On $\alpha_2 - \nu_2$-products of automata

Pál Dömösi & György Maróti

Acta Informatica

ISSN 0001-5903
Volume 48
Combined 7-8

Acta Informatica (2011) 48:397-408
DOI 10.1007/s00236-011-0143-x

Springer

Acta Informatica (2011) 48:397–408
DOI 10.1007/s00236-011-0143-x

ORIGINAL ARTICLE

On $\alpha_2 - \nu_2$-products of automata

Pál Dömösi · György Maróti

Received: 17 March 2011 / Accepted: 29 September 2011 / Published online: 17 November 2011

Abstract Two equivalent sufficient conditions are given for the completeness of classes of finite automata with respect to the isomorphic simulation under the $\alpha_2 - \nu_2$-product. It is conjectured that these conditions are also necessary with respect to the isomorphic or homomorphic simulation too.

Keywords Product of automata · Homomorphic simulation · Letichevsky criterion

1 Introduction

The concept of general product was introduced by Gluškov [12]. The family of $\alpha_i - \nu_j$-products was described by Gécseg and Jürgensen [10]. It originates from the family of α_i-products introduced by Gécseg [7–9] and the family of ν_i-products defined by Dömösi and Imreh [3]. Dömösi [1] showed that a class of finite automata is complete with respect to the isomorphic or homomorphic simulation under the $\alpha_2 - \nu_3$-product if and only if it has the same property under the general product. Dömösi and Ésik [2] proved that the ν_2-product (and thus, for every nonnegative integer i, the $\alpha_i - \nu_2$-product) does not have this property with respect to the homomorphic simulation (and thus with respect to the isomorphic simulation either).

To the honor of Professor Zoltán Ésik on his 60th birthday.

P. Dömösi has been supported by the project NYF-VEKTOR of Nyíregyháza College under grant No. TIOP-1.3.1-10/1-2010-0013 and the Czech-Hungarian Bilateral Research Foundation under grant No. CZ-01/2009.

P. Dömösi
Institute of Mathematics and Informatics, College of Nyíregyháza,
Sóstói út 31/b, 4400 Nyíregyháza, Hungary
e-mail: domosi@nyf.hu

G. Maróti (✉)
Pollack Mihály Faculty of Engineering and Information Technology,
University of Pécs, Boszorkány út 2, 7624 Pécs, Hungary
e-mail: maroti@epigramma.hu

In this paper, along the same line of research, we give sufficient conditions for the completeness of classes of finite automata with respect to the isomorphic simulation under the $\alpha_2 - \nu_2$-product. We also conjecture the necessity of this condition with respect to the homomorphic simulation (and thus with respect to the isomorphic simulation, too).

2 Preliminaries

For all notions and notations not defined here we refer the reader to the books [5,9,11]. By an automaton we mean a finite deterministic automaton $\mathcal{A} = (A, X, \delta)$ (without outputs) with a nonempty finite set of states A, a nonempty finite set X of inputs, and transition function $\delta : A \times X \to A$. We also use δ in an extended sense, i.e., as a mapping $\delta^* : A \times X^* \to A$, where $\delta^*(a, \lambda) = a$ $(a \in A)$ and $\delta^*(a, px) = \delta(\delta^*(a, p), x)$ $(a \in A, p \in X^*, x \in X)$. In the sequel we shall simply write δ for δ^*.

Each automaton can be considered as an algebra with unary operational symbols (corresponding to each input letter). Therefore, notions such as subautomaton, homomorphism, isomorphism can be defined in the natural way. We say that an automaton $\mathcal{A}$ homomorphically (isomorphically) *represents* an automaton $\mathcal{B}$ if $\mathcal{A}$ has a subautomaton which can be mapped homomorphically onto $\mathcal{B}$. The central notions here are those of homomorphic and isomorphic simulations. An automaton $\mathcal{A} = (A, X, \delta)$ *homomorphically simulates* the automaton $\mathcal{B} = (A', X', \delta')$ under a surjective mapping τ_1 of a subset B of A onto A' and a mapping τ_2 of X' into X^* with $\tau_1(\delta(b, \tau_2(x))) = \delta'(\tau_1(b), x)$ $(b \in B, x \in X')$. (It is understood that $\delta(b, \tau_2(x)) \in B$ holds for every pair $b \in B, x \in X'$.) If τ_1 is bijective then we speak about *isomorphic simulation.* Sometimes we say that $\mathcal{A}$ homomorhically (isomorphically) simulates $\mathcal{B}$ if it does it under some mappings τ_1 and τ_2.

Let $\mathcal{A}_i = (A_i, X_i, \delta_i)$ be automata where $i \in \{1, \ldots, n\}$, $n \geq 1$. Take a finite nonvoid set X and a *feedback function* $\varphi_i : A_1 \times \cdots \times A_n \times X \to X_i$ for every $i \in \{1, \ldots, n\}$. The *general product* (or *Gluškov-type product*) of the automata $\mathcal{A}_i$ with respect to the feedback functions φ_i $(i \in \{1, \ldots, n\})$ is defined to be the automaton $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n(X, (\varphi_1, \ldots, \varphi_n))$ with state set $A = A_1 \times \cdots \times A_n$, input set X and transition function δ given by $\delta((a_1, \ldots, a_n), x) = (\delta_1(a_1, \varphi_1(a_1, \ldots, a_n, x)), \ldots, \delta_n(a_n, \varphi_n(a_1, \ldots, a_n, x)))$ for all $(a_1, \ldots, a_n) \in A$ and $x \in X$. In particular, if $\mathcal{A}_1 = \cdots = \mathcal{A}_n$ then we say that $\mathcal{A}$ is a *(general) power.* In the special case $n = 1$, we have $\mathcal{A} = \mathcal{A}_1(X, \varphi_1)$, and we speak of a *single factor product*.[1]

We shall use the feedback functions $\varphi_i, i = 1, \ldots, n$ in an extended sense as mappings $\varphi_i^* : A_1 \times \cdots \times A_n \times X^*$, where $\varphi_i^*(a_1, \ldots, a_n, \lambda) = \lambda$ and $\varphi_i^*(a_1, \ldots, a_n, px) = \varphi_i^*(a_1, \ldots, a_n, p)\varphi_i(\delta_1(a_1, \varphi_1^*(a_1, \ldots, a_n, p)), \ldots, \delta_n(a_n, \varphi_n^*(a_1, \ldots, a_n, p)), x)$, $a_i \in A_i, i = 1, \ldots, n, p \in X^*$ and, $x \in X$. In the sequel, φ_i^* $(i \in \{1, \ldots, n\})$ will also be denoted by φ_i.

