

Degenerate Coding and Sequence Compacting

Maya Gorel
Kirzhner V.M.

Vienna, Preprint ESI 1819 (2006)

July 25, 2006

Supported by the Austrian Federal Ministry of Education, Science and Culture
Available via <http://www.esi.ac.at>

Degenerate coding and sequence compacting

Maya Gorel

Institute of Evolution, University of Haifa, Haifa 31905, Israel

Kirzhner V.M.

Institute of Evolution, University of Haifa, Haifa 31905, Israel

valery@esti.haifa.ac.il

Contents

1 Introduction.	4
1.1 Biological coding structures.	4
1.2 Some features and purposes of biological coding.	5
1.3 Formulation of the problem and some models of biological coding.	6
1.3.1 Definition of the code and synonymous partitions.	6
1.3.2 Basic model of synonymous coding.	6
1.4 Sequences of symbols and primary graphic constructions.	7
2 Simplest properties of synonymous partitions. Cartesian synonymous partitions	10
2.1 An example of a synonymous partition.	10
2.2 Two-dimensional Cartesian synonymous partitions.	13
2.3 Formulation of the general model of regular Cartesian synonymous partitions.	16
2.4 Dense sequences of quotient words and dense sequences of words over the basic alphabet.	19
3 Abstract dense sequences	22
3.1 Words of length two.	22
3.2 Words of an arbitrary length.	30
3.3 Combinatorial connectivity of the set of dense sequences.	33
4 Application of the general theory to the case of a Cartesian partition	38
4.1 Euler cycles of Cartesian sequences and sequences of letters.	38
4.2 Cartesian synonymous partitions. Words of length two.	39
4.3 Triplet code of amino acids.	41

Abstract

Superposition of signals in DNA molecule is a sufficiently general principle of information coding. The degeneracy of the code is necessary for the effective realization of this principle, with which the possibility to transmit differently certain information also allows to place some other information on the same DNA fragment. Code words that are equivalent in the information sense – synonyms – form synonymous group; and the entire set of code words is broken into the synonymous groups. This paper is dedicated to construction and analysis of the model of synonymous coding. Sequence containing one word from each synonymous group is selected as the model signal. Meaningfully such sequences can be examined, for example, with respect to the triplet code. Cartesian synonymous groups are selected as a model of synonymous partition. Namely, assume that there is given a basic alphabet $A_s = \{a_1, \dots, a_s\}$. Let us examine q of its, generally speaking, different partitions $\mathbf{A}_i = \{\mathbf{a}_{1,i}, \dots, \mathbf{a}_{r_i,i}\}$ where $i = 1, \dots, q$. Then, by definition, the set of the words from $\mathbf{a}_{j1,1} \times \mathbf{a}_{j2,2} \times \dots \times \mathbf{a}_{jq,q}$ forms a Cartesian synonymous group.

We define and examine Cartesian synonymous partitions and their simplest properties. It is shown, that the task of constructing the model signal can be represented as the task of constructing a certain sequence in the special alphabet, letters of which form the subsets of the initial alphabet. The necessary concepts of dense sequence in the terms of this alphabet are introduced and their connection with the dense sequences of the letters of the basic alphabet is investigated. Transition to the new alphabet removes some combinatory difficulties related to the selection of the representative of a synonymous group. This makes possible to apply effectively the modification of the De Bruijn graph when examining the existence of a dense signal. It is convenient to investigate the concepts appearing in this case within the framework of a more general system - the system of abstract alphabets. This system is a straight generalization of sufficiently real biological models on the one side. On the other side it makes possible to describe more complex hypothetical rules of coding. Various conditions of existence of dense coding are obtained for a such system of abstract alphabets, based, in particular, on the theorem of Hall about perfect matchings in a bipartite graph and on the theory of flows in the networks. It is shown also that the sequences over abstract alphabets cannot always be transferred into one another by transposition of fragments, in contrast to the well known property of sequences of letters of a regular alphabet.

The obtained general results are applied to the case of Cartesian synonymous partition for the words of length 2 and to the triplet code of amino acids. In the latter case it is proved that the synonymous groups of the amino-acid code are not always Cartesian, but they can be represented as an association of a small number of Cartesian synonymous groups. Therefore the analysis of the amino-acid code is reduced to the independent analysis of 16 versions of Cartesian synonymous partitions. This decomposition of the amino-acid code into a series of Cartesian codes made possible not only to analyze an existence of dense sequences in the triplet code, but also to calculate them all using a De Bruijn graph.

1. Introduction.

1.1. *Biological coding structures.*

It is well known that the hereditary information contained in DNA is written down using 4-letters alphabet *A*, *T*, *C*, and *G*. In this work we speak only about informational value of DNA and consequently we represent DNA as some sequence of letters of the over mentioned alphabet. In DNA the message with elementary structure is coded by a number of letters going successively - this formation is called a *code word*. The most known and most likely historically first of discovered codes is a triplet code. All possible 3-letters words participate in this code - all of them are code words. There exist totally 64 words for the 4-letter alphabet. This code codes 20 amino acids, therefore it is degenerate. Coding of one amino acid can be made using several alternative words which from the information point of view are absolutely equal (see Table 1).

Amino acids of various organisms are identical and are used to create proteins, which are the final goal of processing of the DNA information. Not only are amino acids coded in DNA but also proteins themselves, which, unlike amino acids, can differ even within the limits of one species (or even at parents and children!). In this sense it is interesting to note that the coding of amino acids does not carry significant informative value – it is more like "an instruction on their construction". The basic information incorporated in DNA contained in the coding of proteins is an instruction how to build proteins out of standard amino acids. For coding of proteins, triplets are placed one after another and form a *coding fragment* of DNA. Without going into details, it is possible to consider this fragment a gene. However, actually it can be only a part of a gene. In the triplet code there exist three words that do not code an amino acid but stand at the end of each such coding fragment. Each of these words refers to as "stop-codon" and serves for interruption of reading codes in a corresponding biological process. Thus, even in a triplet code, which is practically completely devoted to coding of amino acids, there are code words with another function, the controlling function. According to the knowledge gained for the last 25 years about genomes of the most various organisms, the triplet DNA - the protein code - is only one of many ways of genetic coding in DNA. "The genetic contents of DNA sequences is determined by superposition of many codes, among which there is not only triplet code but also various non-triplet codes: modulation code, RNA folding code, genome segmentation code, chromatin code, protein folding code, gene splicing code, DNA shape code, transcription codes in mRNA, translation framing code, and some other codes. These different kinds of coding cooperate in genome, and different codes are quite often overlapped" [1]. These codes are not necessarily connected with the code words. In particular, *distributed* formations (i.e. consisting of several sets of words) can possess coding properties. Thus, various kinds of information can be coded: the information directly connected with production of the proteins and the information connected with DNA molecules servicing. Both

these systems do not cooperate and can overlap. It is necessary to note also coding signals related to "indistinct" coding. For example, the curvature of a DNA molecule in a certain fragment serves as a signal for reading systems. However, the curvature itself is not defined by a fixed set of letters, but by some dominance of the quantity of certain letters on the fragment being investigated. Thus, the curvature does not define a signal, but defines some "functional". The required value of such a "functional" is achieved on a huge number of possible words. A similar example is presented by repetitions, which are likely to be perceived "indistinctly" by the reading system.

Table 1. 20 Amino Acids in Human Protein: Table of DNA Base Triplets

<i>Amino Acid</i>	<i>DNA Base Triplets</i>
alanine	CGA, CGG, CGT, CGC
arginine	GCA, GCG, GCT, GCC, TCT, TCC
asparagine	TTA, TTG
aspartate	CTA, CTG
cysteine	ACA, ACG
glutamate	CTT, CTC
glutamine	GTT, GTC
glycine	CCA, CCG, CCT, CCC
histidine	GTA, GTG
isoleucine	TAA, TAG, TAT
leucine	AAT, AAC, GAA, GAG, GAT, GAC
lysine	TTT, TTC
methionine	TAC
phenylalanine	AAA, AAG
proline	GGA, GGG, GGT, GGC
serine	AGA, AGG, AGT, AGC, TCA, TCG
stop	ATG, ATT, ACT
threonine	TGA, TGG, TGT, TGC
tryptophan	ACC
tyrosine	ATA, ATC
valine	CAA, CAG, CAT, CAC

1.2. *Some features and purposes of biological coding.*

Unlike traditional problems of the coding theory, coding in DNA is not oriented to correction of possible mistakes when transferring information. It is more like an advanced human language. Indeed, degeneration of the majority of codes allows

transferring of the same message A in a variety of ways. The formal contents of message A does not vary from synonymic replacement. There is an example of overlapping of the protein code with the code of spatial structure of DNA in paper [2]. Also it is well-known that the shift of the reading frame on 1 or 2 positions, while coding a virus genome leads to an occurrence of an intelligent (but another!) protein.

Thus, coding of each type of the information in DNA is done using some type of a code. The coding is done based on the words of various lengths, on distributed messages, and on indistinct codes. For any given type of a code there exist groups of words that are equivalent words from the information point of view. We call such groups *synonymous groups*.

What is a possible function of degenerate (*synonymous*) coding? Such coding allows to compress information, and condensation of information is useful for minimization of genome in case of viruses and bacteria. It would seem that for huge DNA of supreme animals it is not so essential. However, in this case it is often necessary to place some polytypic signals, such as the chromatic code, the splicing code and the code of amino acids, on a small fragment of DNA. Just because they are necessary on this fragment. On the other hand, according to conjecture [2], overlapping of code signals in viruses is useful also since it raises the probability of destruction of an organism if mutating in such combined parts, i.e. it increases the stability of a genome at the population level.

1.3. Formulation of the problem and some models of biological coding.

1.3.1. Definition of the code and synonymous partitions.

Assume that there is given an alphabet \mathcal{A}_s , where s is the number of its letters. Let the code words be the words of length q over this alphabet and let them belong to a set \mathcal{K}_q . \mathcal{K}_q does not necessarily coincide with the entire set of words of length q over the alphabet \mathcal{A}_s . Further, let $\mathcal{P}(\mathcal{K}_q)$ be a certain partition of the set \mathcal{K}_q into non-empty subsets. Assume that to each subset is assigned one message, which can be transmitted by producing any element of this subset. Elements of each subset, in that way, are *synonyms*, and such a partition, $\mathcal{P}(\mathcal{K}_q)$, will be called *synonymous*.

1.3.2. Basic model of synonymous coding.

In this work we take as a basis some simplest and most studied elements of the biological coding. Earlier we noted that the code words of the triplet code in DNA are read by the decoding system without overlapping, as successive triplets. Since all 3-letter words are code words for the triplet code, readout by the successive triplets from any position, of course, leads to an occurrence of a sequence of the code words. However, generally speaking, it will be a meaningless combination, to which no protein corresponds. Nevertheless, in principle the standard triplet code possesses the ability of *a dense coding* with a step 1 or 2. That is, code words can overlap in the

message, having, accordingly, 1 or 2 common letters. We already mentioned above an example of coding genome of a virus, where the shift of the frame of readout to 1 or 2 positions also leads to the occurrence of the code of real protein. The possibility of such coding is closely related to the degeneracy of the code. Based on the example of dense coding, let us define a standard message, with respect to which we will test quality and special features of synonymous partitions. This standard message is a sequence that contains exactly one q -character word from each synonymous class as its subword, and every two adjacent q -character subwords have exactly $q-1$ letters in common. We will call this sequence a *dense sequence*. In our work dense sequences are analyzed from several points of view. First of all, in our opinion, such message reflects the formal ability of the code words of different synonymous groups to be combined with each other. From a biological point of view, for the case of the triplet code, a dense sequence is a certain universal element in DNA, since it enables to have all possible types of amino acids in a very limited region. Also dense sequences provide an example of highly informative sequences. Consequently, dense sequences can be complex in the sense of different definitions of complexity. Therefore, first of all, we will investigate the criteria of existence of a dense sequence in different synonymous partitions; evaluate the capacity of the set of all dense sequences and some characteristics of this set of sequences. In particular, we will separately examine the procedure of the transposition of fragments in a dense sequence transferring it into another dense sequence. This transposition procedure has a biologically realizable mechanism, since the possibility of the transposition of DNA fragments is known. This transposition can be considered as a natural process of the evolution of words. We call a subset of sequences that can be transformed into one another a *connected subset*. It is interesting to find the number of maximal connected subsets. If there are several connected subsets, then their relationship in different genomes can be different and is connected with the phylogenetic relations of genomes. One should note that a similar algorithm of transposition is used for the generation of random words with the complete retention of the set of synonyms [3], [4].

In our model it is natural to examine sequences of symbols. However, cyclic sequences (contours) are frequently examined within the framework of study of words over a given alphabet. Although, in certain cases we will examine contours, the basic statements deal with sequences.

1.4. *Sequences of symbols and primary graphic constructions.*

A significant portion of literature is devoted to the study of the properties of sequences over an alphabet of a fixed length [5]. In particular, the possibilities of "packing" of a given set of words into one sequence of minimum length are investigated. The simplest task in this theory consists in the tight packing of all words of length q over the given alphabet - packing without the repetition of words. Namely, if there are two words of form aS and Sb , where S is a certain word of length

$q-1$, then it is possible to form the fragment of the desired sequence - aSb . When reading this sequence from the first position the first word aS occurs, and when reading from the second position the second word Sb occurs. For example, three words $abcde$, $bcdef$, $cdefg$ are tightly packed with the following sequence:

a	b	c	d	e		
	b	c	d	e	f	
		c	d	e	f	g
a	b	c	d	e	f	g

In research of the problem of dense packing of words, the graphic representation of sequences suggested by De Bruijn [6] is frequently used. Let \mathcal{A}_s be an alphabet and let q be the length of a code word w_q . Let us also assume that all words of length q over the alphabet \mathcal{A}_s are code words. Let us examine a set G_{q-1} of all possible words w_{q-1} of length $q-1$ over the given alphabet. Further, let us examine the graph whose set of vertices is identical to the set G_{q-1} . Two vertices w_{q-1} and w'_{q-1} are connected by an arc from w_{q-1} to w'_{q-1} , if the word w_{q-1} without the first letter is equal to the word w'_{q-1} without the last letter. That is, $w_{q-1} = aU$ and $w'_{q-1} = Ub$, where U is a word of length $q-2$, a, b are letters of the alphabet. Then it is possible to assign the word aUb of length q to the arc between these vertices. Thus, to the next arc from the vertex w'_{q-1} to the vertex w''_{q-1} corresponds a word Ubc , where c is a certain letter of the alphabet. Then the path consisting of two sequential arcs (w_{q-1}, w'_{q-1}) and (w'_{q-1}, w''_{q-1}) corresponds to the sequence $aUbc$, which contains two words of length q , with the shift of one letter: aUb and Ubc . This graph is known as De Bruijn graph. It also should be noted that De Bruijn graph is an Euler graph. Indeed, by definition, the number of arcs leaving every vertex and the number of arcs entering it equals to the number of letters of the alphabet. Now let us consider the following Statement.

