walsh functions* was introduced [7], so as to determine the cyclic structure of the state space, completely leaving aside the eigenvalues of the system dynamics and referring to the state equation only. The crucial disadvantage remains: the complexity problem, since solving for certain cyclic states is NP-complete in these settings. This originates from the fact that solving a linear diophantine system of equations for *boolean solutions only* (e.g. cyclic states) is on the class of NP-complete problems. There is no polynomial time algorithm that constructs boolean vectors out of a linear combination of integral or rational vectors. Hence, typical problems in practice, which usually comprise an enormous number of states, have to be considered intractable within these approaches. A further drawback is the missing implementation of input variables in the *Walsh function* framework. Thus the *Walsh function* method does not provide any feedback design. An other shortcoming is its lacking describability of non-determinism, a characteristic property of automata. However, for *arithmetical polynomials* some promising steps for adapting the method to non-determinism have been made [2].

In contrast to the latter, the model to be developed in this paper allows of an efficient analysis of the cyclic behavior for deterministic and non-deterministic automata and is capable of overcoming the afore mentioned obstacles using an algebraic state space description that is formulated strictly in (modulo 2-) operations on the set of boolean numbers, that is the setting resides on the finite field $\mathbb{F}_2$. Finite field models have already been under consideration in the control community [4]. However, neither were they utilized for determining the cyclic structure of automata nor were any analogies drawn to linear continuous time systems. On the other hand concerning linear systems much of the theory was already developed as early as the sixties — for instance the design of linear feedback shift registers [5] — but has not been adapted for control purposes yet. In this contribution, based on the system invariants of a linear system, in particular the elementary divisor polynomials of the system dynamics, the finite field framework is shown to enable the statement of sufficient criteria for determining all cycles of a deterministic automaton (in multiplicity and length). As a consequence, the feedback design problem specifying the cyclic

structure of a controlled linear automaton becomes feasible. In the general, multilinear case this can be done using Gröbner-bases. Specially for linear systems of equations this kind of modeling admits of solving for cyclic states in polynomial complexity, for example by employing the Gauß-algorithm.
The outline of the paper is as follows: Section 2 introduces some necessary algebraic terminology, e. g. finite fields, polynomials over finite fields, invariants. By relating boolean algebra to finite fields two methods for obtaining the multilinear automaton model are presented in Section 3. The analysis of linear systems over $\mathbb{F}_2$, linear modular systems, is dealt with in Section 4. An example illustrates the analysis method. Finally Section 5 gives some hints of how to use and extend the setting.

2 Algebraic Preliminaries

In engineering sciences discrete mathematics and the algebraic fundamentals of finite field theory usually are of minor importance. On this account, some indispendsable algebraic concepts are to be prearranged, which form the base of the automaton model of Section 3. Some remarks spot the differences between finite and infinite fields. For a comprehensive but thorough introduction to finite fields refer to [8].

2.1 Finite Fields

From [8] we recall some basic definitions.

Definition 2.1 (Groups) *A group is a set G together with a binary operation $*$ such that*

1. *For all $a, b \in G$, $a * b \in G$.*
2. *The operation $*$ is associative, i. e. $a * (b * c) = (a * b) * c$ for any $a, b, c \in G$.*
3. *There exists an identity element e such that for all $a \in G$, $a * e = e * a = a$.*
4. *There exists an inverse element $a^{-1} \in G$ for each $a \in G$ such that $a * a^{-1} = a^{-1} * a = e$.*

Moreover, a group is commutative (or abelian) if for all $a, b \in G$, $a * b = b * a$. A group is called finite if the set G contains finitely many elements.

Definition 2.2 (Field) *A set $\mathbb{F}$ with the operations addition and multiplication, $+$ and $\cdot$, is a field if*

1. *$\mathbb{F}$ is a commutative group with respect to addition.*
2. *$\mathbb{F} \setminus \{0\}$ is a commutative group with respect to multiplication.*
3. *$\mathbb{F}$ is distributive with respect to addition and multiplication, that is*
   $a \cdot (b + c) = a \cdot b + a \cdot c$ and $(b + c) \cdot a = b \cdot a + c \cdot a$ for all $a, b, c \in \mathbb{F}$.

A field $\mathbb{F}$ with q elements, denoted by $\mathbb{F}_q$, is called finite if it contains finitely many elements.

In subsequent sections of the paper a special type of field is utilized that is based on the division remainder operation *modulo*.

Definition 2.3 (Galois-Field) *The set of integral numbers $\{0, 1, \ldots, q-1\}$, where q is a prime number, with operations addition and multiplication modulo q, is a finite field called Galois-Field $\mathbb{F}_q$.*

Remark. The primality of q is decisive for the existence of a multiplicative inverse element in general. Otherwise zero divisors occur, as for instance $2 \cdot 3$ modulo $6 = 0$, hence there is no field $\mathbb{F}_6$.

Beginning with Section 3 we will concentrate on Galois-Fields $\mathbb{F}_2$ only. Consequently addition and multiplication implicitly will be understood modulo 2 and subtraction coincides with addition.

