Московский Государственный Университет
имени М.В. Ломоносова

Российская Академия Наук

Международная Академия Технологических Наук

Российская Академия Естественных Наук

Интеллектуальные Системы.

Теория и приложения

ТОМ **25** ВЫПУСК 5 ∗ **2021**

МОСКВА

ISSN 2411–4448

УДК 519.95; 007:159.955
ББК 32.81

Издается с 1996 г.*

В журнале «Интеллектуальные системы. Теория и приложения» публикуются научные достижения в области теории и приложений интеллектуальных систем, новых информационных технологий и компьютерных наук.

Издание журнала осуществляется под эгидой МГУ имени М. В. Ломоносова, Научного Совета по комплексной проблеме «Кибернетика» РАН, Отделения «Математическое моделирование технологических процессов» МАТН, Секции «Информатики и кибернетики» РАЕН.

Учредитель журнала: ООО «Интеллектуальные системы».

Журнал входит в список изданий, включенных ВАК РФ в реестр публикаций материалов по кандидатским и докторским диссертациям по математике и механике.

Индекс подписки на журнал: 64559 в каталоге НТИ «Роспечать».

Адрес редакции: 119991, Москва, ГСП-1, Ленинские Горы, д. 1, механико-математический факультет, комн. 12-01.
Адрес издателя: 115230, Россия, Москва, Хлебозаводский проезд, д. 7, стр. 9, офис 9. Тел. +7 (495) 939-46-37, e-mail: mail@intsysjournal.org

*) Прежнее название журнала: «Интеллектуальные системы».

ОГЛАВЛЕНИЕ

Alekseev D.V. Necessary and sufficient conditions for the existence of an image with a given code 5

Gasanov E.E. Cellular automata with locators 15

Vasilev D. I. The one-dimensional closest neighbor search problem solution using the cellular automata with locators 27

Kalachev G.V. Remarks on the Definition of Cellular Automaton with Locators 45

Sitdikov T. R., Kalachev G. V. The complexity of multilayer d-dimensional circuits 55

Vasilev D. I. The two-dimensional closest neighbor search problem solution using the cellular automaton with locators 75

Gasanov E.E., Propazhin A.A. Implementation of key-value databases by cellular automata with locators 79

Necessary and sufficient conditions for the existence of an image with a given code [1]

D.V. Alekseev[2]

The article introduces an image encoding function which is invariant with respect to affine transform. The properties of the encoding funciton are investigated. Necessary and sufficient conditions are found for a given set of numbers to be a code of nonsingulari image.

Keywords: image code, image encoding, affine equivalence.

Introduction

Recognition tasks often require some image encoding. One of the most commonly used image codes is just the coordinates of its points. This encoding is not invariant under geometric transformations, such as translation, rotation, stretching. Despite that the images obtained by such transformations are considered to be equivalent. In addition, that encoding implies fixing some external (to an image) coordinate system.

An affinity invariant image coding was introduced in the papers [4]-[5]. It was shown that the the codes equality of two images is a necessary and sufficient for them to be affine equivalent. This work introduces a modified coding function and researches the properties of that coding function. As a result the necessary and sufficient conditions are derived for an existence of image producing the given code.

The necessary and sufficient conditions for the existence of a three-dimensional image with two given planar projections were derived in [2].

In [5] an affine invariant coding function ρ was introduced: $\rho_{ijk,lmp} = \frac{S(\triangle a_i a_j a_k)}{S(\triangle a_l a_m a_p)}$ where $S(abc)$ stands for the area of the triangle abc. Thus, a set of n points is encoded by $(C_n^3)^2$ real numbers. Obviously, this code is redundant. This work examines the degree of its redundancy. In case of modified encoding the explicit conditions were derived that an arbitrary list of real numbers is the code of some image. For the original encoding function, the respective (implicit) conditions are also given.

In this paper we consider a modified coding function $r_{ijk,lmp} = \frac{S'(\triangle a_i a_j a_k)}{S'(\triangle a_l a_m a_p)}$, where S' stands for the oriented area, i.e. area with a ± sign depending on the triangle orientation.

[1] Originally published in *Intellektualnye Sistemy. Teoriya i prilogeniya* (2020) **24**, No. 2, 55-66 (in Russian).

[2] Alekseev Dmitriy Vladimirovich — Candidate of Physical and Matematical Sciences, senior staff scientist, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Problems of Theorecical Cybernetics Lab, e-mail: dvalex@rambler.ru

The rest of the paper is organized in the following way: The basic concepts and notation are introduced in section 1. The properties of the image code matrix are researched in section 2. The main result is formulated and proven in section 3. Section 4 is a conclusion.

1. Concepts and notation

Let S' be the oriented triangle area, i.e. $S'(\triangle abc) = S(\triangle abc)$ for the positive triangle orientation and $S'(\triangle abc) = -S(\triangle abc)$ for the negative one. The triangle orientation is considered to be positive when the triangle vertices are traversed in counterclockwise order and negative otherwise.

Consider the points $a_1, ..., a_n$ on a plane, let call the set $A = \{a_1, ..., a_n\}$ an image. An image is called emphdegenerate if all points lie on one straight line and *non-degenerate* otherwise. Fix some (Euclidean) coordinate system, the coordinates of the point a_i will be denoted as $X(a_i)$ and $Y(a_i)$. In the following, for convenience, individual indices will be denoted by lowercase Latin letters.

Let call *multi-index*, a vector comprising three indices. The multi-indices will be denoted later in the text by lowercase Greek letters $\alpha, \beta, \gamma, ...$. The multi-index components will be denoted by $\alpha = [\alpha(1), \alpha(2), \alpha(3)]$. The triangle [1] with the respective vertex indices will be denoted as $\triangle_\alpha = \triangle a_{\alpha(1)} a_{\ alpha(2)} a_{\alpha(3)}$.

Let call multi-indices α and α' *equivalent* if and only if the permutation $\begin{pmatrix} \alpha(1) & \alpha(2) & \alpha(3) \\ \alpha'(1) & \alpha'(2) & \alpha'(3) \end{pmatrix} \in S_3$ is even and denote it $\alpha \simeq \alpha'$. Let call the multi-index conjugate to α and denote it $\bar{\alpha}$ if the permutation $\begin{pmatrix} \alpha(1) & \alpha(2) & \alpha(3) \\ \bar{\alpha}(1) & \bar{\alpha}(2) & \bar{\alpha}(3) \end{pmatrix} \in S_3$ is odd. Obviously the triangles with equivalent multi-indices have the same oriented area, and the triangles with conjugate multi-indices have areas with the same absolute values and different signs. Later we do not distinguish between equivalent multi-indices i.e. regard them as the same multi-index. The same will be applies to the respective triangles.

In total, there are C_n^3 different unoriented triangles with vertices from $A = \{a_1, ..., a_n\}$, and respectively $N = 2 \cdot C_n^3$ oriented ones.

Let enumerate all multi-indices (and the respective triangles): $\alpha_1, ..., \alpha_N$. Let $\mathcal{A} = \{\alpha_1, ..., \alpha_N\}$ be the set of all multi-indices, and $E : \alpha_i \mapsto i$ be the respective enumeration function.

Consider the following set of fractions: $r_{ijk,lmp} = \frac{S'(\triangle a_i a_j a_k)}{S'(\triangle a_l a_m a_p)}$. If triangle $\triangle a_l a_m a_p$ is degenerate, i.e. $S'(\triangle a_l a_m a_p) = 0$, then use formal notation

[1] We will also use this notation in the case when the triangle is degenerate.

$r_{ijk,lmp} = \infty$. Let call the set the *code* for image $\{a_1, ..., a_n\}$. Similar encoding procedure was proposed in [1].

Definition 1. Consider $N \times N$ matrix $R = (r_{ij})$ with the elements $r_{ij} = r_{\alpha_i,\alpha_j} = \frac{S'(\triangle_{\alpha_i})}{S'(\triangle_{\alpha_j}}$. Thus, the elements of the image code are arranged in a square table, in which the rows and columns are enumerated by multi-indices (triangles). Let's call R the image *code matrix.*

Note 1. Leater the notation $r\ _{alpha\ beta} = R_{E(\alpha)E(\beta)}$ will be, i.e. rows and columns of the code matrix can also be indexed with multi-indices.

Example 1. Consider a trapezoid $a_1a_2a_3a_4$, with bases a_1a_2 and a_4a_3 such that $|a_1a_2| : |a_4a_3| = 1 : 2$ (see fig. 1). Let enumerate multi-indices as in table 1. Notice that the last four multi-indices are conjugate to the first four, so it is sufficient to construct only the part of the image code matrix corresponding to the first 4 rows and columns. The submatrix is the following:

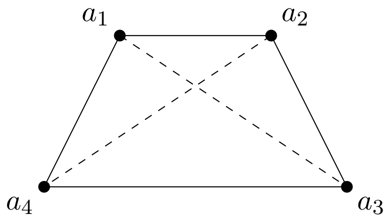


Fig. 1. Example 1.

$R_4 = \begin{pmatrix} 1 & 1 & 1/2 & 1/2 \\ 1 & 1 & 1/2 & 1/2 \\ 2 & 2 & 1 & 1 \\ 2 & 2 & 1 & 1 \end{pmatrix}$. The complete code matrix has the following form:
$R = \begin{pmatrix} R_4 & -R_4 \\ -R_4 & R_4 \end{pmatrix}$.

2. The properties of a code matrix

1) $r_{\alpha\alpha} = 1$ или ∞, for all $\alpha \in \mathcal{A}$ (**reflexivity**).

2) For all $\alpha, \beta \in \mathcal{A}$ such that $r_{\alpha\beta} \notin \{0, \infty\}$ holds $r_{\beta\alpha} = r_{\alpha\beta}^{(-1)}$ (**anti-symmetry**[2]).

3) For all $\alpha, \beta, \gamma \in \mathcal{A}$ such that $r_{\alpha\beta}, r_{\beta\gamma} \notin \{0, \infty\}$ holds $r_{\alpha\gamma} = r_{\alpha\beta} \cdot r_{\beta\gamma}$ (**transitivity**).

[2] If $r_{\alpha\beta} = 0$ then $r_{\beta\alpha} = \infty$. The converse is generally not true.

i	α_i
1	$1,2,3$
2	$1,2,4$
3	$1,3,4$
4	$2,3,4$
5	$1,3,2$
6	$1,4,2$
7	$1,4,3$
8	$2,4,3$

Table 1. Multi-index table

4) Let $\pi, \sigma \in S_3$ and $\alpha, \beta \in \mathcal{A}$. Let multi-indices $\alpha' = \pi(\alpha)$ and $\beta' = \sigma(\beta)$ are the results of permutations π and σ applied to multi-indices α and β, respectively $\alpha' = [\alpha(\pi(1)), \alpha(\pi(2)), \alpha(\pi(3))]$ и $\beta' = [\beta(\sigma(1)), \beta(\sigma(2)), \beta(\sigma(3))]$. Then either $r_{\alpha'\beta'} = (-1)^{\pi} \cdot (-1)^{\sigma} \cdot r_{\alpha\beta}$, or $r_{\alpha\beta} = \infty = r_{\alpha'\beta'}$ (**consistency with index permutations**).

5) Let $i_1, i_2, i_3, i_4 \in \{1, \ldots, N\}$, $\alpha_1 = [i_2, i_3, i_4]$, $\alpha_2 = [i_3, i_4, i_1]$, $\alpha_3 = [i_4, i_1, i_2]$ и $\alpha_4 = [i_1, i_2, i_3]$. Then for any $\beta \in \mathcal{A}$ the equality $r_{\alpha_1\beta} + r_{\alpha_3\beta} = r_{\alpha_2\beta} + r_{\alpha_4\beta}$ holds[3] (**additivity**).

Properties 1-3 are obvious. Property 4 follows from the change in the oriented area sign at permutations of vertices. To prove property 5 we calculate the area of a quadrilateral $a_{i_1}a_{i_2}a_{i_3}a_{i_4}$ (see. fig. 2) by two ways:

$$S'(a_{i_1}a_{i_2}a_{i_3}a_{i_4}) = S'(\triangle a_{i_2}a_{i_3}a_{i_4}) + S'(\triangle a_{i_4}a_{i_1}a_{i_2}) = \\ = S'(\triangle a_{i_3}a_{i_4}a_{i_1}) + S'(\triangle a_{i_1}a_{i_2}a_{i_3}).$$

Divide the equality by $S'(\triangle_\beta)$ and get the property 5.

It is natural to ask: are these conditions sufficient for an arbitrary matrix to be the code of some image? A counterexample below will show that this is not true.

Let prove the following helper lemma:

Lemma 1. *Consider non-degenerate triangle $\triangle_\beta$ and denote $\rho_\alpha = r_{\alpha\beta}$, $\alpha \in \mathcal{A}$. Fix an euclidean coordinates on a plane. Then there exist an affine transform F such that images of $a_i, i = 1, ..., n$ i.e. $c_i = F(a_i)$, have coordinates $X(c_i) = \rho_{i,\beta(3),\beta(1)}$, $Y(c_i) = \rho_{i,\beta(1),\beta(2)}$.*

Proof. Assume without lost of generality that $\beta = [1, 2, 3]$. There exists one and only one affine transform A such that $a_1 \mapsto c_1(0, 0)$, $a_2 \mapsto c_2(1, 0)$

[3] Consider the equation formally, as $\infty + \infty = \infty + \infty$, when the denominator is zero.

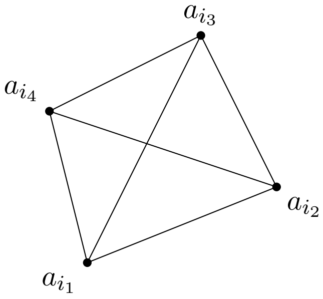


Fig. 2. Area Additivity: $S'_{234} + S'_{124} = S'_{123} + S'_{134}$.

and $a_3 \mapsto c_3(0,1)$. Let (x_i, y_i) be the coordinates of $c_i = A(a_i)$, $i = 1,2,3$. Then $S'(\triangle c_1c_2c_3) = \frac{1}{2}$, $S'(\triangle c_3c_1c_i) = \frac{1}{2}x_i$ and $S'(\triangle c_1c_2c_i) = \frac{1}{2}y_i$. Thus $\rho_{3,1,i} = \frac{S'(\triangle c_3c_1c_i)}{S'(\triangle c_1c_2c_3)} = x_i$ and $\rho_{3,1,i} = \frac{S'(\triangle c_3c_1c_i)}{S'(\triangle c_1c_2c_3)} = y_i$. Proof complete.

Corollary 1. *Let $\triangle_\beta$ be a non-degenerate triangle and*

$$\rho_{i,\beta(1),\beta(2)} = \rho_{j,\beta(1),\beta(2)} = \rho_{k,\beta(1),\beta(2)} = \rho^*,$$

then the points a_i, a_j and a_k are collinear.

Proof. According to lemma 1 there exists an affine transform A such that $A(a_i) = c_i$, $A(a_j) = c_j$,$A(a_j) = c_j$ so that $Y(c_i) = Y(c_j) = Y(c_k) = \rho^*$. Then c_i, c_j and c_k are collinear, therefore a_i, a_j and a_k are collinear too.

Corollary 2. *Two non-degenerate images A and B are affine-equivalent if and only if their code matrices are equal for some points numeration.*

Note 2. This corollary is analogous to Theorem 1 from [4] (for another coding function).

Proof. Oriented areas ratio is conserved under affine transformation so if A is affine image of B their code matrices are the same.

Let us prove the sufficiency. In a non-degenerate image there exists a non-degenerate triangle $\triangle_\beta(A) = \triangle a_ia_ja_k$. As the code matrices are equal then the respective triangle $\triangle_\beta(B) = \triangle b_ib_jb_k$ is non-degenerate too. Consider an image C with points' coordinates $X(c_i) = \rho_{i,\beta(3),\beta(1)}$, $Y(c_i) = \rho_{i,\beta(1),\beta(2)}$). Then by lemma 1 one can construct the affine transforms $F_1 : A \to C$ и $F_2 : B \to C$. Therefore, A and B are affine equivalent. Proof complete.

Let us show by an example that properties 1–5 are not sufficient for the existence of an image, with a given code matrix.

Example 2. Consider a regular pentagon with vertices a_1, ..., a_5. Let us denote the intersection points of the diagonals b_1, ..., b_5 (see Fig. 3). Place the points m_1, m_2 and m_3 inside the triangles $\triangle a_1 a_2 b_4$, $\triangle a_4 a_5 b_2$ and $\triangle a_3 b_1 b_5$, respectively.

Let place unit masses at these points. For a triangle $\triangle a_i a_j a_k$, consider the total mass of points, located inside it. We will call this mass taken with the $+/-$ sign depending on the direction of the bypass, pseudo-area of a triangle $triangle a_i a_j a_k$ and denote $S^*(\triangle a_i a_j a_k)$. Obviously, for the pseudo-area, additivity property holds. Consider the matrix $R = (r_{\alpha\beta})$, $r_{\alpha\beta} = S^*(\triangle_\alpha)/S^*(\triangle_\beta)$. Properties 1-3 are fulfilled for it by construction, property 4 follows from the definition of pseudo-area, and property 5 is due to its additivity.

Suppose that R is code matrix for some image $a'_1, ..., a'_5$. Notice that

$$r_{123,123} = r_{124,123} = r_{125,123} = 1,$$

then by corollary 1, the points a'_3, a'_4 and a'_5 are collinear.

Then $\triangle a'_3 a'_4 a'_5$ has zero area, thus $r_{345,123} = 0$. But it contradicts to $r_{345,123} = 1 \neq 0$.

Note 3. The concept of pseudo-area can be defined more strictly. To do this, place rice. 3 on the complex plane and interpret points as elements of $\mathbb{C}$. Consider a meromorphic function $f(z) = \frac{1}{z-m_1} + \frac{1}{z-m_2} + \frac{1}{z-m_3}$, then, one can define the pseudo-area of a triangle as the following contour integral

$$S^*(\triangle a_i a_j a_k) = \frac{1}{2\pi i} \oint\limits_{\triangle a_i a_j a_k} f(z)\mathrm{dz}.$$

3. Основные результаты

So, conditions 1-5 are not sufficient for the existence of an image with the given code matrix. The theorem below answers the question — what additional conditions can ensure the existence of such an image.

Theorem 1. *Let the matrix R satisfy conditions 1-5. Let there exist $\alpha, \beta \in \mathcal{A}$ such that $r_{\alpha,\beta} \neq \infty$. Then R to is the code matrix of some non-degenerate image, if and only if for any $i, j = 1, ..., n$ the equality*

$$\rho_{\beta(1),i,j} = \rho_{i,\beta(3),\beta(1)} \cdot \rho_{j,\beta(1),\beta(2)} - \rho_{j,\beta(3),\beta(1)} \cdot \rho_{i,\beta(1),\beta(2)}, \qquad (1)$$

is satisfied (here ρ_α stands for $r_{\alpha,\beta}$).

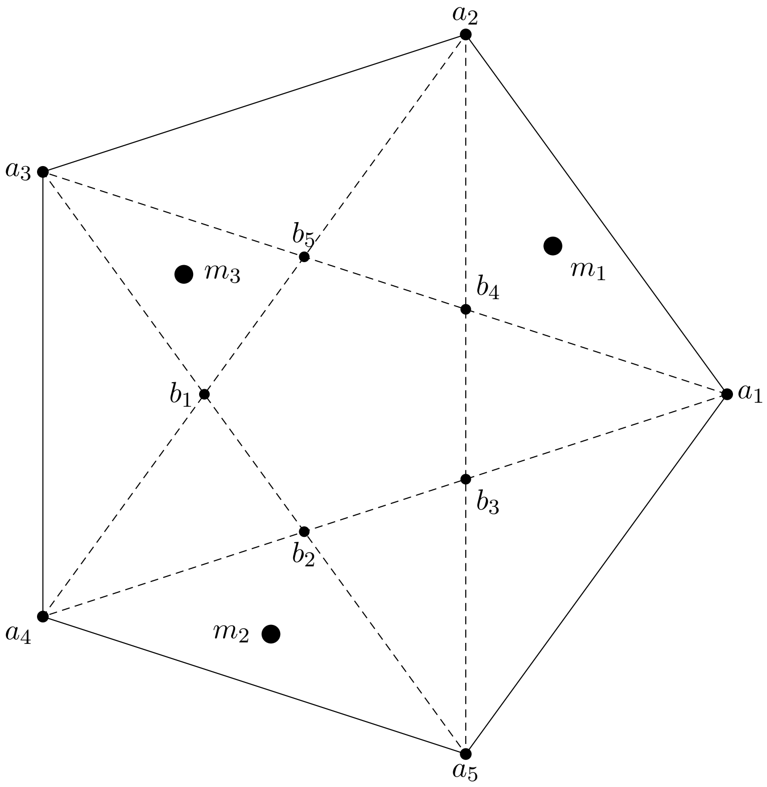


Fig. 3. Example 2.

Proof
Necessity. Let $\beta = [1,2,3]$ without loss of generality. Consider an affine transform A from lemma 1 proof. It maps the points a_s, $s = 1,2,3$ to $c_1(0,0)$, $c_2(1,0)$ and $c_3(0,1)$, respectively. Obviously $S'(\triangle c_1c_2c_3) = \frac{1}{2}$. The transform maps the points a_i and a_j to c_i and c_j, with the coordinates $X(c_i) = \rho_{i,3,1}$, $Y(c_i) = \rho_{i,1,2}$, and $X(c_j) = \rho_{j,3,1}$, $Y(c_j) = \rho_{j,1,2}$, respectively. Then oriented area of $\triangle c_1c_ic_j$ is computed by well-known formula:

$$S(\triangle c_1c_ic_j) = \frac{1}{2}det\begin{pmatrix} X(c_i) & Y(c_i) \\ X(c_j) & Y(c_j) \end{pmatrix} = \frac{1}{2}(\rho_{i,3,1}\cdot\rho_{j,1,2} - \rho_{j,3,1}\cdot\rho_{i,1,2}).$$

Divide the equality by $S'(\triangle c_1c_2c_3) = \frac{1}{2}$, we have an equality (1).

Sufficiency Let the matrix R satisfies the equality(1). Construct the set of points $\{a_i : i = 1,...,N\}$ such that coordinates are $X(a_i) = \rho_{i,3,1}$, $Y(a_i) = \rho_{i,1,2}$. Construct the code matrix for that image $R^* = (r^*_{\alpha\beta})$. Later we will show that it equals to the given matrix R.

Denote $\rho^*_\alpha = r^*_{\alpha\beta}$. Let $\alpha = [i, j, k]$, consider an intersection $P = \{i, j, k\} \cap \{1, 2, 3\}$. Actually P is a common indices set for α and β.

Possible cases:

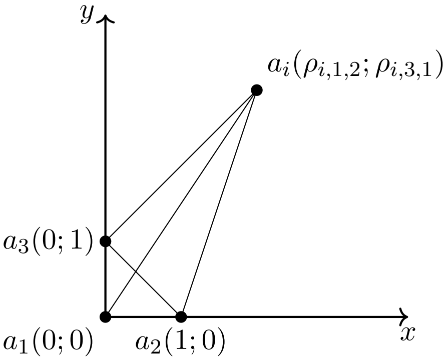


Fig. 4. Случай $|P| = 2$.