Statement 1.4.1 *If G is a connected directed graph, and if at each point of G there are the same number of arcs going out as coming in, then there is a directed cycle in G that goes through every arc of G in its given direction, and uses no arc twice [4].*

The directed cycle in the graph G mentioned in the Statement above is an Euler cycle. We note that each *Euler cycle* on De Bruijn graph corresponds to a cyclic sequence, in which every possible word of length q is obtained exactly once as a result of the shift of the reading frame by one letter. The sequence possessing analogous property can be obtained from the cycle when cutting it in any place and concatenating to the obtained end $q-1$ letters from the beginning of the sequence. If the Euler property is disrupted in two vertices, such that in one vertex the number of entering arcs is greater than the number of emanating arcs by one, and in the other vertex vice versa, then there is no Euler cycle, but a sequence including all

words of the given length exists. There are also formulas for the number of different Euler cycles.

It is obvious that construction of a dense sequence s does not correspond to an Euler cycle in the De Bruijn graph in case of nontrivial synonymous partition. In the case of the synonymous partition the arcs of graph are divided into the subsets in a certain way. Thus the problem of construction of the dense sequence is reduced to the problem of construction of the path that contains exactly one arc from each subset.

2. Simplest properties of synonymous partitions. Cartesian synonymous partitions

2.1. An example of a synonymous partition.

Let us examine now the simplest example of a synonymous partition.

Example 2.1.1. Let $s = 2$ ($\mathcal{A}_2 = \{A, B\}$) and $q = 2$. There are four two-letter words $\{AA, AB, BA, BB\}$. Let us break this set arbitrarily into two non-empty synonymous classes, for example, $\{AA\}$, $\{AB, BA, BB\}$. According to the formulation of the problem, it is necessary to verify whether there exists a dense sequence s (or contour) for this partition. Since there are only two classes, the sequence must contain two words with the shift of 1, i.e. to have length 3. It is obvious that the word AAB is the desired sequence since it contains the two-letter word AA from the first synonymous set $\{AA\}$ and the word AB from the second set $\{AB, BA, BB\}$. There is one additional sequence BAA with the same properties: BA belongs to the second synonymous set and AA to the first one. Thus, there are two words s for this synonymous partition (moreover, the synonym BB is not used), and packing by a contour is impossible at all. Indeed, such a contour must consist of two letters and it is necessary that these letters will be AA . But, obviously, this contour contains no words from the second synonymous group.

Let us examine now all possible synonymous partitions into two groups in this example. There are 7 such combinations:

$$\begin{aligned}
 & \{AA\}, \{AB, BA, BB\}; \\
 & \{AB\}, \{AA, BA, BB\}; \\
 & \{AA, BA\}, \{AB, BB\}; \\
 & \{AA, AB\}, \{BA, BB\}; \\
 & \{AA, BB\}, \{AB, BA\};
 \end{aligned} \tag{2.1}$$

plus those obtained from (2.1) by transposition of symbols A and B . (The last three partitions are preserved under this transposition of letters.) It is sufficient to solve our task only for partitions (2.1). In the partition $\{AB\}, \{AA, BA, BB\}$ there are four complete words s : AAB, ABA, ABB and BAB . In this case all synonyms of both groups are used. There is also a contour AB , which includes two words AB and BA from different synonymous partitions. We can say that this synonymous partition is more effective in terms of quantity of possible versions of the dense packing (capacity of the set \mathcal{S}) than the first one. Further, for the partition $\{AA, BA\}, \{AB, BB\}$ there are four words s : AAB, BAB, ABA, BBA and one contour

AB . For the partition $\{AA, BB\}$, $\{AB, BA\}$ there are words AAB, BAA, ABB, BBA , and no contour. For the partition $\{AA, AB\}$, $\{BB, BA\}$ we have words BAA, ABB, ABA, BAB and a contour AB .

Thus, the example is completely investigated. We can see that for any synonymous partition into two sets there exist dense sequences s , and the quantity of such sequences depends on the partition and varies from 2 to 4. Contour does not exist for all partitions.

Statement 2.1.2. *Let for any alphabet \mathcal{A}_s $s \geq 2$ and any $q \geq 2$ the set of the code words \mathcal{K}_q include all possible words of length q . Then set \mathcal{S} is not empty for any partition into two non-empty synonymous groups.*

Proof. Each synonymous partition is a partition of the set of the arcs of the corresponding De Bruijn graph. Statement 2.1.2 claims that there is a vertex of the graph entered by an arc from one synonymous set and exited by an arc from the other set. Let us prove this. Let us take an arbitrary Euler cycle of this graph. By definition, this is a path through a graph, which includes every arc exactly once. Therefore, we can locate a vertex of the cycle with two arcs from the different synonymous groups incident to it. \square

For the partition into three synonymous groups, the set \mathcal{S} can be empty. Let us examine the following synonymous partition with $s = 2$ and $q = 3$: $\{AAA\}$, $\{BBB\}$, $\{\text{everything else}\}$. It is obvious that the sequence s , containing the words AAA and BBB must be at least six-letters long; whereas with three synonymous groups the sequence s has a length 5. However, the following statement holds.

Statement 2.1.3. *Let the set of code words \mathcal{K}_q include all possible words of length q for any alphabet \mathcal{A}_s where $s \geq 2$ and $q = 2$. Then for any partition into three non-empty synonymous groups the set \mathcal{S} is not empty.*

Proof. Let us prove that for $s \geq 5$ it is possible to select such four symbols of the alphabet \mathcal{A}_s that the intersection of the set of the words over these symbols with each of three synonymous sets is not empty. In other words, the contraction of synonymous partition into this alphabet, reduced to the 4 symbols, also has three non-empty synonymous groups. Let us place the symbols of the alphabet \mathcal{A}_s at the integral points of the X and Y axes of rectangular system of coordinates. (Let us call such points symbolic.) Then the words of length two are the corresponding points of Cartesian plane. Let us consider the following cases:

Case 1: For any $A \in \mathcal{A}_s$ all words BA are synonyms. Then for any $B \in \mathcal{A}_s$ there exist $A_1, A_2, A_3 \in \mathcal{A}_s$ such that BA_1, BA_2, BA_3 belong to different synonymous groups. Four symbols: B, A_1, A_2, A_3 .

Case 2: There exists $A \in \mathcal{A}_s$ such that BA and CA are not synonyms, and DA is synonymous either to BA or to CA for any $D \in \mathcal{A}_s$. Let A_1A_2 be not synonymous to BA and CA . Then A_1A_2 is not synonymous to A_1A , which is not synonymous to either BA or CA . Four symbols: A_1, A_2, A, B (or C).

Case 3: There exists $A \in \mathcal{A}_s$ such that BA, CA and DA are not synonymous, and EA is synonymous to one of them for any $E \in \mathcal{A}_s$. Four symbols: B, C, D, A .

Fig. 1. Case 1. Geometrical illustration of proof of Statement 2.1.3

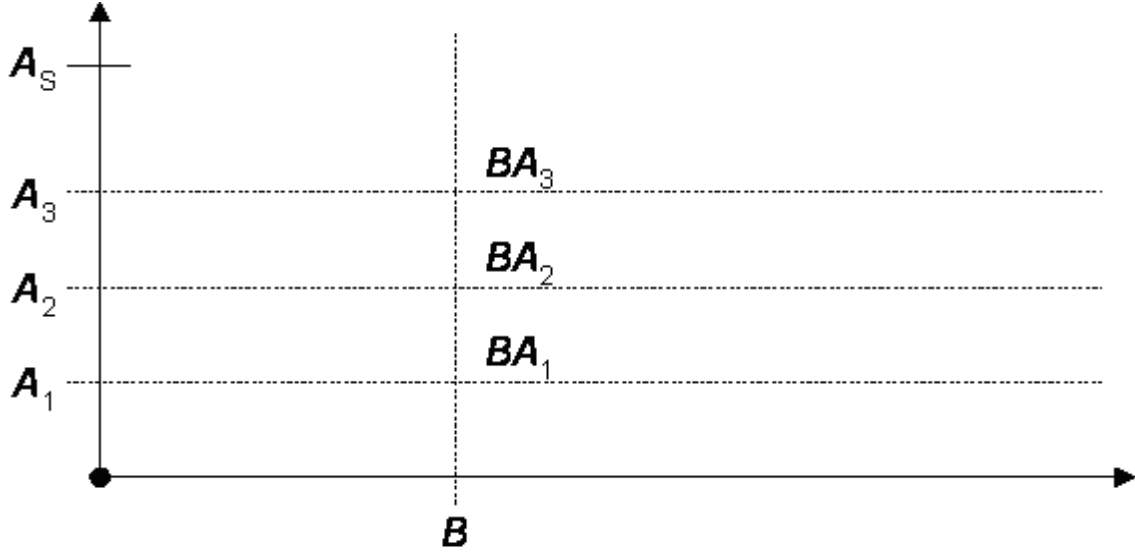
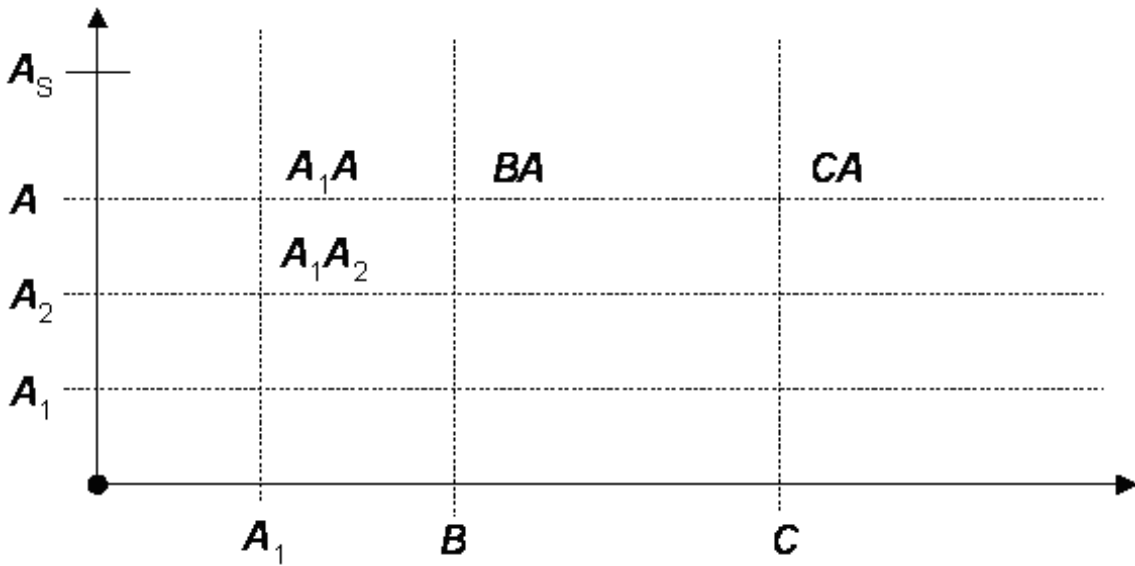
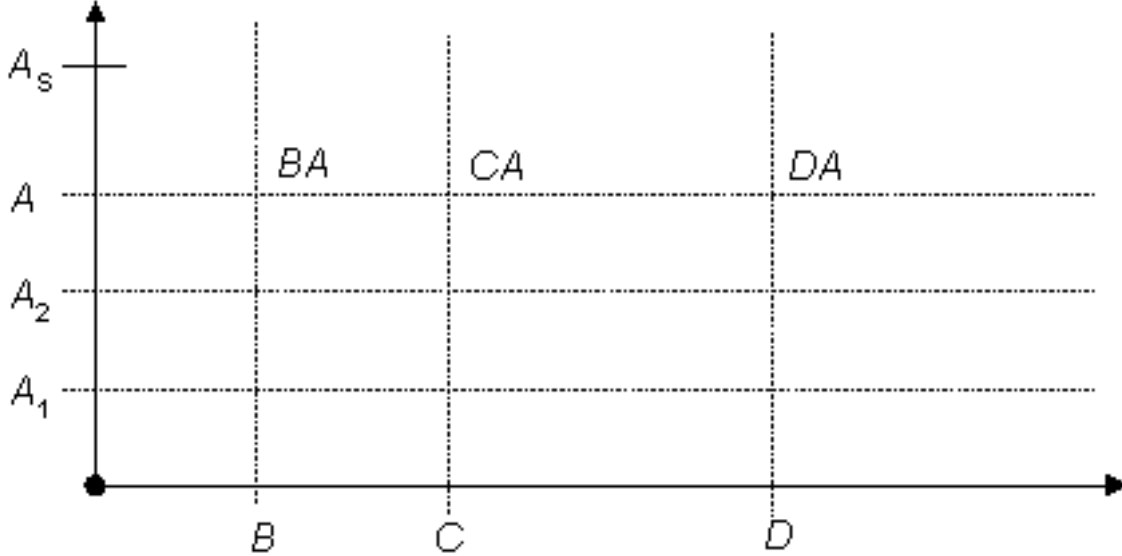


Fig. 2. Case 2. Geometrical illustration of proof of Statement 2.1.3



Thus, it remains to prove Statement 2.1.3 for cases where $s = 2, 3, 4$. This task was solved by sorting out all the possibilities on the computer. It is shown that set \mathcal{S} is not empty in all cases for all nontrivial partitions into three synonymous groups. \square

Fig. 3. Case 3. Geometrical illustration of proof of Statement 2.1.3



Further analysis of the number of synonymous partitions only in terms of s and q is ineffective. Indeed, let $s = 3$ ($\mathcal{A}_3 = \{A, B, C\}$) and $q = 2$. There exist 9 two-letter words. Let us examine the partition $\{AA\}, \{BB\}, \{CC\}$ and $\{\text{everything else}\}$. Here we have 4 synonymous groups. Three groups consist of one word each and cannot be connected directly. It means that for their connection it is necessary to use at least two words from the fourth group, so that the obtained word will have not less than two synonyms. Therefore, there is no dense sequence for this partition, i.e. the set \mathcal{S} is empty. The previous analysis can be summarized by the following table:

Number of synonymous groups	Parameters, when set \mathcal{S} always is not empty
2	$s \geq 2, q \geq 2$
3	$s \geq 2, q = 2$
4	$s = 2, q = 2$

2.2. Two-dimensional Cartesian synonymous partitions.

Assume that there is given a finite alphabet $\mathcal{A}_s = \{a_1, \dots, a_s\}$. We will use a Cartesian plane to represent all possible two-letter words over this alphabet. Let us arrange the letters of the alphabet \mathcal{A}_s on the axes of the rectangular two-dimensional

coordinate system, say, in the ascending order of numbers. Then the set of all possible two-letter words \mathcal{K}_2 can be identified with the set of points (i, j) ($1 \leq i, j \leq s$) on the plane.