Several families of products can be derived from the general product by defining restrictions on the feedback dependency. Thus, for example, A is called *cascade product* or α_0-*product* if for every $i \in \{1, \ldots, n\}$, φ_i does not depend on its jth state variable if $j \geq i$. In general, $\mathcal{A}$ is an α_i-*product* $(i = 0, 1, \ldots)$ if each φ_t $(t = 1, \ldots, n)$ is independent of the input component and its jth state component $(j = 1, \ldots, n)$ whenever $j \geq t + i$. In particular, if $\mathcal{A}$ is an α_0-product, then we often give the system of feedback functions in

[1] Note that a single factor product is different from its factor in general.

the form $\varphi_1 : X \to X_1, \varphi_2 : A_1 \times X \to X_2, \ldots, \varphi_n : A_1 \times \cdots \times A_{n-1} \times X \to X_n$.[2] If i is a positive integer for which every φ_t $(t = 1, \ldots, n)$ depends on its input variable and not more than i state variables, then $\mathcal{A}$ is a *ν_i-product*. In addition, an *$\alpha_i - \nu_j$-product* $(i = 0, 1, \ldots, j = 1, 2, \ldots)$ is an α_i-product that is also a ν_j-product.

Given a nonempty class $\mathcal{K}$ of automata and a type β of the Gluškov-product (including the general product), we say that $\mathcal{K}$ is complete with respect to the homomorphic (isomorphic) representation/simulation under the β-product if every finite automaton can be represented/simulated homomorphically (isomorphically) by a β-product of automata from $\mathcal{K}$.

Theorem 1 (Letichevsky Decomposition Theorem) [14] *A class $\mathcal{K}$ of automata is complete with respect to homomorphic representations under the general product if and only if there exists an automaton $\mathcal{A} = (A, X, \delta) \in K$ which has a state $a_0 \in A$, two input letters $x, y \in X$ and two input words $p, q \in X^*$ for which*

$$\delta(a_0, x) \neq \delta(a_0, y), \text{ and } \delta(a_0, xp) = \delta(a_0, yq) = a_0. \tag{1}$$

We refer to the above property (1) as *Letichevsky's criterion.* If $\mathcal{K}$ is a class of automata containing an element having Letichevsky's criterion then we also say that $\mathcal{K}$ *satisfies Letichevsky's criterion.* This well-known criterion [14] can be used not only for the characterization of complete classes with respect to homomorphic representations under the general product but for the description of complete classes with respect to isomorphic and homomorphic simulations, too [5].

The well-known sharpness of the Letichevsky Decomposition Theorem is due to the following statement by Z. Ésik.

Theorem 2 [6] *A class $\mathcal{K}$ of automata is complete with respect to the homomorphic representation under the α_2-product if (and only if) it is complete with respect to homomorphic representations under the general product. Therefore, $\mathcal{K}$ has these properties if and only if it satisfies Letichevsky's criterion.*

A direct consequence of this result is that a class $\mathcal{K}$ of automata is complete with respect to homomorphic simulation under the α_2-product if it satisfies Letichevsky's criterion. (Because the class of automata not having the condition (1) is closed under general product and homomorphic simulation, the reverse of this statement can be shown easily.)

On the other hand the following two theorems hold.

Theorem 3 [2] *There exists a singleton class $\mathcal{K}$ of automata which is complete with respect to homomorphic representation under the general product but not complete with respect to homomorphic simulation under the ν_2-product.*

Theorem 4 [1] *A class $\mathcal{K}$ of automata is complete with respect to homomorphic simulation under the $\alpha_2 - \nu_3$-product if (and only if) it is complete with respect to homomorphic representations under the general product. Therefore, $\mathcal{K}$ has these properties if and only if it satisfies Letichevsky's criterion.*

For every positive integer $n \geq 1$, denote by $\mathcal{T}_n = (\{1, \ldots, n\}, T_n, \delta_n))$ the automaton for which T_n is the set of all transformations over $\{1, \ldots, n\}$ and $\delta_n(k, t) = t(k)$ for any $k \in \{1, \ldots, n\}$ and $t \in T_n$.

[2] A feedback from a factor to itself is considered to be of length 1. Thus, in a sequence of automata, a feedback of length 2 is understood to be to the preceding factor.

Given a positive integer n, consider the subautomaton $\mathcal{T}'_n = (\{1, \ldots, n\}, \{t_1, t_2, t_3\}, \delta'_n)$ of T_n, where,

$t_1(k) = k + 1, k = 1, \ldots, n-1; t_1(n) = 1,$

$t_2(1) = 2; t_2(2) = 1; t_2(k) = k, k = 3, \ldots, n,$

$t_3(1) = t_3(2) = 1; t_3(k) = k, k = 3, \ldots, n.$

It is proved in [13] (but it can also easily be shown directly) that $\mathcal{T}'_n$ isomorphically simulates $\mathcal{T}_n$. On the other hand, it is clear that for any positive integer $n > 2$, every automaton having not more than n states can be simulated isomorphically by $\mathcal{T}_n$. Since the isomorphic simulation (as an operation on automata) is obviously transitive, we can derive the following.

Fact 5 *Given a positive integer $n > 2$, the automaton $\mathcal{T}'_n$ isomorphically simulates every automaton having not more than n states.*

3 Automata with control words

We shall use some structures defined in [4,5].

Let $\mathbf{a} = a_1 \ldots a_m a_0$ and $\mathbf{b} = b_1 \ldots b_n b_0$ denote nonempty words over an alphabet A having the following properties:

(1) $a_0 = b_0$, the letters of $\mathbf{a}$ are pairwise distinct, the letters of $\mathbf{b}$ are pairwise distinct, and b_1 does not occur in $\mathbf{a}$;
(2) if $\mathbf{a} = wxy$ and $\mathbf{b} = w'xy'$ is any factorization with a letter x and nonempty words w, w', then $y = y'$ ($w, w' \in A^+, x \in A, y \in A^*$);
(3) $m \leq n$ (and $n > 0$), or equivalently, $|\mathbf{a}| \leq |\mathbf{b}|$ (and $|\mathbf{b}| \geq 2$).