- The case $|P| = 3$. Then $\alpha = [j, i, k]$ is a permutation of $1, 2, 3$, i.e. it is eithe $\alpha = \beta$ or $\alpha = \bar{\beta}$. Then $S'(\triangle_\beta) = \frac{1}{2}$ and $S'(\triangle_{\bar{\beta}}) = -\frac{1}{2}$. Thus, $\rho^*_\beta = 1 = \rho_\beta$ and $\rho^*_{\bar{\beta}} = 1 = \rho_{\bar{\beta}}$.

- The case $|P| = 2$ and $1 \in P$ (see fig. 4). In other words the triangles $\triangle_\alpha$ and $\triangle_b$ share two common vertices and one of them is the origin of coordinates. One of i, j, k is not an element of $\{1, 2, 3\}$, let it be i without the lost of generality. If the remaining indices are 1 and 2, then[4] $\alpha = \gamma$ or $\alpha = \bar{\gamma}$, where $\gamma = [1, 2, i]$. If $\alpha = \gamma$ then

  $$S'(\triangle_\gamma) = S'(\triangle a_i a_1 a_2) = \frac{1}{2} Y(a_i) = \frac{1}{2} \rho_{i,1,2}.$$

  Divide the equality by $S'(\triangle_\beta) = 1/2$ we have $\rho^*_\gamma = \rho_\gamma$. If, on the other hand $\alpha = \bar{\gamma}$, then $\rho^*_\alpha = \rho^*_{\bar{\gamma}} = -\rho^*_\gamma = -\rho_\gamma = \rho_{\bar{\gamma}}$. The case $P = \{1, 3\}$ is considered the same way.

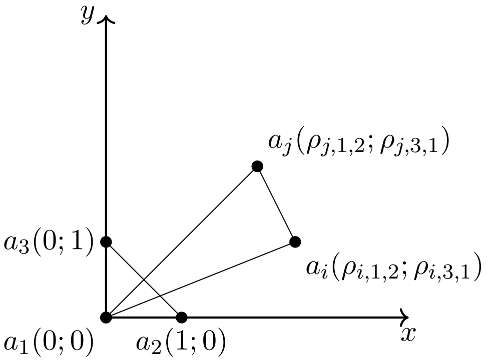
- The case $P = \{2, 3\}$ (see fig. 4). In other words the triangles $\triangle_\alpha$ and $\triangle_b$ share two common vertices a_1 and a_2. One of i, j, k is not an element of $\{2, 3\}$, let it be i without the lost of generality.

  Then either $\alpha = \delta$ or $\alpha = \bar{\delta}$, where $\delta = [i, 3, 2]$. If $\alpha = \delta$ then by property 5 (additivity)

  $$\rho^*_\alpha = \rho^*_{i,3,2} = \rho^*_{i,3,1} + \rho^*_{i,1,2} - \rho^*_{1,2,3} = \rho_{i,3,1} + \rho_{i,1,2} - \rho_{1,2,3} = \rho_{i,3,2} = \rho_\alpha.$$

[4] Recall that the multi-indices are equivalent with respect to a cyclic permutations.

The third equality here follows from $\rho^*_{i,1,2} = \rho_{i,1,2}$ and $\rho^*_{i,3,1} = \rho_{i,3,1}$ proved earlier. If, on the other hand $\alpha = \bar{\delta}$, then $\rho^*_\alpha = \rho^*_{\bar{\delta}} = -\rho^*_\delta = -\rho_\delta = \rho_{\bar{\delta}}$.



y
$a_j(\rho_{j,1,2};\rho_{j,3,1})$
$a_3(0;1)$
$a_i(\rho_{i,1,2};\rho_{i,3,1})$
$a_1(0;0)$
$a_2(1;0)$
x

Fig. 5. The case $P = \{1\}$.

- The case $P = \{1\}$ (see fig. 5). In other words the triangles $\triangle_\alpha$ share $\triangle_\beta$ a single common vertex located in the origin.

  Assume that the rest to vertices are a_i and a_j without the generality loss. Consider the triangle $\triangle a_1 a_i a_j$ with oriented area

  $$S'(\triangle a_1 a_i a_j) = \frac{1}{2}\cdot\det\begin{pmatrix} X(a_i) & Y(a_i) \\ X(a_j) & Y(a_j)\end{pmatrix} = \frac{1}{2}\cdot(\rho_{i,3,1}\cdot\rho_{j,1,2} - \rho_{j,3,1}\cdot\rho_{i,1,2}).$$

  Divide the equality by $S'(\triangle a_1 a_2 a_2) = 1/2$, we will have $\rho^*_{1,i,j} = \rho_{i,3,1}\cdot\rho_{j,1,2} - \rho_{j,3,1}\cdot\rho_{i,1,2} = \rho_{1,i,j}$, where the last equality follows from (1).

- The general case: when i, j, k are arbitrary indices. Both ρ and ρ^* are additive. So $\rho^*_{i,j,k} = \rho^*_{1,i,j} + \rho^*_{1,j,k} - \rho^*_{1,i,k}$, that (as in the previous case) equals to $\rho_{1,i,j} + \rho_{1,j,k} - \rho_{1,i,k} = \rho_{i,j,k}$. Proof complete.

Getting back to the codes using non-oriented area ([4]).

Definition 2. Let call the *sign assignment* an arbitrary set of numbers $s_{\alpha,\beta} \in \{\pm 1\}$, where $\alpha, \beta \in \mathcal{A}$. Let call the sign assignment *consistent* if for all $\alpha, \beta, \gamma \in \mathcal{A}$ the following conditions hold:

1) $s_{\alpha\beta}\cdot s_{\beta\gamma} = s_{\alpha\gamma}$;

2) $s_{\pi(\alpha)\sigma(\beta)} = (-1)^\pi \cdot (-1)^\sigma \cdot s_{\alpha\beta}$, $\pi, \sigma \in S_3$.

Note 4. Obviously, a consistent sign assignment satisfies $s_{\alpha\alpha} = 1$ and $s_{\alpha\beta} = s_{\beta\alpha}$ for all $\alpha, \beta \in \mathcal{A}$.

Corollary 3. *The set of numbers $r^*_{\alpha\beta}$ is the code of a non-degenerate image if and only if there exists a consistent sign assignment $s_{\alpha,\beta}$, such that for $r_{\alpha\beta} = s_{\alpha\beta} \cdot r^*_{\alpha\beta}$ the conditions 1–5 and (1) hold.*

Proof

Necessity Just set $s_{\alpha,\beta} = 1$ if the triangles $\triangle_\alpha$ and $\triangle_\beta$ have the same orientation. Then apply theorem 1.

Sufficiency Construct an image with code matrix $R = ((r_{\alpha\beta}))$ by theorem 1. Then take $r^*_{\alpha\beta} = |r_{\alpha\beta}|$.

4. Conclusion

The main result of this paper completely describe the set of non-degenerate images codes. The future plans are to build similar conditions for other coding functions, e.g., projective equivalence preserving coding functions or for 3-D affine equivalence preserving.

References

[1] Агниашвили П.Г., “Однозначность восстановления изображения по его коду в n-мерном случае”, *Интеллектуальные системы*, **15**:1–4 (2011), 293–332.

[2] Алексеев Д.В., “К вопросу о восстановлении трехмерного тела по его плоским проекциям”, *Интеллектуальные системы. Теория и приложения*, **21**:4 (2017), 66–85.

[3] Козлов В.Н., “Доказательность и эвристика при распознавании визуальных образов”, *Интеллектуальные системы*, **14**:1–4 (2010), 35–52.

[4] Козлов В.Н., *Элементы математической теории зрительного восприятия*, Изд–во ЦПИ при мех.–мат. ф–те МГУ, Москва, 2001, 128 с.

[5] Козлов В.Н., “О кодировании дискретных фигур”, *Дискретная математика*, **8**:6 (1996), 57–61.

[6] Kozlov V.N., “Image Coding and Recognition and Some Problems of Stereovision”, *Pattern Recognition and Image Analysis*, **7**:4 (1997), 448–466.

Cellular automata with locators [1]

E. E. Gasanov[2]

This article introduces a new mathematical object called a cellular automaton with locators. It was created by implementing new functionality for an automaton to broadcast broadcasting signals and to receive summarized broadcasting signal of all elementary automata. This article highlights several problems which solution is greatly simplified by using cellular automata with locators instead of traditional cellular automata.

Keywords: cellular automata, homogeneous structures, firing squad problem, motion picture design, constructing the shortest path.

1. Introduction

Cellular automata (other names: self-reproducing automata and homogeneous structures) are discrete mathematical models of a wide class of real systems along with the processes taking place in them.

Theory of self-reproducing automata was introduced by John von Neumann[1, 2] to describe the processes self-reproduction in biology and technology. His model was further developed and the term “Cellular automaton” as it described below was used by A. Burks [3], E. Moore [4], V. B. Kudryavtsev, A. S. Podkolzin, A. A. Bolotov [5] and other researchers.

Cellular automaton is a mathematical object with discrete space and time. Its every position in space represented by a single cell, and each moment in time represented by discrete time step or generation. The state of each spatial cell is determined by very simple rules of interaction. These rules prescribe changes in the state of each cell in the next time step in response to the current state of neighboring cells. Moreover, for different cells, the rules for changing states may be different.

If we choose a finite automaton as a transformer of information standing in a cell of space, the same one for all cells, then we come to the concept of a homogeneous structure. In this case, the cellular automaton is an infinite automaton circuit constructed as follows. Consider the k-dimensional Euclidean space. We divide it into hypercubes with a unit edge, the edges of which are parallel to the coordinate axes. In each hypercube we put the same finite automaton V with m inputs and one output. We branch the output of

[1] Originally published in *Intellektualnye Sistemy. Teoriya i prilogeniya* (2020) **24**, No. 2, 121-133 (in Russian).

[2] *Gasanov Elyar Eldarovich* — professor, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intelligent Systems, e-mail: el_gasanov@gmail.com.

the automaton and connect it with the inputs of its neighbors in the same way for all hypercubes in space. We get an infinite homogeneous way arranged automaton scheme, which is called a cellular automaton. The sequence of states of individual automata V, containing the states of all automata of the circuit, form the state of the cellular automaton. The sequence of states of a cellular automaton arising from the synchronous operation of all its individual automata is called the functioning of a cellular automaton.

Cellular automata are a discrete mathematical model of a wide class of real systems along with processes occurring in them, such as physical media in which thermal and wave phenomena are realized, chemical solutions with reactions in them, biological tissues in which metabolism occurs, and technical control schemes processing mechanical and electrical signals, computational circuits, etc.

If we set the initial states of the automata, then in the circuit the states of the automata start to change in the way determined by the laws of the functioning of automata and the relationships between them. The phenomenon of a global change in these states is the main object of study in the theory of cellular automata.

This article introduces the generalization of a cellular automaton, which is proposed to be called a cellular automaton with locators.

One of the serious limitations of cellular automata is the limitedness of the neighborhood pattern, i.e. each automaton can see a certain number of its neighbors, and thus signals in cellular automata propagate relatively slowly. It is proposed to provide cellular automata with the ability to transmit some signals to all elementary automata at the same time, which will overcome the locality property.

Here we can recall the model of an incompressible fluid, in which the signals also instantly propagate throughout the entire volume. A similar picture is observed in quantum mechanics and in quantum cellular automata [6], when a change in the state of one automaton causes a change in the state of all automata “entangled” with it. In the work [7], the concept of nonlocal cellular automata is introduced. In this paper, nonlocality means that for each elementary automaton, the set of its neighbors is chosen randomly, and thus, elementary automata that are far apart from each other can be neighboring.

In real life, a person, when he wants to transmit information not only to visible neighbors, he can take advantage of such techniques as the supply of light signals using signal flares. An even more common method is the use of radio and television broadcasts.

Here, we also introduce the concept of broadcasting. Each elementary automaton is considered to be able to broadcast some signal from the finite alphabet on the air. Elements of the alphabet form a finite additive commutative semigroup, and the air itself is a potentially infinite adder of signals of elementary automata, where the defining operation of this semigroup acts as a sum. At the next clock, each elementary automaton receives a total signal from the air and changes its state according to the signal. In nature, such an adder is air that sums all the radio signals in a natural way, and in fact each of the receivers gets the same signal at the input, and only then it extracts the necessary component from the general signal.

With the help of this principle, one can implement a new type of integrated circuits that use some substrate as an adder, onto which all elementary automata will dump some switching or emergency signals.

Introduction of broadcasting concept and the ability to transmit signals instantly at any distance allows one elementary automaton to control the behavior of another elementary automaton arbitrarily far from it. We consider cellular automata with locators that can receive signals from certain directions. In other words, each elementary automaton has several locators directed in different directions, and it can use these locators to receive signals from the very directions.

This article introduces a formal model of cellular automata with locators. The solution of several tranditional problems and new challenges using standard cellular automata is given. Then, it is shown that the same problems can be solved much easier using cellular automata with locators.

2. The concept of cellular automaton with locators

We introduce the concept of a cellular automaton with locators based on the definition of a cellular automaton from [8].

By a *solid angle* in $\mathbb{R}^k$ we mean the union of all the rays in the space $\mathbb{R}^k$ emanating from a given point (*vertex of an angle*) and intersecting some hypersurface in $\mathbb{R}^k$. We assume that a solid angle does not contain its vertex. In particular, in this paper we consider two degenerate cases: the full solid angle coinciding with $\mathbb{R}^k$ without the vertex of the angle, which we denote by Ω, and solid angles equal to one ray. If a solid angle is a ray, we denote it by a vector defining its direction.

A *cellular automaton with locators* is a 8-tuple

$$\sigma = (\mathbb{Z}^k, E_n, V, E_q, +, L, \varphi, \psi)$$

where $\mathbb{Z}^k$ is the set of k-dimensional vectors with integer coordinates, $E_n = \{0, 1, \ldots, n-1\}$, $V = (\alpha_1, \ldots, \alpha_{h-1})$ is an ordered set of pairwise different nonzero vectors from $\mathbb{Z}^k$, $E_q = \{0, 1, \ldots, q-1\}$, $+$ is a commutative semigroup operation defined on E_q, $L = (\nu_1, \ldots, \nu_m)$ is an ordered set of pairwise different solid angles in $\mathbb{R}^k$ with a vertex at the origin, $\varphi : E_n^h \times E_q^m \to$

E_n is a function depending on the variables $x_0, x_1, \ldots, x_{h-1}, z_1, \ldots, z_m$ such that $\varphi(0, \ldots, 0) = 0$, $\psi : E_n^h \times E_q^m \to E_q$ is a function that depends on the variables $x_0, x_1, \ldots, x_{h-1}, z_1, \ldots, z_m$. Here the variables $x_0, x_1, \ldots, x_{h-1}$ take values from E_n and the variables $z_1, \ldots, z_m$ take values from E_q. Elements of the set $\mathbb{Z}^k$ are called *cells* of the cellular automaton σ; elements of the set E_n are called *cell states* of the cellular automaton σ; the set V is called the *neighborhood pattern* of the cellular automaton σ; elements of the set E_q are called *broadcasting signals*; the set L is called the *locator pattern* of the cellular automaton σ; the function φ is called the *local transition function* of the automaton σ; the function ψ is called the *broadcasting function* of the automaton σ. The state 0 is interpreted as *quiescent state* and the condition $\varphi(0, \ldots, 0) = 0$ is interpreted as a condition for maintaining the quiescent state.

Here we need to introduce an ordering of the neighborhood pattern V and the locator pattern L in order to establish a one-to-one correspondence between vectors from V and solid angles from L and variables x_0, $x_1, \ldots, x_{h-1}$, $z_1, \ldots, z_m$ of the local transition function φ and the broadcasting function ψ respectively. We can make this correspondence more explicit if we index the variables of the functions φ and ψ by the vectors and solid angles themselves, i.e. assume that the local transition function φ and the broadcasting function ψ depend on the variables $x_{\mathbf{0}}$, $x_{\alpha_1}, \ldots, x_{\alpha_{h-1}}$, $z_{\nu_1}, \ldots, z_{\nu_m}$, where the index of the first variable is the zero vector $\mathbf{0} = (0, \ldots, 0) \in \mathbb{Z}^k$. If we index the variables of the local transition function and broadcasting function in this way, we can write them in any order, and then we can define the neighborhood pattern and the locator pattern simply as a set, not an ordered set.

In the quiescent of this section we use these conventions: consider the neighborhood pattern as a set of vectors, and the locator pattern as a set of solid angles and index the variables of the local transition function and broadcasting function by the vectors from the neighborhood pattern and solid angles from the locator pattern. At the same time, we often omit the outer parentheses of the vectors in the indices. For example, if $k = 2$, $n = 2$, $q = 2$, $V = \{(-1, 0), (1, 0)\}$, and $L = \{\Omega, (0, 1)\}$, then a local transition function may look like this: $\varphi = x_{-1,0} \& z_\Omega \vee x_{1,0} \& z_{0,1}$.

If $\alpha \in \mathbb{Z}^k$, ν is a solid angle with vertex at the origin, then we denote by $\nu(\alpha)$ the solid angle obtained by translation of the angle ν to the point α.

If $\alpha \in \mathbb{Z}^k$ is a cell of a cellular automaton with locators σ, then the set $V(\alpha) = \{\alpha, \alpha + \alpha_1, \ldots, \alpha + \alpha_{h-1}\}$ is called the *neighborhood of the cell* α, and elements of the set $L(\alpha) = \{\nu_1(\alpha), \ldots, \nu_m(\alpha_m)\}$ are called *locators of the cell* α.

A *state of a cellular automaton with locators* σ is a pair (e, f), where e is an arbitrary function from the set $\mathbb{Z}^k$ to the set E_q, called *broadcast state*, f

is an arbitrary function from the set $\mathbb{Z}^k$ to the set E_n and called *distribution of states of the cellular automaton with locators* σ. Such a function can be interpreted as a certain mosaic arising in the k-dimensional space as a result of assigning a certain state from the set E_n and some signal from the set E_q to each point with integer coordinates. The set of all possible states of a cellular automaton with locators is denoted by Σ.

If $\alpha \in \mathbb{Z}^k$ and (e, f) is a state of a cellular automaton with locators σ, then the value $e(\alpha)$ is called *the signal of the cell* α*, defined by the state* (e, f), and the value $f(\alpha)$ is *the state of the cell* α*, determined by the state* (e, f). For each $i \in \{1, \ldots, m\}$ the value

$$s_i(\alpha) = \sum_{\beta \in \nu_i(\alpha) \cap \mathbb{Z}^k} e(\beta) \tag{1}$$

we call *the value of the locator* ν_i*, determined by the state* (e, f). Here, the semigroup operation $+$ defined on E_q is used to sum signals.

On the set Σ we define the *global transition function* Φ of a cellular automaton with locators σ, putting $\Phi(e, f) = (e', f')$, where $(e, f), (e', f') \in \Sigma$ and for any cell $\alpha \in \mathbb{Z}^k$ the following identities hold

$$f'(\alpha) = \varphi(f(\alpha), f(\alpha + \alpha_1), \ldots, f(\alpha + \alpha_{h-1}), s_1(\alpha), \ldots, s_m(\alpha)), \tag{2}$$

$$e'(\alpha) = \psi(f(\alpha), f(\alpha + \alpha_1), \ldots, f(\alpha + \alpha_{h-1}), s_1(\alpha), \ldots, s_m(\alpha)). \tag{3}$$

A meaningful interpretation of the mapping Φ is that the signal of each cell and the state of each cell "after the transition" is determined by the state of the neighborhood of the cell and by the values of the locators "before the transition" using the rules ψ and φ in the same way for all cells.

By the *behavior of a cellular automaton with locators* σ we call a sequence $(e_0, f_0), (e_1, f_1), (e_2, f_2), \ldots$ of states such that the equation $(e_{i+1}, f_{i+1}) = \Phi(e_i, f_i)$ holds for all $i = 0, 1, 2, \ldots$. The state (e_i, f_i) is called the *state of the cellular automaton with locators* σ *at the time* i, and (e_0, f_0) is also called the *initial state of the cellular automaton with locators* σ.

A state of a cellular automaton is called a *configuration* if only a finite number of cells are in a state other than 0 and the signals of all the cells are zero. The set of configurations is denoted by Σ'.

If a certain state of a cellular automaton is specified, then cells that are in a state other than 0 are called *active*.

Furher, we demonstrate several problems for cellular automata and show how their solutions are significantly simplified in case of using cellular automata with locators.

3. A firing squad problem

A firing squad problem was first proposed by J. Mayhill in 1957 and published (with a solution) in 1968 by F. Moore [9].

In this problem, we consider a one-dimensional cellular automaton on ^{1}. The neighborhood pattern is $V = \{-1, 1\}$, i.e. each cell has two neighbors left and right. The set of states have at least three states: 0 — quiescent state, 1 — soldier in initial state, 2 — fire. There is a restriction on the transition function that a soldier in the initial state, having neighbors of such soldiers, does not change the state, i.e. $\varphi(1, 1, 1) = 1$. The initial configuration is a continuous segment of r cells in state 1 (soldiers), and all other cells are in quiescent state 0. It is necessary that at some point in time all active cells switch to the state 2 at the same time (fired).

The standard solution to the above problem contains two waves of states propagating through a number of soldiers, one of which moves three times faster than the other one. The faster wave is reflected from the far edge of the row and meets the slower one in the center. After that, two waves are divided into four waves, moving in different directions from the center. The process continues, each time doubling the number of waves, until the length of the segments of the row becomes equal to 1. At this moment, all the soldiers shoot. This solution requires $3r$ time units for r soldiers.

The cellular automaton with the locators σ_0, which solves the firing squad problem, has the following form $\sigma_0 = (^1, E_3, V = \{-1, 1\}, E_2, \vee, L = \{\Omega\}, \varphi, \psi)$, where $\vee$ is the disjunction taken as the determining operation on broadcasting function ψ takes the value 1 only for the leftmost soldier in the initial state, i.e. $\psi(x_0, x_{-1}, x_1, z_\Omega) = x_0\bar{x}_{-1}\bar{z}_\Omega$, the local transition function takes the value 2 only if the cell is in state 1 and the broadcasting signal is 1, and does not change state in all other cases, i.e.

$$\varphi(x_0, x_{-1}, x_1, z_\Omega) = \max(2 \cdot ((x_0 = 1)\&(z_\Omega = 1) \vee (x_0 = 2)), 1 \cdot (x_0 = 1)).$$

Hence, the firing squad problem can be solved in 2 clocks using three states and two broadcasting signals.

Note that the described solution is fully consistent with the real-life protocol used by the military: the commander (the leftmost soldier) commands “fire” and the whole squad shoots.

4. Unidirectional movement of a point on the ray

The problem of unidirectional motion of a point on the ray, proposed and solved in the paper [10] by E.E. Titova, is as follows.

The set of cells is a set of natural numbers, i.e. ray is directed to the right. Each cell has two neighbors, one on the left and one on the right of itself. Some of the states of the cells are called labels and considered black, other states are considered white. Configurations are considered to be correct when there is exactly one black cell (called a point) on the ray. The cell corresponding to the number 1 (the leftmost cell) does not have a neighbor on the left of itself, and we will consider the variable corresponding to the state of the neighbor on the left of this cell as a control input, to which we can apply any control actions. The described set of cells with one control input will be called the *screen*.

Formally, the *screen* is the cellular automaton $S = (, E_n, V = \{-1, 1\}, \varphi, M)$, where n is the number of states of the cell of the cellular automaton, and $\varphi : E_n^3 \to E_n$ is a local transition function, M is a set of labels, $M \subset E_n$, $0 \notin M$. If the cell is in a state of M, then informally we believe that it is painted black, otherwise it is painted white. We call the variable x_{-1} of the local transition function of the cell corresponding to the number 1, (of the *leftmost cell*) *the control input of the screen S*.