Example 2.2.1. Let $\mathbf{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_r\}$ be a partition of the set \mathcal{A}_s into the pairs of disjoint sets (classes) \mathbf{a}_r ($1 \leq r \leq s$; $\mathbf{a}_i \cap \mathbf{a}_j = \emptyset$, if $i \neq j$ and $\cup \mathbf{a}_i = \mathcal{A}_s$ for $i = 1, \dots, r$). Let us define a new alphabet. We will call the set \mathbf{a}_r a *quotient letter*, the set \mathbf{A} the *alphabet*, and the set \mathcal{A}_s in this context, the *basic alphabet*. The above partition induces the partition $\mathcal{P}(\mathcal{K}_2)$ of all code words into the groups. All words obtained by the direct product of \mathbf{a}_i by \mathbf{a}_j belong to one group. We will assume that each such group is synonymous, i.e. all code words included into this group contain the same message. It is convenient to denote synonymous group as a pair $(\mathbf{a}_i, \mathbf{a}_j)$ (see Figure 5). Let us call this pair of classes a *quotient word*. Let us define now the rules of the formation of the sequences of quotient words. Namely, word $(\mathbf{a}_i, \mathbf{a}_j)$ follows the word $(\mathbf{a}_u, \mathbf{a}_v)$, if class \mathbf{a}_v is equal to class \mathbf{a}_i . Let us write down this chain as

$$(\mathbf{a}_u, \mathbf{a}_v)(\mathbf{a}_v, \mathbf{a}_j), \quad (2.2)$$

Clearly, such definition is coherent with the standard one for the sequence of letters. Let x be any element of the class \mathbf{a}_v . Then (2.2) can be understood as any triplet $\alpha x \beta$, where α and β are any elements of the corresponding sets ($\alpha \in \mathbf{a}_u$, $\beta \in \mathbf{a}_j$). Let us call such set of sequences of letters matched with the chain of quotient words the *projection of the chain of quotient words*. Further, by analogy for the quotient words $(\mathbf{a}_v, \mathbf{a}_j)$ and $(\mathbf{a}_p, \mathbf{a}_q)$ sets \mathbf{a}_j and \mathbf{a}_p coincide and thus we can write down sequence of three words of the classes

$$(\mathbf{a}_u, \mathbf{a}_v)(\mathbf{a}_v, \mathbf{a}_j)(\mathbf{a}_j, \mathbf{a}_q). \quad (2.3)$$

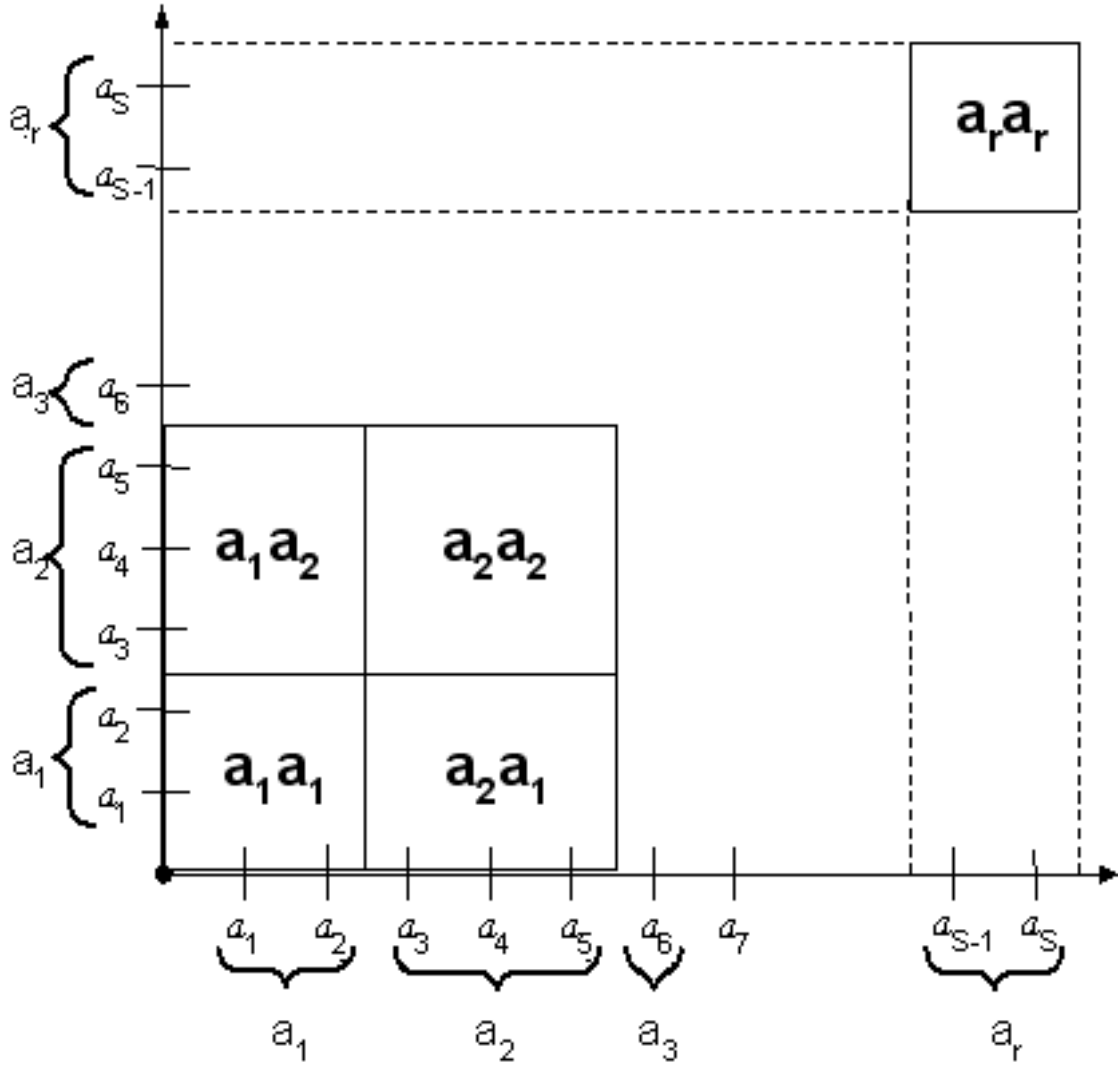
Obviously, the projection of this chain consists of all possible words of the form $\alpha x y \beta$, where y is any element from the set \mathbf{a}_j , selected independently of x .

In the standard situation the alphabet \mathbf{A} coincides with the initial alphabet $\mathcal{A}_s = \{a_1, \dots, a_s\}$, when the partition is trivial ($\mathbf{a}_i \equiv a_i$). Thus, in this situation we can use a Cartesian plane too.

Example 2.2.2. Let $\mathbf{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_r\}$ and $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_h\}$ be two partitions of the set \mathcal{A}_s . The pair (\mathbf{A}, \mathbf{B}) induces the partition of all code words into the groups as in the previous example: all words obtained by the direct product of the set \mathbf{a}_i by the set \mathbf{b}_j belong to the same group. We will also assume that each such group is synonymous and will denote it as the quotient word $(\mathbf{a}_i, \mathbf{b}_j)$. Let us define now the rules of the formation of the sequences in the set of classes. Namely, word $(\mathbf{a}_i, \mathbf{b}_j)$ follows word $(\mathbf{a}_u, \mathbf{b}_v)$ if the intersection of the second element of the latter word (class \mathbf{b}_v) with the first element of the former word (class \mathbf{a}_i) is not empty. Let us write down this chain as

$$(\mathbf{a}_u, \mathbf{b}_v)(\mathbf{a}_i, \mathbf{b}_j), \mathbf{b}_v \cap \mathbf{a}_i \neq \emptyset. \quad (2.4)$$

Fig. 4. Partition of the set \mathcal{A}_s into the pairs of disjoint sets \mathbf{a}_r ; a pair $(\mathbf{a}_i, \mathbf{a}_j)$ is a *synonymous group*



It is easy to see that such definition is coherent with the standard one for the sequence of letters (see Section 1.4). Namely, let x be any element from the intersection of sets \mathbf{b}_v and \mathbf{a}_i . Then (2.4) can be understood as any triplet $\alpha x \beta$, where $\alpha \in \mathbf{a}_u$, $\beta \in \mathbf{b}_j$. Further, by analogy for the quotient words $(\mathbf{a}_i, \mathbf{b}_j)$ and $(\mathbf{a}_p, \mathbf{b}_q)$ we can write down a sequence of three quotient words

$$(\mathbf{a}_u, \mathbf{b}_v)(\mathbf{a}_i, \mathbf{b}_j)(\mathbf{a}_p, \mathbf{b}_q), \quad (2.5)$$

if the intersection of sets \mathbf{b}_j and \mathbf{a}_p is non-empty. It is obvious that the projection of this chain consists of all possible words of the form $\alpha xy\beta$, where y can be any element from the intersection of sets \mathbf{b}_j and \mathbf{a}_p selected independently of x .

By definition, for any basic alphabet \mathcal{A}_s , a *Cartesian synonymous partition* is a partition of $\mathcal{A}_s \times \mathcal{A}_s$ into pairs of type $\mathbf{a}_i \times \mathbf{b}_j$, where $\mathbf{a}_i, \mathbf{b}_j$ are some subsets of the letters of the basic alphabet. In the previous examples sets $\{\mathbf{a}_i\}$ and $\{\mathbf{b}_i\}$ form the partition of the letters of the basic alphabet \mathcal{A}_s . Let us call such partitions *regular Cartesian synonymous partitions*.

Example 2.2.3. Let a set of the code words over the alphabet \mathcal{A}_s be divided into "rectangles", which do not intersect and cover the entire Cartesian square $\mathcal{A}_s \times \mathcal{A}_s$. The projections of the sides of these rectangles on X and Y axes form on these axes, generally speaking, two systems of intersecting intervals such that their union on each axis, obviously, is equal to \mathcal{A}_s . Let us denote both these systems as earlier $\mathbf{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_r\}$ and $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_h\}$. In this case, however, sets $\mathbf{a}_i, \mathbf{a}_j$ can intersect and the amount of sets r and h in each system can be greater than the length of the initial alphabet, s . For example, let us examine a partition containing pairs (a_i, b_1) with $i = 1, \dots, s$ and square $(a_2, a_3) \times (b_2, b_3)$ (see Figure 6). There exist already $s+1$ intervals on the X -axis. Nevertheless, it is possible to describe each "rectangle" (a synonymous set), as earlier, by an appropriate pair $(\mathbf{a}_i, \mathbf{b}_j)$ or by a single quotient word. The definition of a sequence of quotient words is transferred from Example 2.2.2. It is obvious that this is an *irregular Cartesian synonymous partition*.

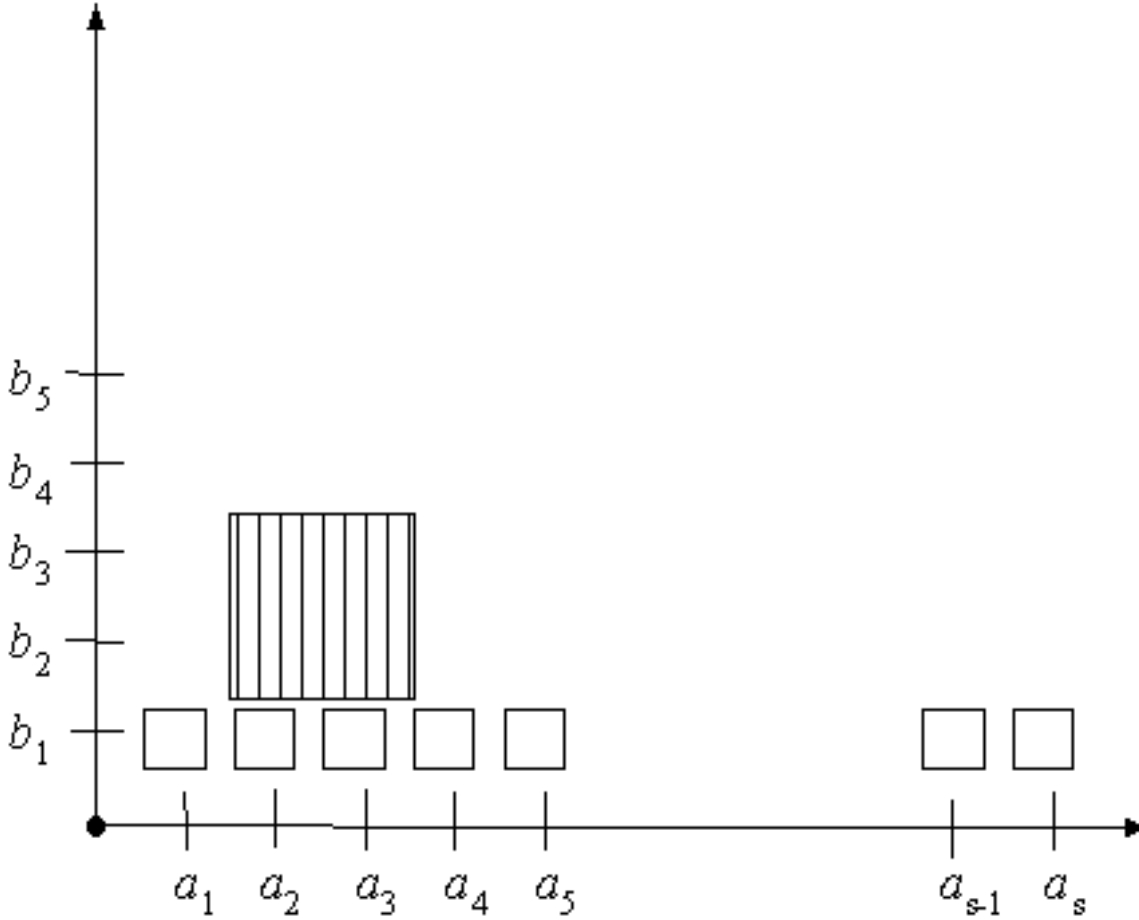
2.3. Formulation of the general model of regular Cartesian synonymous partitions.