Given $\mathbf{a}$ and $\mathbf{b}$ as above, define *control words* $\mathbf{u} = u_1 \ldots u_s$ and $\mathbf{v} = v_1 \ldots v_s$ as follows:

$$(4)\quad u_1 \ldots u_s = \begin{cases} {a_0}^{n+1} & \text{if } m = 0, \\ (a_1 \ldots a_m a_0)^k & \text{if } m+1 \mid n+1, m \neq 0, \\ & \quad n+1 = k(m+1), \\ a_1 \ldots a_m a_0 b_1 \ldots b_n a_0 & \text{if } m+1 \nmid n+1, \end{cases}$$

$$(5)\quad v_1 \ldots v_s = \begin{cases} b_1 \ldots b_n a_0 & \text{if } m+1 \mid n+1 \text{ (including} \\ & \quad \text{the case } m = 0), \\ b_1 \ldots b_n a_0 a_1 \ldots a_m a_0 & \text{if } m+1 \nmid n+1. \end{cases}$$

The following lemma can be derived from (1) and (2).

Lemma 6 [4,5] *Given control words* $\mathbf{u}, \mathbf{v}$*, for all* $1 \leq i, j \leq s-1$ *we have*

(1a) $u_i = u_j \neq a_0$ *implies* $u_{i+1} = u_{j+1}$,
(2a) $v_i = v_j \neq a_0$ *implies* $v_{i+1} = v_{j+1}$,
(3a) $u_i = v_j \neq a_0$ *implies* $u_{i+1} = v_{j+1}$.

4 Results

Next we consider automata having Letichevsky's criterion and some additional properties.

Lemma 7 *Let $\mathcal{A} = (A, X, \delta)$ be an automaton having a state $a_0 \in A$, two input letters $x, y \in X$ and two input words $p, q \in X^*$ for which*

(1b) $\delta(a_0, x) \neq \delta(a_0, y)$ *and* $\delta(a_0, xp) = \delta(a_0, yq) = a_0$;

(2b) if $p'z_1$ *with* $p' \in X^*, z_1 \in X$ *is a prefix of* xp *and* $q'z_2$ *with* $q' \in X^*, z_2 \in X$ *is a prefix of* yq *for which* $\delta(a_0, p') \neq \delta(a_0, q'), \delta(a_0, p'z_1) = \delta(a_0, q'z_2)$, *then* $z_1 = z_2$.

Then there exists an $a_0 \in A, x, y \in X, p, q \in X^*$ *satisfying (1b) for which*

(3b) for each pair of distinct nonempty prefixes p' *and* p'' *of* $xp, \delta(a_0, p') \neq \delta(a_0, p'')$;

(4b) for each pair of distinct nonempty prefixes q' *and* q'' *of* $yq, \delta(a_0, q') \neq \delta(a_0, q'')$;

(5b) there exists at most one pair $p'z_1$ *and* $q'z_2$ *of words, where* $p' \in X^*, z_1 \in X$ *is a prefix of* xp *and* $q'z_2$ *with* $q' \in X^*, z_2 \in X$ *is a prefix of* yq *for which*

$$\delta(a_0, p') \neq \delta(a_0, q') \tag{2}$$

and

$$\delta(a_0, p'z_1) = \delta(a_0, q'z_2). \tag{3}$$

Furthermore, if (2) and (3) hold, then ${z_1}' = {z_2}'$.[3]

(6b) If $p'z_1$ *is a prefix of* xp *with* $p' \in X^*$ *and* $z_1 \in X$, *moreover,* $q'z_2$ *is a prefix of* yq *with* $q' \in X^*$ *and* $z_2 \in X$, *and* $\delta(a_0, p') = \delta(a_0, q')$, *then* $z_1 = z_2$ *(and thus* $\delta(a_0, p'z_1) = \delta(a_0, q'z_2)$*).*

(7b) If $p'z_1$ *is a prefix of* xp *with* $p' \in X^*$ *and* $z_1 \in X$, *moreover,* $q'z_2$ *is a prefix of* yq *with* $q' \in X^*$ *and* $z_2 \in X$, *then* $\delta(a_0, p') = \delta(a_0, q')$ *and* $\delta(a_0, p'z_1) = \delta(a_0, q'z_2)$ *implies* $z_1 = z_2$.

Proof Suppose (1b) and (2b) hold. We start by proving that (7b) can be assumed. First we observe that if $xp = p'z_1p''$ with $p', p'' \in X^*$ and $z_1 \in X$, moreover, $yq = q'z_2q''$ with $q', q'' \in X^*$ and $z_2 \in X$, then $\delta(a_0, p') = \delta(a_0, q')$ and $\delta(a_0, p'z_1) = \delta(a_0, q'z_2)$, and then because of $\delta(a_0, x) \neq \delta(a_0, y)$, $p' = q' = \lambda$ is impossible, we get $z_1 = z_2$.

If $p' \neq \lambda$ then we can replace xp with $p'z_2p''$ so that all of the properties (1b)–(6b) remain true.

Similarly, if $p' = \lambda$ then $q' \neq \lambda$ and thus we can replace yq with $q'z_1q''$ so that all of the properties (1b)–(6b) remain true.

Thus, indeed, we can assume (7b).

Next we consider an automaton $\mathcal{A} = (A, X, \delta)$ with a state a_0, input letters $x, y \in X$, input words $p, q \in X^*$ having the properties (1a)–(2a). Put $x_1 = x, y_1 = y$ and $x_1p = x_1 \cdots x_{m+1}, y_1q = y_1 \cdots y_{n+1} \in X^*$ $(x_1, x_2, \ldots, x_{m+1}, y_1, y_2, \ldots, y_{n+1} \in X, m, n \geq 0)$.

By (1b), $\delta(a_0, x_1) \neq \delta(a_0, y_1), \delta(a_0, x_1 \cdots x_{m+1}) = \delta(a_0, y_1 \cdots y_{m+1}) = a_0$ and $x_{m+1} = y_{m+1} = z$.

By (2b), for every $i = 1, \ldots, m$, $j = 1, \ldots, n, \delta(a_0, x_1 \cdots x_i) \neq \delta(a_0, y_1 \cdots y_j)$ and $\delta(a_0, x_1 \cdots x_{i+1}) = \delta(a_0, y_1 \cdots y_{j+1})$ implies $x_{i+1} = y_{j+1}$.

Put $a_i = \delta(a_0, x_1 \cdots x_i)$ and $b_i = \delta(a_0, y_1 \cdots y_j), i = 1, \ldots, m+1, j = 1, \ldots, n+1$.[4]

Without loss of generality we may assume that p has the minimal length among the words satisfying our conditions for appropriate a_0, x, y, q and that q is chosen to this p so that q has the minimal length among the words satisfying our conditions for an appropriate a_0, x, y together with this p.