Law of motion is an infinite sequence (superword) of zeros and ones. If $F = f_1, f_2, f_3, \dots$ is the law of motion, then we denote by $F(t)$ the t-th element of the sequence, i.e. $F(t) = f_t$.

We say that *on the screen S the point moves according to the law F*, if the following conditions are satisfied:

1) at some point in time, a label appears in the leftmost cell of the screen (before that there are no labels on the screen), this moment is called *the movement start*;

2) changing the position of the label on the screen at the t-th moment from the movement start corresponds to the t-th letter in the superword F, namely, if $F(t) = 0$, then in $(t+1)$-th moment the label remains in the same cell where it was at the current moment, if $F(t) = 1$, then in $(t+1)$-th moment the label moves one cell to the right, compared to its current position;

3) at each moment of time after the movement start, there is exactly one label on the screen.

A S screen will be called *universal for the set of laws of motion $\mathcal{F}$* if for any F from $\mathcal{F}$ there is such a control sequence supplied to the control input of the screen that provides a point movement by the law F on the screen.

We denote by $\mathcal{F}^s$ the set of such laws of motion F that do not contain more than s ones in a row.

The following theorems are proved in [10].

Theorem 1. *For any screen S, there exists a law of motion $F \in \{0, 1\}^\infty$ such that it is impossible to realize the motion of a point according to the law F on the screen S.*

Theorem 2. *There is a law of motion $F \in \{0,1\}^\infty$, the movement of which cannot be realized on any screen S.*

Theorem 3. *There is a universal screen with $2s+2$ states for the set of laws of motion $\mathcal{F}^s$.*

The question of describing the set of all realized laws of motion remains open, although G.V. Kalachev and E.E. Titova [11] have significantly advanced in this direction.

We present a cellular automaton with locators that solves the problem of unidirectional motion of a point on the ray.

Consider the following cellular automaton with the locators $\sigma_1 = (, E_2, V = \{-1,1\}, E_2, \vee, L = \{\Omega\}, \varphi, \psi, M = \{1\})$, where $\vee$ is the disjunction taken as the determining operation on the semigroup of signals $E_2 = \{0,1\}$, the locator pattern consists of one full solid angle Ω, the set of labels consists of one character 1, broadcasting function ψ is identically zero, the local transition function takes the value 1 in only two cases: if the cell is in state 1 and the broadcasting signal is 0, or if the cell on the left is in state 1 and the broadcasting signal is 1, i.e. $\varphi(x_0, x_{-1}, x_1, z_\Omega) = x_0 \& \bar{z}_\Omega \vee x_{-1} \& z_\Omega$.

We assume that the variable x_{-1} of the local transition function of the leftmost cell is the control input. In addition, we will consider that signals from E_2 can be broadcasted as control actions.

It is clear that in order to start moving, you need to send 1 to the control input, as well as send 1 to the air. As a result, the label appears on the screen in the leftmost cell. Further, in order to realize the law of motion F, it is necessary to send the value $F(t)$ to the air at the time t.

Thus, using cellular automata with locators, any law of motion can be realized, and the number of states of this automaton is 2, and the cardinaity of the broadcasing alphabet is 2.

In fact, with the help of signals on the air we give commands to the point, move it or stand.

5. Construction of the shortest path

The problem of constructing the shortest path for cellular automata is as follows. In the initial configuration, there are only two cells in the active state, which we will call the starting points. The shortest path is considered to be built if, at some point in time, the configuration becomes stable, and the active cells of this configuration form the shortest path between the starting points.

An adaptation of the traditional way to solve this problem to cellular automata is provided in [12]. Such adaptation involves the presence of three stages:

1) A propagation of an expanding signal from one of the starting points. When expanding, each cell remembers where the signal came from. This will allow to carry out a reverse move later.

2) When the wave reaches the second starting point, a reverse movement is carried out, leading to the first point, which gives the shortest path.

3) At the same time, a purification wave starts. This wave switches all the cells except the path cells into the quiescent state. In order for this wave to catch up with the expanding wave, the expanding wave from the first stage must expand at a half speed, and the purification wave must expand at a unit speed.

The work [12] gives a proof that the automaton proposed by the authors has 14 states. This work does not estimate the time it takes to build the path, but it is not difficult to see that the time to build the path is no less than $6n$, where n is the Manhattan distance between the starting points. Here, $2n$ clock cycles are used at the first stage, and $4n$ cycles are necessary for the third stage. It is possible to speed up the process by launching an expanding wave from both starting points, but it is clear that the path construction time will be proportional to the distance between the starting points.

Now consider the solution to the problem by cellular automata with locators.

Consider the following cellular automaton with locators $\sigma_2 = ({}^2, E_2, V = \{(-1,0),(0,1),(1,0),(0,-1)\}, E_2, \vee, L = \{(-1,0),(0,1),(1,0),(0,-1)\}, \varphi, \psi)$, where $\vee$ — a disjunction taken as a determining operation on a semigroup of signals $E_2 = \{0,1\}$, a neighborhood pattern is "cross", a pattern of locators consists of four rays directed left, up, right and down, broadcasting function ψ takes the value 1 if the cell is in state 1 and one of four cases occurs: if all its neighbors are in state 0; the cell does not have a neighbor from above in state 1 and the upper locator receives signal 1; the cell does not have a left neighbor in state 1 and the left locator receives signal 1; the cell has no neighbor to the right in state 1 and the right locator receives signal 1; i.e.

$$\psi(x_0, x_{-1,0}, x_{0,1}, x_{1,0}, x_{0,-1}, z_{-1,0}, z_{0,1}, z_{1,0}, z_{0,-1}) = \\ = x_0(\bar{x}_{-1,0}\bar{x}_{0,1}\bar{x}_{1,0}\bar{x}_{0,-1} \vee \bar{x}_{0,1}z_{0,1} \vee \bar{x}_{-1,0}z_{-1,0} \vee \bar{x}_{1,0}z_{1,0});$$

the local transition function takes on value 1 if the cell was in state 1, or if signals from the air came to one of four pairs of locators at the same time: top and right, top and left, top and bottom, left and right, i.e.

$$\varphi(x_0, x_{-1,0}, x_{0,1}, x_{1,0}, x_{0,-1}, z_{-1,0}, z_{0,1}, z_{1,0}, z_{0,-1}) = \\ = x_0 \vee z_{0,1}z_{1,0} \vee z_{0,1}z_{-1,0} \vee z_{0,1}z_{0,-1} \vee z_{-1,0}z_{1,0}.$$

We show that the aforementioned cellular automaton with locators solves the problem of constructing the shortest path.

In the initial (zero) clock cycle on the plane, there are only two active cells, which we call the initial ones.

We consider various cases of the location of the initial cells.

Case 1. The initial cells are located on the same horizontal. We denote by A the left initial cell, and by B the right one.

Case 1.1. If the initial cells are adjacent, then this pair of cells makes up the shortest path. It remains to note that the signals will not be broadcasted on the air, therefore, new active cells will not appear, and, therefore, the configuration will remain stable.

Case 1.2. If the initial cells are not adjacent, then the broadcasting function of each of the initial cells will become equal to 1, since these cells have no neighbors. Consequently, on step 1, a signal will be broadcast from A and B cells, and for all cells between A and B cells, the left and right locators will receive signals. Therefore, the local transition functions of these cells will take the value 1. Therefore, on step 2, all cells between A and B will go to state 1. The shortest path is constructed. Since all cells have neighbors, the broadcasting function of all cells will take the value 0, and the resulting configuration will remain stable.

Case 2. Initial cells are located on one vertical. This case is proved similarly to the case 1.

Case 3. The initial cells are in general position. Let's mentally draw vertical and horizontal lines through the initial cells. As a result, we get an imaginary rectangle with sides parallel to the coordinate axes, at the two diagonal vertices of which the initial cells are located.

Case 3.1. One initial cell is located in the upper left corner of the rectangle (we denote it by A), and the second initial cell is located in the lower right corner (we denote it by B).

Since the initial cells have no neighbors, the broadcasting function of these cells will take the value 1. Therefore, on step 1, a signal will go from the cells A and B on the air. The cell located in the lower left corner of the rectangle (denoted by C) will receive signals from the upper and right locators. Therefore, its local transition function will take the value 1. Therefore, at step 2, the cell C will become active.

Case 3.1.1. The cell C is adjacent to both A and B. Therefore, the shortest path is built. All cells have neighbors, therefore, signals will not be broadcasted anymore, and the configuration will remain stable.

Case 3.1.2. The cell C is not adjacent to A, and is adjacent to B. Then the broadcasting function of the cell C will take the value 1. Therefore, on step 3, signals from two cells will go on the air: A and C. This means that all cells between A and C will receive signals to their upper and lower locators. As a consequence, their local transition function will take the value 1. Hence, on step 4, all cells between A and C will become active. Now all cells have neighbors, therefore, signals will not be broadcasted anymore, and the configuration will remain stable.

Case 3.1.3. The cell C is adjacent to A, and is not adjacent to B. This case is proved similarly to the case 3.1.2.

Case 3.1.4. The cell C is adjacent to neither A nor B. Then the broadcasting function of the cell C will take the value 1. Thus, on step 3, signals from three cells will go on the air: A, B and C. This means that all cells between A and C will receive signals to their upper and lower locators, and all cells between B and C will receive signals to their left and right locators. Thereby, the local transition function of all these cells will take the value 1. Then, on step 4, all cells between A and C and all cells between B and C will become active. The shortest path will be built. All cells have neighbors, with the result that signals will not be broadcasted anymore, and the configuration will remain stable.

Case 3.2. One initial cell is located in the lower left corner of the imaginary rectangle, and the second initial cell is located in the upper right one. The proof is similar to the proof of the case 3.1.

Finally, we have shown that the proposed cellular automaton with locators allows us to build the shortest path in no more than 4 clocks. Moreover, it has only 2 states and 2 broadcasting signals.

Список литературы

[1] Von Neumann J., *Collected works*, New York, 1961 – 1963.

[2] Von Neumann J., *Theory of self-reproducing automata*, London, 1966.

[3] Burks A. W., *Essays on Cellular Automata*, Urban, IL: University of Illinois Press, 1970.

[4] Moore E. F., “Machine models of self-reproduction”, *Proceedings pf Symposia in Applied Mathematics*, **14** (1962), 17–33.

[5] Kudryavtsev V. B., Podkolzin A. S., Bolotov A. A., *Fundamentals of the theory of homogeneous structures*, Nauka, Moscow, 1990 (In Russian).

[6] Arrighi P., “An overview of Quantum Cellular Automata”, *arXiv:1904.12956v2 [quant-ph] 6 Sep 2019*, September 9, 2019, 1–23.

[7] Li W., “Phenomenology of Non-local Cellular Automata”, *Stat. Phys.l*, **68**:5/6 (1992), 829-882.

[8] Kudryavtsev V. B., Gasanov E. E., Podkolzin A. S., *Theory of Intelligent Systems: in 4 books. Book Four. Theory of Automata*, Publishing Solutions, Moscow, 2018 (In Russian).

[9] Moore F. R., Langdon G. G., “A generalized firing squad problem”, *Information and Control*, **12**:3 (March 1968), 212–220.

[10] Titova E. E., “Designing moving images by cellular automata”, *Intelligent systems*, **18**:1 (2014), 153–180 (In Russian).

[11] Kalachev G. V., Titova E. E., “On the measure of the set of laws of motion of a point realized by cellular automata”, *Intelligent systems. Theory and Applications*, **22**:3 (2018), 105–125 (In Russian).

[12] Hochberger C., Hoffmann R., “Solving routing problems with cellular automata”, *Proceedings of the Second Conference on Cellular Automata for Research and Industry*, Octber 1996, 89–98.

The one-dimensional closest neighbor search problem solution using the cellular automata with locators [1]

D. I. Vasilev[2]

The paper considers applying the locator cellular automaton model to the closest neighbour search problem. The locator cellular automaton model assumes the possibility for each cell to translate a signal through any distance using ether. It is proven in this paper that such possibility allows to decrease the problem complexity from linear to logarithmic (against the classic cellular automaton model).

Keywords: cellular automaton, homogeneous structures,the closest neighbour search problem.

1. Introduction

Cellular automata (other names: self-reproducing automata and homogeneous structures) are discrete mathematical models of a wide class of real systems along with the processes taking place in them.

Theory of self-reproducing automaton was introduced by John von Neumann[2, 1] to describe the processes self-reproduction in biology and technology. His model was further developed and the term "Cellular automaton" as it described below was used by A. Burks [3], E. Moore [4], V. B. Kudryavtsev, A. S. Podkolzin, A. A. Bolotov [5] and other researchers.

Cellular automaton — is a mathematical object with discrete space and time. Its every position in space represented by a single cell, and each moment in time represented by discrete time step or generation. The state of each spatial cell is determined by very simple rules of interaction. These rules prescribe changes in the state of each cell in the next time step in response to the current state of neighboring cells.

In the paper of Gasanov E.E. [9] the concept of a cellular automaton with locators was introduced, which differs from the concept of a classic cellular automaton in that it allows the transmission of information not only between neighboring cells, but also at any distance, by means of transmitting a signal to the ether. The paper considers the application of this model to the one-dimensional closest neighbor search problem: a special point called

[1] Originally published in *Intellektualnye Sistemy. Teoriya i prilogeniya* (2020) **24**, No. 3, 99-119 (in Russian).

[2] *Vasilev Denis Igorevich* — junior researcher, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intelligent Systems, email: denis.vasilev.igor@gmail.com

"central"and some finite set of "target"cells are arbitrarily marked on $\mathbb{Z}^1$; the problem is to understand which of the target points is closer to the central one. The classic model of a cellular automaton solves this problem in linear time (by the minimal distance between the central and the target points). In this paper, it will be shown that the problem can be solved in logarithmic time via the cellular automaton with locators model.

The author expresses gratitude to Professor E.E.Gasanov for setting the problem and Ph.D. G.V.Kalachev for valuable comments and suggestions.

2. The problem description and results formulation.

In the paper of Gasanov E.E. [9] the concept of a cellular automaton with locators was introduced. Here we will give this concept, narrowing it down to the one-dimensional case.

By a *solid angle* in $\mathbb{R}^k$ we mean the union of all the rays in the space $\mathbb{R}^k$ emanating from a given point (*vertex of an angle*) and intersecting some hypersurface in $\mathbb{R}^k$. In the definition, we assume that a solid angle does not contain its vertex. In particular, in this paper we consider two degenerate cases: the full solid angle coinciding with $\mathbb{R}^k$ without the vertex of the angle, which we denote by Ω, and solid angles equal to one ray. If a solid angle is a ray, we denote it by a vector defining its direction.

A *cellular automaton with locators* is a 8-tuple

$$\sigma = (\mathbb{Z}^k, E_n, V, E_q, +, L, \varphi, \psi)$$

where $\mathbb{Z}^k$ is the set of k-dimensional vectors with integer coordinates, $E_n = \{0, 1, \ldots, n-1\}$, $V = (\alpha_1, \ldots, \alpha_{h-1})$ is an ordered set of pairwise different nonzero vectors from $\mathbb{Z}^k$, $E_q = \{0, 1, \ldots, q-1\}$, $+$ is a commutative semigroup operation defined on E_q, $L = (\nu_1, \ldots, \nu_m)$ is an ordered set of pairwise different solid angles in $\mathbb{R}^k$ with a vertex at the origin, $\varphi : E_n^h \times E_q^m \to E_n$ is a function depending on the variables $x_0, x_1, \ldots, x_{h-1}, z_1, \ldots, z_m$ such that $\varphi(0, \ldots, 0) = 0$, $\psi : E_n^h \times E_q^m \to E_q$ is a function that depends on the variables $x_0, x_1, \ldots, x_{h-1}, z_1, \ldots, z_m$. Here the variables $x_0, x_1, \ldots, x_{h-1}$ take values from E_n and the variables $z_1, \ldots, z_m$ take values from E_q. Elements of the set $\mathbb{Z}^k$ are called *cells* of the cellular automaton σ; elements of the set E_n are called *cell states* of the cellular automaton σ; the set V is called the *neighborhood pattern* of the cellular automaton σ; elements of the set E_q are called *broadcasting signals*; the set L is called the *locator pattern* of the cellular automaton σ; the function φ is called the *local transition function* of the automaton σ; the function ψ is called the *broadcasting function* of the automaton σ. The state 0 is interpreted as *rest state* and the condition $\varphi(0, \ldots, 0) = 0$ is interpreted as a condition for maintaining the rest state.

Here we need to introduce an ordering of the neighborhood pattern V and the locator pattern L in order to establish a one-to-one correspondence between vectors from V and solid angles from L and variables $x_0,\ x_1,\dots,x_{h-1},\ z_1,\dots,z_m$ of the local transition function φ and the broadcasting function ψ respectively. We can make this correspondence more explicit if we index the variables of the functions φ and ψ by the vectors and solid angles themselves, i.e. assume that the local transition function φ and the broadcasting function ψ depend on the variables $x_{\mathbf{0}}$, $x_{\alpha_1},\dots,x_{\alpha_{h-1}},\ z_{\nu_1},\dots,z_{\nu_m}$, where the index of the first variable is the zero vector $\mathbf{0}=(0,\dots,0)\in\mathbb{Z}^k$. If we index the variables of the local transition function and broadcasting function in this way, we can write them in any order, and then we can define the neighborhood pattern and the locator pattern simply as a set, not an ordered set.

In the rest of this section we use these conventions: consider the neighborhood pattern as a set of vectors, and the locator pattern as a set of solid angles and index the variables of the local transition function and broadcasting function by the vectors from the neighborhood pattern and solid angles from the locator pattern. At the same time, we often omit the outer parentheses of the vectors in the indices. For example, if $k=2$, $n=2$, $q=2$, $V=\{(-1,0),(1,0)\}$, and $L=\{\Omega,(0,1)\}$, then a local transition function may look like this: $\varphi = x_{-1,0}\&z_\Omega \vee x_{1,0}\&z_{0,1}$.

If $\alpha\in\mathbb{Z}^k$, ν is a solid angle with vertex at the origin, then by $\nu(\alpha)$ we denote the solid angle obtained by translation of the angle ν to the point α.

If $\alpha\in\mathbb{Z}^k$ is a cell of a cellular automaton with locators σ, then the set $V(\alpha)=\{\alpha,\alpha+\alpha_1,\dots,\alpha+\alpha_{h-1}\}$ is called the *neighborhood of the cell* α, and elements of the set $L(\alpha)=\{\nu_1(\alpha),\dots,\nu_m(\alpha_m)\}$ are called *locators of the cell* α.

A *state of a cellular automaton with locators* σ is a pair (e,f), where e is an arbitrary function from the set $\mathbb{Z}^k$ to the set E_q, called *broadcast state*, f is an arbitrary function from the set $\mathbb{Z}^k$ to the set E_n and called *distribution of states of the cellular automaton with locators* σ. Such a function can be interpreted as a certain mosaic arising in the k-dimensional space as a result of assigning a certain state from the set E_n and some signal from the set E_q to each point with integer coordinates. The set of all possible states of a cellular automaton with locators is denoted by Σ.

If $\alpha\in\mathbb{Z}^k$ and (e,f) is a state of a cellular automaton with locators σ, then the value $e(\alpha)$ is called *the signal of the cell* α*, defined by the state* (e,f), and the value $f(\alpha)$ is *the state of the cell* α*, determined by the state* (e,f). For each $i\in\{1,\dots,m\}$ the value

$$s_i(\alpha)=\sum_{\beta\in\nu_i(\alpha)\cap\mathbb{Z}^k} e(\beta) \qquad (1)$$

we call *the value of the locator ν_i, determined by the state* (e, f). Here, in the summation the semigroup operation + defined on E_q is used.

On the set Σ we define the *global transition function* Φ of a cellular automaton with locators σ, putting $\Phi(e, f) = (e', f')$, where $(e, f), (e', f') \in \Sigma$ and for any cell $\alpha \in \mathbb{Z}^k$ the following identities hold

$$f'(\alpha) = \varphi(f(\alpha), f(\alpha + \alpha_1), \ldots, f(\alpha + \alpha_{h-1}), s_1(\alpha), \ldots, s_m(\alpha)), \quad (2)$$

$$e'(\alpha) = \psi(f(\alpha), f(\alpha + \alpha_1), \ldots, f(\alpha + \alpha_{h-1}), s_1(\alpha), \ldots, s_m(\alpha)). \quad (3)$$

A meaningful interpretation of the mapping Φ is that the signal of each cell and the state of each cell “after the transition” is determined by the state of the neighborhood of the cell and by the values of the locators “before the transition” using the rules ψ and φ in the same way for all cells.

By the *behavior of a cellular automaton with locators σ* we call a sequence $(e_0, f_0), (e_1, f_1), (e_2, f_2), \ldots$ of states such that the equation $(e_{i+1}, f_{i+1}) = \Phi(e_i, f_i)$ holds for all $i = 0, 1, 2, \ldots$. The state (e_i, f_i) is called the *state of the cellular automaton with locators σ at the time i*, and (e_0, f_0) is also called the *initial state of the cellular automaton with locators σ*.

Let’s formulate the closest neighbour search problem on the line. Let the I be the initial state of a cellular automaton on $\mathbb{Z}^1$ which satisfies the following conditions:

1) Any cell is on one of $\{q_S; q_{C_0}, *\}$ states.

2) There is only one q_{C_0} cell.

3) There is a finite and non-empty set of q_S cells.

We will define that a cellular automaton state I' is solution for the problem I if I' satisfies the following conditions:

1) The q_{C_0} cell from I is in q_{CF} state in I'.

2) The cell which is the closest to the q_{C_0} cell in I is in the q_{LE} state if it’s to the left and in q_{RE} state if it is ti the right. If there are two closest cell then the right cell must be in $*$ state and the left one — in q_{LE} state.

3) The cells which lie between q_{CF} and q_{LE} cells are in q_{LF} state. The cells which lie between q_{CF} and q_{RE} cells are in q_{RF} state.

4) The rest cells are in $*$ state.

We define that cellular automaton σ solves the closest neighbour search problem if it satisfies the following conditions:

1) If the initial state I of the cellular automaton is a closest neighbour search problem then the automaton must end up in I' state which is solution for I.

2) If the automaton takes state S which is solution for some closest neighbour search problem this state must be kept for all the next tacts.

There is a cell automaton with locators σ with 25 states and the ether alphabet power 12 which solves the closest neighbour search problem for not longer than $\log_2 s + 7$, where s is the distance between the q_{C_0} cell and the closest to it q_S cell.

No cell automaton with locators σ can solve the closest neighbour search problem faster than $\log_M(\frac{s}{5})$, where s is the distance between the q_{C_0} cell and the closest to it q_S cell and M is the ether alphabet power.

3. Formal automaton description

Let's consider cellular automaton $\sigma = (\mathbb{Z}^1, E_n, V, E_q, +, L, \varphi, \psi)$, where $V = \{(1), (-1)\}$, $E_q = \{0,1\}^2 \times \{0,1,2\}$, and $L = (\nu_{-1}, \nu_1)$, where ν_{-1}, ν_1 — degenerate solid angles, corresponding to vectors (-1) and (1).