At this point let us give a completely obvious generalization of regular Cartesian synonymous partitions to the words of arbitrary length q . Assume that there are given an alphabet \mathcal{A}_s and a set of q , generally speaking, distinct partitions $\mathbf{A}_i = \{\mathbf{a}_{1,i}, \dots, \mathbf{a}_{r_i,i}\}$ where $i = 1, \dots, q$. As earlier, let us combine words that belong to the Cartesian product $\mathbf{a}_{j_1,1} \times \mathbf{a}_{j_2,2} \times \dots \times \mathbf{a}_{j_q,q}$, where at place t stands any element of the set \mathbf{A}_t , which is a quotient letter. We call the corresponding sequence $\sigma = \mathbf{a}_{j_1,1} \mathbf{a}_{j_2,2} \dots \mathbf{a}_{j_q,q}$ a *quotient word*. Let us define $P_i(\sigma) = \mathbf{a}_{j_i,i}$, i.e. $P_i(\sigma)$ is the quotient letter standing in the word σ at place i . Consider now the set \mathbf{A} , which is the union of all letters of all alphabets: $\mathbf{A} = \cup \mathbf{A}_i, i = 1, \dots, q$. In this case two formally identical sets viewed as the letters of two different alphabets are considered different. Let us define a *Z-family* $Z = \{\mathbf{a}_t\}$ in \mathbf{A} by the following properties:

- (1) the intersection of all elements in Z is non-empty ($\cap \mathbf{a}_t = u(Z) \neq \emptyset, \mathbf{a}_t \in Z$),
- (2) for any $\mathbf{a}_t \notin Z$ $\mathbf{a}_t \cap u(Z) = \emptyset$.

$$(2.6)$$

Fig. 5. Pairs (a_i, b_1) with $i = 1, \dots, s$ and a square $(a_2, a_3) \times (b_2, b_3)$



Let us call the set $u(Z)$ the *basis of the Z-family Z*.

Proposition 2.3.1. *For a regular Cartesian synonymous partition each Z-family Z contains exactly one set (one letter) of each alphabet, $|Z|=q$.*

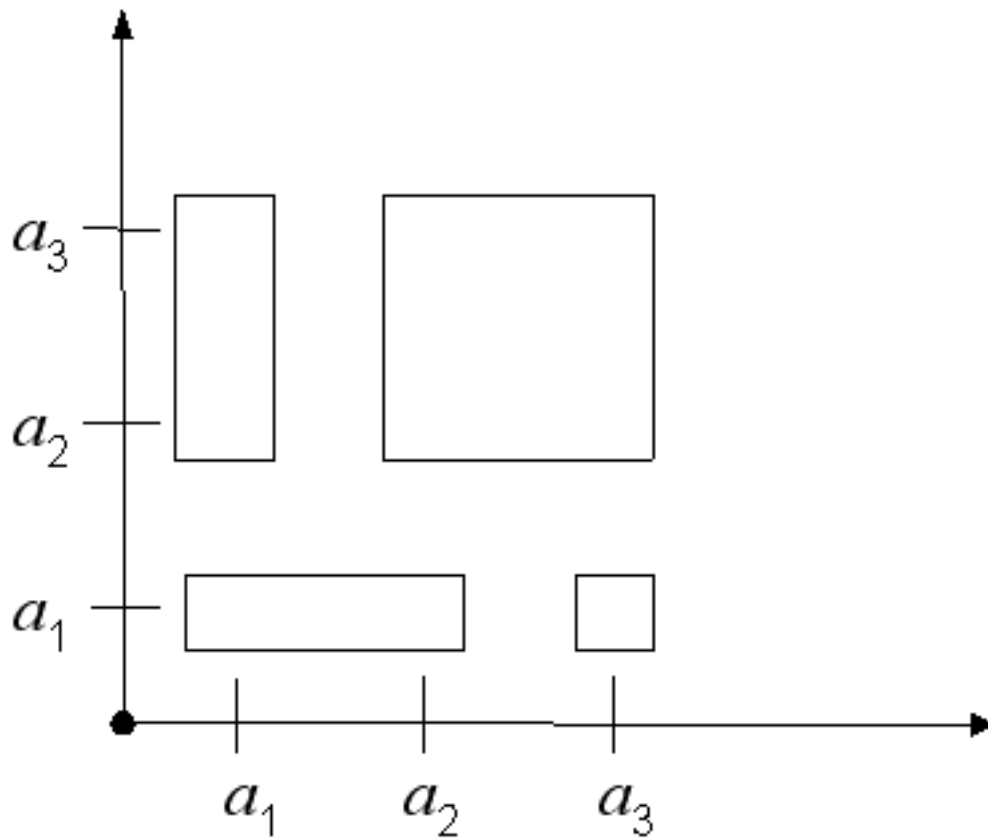
Proof. Indeed, in the regular case, letters of each alphabet, by definition, form a partition of the basic alphabet A_s , i.e. the intersection of any two letters of one alphabet is empty. Consequently, the presence of more than one letter of one alphabet in the set Z contradicts property 1 in (2.6). Assume now that a certain alphabet A_i is not represented in Z . By definition, there is at least one letter of this alphabet such that its intersection with $u(Z)$ is not empty. It is possible to add any such letter to Z . This contradicts property 2 in (2.6) and, thus, our assumption is erroneous. \square

It follows from the proof of Proposition 2.3.1 that the collection of different Z -families, generally speaking, is not the partition of the set \mathbf{A} , since Z -families can intersect.

It is obvious that the assertion in Proposition 2.3.1 is erroneous for the irregular case.

Example 2.3.1. Let a set of the code words over the alphabet \mathcal{A}_3 be divided into "rectangles", which do not intersect and cover the entire Cartesian square $\mathcal{A}_3 \times \mathcal{A}_3$ (see Figure 2-5). Thus we have two quotient alphabets: $\mathbf{A}_1 = \{\{a_1, a_2\}, \{a_2, a_3\}, \{a_1\}, \{a_3\}\}$ and $\mathbf{A}_2 = \{\{a_1\}, \{a_2, a_3\}\}$. One of the Z -families in this case is $Z_1 = \{\{a_1, a_2\}, \{a_1\}, \{a_1\}\}$, whereas the second quotient letter belongs to \mathbf{A}_1 and the third quotient letter belongs to \mathbf{A}_2 . Z_1 contains two quotient letters from the alphabet \mathbf{A}_1 . Therefore, Proposition 2.3.1 is true only for the regular Cartesian synonymous partition.

Fig. 6. Illustration of Example 2.3.1.



Proposition 2.3.2. *If $u(Z_1)$ and $u(Z_2)$ are the bases of two different Z -families Z_1 and Z_2 in a regular Cartesian synonymous partition, then $u(Z_1) \cap u(Z_2) = \emptyset$.*

Proof. Indeed, if Z_1 and Z_2 are different, then there is at least one alphabet \mathbf{A}_i that is represented in these Z -families by different letters. But different letters of one alphabet in the regular case have an empty intersection. \square

Proposition 2.3.3. *In the regular case, for any letter of the basic alphabet \mathcal{A}_s there exists a Z -family containing this letter in its basis.*

Proof. Let $x \in \mathcal{A}_s$. By definition, there is at least one quotient letter, containing x , in each of the alphabets \mathbf{A}_i . The set of these letters obviously satisfies (2.6) and, thus, forms a Z -family. \square

Combining Proposition 2.3.2 and Proposition 2.3.3 we obtain:

Proposition 2.3.4. *In the regular case the bases of all Z -families form a partition of the basic alphabet \mathcal{A}_s .*

Let us examine now an ordered set of quotient words of length q

$$\sigma_1, \dots, \sigma_N. \tag{2.7}$$

Let the following condition be true for the set (2.7):
the set of letters

$$T(i, t) = \{P_q(\sigma_{i+t-q}), \dots, P_{t+1}(\sigma_{i-1}), P_t(\sigma_i), P_{t-1}(\sigma_{i+1}), \dots, P_1(\sigma_{i+t-1})\} \tag{2.8}$$

is a subset of a certain Z -family for any pair (i, t) .

Such ordered set of quotient words (2.7) we call a sequence of quotient words. Let us call a set $T(i, t)$ a *cut of the sequence at point (i, t)* . In the cyclic version and "in the middle of sequence" set in (2.8) is obviously a Z -family. For the irregular case, generally speaking, for any point (i, t) the cut is a subset of the certain Z -family.

It is obvious that the general construction includes definition for the sequence of the quotient words of length $q = 2$ given in Example 2.2.2.

The described construction is a simple generalization of the standard one, when a match of $2, \dots, q$ letters of the previous word with $1, \dots, q-1$ of those of the following word is required to record the sequence of the type (2.7).

2.4. Dense sequences of quotient words and dense sequences of words over the basic alphabet.

Sequences of quotient words are defined in the model of Cartesian synonymous partitions described above. Since there are no nontrivial synonymous partitions on the level of the quotient words, by definition, a *dense sequence of the quotient words* is a sequence, where each "quotient word" is encountered exactly once. What is the relation between dense sequences of the quotient words of length q and dense sequences of code words of length q over the basic alphabet \mathcal{A}_s ? Let us examine an arbitrary dense sequence of words $\mathcal{S}(\mathcal{A}_s)$ in the basic alphabet \mathcal{A}_s and let us

compare it with a certain sequence $\mathcal{S}(\mathcal{P})$ of the quotient words as follows. For the convenience let us write down the sequence $\mathcal{S}(\mathcal{A}_s)$ as the following table:

a	b	c	d	e	...
	b	c	d	e	...
		c	d	e	...

(2.9)

which is equivalent to the definition of the dense sequence. Let us build the table of quotient words according to this table. The letter standing in each vertical line of this table, according to Proposition 2.3.4, defines a certain Z -family. Let us arrange the quotient letters that belong to this Z -family also on one vertical line in the table of quotient words. Since each alphabet corresponds to a specific position in the word, the arrangement of the letters of Z -family is defined unambiguously. For example, in the column of letter d in the first quotient word there must stand the letter of Z -family that corresponds to the alphabet A_4 , since the letter d is located at the fourth position in the word $abcde\dots$; in the column of letter d in the second quotient word there must stand the letter of Z -family that corresponds to the alphabet A_3 since the letter d is located at the third position in the word $bcde\dots$ and etc. In order to obtain the complete Z -family of classes for each vertical line we rewrite table (2.9) such that in every row every code word appears twice one after another:

a	b	c	d	e	...	a	b	c	d	e	...
	b	c	d	e	b	c	d	e	...
		c	d	e	c	d	e	...

(2.10)

Instead of referring to the incomplete column of the original part of the table (for example, column of letter 'a') we are referring to the complete column of the new part of the table. However, it is possible, to consider an "open" table as well, selecting any of the possible Z -families for the incomplete columns. As a result we will obtain the following table of the quotient words:

*****...	(2.11)
*****...	
*****...	

where the corresponding quotient letters are denoted by asterisks. Table (2.11) by construction defines a sequence of the quotient words. Let us show that it is a dense sequence, provided the initial sequence of words (2.9) is dense. Indeed, the number of words of length q is identical in both tables. Further, it is obvious that for each word X of table (2.11) there is a corresponding word x from table (2.9). By construction, the letter of the basic alphabet from word x enters into each quotient

letter for word X . Therefore, x belongs to the set defined by the word X , as a Cartesian product, i.e. it belongs to the corresponding synonymous class. Let now two words X and Y from (2.11) coincide. Then words x and y from (2.9) coincide or belong to one synonymous group $X = Y$, which contradicts the assumption of density of sequence (2.9).

We call the described procedure a *lift of a sequence of words of the basic alphabet to a sequence of the quotient words*. It is obvious that different sequences in the basic alphabet can be lifted to the same sequence of quotient words. An opposite operation, the *projection of a sequence of quotient words on sequences of words of the basic alphabet* is defined as follows. Let there be a certain dense sequence of classes $\sigma_1, \dots, \sigma_N$. Let us write it down in the form of table (2.11). Now we replace each Z -family (on the vertical line) by one (one and the same) of the letters of the basis of Z -family. The obtained table of form (2.9) is a sequence of letters of the basic alphabet. The number of words in (2.9) and (2.11) is obviously identical, therefore if (2.9) is not dense, then there exist two words x and y of the same synonymous class. Two quotient words X and Y in table (2.11) correspond to those words, moreover, according to the definition of the dense sequence of quotient words, $X \neq Y$. As it was already shown above, x and y belong to the sets X and Y respectively, whereas X and Y are considered as sets in the appropriate q -dimensional space. However, X and Y do not have common elements since synonymous classes are partitions. Thus, the following Proposition holds.

Proposition 2.4.1. *Each dense sequence of quotient words can be projected on a dense sequence of words of the basic alphabet. This projection, generally speaking, is not unique. Each dense sequence of words of the basic alphabet can be lifted uniquely to a certain dense sequence of quotient words.*

3. Abstract dense sequences

3.1. Words of length two.

Formally, different alphabets already appeared in examples of Cartesian synonymous groups. For example, the collections of sets $\mathbf{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_r\}$ and $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_h\}$ in Example 2.2.2 can be considered as two different alphabets for constructing the quotient words. Letters of the first alphabet can stand only at the beginning of code word, and letters of the second alphabet can stand only at the end of the word. The rules of the construction of sequences of code words in these examples were connected with the origin of letters of the both alphabets, while these alphabets were the subsets of a certain initial set.