Theorem 2.1 (Fermat's Little Theorem) *Let $q \in \mathbb{Z}$ be prime. Then for all integers λ, which are not divisible by q, q divides $\lambda^{q-1} - 1$.*

Corrolary 2.1 *Every $\lambda \in \mathbb{F}_q$ satisfies $\lambda^q = \lambda$.*

Hence, a polynomial $p \in \mathbb{F}_q[\lambda]$ over a finite field $\mathbb{F}_q$, where $\mathbb{F}_q[\lambda]$ denotes the ring of polynomials with coefficients in $\mathbb{F}_q$, can be identical to zero for arbitrary $\lambda \in \mathbb{F}_q$, since p may contain zero-polynomials $\lambda^q - \lambda$. In contrast to finite fields, a polynomial $p \in \mathbb{R}[\lambda]$ over the infinite field of real numbers $\mathbb{R}$ is identical to zero if and only if all coefficients are zero.

2.2 Polynomials over Finite Fields

2.2.1 Factorization of Polynomials

By Gauß' well-known fundamental theorem of algebra all polynomials over the field of real numbers $\mathbb{R}$ can be factorized (reduced) in quadratical factors in $\mathbb{R}[\lambda]$. As the real numbers refer to an infinite field this need not be the case for finite fields $\mathbb{F}_q$, which will be shown in the following.

Definition 2.4 (Monic Polynomial) *A polynomial $p(\lambda) = \sum_{i=0}^{d} a_i \lambda^i$ and degree d is called monic if $a_d = 1$.*

Definition 2.5 (Irreducible Polynomial) *A non-constant polynomial $p \in \mathbb{F}[\lambda]$ is called irreducible over $\mathbb{F}$ if whenever $p(\lambda) = g(\lambda)h(\lambda)$ in $\mathbb{F}[\lambda]$ then either $g(\lambda)$ or $h(\lambda)$ is a constant.*

Theorem 2.2 (Unique Factorization Theorem) *Any polynomial $p \in \mathbb{F}[\lambda]$ can be written in the form*

$$p = a\, {p_1}^{e_1} \cdots {p_k}^{e_k}\,, \tag{1}$$

where $a \in \mathbb{F}$, $p_1, \ldots, p_k$ are distinct monic irreducible polynomials in $\mathbb{F}[\lambda]$, and $e_1, \ldots, e_k$ are positive integers. Moreover, this factorization is unique apart from the order of the factors.

Remark. For the field $\mathbb{F} = \mathbb{R}$ all polynomials p_i in Theorem 2.2 are of at most second degree. This does not hold for a finite field, for example: $p(\lambda) = \lambda^5 + \lambda^2 + \lambda + 1 = (\lambda^3 + \lambda + 1)(\lambda + 1)^2$, $p \in \mathbb{F}_2[\lambda]$, because $\lambda^3 + \lambda + 1$ and $\lambda + 1$ are irreducible over $\mathbb{F}_2$. However, $\lambda^3 + \lambda + 1 = (\lambda^2 + \lambda + 2)(\lambda + 2)$ in $\mathbb{F}_3[\lambda]$, that is reducibility depends on the field.

2.2.2 Period of Polynomials

Unlike polynomials over real numbers, polynomials over finite fields show a periodicity property.

Definition 2.6 (Period of a Polynomial) *Let $p \in \mathbb{F}_q[\lambda]$ be a non-zero polynomial. If $p(0) \neq 0$, then the least positive integer τ for which $p(\lambda)$ divides $\lambda^\tau - 1$ is called the period of the polynomial p. If $p(0) = 0$, then $p(\lambda) = \lambda^h g(\lambda)$, where $h \in \mathbb{N}$ and $g \in \mathbb{F}_q[\lambda]$ with $g(0) \neq 0$, and τ is defined as the period of g.*

Theorem 2.3 (Period of a Multiple Polynomial) *Let $p \in \mathbb{F}_q[\lambda]$ be irreducible over $\mathbb{F}_q$ with $p(0) \neq 0$ and period τ. Let $f \in \mathbb{F}_q[\lambda]$ be $f = p^k$ with $k \in \mathbb{N}$. Let l be the least $l \in \mathbb{Z}$ such that $q^l \geq k$. Then the multiple polynomial p has the period $q^l \tau$.*

Remark. Nilpotent polynomials $p \in \mathbb{F}_q[\lambda]$ with $p = \lambda^k$ for some $k \in \mathbb{N}$ are not periodic by definition. Hence, polynomials over finite fields are either periodic or nilpotent.

In practice, periods of polynomials do not have to be calculated. They can be found in tabulars like in [8], or are internally tabulated in computer algebra software like Maple or Mathematica.

2.3 Invariants of Linear Systems over the finite field $\mathbb{F}_q$

2.3.1 Similarity of Matrices

The major properties of a matrix reside in its structural invariants. These are preserved under so-called similarity transforms. Some definitions prepare the introduction of the smith normal form of a matrix.