Let's define the semigroup operation on E_q as follows: $(a_1, b_1, c_1) + (a_2, b_2, c_2) = (a_1 + a_2, max(b_1, b_2), max(c_1, c_2))$

Let the state set $E_n = \{q^{=}_{CC}; q^{<}_{CC}; q^{>}_{CC}; q_S; q_{C_0}; q_L; q_R;$ $q^C_2; q^C_1; q_{C_{L_1}}; q_{C_{R_1}}; q_{LF}; q_{RF}; q_{LE}; q_{RE}; q_{CF}; q^L_*; q^R_*; q^L_1; q^R_1; q^L_0; q^R_0; q^*_L; q^*_R; *\}$

Let's define state $*$ as the rest state. The automaton is designed in a way that only a limited set of cells will be in a non-rest state on each tact. Considering that operation $+$ has the property $(0,0,0) + (0,0,0) = (0,0,0)$, we can conclude that any locator's value is calculated from a limited set of non-zero terms, so it is defined correctly.

Let's describe φ and ψ functions for each automaton state:

q^L_k and q^R_k, $k \in 0;1$ are the key autmaton states. If there are some amount of q^L_1 consecutive cells then on the next tact every second q^L_1 cell will go to the q^L_0 state and the rest q^L_1 cells will keep their state. It will be proven later that such behavior allows to translate a segment length bit by bit.

$$\varphi(q^L_k, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} q^L_{(k \wedge z^1_{\nu_{-1}})}, \text{ if } z^3_{\nu_1} = 0 \\ q_{LF}, \text{ if } z^3_{\nu_1} = 1 \\ * \text{ in other cases} \end{cases}, \tag{4}$$

$$\psi(q^L_k, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (k \wedge z^1_{\nu_{-1}}, k \wedge z^1_{\nu_{-1}}, 0), \tag{5}$$

$$\varphi(q_k^R, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} q^R_{(k \wedge z^1_{\nu_1})}, \text{ if } , z^3_{\nu_{-1}} = 0 \\ q_{RF}, \text{ if } z^3_{\nu_{-1}} = 2 \\ * \text{ in other cases} \end{cases}, \tag{6}$$

$$\psi^1(q_k^R, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (k \wedge z^1_{\nu_1}, k \wedge z^1_{\nu_1}, 0). \tag{7}$$

q_{C_0} is the initial central cell state. The cell in this state will translate $(0, 0, 1)$ signal for other cells to determine if they are on the left or on the right side.

$$\varphi(q_{C_0}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = q_1^C, \tag{8}$$

$$\psi(q_{C_0}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 1). \tag{9}$$

q_S is the initial target cells state. The cell in this state will wait for a special signal to change it's state to the left or right version.

$$\varphi(q_S, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} q_L^*, \text{ if } z^3_{\nu_1} = 1 \\ q_R^*, \text{ if } z^3_{\nu_{-1}} = 1 \\ q_S \text{ in other cases} \end{cases}, \tag{10}$$

$$\psi(q_S, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} (0, 0, 1) \text{ if } z^3_{\nu_1} = 1 \\ (0, 1, 0) \text{ in other cases} \end{cases}. \tag{11}$$

$*$ is the initial state of internal cells (cells which are not central or target). If $*$-cell is part of the problem (i.e. it lies between the central cell and the side-closest target cell) it will wait for a special signal to change it's state on the left or right version. Otherwise such cell will keep calm as a rest cell.

$$\varphi(*, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} q_*^L, \text{ if } z^3_{\nu_1} = 1, z^2_{\nu_{-1}} = 1 \\ q_*^R, \text{ if } z^2_{\nu_1} = 1, z^3_{\nu_{-1}} = 1 \\ * \text{ in other cases} \end{cases}, \tag{12}$$

$$\psi(*, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0). \tag{13}$$

States q_*^L and q_*^R are designed for two purposes. First of all, they will provide right ether structure right before the main part of the algorithm will begin. Secondly, internal cells in this state are waiting for a special signal to determine if they lie between the side-closest and the central cell. If they are not, they go to the rest state.

$$\varphi(q_*^L, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} * \text{ if } z_{\nu_1}^3 = 1 \\ q_1^L \text{ in other cases} \end{cases}, \tag{14}$$

$$\psi(q_*^L, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} (0,0,0) \text{ if } z_{\nu_1}^3 = 1 \\ (0,1,1) \text{ in other cases} \end{cases}, \tag{15}$$

$$\varphi(q_*^R, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} * \text{ if } z_{\nu_{-1}}^2 = 1 \\ q_1^R \text{ in other cases} \end{cases}, \tag{16}$$

$$\psi(q_*^R, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} (0,0,0) \text{ if } z_{\nu_{-1}}^2 = 1 \\ (0,1,1) \text{ in other cases} \end{cases}. \tag{17}$$

q_L^* and q_R^* are special target cells states. A cell in this state will wait for a special signal to determine if it is the side-closest target cell. If it is not, the cell go to the rest state.

$$\varphi(q_L^*, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} * \text{ if } z_{\nu_1}^3 = 1 \\ q_L \text{ in other cases} \end{cases}, \tag{18}$$

$$\psi(q_*^L, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (0,0,0), \tag{19}$$

$$\varphi(q_R^*, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} * \text{ if } z_{\nu_{-1}}^2 = 1 \\ q_R \text{ in other cases} \end{cases}, \tag{20}$$

$$\psi(q_*^R, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (0,0,0). \tag{21}$$

$q_{CC}^=, q_{CC}^>, q_{CC}^<$ are working central cell states. The central cell compares lengths of the left and the right segments. Automaton works in a way that those lengths binary notation come to the central cell as ether signals: from the lowest bit to the highest. The central cell can change it's status depending on the current bit pair: $q_{CC}^>$ — if current left bit is greater than the right one, $q_{CC}^<$ — if it is less. If the left bit is equal to the right one, the central cell state inherits from the previous tact. State $q_{CC}^=$ occurs in the beginning and may stay until the first non-equal bit pair.

$$\varphi(q_{CC}^X, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} q_{CC}^X, \text{ if } z_{\nu_{-1}}^2 = 1, z_{\nu_1}^2 = 1, z_{\nu_{-1}}^1 = z_{\nu_1}^1 \\ q_{CC}^>, \text{ if } z_{\nu_{-1}}^2 = 1, z_{\nu_1}^2 = 1, z_{\nu_{-1}}^1 > z_{\nu_1}^1 \\ q_{CC}^<, \text{ if } z_{\nu_{-1}}^2 = 1, z_{\nu_1}^2 = 1, z_{\nu_{-1}}^1 < z_{\nu_1}^1 \\ q_{CF} \text{ in other cases} \end{cases}, \tag{22}$$

$$\psi(q_{C_1}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} (0,0,0), \text{ if } z^2_{\nu_{-1}} = 1, z^2_{\nu_1} = 1 \\ (0,0,1), \text{ if } (z^2_{\nu_{-1}} = 0 \vee z^2_{\nu_1} = 0) \wedge \\ \wedge((z^2_{\nu_{-1}} > z^2_{\nu_1}) \vee (z^2_{\nu_{-1}} = z^2_{\nu_1} \wedge q^X_{CC} \neq q^>_{CC})) \\ (0,0,2) \text{ in other cases} \end{cases}. \tag{23}$$

q_L and q_R are side-closest target cells states. They just mark the end of the segment and go to their final state when hear a special ether signal.

$$\varphi(q_L, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} q_{LE}, \text{ if } z^3_{\nu_1} = 1 \\ *, \text{ if } z^3_{\nu_1} = 2 \\ q_L \text{ in other cases} \end{cases}, \tag{24}$$

$$\psi(q_L, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (0,0,0), \tag{25}$$

$$\varphi(q_R, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} q_{RE}, \text{ if } z^3_{\nu_{-1}} = 2 \\ *, \text{ if } z^3_{\nu_{-1}} = 1 \\ q_R \text{ in other cases} \end{cases}, \tag{26}$$

$$\psi(q_R, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (0,0,0). \tag{27}$$

q^C_2 state is designed to for the central cell to looks for a special signal from each side so it could find out if there is at least one target cell at each side. If not, the problem is much easier and can be resolved in a constant time.

$$\varphi(q^C_2, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \begin{cases} q_{C_{L_1}}, \text{ if } z^2_{\nu_1} = 0 \\ q_{C_{R_1}}, \text{ if } z^2_{\nu_{-1}} = 0 \\ q^C_1, \text{ in other cases} \end{cases}, \tag{28}$$

$$\psi(q^C_2, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (0,0,0). \tag{29}$$

q^C_1 is a 1-tact sleep state. The central cell in this state don't send anything in the ether.

$$\varphi(q^C_1, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = q_{C_1}, \tag{30}$$

$$\psi(q^C_1, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (0,0,0). \tag{31}$$

$q_{C_{L_1}}$ and $q_{C_{R_1}}$ are states which occur when the problem is trivial in a way that all the target cells are located at the same side of the central cell.

$$\varphi(q_{C_{L_1}}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = q_{CF}, \tag{32}$$

$$\psi(q_{C_{L_1}}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 1), \tag{33}$$

$$\varphi(q_{C_{R_1}}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = q_{CF}, \tag{34}$$

$$\psi(q_{C_{R_1}}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 2). \tag{35}$$

ψ and φ functions below are function for the finish states. Those state do not change or send anything to the ether.

$$\varphi(q_{LF}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = q_{LF}, \tag{36}$$

$$\psi(q_{LF}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0), \tag{37}$$

$$\varphi(q_{RF}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = q_{RF}, \tag{38}$$

$$\psi(q_{RF}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0), \tag{39}$$

$$\varphi(q_{LE}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = q_{LE}, \tag{40}$$

$$\psi(q_{LE}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0), \tag{41}$$

$$\varphi(q_{RE}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = q_{RE}, \tag{42}$$

$$\psi(q_{RE}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0), \tag{43}$$

$$\varphi(q_{CF}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = q_{CF}, \tag{44}$$

$$\psi(q_{CF}, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = (0, 0, 0). \tag{45}$$

Let's define *left cells* as cells with L character in the state name (and correspondingly define *right cells*). Let $Q_1 = \{q_k^X\}$, where $X \in \{R, L\}, k = 1$, $Q_0 = \{q_k^X\}$, where $X \in \{R, L\}, k = 0$.

4. Automaton's behavior.

Let's describe automaton's behavior on each algorithm phase. We also will provide a simple example for better understanding. We will describe automaton states as follows:

	t=0															
Q	q_S	$*$	q_S	$*$	$*$	$*$	$*$	$*$	$*$	q_{C_0}	$*$	$*$	$*$	$*$	$*$	q_S
L^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
ψ^3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Where $Q = q_1, q_2, ..., q_n$ — automaton cells states. We only consider cells which lie between non-$*$ cell in the initial automaton state. All q_S and q_{C_0} cells are also considered.

$L^1 = l^1_1, l^1_2, ..., l^1_n$ — first components of the left ether sum.

$L^2 = l^2_1, l^2_2, ..., l^2_n$ — second components of the left ether sum.

$L^3 = l^3_1, l^3_2, ..., l^3_n$ — third components of the left ether sum.

$R^1 = r^1_1, r^1_2, ..., r^1_n$ — first components of the right ether sum.

$R^2 = r^2_1, r^2_2, ..., r^2_n$ — second components of the right ether sum.

$R^3 = r^3_1, r^3_2, ..., r^3_n$ — third components of the right ether sum.

$\psi^1 = \psi^1_1, \psi^1_2, ..., \psi^1_n$ — the first component of the signal, being sent to the ether

$\psi^2 = \psi^2_1, \psi^2_2, ..., \psi^2_n$ — the second component of the signal, being sent to the ether

$\psi^3 = \psi^3_1, \psi^3_2, ..., \psi^3_n$ — the third component of the signal, being sent to the ether

4.1. Phase 1: determine the orientation.

Let the initial automaton state satisfy the Theorem 1 conditions:

	t=0															
Q	q_S	$*$	q_S	$*$	$*$	$*$	$*$	$*$	$*$	q_{C_0}	$*$	$*$	$*$	$*$	$*$	q_S
L^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
ψ^3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

There are only $(0, 0, 0)$ signals in the ether on the first tact. Transition functions of q_S and $*$ states are equal to q_S and $*$ respectively when the input ether sums are $(0,0,0)$. Broadcasting functions of these states are equal to $(0,1,0)$ and $(0,0,0)$ respectively. The transition function of the q_{C_0} state is equal to q_2^C, and the broadcasting function takes the value $(0,0,1)$. Overall, only the central cell will change it's state on the first tact. Besides, the ether space will fill with central and target cells signals:

	t=1															
Q	q_S	$*$	q_S	$*$	$*$	$*$	$*$	$*$	$*$	q_2^C	$*$	$*$	$*$	$*$	$*$	q_S
L^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^2	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
L^3	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
R^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
R^3	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
ψ^3	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

The transition functions values of q_S and $*$ states depend on the side from which signals $(0,1,0)$ and $(0,0,1)$ came. q_S cells could either go to the q_L^* state and translate a $(0,0,1)$ signal or go to the q_R^* state and translate a $(0,1,0)$ signal. $*$ cells can go to q_*^L or q_*^R states and always output a $(0,0,0)$ signal. q_2^C cell looks for a $(0,1,0)$ signal from each side. If such a signal is only present on one side, central cell goes to $q_{C_{L_1}}$ or $q_{C_{R_1}}$ state (depending on the side on which $(0,1,0)$ signal occurs) and we don't need to compare segments from different sides. This is an easy case and the problem can be resolved in a constant time at this point. In our case the signal is present on both sides so the central cell goes to the q_2^C state to sleep for 2 tacts and wait before the other cells are ready to begin the length calculation:

	t=2															
Q	q_L^*	q_*^L	q_L^*	q_*^L	q_*^L	q_*^L	q_*^L	q_*^L	q_*^L	q_1^C	q_*^R	q_*^R	q_*^R	q_*^R	q_*^R	q_R^*
L^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^3	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
R^3	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	0
ψ^2	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	0
ψ^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This is how Phase 1 ends. To sum up, from all the initial q_S cells we only keep the two closest to the center. Those cells are now oriented (i.e. are either in q_L or q_R states); the $*$ cells which lie between the central cell and one of the two kept target cells will also go to the oriented state q_*^L or q_*^R; the central cell will go to the q_1^C state.

4.2. Phase 2: length comparison.

This is a complicated iterative phase.

We will call q_L and q_R cells *extreme*, and cells which lie between the central and an extreme cell — *internal*. Let's define that this phase ends when the central cell translate a signal $\psi \in \{(0,0,1);(0,0,2)\}$. The transition functions are designed in a way that the central cell is always in $q^{=}_{CC}, q^{>}_{CC}, q^{<}_{CC}$ state in Phase 2 and it can only go to another state when it send one of $\{(0,0,1);(0,0,2)\}$ signals. Finally, it's easy to see that signals $\{(0,0,1);(0,0,2)\}$ are only sent when all the internal cells from some side are in $q \in Q_0$ state.

Let's consider the left side of the cellular automaton. We will enumerate the cells from left to right and denote $q^{i,t}, i \in \{1,2,..s_L\}, t \geq 0$ as an i-th cell state in the t-th tact. We assume that $t = 0$ — the beginning of Phase 2.

$q^{i,t} \in Q_1$ if and only if $i(mod\ 2^t) = 0$.

Доказательство. Let 's prove this statement by induction:

At the $t = 0$ moment all the cells are $q^L_1 \in Q_1$ states, which is equivalent to $i(mod\ 2^0) = 0$.

Let's prove the Lemma for $t = 1$. From the transition functions of the $\{q^L_1; q^R_1; q^L_0; q^R_0\}$ states we can see that $q^{i,1} \in Q_1 \leftrightarrow q^{i,0} \in Q_1 \wedge z^{i,0}_{\nu_{-1}} = (1,\alpha,\beta) \leftrightarrow z^{i,0}_{\nu_{-1}} = (1,\alpha,\beta)$ (here and further α and β are arbitrary elements of the {0;1} and {0;1;2} sets respectively). We know that $z^{i,0}_{\nu_{-1}} = \sum\limits_{j=1}^{i} \psi(q^L_*, q_{-1}, q_1, z_{\nu_{-1}}, z_{\nu_1}) = \sum\limits_{j=1}^{i-1}(1,1,0) = (i(mod\ 2),1,0)$, therefore $q^{i,1} \in Q_1 \leftrightarrow i(mod\ 2) = 0$.

Let's assume the Lemma is true for $t = 0,1,2,...,k$ and prove it is true for $t = k+1$. From the transition functions of the $\{q^L_1; q^R_1; q^L_0; q^R_0\}$ states we can see that $q^{i,k+1} \in Q_1 \leftrightarrow q^{i,k} \in Q_1 \wedge z^{i,k}_{\nu_{-1}} = (1,\alpha,\beta)$. The broadcasting functions of the $\{q^L_1; q^R_1; q^L_0; q^R_0\}$ states are designed in a way that $\psi^{i,k-1} = (1,\alpha,\beta) \leftrightarrow q^{i,k} \in Q_1$, therefore $z^{i,k}_{\nu_{-1}} = (\sum\limits_{j(mod\ 2^k)=0, j<i, j>0} 1, \alpha, \beta)$. Thus, by the induction assumption:

$q^{i,k+1} \in Q_1 \leftrightarrow q^{i,k} \in Q_1 \wedge z^{i,k}_{\nu_{-1}} = (1,\alpha,\beta) \leftrightarrow i(mod\ 2^k) = 0 \wedge \sum\limits_{j(mod\ 2^k)=0, j<i, j>0} 1 = 1 \leftrightarrow i(mod\ 2^k) = 0 \wedge (i(mod\ 2^{k+1}) > 2^k \vee i(mod\ 2^{k+1}) = 0) \leftrightarrow i(mod\ 2^{k+1}) = 0$ □

We can also prove such lemma for the right side because the transition and broadcasting functions of the left and right cells only depend on left and right ether signals respectively.

It follows from the Lemma that at the t moment the central cell hear the value $\sum\limits_{i(mod\ 2^t)=0, i \in [1;s]} 1 = \lfloor \frac{s}{2^t} \rfloor (mod 2)$ at the 1-st ether dimension, which is equal to $(t+1)$-th bit in s length binary notion. Thus, the central cell can compare the values s_L and s_R bit by bit, from the lowest to the highest position. The transition and broadcasting functions of the q^X_{CC}, $X \in \{=, <, >\}$ states implement such length comparison algorithm: if left i-th bit is greater than such bit from the right then the central cell goes to the $q^{>}_{CC}$ state; if it is lower — to the $q^{<}_{CC}$ state; if they are equal, the state doesn't change. This algorithm continue working until one of the lengths translation ends ($z^2_{\nu_{-1}} = 0 \vee z^2_{\nu_1} = 0$). In this moment the central cell sends one of the signals $(0, 0, 2)$; $(0, 0, 1)$ depending on which side appeared to be shorter.

Phase 2 process for the given example:

	t=3															
Q	$*$	$*$	q_L	q^L_1	q^L_1	q^L_1	q^L_1	q^L_1	q^L_1	$q^{=}_{CC}$	q^R_1	q^R_1	q^R_1	q^R_1	q^R_1	q_R
L^1	0	0	0	0	1	0	1	0	1	0	0	1	0	1	0	1
L^2	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
L^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^1	1	1	1	0	1	0	1	0	1	1	0	1	0	1	0	0
R^2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
R^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	0	1	0	1	0	1	0	0	1	0	1	0	0
ψ^2	0	0	0	0	1	0	1	0	1	0	0	1	0	1	0	0
ψ^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	t=4															
Q	$*$	$*$	q_L	q^L_0	q^L_1	q^L_0	q^L_1	q^L_0	q^L_1	$q^{<}_{CC}$	q^R_0	q^R_1	q^R_0	q^R_1	q^R_0	q_R
L^1	0	0	0	0	0	1	1	0	0	1	1	1	0	0	1	1
L^2	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
L^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^1	1	1	1	1	0	0	1	1	0	0	0	1	1	0	0	0
R^2	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
R^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
ψ^2	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
ψ^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	t=5															
Q	$*$	$*$	q_L	q^L_0	q^L_0	q^L_0	q^L_1	q^L_0	q^L_0	$q^{>}_{CC}$	q^R_0	q^R_1	q^R_0	q^R_0	q^R_0	q_R
L^1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
L^2	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
L^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^1	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
R^2	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
R^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	t=6															
Q	$*$	$*$	q_L	q^L_0	q^L_0	q^L_0	q^L_0	q^L_0	q^L_0	$q^{>}_{CC}$	q^R_0	q^R_0	q^R_0	q^R_0	q^R_0	q_R
L^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^3	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0

4.3. Phase 3: finalize.

This is a very straight forward phase. It begins after the central cell send on of the signals $\{(0,0,1),(0,0,2)\}$ during Phase 2. Let's assume that the left side were longer and it was the $(0,0,2)$ signal:

	t=6															
Q	$*$	$*$	q_L	q_0^L	q_0^L	q_0^L	q_0^L	q_0^L	q_0^L	$q_{CC}^{>}$	q_0^R	q_0^R	q_0^R	q_0^R	q_0^R	q_R
L^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^3	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0

This signal will take place in the ether in the next tact:

	t=7															
Q	$*$	$*$	q_L	q_0^L	q_0^L	q_0^L	q_0^L	q_0^L	q_0^L	q_{CF}	q_0^R	q_0^R	q_0^R	q_0^R	q_0^R	q_R
L^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^3	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2
R^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^3	2	2	2	2	2	2	2	2	2	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

When left cells hear this signal they go to the $*$ state. The right target cell goes to the q_{RE} state and the right internal cells — to the q_{RF} state:

	t=8															
Q	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	q_{CF}	q_{RF}	q_{RF}	q_{RF}	q_{RF}	q_{RF}	q_{RE}
L^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5. Complete calculation process for the given example

For clarity we write the whole process for the given example below:

	t=0															
Q	q_S	$*$	q_S	$*$	$*$	$*$	$*$	$*$	$*$	q_{C_0}	$*$	$*$	$*$	$*$	$*$	q_S
L^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
ψ^3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

	t=1															
Q	q_S	$*$	q_S	$*$	$*$	$*$	$*$	$*$	$*$	q_2^C	$*$	$*$	$*$	$*$	$*$	q_S
L^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^2	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
L^3	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
R^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
R^3	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
ψ^3	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

	t=2															
Q	q_L^*	q_*^L	q_L^*	q_*^L	q_*^L	q_*^L	q_*^L	q_*^L	q_*^L	q_1^C	q_*^R	q_*^R	q_*^R	q_*^R	q_*^R	q_R^*
L^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^3	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
R^3	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	0
ψ^2	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	0
ψ^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	t=3															
Q	$*$	$*$	q_L	q_1^L	q_1^L	q_1^L	q_1^L	q_1^L	q_1^L	$q_{CC}^{=}$	q_1^R	q_1^R	q_1^R	q_1^R	q_1^R	q_R
L^1	0	0	0	0	1	0	1	0	1	0	0	1	0	1	0	1
L^2	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
L^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^1	1	1	1	0	1	0	1	0	1	1	0	1	0	1	0	0
R^2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
R^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	0	1	0	1	0	1	0	0	1	0	1	0	0
ψ^2	0	0	0	0	1	0	1	0	1	0	0	1	0	1	0	0
ψ^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	t=4															
Q	$*$	$*$	q_L	q_0^L	q_1^L	q_0^L	q_1^L	q_0^L	q_1^L	$q_{CC}^{<}$	q_0^R	q_1^R	q_0^R	q_1^R	q_0^R	q_R
L^1	0	0	0	0	0	1	1	0	0	1	1	1	0	0	1	1
L^2	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
L^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^1	1	1	1	1	0	0	1	1	0	0	0	1	1	0	0	0
R^2	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
R^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
ψ^2	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
ψ^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	t=5															
Q	$*$	$*$	q_L	q_0^L	q_0^L	q_0^L	q_1^L	q_0^L	q_0^L	$q_{CC}^{>}$	q_0^R	q_1^R	q_0^R	q_0^R	q_0^R	q_R
L^1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
L^2	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
L^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^1	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
R^2	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
R^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	t=6															
Q	$*$	$*$	q_L	q_0^L	q_0^L	q_0^L	q_0^L	q_0^L	q_0^L	$q_{CC}^{>}$	q_0^R	q_0^R	q_0^R	q_0^R	q_0^R	q_R
L^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^3	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0

	t=7															
Q	$*$	$*$	q_L	q_0^L	q_0^L	q_0^L	q_0^L	q_0^L	q_0^L	q_{CF}	q_0^R	q_0^R	q_0^R	q_0^R	q_0^R	q_R
L^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^3	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2
R^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^3	2	2	2	2	2	2	2	2	2	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	t=8															
Q	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	q_{CF}	q_{RF}	q_{RF}	q_{RF}	q_{RF}	q_{RF}	q_{RE}
L^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6. Automaton runtime

Let's calculate the time T need for σ to work out. From the previous section:

$$T = T_1 + T_2 + T_3. \tag{46}$$

where T_k is duration of corresponding phase. It's easy to see that $T_1 = 3$, $T_3 = 2$. It follows from the Lemma 2 that if Phase 2 runtime is t then $t = \min\limits_{2^r > s} r = \lceil \log_2(s + 0.5) \rceil$, where $s = \min(s_L, s_R)$. Because we start time t from 0 in Lemma 1, the overall runtime of Phase 2 will be:

$$T_2 = \lceil \log_2(s + 0.5) \rceil + 1 \le \log_2(s) + 2.$$

Therefore,

$$T \le \log_2(s) + 7.$$

References

[1] Von Neumann J., *Theory of self-reproducing automata*, London, 1966.