Let us consider now abstract sequences, i.e., sequences of code words of abstract alphabets. In this case code words can be combined if the letters coincide in the corresponding positions. We, however, can discuss this in a more general manner. If it is necessary to combine two code words and if the letters in the required positions do not coincide, it is possible to replace both letters by a certain third letter, a "compromise" letter. Other solutions within biological systems are also possible; for example, the possibility to leave both competing letters in one "quasi-position". There are also solutions connected with an addition of signals of another nature in an appropriate position. For future reference only the formal permission for "combination" of these positions filled in different ways, is important. This approach leads to the replacement of the meaningful condition of type (2.8) with a certain formal table of correspondence. Consider the case of two different alphabets $A = \{\mathbf{a}_1, \dots, \mathbf{a}_r\}$ and $B = \{\mathbf{b}_1, \dots, \mathbf{b}_h\}$. Let code words be all words of length two of the form $(\mathbf{a}_i, \mathbf{b}_j)$ $i = 1, \dots, r; j = 1, \dots, h$. That is, code word will be any pair of letters such that its first element (letter) belongs to the alphabet A , and the second one belongs to the alphabet B . To construct a sequence of such words it is necessary to define a table of correspondence of the letters of different alphabets. We define this correspondence using a many-valued mapping φ : set $\varphi(\mathbf{a}_i)$ consists of the letters of the alphabet B , which are compatible with the symbol \mathbf{a}_i . Let us call this mapping a *table of correspondence of symbols*. Note that in the case of Cartesian synonymous partitions, when the quotient letters are subsets of a certain set, the table of correspondence of symbols is defined by the relationship (2.4), i.e. by the non-empty intersection of the corresponding subsets. The table of correspondence of symbols makes possible to form sequences of words of the form

$$\dots (\mathbf{a}_i, \mathbf{b}_j) (\mathbf{a}_s, \mathbf{b}_t) \dots \quad (3.1)$$

if $\mathbf{b}_j \in \varphi(\mathbf{a}_s)$. Recall that in the case of the abstract alphabet the sequence is composed of the words following one another whereas the letters standing next to each other must satisfy the table of correspondence. In this case, a dense sequence is a sequence of words of the form (3.1), where each word is encountered exactly once.

Let us analyze sequences of words over the alphabets A and B with the help

of the following bipartite graph G . The sets of vertices σ^- and σ^+ of this graph correspond to the letters of the alphabets A and B , respectively. Arcs, leaving the vertices of the set σ^- and entering the vertices of the set σ^+ , correspond to the code words. An arc connecting the vertex of the set σ^- associated with a letter $\mathbf{a} \in A$ with the vertex of the set σ^+ associated with a letter $\mathbf{b} \in B$, corresponds to the word (\mathbf{ab}) . We call these arcs *arcs of words*. Let us denote the set of all arcs of words by V . Arcs of another kind connect the vertices of the set σ^+ with the vertices of the set σ^- . These arcs are defined by the table of correspondence of symbols. Namely, if $\mathbf{b} \in \varphi(\mathbf{a})$ then there is an arc leaving the vertex of the set σ^+ , associated with a letter $\mathbf{b} \in B$ and entering the vertex of the set σ^- associated with a letter $\mathbf{a} \in A$. Let us call these arcs *arcs of recovery*. We denote the set of the arcs of recovery by U . Several arcs of recovery can leave each vertex, including none, depending on the table of correspondence.

Example 3.1.1. There are given two different alphabets $A = \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4\}$ and $B = \{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4, \mathbf{b}_5\}$ and a table of correspondence:

$\mathbf{a}_i \in A$	$\varphi(\mathbf{a}_i) \in B$
\mathbf{a}_1	-
\mathbf{a}_2	\mathbf{b}_3
\mathbf{a}_3	$\mathbf{b}_1, \mathbf{b}_2$
\mathbf{a}_4	\mathbf{b}_4

Let us assume that the code words are $\mathbf{a}_1\mathbf{b}_1, \mathbf{a}_2\mathbf{b}_1, \mathbf{a}_3\mathbf{b}_3, \mathbf{a}_3\mathbf{b}_4, \mathbf{a}_3\mathbf{b}_5$ and $\mathbf{a}_4\mathbf{b}_2$. Therefore, not all arcs of words should be drawn in the corresponding bipartite graph. Figure 7 illustrates the corresponding bipartite graph G , where thin arrows depict arcs of words, and thick arrows depict arcs of recovery.

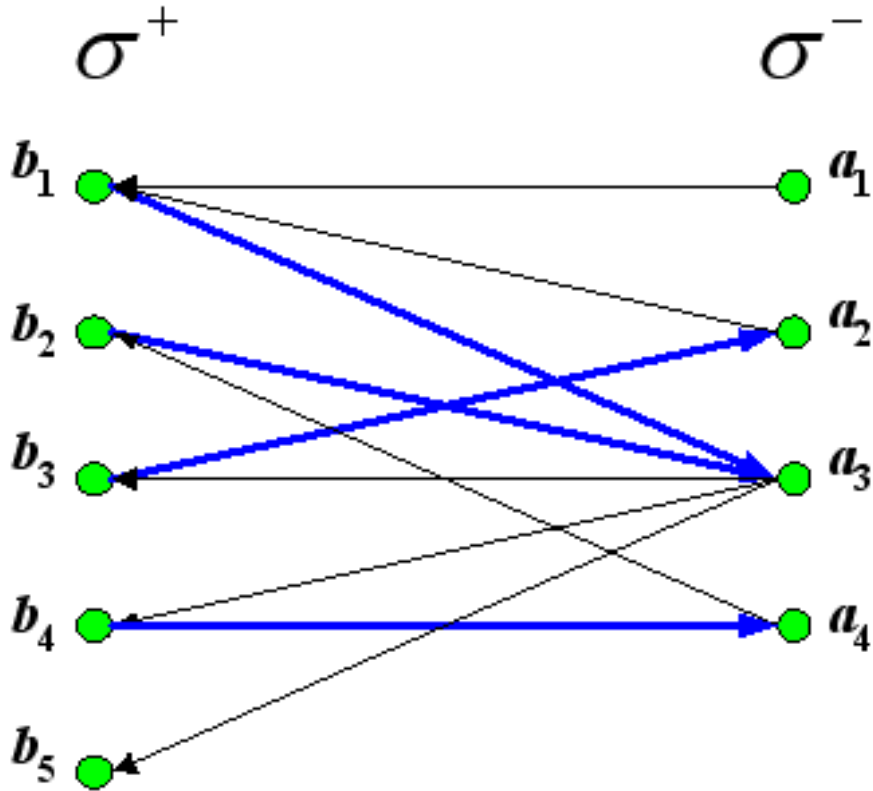
In contrast to the standard De Bruijn graph, arcs of recovery do not correspond to words, and they, in some sense, are intermediate steps, which are represented neither by letters nor by words in the formed sequence. Therefore, it is possible either to traverse the arc of recovery several times or not to traverse it at all, i.e., these arcs (multi-arcs) can have an arbitrary non-negative multiplicity. The table of correspondence of symbols defines the set U of the arcs of recovery; however, the multiplicities of these arcs can be assigned in several possible ways. Using table of correspondence given in Example 3.1.1 we can form the following sequence of words over the given alphabets A and B :

$$(\mathbf{a}_1, \mathbf{b}_1)(\mathbf{a}_3, \mathbf{b}_3)(\mathbf{a}_2, \mathbf{b}_1)(\mathbf{a}_3, \mathbf{b}_4)(\mathbf{a}_4, \mathbf{b}_2).$$

Note that in construction of the above sequence the arc of recovery $\mathbf{b}_1\mathbf{a}_3$ was traversed twice and the arc of recovery $\mathbf{b}_2\mathbf{a}_3$ was not traversed at all.

By definition an *Euler path* is a path that uses each arc once, and only once. Since in our case each arc has a certain multiplicity, we formulate the following

Fig. 7. textitThe bipartite graph G in Example 3.1.1.



generalization of the Euler path: a *generalized Euler path* is a path that uses each arc as many times as prescribed by its multiplicity. Similar conditions are required for the existence of our generalized Euler path as for the existence of the ordinary Euler path. Thus the task of finding a generalized Euler path with a given multiplicities is equivalent to the task of finding an Euler path, and the task of finding a generalized Euler cycle with a given multiplicities is equivalent to the task of finding an Euler cycle in the graph G . It follows from Good's Theorem (Section 1.4) that for the existence of a generalized Euler cycle it is necessary and sufficient that the sum of the multiplicities of the entering arcs is equal to the sum of the multiplicities of the leaving arcs for all vertices of the graph G . This fact reduces the problem of construction of a dense sequence to the problem of definition of the multiplicities of arcs of recovery with the given sets V and U , so that the graph G satisfies the Eulerian condition (Section 1.4).

Let us investigate the problem of construction of a dense sequence that contains all possible words over the alphabets A and B exactly once using graph G . Let the

alphabets A and B have the same cardinality ($r = h$). In this case d arcs leave each vertex of the set σ^- . Let $Wg \subseteq \sigma^-$. Let us denote the set of vertices in $\{\sigma^+\}$ adjacent to W via arcs of recovery by $S(W)$.

Theorem 3.1.1. *Let $d = |\sigma^-| = |\sigma^+|$. For the existence of a cyclic sequence of words including each word exactly once it is necessary and sufficient that for any set $Wg \subseteq \sigma^-$ the following inequality holds:*

$$|S(W)| \geq |W|$$

Proof. Necessity. Since all arcs of words are drawn in this graph, each vertex of the set σ^- has the out-degree equal to d . For the same reason, each vertex of the set σ^+ has the in-degree also equal to d . Let us examine an arbitrary set $Wg \subseteq \sigma^-$. In order that the out-degrees and the in-degrees of all vertices of this set would be equal, it is necessary that the sum of multiplicities of the arcs of recovery of the set $S(W)$ incident to W would be equal $|W|d$. However, the vertices of the set σ^+ also must have equal out-degrees and in-degrees, therefore the multiplicity of the arc of recovery is limited by value d . (Multiplicity of the arc of recovery can be less than d , if there is more than one arc of recovery leaving that vertex.) Therefore, the sum of multiplicities of the arcs of recovery does not exceed the value of $|S(W)|d$, hence it is necessary that $|S(W)|d \geq |W|d$.

Sufficiency. Conditions of this theorem coincide with the conditions of the Hall's theorem [4] of existence of perfect matching in the set of arcs U , i.e. matching, containing all vertices of graph. Let P be such a matching. Let us assume that the multiplicities of all arcs not in P equal zero, and the multiplicities of arcs in P equal d . It is obvious that in this case the out-degrees and the in-degrees of all vertices in G are equal. Therefore, an Euler cycle exists. The Theorem is proven. \square

Note that the matching mentioned in the proof of Theorem 3.1.1 defines a one-to-one correspondence between letters of both alphabets. In this case it is possible to return to the initial situation when the first and the second letters of each word belong to the same alphabet. In particular, it is possible to identify the corresponding vertices of the graph G from σ^- and σ^+ and thus to obtain the standard De Bruijn graph.

Consider now the general case, when the alphabets A and B are of different cardinality ($r \neq h$) and the set of the code words M does not necessarily coincide with the set of all possible words. It means that the sets of left and right vertices have different size, $|\sigma^-| \neq |\sigma^+|$, and not all arcs of words are drawn in the graph G . In this case let us denote the out-degree of vertex $i \in \sigma^-$ by s_i and the in-degree of vertex $i \in \sigma^+$ by t_i . It is obvious that

$$\sum s_i = \sum t_i. \tag{3.2}$$

Let us define the *generalized out-degree* of the vertex as the number of arcs leaving it, each taken with its multiplicity and the *generalized in-degree* of the vertex as the number of arcs entering it, each taken with its multiplicity. Let us

denote an arbitrary subset of vertices of graph G by X and the sum of generalized out-degrees of these vertices by $P(X)$. Let us denote, as earlier, the set of vertices in σ^- by W and the set of vertices in σ^+ that are adjacent to W via arcs of recovery, by $S(W)$.

Theorem 3.1.2. *For the existence of a cyclic sequence of words including each word from the set M exactly once it is necessary and sufficient that the graph G is connected, and for any set $Wg \subseteq \sigma^-$ the following inequality holds:*

$$P(S(W)) \geq P(W). \quad (3.3)$$

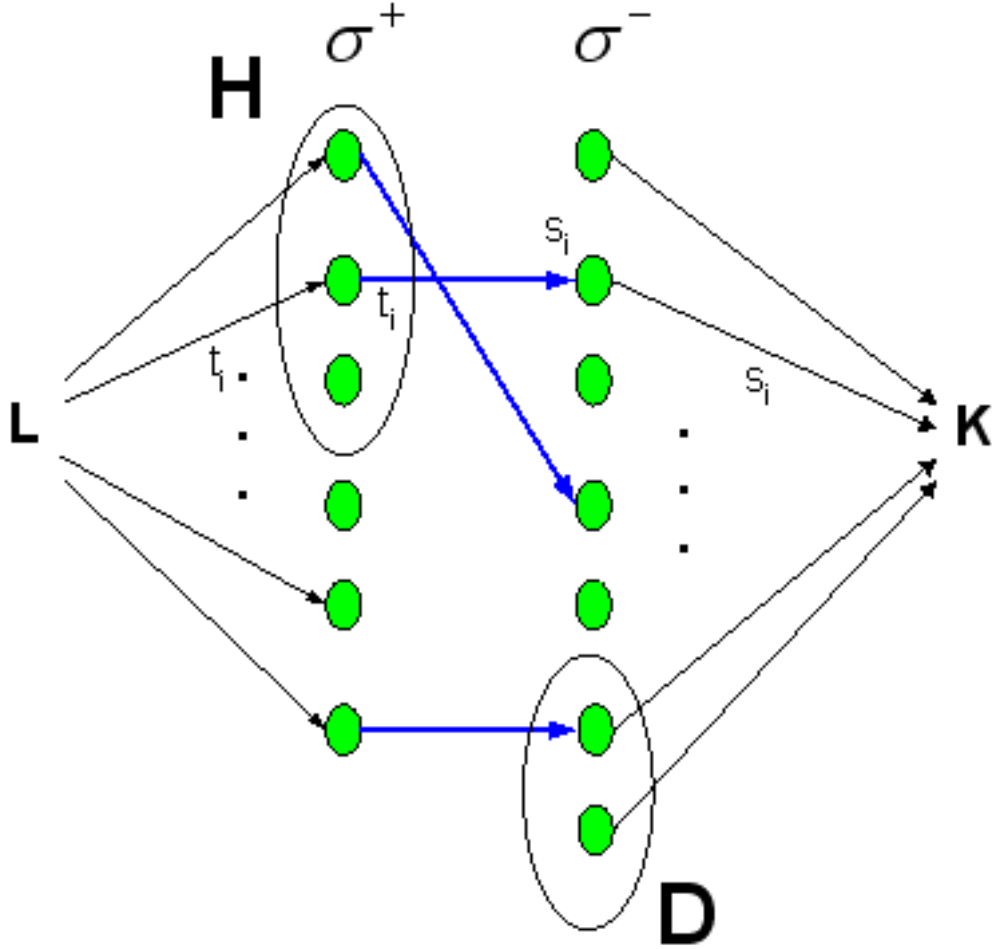
Proof. Necessity. Indeed, some of the arcs of recovery leaving the set $S(W)$, possibly all of them, enter the vertices of the set W , but no other arc of recovery enters this set according to the definition of the set $S(W)$. If graph G is a generalized Euler graph, i.e. each vertex has equal generalized out-degree and generalized in-degree, then the following inequality holds: $P(S(W)) \geq P(W)$.