First we show that, by these minimality conditions, we can choose the input words p and q such that there exists no repetition in the state sequences $a_1 \cdots a_{m+1}$ and $b_1 \cdots b_{n+1}$, respectively.

[3] Note that (2b) is a direct consequence of (5b).

[4] Note that $a_{m+1} = b_{n+1} = a_0$.

Suppose, say $a_k = a_\ell$ for some $1 \leq k < \ell \leq m+1$.

Then let us consider the input word $x_1 \cdots x_k x_{\ell+1} \cdots x_m$ instead of $x_1 p$ and thus the state sequence $a_1 \cdots a_k a_{\ell+1} \cdots a_m$ satisfying the conditions (1b). On the other hand, new coincidences between the two state sequences did not arise. Of course, it may be possible that $a_k \neq b_j$ and $a_{\ell+1} = b_{j+1}$. In that case, however, using (2a), $x_\ell = y_j$. Thus $a_k = a_\ell$ implies that (2a) remain valid. This contradicts the minimality of the length of p. Therefore, we may assume that there is no repetition in the state sequence $a_1 \cdots a_{m+1}$.

By the same treatment we get that, by the minimality of the length of q, there is no repetition in the state sequence $b_1 \cdots b_{n+1}$.

Next we show the following: if $a_i = b_j$ for some $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$ then $a_i \cdots a_m = b_j \cdots b_n$ can be assumed.

Indeed, suppose that $a_i = b_j$. Two cases arise:

Case 1 $m \leq n$. Let j be the minimal index with $a_i = b_j$ for some $i \in \{1, \ldots, m+1\}$. First we observe $j > 1$. Indeed, $a_1 \neq b_1$ is assumed and thus $b_1 = a_1$ is impossible. Moreover, if $i > 1$ and $j = 1$ then $a_i \neq a_1$. Therefore, we could consider $x_i \cdots x_{m+1}$ instead of yq (and thus $a_i \cdots a_{m+1}$ instead of $b_1 \cdots b_{m+1}$), which contradicts the minimality of the length of q.

Thus $j > 1$ and we can consider $y_1 \cdots y_{j-1} x_i \cdots x_{m+1}$ instead of $y_1 \cdots y_{n+1}$, moreover, $b_1 \cdots b_{j-1} a_i \cdots a_{m+1}$ instead of $b_1 \cdots b_{n+1}$. Thus we can obtain property (5b) while property (4b) is preserved.

Case 2 $m > n$. Let i be the minimal index with $a_i = b_j$ for some $j \in \{1, \ldots, n+1\}$. Similarly to the previous case, it can be proved that we can consider $x_1 \cdots x_{i-1} y_j \cdots y_{n+1}$ instead of $x_1 \cdots x_{m+1}$, moreover, $a_1 \cdots a_{i-1} b_j \cdots b_{n+1}$ instead of $a_1 \cdots a_{m+1}$ and that with these changes we can reach property (5b) while property (4b) is preserved. □

Lemma 8 *Let $\mathcal{A} = (A, X, \delta)$ be an automaton having the properties (1b) and (2b). Then there is a state $a_0 \in A$, two input letters $x, y \in X$ and two input words $p, q \in X^*$ under which we have*

(1c) $\delta(a_0, x) \neq \delta(a_0, y)$ *and* $\delta(a_0, xp) = \delta(a_0, yq) = a_0$,

(2c) $|p| = |q|$,

(3c) if $p'z_1$ with $p' \in X^*, z_1 \in X$ is a prefix of xp and $q'z_2$ with $q' \in X^*, z_2 \in X$ is a prefix of yq for which $\delta(a_0, p') \neq \delta(a_0, q')$, $\delta(a_0, p'z_1) = \delta(a_0, q'z_2)$ and $|p'| = |q'|$ then $z_1 = z_2$,

(4c) if $p''z_1$ with $p'' \in X^*, z_1 \in X$ is a prefix of xp and $q''z_2$ is a prefix of yq with $q'' \in X^*, z_2 \in X$, moreover, $\delta(a_0, p'') = \delta(a_0, q'')$ and $|p''| = |q''|$ then $z_1 = z_2$ (and thus $\delta(a_0, p''z_1) = \delta(a_0, q''z_2)$).

Proof Consider an automaton $\mathcal{A} = (A, X, \delta)$ with a state a_0, input letters $x, y \in X$, input words $p, q \in X^*$ having the properties (1a)-(2a). Put again $x_1 = x, y_1 = y$ and $x_1 p = x_1 \cdots x_{m+1}, y_1 q = y_1 \cdots y_{n+1} \in X^*$ ($x_1, x_2, \ldots, x_{m+1}, y_1, y_2, \ldots, y_{n+1} \in X, m, n \geq 0$).

Moreover, put again $a_i = \delta(a_0, x_1 \cdots x_i)$ and $b_i = \delta(a_0, y_1 \cdots y_j)$, $i = 1, \ldots, m+1$, $j = 1, \ldots, n+1$.[5]

Without loss of generality we may assume, say, $m \leq n$.

Next we show that b_1 does not occur in $\mathbf{a}$.

Suppose the contrary and let $b_1 = a_j$ for some $j \in \{1, \ldots, m+1\}$. Then, using (6b), $\mathbf{b} = a_j \cdots a_{m+1}$, i.e., either $b_1 = a_1$ (with $j = 1$) or $|\mathbf{b}| < |\mathbf{a}|$ (with $j > 1$). Because of (1c) and $m \leq n$, both cases are impossible. Thus we have that indeed, b_1 does not occur in $\mathbf{a}$.

[5] Note that $a_{m+1} = b_{n+1} = a_0$.

Finally, we remark that we may assume (7b) as well.