Definition 2.7 (Similarity of a Matrix) *Matrices $\mathbf{A}_1$, $\mathbf{A}_2 \in \mathbb{F}^{n\times n}$ are similar if for some invertible matrix $\mathbf{T} \in \mathbb{F}^{n\times n}$*

$$\mathbf{A}_1 = \mathbf{T}^{-1}\mathbf{A}_2\mathbf{T}. \tag{2}$$

Definition 2.8 (Rational Matrix) *A matrix $\mathbf{R}(\lambda)$, the elements of which are fractions of polynomials over a field $\mathbb{F}[\lambda]$ is called a rational matrix. If the denominator polynomial of each element of $\mathbf{R}(\lambda)$ is equal to one, the matrix is a polynomial matrix.*

Definition 2.9 (Unimodular Matrix) *If the determinant of a polynomial matrix is a scalar in the underlying field $\mathbb{F}$, the matrix is called unimodular.*

Theorem 2.4 (Smith Form of a Matrix) *For any* $\mathbf{A} \in \mathbb{F}_q^{n \times n}$ *there exist unimodular matrices* $\mathbf{U}(\lambda)$ *and* $\mathbf{V}(\lambda)$ *such that*

$$\mathbf{U}(\lambda)(\lambda\mathbf{I} - \mathbf{A})\mathbf{V}(\lambda) = \mathbf{S}(\lambda) \tag{3}$$

with

$$\mathbf{S}(\lambda) = \begin{bmatrix} c_1(\lambda) & 0 & \cdots & 0 \\ 0 & c_2(\lambda) & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & c_n(\lambda) \end{bmatrix}, \tag{4}$$

where the monic polynomials $c_{i+1}|c_i,\ i = 1, \ldots, n-1$. *The polynomial matrix* $\mathbf{S}(\lambda)$ *is called the smith (normal) form of* $\mathbf{A}$.

Matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ are similar if and only if they have the same smith form. Since the polynomials $c_i(\lambda)$ are preserved under similarity transforms this gives rise to a further definition.

2.3.2 Invariant Polynomials

Definition 2.10 (Similarity Invariants) *The monic polynomials* $c_i(\lambda),\ i = 1, \ldots, n$ *referring to the smith form* $\mathbf{S}(\lambda)$ *of a matrix* $\mathbf{A}$ *are the similarity invariants of* $\mathbf{A}$.

Note that the uppermost polynomial $c_1(\lambda)$ is the minimal polynomial of the dynamics $\mathbf{A}$. The product of all similarity invariants is its characteristic polynomial $\det(\lambda\mathbf{I} - \mathbf{A})$.

Definition 2.11 (Elementary Divisor Polynomials) *The unique irreducible factor polynomials* $p_j(\lambda)$ *of each* $c_i(\lambda),\ i = 1, \ldots, n$ *referring to the smith form* $\mathbf{S}(\lambda)$ *of* $\mathbf{A}$ *are called elementary divisor polynomials of* $\mathbf{A}$.

Remark. We did not mention the jordan normal form of a matrix. The reason is that the jordan normal form is accompanied by the notion of an extension field $\mathbb{F}_{q^k}, k = 1, 2, \ldots$ of $\mathbb{F}_q$. In case of a finite field, the calculation of roots in the associated extension field $\mathbb{F}_{q^k}$ is much more cumbersome than it is in the field extension associated to the real numbers, which is $\mathbb{C}$, the field of complex numbers.

3 Multilinear Automaton Model over the Finite Field $\mathbb{F}_2$

In this section we develop an algebraic model for a non-deterministic finite state automaton with multiple inputs. It takes the form

$$f(\mathbf{x}[k+1], \mathbf{x}[k], \mathbf{u}[k]) = 0, \quad \mathbf{x} \in \{0,1\}^n, \mathbf{u} \in \{0,1\}^p, \tag{5}$$

where f marks an implicit scalar transition function over the finite field $\mathbb{F}_2$, which relates the n states $\mathbf{x}[k]$ and the p inputs $\mathbf{u}[k]$ in an instant k with the possibly multiple successor states $\mathbf{x}[k+1]$ in instant $k+1$. The transition function is multilinear in the elements of $\mathbf{x}[k+1]$, $\mathbf{x}[k]$ and $\mathbf{u}[k]$ except for a constant.

3.1 The Relation of Boolean Algebra and the Finite Field $\mathbb{F}_2$

Some boolean algebra is required for calculating the automaton model over finite fields. Therefore the necessary basics of boolean algebra are recalled for convenience; a concise introductory is given by [1].

Definition 3.1 (Boolean Operations) *Given the set* $\mathbb{B} = \{0,1\}$. *Then the operations AND ”*$\wedge$*”, OR ”*$\vee$*”, XOR ”*$\oplus$*” and NOT ”*$\bar{\ }$*”are defined on* $\mathbb{B}$ *as follows:*

x_1	x_2	$x_1 \wedge x_2$
0	0	0
0	1	0
1	0	0
1	1	1

x_1	x_2	$x_1 \vee x_2$
0	0	0
0	1	1
1	0	1
1	1	1

x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

x	$\bar{x}$
0	1
1	0

Boolean operations form boolean functions, which usually are expressed in normal forms. Those admit an easier decomposition into subfunctions.

Definition 3.2 (Disjunctive Normal Form) *Let f be a boolean function of indeterminates $x_1,\dots,x_n \in \mathbb{B} = \{0,1\}$ and $\mathbf{c}$ be a vector $\in \mathbb{B}^n$. Then the disjunctive normal form of f is*

$$f(x_1,\dots,x_n) = \bigvee_{\mathbf{c}\in\mathbb{B}^n} f(\mathbf{c}) \wedge \bigwedge_{i=1}^{n} (x_i \oplus c_i) \; . \tag{6}$$