[2] Neumann J., von, *Collected works*, New York, 1961 – 1963.

[3] Burks A., *Essays on Cellular Automata*, University of Illinois Press, 1971.

[4] Moore E. F., “Machine models of self-reproduction”, *Proceedings pf Symposia in Applied Mathematics*, **14** (1962), 17–33.

[5] Kudryavtsev V. B., Podkolzin A. S., Bolotov A. A., *Fundamentals of the theory of homogeneous structures*, Nauka, Moscow, 1990 (In Russian).

[6] Kudryavtsev V. B., Gasanov E. E., Podkolzin A. S., *Theory of Intelligent Systems: in 4 books. Book Four. Theory of Automata*, Publishing Solutions, Moscow, 2018 (In Russian).

[7] Titova E. E., “Designing moving images by cellular automata”, *Intelligent systems*, **18**:1 (2014), 153–180 (In Russian).

[8] Kalachev G. V., Titova E. E., “On the measure of the set of laws of motion of a point realized by cellular automata”, *Intelligent systems. Theory and Applications*, **22**:3 (2018), 105–125 (In Russian).

[9] Gasanov E. E., “Cellular automata with locators”, *Intelligent systems. Theory and Applications*, **20**:2 (2020), 121–133 (In Russian).

Remarks on the Definition of Cellular Automaton with Locators [1]

Kalachеv G. V.[2]

In [1], a cellular automaton with locators is defined. In this paper we indicate some inaccuracies and issues of this definition and clarify it to get rid of these issues. We also give examples of cellular automata classes with locators that have good properties in a certain sense.

Keywords: cellular automata, homogeneous structures.

1. Introduction

The concept of a cellular automaton (CA) with locators was introduced in [1]. CA with locators is defined by a 8-tuple $(\mathbb{Z}^n, Q, V, E, +, L, \varphi, \psi)$. CA with locators in comparison with a conventional CA $(\mathbb{Z}^n, Q, V, \varphi)$ contains additional structure where different elementary automata (*cells*) can broadcast signals from the set of *broadcasting signals* E computed by the *broadcasting function* ψ. The signals are summed with the commutative semigroup operation $+$. Locators of each cell receive the sum of signals from the directions specified by the solid angles from the set L. CA with locators can be considered as a mathematical model of a device where there are both local interactions between adjacent cells and non-local interactions through broadcasting, which can be implemented using some kind of substrate that sums the signals from the cells due to some physical principle. Such devices can potentially solve some problems in a more natural way than conventional cellular automata, where sometimes we need to develop complicated algorithms, in particular when we need to transmit control signals.

2. Definition of a cellular automaton with locators according to Gasanov

Let us recall the definition of a cellular automaton with locators introduced by E. E. Gasanov in [1].

By a *solid angle* in $\mathbb{R}^k$ we mean the union of all the rays in the space $\mathbb{R}^k$ emanating from a given point (*vertex of an angle*) and intersecting some

[1] Originally published in *Intellektualnye Sistemy. Teoriya i prilogeniya* (2020) **24**, No. 4, 47-56 (in Russian).

[2] Kalachev Gleb Vyacheslavovich — Candidate of Physical and Mathematical Sciences, Junior Researcher, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Problems of Theorecical Cybernetics Lab.

hypersurface in $\mathbb{R}^k$. In the definition, we assume that a solid angle does not contain its vertex. In particular, in this paper we consider two degenerate cases: the full solid angle coinciding with $\mathbb{R}^k$ without the vertex of the angle, which we denote by Ω, and solid angles equal to one ray. If a solid angle is a ray, we denote it by a vector defining its direction.

A *cellular automaton with locators* is a 8-tuple

$$\sigma = (\mathbb{Z}^k, E_n, V, E_q, +, L, \varphi, \psi)$$

where $\mathbb{Z}^k$ is the set of k-dimensional vectors with integer coordinates, $E_n = \{0, 1, \ldots, n-1\}$, $V = (\alpha_1, \ldots, \alpha_{h-1})$ is an ordered set of pairwise different nonzero vectors from $\mathbb{Z}^k$, $E_q = \{0, 1, \ldots, q-1\}$, $+$ is a commutative semigroup operation defined on E_q, $L = (\nu_1, \ldots, \nu_m)$ is an ordered set of pairwise different solid angles in $\mathbb{R}^k$ with a vertex at the origin, $\varphi : E_n^h \times E_q^m \to E_n$ is a function depending on the variables $x_0, x_1, \ldots, x_{h-1}, z_1, \ldots, z_m$ such that $\varphi(0, \ldots, 0) = 0$, $\psi : E_n^h \times E_q^m \to E_q$ is a function that depends on the variables $x_0, x_1, \ldots, x_{h-1}, z_1, \ldots, z_m$. Here the variables $x_0, x_1, \ldots, x_{h-1}$ take values from E_n and the variables $z_1, \ldots, z_m$ take values from E_q. Elements of the set $\mathbb{Z}^k$ are called *cells* of the cellular automaton σ; elements of the set E_n are called *cell states* of the cellular automaton σ; the set V is called the *neighborhood pattern* of the cellular automaton σ; elements of the set E_q are called *broadcasting signals*; the set L is called the *locator pattern* of the cellular automaton σ; the function φ is called the *local transition function* of the automaton σ; the function ψ is called the *broadcasting function* of the automaton σ. The state 0 is interpreted as *rest state* and the condition $\varphi(0, \ldots, 0) = 0$ is interpreted as a condition for maintaining the rest state.

Here we need to introduce an ordering of the neighborhood pattern V and the locator pattern L in order to establish a one-to-one correspondence between vectors from V and solid angles from L and variables x_0, $x_1, \ldots, x_{h-1}$, $z_1, \ldots, z_m$ of the local transition function φ and the broadcasting function ψ respectively. We can make this correspondence more explicit if we index the variables of the functions φ and ψ by the vectors and solid angles themselves, i.e. assume that the local transition function φ and the broadcasting function ψ depend on the variables $x_{\mathbf{0}}$, $x_{\alpha_1}, \ldots, x_{\alpha_{h-1}}$, $z_{\nu_1}, \ldots, z_{\nu_m}$, where the index of the first variable is the zero vector $\mathbf{0} = (0, \ldots, 0) \in \mathbb{Z}^k$. If we index the variables of the local transition function and broadcasting function in this way, we can write them in any order, and then we can define the neighborhood pattern and the locator pattern simply as a set, not an ordered set.

In the rest of this section we use these conventions: consider the neighborhood pattern as a set of vectors, and the locator pattern as a set of solid angles and index the variables of the local transition function and broadcasting function by the vectors from the neighborhood pattern and solid angles from the locator pattern. At the same time, we often omit the outer parentheses of the vectors in the indices. For example, if $k = 2$, $n = 2$, $q = 2$, $V = \{(-1, 0), (1, 0)\}$, and $L = \{\Omega, (0, 1)\}$, then a local transition function may look like this: $\varphi = x_{-1,0} \& z_\Omega \vee x_{1,0} \& z_{0,1}$.

If $\alpha \in \mathbb{Z}^k$, ν is a solid angle with vertex at the origin, then by $\nu(\alpha)$ we denote the solid angle obtained by translation of the angle ν to the point α.

If $\alpha \in \mathbb{Z}^k$ is a cell of a cellular automaton with locators σ, then the set $V(\alpha) = \{\alpha, \alpha + \alpha_1, \ldots, \alpha + \alpha_{h-1}\}$ is called the *neighborhood of the cell* α, and elements of the set $L(\alpha) = \{\nu_1(\alpha), \ldots, \nu_m(\alpha_m)\}$ are called *locators of the cell* α.

A *state of a cellular automaton with locators* σ is a pair (e, f), where e is an arbitrary function from the set $\mathbb{Z}^k$ to the set E_q, called *broadcast state*, f is an arbitrary function from the set $\mathbb{Z}^k$ to the set E_n and called *distribution of states of the cellular automaton with locators* σ. Such a function can be interpreted as a certain mosaic arising in the k-dimensional space as a result of assigning a certain state from the set E_n and some signal from the set E_q to each point with integer coordinates. The set of all possible states of a cellular automaton with locators is denoted by Σ.

If $\alpha \in \mathbb{Z}^k$ and (e, f) is a state of a cellular automaton with locators σ, then the value $e(\alpha)$ is called *the signal of the cell* α*, defined by the state* (e, f), and the value $f(\alpha)$ is *the state of the cell* α*, determined by the state* (e, f). For each $i \in \{1, \ldots, m\}$ the value

$$s_i(\alpha) = \sum_{\beta \in \nu_i(\alpha) \cap \mathbb{Z}^k} e(\beta) \tag{1}$$

we call *the value of the locator* ν_i*, determined by the state* (e, f). Here, in the summation the semigroup operation $+$ defined on E_q is used.

On the set Σ we define the *global transition function* Φ of a cellular automaton with locators σ, putting $\Phi(e, f) = (e', f')$, where $(e, f), (e', f') \in \Sigma$ and for any cell $\alpha \in \mathbb{Z}^k$ the following identities hold

$$f'(\alpha) = \varphi(f(\alpha), f(\alpha + \alpha_1), \ldots, f(\alpha + \alpha_{h-1}), s_1(\alpha), \ldots, s_m(\alpha)), \tag{2}$$

$$e'(\alpha) = \psi(f(\alpha), f(\alpha + \alpha_1), \ldots, f(\alpha + \alpha_{h-1}), s_1(\alpha), \ldots, s_m(\alpha)). \tag{3}$$

A meaningful interpretation of the mapping Φ is that the signal of each cell and the state of each cell “after the transition” is determined by the state of the neighborhood of the cell and by the values of the locators “before the transition” using the rules ψ and φ in the same way for all cells.

By the *behavior of a cellular automaton with locators* σ we call a sequence $(e_0, f_0), (e_1, f_1), (e_2, f_2), \ldots$ of states such that the equation $(e_{i+1}, f_{i+1}) = \Phi(e_i, f_i)$ holds for all $i = 0, 1, 2, \ldots$. The state (e_i, f_i) is called the *state of* *the cellular automaton with locators σ at the time i*, and (e_0, f_0) is also called the *initial state of the cellular automaton with locators σ.*

A state of a cellular automaton is called a *configuration* if only a finite number of cells is in a state other than 0 and the signals of all cells are zero. The set of configurations is denoted by Σ'.

If a certain state of a cellular automaton is specified, then cells that are in a state other than 0 are called *active.*

3. Corrections for the definition

3.1. Restriction on solid angles

According to the definition in Section 2, a solid angle is a union of rays intersecting some hypersurface. However, even in the two-dimensional case, an angle is defined by a real number which can be used to encode an infinite amount of information. For the two-dimensional case, we propose to restrict the set of solid angles to the set of angles bounded by rays going through points with rational coefficients.

For the multidimensional case, there is even more freedom of choice of a solid angle. In this case, we propose to introduce the following restriction: the boundary of a solid angle should consist of hyperplanes spanned by points with integer coordinates. Note that degenerate solid angles completely contained in a subspace of a lower dimension are also allowed. Boundary of such a degenerate angle should consist of parts of hyperplanes specified by linear equations with integer coefficients.

3.2. Restrictions on the semigroup and the broadcasting function

The definition of cellular automaton requires the existence of a distinguished zero state which is preserved by the transition function. It is natural to add a similar requirement for the broadcasting alphabet. Formally, in [1] the set E always has a form $\{0, ..., q-1\}$ and contains 0, however, there is no requirement that $0 + x = 0$. We propose not to require that E has the form $\{0, ..., q-1\}$ but could contain elements of arbitrary type (apart from numbers, it is often convenient to use pairs or sets of numbers), but require that the semigroup $(E, +)$ is a monoid, i.e. there exists a neutral element $0 \in E$ such that $0 + x = x$ for all $x \in E$.

In [1] there is a restriction on the transition function $\varphi(\mathbf{0}, \mathbf{0}) = 0$. It is natural to add a similar restriction on the broadcasting function:

$$\psi(\mathbf{0}, \nu) = 0,$$

i.e. an inactive cell that doesn't have active neighbors cannot broadcast nonzero signals.

3.3. Partial definiteness of the global transition function

In equation (1) the value of locator $s_i(\alpha)$ is defined as a sum of the infinite number of terms by the integer points of the solid angle where a semigroup operation is used as an addition. An infinite sum is understood here in the usual sense (as the limit of partial sums) with the clarification that a discrete topology is introduced on the set E. In this case, for the series to converge, it is necessary that starting from some moment, the partial sums are equal to a constant, which is the sum of the series. This sum can be undefined if the sum involves an infinite number of nonzero terms. In the general case, the value of the locator is a partially defined function. Hence the global transition function of the CA with locators is also partially defined. However, even here a proof of correctness is required, namely, we need to prove that the convergence of the series (1) and the value of the sum does not depend on the order of terms (in the case of numerical series, this is true only for absolutely convergent series).

Proposition 1. *Let $(E, +)$ be a commutative semigroup with discrete topology. Let $\{x_j\}_{j=1}^{\infty}$ be a sequence of elements E, $\{y_j\}_{j=1}^{\infty}$ be its permutation ($y_j = x_{i_j}$). Then if one of the series $\sum_{j=1}^{\infty} x_j$ and $\sum_{j=1}^{\infty} y_j$ converges, then the second also converges and their sums coincide.*

Доказательство. The proof goes by way of contradiction. Without loss of generality, assume that $\sum_{j=1}^{\infty} y_j = a$, and the series $\sum_{j=1}^{\infty} x_j$ either diverges or its sum is not equal to a. This means that in the sequence of partial sums $(X_n)_{n=1}^{\infty}$, $X_n = \sum_{j=1}^{n} x_j$ there is an infinite number of terms not equal to a. Since the first series converges, there exists N_0 such that for all $n \geq N_0$ the partial sum $Y_n = \sum_{j=1}^{n} y_j$ is equal to a. This means that $a + y_j = a$ for all $j > N_0$.

We denote $K_n = \{j \mid i_j \leq n\}$. Take $N \geq \max_{j \leq N_0} i_j$ such that $X_N = b \neq a$. By construction $1, 2, ..., N_0 \in K_N$. Hence

$$b = X_N = \sum_{j=1}^{N} x_j = \sum_{k \in K_N} y_k = \sum_{k=1}^{N_0} y_j + \sum_{j > N_0, j \in K_N} y_j = a + \sum_{j > N_0, j \in K_n} y_j = a.$$

However $b \neq a$ by our assumption, and we have a contradiction. Hence $\sum_{j=1}^{\infty} x_n = a$, as required. □

A state of a CA with locators is called *finite* if there is only a finite number of active cells. Note that, taking into account the previous corrections, for finite states the global function is defined since only active cells can broadcast nonzero signals. However, consider such a CA with locators:

$$\sigma = (\mathbb{Z}, \{0,1\}, \varnothing, \{0,1\}, \max, \{\Omega\}, \max, \max),$$

where Ω corresponds to the locator that receives signals from all directions. Suppose at the first moment there is exactly one cell is in state 1, and thus the state is finite. Then this call broadcasts signal 1 and all the cells at the second moment receive signal $\max(0,1) = 1$, hence they go to state 1 at the third moment. Therefore, the state at the third moment is not finite. If we take $\oplus$ instead of max as a semigroup operation, then the functioning at the first two moments will be the same, and at the third moment, the transition function will not be defined.

4. Interesting classes of CA with locators

4.1. Classes solving the problem of partial definiteness of the transition function

Taking into account the example from the Section 3.3, it is important to find classes of CA with locators $(\mathbb{Z}, Q, V, E, +, L, \varphi, \psi)$, where the definiteness of the global transition function is guaranteed at any moment of time for some class of initial conditions.

4.1.1. Idempotent monoid

Consider the case when the monoid $(E, +)$ is idempotent (is a semilattice), that is, $x + x = x$ for all $x \in E$. In this case, the sum of an infinite number of terms depends only on the set of terms present in the sum, and thus reduces to a finite sum. Therefore, we have the following statement.

Proposition 2. *If the monoid $(E, +)$ is idempotent, then the global transition function of a CA with locators $\sigma = (\mathbb{Z}, Q, V, E, +, L, \varphi, \psi)$ is defined everywhere.*

For example, if E is a linearly ordered set, then $(E, \max)$ is an idempotent monoid with neutral element $\min E$.

4.1.2. Finite CA with locators

We will say that the CA with locators σ is *finite* if for any finite state S of σ the next state $\Phi(S)$ is also finite.

Proposition 3 (Sufficient condition for the finiteness of a CA with locators)**.** *Let* $\sigma = (\mathbb{Z}^n, Q, V, E, +, \{\nu_1, ..., \nu_m\}, \varphi, \psi)$ *be a CA with locators satisfying the following condition:*

$$\textit{if } \varphi(\vec{\mathbf{0}}, (e_1, ..., e_m)) \neq 0, \ \textit{then} \bigcap_{i: e_i \neq 0} \nu_i = \varnothing.$$

Then σ *is finite.*

In this statement it is important that we exclude vertex from the solid angle, otherwise, the intersection of the solid angles ν_i would always contain the origin.

Доказательство. Consider an arbitrary finite state s. Let A be a set of all cells that are either active itself or have active neighbors, r be the maximum Euclidean distance between elements of A.

Suppose the state $\Phi(s)$ is not finite. In this case, there is an infinite set M of cells that was not active and didn't have active neighbors in the finite state s and that became active in the state $\Phi(s)$. For each cell x from M consider the set of its active locators $a(x)$ and choose such a set of locators $L' \subset L$ that occurs infinitely many times among $a(x)$ for $x \in M$. Let $M' = \{x \in M : a(x) = L'\}$.

Without loss of generality we assume that $L' = \nu_1, ..., \nu_k$. From the statement condition we have $\bigcup_{j=1}^{k} \nu_j = \varnothing$.

Let S be the unit sphere in $\mathbb{R}^n$, $P = \prod_{j=1}^{k} (\nu_{i_j} \cap S)$. Let us show that

$$\hat{d} := \inf_{p \in P} \max_{1 \leq j, j' \leq k} \|p_j - p_{j'}\| > 0, \tag{4}$$

where $\|\cdot\|$ is the Euclidean norm.

Note that each set $\nu_{i_j} \cap S$ is compact, therefore, their product P is also a compact set. Hence continuous function $d(p) := \max_{j \neq j'} \|p_j - p_{j'}\|$ reaches its minimum on the compact set P. Suppose, this minimum is 0. Then there exist $p \in P$, $p_j \in \nu_j$ such that $p_j = p_{j'}$ for all $1 \leq j, j' \leq k$, that is, $p_1 = ... = p_k \in \bigcap_{j=1}^{k} \nu_j = \varnothing$, and we obtain a contradiction. Hence (4) is satisfied.

Since the set M' is infinite, there exists an element $x \in M'$ located at a distance $D > r/d$ from the set A. Since the cell x has the locators $\nu_1, ..., \nu_k$ active, there exist elements $y_1, ..., y_k \in A$ such that $v_j = y_j - x \in \nu_j$. Put

$p_j = \frac{v_j}{\|v_j\|}$. Then for any $1 \leq i, j \leq k$ the following holds:

$$\|p_i - p_j\|^2 = \|p_i\|^2 + \|p_j\|^2 - 2(p_i, p_j) = 2 - 2\frac{(v_i, v_j)}{\|v_i\|\|v_j\|} \leq$$
$$\leq \frac{\|v_i\|}{\|v_j\|} + \frac{\|v_j\|}{\|v_i\|} - 2\frac{(v_i, v_j)}{\|v_i\|\|v_j\|} = \frac{\|v_i - v_j\|^2}{\|v_i\|\|v_j\|} \leq$$
$$\leq \frac{\|v_i - v_j\|^2}{D^2} = \frac{\|y_i - y_j\|^2}{D^2} \leq \frac{r^2}{D^2} < d^2.$$

Thus, $\max_{1 \leq i,j \leq k} \|p_i - p_j\| < d$. On the other hand, $p_j \in \nu_j \cap S$, that is, $p = (p_1, ..., p_k) \in P$ which contradicts (4). Hence, our assumption is wrong, and the state $\Phi(s)$ is finite which completes the proof. □

4.2. Class with simple physical implementation

It is most natural to imagine the implementation of a CA with locators as a chip. The broadcasting should be implemented by some device that "sums up" an unlimited number of electrical signals. Such a device can consist of the following elements: a conductor connected to all cell outputs to be summed; an amplifier with the input connected to the conductor and output connected to the locator inputs of all the cells. Thus, if one of the cells emits a signal, this signal will be amplified and signal 1 will come to the locators of all cells. If all the cells emit 0, then 0 will come to the locators of all cells as well. In such a way we can implement the operation max from an unlimited number of arguments taking values from the set $\{0, 1\}$.

However, for a CA with locators, it is required to be able to calculate $\max_{i \neq j} a_j$ for all $i = 1, ..., m$. Note that

$$\max_{j \neq i} a_j = \min\Big(\sum_{j \neq i} a_j, 1\Big) = \min\Big(\min\Big(\sum_{j=1}^{m} a_j, 2\Big) - a_i, 1\Big).$$

it is also possible to implement operation $M_2(a_1, ..., a_n) = \min(\sum_{j=1}^{m}, 2)$, but more difficult than the operation max. For example, this can be done as follows. Each input representing an operation argument equal to 1 outputs a limited current to the wire connecting all the arguments and connected to the neutral wire through a resistor. Depending on the number of inputs equal to 1, there would be different voltages on the connecting conductor. The conductor itself can be connected to two comparators, of which one is triggered at voltage when at least one input is active, and the other is triggered when at voltage when at least 2 inputs are active. Using the results of these comparators, it is easy to obtain the value of the function M_2. Then, through the common wire, we can connect the result $s = M_2(a_1, ..., a_m)$ back to all cells, and calculate $\min(s - a_i, 1)$ in the i-th cell. Thus, the result in i-th cell is $\max_{j\neq i} a_j$ as required.