Sufficiency. It is necessary to show that if (3.3) holds then there exist non-negative integral values on the arcs of recovery (multiplicities) such that the generalized out-degree and the generalized in-degree at each vertex of the graph G are equal. We apply the theory of flows in networks. Let us assume that for every i the *capacity of vertex i from the set σ^-* is equal to s_i (out-degree) and the *capacity of vertex i from the set σ^+* is equal to t_i (in-degree). For convenience, let us add to the graph G two new vertices in the following way. Let us join all vertices of the set σ^- to a new vertex \mathbf{K} by arcs directed to vertex \mathbf{K} . Let us join a new vertex \mathbf{L} to all vertices of the set σ^+ by arcs directed from vertex \mathbf{L} . In the theory of network flows these vertices are conventionally called a *sink* and a *source* respectively. We will further examine only the arcs of recovery (set U) on the graph G discarding the arcs of words from the graph G , and denote the graph obtained in this way by G' . By construction, graph G' is oriented from the vertex \mathbf{L} , the source, to the vertex \mathbf{K} , the sink (see Figure 8).

In the network flow problem it is required to find a non-negative function on the arcs of the graph (called *the maximum flow*), possessing specific properties. By definition, the sum of the values of the flow function on all arcs incident to the sink is called the *flow value*. It is required that: (1) for each vertex, except for the sink and the source, the total value of the flow on all arcs entering the vertex and the total value on all arcs leaving the vertex must be equal to each other and must not exceed the capacity of the vertex, (2) it is not possible to increase the flow value without destroying condition (1). A vertex of graph is *saturated*, if the total value of the flow on entering (and, consequently, on leaving) arcs is equal to its capacity.

Let us show that if there is a flow of value $Q = \sum s_i$ in the graph G' , then all vertices of the sets σ^+ and σ^- are saturated. Indeed, according to the definition of the flow value, Q is equal to the sum of flows on all the arcs going from each vertex $i \in \sigma^-$ to the vertex \mathbf{K} . However, the flow for each such arc does not exceed the capacity s_i of the corresponding vertex according to the property (1) of the flow.

Fig. 8. Graph G' , oriented from the vertex L , the source, to the vertex K , the sink



Consequently, the sum of all such flows does not exceed Q and is equal to Q only when the flow on arc is equal to the capacity of the vertex. When the flow value is Q , all the vertices of the set σ^- are saturated. Further, for any vertex $i \in \sigma^-$ we will denote the sum of the flows on all arcs entering such a vertex by S_i . According to the property (1) of the flow, S_i is equal to the sum of the flows on all leaving arcs, i.e. $S_i = s_i$. Let us denote now the sum of flows on all arcs leaving any vertex $i \in \sigma^+$ by T_i . According to the definition of the flow, $T_i \leq t_i$, i.e. $\sum T_i \leq \sum t_i = Q$ (see 3.2). On the other hand, it is clear that $\sum T_i = \sum S_i = Q$, since both sums are taken on the same set of arcs. From this follows that $T_i = t_i$ for any vertex $i \in \sigma^+$. Therefore, when the flow value is Q , all vertices of the set σ^+ are also saturated. It remains to assign flow t_i to each arc leaving vertex L and entering vertex $i \in \sigma^+$

of graph G' . In this case all the conditions of the flow are maintained.

We assume now that the value of flow Q not only exists, but also is integral, i.e. the values of the functions on the arcs of the graph are integral non-negative numbers. In this case for every arc of recovery its multiplicity must be defined as the value of flow on this arc. Since all vertices of the set σ^+ in graph G' are saturated, every vertex of the set σ^+ in the initial graph G has equal generalized out-degree and in-degree (number of arcs of words). The same is true for vertices of the set σ^- in the initial graph G . As it was proven earlier, the values of multiplicities are equal to the capacities of vertices, i.e. to the out-degrees for vertices of the set σ^- and to the in-degrees for vertices of the set σ^+ .

With such multiplicities of the arcs of recovery the initial graph G is an Euler graph. It remains to prove that with the theorem condition the flow value Q exists and it is integral.

We call a set of vertices a *vertex-cut* (or just a *cut*) of the graph G' if it does not contain the sink and the source, and any path from \mathbf{L} into \mathbf{K} contains at least one vertex of this set. Let us call the sum of the capacities of the cut's vertices the *weight* of the cut. The cut is called *minimal* if it has the smallest weight. According to the Ford-Fulkerson theory [7], the value of the maximum flow is equal to the value of the minimal cut. Now we determine the value of the minimal cut in graph G' . In graph G' there are two obvious cuts: the entire set of vertices σ^- and the entire set of vertices σ^+ . Both these cuts have an equal weight $P(\sigma^-) = P(\sigma^+) = Q$ (see equality (3.2)). Let us demonstrate that the cuts σ^- and σ^+ are minimal. That is, the weight of any other cut is greater than or equal to Q . Consider a cut (D, H) , where $D \subset \sigma^-$, $H \subset \sigma^+$ respectively; its weight is equal to $P(D) + P(H)$. $S(\sigma^- \setminus D)$ is a set of vertices in σ^+ that are adjacent to the vertices of $\sigma^- \setminus D$ via arcs of recovery and $S(\sigma^- \setminus D) \cap (\sigma^+ \setminus H) = \emptyset$. Indeed, otherwise there would be an arc from the vertex of the set $(\sigma^+ \setminus H)$, not included in the cut, into the vertex of the set $\sigma^- \setminus D$, also not included in the cut. This contradicts the definition of the cut. In other words, $S(\sigma^- \setminus D) \subset H$. Hence, taking into account the condition of the theorem $P(S(\sigma^- \setminus D)) \geq P(\sigma^- \setminus D)$, the following inequality holds:

$$P(D) + P(H) \geq P(D) + P(S(\sigma^- \setminus D)) \geq P(D) + P(\sigma^- \setminus D) = P(\sigma^-) = Q.$$

Thus, the weight of the cut $P(D) + P(H)$ is not less than the value of Q . Therefore, the value of the maximum flow on graph G' is equal to Q .

The integrality of this flow easily follows from the standard algorithm of the construction of the maximum flow [7], which results in integral flow if the capacities of the vertices are integral numbers. The Theorem is proven. \square

Corollary 3.1.3. *Assume that in the graph G all arcs of words are drawn, and $r \neq h$. Then for the existence of a cyclic sequence of words that includes each word exactly once it is necessary and sufficient that for any set $Wg \subseteq \sigma^-$ the following inequality holds:*

$$|(Wg) \cap \sigma^+| \geq |Wg| - |\sigma^-|. \quad (3.4)$$

Proof. Indeed, the in-degree of every vertex of the set σ^+ is equal to $|\sigma^-|$, and the out-degree of every vertex of the set σ^- is equal to $|\sigma^+|$. Therefore, inequality (3.3) can be re-written as

$$|\sigma^-||S(W)| \geq |W||\sigma^+|.$$

From there follows (3.4). \square

Theorems 3.1.1 and 3.1.2 guarantee the existence of the corresponding distribution of the multiplicities of arcs that makes graph G a generalized Euler graph. But this distribution can be not unique. We will call all such distributions *compatible with an Eulerian property*. Selection of two possible distributions of the multiplicities of the arcs of recovery compatible with an Eulerian property does not change the set of the words of a dense sequence. But this substantially influences "joints" between the words. In particular, consider the sequence

$$\dots(\mathbf{a}_i, \mathbf{b}_j)(\mathbf{a}_s, \mathbf{b}_t)\dots$$

Let us call the pair $(\mathbf{b}_j, \mathbf{a}_s)$ a *joint between the words* $(\mathbf{a}_i, \mathbf{b}_j)$ and $(\mathbf{a}_s, \mathbf{b}_t)$; this joint is a word in the "inverted" system of alphabets, that is, the first letter belongs to the second alphabet and the second letter belongs to the first alphabet. We call a set of all such words the *spectrum of the sequence*. The joint $(\mathbf{b}_j, \mathbf{a}_s)$ corresponds to the arc of recovery between the vertex $\mathbf{b}_j \in \sigma^+$ and the vertex $\mathbf{a}_s \in \sigma^-$. The number of such words in the spectrum of the sequence is equal to the multiplicity of this arc. Thus, the spectrum of a dense sequence is different for the different distributions of the multiplicities of the arcs of recovery and completely defines the corresponding distribution of the multiplicities, except for the arcs of recovery with zero multiplicity. That is, the spectrum does not determine the table of correspondence but only the subset of the arcs of recovery that have non zero multiplicities in the given distribution of multiplicities compatible with an Eulerian property.

Consider two different spectra of a dense sequence created from the same set of words. The number of elements in the spectrum obviously depends only on the number of words included in the sequence: in the sequence consisting of N words there are $N-1$ joints. Therefore, if in one of the spectra one of the joints has a multiplicity less than in another spectrum by one, then there must exist one additional element of the spectrum, which is also different in these spectra. Thus, it follows from the balance of the number of the elements of the spectrum that the spectra must differ not less than by two elements. Actually this difference is greater.

For our convenience let us define the *weight of vertex* $i \in \sigma^-$ as its out-degree and a *weight of vertex* $i \in \sigma^+$ as its in-degree.

Theorem 3.1.4. *Any two different spectra of a dense sequence created from the same set of words differ on the set that has not less than four elements.*

Proof. Let us examine the subgraph G^* of the graph G , in which only the arcs of recovery are left, and the multiplicities of the arcs of words are substituted with

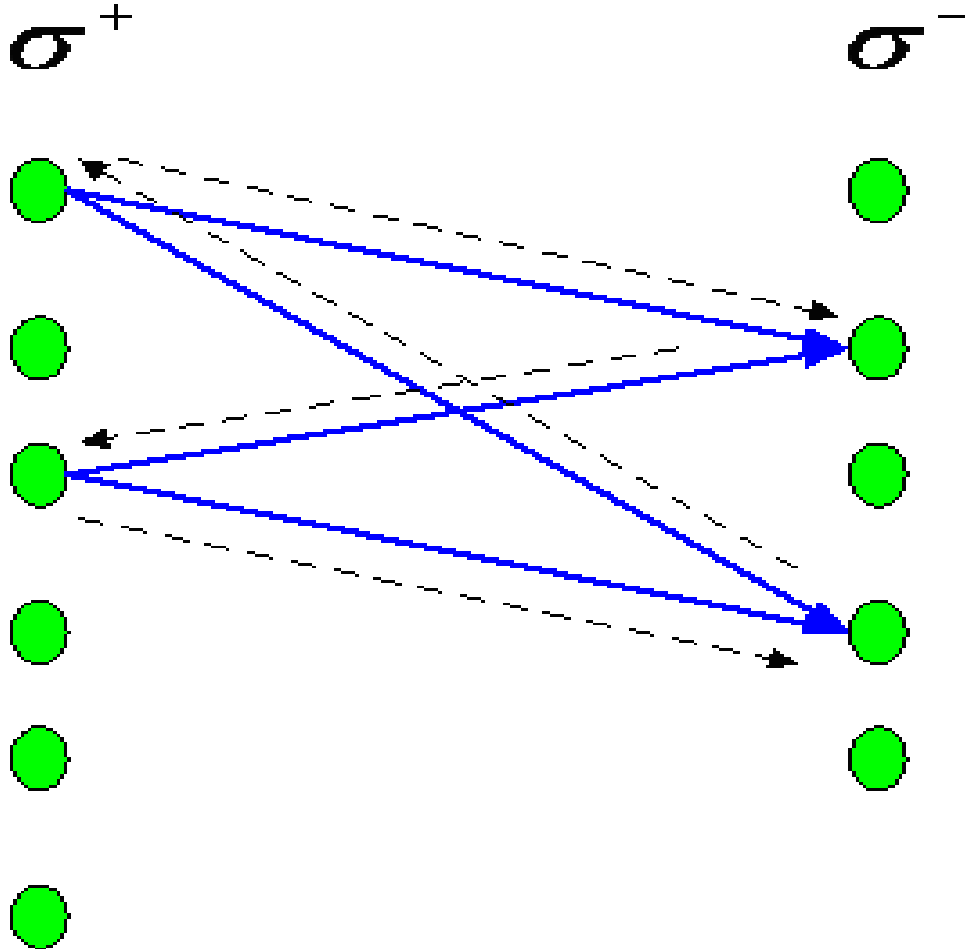
the corresponding *weights* of the vertices: the *weight of vertex* $i \in \sigma^-$ is its out-degree and a *weight of vertex* $i \in \sigma^+$ is its in-degree (by definition of the graph G , these values are determined by the arcs of words). Let us assume that we have two such subgraphs, G_1^* and G_2^* , with distributions of the multiplicities of the arcs of recovery; these distributions are different but compatible with an Eulerian property. Let us define the residual graph $G_0^* = G_1^* - G_2^*$ as a graph, in which the weight of every vertex is equal to the difference of the weights of the corresponding vertices and the multiplicity of every arc is equal to the difference of the multiplicities of the corresponding arcs of the graphs G_1^* and G_2^* . By construction, in the graph G_0^* the weights of all the vertices are equal to zero and the multiplicities of all the arcs, on which the distributions coincide, are also equal to zero. In this graph the multiplicities of the remaining arcs have positive or negative values, but since the graphs G_1^* and G_2^* are compatible with an Eulerian property, the generalized out-degree of every vertex $i \in \sigma^+$ is equal to zero and the generalized in-degree of every vertex $i \in \sigma^-$ is equal to zero. Consider a subgraph H , which consists of the arcs with non zero multiplicities and vertices incident to them. We demand that for every vertex in subgraph H its out-degree or in-degree is equal to zero. Therefore, it is necessary that there would be at least two arcs incident to it. Thus, there are no "hanging" vertices in the graph H , therefore this graph contains a contour, if we disregard the directivity of its arcs. But, as it can easily be seen, the minimum contour on the set of the arcs of recovery has a length four (see Figure 9). The Theorem is proven. \square

The proof of the previous Theorem prompts a general approach to finding different distributions of the multiplicities of the arcs of recovery compatible with an Eulerian property. But the question whether we can find all such distributions using this approach remains open. Let us examine an arbitrary contour created on the arcs of recovery if such exists, without taking into account their directions. According to the property of the arcs of recovery, the contour must have an even length. Let there be a certain distribution of the multiplicities of the arcs of recovery. Starting with an arbitrary arc of the contour, we will add and subtract the same integer number which is not greater than the minimal multiplicity of the arcs in this contour from the multiplicities of arcs. It is easy to see that the weights of the vertices that are incident to this contour will not change, i.e., they will remain equal to zero; the multiplicities of the arcs will change but they will remain integral non-negative values.