Example: The disjunctive normal form of $f(x_1,x_2) = x_1 \oplus x_2$ is

$$\begin{aligned} f(x_1,x_2) \;=\; & \Big(f(0,0)\wedge(x_1\oplus 0)\wedge(x_2\oplus 0)\Big) \vee \Big(f(0,1)\wedge(x_1\oplus 0)\wedge(x_2\oplus 1)\Big) \vee \Big(f(1,0)\wedge(x_1\oplus 1)\wedge(x_2\oplus 0)\Big) \vee \\ & \Big(f(1,1)\wedge(x_1\oplus 1)\wedge(x_2\oplus 1)\Big) = \Big((x_1\wedge(x_2\oplus 1)\Big) \vee \Big((x_1\oplus 1)\wedge x_2\Big) = (x_1\wedge\overline{x_2})\vee(\overline{x_1}\wedge x_2)\; . \end{aligned} \tag{7}$$

Instead of introducing all boolean operations from Definition 3.1 it is sufficient to confine oneself to the operations $\oplus$ and $\wedge$. This can be done by using DeMorgan's Law and observing $\overline{x} = 1\oplus x$, hence by

$$x_1 \vee x_2 = \overline{\overline{x_1}\wedge\overline{x_2}} = 1\oplus((1\oplus x_1)\wedge(1\oplus x_2)) = x_1\oplus x_2\oplus x_1x_2, \quad x_1,x_2\in\mathbb{B}\, .$$

Thus, if the operations XOR and AND on the set $\mathbb{B} = \{0,1\}$ are identified with addition modulo 2 and multiplication modulo 2 on the field $\mathbb{F}_2$ then the following important theorem can be stated.

Theorem 3.1 (Isomorphism of $\mathbb{F}_2$ and $\mathbb{B}$) *The set $\mathbb{B} = \{0,1\}$ with the operations $+ := \oplus$ and $\cdot := \wedge$ is a finite field. The finite field $\mathbb{B}$ is isomorphic to the Galois-Field $\mathbb{F}_2$.*

Since any boolean function f can be manipulated so as to obtain a polynomial in $\oplus$ and $\wedge$ only, the calculation of the finite field representation of f over $\mathbb{F}_2$ amounts to simply interchange $\oplus$ by $+$ and $\wedge$ by $\cdot$, respectively (from now on all additions and multiplications taken modulo 2). Then for $x_1,x_2 \in \{0,1\}$ the following applies:

$$\begin{aligned} x_1\wedge x_2 &\iff x_1x_2 \qquad (8)\\ x_1\vee x_2 &\iff x_1+x_2+x_1x_2 \qquad (9)\\ x_1\oplus x_2 &\iff x_1+x_2 \qquad (10)\\ \overline{x} &\iff 1+x \qquad (11) \end{aligned}$$

These equivalences are of major significance in the next sections.

3.2 Deriving the Algebraic Model by Use of the Disjunctive Normal Form

In the following, a single input example is to introduce the main steps for obtaining the transition function for a non-deterministic automaton over the finite field $\mathbb{F}_2$. The underlying algorithm can be generalized easily and is left out for clearness.

Consider the automaton depicted in Figure 1. The nodes are coded by binary vectors, which represent values for the states $\mathbf{x}^{\mathrm{T}} = (x_1,x_2)$. Arcs connect the states and indicate possible transitions between the states. Marked arcs denote that the transition is possible only if a certain condition on the input variables u is satisfied (here $u = 1$). If no marking is specified on an arc a transition is possible for any choice of inputs. In general, leaving arcs do not determine unique successor states, i. e. the automaton is non-deterministic.

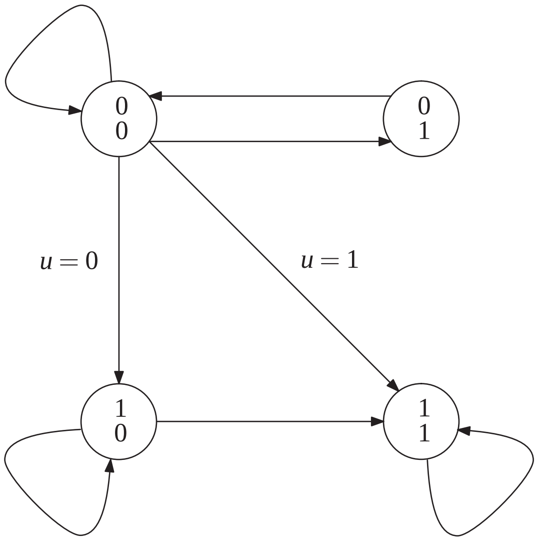
We will omit the symbol k in the denotation of $x_i[k]$ and $u[k]$ and abbreviate $x_i[k+1]$ by x_i' and represent the logical interconnection of states x_1,x_2, input u and successor states x_1',x_2' by a state table (right hand side of Figure 1). Regarding each row in the state table, a function value of $f_c = 1$ signifies that a transition is possible, $f_c = 0$ indicates that not.