Similarly to [4,5] consider the following structures:

Given **a** and **b** as above, define *control words*, $\mathbf{u} = u_1 \dots u_s$ and $\mathbf{v} = v_1 \dots v_s$:

$$(*)\ u_1 \dots u_s = \begin{cases} {a_0}^{n+1} & \text{if } m = 0, \\ (a_1 \dots a_m a_0)^k & \text{if } m+1 \mid n+1,\ m \neq 0, \\ & \quad n+1 = k(m+1), \\ a_1 \dots a_m a_0 b_1 \dots b_n a_0 & \text{if } m+1 \nmid n+1, \end{cases}$$

$$(**)\ v_1 \dots v_s = \begin{cases} b_1 \dots b_n a_0 & \text{if } m+1 \mid n+1 \text{(including the case} \\ & \quad m = 0), \\ b_1 \dots b_n a_0 a_1 \dots a_m a_0 & \text{if } m+1 \nmid n+1. \end{cases}$$

On the basis of these constructions, let

$$(\star)\ {}^{6}\ {x'}_1 \dots {x'}_s = \begin{cases} {x_1}^{n+1} & \text{if } m = 0 \\ (x_1 \dots x_m x_{m+1})^k & \text{if } m+1 \mid n+1,\ m \neq 0, \\ & \quad n+1 = k(m+1), \\ x_1 \dots x_m x_{m+1} y_1 \dots y_n y_{n+1} & \text{if } m+1 \nmid n+1, \end{cases}$$

$$(\star\star)\ {x''}_1 \dots {x''}_s = \begin{cases} y_1 \dots y_n y_{n+1} & \text{if } m+1 \mid n+1 \text{ (including} \\ & \quad \text{the case } m = 0), \\ y_1 \dots y_n y_{n+1} x_1 \dots x_m x_{m+1} & \text{if } m+1 \nmid n+1. \end{cases}$$

Clearly, we have immediately (1c) and (2c) by choosing x, y, p, q as ${x_1}', {x_1}''$, ${x'}_2 \cdots {x'}_s, {x_2}'' \cdots {x''}_s$, respectively.

Observe that (3b), (4b) and $a_{m+1} = b_{m+1} = a_0$ imply (1), and (7b) implies (2). Therefore, Lemma 6 holds for **u** and **v**.

By (4) and (5), $m = 0$ implies $u_i = v_i$ if and only if $i = s$. On the other hand, using (2b), we have ${x'}_s = {x''}_s$. Then we obtain (3c). Moreover, there are no proper prefixes p'' of ${x_1}' \cdots {x'}_{m+1}$ and q'' of ${x_1}'' \cdots {x''}_{m+1}$ with $\delta(a_0, p'') = \delta(a_0, q'')$ and $|p''| = |q''|$. Thus (4c) holds because appropriate input words and letters do not appear.

Suppose $m+1 \mid n+1$ with $m > 0$. By (6b), (3b), and (4b), there is no repetition in **b** which is equal to **v** in this case. Therefore, $u_i = v_i$ implies $n - m + 1 \leq i \leq n+1$ with $u_i = a_{i-n+m}$ and $v_i = b_i$. Using (6b), we get $u_i \cdots u_{n+1} = v_i \cdots v_{n+1}$. Let i be the minimal index with $u_i = v_i$. Obviously, $a_1 \neq b_1$ implies $i > 1$. Moreover, $i - 1$ is the only index for which $u_{i-1} \neq v_{i-1}$ and $u_i = v_i$. Thus we obtain (3c) because (2b) implies ${x'}_i = {x''}_i$. In addition, (4c) follows from (7b).

It remains to study the case $m+1 \nmid n+1$. Because of (4b), there is no repetition in the string **b**. Therefore, if there exists a $j \in \{1, \dots, m+1\}$ such that $n+1 < j \leq m+1$ then $u_j \neq v_j$ necessarily holds. Suppose $j \leq n+1$; then $u_j = v_j$ implies $a_j = b_j$. Let j be the minimal index having this property. Recall that $a_1 \neq b_1$. and thus $j > 1$. Hence, $\mathbf{v} = b_1 \cdots b_{j-1} a_j \cdots a_{m+1} a_1 \cdots a_{m+1}$ and thus $|\mathbf{b}| = |\mathbf{a}|$ contradicting $m+1 \nmid n+1$. Therefore, for every $j \in \{1, \dots, m+1\}$, $u_j \neq v_j$. Thus coincidences in **u** and **v** may appear in their last $m+1$ positions. Moreover, the last $m+1$ positions of **u** form the suffix of length $n+1$ of **b** and the last $m+1$ positions of **v** form the word **a**. Because $u_{m+1} = v_{m+1} = a_0$, there exists a $j \in \{n+2, \dots, n+m+2\}$ with $u_j = v_j$. Let j be the minimal index with this property. Recall that $u_{n+1} = b_{n-m}$ with $b_{n-m} \neq b_{n+1}$ and $v_{n+1} = b_{n+1} (= a_0)$. Therefore, we can

[6] Note that, by (2a), $x_{m+1} = y_{n+1}$ holds for all of the listed cases.

be sure that $u_{j-1} \neq v_{j-1}$. Moreover, by our constructions, $u_j = b_{j-m-1}$ and $v_j = a_{j-n-1}$. Two cases arise:

Case 1 If $j = n+2$ then $u_{j-1} = b_{n-m}$ and $v_{j-1} = b_{n+1}$. Then $\delta(b_{n-m}, x'_j) = \delta(b_{n+1}, x''_j)$, where $b_{n+1} = a_{m+1}$ $(= a_0)$. Thus we have $\delta(a_0, p'x''_j) = a_1$ with $p' = \lambda$ and $\delta(a_0, q'x'_j) = a_1$, where q' is a prefix of yq. Because $\delta(a_0, p') \neq \delta(a_0, q')$ (where $p' = \lambda$), applying (2b), we have $x'_j = x''_j$. Moreover, there are no more prefixes of xp and yq having these properties. Thus we get (3c). Furthermore, (4c) follows from (7b)

Case 2 If $j > n+2$ then $u_{j-1} = b_{j-m-2}$ and $v_{j-1} = a_{j-n-2}$. By the minimality of j, $b_{j-m-2} \neq a_{j-m-2}$. On the other hand, $b_{j-m-1} = a_{j-n-1}$ such that $\delta(b_{j-m-2}, x_j') = b_{j-m-1}$ and $\delta(a_{j-n-2}, x''_j) = a_{j-m-1}$. Therefore, considering the prefixes p' of xp and q' of yq with $\delta(a_0, p') = a_{j-m-1}$ and $\delta(a_0, q') = b_{j-m-2}$, using (2b), it follows that (3c) holds. In addition, (4c) follows from (7b) again. □

Lemma 9 *Let $\mathcal{K}$ be a class of automata containing an element $\mathcal{A}$ having the conditions (1c)–(4c) of Lemma 8. Then $\mathcal{K}$ is complete with respect to isomorphic simulation under the $\alpha_2 - \nu_2$-product.*