3.2. *Words of an arbitrary length.*

In the previous Section we examined the words of length two over the abstract alphabets. Thus, we generalized constructions, carried out in Section 2.2 for the case of words of length two with Cartesian synonymous partitions. In this Section we will do the same for the words of an arbitrary length, resting in this case on the constructions introduced in Section 2.3. In a contrast to the scheme given

Fig. 9. The minimum contour on the set of the arcs of recovery



in the mentioned section, we will introduce q different alphabets. Namely, let us assume there are alphabets $A_i = \{\mathbf{a}_{1,i}, \dots, \mathbf{a}_{r_i,i}\}$, $i = 1, \dots, q$, and the code word is of the form $\mathbf{a}_{j_1,1} \mathbf{a}_{j_2,2} \dots \mathbf{a}_{j_q,q}$, where at place s stands any element of the set A_s . The dense arrangement of these words into the sequence requires the definition of the rules of correspondence that are more complex than in the case of two-letter words. Indeed, in the case of two-letter words it is sufficient to indicate the table of correspondence at the level of the pairs of letters. Recall that this table generated the arcs of recovery in the graph G . However, as it can be seen already in Section 2.3, with the words of length q , it is necessary to define correspondence of q letters, one from each alphabet. If the table of correspondence at the level of the pairs of letters was transitive, then it was not required to introduce more complex rules.

However, this is not true for the Cartesian synonymous partitions that are our basic non-formal model. Therefore, it is also necessary to avoid using this property in the general case. Let us call a certain set of Z -families $Z = \{Z\}$ *permissible* if all letters belonging to one Z -family are compatible in the sequence in the sense of constructions presented in Section 2.3. It is obvious that the existence of a dense sequence of code words depends on the set Z : if it is empty then there is no dense sequence; if the set Z includes all possible Z -families then any sequence that includes each code word exactly once is a dense sequence. By analogy with the two-letter case, we call each Z -family from the set Z a *joint between the code words*. Note also that each Z -family actually is a word in the corresponding alphabets A_i , $i = 1, \dots, q$. Indeed, since any letter of each alphabet can stand in the word only in one position defined by this alphabet, Z -family can be considered as a corresponding word. Thus, two Z -families of words originated: Z -family of the code words and Z -family of the joints. If we write the sequence of code words in such a way that the overlapping parts of the words are in the same column, then we obtain the following table:

```

***...***
****...***
***...***
****...***

```

There are q symbols in each line and there are q symbols in each complete column of this table. Code words go from left to right horizontally and joints go vertically from bottom to top: the letter of the first alphabet of joint words is located at the bottom and all others are also arranged from bottom to top. Letters in the positions of the intersection of code words and joints must coincide. For example, it is possible to fill the table placing code words horizontally; otherwise, this can be done by placing words of joints vertically. Let us note that for simplicity we speak here about the “open” table (see Section 2.4) and do not worry about the “end effect” (incomplete columns), when joint has a number of letters that is less than q .

It is possible to apply the bipartite graph G used earlier for the two-letter words to the problem of construction of a dense sequence in the case of words of length q . For this purpose it is necessary to represent each word of length q as two words $\mathbf{a}_{j_1,1} \mathbf{a}_{j_2,2} \dots \mathbf{a}_{j_{q-1},q-1}$ and $\mathbf{a}_{j_2,2} \dots \mathbf{a}_{j_q,q}$ of length $q-1$ each: the former $q-1$ symbols represent a word that corresponds to one of the vertices of the set σ^- , the latter $q-1$ symbols represent the second word that corresponds to one of the vertices of the set σ^+ . The arc of recovery is drawn from the vertex of the set σ^+ into the vertex of the set σ^- if in each of $q-1$ positions in both words there are letters that belong to at least one Z -family from Z . It is obvious that the last condition is necessary for an Euler cycle (or Euler path) in this graph to produce a dense sequence. Also the presence of an Euler cycle (or Euler path) is the necessary condition for the existence of a dense sequence. Therefore, the necessary and sufficient conditions given in Section 3.1 here are just necessary. Nevertheless,

using the described graph G , it is possible to generate sequences that contain each code word only once, and to check them against the permissibility of the joints.

3.3. Combinatorial connectivity of the set of dense sequences.

We will characterize mutual arrangement of the code words in a dense sequence by the possibility to pass from one dense sequence to another using a certain "transposition" operation. Namely, an *elementary transposition* consists of the transposition of two nonintersecting fragments in this sequence in such a way that the obtained sequence is a dense sequence as well. The number of components connected in this sense is a characteristic of the structure of the set of dense sequences. We will call this characteristic the *combinatorial connectivity*. Many dense sequences of letters are connected as we can learn from [8], [9]. This is not always true for the set of dense sequences of the quotient letters.

Example 3.3.1. Let $\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$ and $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$ be two Cartesian partitions of the set $\mathcal{A}_6 = \{a_1, a_2, a_3, a_4, a_5, a_6\}$, where $\mathbf{a}_1 = \{a_1, a_2\}$, $\mathbf{a}_2 = \{a_3, a_4\}$, $\mathbf{a}_3 = \{a_5, a_6\}$, $\mathbf{b}_1 = \{a_1, a_3\}$, $\mathbf{b}_2 = \{a_4, a_5\}$, $\mathbf{b}_3 = \{a_1, a_6\}$. Let us build a bipartite graph G that consists of two sets of vertices σ^- and σ^+ . σ^- corresponds to the partition \mathbf{A} and σ^+ corresponds to the partition \mathbf{B} . In our example all words are code words, that is, there are three

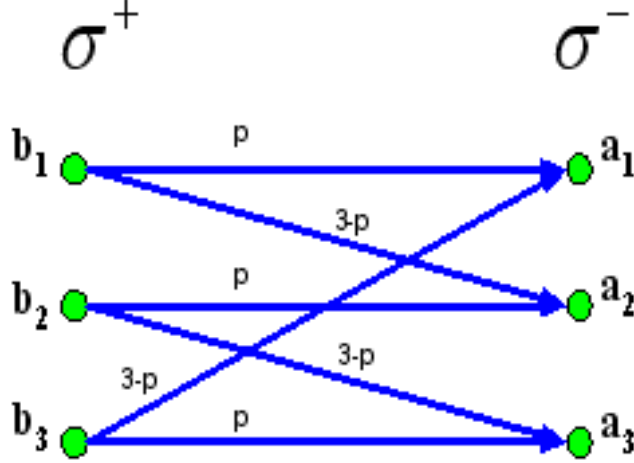
arcs leaving from each vertex of the set σ^- and there are three arcs entering each vertex of the set σ^+ . For our purposes we draw only arcs of recovery. Recall that in the case of Cartesian partitions the arc of recovery occurs from vertex \mathbf{b}_j into vertex \mathbf{a}_i if the intersection of the corresponding sets is not empty $\mathbf{a}_i \cap \mathbf{b}_j \neq \emptyset$. Therefore, there are exactly two arcs of recovery leaving each vertex of the set there are exactly two arcs of recovery leaving each vertex of the set σ^+ (see Figure 10).

It is necessary for existence of a dense sequence that the sum of multiplicities of the arcs of recovery leaving each vertex is equal to three. Let the multiplicity of the arc $(\mathbf{b}_1 \mathbf{a}_1)$ be equal \mathbf{p} , where $0 \leq \mathbf{p} \leq 3$, then the multiplicity of the arc $(\mathbf{b}_1 \mathbf{a}_2)$ is equal to $3-\mathbf{p}$ and since the in-degree of the vertex \mathbf{a}_2 is equal to 3, the multiplicity of the arc $(\mathbf{b}_2 \mathbf{a}_2)$ is equal to \mathbf{p} and so on (see Figure 10). Therefore, there are four different distributions of multiplicities of the arcs of recovery:

0	3	0	3	0	3
1	2	1	2	1	2
2	1	2	1	2	1
3	0	3	0	3	0

and thus there are four different spectra. There are six differences between any two spectra. Since one elementary transposition may change four joints at most, it is impossible to transform one dense sequence into another by one elementary transposition.

Fig. 10. The bipartite graph G with the arcs of recovery



Let $A = \{a_1, \dots, a_s\}$ and $B = \{b_1, \dots, b_h\}$ be two abstract alphabets. Consider two dense sequences

$$\begin{aligned} & (a_u, b_v)(a_i, b_j) \dots (a_r, b_r), \\ & (a'_u, b'_v)(a'_i, b'_j) \dots (a'_r, b'_r) \end{aligned} \tag{3.5}$$

containing the identical set of code words, each of them of length two. Let us remove brackets in these sequences and let us call the obtained sequences *free sequences*. By construction, a free sequence consists of two-letter words overlapping with the shift of 1, among which the code words of form ab alternate with non code words of form ba .

Theorem 3.3.1. *The set of all dense sequences with an identical spectrum is combinatorially connected.*

Proof. Thus we have to prove that any two dense sequences with an identical spectrum can be transformed from one into another by elementary transpositions; moreover, all intermediate sequences have the same spectrum. Let w_1 and w_2 be two such sequences. Let these sequences coincide to some position, for example, the one where letter T is located; and they further continue with letters X and Y respectively (3.6)

$$\begin{aligned} w_1 & : A \dots TX \dots Y \dots \\ w_2 & : A \dots TY \dots X \dots \end{aligned} \tag{3.6}$$

Since the spectra of the both sequences coincide, there must be the word TX in the continuation of the sequence w_2 , and there must be the word TY in the continuation

of the sequence w_1 . Consider the case when the word TX is at the end of the sequence w_2 :

$$\begin{aligned} w_1 &: \mathbf{A} \dots \mathbf{TXN} \dots \mathbf{Y} \dots \\ w_2 &: \mathbf{A} \dots \mathbf{TY} \dots \mathbf{N} \dots \mathbf{X}. \end{aligned} \tag{3.7}$$

In sequences (3.7) N denotes the letter standing after X in the sequence w_1 ; by the definition, it occurs also in the continuation of the sequence w_2 . Then it is possible to transpose two sub-words $[Y \dots]$ and (X) in the sequence w_2 shown at the following scheme:

$$w_2 : \mathbf{A} \dots \mathbf{T[Y \dots]N} \dots (\mathbf{X}) \tag{3.8}$$

The sub-word X can stand between the letters T and N , as it can be seen from w_1 (3.7). Letter Y can stand after the letter T , as it can be seen from (3.8). Therefore, it is possible to transform (3.8) to:

$$w_2 : \mathbf{A} \dots \mathbf{TXN} \dots \mathbf{Y} \dots \tag{3.9}$$

Taking the word w_1 from (3.7) we finally obtain:

$$\begin{aligned} w_1 &: \mathbf{A} \dots \mathbf{TX} \dots \dots \\ w_2 &: \mathbf{A} \dots \mathbf{TX} \dots \dots \end{aligned} \tag{3.10}$$

i.e. now there is one more coinciding position. Thus, we conducted an inductive step with the assumptions made regarding the position of the word TX in the sequence w_2 . It is now necessary to note that if the word TX in (3.6) was originally a joint word, then the word TY in (3.6) is also a joint word. Therefore, the spectrum did not change in sequences (3.10) comparing to (3.6). The same would be true if words TX and TY would be code words.

Let us return now to (3.6) and let the word TX stand in an arbitrary position but not at the end of the word, as shown in (3.11):

$$\begin{aligned} w_1 &: \mathbf{A} \dots \mathbf{TX} \dots \mathbf{Y} \dots \\ w_2 &: \mathbf{A} \dots \mathbf{TY} \dots \mathbf{X} \dots \end{aligned} \tag{3.11}$$

Lemma 3.3.2. *Assume that it is impossible to make transposition in the sequences (3.11) that moves the point of coincidence by one step. Then no letter standing in the sequence w_1 in the range $TX \dots Y$ can stand in the range $TY \dots$. Similarly, the set of letters in the range $TY \dots X$ in the sequence w_2 has an empty intersection with the set of letters in the range $X \dots$.*

Proof. Indeed, if

$$w_1 : \mathbf{A} \dots \mathbf{TX} \dots \mathbf{N} \dots \mathbf{Y} \dots \mathbf{N} \dots$$

then

$$w_1 : A...T[X...]N...(Y...)N...$$

and it is possible to swap the word [...] with the word (...). Then we will move by one letter in the sequence w_1 (3.11). The same is true for the sequence w_2 . Thus, Lemma 3.3.2 is proven. \square

Assume now that there is no such transposition that increases coinciding interval by one letter. Then, according to Lemma 3.3.2, the set of letters of the united alphabets $W = W_1 \cup W_2$ is partitioned into two parts: W_1 that is located in the sequence w_1 between the words TX and TY , and W_2 that is located after the word Y ; moreover, $W_1 \cap W_2 = \emptyset$. Let us schematically write this down as

$$w_1 : A...TX\{W_1\}Y\{W_2\}.$$

It is obvious that the word w_2 takes the form

$$w_2 : A...TY\{W_2\}X\{W_1\}.$$

Indeed, it is clear that the partition of the set W for the sequence w_2 also must be the partition into W_1 and W_2 , since spectra of both words are equal. Furthermore, there is a certain letter from the set W_1 in the sequence w_1 following the letter X . Respectively, the same two-letter word must be found in the sequence w_2 . Therefore, the set W_1 must stand after the letter X . Further, the letter T forms a two-letter word in the sequences w_1 and w_2 with the sets W_1 and W_2 respectively: N_1T where $N_1 \in W_1$, and N_2T where $N_2 \in W_2$. These are two different words, since, by the hypothesis, $W_1 \cap W_2 = \emptyset$. According to the coincidence of spectra, the word N_2T from w_2 must be present in w_1 . Since the letter N_2 is from the set W_2 , it can occur after TY . But then the letter T also belongs to the set W_2 . By analogy, considering the word N_1T from w_1 , we will come to the conclusion that it is located after the word TX in the sequence w_2 and the letter T belongs to the set W_1 . Thus, both sets W_1 and W_2 have a common letter T and this contradicts to the assumption that $W_1 \cap W_2 = \emptyset$. Thus, it is always possible to advance by one step and the Theorem is proven. \square

If we examine restricted dense sequences, i.e. we write them in terms of the words

$$\begin{aligned} w_1 &= \sigma_1, \dots, \sigma_r, \\ w_2 &= s_1, \dots, s_r, \end{aligned} \tag{3.12}$$

then by an elementary transposition we should understand a transposition of the entire word. This version of transpositions is a special case of the previous one and, formally speaking, gives us fewer possibilities. However, if the code words σ and s are of length two, Theorem 3.3.1 holds true for this case as well. Indeed, the words σ and s in this case can be considered as letters. Then the proof of Theorem 3.3.1 applies literally to this case.