Therefore, in view of Definition 3.2 and with (7) the disjunctive normal form of the function f with respect to the state table of Figure 1 reads

$$\begin{aligned} f(x_1'x_2',x_1,x_2,u) \;=\; & \overline{u}\overline{x}_2'\overline{x}_1'\overline{x}_2\overline{x}_1 \;\vee\; \overline{u}\overline{x}_2'\overline{x}_1'x_2\overline{x}_1 \;\vee\; \overline{u}\overline{x}_2'x_1'\overline{x}_2\overline{x}_1 \;\vee\; \overline{u}\overline{x}_2'x_1'\overline{x}_2x_1 \;\vee\; \overline{u}x_2'\overline{x}_1'\overline{x}_2\overline{x}_1 \;\vee \\ & \overline{u}x_2'x_1'\overline{x}_2x_1 \;\vee\; \overline{u}x_2'x_1'x_2x_1 \;\vee\; u\overline{x}_2'\overline{x}_1'\overline{x}_2\overline{x}_1 \;\vee\; u\overline{x}_2'\overline{x}_1'x_2\overline{x}_1 \;\vee\; u\overline{x}_2'x_1'\overline{x}_2x_1 \;\vee \\ & ux_2'\overline{x}_1'\overline{x}_2\overline{x}_1 \;\vee\; ux_2'x_1'\overline{x}_2\overline{x}_1 \;\vee\; ux_2'x_1'\overline{x}_2x_1 \;\vee\; ux_2'x_1'x_2x_1 \;=\; 1\,, \end{aligned} \tag{12}$$

$$\begin{aligned} \Leftrightarrow \quad f(x_1'x_2',x_1,x_2,u) \;=\; & \overline{x}_2'\overline{x}_1'\overline{x}_2\overline{x}_1 \;\vee\; \overline{x}_2'\overline{x}_1'x_2\overline{x}_1 \;\vee\; \overline{u}\overline{x}_2'x_1'\overline{x}_2\overline{x}_1 \;\vee\; \overline{x}_2'x_1'\overline{x}_2x_1 \;\vee\; x_2'\overline{x}_1'\overline{x}_2\overline{x}_1 \;\vee \\ & x_2'x_1'\overline{x}_2x_1 \;\vee\; x_2'x_1'x_2x_1 \;\vee\; ux_2'x_1'\overline{x}_2\overline{x}_1 \;=\; 1\,, \end{aligned} \tag{13}$$

where we used the abbreviation $a\wedge b = ab$.



u	x_2'	x_1'	x_2	x_1	f_c
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	0	1	0
0	1	0	1	0	0
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	0	1	1
0	1	1	1	0	0
0	1	1	1	1	1
1	0	0	0	0	1
1	0	0	0	1	0
1	0	0	1	0	1
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	0	1	1
1	0	1	1	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	0	1	0
1	1	0	1	0	0
1	1	0	1	1	0
1	1	1	0	0	1
1	1	1	0	1	1
1	1	1	1	0	0
1	1	1	1	1	1

Figure 1: Graph of an example automaton (above) and its state table (right hand side). The column marked f_c signifies whether a transition from $(x_1,x_2)^{\mathrm{T}}$ to $(x_1',x_2')^{\mathrm{T}}$ under input u is possible and vice versa.

Observe that by DeMorgan's law

$$a_1 \vee a_2 \vee \ldots \vee a_k = 1 \quad \Longleftrightarrow \quad (1 \oplus a_1) \wedge (1 \oplus a_2) \wedge \ldots \wedge (1 \oplus a_k) = 0 \tag{14}$$

all disjunctions $\vee$ can be eliminated. The remaining negations in (13) vanish by setting $\overline{a} = a \oplus 1$. As a result we get a function consisting of the operations $\wedge$ and $\oplus$ only. Thus, via (8) and (10) we finally obtain the representation of the transition function in the finite field $\mathbb{F}_2$

$$\begin{aligned} f(x_1'x_2',x_1,x_2,u) &= \big(1+(1+x_2')(1+x_1')(1+x_2)(1+x_1)\big)\big(1+(1+x_2')(1+x_1')x_2(1+x_1)\big)\ldots \\ &\quad \big(1+x_2'x_1'x_2x_1\big)\big(1+ux_2'x_1'(1+x_2)(1+x_1)\big) = 0\,, \end{aligned} \tag{15}$$

$$\begin{aligned} \Longleftrightarrow \quad f(x_1'x_2',x_1,x_2,u) &= x_1+x_1x_1'+x_2x_1'+x_2x_2'+x_1x_2x_2'+x_1'x_2'+x_1x_1'x_2'+x_1x_2x_1'x_2'+ \\ &\quad x_1'u+x_1x_1'u+x_2x_1'u+x_1x_2x_1'u = 0\,. \end{aligned} \tag{16}$$

3.3 Simplifications Using Reed-Muller Generator Matrices

The regular tabulation of the state table in Figure 1 — binary counting, row by row — allows a much more efficient calculation technique of the transition function f. So-called Reed-Muller codes, well-known from linear coding theory, exploit this property; for further details see [6].
Consider the recursively defined Reed-Muller generator matrices

$$\mathbf{G}_n \overset{\text{def}}{=} \begin{pmatrix} \mathbf{G}_{n-1} & \mathbf{0} \\ \mathbf{G}_{n-1} & \mathbf{G}_{n-1} \end{pmatrix}, \quad \mathbf{G}_0 \overset{\text{def}}{=} 1\,. \tag{17}$$

Then referring to [6] we have the simple matrix–vector product over $\mathbb{F}_2$

$$\mathbf{c}_{2n+p} = \mathbf{G}_{2n+p}\mathbf{f}_c\,, \tag{18}$$

in which $\mathbf{c}_{2n+p}$ is the $(2^{2n+p},1)$-vector of coefficients associated to a particular tabulation of monomials in a $(2^{2n+p},1)$-vector $\boldsymbol{\varphi}_{2n+p}$. In general, $\boldsymbol{\varphi}_{2n+p}$ contains all monomials of the n states x_i, of the p inputs u_i and of the next states x_i'. The vector $\mathbf{f}_c$ is the $(2^{2n+p},1)$-vector with respect to the rightmost column in the state table of Figure 1. Using equation (18) the demanded transition function then reads

$$f(x_1',x_2',\ldots,x_n',x_1,x_2,\ldots,x_n,u_1,u_2,\ldots,u_p) = \mathbf{c}_{2n+p}^{\mathrm{T}}\boldsymbol{\varphi}_{2n+p} + 1 = 0\,. \tag{19}$$