Proof By Fact 5, to show the validity of our statement, it is enough to prove that for every positive integer $n \geq 2$, the automaton $\mathcal{T}'_n$ can be simulated isomorphically by an $\alpha_2 - \nu_2$-power of an automaton satisfying (1c)–(4c). For the sake of simplicity, for every $n \geq 1$ consider the automaton $\mathcal{T}''_n = (\{1, \ldots, n\}, \{t'_1, t_2, t_3\}, \delta''_n)$ with $\delta(k, t_i) = t_i(k)$, for any $k \in \{1, \ldots, n\}$ and $t_i \in \{t_2, t_3\}$, where t_2 and t_3 are defined as before: $t_1(k) = k+1, k = 1, \ldots, n-1; t_1(n) = 1$, and $t_2(1) = 2; t_2(2) = 1; t_2(k) = k, k = 3, \ldots, n,$. Moreover, let $\delta(k, t'_1) = t'_1(k)$, for any $k \in \{1, \ldots, n\}$, where $t'_1(k) = k-1$ if $k > 1$ and $t_1'(1) = n$. Obviously, the transformation $(t_1')^{n-1}$ coincides with the transformation t_1. Therefore, it is clear that $\mathcal{T}''_n$ isomorphically simulates $\mathcal{T}'_n$ under the mappings $\tau_1 : \{1, \ldots, n\} \to \{1, \ldots, n\}, \tau_1(i) = i, i \in \{1, \ldots, n\}$ and $\tau_2 : \{t_1, t_2, t_3\} \to \{t'_1, t_2, t_3\}^*, \tau_2(t_1) = (t_1')^{n-1}, \tau_2(t_2) = t_2$, and $\tau_2(t_3) = t_3$. Thus, it is enough to prove (for simplicity's sake) that $\mathcal{T}''_n$ can be simulated isomorphically by an $\alpha_2 - \nu_2$-power of $\mathcal{A}$.

Put $m = |xp|$ $(= |yq|)$ and define the power

$$\mathcal{A}^{nm}(\{x_{1,1}, \ldots, x_{1,m}, x_{2,1}, \ldots, x_{2,m}, x_{3,1}, \ldots, x_{3m}\}, \varphi_1, \ldots, \varphi_{nm})$$

of the automaton $\mathcal{A}$ in the following way.

For $i = 1$ and every $2 < i < n$, let φ_{im} really depend on its input component, moreover, its $((i-1)m+1)$th and $(im+1)$th state variable. Furthermore, let φ_{nm} really depend on its input variable and its $((n-1)m+1)$th and first state variable. Furthermore, let φ_{2n} really depend on its input variable, its first and $(2n+1)$th state variable. Therefore, φ_2 is the only feedback function among $\varphi_n, \varphi_{2n}, \ldots, \varphi_{mn}$ which really depends on its previous with $(2n-1)$ state variable. For this reason, the description of φ_2 is more complex in some sense than the description of the other feedback functions. Each of the other feedback functions φ_j with $j \in \{1, \ldots, nm \mid m \not| j\}$, really depends on its input variable and the $(j+1)$th state variable. Then, obviously, $\mathcal{A}^{nm}(\{x_{1,1}, \ldots, x_{3,m}\}, \varphi_1, \ldots, \varphi_{nm})$ is a ν_2-power.

Let $xp = x_1 \cdots x_{m+1}, yq = y_1 \cdots y_{m+1}, x_1, \ldots, x_{m+1}, y_1, \ldots, y_{m+1} \in X$. We introduce the notation $a_u = \delta(a_0, x_1 \ldots x_u)$ $(u = 1, \ldots, m)$ and $b_v = \delta(a_0, y_1 \ldots y_v)$ $(v = 1, \ldots, m)$. Furthermore, we set $b_0 = a_0, \mathbf{a} = a_0 \ldots a_m$, and $\mathbf{b} = b_0 \ldots b_m$. Recall that $\delta(a_m, x_{m+1}) = \delta(b_m, y_{m+1}) = a_0$. Moreover, by (3c),

$$\text{either } a_m = b_m \text{ or } x_{m+1} = y_{m+1}. \tag{4}$$

Similarly, by (3c),

$$\text{for every } j = 2, \ldots, m,\ b_j = a_j \text{ implies either } b_{j-1} = a_{j-1} \text{ or } x_j = y_j. \tag{5}$$

Let

$$\tau_1 : \{\mathbf{b}(\mathbf{a})^{n-1}, \mathbf{ab}(\mathbf{a})^{n-1}, \ldots, (\mathbf{a})^n\mathbf{b}\} \to \{1, \ldots, n\}$$

and

$$\tau_2 : \{t'_1, t_2, t_3\} \to \{x_{1,1} \cdots x_{1,m}, x_{2,1} \cdots x_{2,m}, x_{3,1} \cdots x_{3,m}\}$$

be the mappings of the isomorphic simulation defined by

$$\tau_1(\mathbf{b}(\mathbf{a})^{n-1}) = 1, \tau_1((\mathbf{a})^{i-1}\mathbf{b}(\mathbf{a})^{n-i}) = i \ (i = 2, \ldots, n-1) \text{ and } \tau_1((\mathbf{a})^{n-1}\mathbf{b}) = n,$$

and

$$\tau_2(t_1{}') = x_{1,1} \cdots x_{1,m}, \tau_2(t_2) = x_{2,1} \cdots x_{2,m}, \tau_2(t_3) = x_{3,1} \cdots x_{3,m},$$

where $\mathbf{a}$ denotes the state vector $(a_1, \ldots, a_m)$ and $\mathbf{b}$ denotes the state vector $(b_1, \ldots, b_m)$. The procedure of the isomorphic simulation is as follows.