Using n copies of such a circuit, we can implement the Max operation on the set $\{0,1\}^n$, which is a component-wise max operation:

$$\mathrm{Max}((a_1^1, ..., a_n^1), \cdots, (a_1^m, ..., a_n^m)) = (\max(a_1^1, ..., a_1^m), \cdots, \max(a_n^1, ..., a_n^m)).$$

Let show that arbitrary idempotent commutative monoid $(E, +)$ where $|E| = n < \infty$ can be implemented using operation Max and ordinary logic gates. To do this, we encode nonzero elements of E by the tuples $(1, 0, ..., 0)$, $(0, 1, 0, ..., 0)$, ..., $(0, ..., 0, 1) \in \{0,1\}^{n-1}$, and we encode $0 \in E$ by the all-zero tuple. Let v be the described encoding function. For the set $E' = \{e_1, ..., e_m\} \subseteq E$, we define $\hat{v}(E') = \mathrm{Max}_{e\in E'} v(e)$. In the tuple $\hat{v}(E')$ ones occur at positions corresponding to nonzero elements of the set E'. Boolean operator $F : \hat{v}(E') \mapsto v(\sum_{e\in E'} e)$ can be implemented by a logic circuit. Using the idempotency of the monoid, for an arbitrary number of arguments we have

$$v\Big(\sum_{i\in I} e_i\Big) = v\Big(\sum_{e\in\{e_i | i\in I\}} e\Big) = F(\hat{v}(\{e_i \mid i \in I\})) = F(\underset{i\in I}{\mathrm{Max}}\, v(e_i)).$$

So, we proved that for any finite idempotent monoid it is possible to implement its semigroup operation from an unlimited number of elements using a fixed logic circuit and several conductors connected to all cells whose outputs are summed up.

This is exactly the class of monoids from Section 4.1.1, for which the global transition function is defined everywhere. The situation with the implementation of locators is worse. The conductor conducts in the same way in all directions. If we use diodes that pass current only in one direction, the depth of the circuit will immediately become linear in the number of arguments, and in this case, it is no longer possible to say that the broadcast is instant, thus the goal of using this model is lost. Therefore, only solid angles coinciding with subspaces can be implemented by the described method. For example, Ω is implemented if the outputs of all cells are connected with a plate. We can make a layer with many wires going in the same direction. Thus we can implement the locator $\{v, -v\}$, where v is the direction of the wires in this layer.

Implementation of other locators requires the use of some other physical principles that go beyond conventional circuit design.

References

[1] E.E. Gasanov, “Celluar automata with locators”, *Intelligent systems. Theory and applications*, **24**:2 (2020), 121–133.

The complexity of multilayer d-dimensional circuits [1]

T. R. Sitdikov[2], G. V. Kalachev[3]

In this paper we research a model of multilayer circuits with a single logical layer. We consider λ-separable graphs as a support for circuits. We establish the Shannon function lower bound $\max\left(\frac{2^n}{n}, \frac{2^n(1-\lambda)}{\log k}\right)$ for this type of circuits where k is the number of layers. For d-dimensional graphs, which are λ-separable for $\lambda = \frac{d-1}{d}$, this gives the Shannon function lower bound $\frac{2^n}{\min(n, d\log k)}$. For multidimensional rectangular circuits the proved lower bound asymptotically matches to the upper bound.

Keywords: multilayer circuits, multidimensional circuits, Shannon function asymptotics, circuit complexity, graph separators.

1. Introduction

The problem of designing circuits which compute Boolean functions and are optimal or suboptimal in some sense appeared in the middle of the 20th century due to the rapid development of computer technology. One of the most intensive studied circuit models since the 1950s is Boolean circuits. The number of gates (also *size* or *complexity*) is a natural complexity measure for Boolean circuits. One may define the complexity of a Boolean function as the minimal size of a Boolean circuit computing the function. Muller [22] showed that the size of every Boolean function of n variables does not exceed $O\left(\frac{2^n}{n}\right)$. Lupanov [19] proved that the complexity of almost all Boolean functions over the standard basis $\{\vee, \&, \neg\}$ is asymptotically equal to $\frac{2^n}{n}$. Also Lupanov obtained asymptotic bound for the complexity of Boolean functions with respect to any finite basis.

In practice when designing Boolean circuits one must take into account several factors including placement of gates, wiring and others. Models of Boolean circuits considering these factors to some extent were studied in some papers appeared since the 1960s. Korshunov [15] obtained size bounds for Boolean circuits embedded in a 3-dimensional space with lower-bounded

[1] Originally published in *Intellektualnye Sistemy. Teoriya i prilogeniya* (2021) **25**, No. 2, 131-154 (in Russian).

[2] *Sitdikov Timur Rashidovich* — Software Engineer, Google LLC, e-mail: s7t1r9@gmail.com.

[3] *Kalachev Gleb Vyacheslavovich* — Candidate of Physical and Mathematical Sciences, Junior Researcher, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Problems of Theorecical Cybernetics Lab, e-mail: gleb.kalachev@yandex.ru.

distances between gates, lower-bounded distances between wires and upper-bounded lengths of wires. Kravtsov [17] considered Boolean circuits with gates placed in cells of a rectangular grid and proved the order of 2^n for the Shannon function. McColl [20] obtained Shannon function lower bound $\Omega(2^n)$ for planar circuits.

Models of cellular circuits similar to Kravtsov's model were considered in several more recent papers. Albrecht [1] showed that Shannon function asymptotics for cellular circuits has a form $c \cdot 2^n$, where c is a constant dependent on a basis. Gribok [8] obtained Shannon function asymptotics to 2^n for a special basis of cellular elements. The connection between size and other complexity measures for cellular circuits also has been examined. Cheremisin [4] showed that it is impossible to design a cellular circuit of optimal size and activity simultaneously for a binary decoder. Kalachev [9, 10, 11, 12, 13] researched simultaneous minimization of a size, depth and activity for cellular circuits. Efimov [5, 6] examined potential of three-dimensional cellular circuits.

VLSI circuits are one of the closest to practice circuit models. In VLSI circuits length of wires define the signal propagation time between gates. VLSI circuits have been studied in a number of papers and books (Thompson [27], Ullman [28]). Kramer and van Leeuwen [16] researched simultaneous minimization of size (area) and period.

Another direction of research is studying connections between complexity measures for different circuit models. Savage [23, 24] examined the connection between VLSI circuits and planar circuits. Shkalikova [25] showed a relation between the area of flat circuits and the volume of three-dimensional circuits.

The bounds of size proved for the mentioned above circuit models are above Lupanov's bound $\frac{2^n}{n}$ for Boolean circuits size. One of the reasons for this difference is that it's impossible to conduct arbitrary number of wires between gates under spatial constraints. If Boolean circuits are embedded into a graph (e.g. rectangular grid), the number of wires that can be conducted between fragments of the graph is naturally bounded by the size of edge separator in the graph.

In this paper we examine the relation between Shannon function and separability properties of the graph where Boolean circuits are embedded. We consider embeddings with constraints from [26]:

- No more than 1 non-identity gate of a Boolean circuit can be embedded into any vertex of a graph.

- No more than k wires of a Boolean circuit can be embedded into any edge of a graph.

The main result of this paper is the lower bound of Shannon function for graphs with a separator of size $O(p^\lambda)$, where p is the order of a graph and

$0 < \lambda < 1$. We call such graphs as λ-separable. We also show that the proved lower bound is applicable to circuits embedded into a space with two or more dimensions. Given the Shannon function upper bound for multidimensional rectangular circuits [26], we obtain the Shannon function asymptotics for this model of circuits.

2. Key definitions and results

2.1. Multilayer circuits

The model of multilayer circuits with a single logical layer was introduced in [26]. Let us briefly summarize key definitions.

According to [3, p. 148], a Boolean circuit in a basis B is a labeled directed acyclic graph. The labeling of vertices defines which vertices are inputs or outputs. It also maps all non-input vertices to Boolean functions from the basis B. Edges (wires) of a Boolean circuit are labeled by integers, and for each vertex the labeling of its input edges defines the order of arguments for the Boolean function mapped to the vertex.

A *support* is a nonempty graph with a finite or countable number of vertices. In general a support may contain multiple edges or self-loops.

An *embedding* of a Boolean circuit S into a support T is a homomorphism $h\colon S \to T$.

A *circuit with a support* T is a pair (S, h) where S is a Boolean circuit and h is an embedding of S into T. We use the term “circuit” instead of “circuit with a support” for brevity where it would not cause a misreading. A circuit (S, h) computes a Boolean function f if a Boolean circuit S computes f.

In practical terms these definitions may be interpreted as follows. One of the problems in VLSI design is an embedding of gates and wires into a plate. The plate may be considered as a graph, i. e. as a support for Boolean circuits.

Usually there are various constraints on embeddings in circuit design problems. In this paper we consider the following constraints.

Constraint 1. Any vertex of a support may contain no more than 1 gate computing non-identity Boolean function.

Constraint 2. Any edge of a support may contain no more than k wires of a Boolean circuit.

These constraints may be interpreted as follows. A circuit consists of k “layers” where only one layer is “logical” (i. e. may contain gates computing non-identity Boolean functions). The remaining layers are used only for wiring. Therefore we call the circuits under constraints 1–2 as *multilayer circuits*.

Let us denote by M_k^T the set of all k-layer circuits with a support T.

The *complexity* of a multilayer circuit is the number of support vertices used in the corresponding embedding. If M is a set of multilayer circuits and f is a Boolean function, one may naturally define the complexity of the function f in the set M as the minimal complexity of a circuit from M computing f. If no such a circuit exists in M, we may formally consider infinity as the complexity of f. Let us denote the complexity of the function f in the set M as $L(M, f)$.

One may naturally define the Shannon function of the complexity of k-layer circuits with the support T:

$$L(M_k^T, n) = \max_{f \in B_n} L(M_k^T, f).$$

2.2. Supports

The properties of a support are crucial for embeddings, as under the same constraints in general different supports admit different sets of embeddings. In this paper we consider λ-separable graphs as supports. We also consider d-dimensional graphs as an important special case of λ-separable graphs.

A class $\mathcal{G}$ of graphs is *monotone* if every subgraph of a graph in $\mathcal{G}$ also belongs to $\mathcal{G}$.

2.2.1. Classes of graphs $\mathcal{G}(q, \theta)$ and $\mathcal{G}(\lambda, q, \theta)$

Let $q \in \mathbb{N}$ and $\theta > 1$ be some constants. Let us define class of graphs $\mathcal{G}(q, \theta)$ as the set of all supports with the following properties:

- Degree of each vertex in T is bounded by q.
- For any integer p the number of different non-isomorphic subgraphs of T with p vertices does not exceed θ^p.

The first property (bounding for vertex degree) is a natural limitation for circuit design problems. The second property is met for several important categories of graphs, including planar graphs [2] and d-dimensional graphs defined below.

The formal definition of λ-separability is considered in the section 3. Substantively each subgraph of a λ-separable support can be split into smaller fragments by removing $O(p^\lambda)$ vertices (edges), where p is the number of vertices in the subgraph and $0 < \lambda < 1$.

We denote the subclass of λ-separable supports from $\mathcal{G}(q, \theta)$ as $\mathcal{G}(\lambda, q, \theta)$.

2.2.2. d-dimensional graphs

Let $d \geq 2$ be an integer. A support T is a *d-dimensional graph*, if there are constants $c_v > 0, c_e > 0$ such that T can be embedded into d-dimensional Euclidean space with pairwise distances between vertices no less than c_v and edge lengths no greater than c_e.

Remark 1. The constraints in the definition of d-dimensional graphs are similar to the constraints in the definition of circuits with volumetric gates from Korshunov's paper [15].

Remark 2. We can always assume that one of the constants c_v and c_e is equal to 1. Below we assume that $c_v = 1$.

Remark 3. It's clear that every finite support is a d-dimensional graph with a great enough value of c_e. Therefore the definition of d-dimensional graphs is senseless for finite supports. However one may define a *class of d-dimensional graphs with a parameter c_e*, where the constant c_e is common for all graphs in the class. It's obvious that such a class is a monotone class of graphs. Below we omit the constant c_e and speak about a *d-dimensional class of graphs* in cases when the value of c_e is not important.

Example 1. The graph of a d-dimensional grid is a d-dimensional graph. It's sufficient to consider $c_e = 1$.

Example 2. One can prove that the graph of an infinite binary tree is not d-dimensional for any d. Indeed, the number of vertices at distance p from the root depends on p exponentially, though the number of d-dimensional balls with radii 1 that can be placed into a ball with radii $c_e \cdot d$ is $O(p^d)$.

Embedding of Boolean circuits into d-dimensional grid was considered in [26]. As in that paper, we use the term *multidimensional rectangular circuits* for such circuits and use the notation M_k^d instead of $M_k^{\mathbb{Z}^d}$.

In section 5 we prove that all d-dimensional supports belong to classes $\mathcal{G}(\lambda, q, \theta)$ for $\lambda = \frac{d-1}{d}$ and some values of q and θ.

2.3. Other designations and agreements

The expression $\log a$ always denotes a base two logarithm of a. We formally assume that $x \log x = 0$ for $x = 0$.

We denote by $B_{n,m}$ the set of Boolean functions with n inputs and m outputs ($n \geq 0$, $m \geq 1$).

The expression $f(x) \lesssim g(x)$ corresponds to the inequality $\overline{\lim}_{x\to\infty} \frac{f(x)}{g(x)} \leq 1$. Similarly we use the expression $f(x) \gtrsim g(x)$. We may use a complex condition when passing to a limit, e. g. $f(n,k) \lesssim g(n,k)$ as $k \to \infty$, $\log k \leq n$.

2.4. Results

In this paper for every support $T \in \mathcal{G}(\lambda, q, \theta)$ we prove that

$$L(M_k^T, n) \gtrsim \max\left(\frac{2^n}{n}, \frac{2^n(1-\lambda)}{\log k}\right) \quad \text{as} \quad k \to \infty,\ n \to \infty.$$

It is also proved that every d-dimensional support belongs to the class $\mathcal{G}(\lambda, q, \theta)$ for $\lambda = \frac{d-1}{d}$ and some constants q and θ. Therefore the following lower bound holds for for d-dimensional supports:

$$L(M_k^T, n) \gtrsim \frac{2^n}{\min(n, d\log k)} \quad \text{as} \quad k \to \infty,\ n \to \infty.$$

An upper bound of Shannon function for multidimensional rectangular circuits matching the lower bound above was proved in [26]. Thus we have the asymptotics of Shannon function for multidimensional rectangular circuits:

$$L(M_k^d, n) \sim \frac{2^n}{\min(n, d\log k)} \quad \text{as} \quad k \to \infty,\ n \to \infty.$$

2.5. The structure of the paper

In this paper all the proofs are divided into three sections.

Section 3 contains definitions related to graph separators. The main result of the section is lemma 2. The point of the lemma is that λ-separable graphs supporting “good” (in some sense) partitioning into two parts also support “good” partitioning into many parts.

Section 4 contains the proof of the lower bound for Shannon function of the complexity for supports from classes $\mathcal{G}(\lambda, q, \theta)$. The key part of this section is lemma 7.

Section 5 is devoted to the proof of the lower bound for Shannon function for d-dimensional supports. The section also contains the asymptotics of Shannon function for multidimensional rectangular circuits as a corollary. Essentially it’s proved that every d-dimensional support belongs to a class $\mathcal{G}(\lambda, q, \theta)$ for $\lambda = \frac{d-1}{d}$ and some constants q and θ.

3. Graph separators and their properties

3.1. Definitions and the simplest properties of separators

In this section we provide the definitions of edge and vertex separators in graphs and prove some of the simplest properties of separators.

Edge separators. We define edge separators similarly to the definitions of vertex separators from [18].

Definition 1. Let $f\colon \mathbb{N} \to \mathbb{R}$ be a function. A monotone class of graphs $\mathcal{G}$ is *edge $f(p)$-separable* if there exist constants $\frac{1}{2} \le \alpha < 1$, $\beta \ge 0$, $m \ge 2$ such that any graph $G \in \mathcal{G}$ with p vertices ($p \ge m$) can be split into two subgraphs with no more than αp vertices each and no more than $\beta f(p)$ edges between the subgraphs.

Remark 4. The constant m is technically important, since it allows not to consider some corner cases. For example graph K_1 cannot be split into two nonempty subgraphs in principle, thus we may always assume that $m \ge 2$. In general m may be greater than 2.

Definition 2. Let $f\colon \mathbb{N} \to \mathbb{R}$ be a function. A support T is *edge $f(p)$-separable* if the monotone class of all finite subgraphs of T is edge $f(p)$-separable.

The interesting case is when $f(p)$ is a slowly growing function. Essentially this allows to use the divide-and-conquer technique for obtaining effective algorithms and non-trivial lower bounds in proofs. In this paper we consider the function p^λ with $0 < \lambda < 1$ as $f(p)$. We also call edge p^λ-separable supports and monotone classes of graphs as *edge λ-separable.*

Vertex separators. The following definition of a vertex separator is a modification of definition 2.1 from [21] applied to monotone classes of graphs.

Definition 3. Let $f\colon \mathbb{N} \to \mathbb{R}$ be a function. A monotone class of graphs $\mathcal{G}$ is *vertex $f(p)$-separable* if there exist constants $\frac{1}{2} \le \alpha < 1$, $\beta \ge 0$, $m \ge 2$ such that for any graph $G \in \mathcal{G}$ with p vertices ($p \ge m$) there exists a partition of $V(G)$ into three parts A, B, C satisfying the following conditions:

- There are no edges from A to B.
- $|A|, |B| \le \alpha p$.
- $|C| \le \beta f(p)$.

It's obvious that for any monotone class of graphs edge $f(p)$-separability implies vertex $f(p)$-separability, as one may consider endpoints of an edge separator as a vertex separator. The converse is not always true. For example, the class of stars $K_{1,p}$ and their subgraphs is vertex 1-separable, but is not edge 1-separable.

The following simple lemma shows that vertex $f(p)$-separability implies edge $f(p)$-separability for monotone classes of graphs with bounded vertex degree.

Lemma 1. *Let $\mathcal{G}$ be a monotone class of vertex $f(p)$-separable graphs with the parameters α, β, m where vertex degree of any graph is bounded by q. Then $\mathcal{G}$ is edge $f(p)$-separable with parameters* $\max\left(\frac{2}{3}, \alpha\right)$, $q\beta$ *and* $\max(m, 2)$.

Proof. We'll show how to obtain an edge separator from a vertex separator.

Let $G \in \mathcal{G}$, where $|V(G)| = p \geq \max(m, 2)$. By the definition of vertex separability $V(G)$ can be divided into three sets A, B, C, where C is a separator. Here $|A|, |B| \leq \alpha p$, $|C| \leq \beta f(p)$.

Let us move vertices from C to A and B in a way to keep the sizes of the resulting sets as close to each other as possible. We denote the resulting sets by A' and B'. Considering the way of constructing A' and B', we obtain $1 \leq |A'|, |B'| \leq \max\left(\frac{2}{3}, \alpha\right) \cdot p$.

Each edge connecting A' and B' is incident to at least one vertex from C. Since vertex degree is bounded by q, the total number of such edges does not exceed $q|C| \leq q\beta f(p)$.

Since values $\max\left(\frac{2}{3}, \alpha\right)$, $q\beta$ and $\max(m, 2)$ do not depend on a graph, edge $f(p)$-separability of $\mathcal{G}$ is proved. □

Thereby when we define a class $\mathcal{G}(\lambda, q, \theta)$ it does not matter whether we use edge λ-separability or vertex λ-separability, as all graphs from the class have vertex degree bounded by q.

3.2. Partitioning of λ-separable graphs

Informally the key result of this section is the following statement. Since a λ-separable graph can be split into two disconnected parts of comparable size by removing a small number of edges, the graph can also be split into many disconnected parts of bounded size by removing a small number of edges.

The following lemma is the modification of lemma 1 from [7] for planar graphs.

Lemma 2. *Let $\mathcal{G}$ be a monotone class of edge λ-separable graphs with the parameters α, β and m, where $0 < \lambda < 1$, $\frac{1}{2} \leq \alpha < 1$, $\beta \geq 0$, $m \geq 2$. Then for each $r \geq m - 1$ and for each graph $G \in \mathcal{G}$ with p vertices there exists partition of G into subgraphs such that*

- *The number of vertices in each subgraph does not exceed r.*
- *The total number of edges between subgraphs does not exceed $\frac{\delta p r^\lambda}{r}$, where δ is a constant common for all graphs of the class and for all values of r.*

We call the corresponding partition of the graph as r^λ-*partition.*

Proof. The proof of the lemma is similar to the proof of lemma 1 from [7]. We provide the detailed version of the proof for completeness.

Let $r \geq m-1$, $G \in \mathcal{G}$, $|V(G)| = p$. If $p \leq r$, then the trivial partition containing a single graph G suffices.

Let $p > r$. By the definition of edge λ-separability graph G can be partitioned into two subgraphs A and B with no more than αp vertices each and no more than βp^λ mutually connecting edges. Since $\mathcal{G}$ is a monotone class, $A, B \in \mathcal{G}$. Thus both A and B can be similarly partitioned into two subgraphs. Let us recursively partition all the subgraphs until we have only pieces with no more than r vertices.

Let us prove that the obtained partition is a r^λ-partition.

The constraint on the number of vertices in subgraphs (no more than r vertices per subgraph) is satisfied by the algorithm of partitioning.

Let X be the total number of edges deleted during the algorithm. We prove an upper bound for X. Let us split all subgraphs partitioned at any step of the algorithm into sets $\mathcal{G}_i$ depending on the size of a subgraph. We include into $\mathcal{G}_1$ subgraphs with a size from a half-open interval $(r, r\alpha^{-1}]$. Similarly we include into $\mathcal{G}_2$ subgraphs with a size from a half-open interval $(r\alpha^{-1}, r\alpha^{-2}]$, and so on. If $t = \lceil \log_\alpha \frac{r}{p} \rceil$, the last set $\mathcal{G}_t$ includes subgraphs with a size from $(r\alpha^{-(t-1)}, r\alpha^{-t}]$.

Let $1 \leq i \leq t$. Consider the set $\mathcal{G}_i$. Note that vertex sets of distinct subgraphs from $\mathcal{G}_i$ do not intersect, since the ratio of sizes of such subgraphs is less than α. Therefore the total size of all subgraphs in $\mathcal{G}_i$ does not exceed p. Hence $|\mathcal{G}_i| \leq \frac{p}{r} \cdot \alpha^{i-1}$. At the same time the total number of edges deleted when partitioning a graph from $\mathcal{G}_i$ does not exceed $\beta(r/\alpha^i)^\lambda$.