It remains to explain how to tabulate the monomials in $\boldsymbol{\varphi}_{2n+p}$. We fall back on the example of Figure 1 with $n = 2$ states und $p = 1$ inputs. In this case we have

$$\mathbf{G}_5 = \left(\begin{array}{cccccccccccccccccccccccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}\right), \qquad \boldsymbol{\varphi}_5 = \left(\begin{array}{l}
1 \\ x_1 \\ x_2 \\ x_1x_2 \\ x_1' \\ x_1x_1' \\ x_2x_1' \\ x_1x_2x_1' \\ x_2' \\ x_1x_2' \\ x_2x_2' \\ x_1x_2x_2' \\ x_1'x_2' \\ x_1x_1'x_2' \\ x_2x_1'x_2' \\ x_1x_2x_1'x_2' \\ u \\ x_1u \\ x_2u \\ x_1x_2u \\ x_1'u \\ x_1x_1'u \\ x_2x_1'u \\ x_1x_2x_1'u \\ x_2'u \\ x_1x_2'u \\ x_2x_2'u \\ x_1x_2x_2'u \\ x_1'x_2'u \\ x_1x_1'x_2'u \\ x_2x_1'x_2'u \\ x_1x_2x_1'x_2'u
\end{array}\right) \qquad (20)$$

with the Reed-Muller generator matrix $\mathbf{G}_5$ and the vector of monomials $\boldsymbol{\varphi}_5$. The tabulation regarding the elements of $\boldsymbol{\varphi}_5$ is obviously recursive: if we start the inspection from the top then every new variable is set left to a copy of the former part of the vector, and so on. Substitution of (20) in (18) and (19) easily verifies the result from (16).

3.4 Enhancements and Generalizations

Abstracting from the latter example under discussion, in general we obtain a multilinear transition function

$$f(\mathbf{x}[k+1], \mathbf{x}[k], \mathbf{u}[k]) = 0 = \sum_{S_1 \in 2^N} \sum_{S_2 \in 2^N} \sum_{S_3 \in 2^P} \delta^{S_1,S_2,S_3} \Big(\prod_{j \in S_1} x_j[k+1] \Big) \Big(\prod_{l \in S_2} x_l[k] \Big) \Big(\prod_{m \in S_3} u_m[k] \Big), \qquad (21)$$

where the sets $N = \{1, 2, \ldots, n\}$, $P = \{1, 2, \ldots, p\}$ are index sets, 2^N denotes the (possibly empty) power set of N and δ^{S_1,S_2,S_3} are constants $\in \{0, 1\}$.

3.4.1 Additional States

Further states and inputs may be added by concatenating the state table on the left with the respective columns of the new states and inputs. Using the associated, bigger Reed-Muller generator matrices the calculation of the monomial coefficients still amounts to the same procedure. Note that only the coefficients referring to the new variables need to be calculated, the coefficients of the former representation are left unchanged; a chief advantage of the Reed-Muller generator matrix technique.

3.4.2 Partially Defined Transition Functions

In this context, a partially defined transition function is a function that is defined on a proper subspace $X \subset \mathbb{F}_2^n$. As a consequence, only a few rows may be defined in the entire state table. To this account a check function $r(x_1, \ldots, x_n) = 0$ can be introduced in the same manor as f_c. The value of the check function r is equal 0 if the state is defined and 1 elsewhere. After all, the check function r and the transition function f can be combined in one single equation.

3.4.3 Determinism

In case of deterministic automata the state tables can be reshaped as illustrated in Figure 2. Thus, by employing the methods of Section 3.2 and 3.3 a state equation for each state variable x_i, $i = 1, \ldots, n$ can be determined in explicit form, which results in

$$\mathbf{x}[k+1] = \mathbf{f}(\mathbf{x}[k], \mathbf{u}[k]), \quad \mathbf{x} \in \{0,1\}^n, \mathbf{u} \in \{0,1\}^p \tag{22}$$

and reminds of a discrete time system in the continuous world.

u_p	$\cdots$	u_2	u_1	x_n	$\cdots$	x_2	x_1	x'_n	$\cdots$	x'_2	x'_1
0	$\cdots$	0	0	0	$\cdots$	0	0	$f_{n,1}$	$\cdots$	$f_{2,1}$	$f_{1,1}$
0	$\cdots$	0	0	0	$\cdots$	0	1	$f_{n,2}$	$\cdots$	$f_{2,2}$	$f_{1,2}$
0	$\cdots$	0	0	0	$\cdots$	1	0	$f_{n,3}$	$\cdots$	$f_{2,3}$	$f_{1,3}$
$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
1	$\cdots$	1	1	1	$\cdots$	1	1	$f_{n,2^{n+p}}$	$\cdots$	$f_{2,2^{n+p}}$	$f_{1,2^{n+p}}$

Figure 2: Typical shape of a state table regarding a deterministic automaton

4 Linear Modular Systems over $\mathbb{F}_2$

By means of an autonomous deterministic linear system

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k], \quad \mathbf{x} \in \{0,1\}^n\ , \tag{23}$$

a so-called *linear modular system* (LMS) with matrix $\mathbf{A} \in \mathbb{F}_2^{n \times n}$, called dynamics, the modeling power of the finite field framework shall be examined. With regard to these systems the analysis for cyclic (periodic) states is carried out, briefly recalling some results from [5, 9]. The properties of finite fields and polynomials over finite fields, which have been presented in Section 2, will provide the necessary key concepts for solving the analysis problem.