Applying the input string $x_{1,1} \cdots x_{1,m}$, each of the 1st, $\ldots$, $(nm-1)$th factors of the product receives the state of the next factor, while the nmth one receives the state of the first one using the appropriate values of its feedback function. In the first step of this procedure, each of the mth,$2m$th,…,nmth factors are in the state a_0 while each of the $(m+1)$th, $(2m+1)$th, $\ldots$, $((n-1)m+1)$th, and the first factors are in one of the states a_1 or b_1. By the effect of a_1 the input letter x_1 and by the effect of b_1 the input letter y_1 is presented by the feedback functions of the mth, $2m$th, $\ldots$, nmth factors. Similarly, for every $i = 1, \ldots, m-1$, the ith and the $(i+1)th$, the $(i+m)$th and the $(i+m+1)$th, …, the $(i+(n-1)m)$th and the $(i+(n-1)m+1)$th factors are in either their a_i and a_{i+1} or their b_i and b_{i+1} states. If $a_{i+1} \neq b_{i+1}$, then by the effect of a_{i+1} the input letter x_{i+1} and by the effect of b_{i+1} the input letter y_{i+1} is presented by the feedback functions of the ith, $(i+m)$th, $\ldots$, $(i+(n-1)m)$th factors. If $a_{i+1} = b_{i+1}$ and $a_i \neq b_i$ then using (4) and (5), $x_i = y_i$. Therefore, it does not lead to a contradiction if in this case the appropriate feedback function presents x_i $(= y_i)$. In the next steps this procedure is repeated m-times such that each next step results in a shift to the left in the indices of the factors.

During this procedure, the α_2-power goes, in order, from its state $\mathbf{b}(\mathbf{a})^{n-1}$ into the state $(\mathbf{a})^{n-1}\mathbf{b}$, and for every $i = 2, \ldots, n$ from its state $(\mathbf{a})^{i-1}\mathbf{b}(\mathbf{a})^{n-i}$ into the state $(\mathbf{a})^{i-2}\mathbf{b}(\mathbf{a})^{n-i+1}$ according to $\mathcal{T}''_n$, which goes from its state 1 into the state n and for every $i = 2, \ldots, n$, from its state i into the state $i-1$ by the effect of its input letter t'_1.

Applying the input strings $x_{2,1} \cdots x_{2,m}$, the mth factor receives the state of the $(m+1)$th factor, the $2m$th factor receives the state of the first factor, and in order, the $3m$th, $\ldots$, nmth factors receive the state of the $(2m+1)$th, $\ldots$, $((n-1)m+1)$th factors. All the other factors receive the states of the next factors. The steps of the state transitions work in the same way as in the previous explanation.

During this procedure, the $\alpha_2 - \nu_2$-power goes, in order, from its state $\mathbf{b}(\mathbf{a})^{n-1}$ into the state $\mathbf{ab}(\mathbf{a})^{n-2}$, from its state $\mathbf{ab}(\mathbf{a})^{n-2}$ into the state $\mathbf{b}(\mathbf{a})^{n-1}$ and for every $i = 3, \ldots, n$ from its state $(\mathbf{a})^{i-1}\mathbf{b}(\mathbf{a})^{n-i}$ into itself according to $\mathcal{T}''_n$, which goes from its state 1 into the state 2, from its state 2 into the state 1 and for every $i = 3, \ldots, n$, it remains in its state by the effect of its input letter t_2.

Finally, applying the input strings $x_{3,1} \cdots x_{3,m}$, the mth factor receives the state of the $(m+1)$th factor, the $2m$th factor goes into the states $a_1, a_2, \ldots, a_n$, and in order,

the $3m$th, ..., nmth factors receive the state of the $(2m+1)$th, ..., $((n-1)m+1)$th factors. All the other factors receive the states of the next factors. Apart from the mth factor, the steps of the state transitions work in the same way as in the previous two explanations. By these state transitions, the mth factor takes into consideration only the values of the input components of the feedback function. Thus $x_{3,1}$ results the x_1, $x_{3,2}$ results the x_2, and so on, $x_{3,m}$ results the x_m value of the feedback. Under the first step of the simulation, the $2m$th factor of the $\alpha_2 - \nu_2$ power is in the state a_0. Therefore, by the effect of the value of the feedback x_1, it goes into the state a_1. For every further $i = 2, \ldots, m$, the $2m$th factor is in the state a_{i-1} under the ith step of the simulation. For this state the feedback function φ_{2m} presents the input letter x_i taking this factor from the state a_{i-1} into the state a_i.

During this procedure, the $\alpha_2 - \nu_2$-power goes, in order, from its state $\mathbf{ab}(\mathbf{a})^{n-2}$ into the state $\mathbf{b}(\mathbf{a})^{n-1}$ and from all the other states into itself according to $\mathcal{T}''_n$, which goes from its state 2 into the state 1, and for every $i = 1, 3, \ldots, n$, it remains in its state by the effect of its input letter t_2.

Obviously, then, the considered $\alpha_2 - \nu_2$-product isomorphically simulates $\mathcal{T}''_n$, as we stated. To the completeness of the proof it remains to give the formal definition of the feedback functions. Fix an arbitrary input letter $z \in \{x_{1,1}, \ldots, x_{3,m}\}$ and let for every $(d_1, \ldots, d_{mn} \in A^{nm}, x_{i,j} \in \{x_{1,1}, \ldots, x_{3,m}\}$,

$$
\begin{aligned}
&\varphi_k(d_1, \ldots, d_{mn}, x_{i,j}) \\
&= \begin{cases}
x_u & \text{if } k = \ell m + t, \ell = 0, \ldots, n-1, t = 1, \ldots, m-1, j = 1, \ldots, m-t, \\
& \quad d_{k+1} = a_u, u = t + j, \\
& \text{or } k = \ell m + t, \ell = 0, \ldots, n-1, t = 1, \ldots, m-1, j = m-t+1, \ldots, m, \\
& \quad d_{k+1} = a_u, u = j - m + t, \\
& \text{or } k = \ell m, \ell = 1, \ldots, n, i = 1, d_{k+1} = a_u, u = j, \\
& \text{or } k = mn, i = 1, d_1 = a_u, u = j, \\
& \text{or } k = m, i = 2, d_{m+1} = a_u, u = j, \\
& \text{or } k = 2m, i = 2, d_1 = a_u, u = j, \\
& \text{or } k = m, i = 3, b_j \notin \{d_1, d_{m+1}\}, u = j, \\
& \text{or } k = 2m, i = 3, u = j, \\
& \text{or } k = \ell m, \ell = 3, \ldots, m, i \in \{2, 3\}, d_{k-m+1} = a_u, u = j, \\
y_u & \text{if } k = \ell m + t, \ell = 0, \ldots, n-1, t = 1, \ldots, m-1, j = 1, \ldots, m-t, \\
& \quad d_{k+1} = b_u, u = t + j, \\
& \text{or } k = \ell m + t, \ell = 0, \ldots, n-1, t = 1, \ldots, m-1, j = m-t+1, \ldots, m, \\
& \quad d_{k+1} = b_u, u = j - m + t, \\
& \text{or } k = \ell m, \ell = 1, \ldots, n, i = 1, d_{k+1} = b_u, u = j, \\
& \text{or } k = mn, i = 1, d_1 = b_u, u = j, \\
& \text{or } k = m, i = 2, d_{m+1} = b_u, u = j, \\
& \text{or } k = 2m, i = 2, d_1 = b_u, u = j, \\
& \text{or } k = m, i = 3, b_j \in \{d_1, d_{m+1}\}, u = j, \\
& \text{or } k = \ell m, \ell = 3, \ldots, m, i \in \{2, 3\}, d_{k-m+1} = b_u, u = j, \\
z & \text{otherwise.}
\end{cases}
\end{aligned}
$$

The property (4c) assures the unambiguity of the above defined feedback functions. The proof is complete. □

By Lemmas 8 and 9 we obtain the following.