By summing over all subgraphs from sets $\mathcal{G}_i$ we obtain

$$X \leq \beta \sum_{i=1}^{t} \frac{\alpha^{i-1} p}{r} \left(\frac{r}{\alpha^i}\right)^\lambda \leq \frac{\beta}{\alpha^\lambda (1-\alpha^{1-\lambda})} \cdot \frac{p r^\lambda}{r}.$$

This matches the constraint on the number of edges mutually connecting subgraphs of a r^λ-partition. □

We use the following auxiliary notation below. Let $M, S > 0$. We denote by $K(M,S)$ the number of tuples $(x_1, \ldots, x_t)$ satisfying the condition

$$1 \leq x_i \leq M, \qquad \sum_{i=1}^{t} x_i \leq S. \tag{1}$$

If $\mathcal{G}$ is a monotone λ-separable class of graphs with vertex degree bounded by q, the properties of r^λ-partition may be stated as follows. Let $\bar{p} = \{p_i\}_{i=1}^t$ be a tuple of sizes of r^λ-partition subgraphs, and let $\bar{s} = \{s_i\}_{i=1}^t$ be a tuple of numbers of edges connecting r^λ-partition subgraphs with the rest of the graph. Then

$$\bar{p} \in K(r, p), \qquad \bar{s} \in K\left(qr, \frac{\delta p r^\lambda}{r}\right). \tag{2}$$

4. Lower bound for λ-separable supports

In this section we prove the key result of this paper namely a theorem on the lower bound for all supports from classes $\mathcal{G}(\lambda, q, \theta)$.

$$L(M_k^T, n) \gtrsim \max\left(\frac{2^n}{n}, \frac{2^n(1-\lambda)}{\log k}\right).$$

Note that the lower bound depends only on the separability function. Parameters q and θ do not affect the lower bound.

Substantively the proof is obtained in the following way. We partition a subgraph of a support into small fragments. Then we bound the number of Boolean functions computable in the subgraph by the number of Boolean functions computable in the fragments and the number of ways to conduct wires between the fragments.

Since the proof is technically involved we prove several auxiliary lemmas in a separate section 4.1. The proof of the main theorem is finished in section 4.2.

4.1. Auxiliary lemmas

The following lemma is an immediate corollary of a classic lemma [3].

Lemma 3 ([3], p. 198)**.** *Let $N(n, m, L)$ be the number of Boolean functions with no more than n inputs, no more than m outputs and the complexity not greater than L. Then there exists a constant c such that*

$$N(n, m, L) \le \big(c(n + L)\big)^{n+m+L}.$$

We use r^λ-partitioning of support subgraphs to obtain the lower bound. Technical lemma 4 bounds the number of Boolean operators computable in fragments of a r^λ-partition.

Let p and s be positive integers. We denote by $Z(p, s)$ the number of Boolean functions with no more than s inputs and outputs in total and the complexity not greater than p.

Recall the notation $K(M, S)$ introduced in section 3 for sets of tuples satisfying conditions (1). When proving lemma 4 we use the following simple property:

$$\text{If} \quad \bar{x} = \{x_i\}_{i=1}^t \in K(M, S), \quad \text{then} \quad \sum_{i=1}^t x_i \log x_i \le S \log M. \tag{3}$$

We also use the following inequality for non-negative x and y:

$$(x+y)\log(x+y) \le x\log x + y\log y + x + y, \tag{4}$$

which under the assumption $0\log 0 = 0$ is a corollary of the binary entropy bound $-a\log a - (1-a)\log(1-a) \le 1$ for $a = \frac{x}{x+y}$.

Lemma 4. *Let $q \ge 1$, $b \ge 0$, $d > 0$ be constants and let $k \to \infty$ be a parameter. Denote $r = (k\log k)^d$. Let L, M be numbers and $\bar{p} = \{p_i\}_{i=1}^t$, $\bar{s} = \{s_i\}_{i=1}^t$, $\bar{u} = \{u_i\}_{i=1}^t$ be tuples satisfying the following conditions:*

$$\bar{p} \in K(r, L), \qquad \bar{s} \in K\left(qkr, \frac{bL}{\log k}\right), \qquad u_i \ge 0, \qquad \sum_{i=1}^t u_i \le M. \tag{5}$$

Then

$$\sum_{i=1}^t \log Z(p_i, s_i+u_i) \le d\left(1+O\left(\frac{\log\log k}{\log k}\right)\right)L\log k + M\log M + O(M). \tag{6}$$

Proof. Using lemma 3, we have

$$Z(p_i, s_i+u_i) \le \big(c(p_i+s_i+u_i)\big)^{p_i+s_i+u_i}.$$

Taking the logarithm and summing by all tuple elements, we obtain

$$\sum_{i=1}^t \log Z(p_i, s_i+u_i) \le \sum_{i=1}^t (p_i+s_i+u_i)(\log c + \log(p_i+s_i+u_i)). \tag{7}$$

Using twice (4), then (3) with (5), we bound the right side of (7):

$$\begin{aligned}
&\sum_{i=1}^t (p_i+s_i+u_i)(\log c + \log(p_i+s_i+u_i)) \le \\
&\quad \le \sum_{i=1}^t (p_i\log p_i + s_i\log s_i + u_i\log u_i + (p_i+s_i+u_i)(\log c+2)) \le \\
&\quad \le L\log r + \frac{bL}{\log k}\underbrace{\log(qkr)}_{O(\log k)} + M\log M + \underbrace{(\log c+2)\left(L + \frac{bL}{\log k} + M\right)}_{O(L+M)} = \\
&\quad = L\log r + M\log M + O(L+M).
\end{aligned}$$

Substituting the bound into (7), we obtain

$$\begin{aligned}
\sum_{i=1}^t \log Z(p_i, s_i+u_i) &\le L\log r + M\log M + O(L+M) = \\
&= Ld(\log k + \log\log k) + M\log M + O(L+M) = \\
&= \left(1+O\left(\frac{\log\log k}{\log k}\right)\right)Ld\log k + M\log M + O(M).
\end{aligned}$$

□

The following two lemmas allow to obtain a trivial lower bound for Shannon function for circuits with an arbitrary support.

Lemma 5 ([3], theorem 11.5)**.** *For each constant $\epsilon > 0$ the ratio of Boolean functions of n variables satisfying the inequality*

$$L(f) \geq (1-\epsilon)\frac{2^n}{n}$$

approaches 1 *as $n \to \infty$.*

Lemma 6. *Let T be an arbitrary support, $k \in \mathbb{N}$, $n \to \infty$. Then*

$$L(M_k^T, n) \gtrsim \frac{2^n}{n}.$$

Proof. The lemma is an immediate corollary of lemma 5 and the fact that the complexity of a multilayer circuit is not less than the size of the corresponding Boolean circuit. □

4.2. The lower bound theorem

In this section we finish the proof of lower asymptotic bound for $L(M_k^T, n)$, where $T \in \mathcal{G}(\lambda, q, \theta)$. We also prove a corollary allowing to obtain a lower bound for supports having separability function of more general type, e. g. $\log p$, $\sqrt{p}\log\log p$, etc.

Lemma 7. *Let $T \in \mathcal{G}(\lambda, q, \theta)$. Let $N_k^T(n, m, L)$ be the number of Boolean functions in $B_{n,m}$ computable by k-layer circuits in T with size not greater than L. Then as $k \to \infty$, the following inequality holds:*

$$\log N_k^T(n,m,L) \leq \frac{L\log k}{1-\lambda}\left(1 + O\left(\frac{\log\log k}{\log k}\right)\right) + \\ + (n+m)\big(\log L + \log(n+m)\big) + O(n+m).$$

Proof. Let us denote $r = (k\log k)^{\frac{1}{1-\lambda}}$. We consider only great enough values of k to suffice the conditions on r from lemma 2. Thus every finite subgraph G of the support T has a r^λ-partition which we denote by $P(G)$.

We can build a mapping between k-layer circuits of size not greater than L computing a Boolean function in $B_{n,m}$ and tuples of the following objects:

1) Subgraph G of the support where we embed the corresponding Boolean circuit.

2) A tuple $\bar{v}$ of vertices of G where we embed inputs and outputs of the Boolean circuit.

3) A set of directed wires between fragments of $P(G)$.

4) A tuple of Boolean functions computed in fragments of $P(G)$.

It is easy to see that circuits computing different Boolean functions are mapped to different tuples. Thus we can bound the number of Boolean functions by the number of possible tuples. Obviously we can bound the number of elements per each tuple item and find the product of the bounds.

Denote the corresponding upper bounds as A_1, A_2, A_3, A_4, where the correspondence is defined by the order of items above.

As $T \in \mathcal{G}(\lambda, q, \theta)$, we have

$$A_1 \le \theta + \theta^2 + \cdots + \theta^L \le \frac{\theta^{L+1}}{\theta - 1}.$$

Let us bound A_2. Obviously there exist L^{n+m} tuples of $n+m$ vertices of G. Hence $A_2 \le L^{n+m}$.

The number of edges between the fragments of the r^λ-partition $P(G)$ is bounded by $\frac{\delta L r^\lambda}{r} = \frac{\delta L}{k \log k}$, where δ is a constant. In each of these edges we can conduct no more than k wires. Thus there can be no more than $\frac{\delta L}{\log k}$ wires mutually connecting the fragments of $P(G)$. For each of these wires there are three options: directed in one way, directed in the opposite way and missed. Therefore

$$A_3 \le 3^{\frac{\delta L}{\log k}}.$$

We introduce the following notation to obtain the bound for A_4. Let G_i be the fragments of $P(G)$, t be the number of the fragments, p_i be the number of vertices in i-th fragment, s_i be the number of wires that can be conducted outside from G_i, and u_i be the total number of inputs and outputs of the Boolean circuit embedded into G_i (i. e. the number of items in the tuple $\bar{v}$ corresponding to the vertices of G_i). It's clear that each Boolean function computable in G_i must have no more than p_i gates and no more than $s_i + u_i$ inputs and outputs in total. Hence the following inequality holds:

$$A_4 \le \prod_{i=1}^{t} Z(p_i, s_i + u_i).$$

By multiplying the bounds for A_i we obtain

$$N_k^T(n, m, L) \le \frac{\theta^{L+1}}{\theta - 1} \cdot L^{n+m} \cdot 3^{\frac{\delta L}{\log k}} \cdot \prod_{i=1}^{t} Z(p_i, s_i + u_i).$$

Taking the logarithm and omitting the negative addend be obtain

$$\log N_k^T(n,m,L) \le \underbrace{(L+1)\log\theta}_{O(L)} + (n+m)\log L + \\ + \underbrace{\frac{\delta L}{\log k}\log 3}_{o(L)} + \sum_{i=1}^{t} \log Z(p_i, s_i+u_i). \tag{8}$$

Let us bound the sum $\sum_{i=1}^{t} \log Z(p_i, s_i+u_i)$ in the right side of (8). We claim that we can apply lemma 4.

We obtain the following conditions for tuples $\bar{p} = \{p_i\}_{i=1}^t$, $\bar{s} = \{s_i\}_{i=1}^t$ by using (2): $\bar{p} \in K(r, L)$, $\bar{s} \in K\left(qkr, \frac{\delta L}{\log k}\right)$. For tuple $\bar{u} = \{u_i\}_{i=1}^t$ it's obvious that $u_i \ge 0$, $\sum_{i=1}^t u_i \le n+m$. Finally we have $q \ge 1$, $\delta \ge 0$, $\frac{1}{1-\lambda} > 0$, $k \to \infty$ and $r = (k\log k)^{\frac{1}{1-\lambda}}$. Thus all conditions of lemma 4 are satisfied. Hence

$$\sum_{i=1}^{t} \log Z(p_i, s_i+u_i) \le \frac{L\log k}{1-\lambda}\left(1 + O\left(\frac{\log\log k}{\log k}\right)\right) + \\ + (n+m)\log(n+m) + O(n+m).$$

Combining the bounds for addends in the right side of (8), we obtain

$$\log N_k^T(n,m,L) \le \frac{L\log k}{1-\lambda}\left(1 + O\left(\frac{\log\log k}{\log k}\right)\right) + \\ + (n+m)\big(\log L + \log(n+m)\big) + O(n+m).$$

□

In this paper we are interested in the case when $n+m$ is small compared to L. In this case we can simplify the inequality in lemma 7.

Corollary 1. *Under the conditions of lemma 7, if $k \to \infty$ and $n+m \le L/\log L$, then*

$$\log N_k^T(n,m,L) \le \frac{L\log k}{1-\lambda}\left(1 + O\left(\frac{\log\log k}{\log k}\right)\right).$$

Lemma 8. *Let $T \in \mathcal{G}(\lambda, q, \theta)$, $k \to \infty$, $n \to \infty$. Then*

$$L(M_k^T, n) \ge \frac{2^n(1-\lambda)}{\log k}\left(1 + O\left(\frac{\log\log k}{\log k}\right)\right).$$

Proof. It follows from lemma 6 that for great enough values of n and any k the inequality below holds:

$$L(M_k^T, n) \geq \frac{1}{2} \cdot \frac{2^n}{n}. \tag{9}$$

Let $k \to \infty$, $n \to \infty$. We denote $L = L(M_k^T, n)$ for brevity. Using the term $N_k^T(n, m, L)$ defined in lemma 7, we obtain the identity

$$N_k^T(n, 1, L) = 2^{2^n}.$$

It follows from (9) that $n = O(\log L) = o(L/\log L)$. By applying corollary 1 of lemma 7 we obtain

$$2^n \leq \frac{L \log k}{1 - \lambda}\left(1 + O\left(\frac{\log\log k}{\log k}\right)\right).$$

This implies the claim of the lemma. □

Theorem 1. *Let $0 < \lambda < 1$, $T \in \mathcal{G}(\lambda, q, \theta)$. Then*

$$L(M_k^T, n) \gtrsim \max\left(\frac{2^n}{n}, \frac{2^n(1-\lambda)}{\log k}\right) \quad \text{as} \quad k \to \infty,\ n \to \infty.$$

Proof. It follows from lemmas 6 and 8. □

Hereby we obtained the lower bound for λ-separable supports, i. e. for supports with separability function like p^λ. Using theorem 1 one can obtain lower bound for supports with separability function like $\log p$, $\sqrt{p}\log\log p$, etc.

Corollary 2. *Let $0 \leq \lambda_0 < 1$, $f(p) = O(p^\lambda)$ for all $\lambda > \lambda_0$. Let $T \in \mathcal{G}(q, \theta)$ be a $f(p)$-separable support. Then*

$$L(M_k^T, n) \gtrsim \max\left(\frac{2^n}{n}, \frac{2^n(1-\lambda_0)}{\log k}\right) \quad \text{as} \quad k \to \infty,\ n \to \infty. \tag{10}$$

Proof. Let $\lambda > \lambda_0$. It's clear that T is p^λ-separable. From theorem 1 we have $L(M_k^T, n) \gtrsim \max\left(\frac{2^n}{n}, \frac{2^n(1-\lambda)}{\log k}\right)$ as $k \to \infty$, $n \to \infty$.

Let us denote $g(\lambda, k, n) = \max\left(\frac{2^n}{n}, \frac{2^n(1-\lambda)}{\log k}\right)$. We have

$$\liminf_{\substack{k\to\infty \\ n\to\infty}} \frac{L(M_k^T, n)}{g(\lambda, k, n)} \geq 1 \quad \text{as} \quad \lambda_0 < \lambda < 1.$$

It's easy to see that $\frac{g(\lambda,k,n)}{g(\lambda_0,k,n)} \geq \frac{1-\lambda}{1-\lambda_0}$, therefore

$$A := \liminf_{\substack{k\to\infty \\ n\to\infty}} \frac{L(M_k^T, n)}{g(\lambda_0, k, n)} \geq \liminf_{\substack{k\to\infty \\ n\to\infty}} \frac{1-\lambda}{1-\lambda_0} \cdot \frac{L(M_k^T, n)}{g(\lambda, k, n)} \geq \frac{1-\lambda}{1-\lambda_0}$$

for all $\lambda_0 < \lambda < 1$.

Hence $A \geq \sup\limits_{1>\lambda>\lambda_0} \frac{1-\lambda}{1-\lambda_0} = 1$. The latter implies (10). □

5. Lower bound for d-dimensional circuits

In this section we obtain the lower bound for d-dimensional circuits and the asymptotics for d-dimensional rectangular circuits. In substance the lower bound for d-dimensional circuits is a corollary for the lower bound for λ-separable supports, since we prove that all d-dimensional supports belong to classes $\mathcal{G}(\lambda, q, \theta)$.

5.1. Properties of d-dimensional graphs

Essentially we have to prove that d-dimensional graphs have three properties: bounded vertex degree, exponentially bounded number of non-isomorphic subgraphs, and λ-separability.

Lemma 9. *Let T be a d-dimensional support (accordingly let $\mathcal{G}$ be a class of d-dimensional graphs) with a parameter c_e. Then degree of any vertex in T (accordingly in any graph from $\mathcal{G}$) is bounded by $(2c_e + 1)^d$.*

Proof. When placing arbitrary d-dimensional graph into d-dimensional space the neighborhood of any vertex is placed into a ball of radii c_e. Since d-dimensional balls with radii 0.5 and centers in graph vertices do not intersect and lie inside a ball with radii $c_e + 0.5$, the number of such balls cannot exceed the ratio of volumes of d-dimensional balls with radii $c_e + 0.5$ and 0.5 respectively. This ratio is equal to $(2c_e + 1)^d$. □

Lemma 10. *Let T be a d-dimensional graph. Then the number of non-labeled subgraphs of T with n vertices does not exceed θ^n, where θ is a constant.*

Proof. Immediately follows from the remark to lemma 2 in [15]. □

We apply the results of [21] to prove λ-separability of d-dimensional graphs.

Definition 4 ([21], definition 2.3)**.** Let $\alpha \geq 1$ be given, and let $B = \{B_1, B_2, \ldots, B_p\}$ be a set of closed d-dimensional balls with non-overlapping interiors. The α*-overlap graph* for B is the undirected graph with vertices $V = \{1, 2, \dots, p\}$ and edges

$$E = \{\{i, j\} \colon B_i \cap (\alpha \cdot B_j) \neq \varnothing \text{ and } (\alpha \cdot B_i) \cap B_j \neq \varnothing\},$$

where $\alpha \cdot B_j$ is a ball centered as B_j and having α times greater radii.

The following lemma shows the connection between d-dimensional graphs and α-overlap graphs.

Lemma 11. *Let $\mathcal{G}$ be a class of d-dimensional graphs with a parameter c_e. Then each graph in $\mathcal{G}$ can be supplemented by some number of edges (possibly* 0*) resulting to a* $2c_e$*-overlap graph in a d-dimensional space.*

Proof. Let $G \in \mathcal{G}$. Consider the placement of G into a d-dimensional space, and let G' be the $2c_e$-overlap graph for the balls of radii 0.5 and centers in the vertices of G. Since the distance between centers of any two balls is not less than 1, interiors of the balls do not intersect.

If there is an edge between two vertices in G, the distance between the centers of the corresponding balls does not exceed c_e. Therefore balls with the same centers and radii 0.5 and $0.5 \cdot 2c_e = c_e$ respectively would intersect. Hence all edges of G are also edges of G'. □

Lemma 12 ([21], theorem 2.4)**.** *Let $d \geq 1$, $\alpha \geq 1$ be constants. Then there exists a function*

$$f(p) = O\left(\alpha \cdot p^{\frac{d-1}{d}} + c(\alpha, d)\right)$$

such that each α-overlap graph in a d-dimensional space is vertex $f(p)$-separable. The separator splits its parent graph into pieces with no more than $\frac{d+1}{d+2}$ of the initial number of vertices.

Essentially lemma 12 claims $\frac{d-1}{d}$-separability of all α-overlap graphs in a d-dimensional space.

Remark 5. In the source [21] lemma 12 was stated in a slightly different way. Considering any α-overlap graph in d-dimensional space, it was claimed that the graph has a separator of size bounded by $O\left(\alpha \cdot p^{\frac{d-1}{d}} + c(\alpha, d)\right)$. Since the separability function is common for all graphs in a monotone class, we modified the statement of the lemma in this paper to emphasize the independence of the separability function from individual graphs.

Corollary 3. *Let $\mathcal{G}$ be a class of d-dimensional graphs. Then $\mathcal{G}$ is $p^{\frac{d-1}{d}}$-separable.*

Proof. Immediately follows from lemmas 11 and 12. □

5.2. Shannon function bounds

d-dimensional circuits. We apply the properties of d-dimensional graphs proved in the previous section and obtain the lower bound for d-dimensional circuits.

Theorem 2. *Let T be a d-dimensional support. Then*

$$L(M_k^T, n) \gtrsim \frac{2^n}{\min(n, d\log k)} \quad \textit{as} \quad k \to \infty,\ n \to \infty.$$

Proof. Immediately follows from theorem 1, lemmas 9, 10 and corollary 3. □

Multidimensional rectangular circuits. Multidimensional rectangular circuits are a special case of d-dimensional circuits, thus the lower bound from theorem 2 is also applicable for these circuits.

An upper bound of Shannon function for multidimensional rectangular circuits was proved in [26].

Lemma 13 ([26], theorem 1)**.**

$$L(M_k^d, n) \lesssim \frac{2^n}{\min(n, d\log k)} \quad \textit{as} \quad k \to \infty,\ n \to \infty.$$

Applying theorem 2 and lemma 13 we obtain the asymptotics of Shannon function for multidimensional rectangular circuits.

Corollary 4.

$$L(M_k^d, n) \sim \frac{2^n}{\min(n, d\log k)} \quad \textit{as} \quad k \to \infty,\ n \to \infty.$$

6. Conclusion

In this paper we proved the lower bound for Shannon function $L(M_k^T, n) \gtrsim \max\left(\frac{2^n}{n}, \frac{2^n(1-\lambda)}{\log k}\right)$ for any support T from a class $\mathcal{G}(\lambda, q, \theta)$. An important special case of such supports are d-dimensional graphs for which thereby we proved the lower bound $L(M_k^T, n) \gtrsim \frac{2^n}{\min(n,d\log k)}$.

A natural direction of developing the obtained results is examining classes of graphs with a separability function different from p^λ. For example, graphs supporting a placement in a hyperbolic space are of interest. It was proved in [14] that such graphs have logarithmic separability function. Corollary 2 of theorem 1 allows to obtain a lower bound for Shannon function for such graphs. However the question about upper bounds remains open.

References

[1] A. Albrecht, “On circuits of cellular elements”, *Problemi Kibernetiki*, **33** (1978), 209–214 (in Russian).

[2] N. Bonichon, C. Gavoille, N. Hanusse, D. Poulalhon, G. Schaeffer, “Planar Graphs, via Well-Orderly Maps and Trees”, *Graphs and Combinatorics*, **22** (2006), 185–202.

[3] A. V. Chashkin, *Discrete Mathematics*, Akademiya, Moscow, 2012 (in Russian), 352 pp.

[4] O. V. Cheremisin, “On the activity of cell circuits realising the system of all conjunctions”, *Discrete Mathematics*, **15**:2 (2003), 113–122 (in Russian); English translation in *Discrete Mathematics and Applications*, **13**:2 (2003), 209–219.

[5] A. A. Efimov, “The top assessment of energy consumption in a class of volume schemes”, *Intelligent Systems. Theory and Applications*, **23**:1 (2019), 117–132 (in Russian).

[6] A. A. Efimov, “The upper estimate of the volumetric power consumption of the circuits that implement boolean operators.”, *Intelligent Systems. Theory and Applications*, **23**:2 (2019), 105–124 (in Russian).

[7] G. N. Frederickson, “Fast algorithms for shortest paths in planar graphs, with applications”, *SIAM Journal on Computing*, **16**:6 (1987), 1004–1022.

[8] S. V. Gribok, “On one base for circuits of cellular elements”, *Vestnik Moskovskogo Universiteta. Seriya 15: Vichislitelnaya matematika i kibernetika*, **4** (1999), 36–39 (in Russian).