4.1 Cycle Sum of a Linear Modular System

The state space of an LMS typically decomposes in aperiodic and periodic subspaces. It is clear that in the autonomous case any information must be included in the structural invariants of the dynamics **A**. Thus, there we may search for information about periodic states. Periodic states are constituted by the following definition.

Definition 4.1 (Period of States) *The period of a state* $\mathbf{x}[k] \in \mathbb{F}_q^n$ *is the least* $\tau \in \mathbb{N}$ *such that* $\mathbf{x}[k+\tau] = \mathbf{x}[k]$*.*

Generally, state spaces decompose in more than one periodic subspace. Let the number of different-length cycles be N. All occurring subspace periodicities can be written in a more convenient form by applying

Definition 4.2 (Cycle Sum) *The cycle sum* Σ *is the formal sum of cycle terms*

$$\Sigma = \nu_1[\tau_1] \dotplus \nu_2[\tau_2] \dotplus \ldots \dotplus \nu_N[\tau_N]\,, \tag{24}$$

where ν_i *is the number of cycles of length* τ_i *and* $\dotplus$ *satisfies the relation* $\nu_i[\tau] \dotplus \nu_j[\tau] = (\nu_i + \nu_j)[\tau]$ *on the cycle terms.*

Definition 4.3 (Product of Cycle Terms) *The product*

$$\nu_1[\tau_1]\nu_2[\tau_2] = \nu_1\nu_2 \operatorname{gcd}(\tau_1,\tau_2)[\operatorname{lcm}(\tau_1,\tau_2)] \tag{25}$$

is called cycle term product. The expressions $\operatorname{gcd}(\tau_1,\tau_2)$ *and* $\operatorname{lcm}(\tau_1,\tau_2)$ *are greatest common divisor and least common multiple of* τ_1*,* τ_2 *respectively.*

Theorem 4.1 (Superposition) *The cycle sum* Σ *superposing e cycle sums* Σ_i *can be calculated distributively by the product*

$$\Sigma = \Sigma_1 \Sigma_2 \cdots \Sigma_e\ . \tag{26}$$

For brevity the main theorem is recalled from [5, 9].

Theorem 4.2 (Cycle Sum of an Autonomous LMS) *Let $\mathbf{S}(\lambda)$ be the smith normal form of the dynamics of an autonomous LMS, P the set of factorized elementary divisor polynomials $p_i = (p_{i,\mathrm{irr}})^{e_i}$, where $p_{i,\mathrm{irr}}$ is an irreducible basis polynomial with $p_{i,\mathrm{irr}}(0) \neq 0$. Then each $p_i \in P$ contributes the cycle sum*

$$\Sigma_i = 1[1] \dot{+} \frac{2^{d_i}-1}{\tau_1^{(i)}}[\tau_1^{(i)}] \dot{+} \frac{2^{2d_i}-2^{d_i}}{\tau_2^{(i)}}[\tau_2^{(i)}] \dot{+} \ldots \dot{+} \frac{2^{e_i d_i}-2^{(e_i-1)d_i}}{\tau_{e_i}^{(i)}}[\tau_{e_i}^{(i)}]\,, \tag{27}$$

where d_i marks the degree of $p_{i,\mathrm{irr}}$ and $\tau_j^{(i)}$ denotes the period[1] *of $(p_{i,\mathrm{irr}})^j$. For the entire LMS the cycle sum Σ follows by superposition of all $|P\,|$ cycle sums Σ_i.*

Remark. From Theorem 4.2 we can conlude that nilpotent elementary divisor polynomials are not related to periodic subspaces.

Thus, the whole cycle sum of a linear modular system over $\mathbb{F}_2$ can be calculated along the following algorithm:

1. Calculate the smith normal form $\mathbf{S}(\lambda)$ of $\mathbf{A}$ by left and right transforms on $\lambda\mathbf{I}+\mathbf{A}$ using appropriate unimodular polynomial matrices.
2. Determine the elementary divisor polynomials p_i of $\mathbf{A}$ by factorizing the system invariants in $\mathbf{S}(\lambda)$.
3. Assign the periods $\tau_j^{(i)}$ to each polynomial $p_{i,\mathrm{irr}}^j$, $j = 1,\ldots,e_i$ with $p_i = (p_{i,\mathrm{irr}})^{e_i}$ and $p_{i,\mathrm{irr}}(0) \neq 0$, that is polynomials $p_i(\lambda) = \lambda^k, k \in \mathbb{N}$ need not be considered (see Remark).
4. Compute the cycle sum Σ_i with regard to each elementary divisor polynomial p_i.
5. The cycle sum Σ of the entire automaton then follows by superposing all cycle sets Σ_i.