Theorem 10 *Given a class $\mathcal{K}$ of automata, let $\mathcal{A} = (A, X, \delta)$ be an automaton in $\mathcal{K}$ having a state $a_0 \in A$, two input letters $x, y \in X$ and two input words $p, q \in X^*$ under which*

(1d) $\delta(a_0, x) \neq \delta(a_0, y)$ *and* $\delta(a_0, xp) = \delta(a_0, yq) = a_0$,
(2d) if $p'z_1$ with $p' \in X^*, z_1 \in X$ is a prefix of xp and $q'z_2$ with $q' \in X^*, z_2 \in X$ is a prefix of yq for which $\delta(a_0, p') \neq \delta(a_0, q')$, $\delta(a_0, p'z_1) = \delta(a_0, q'z_2)$, then $z_1 = z_2$.

Then, $\mathcal{K}$ is complete with respect to isomorphic simulation under the $\alpha_2 - \nu_2$-product.

Because property (5b) in Lemma 7 is a consequence of (1b) and (2b), we have the following equivalent form of our theorem.

Theorem 11 *Given a class $\mathcal{K}$ of automata, let $\mathcal{A} = (A, X, \delta)$ be an automaton in $\mathcal{K}$ having a state $a_0 \in A$, two input letters $x, y \in X$ and two input words $p, q \in X^*$ under which*

(1e) $\delta(a_0, x) \neq \delta(a_0, y)$ *and* $\delta(a_0, xp) = \delta(a_0, yq) = a_0$;
(2e) there exists at most one pair $p'z_1$ and $q'z_2$ of words, where $p' \in X^*, z_1 \in X$ is a prefix of xp and $q'z_2$ with $q' \in X^*, z_2 \in X$ is a prefix of yq for which (i) $\delta(a_0, p') \neq \delta(a_0, q')$ and (ii) $\delta(a_0, p'z_1) = \delta(a_0, q'z_2)$. Furthermore, if (i) and (ii) hold, then ${z_1}' = {z_2}'$.

Then, $\mathcal{K}$ is complete with respect to isomorphic simulation under the $\alpha_2 - \nu_2$-product.

Conjecture 12 *The reverse of Theorem 10 is also true not only with respect to the isomorphic simulation but with respect to the homomorphic simulation as well. In other words, if $\mathcal{K}$ is a class of automata which is complete with respect to the homomorphic simulation under the $\alpha_2 - \nu_2$-product then there exists an automaton $\mathcal{A} \in \mathcal{K}$ satisfying conditions (1d)–(2d).*

5 Conclusions and future work

In this paper we proved two equivalent sufficient conditions for the completeness of automata classes with respect to the isomorphic simulation under the $\alpha_2 - \nu_2$-product. Moreover, it is conjectured that both of these conditions are necessary.

A challenge for further research is to prove our conjecture. Another possible line of investigation is to give, based on our results, a characterization of complete classes of automata with respect to the homomorphic realization under the $\alpha_2 - \nu_2$-product.

Acknowledgments The first author is grateful to the Scientific Board of Nyíregyháza College and to the Scientific Committee of Faculty of Science and Informatics of Nyíregyháza College for their grants supporting him in developing this joint work with the second author.

References

1. Dömösi, P.: On α_2-ν_3-products of automata. 7th International Conference on Automata and Formal Languages. Salgótarján (1993). Publ. Math. **48**(suppl.), 233–242 (1996)
2. Dömösi, P., Ésik, Z.: Homomorphic simulation and Letichevsky's criterion. In: 2nd Workshop on Descriptional Complexity of Automata, Grammars and Related Structures. London (2000). J. Automata Lang. Comb. **6**(4), 427–436 (2001)

3. Dömösi, P., Imreh, B.: On ν_i-products of automata. Acta Cybern. **6**, 149–162 (1989)
4. Dömösi, P., Nehaniv, C.L.: On complete systems of automata. Theor. Comput. Sci. **245**, 27–54 (2000)
5. Dömösi, P., Nehaniv, C.L.: Algebraic Theory of Automata Networks. An Introduction. SIAM, Philadelphia (2005)
6. Ésik, Z.: Homomorphically complete classes of automata with respect to the α_2-product. Acta Sci. Math. **48**, 135–141 (1985)
7. Gécseg, F.: Composition of automata. In: Automata, Languages and Programming: 2nd Colloquium, Saarbrücken, Lecture Notes in Computer Science, vol. 14, pp. 351–363. Springer, Berlin (1974)
8. Gécseg, F.: On products of abstract automata. Acta Sci. Math. **38**, 21–43 (1976)
9. Gécseg, F.: Products of Automata. EATCS Monographs on Theoretical Computer Science, vol. 7. Springer, Berlin (1986)
10. Gécseg, F., Jürgensen, H.: On $\alpha_0 - \nu_1$-products of automata. Theor. Comput. Sci. **80**, 35–51 (1991)
11. Gécseg, F., Peák, I.: Algebraic Theory of Automata. Disquisitiones Mathematicae Hungaricae, vol. 2. Akadémiai Kiadó, Budapest (1972)
12. Gluškov, V.M.: The abstract theory of automata. Uspekhi Mat. Nauk, **16**(5), 3–62 (1961); Correction: ibid., **17**(2), 270 (1962)
13. Gluškov, V.M.: On complete system operations in computers (in Russian). Kibernetika (Kiev) **2**, 1–5 (1968)
14. Letichevsky, A.A.: Completeness conditions for finite automata (in Russian). Ž. Vyčisl. Mat. i Mat. Fiz. **1**, 702–710 (1961); translated as Comut. Math. and Math. Phys. **1**, 702–710 (1961)