[9] G. V. Kalachev, “Order of power of planar circuits implementing Boolean functions”, *Discrete Mathematics*, **26**:1 (2014), 49–74 (in Russian); English translation in *Discrete Mathematics and Applications*, **24**:4 (2014), 185–205.

[10] G. V. Kalachev, “On the simultaneous minimization of area, power and depth of planar circuits computing partial Boolean operators”, *Intelligent Systems*, **20**:2 (2016), 203–266 (in Russian).

[11] G. V. Kalachev, “Bounds on power of planar circuits computing functions with limited number of ones”, *Intelligent Systems. Theory and Applications*, **21**:1 (2017), 28–96 (in Russian).

[12] G. V. Kalachev, “Bounds on power of planar circuits computing monotone functions”, *Intelligent Systems. Theory and Applications*, **21**:2 (2017), 163–192 (in Russian).

[13] G. V. Kalachev, “On the lower bound for the maximum potential of plain circuits with several outputs through the area”, *Intelligent Systems. Theory and Applications*, **22**:1 (2018), 111–117 (in Russian).

[14] S. Kisfaludi-Bak, “Hyperbolic Intersection Graphs and (Quasi)-Polynomial Time”, *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’20, Society for Industrial and Applied Mathematics, USA, Utah, Salt Lake City, 2020, 1621–1638.

[15] A. D. Korshunov, “On the complexity bounds of circuits of volumetric elements and volumetric Boolean circuits”, *Problemi Kibernetiki*, **19** (1967), 275–283 (in Russian).

[16] M. R. Kramer, J. van Leeuwen, “The VLSI complexity of Boolean functions”, *Logic and Machines: Decision Problems and Complexity*, eds. Börger E., Hasenjaeger G., Rödding D., Springer, Berlin, Heidelberg, 1984, 397–407.

[17] S. S. Kravtsov, “On the realization of Boolean functions in one class of logic elements and connectors”, *Problemi Kibernetiki*, **19** (1967), 285–293 (in Russian).

[18] R. J. Lipton, R. E. Tarjan, “A separator theorem for planar graphs”, *SIAM Journal on Applied Mathematics*, **36**:2 (1979), 177–189.

[19] O. B. Lupanov, “On the synthesis of some classes of control systems”, *Problemi Kibernetiki*, **10** (1963), 63–97 (in Russian).

[20] W. F. McColl, “Planar circuits have short specifications”, *2nd STACS. Lecture Notes in Computer Science*, **182** (1985), 231–242.

[21] G. L. Miller, S. Teng, W. Thurston, S. A. Vavasis, “Geometric separators for finite-element meshes”, *SIAM Journal on Scientific Computing*, **19**:2 (1998), 364–386.

[22] D. E. Muller, “Complexity in Electronic Switching Circuits”, *IRE Transactions on Electronic Computers*, **EC-5**:1 (1956), 15–19.

[23] J. E. Savage, “Planar Circuit Complexity and The Performance of VLSI Algorithms”, *VLSI Systems and Computations*, eds. Kung H.T., Sproull B., Steele G., Springer, Berlin, Heidelberg, 1981, 61–68.

[24] J. E. Savage, “The performance of multilective VLSI algorithms”, *Journal of Computer and System Sciences*, **29**:2 (1984), 243–273.

[25] N. A. Shkalikova, “On the implementation of Boolean functions by schemes of cellular elements”, *Mathematical Problems of Cybernetics.* V. 2, Nauka, Moscow, 1989, 177–197 (in Russian).

[26] T. R. Sitdikov, “The complexity of multidimensional rectangular circuits design”, *Intelligent Systems. Theory and Applications*, **23**:3 (2019), 61–80 (in Russian).

[27] C. D. Thompson, “Area-Time Complexity for VLSI”, *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC ’79, Association for Computing Machinery, New York, NY, USA, 1979, 81–88.

[28] J. D. Ullman, *Computational Aspects of VLSI*, W. H. Freeman & Co, USA, 1984.

The two-dimensional closest neighbor search problem solution using the cellular automata with locators [1]

D. I. Vasilev[2]

This article describes a cellular automaton with locators that solves the problem of finding the nearest neighbour. The problem is to find from a finite set of points the one closest to a predetermined "central"point. In contrast to the classical model of a cellular automaton, in the model under consideration, instantaneous transmission of signals through the ether at an arbitrary distance is allowed. It is shown that this possibility makes it possible to solve the problem in constant time, which is strikingly different from the one-dimensional case, where a logarithmic lower complexity estimate by the minimal distance is obtained.

Keywords: cellular automata, homogeneous structures, the closest neighbour search problem.

A *cellular automaton with locators* is a 8-tuple

$$\sigma = (\mathbb{Z}^k, E_n, V, E_q, +, L, \varphi, \psi)$$

where $\mathbb{Z}^k$ is the set of k-dimensional vectors with integer coordinates, $E_n = \{0, 1, \ldots, n-1\}$, $V = (\alpha_1, \ldots, \alpha_{h-1})$ is an ordered set of pairwise different nonzero vectors from $\mathbb{Z}^k$, $E_q = \{0, 1, \ldots, q-1\}$, $+$ is a commutative semigroup operation defined on E_q, $L = (\nu_1, \ldots, \nu_m)$ is an ordered set of pairwise different solid angles in $\mathbb{R}^k$ with a vertex at the origin, $\varphi : E_n^h \times E_q^m \to E_n$ is a function depending on the variables $x_0, x_1, \ldots, x_{h-1}, z_1, \ldots, z_m$ such that $\varphi(0, \ldots, 0) = 0$, $\psi : E_n^h \times E_q^m \to E_q$ is a function that depends on the variables $x_0, x_1, \ldots, x_{h-1}, z_1, \ldots, z_m$. Here the variables $x_0, x_1, \ldots, x_{h-1}$ take values from E_n and the variables $z_1, \ldots, z_m$ take values from E_q. Elements of the set $\mathbb{Z}^k$ are called *cells* of the cellular automaton σ; elements of the set E_n are called *cell states* of the cellular automaton σ; the set V is called the *neighborhood pattern* of the cellular automaton σ; elements of the set E_q are called *broadcasting signals*; the set L is called the *locator pattern* of the cellular automaton σ; the function φ is called the *local transition function* of the automaton σ; the function ψ is called the *broadcasting function* of

[1] Originally published in *Intellektualnye Sistemy. Teoriya i prilogeniya* (2021) **25**, No. 4, 83-87 (in Russian).

[2] *Vasilev Denis Igorevich* — junior researcher, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intelligent Systems, email: denis.vasilev.igor@gmail.com

the automaton σ. The state 0 is interpreted as *rest state* and the condition $\varphi(0, \ldots, 0) = 0$ is interpreted as a condition for maintaining the rest state.

This definition was introduced by Gasanov E.E. [1] and improved by Kalachev G.V. [2].

Let's formulate the closest neighbour search problem on the line. Let the I be the initial state of a cellular automaton on $\mathbb{Z}^1$ which satisfies the following conditions: a) Any cell is on one of $\{q_S; q_{C_0}, *\}$ states; b) There is only one q_{C_0} cell; c) There is a finite and non-empty set of q_S cells.

We will define that a cellular automaton state I' is solution for the problem I if I' satisfies the following conditions: a) The q_{C_0} cell from I is in q_{CF} state in I'; b) The cell which is the closest to the q_{C_0} cell in I is in the q_{SE} state. If there are several closest cells then an arbitrary one must be chosen; c) The rest cells are in $*$ state.

We define that cellular automaton σ solves the closest neighbour search problem if it satisfies the following conditions: a) If the initial state I of the cellular automaton is a closest neighbour search problem then the automaton must end up in I' state which is solution for I; b) If the automaton takes state S which is solution for some closest neighbour search problem this state must be kept for all the next tacts.

Let's call *the general position of the closest neighbour search problem* a problem in which there is at least one cell in the state q_S on both sides of the cell in the state q_{C_0}.

In the article [3], the theorem was proved for the one-dimensional case:

Theorem 1. *There is a cellular automaton σ with 25 states and with the power of the broadcasting alphabet 12, which solves the problem of finding the closest neighbour in a time not exceeding $\log_2 s + 7$, where s is the distance from the central cell with the initial state q_{C_0} to its nearest neighbour with the initial state q_S.*

An article with a similar lower complexity estimate was sent to the editorial office of the Vestnik of the MSU journal:

Theorem 2. *For any cellular automaton with locators σ with the power of the broadcasting alphabet M and any general position of the nearest neighbour search problem I, $T_I^\sigma > \log_M(\frac{s}{5})$ is performed, where s is the distance from the cell in the state q_{C_0} to the nearest cell in the state q_S in the problem I, and T_I^σ is the number of the clock cycles for which the automaton σ solves the problem I.*

Thus, for the one-dimensional case of the nearest neighbour search problem, the order of complexity of the problem is obtained.

It turned out that for the dimension $n \geq 2$, similar estimates are incorrect, since for such problems it was possible to build an automaton with locators that solves them in constant time. Here is an example of such an automaton for the case of $n = 2$:

Theorem 3. *There is a cellular automaton σ with 15 states and with the power of the broadcasting alphabet* 40*, which solves the two-dimensional problem of finding the nearest neighbour in a time not exceeding* 13*.*

Consider a cellular automaton with locators σ with a set of locators consisting of locators from Fig. 1 and one expanded locator that reads the sum of the signals of all cells of the cellular space. Let's define the broadcasting alphabet as a subset $\{0; 1\}^{20}$ and a semigroup operation of a component-by-component maximum on it. For convenience, we will denote the cell signals by one or more numbers - the numbers of non-zero positions in the ether signal. So the signal $(0, 1, 0, 1, 0, ..., 0, 0, 0)$ we will record as a pair of signals 2 and 4.

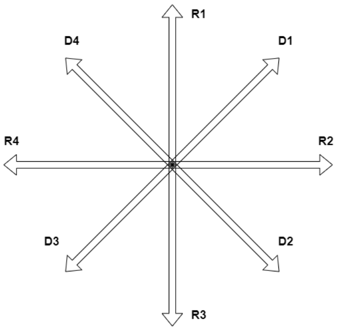


Fig. 1. Locators layout and names

Let's define a coordinate system on the cellular space with the center in the central cell. The central cell in the constructed automaton constantly sends a 1 signal to the ether. Cells that receive such a signal from the $R3$ locator will realize that they are on the upper coordinate semi-axis. Similarly, each cell can identify its location on the other three coordinate semi-axes. The cells located on the semi-axes constantly send a signal to the ether with the number of their semi-axis (upper — 2, right —3, lower —4 and left — 5). By these signals, each cell can recognize in which coordinate quarter it is located. For example, having received the signal 2 from the locator $R4$ and the signal 3 from the locator $R3$, it is possible to uniquely determine that the cell in question is in the first coordinate quarter. The idea of functioning of the constructed automaton is to project along the Manhattan circle all the points from the problem on one semi-axis, find the projection closest to the center, and then restore its prototype. For example, a point from the first quarter can send a special signal that is read only by the right semi-axis. A point from the right semi-axis, having received such a signal from the locator

$D4$, will understand that it is a projection of one of the points of the problem (Fig. 2 on the left). By repeating this iteration 4 times, you can project all the points onto the upper semi-axis (Fig. 2 on the right).

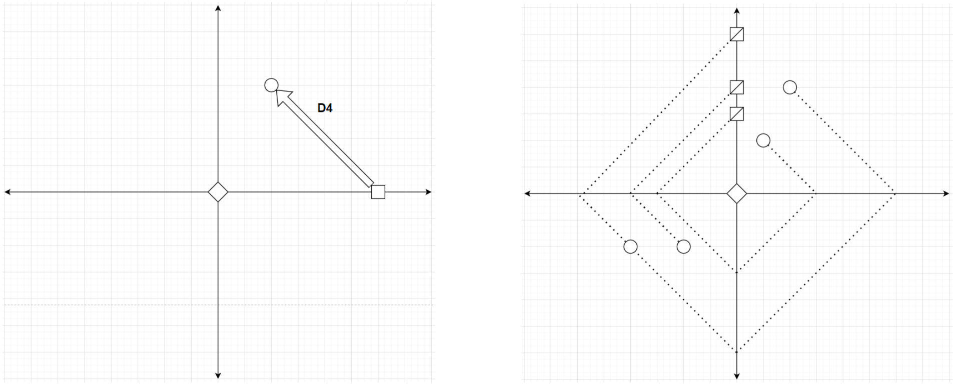


Fig. 2. On the left is an example of projecting a single point onto a semi-axis. On the right, the progress of projecting the problem onto the upper semi-axis.

To find the nearest neighbour on the upper semi-axis, it is enough for each candidate to broadcast a special signal, and after receiving such a signal from the $R3$ locator, he will withdraw (i.e., switch to the default state). After the nearest projection is found, it is enough to restore its prototype by the reverse course of the described algorithm.

References

[1] Gasanov E. E., "Cellular automata with locators", *Intelligent systems. Theory and Applications*, **20**:2 (2020), 121–133 (In Russian).

[2] Kalachev G. V., "Remarks on the definition of a cellular automaton with locators", *Intelligent systems. Theory and Applications*, **24**:4 (2020), 47–57 (In Russian).

[3] Vasilev D. I., "The closest neighbour problem solution using the cellular automata with locators model", *Intelligent systems. Theory and Applications*, **24**:3 (2020), 99–120 (In Russian).

Implementation of key-value databases by cellular automata with locators [1]

E. E. Gasanov[2], A. A. Propazhin[3]

In this paper, it is shown that key-value databases can be implemented by cellular automata with locators in such a way that the execution time of basic operations, such as search, insert, delete, will not depend on the size of the database and will be equal to the total length of the key and value.

Keywords: Cellular automata with locators, key-value databases.

The key-value database is a popular data storage paradigm now, also called a dictionary. Such a database can be represented as a set of pairs of strings (k, v), where the first string k is called the key and serves as the identifier of the pair, and the second string v is called the value. *String* is a sequence of characters of some alphabet A, ending with a special character 0, called *the string ending character*, and the character 0 does not belong to the alphabet A.

The key-value database supports the following operations:

1) *inserting a pair* (k, v) — an entry with the key k and the value v appears in the database; if an entry with the key k already existed in the database, then the value is replaced with v;

2) *deleting an entry with a key* k — an entry (k, v) is deleted from the database; if there is no entry with the key k in the database, then the database does not change;

3) *searching for an element by key* k — there is an entry (k, v) in the database, and the value v is returned as an answer; if there is no entry with the key k in the database, then the answer is an empty set.

The concept of a cellular automaton with locators was introduced by E. E. Gasanov in [1] and refined by G. V. Kalachev in [2]. The exact definition can be found in the above-mentioned works, but here we give an informal definition of a one-dimensional cellular automaton with one complete locator, which will be used in this work.

A one-dimensional cellular automaton is a set of identical elementary automata located at integer nodes of the real line, and called cells. The

[1] Originally published in *Intellektualnye Sistemy. Teoriya i prilogeniya* (2021) **25**, No. 4, 108-112 (in Russian).

[2] *Gasanov Elyar Eldarovich* — professor, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems, e-mail: el_gasanov@gmail.com.

[3] *Propazhin Artem Alekseevich* — student, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems, e-mail: artem.propazhin@mail.ru.

behavior of a cellular automaton is specified by the transition function, namely, the state of the cell at the next moment is uniquely determined by its own state at the current moment and the states of its neighbors. In what follows, we will assume that each cell has exactly two neighbors: the closest to the left and the closest to the right. One of the states of a cell is called an quiescent state, and if the cell and its neighbors are in an quiescent state, then the next moment the cell will remain in an quiescent state. In a cellular automaton with one complete locator, in addition to the alphabet of states, there is a broadcasting alphabet with a commutative semigroup operation specified on it. In this paper, the maximum operation will be used as such an operation. In a cellular automaton with one complete locator, each cell sends a certain signal from the broadcasting alphabet, determined by the broadcasting function, to the air every moment. The value of the cell broadcasting function is determined by the cell's own state, the states of its neighbors and the total broadcasting signal. The total broadcasting signal is formed by summing up the broadcasting signals of all cells with the exception of the signal of this cell using a semigroup operation. In a cellular automaton with one complete locator, the value of the cell transition function is also determined by the cell's own state, the states of its neighbors and the total broadcasting signal.

Let's introduce another entity — *the database user*. We will assume that the database user has the ability to send signals from the broadcasting alphabet to the air, i.e. his broadcasting signal is summarized with broadcasting signals of all automata. And database user can receive total broadcasting signal from the air. We will assume that a cellular automaton with locators, together with the user, implements a key-value database if the broadcasting alphabet is a set consisting of pairs of the form ("command", $A \cup \{0\}$), where "command" takes one of the values: "search", "insert", "delete", "answer", "no answer", "no command" (here "no command" is the minimum element), and the behavior of the cellular automaton is set as follows.

1) The user broadcasts the "search" command and the first character of the key k. Then the user sequentially broadcasts the "no command" command and all the other characters of the key, including the character 0. If there is no entry with the key k, in the database, then on the next clock cycle after the 0 character is given, the cellular automaton broadcasts the command "no answer". If there is an entry (k, v) in the database, then on the next clock cycle after the character 0 is submitted, the cellular automaton broadcasts a pair ("answer", a), where $a \in A$ is the first element of the value v, and in subsequent cycles the cellular automaton sequentially broadcasts all other symbols of the value v, including the character 0.

2) The user broadcasts the "insert" command and the first character of the key k. Then the user sequentially broadcasts the "no command" command

and all the other characters of the key, including the character 0. After that the user sequentially broadcasts the “no command“ command and all characters of the value v, including the character 0. As a result, a pair (k, v) appears in the database implemented by the cellular automaton, i.e. if the “search“ command and the key k are subsequently submitted to the cellular automaton, then the cellular automaton will return the value of v in response.

3) The user submits the “delete“ command and the first character of the key k. Then the user sequentially broadcasts the “no command“ command and all the other characters of the key, including the character 0. As a result, the entry with the key k disappears from the database implemented by the cellular automaton, i.e., during the subsequent search for the key k, the cellular automaton returns “no answer“.

Theorem 1. *There is a cellular automaton with locators and a user which implements a key-value database, and for which the execution time of the search, insert, delete operations will not exceed the total length of the key and value.*

Here is the idea of proving this theorem.

We will use a one-dimensional cellular automaton with one complete locator, and elementary automata lying in the negative region of the numerical line will not be used. The alphabet of broadcasting has already been described above. The alphabet of states consists of triples of the form (“command“, “cell type“, $A \cup \{0, *\}$), where “command“ takes one of the values: “search“, “insert“, “delete“, “answer“; “cell type“ takes one of the values: “commander“, “current cell type key“, “current cell type value“, “unprocessed cell“, “receiver for recording“; $*$ is a special character. If an elementary automaton will be marked with the symbol “receiver for recording“ then starting from this automaton the next record will be written to the database. At the initial moment, the automaton with the number 0 will be marked with symbol “receiver for recording“. Database records will be pairs of key-value strings, and the first character of each record will be marked with the state “commander“.

Let’s describe the functioning of this automaton. When the insertion begins, all commanders receive a signal from the air in the form (“insert“, k_1), where k_1 is the first character of the key. If k_1 matches the value stored in the state of the commander cell, then in the state of the cell next to the right, the command changes to “insert“ and the cell type to “current cell type key“. On the next clock cycle, the cell next to the right of the commander, also receiving a signal from the air, checks for a match with the stored symbol. After that the current cell changes its type to “unprocessed cell“. When a match occurs, in the state of the cell next to the right of current cell, the command changes to “insert“ and the cell type to “current cell type key“ and a similar process occurs to the previous one. The process occurs sequentially up to and including the 0 symbol. The key verification process starts simultaneously with all commanders. If at some point there is a mismatch, then the type of the next cell changes to “unprocessed cell“. If we have reached the 0 symbol, it means that we have found a record with the desired key, and this record should be excluded from the database. Exclusions from the database are achieved by the fact that 0 in the state of the current cell changes to the symbol $*$. As a result, there will be no match in subsequent searches.

Simultaneously with the search for the key, starting from the cell marked with the state “receiver for recording“ the key-value pair is sequentially recorded into the database. When the insert command arrives, the cell in the “receiver for recording“ state becomes the commander and stores the first character of the key k_1 in its state. At the same time, the cell to the right of this cell changes the command to “insert“ and the cell type to “receiver for recording“. Further, the cells that have the “insert“ command and the “receiver for recording“ cell type retain the key symbols and values coming from the air in their state. In this case, the state (“insert“, “receiver for recording“) moves to the next cell on the right. The exception is when the 0 character arrives for the value. At this moment, the next cell on the right switches to the state (“search“, “receiver for recording“).

Deletion is similar to insertion. Only during deletion does the process of writing to the end of the database not begin.

During the search, the key is read in the same way as when inserting. But after reaching the 0 symbol, the replacement of the symbol 0 with $*$ does not occur. The cell next to the right of the cell with the 0 symbol changes the command to “search“ and the cell type to “current cell type value“. Cells in this state send a signal to the air (“answer“, a), where $a \in A$ is the stored value. The current cell changes its type to “unprocessed cell“ after sending the signal. Signals with value symbols are sequentially sent to the air up to and including the signal with the symbol 0.

Список литературы

[1] Gasanov E. E., “Cellular automata with locators”, *Intelligent Systems. Theory and applications*, **24**:2 (2020), 121–133 (In Russian).

[2] Kalachev G. V., “Remarks on the definition of a cellular automaton with locators”, *Intelligent Systems. Theory and applications*, **24**:4 (2020), 47–56 (In Russian).

К сведению авторов публикаций в журнале «Интеллектуальные системы. Теория и приложения»

В соответствии с требованиями ВАК РФ к изданиям, входящим в перечень ведущих рецензируемых научных журналов и изданий, в которых могут быть опубликованы основные научные результаты диссертаций на соискание ученой степени доктора и кандидата наук, статьи в журнал «Интеллектуальные системы. Теория и приложения» предоставляются авторами в следующей форме:
1. Статьи, набранные в пакете LaTeX, предоставляются к загрузке через WEB-форму http://intsysjournal.org/generator_form .
2. К статье прилагаются файлы, содержащие название статьи на русском и английском языках, аннотацию на русском и английском языках (не более 50 слов), список ключевых слов на русском и английском языках (не более 20 слов), информация об авторах: Ф.И.О. полностью, место работы, должность, ученая степень и/или звание (если имеется), контактные телефоны (с кодом города и страны), e-mail, почтовый адрес с индексом города (домашний или служебный).
3. Список литературы оформляется в едином формате, установленном системой Российского индекса научного цитирования.
4. За публикацию статей в журнале «Интеллектуальные системы. Теория и приложения» с авторов (в том числе аспирантов высших учебных заведений) статей, рекомендованных к публикации, плата не взимается. Оттиски статей авторам не предоставляются. Журнал распространяется по подписке, экземпляры журнала рассылаются подписчикам наложенным платежом. Условия подписки публикуются в каталоге НТИ «Роспечать», индекс журнала 64559.
5. Доступ к электронной версии последнего вышедшего номера осуществляется через НЭБ «Российский индекс научного цитирования». Номера, вышедшие ранее, размещаются на сайте http://intsysjournal.org, и доступ к ним бесплатный. Там же будут размещены аннотации всех публикуемых статей.

Подписано в печать: 20.12.2021
Дата выхода: 25.12.2021
Тираж: 200 зкз.
Цена свободная
Свидетельство о регистрации СМИ: ПИ № ФС77–58444 от 25 июня 2014 г.,
выдано Федеральной службой по надзору в сфере связи, информационных
технологий и массовых коммуникаций(Роскомнадзор).