4.2 Example

Consider the following linear modular system

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \xrightarrow{\mathbf{S}(\lambda)=\mathbf{U}(\lambda)(\lambda\mathbf{I}+\mathbf{A})\mathbf{V}(\lambda)} \quad \mathbf{S}(\lambda) = \begin{pmatrix} (\lambda^2+\lambda+1)(\lambda+1)^2 & 0 & 0 & 0 & 0 \\ 0 & \lambda+1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

where $\mathbf{A}$ is the dynamics and $\mathbf{S}(\lambda)$ is the smith normal form of $\mathbf{A}$. Then the only similarity invariants $\neq 1$ of matrix $\mathbf{A}$ are

$$c_1(\lambda) = (\lambda^2+\lambda+1)(\lambda+1)^2, \quad c_2(\lambda) = \lambda+1\,.$$

Hence, $\mathbf{A}$ has the elementary divisor polynomials

$$p_1(\lambda) = \lambda^2+\lambda+1, \quad p_2(\lambda) = (\lambda+1)^2, \quad p_3(\lambda) = \lambda+1\,,$$

the base polynomial degrees of which are $d_1 = 2$, $d_2 = 1$ and $d_3 = 1$, respectively. In view of Definition 2.6 and Theorem 2.3 we calculate the associated periods:

$$\begin{aligned} p_{1,\mathrm{irr}}(\lambda) = p_1(\lambda) | \lambda^3+1 \quad &\Longrightarrow \quad \tau_1^{(1)}{=}3 \\ p_{2,\mathrm{irr}}(\lambda) = \lambda+1 \quad &\Longrightarrow \quad \tau_1^{(2)}{=}1 \\ \big(p_{2,\mathrm{irr}}(\lambda)\big)^2 = (\lambda+1)^2 = \lambda^2+1 \quad &\Longrightarrow \quad \tau_2^{(2)}{=}2 \\ p_{3,\mathrm{irr}}(\lambda) = \lambda+1 \quad &\Longrightarrow \quad \tau_1^{(3)}{=}1 \end{aligned}$$

Theorem 4.2 yields

$$\Sigma_1 = 1[1] \dot{+} 1[3], \quad \Sigma_2 = 2[1] \dot{+} 1[2], \quad \Sigma_3 = 2[1]$$

[1] Beginning from here we are justified to have introduced the same symbol τ for the period of a state although firstly τ was introduced for the period of polynomials in Definition 2.6.

and by superposition according to Theorem 4.1 using (24) and (25) we get

$$\Sigma = \Sigma_1\Sigma_2\Sigma_3 = (1[1] \dotplus 1[3])(2[1] \dotplus 1[2])(2[1]) = (2[1] \dotplus 1[2] \dotplus 2[3] \dotplus 1[6])(2[1]) = 4[1] \dotplus 2[2] \dotplus 4[3] \dotplus 2[6]\,.$$

Therefore, the considered linear automaton described by the dynamics **A** comprises 4 cycles of length 1, 2 cycles of length 2, 4 cycles of length 3 and 2 cycles of length 6.

5 Conclusion

Beginning from a state table or coding scheme associated to a non-deterministic automaton an algebraic state space description has been developed. To this end, it was taken advantage of the logical interconnection between the automaton states, successor states and inputs. The first method invokes the calculation of the disjunctive normal form, elimination of negations and using the law of DeMorgan. Then a numerically improved, efficient procedure involving Reed-Muller generator matrices was presented. The result of both methods is a multilinear implicit transition function over the finite field $\mathbb{F}_2$. For the subclass of linear modular systems a method for analyzing the automaton's cycle sum was recalled. The method offers necessary and sufficient criteria for determining all automaton cycles in length and number. This is achieved using the invariant polynomials of the dynamics of the linear modular system. Further work will deal with setting the cyclic behavior by static state feedback. Nevertheless, almost all practically important cases are multilinear. These systems require more sophisticated, namely exact non-linear design methods. The notion of Gröbner-bases is considered to close this gap.

Acknowledgments

Research partially supported by Studienstiftung des deutschen Volkes and by Deutsche Forschungsgemeinschaft (DFG) under Grant No. RO 2262/3-1.

References

[1] D. Bochmann and C. Posthoff, Binäre Dynamische Systeme, Oldenbourg, Munich, 1981.

[2] D. Franke, Modelling Nondeterministic Discrete-Event Behaviour by Descriptor Systems, in: Proc. 3rd MATHMOD, Vienna, 2000.

[3] D. Franke, Sequentielle Systeme, Binäre und Fuzzy Automatisierung mit arithmetischen Polynomen, Vieweg, Braunschweig, 1994.

[4] R. Germundsson, Symbolic Systems — Theory, Computation and Applications, Linköping, 1995.

[5] A. Gill, Graphs of Affine Transformations, with Applications to Sequential Circuits, in: Proc. 7th IEEE International Symposium on Switching and Automata Theory, Berkeley (1966) 127–135.

[6] D. Hankerson et al., Coding Theory and Cryptography — The Essentials, Marcel Dekker Inc., New York, 2000.

[7] U. Konigorski, Modeling of Linear Systems and Finite Deterministic Automata by means of Walsh Functions, in: Proc. 3rd MATHMOD, Vienna, 2000.

[8] R. Lidl and H. Niederreiter, Introduction to Finite Fields and their Application, Cambridge Univ. Press, New York, 1994.

[9] J. Reger, Cycle Analysis for Deterministic Finite State Automata, in: Proc. 15th IFAC World Congress, Barcelona, 2002.