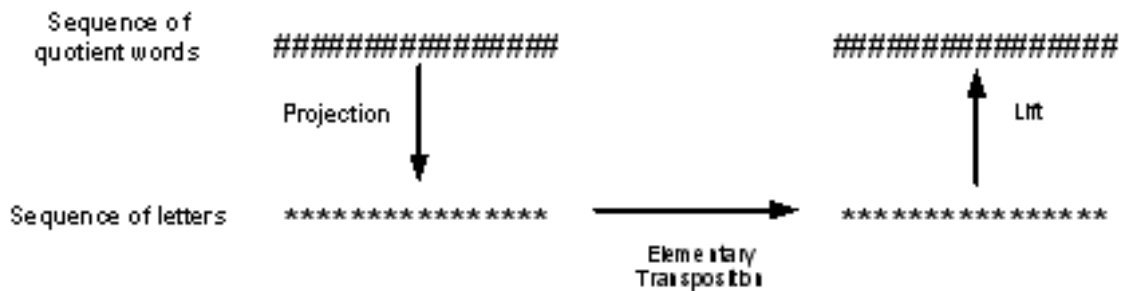
As it was noted above, the spectrum of a sequence is equivalent to the distribution of the multiplicities of the arcs of recovery and completely determines the corresponding distribution of multiplicities, except for the arcs of recovery of zero multiplicity. Since selection of distribution of multiplicities of the arcs of recovery is frequently ambiguous, there exists a natural partition of the set of all dense sequences into the subsets with an identical spectrum. Every such subset is combinatorially connected according to Theorem 3.3.1. These subsets can be united into combinatorially connected components. However if the sets of joints of two dense sequences differ significantly these two dense sequences can not be transformed from one into another by one elementary transposition, since one elementary transposition may change four joints at most. Therefore, for existence of combinatorial connectivity there must exist a sequence of elementary transpositions that generates a dense sequence at every step with the spectrum that differs from the spectrum of the previous sequence. The existence of such a sequence of elementary transpositions depends on the table of correspondence.

4. Application of the general theory to the case of a Cartesian partition

4.1. Euler cycles of Cartesian sequences and sequences of letters.

In Section 3.3 we observed the "combinatorial disconnectedness" of the sequences of quotient words, in contrast to the sequences of letters. The existence of this property does not raise a question in the case of the formal scheme of the abstract alphabets. However, for Cartesian partitions there is a close connection between the sequences of letters and the sequences of quotient words. The connection (lift and projection) between the dense sequences of quotient words and the dense sequences of words over the basic alphabet is shown in Section 2.4. The following diagram (Figure 11) illustrates the corresponding relations.

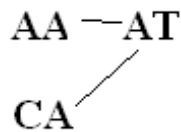
Fig. 11. Illustration of relations between dense sequences of quotient words and dense sequences of words over the basic alphabet



Nevertheless, any two sequences of the letters for an identical set of code words can be transformed into each other by transpositions of fragments. This is not always true for the sequences of quotient words, hence the diagram on Figure 11 is not closed. The following example explains the reason. Consider the sequence of words of length three in the four-letter alphabet A, T, C, G.

$$ACGTCTAAATTGGTACCTTCATA \quad (4.1)$$

It is easy to build a bipartite graph for this sequence such that this sequence represents an Euler path in this graph. In particular, this graph will contain the fragment consisting of the words of length three, isolated in (4.1).



Now let the Cartesian synonymous partition contain the word AAT in the Cartesian class $A \times A \times (T, C)$ and contain the word CAT in the class $C \times A \times (A, T, C, G)$. (This is a real case of the triplet code of amino acids, which is examined below.) If we lift the sequence (4.1) to the corresponding sequence of quotient words, then the graph G of this sequence would have the following fragment

$$\begin{array}{l} \mathbf{AA} \text{ --- } \mathbf{Ax}(T, C) \\ \\ \mathbf{CA} \text{ --- } \mathbf{Ax}(A, T, C, G) \end{array}$$

Consequently, bipartite graphs for the sequence of the letters (4.1) and for the sequence of the corresponding quotient words are different. Thus, bipartite graphs for two levels of sequence on diagram Figure 11, generally speaking, are different. This difference means that there is no direct correspondence between combinatorial connectivity of the sequence of letters and combinatorial connectivity of the corresponding quotient words.

4.2. Cartesian synonymous partitions. Words of length two.

Let us consider the case of Cartesian synonymous partitions and words of length two. Two different partitions of the basic alphabet A_s produce two quotient alphabets $\mathbf{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_r\}$ and $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_h\}$. We have described in detail the corresponding constructions of words and sequences in this case in Example 2.2.2. Let us examine the problem of construction of a dense sequence that includes all possible words of length two over the given alphabets. Let us use for this purpose the bipartite graph G with the sets of vertices σ^- and σ^+ introduced in Section 3.1; $|\sigma^-| = r$, $|\sigma^+| = h$. The set σ^- corresponds to the alphabet \mathbf{A} and contains the first letters of the words, and the set σ^+ corresponds to the alphabet \mathbf{B} and contains the second letters of the words. All possible arcs of words are drawn in this graph. The arcs of recovery are determined by the table of correspondence of symbols, which in this case is not a free parameter. Specifically, we assume that the arc of recovery occurs from vertex \mathbf{b}_j into vertex \mathbf{a}_i if the intersection of the corresponding sets is not empty: $\mathbf{a}_i \cap \mathbf{b}_j \neq \emptyset$. This definition is reasonable, since the sequences in the terms of a Cartesian synonymous partition must correspond to the sequences of letters of the basic alphabet in the sense of Section 2. Thus, the possibility of construction of a dense sequence is determined by the conditions given in Section 3.1. Some of the \mathbf{A} and \mathbf{B} partitions of the basic alphabet satisfy these conditions, and some of them do not.

Example 4.2.1. Let the alphabet \mathbf{A} consist only of two sets of elements $\{a_1, \dots, a_{s-1}\}, \{a_s\}$, and let the alphabet \mathbf{B} coincide with \mathcal{A}_s ($\mathbf{B} \equiv \mathcal{A}_s$). In this case $|\sigma^-| = 2$ and the out-degree of each vertex of the set σ^- is equal to s ; also $|\sigma^+| = s$ and the in-degree of each vertex of the set σ^+ is equal to 2. If $W = \{a_s\} \in \sigma^-$, then $|S(W)| = 1$, since only one arc of recovery enters the vertex $\{a_s\}$ according to the definition of the arcs of recovery: only vertex $\{a_s\} \in \sigma^+$ has a non-empty intersection with the vertex $\{a_s\} \in \sigma^-$. Thus, inequality $2|S(W)| \geq s|W|$ is true only when $s = 1, 2$. According to Corollary 3.1.3, this indicates an absence of an Euler cycle in the graph with $s > 2$. Consider now the same construction with the transposition of the alphabet: $\mathbf{A} \equiv \mathcal{A}_s$ and \mathbf{B} consists only of two sets of elements $\{a_1, \dots, a_{s-1}\}, \{a_s\}$. In this case $|\sigma^-| = s$ and the out-degree of each vertex of the set σ^- is equal to 2; also $|\sigma^+| = 2$ and the in-degree of each vertex of the set σ^+ is equal to s . Let $W = \{a_1, \dots, a_{s-1}\} \in \sigma^-$, $|W| = s-1$. All arcs of recovery, by construction, are drawn from the sole vertex $\{a_1, \dots, a_{s-1}\}$ of the set σ^+ that has a non-empty intersection with each of the vertices $a_i \in \sigma^-$, $i = 1, \dots, s-1$. Therefore, $|S(W)| = 1$. According to Corollary 3.1.3, an Euler cycle exists if $|\sigma^-||S(W)| \geq |W||\sigma^+|$. In our case this inequality yields $s \geq 2(s-1)$ and it is only fulfilled with $s = 1, 2$.

Let us suppose that positive integers ν_1 and ν_2 are such that s/ν_1 and s/ν_2 are integers.

Statement 4.2.2. *Let the set $A = \{a_1, \dots, a_r\}$ and the set $B = \{b_1, \dots, b_h\}$ be the partitions of the basic alphabet \mathcal{A}_s , where the set A consists of subsets of size ν_1 and the set B consists of subsets of size ν_2 . There exists a cyclic dense sequence in the set of all possible words over these alphabets.*

Proof. Let us examine the corresponding bipartite graph G . The set of vertices σ^- corresponds to the quotient letters of the set \mathbf{A} and the set of vertices σ^+ corresponds to the quotient letters of the set \mathbf{B} . $|\sigma^-| = s/\nu_1$ and $|\sigma^+| = s/\nu_2$ by construction. Consider an arbitrary set $W \subseteq \sigma^-$. Then $S(W) \subseteq \sigma^+$, by definition, is a set of all vertices from which the arcs of recovery are drawn into the vertices of the set $W \subseteq \sigma^-$. Further, the number of different letters of the basic alphabet \mathcal{A}_s in all words of the set W is equal to $\nu_1|W|$; the same letters, by the definition of the arcs of recovery, can be found in all words of the set $S(W)$. Thus, $|S(W)| \geq \nu_1|W|/\nu_2$. Therefore the following inequality holds:

$$|W|s/\nu_1 \geq (\nu_1|W|/\nu_2)s/\nu_1 = |W|s/\nu_2.$$

Since an Eulerian conditions, according to Corollary 3.1.3, take in this case the form $|S(W)|s/\nu_1 \geq |W|s/\nu_2$, the Statement is proven. \square

Theorem 3.1.4 guarantees the existence of an Euler graph with the arcs of recovery constituting the matching in graph G . However, there can exist other systems of the arcs of recovery. Let us assume $\nu_1 = \nu_2 = 2$ and $|\sigma^-| = |\sigma^+| = n$. At most two arcs of recovery leave each vertex of the set σ^+ of the corresponding bipartite graph G . Specifically, each quotient letter of the alphabet \mathbf{B} contains two letters of the basic alphabet. Therefore, it can correspond either to two quotient letters of the alphabet \mathbf{A} (if these letters of the basic alphabet stand in the different quotient

letters in alphabet \mathbf{A}) or to one quotient letter in the alphabet \mathbf{A} (if this is the same set as in \mathbf{B}). For simplicity, let us assume that the alphabets \mathbf{A} and \mathbf{B} do not contain identical quotient letters. Then exactly two arcs of recovery leave each vertex σ^+ ; moreover one of them obviously participates in the matching, existence of which is guaranteed by Theorem 3.1.4. The multiplicity of this arc is obviously equal to n , and the multiplicity of the second arc is zero. Let us now change multiplicity n to $n - m$, and multiplicity 0 to m . It is easy to see that the obtained graph G is also an Euler graph. Indeed, the total multiplicity of the leaving arcs of each vertex of the set σ^+ is equal to n as before. There are two arcs of recovery that enter each vertex of the set σ^- , one of which now has multiplicity $n - m$, and the second one has multiplicity m , so that the total multiplicity of the entering arcs is equal to n as before. Let us denote by $(n - m, m)$ the spectrum that corresponds to each such distribution of multiplicities. The above construction generalizes Example 3.3.1. This example demonstrates that the described above distribution of multiplicities covers the entire set of all possible spectra.

Above we analyzed the case when the quotient letters of each alphabet consisted of the same number of letters of the basic alphabet. Now let us consider the case when the number of letters of the basic alphabet in the quotient letters is not fixed. For two partitions $\mathbf{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_r\}$ and $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_h\}$ of the basic alphabet \mathcal{A}_s with an equal number of subsets ($r = h$), the in-degree and out-degree of each vertex are equal. According to the Theorem 3.1.1, in this case the existence of an Euler cycle depends only on Cartesian partitions that produce alphabets \mathbf{A} and \mathbf{B} , since the table of correspondence is determined by the composition of the sets. It is possible to reformulate the criterion of the existence of the dense cycle obtained in Theorem 3.1.1. Let us examine the set of words $W \subset \mathbf{A}$. Partition \mathbf{B} in an obvious manner induces the partition of the set of all the letters over the basic alphabet included in the words from the set W . Let us denote this partition by $W|\mathbf{B} = \{\mathbf{b}_1 \cap W, \mathbf{b}_2 \cap W, \dots, \mathbf{b}_h \cap W\}$ and let us denote the number of non-empty subsets of $W|\mathbf{B}$ by $|W|\mathbf{B}|$. By definition, the arc of recovery occurs from vertex \mathbf{b}_i into vertex of the set W if $\mathbf{b}_i \cap W \neq \emptyset$. Therefore, the following inequality always holds: $|S(W)| \geq |W|\mathbf{B}|$.

Statement 4.2.3. *Let the sets $\mathbf{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_r\}$ and $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_h\}$ be the partitions of the basic alphabet \mathcal{A}_s into an equal quantity of subsets ($r = h$). Let for each subset $W \subset \mathbf{A}$ the inequality $|W|\mathbf{B}| \geq |W|$ be fulfilled. Then there is a dense cyclic sequence in the set of all possible words over these alphabets.*

The proof is obvious in a view of the observations made above. Specifically, from the inequalities $|S(W)| \geq |W|\mathbf{B}|$ and $|W|\mathbf{B}| \geq |W|$ we obtain the following inequality: $|S(W)| \geq |W|$, and apply Theorem 3.1.1.

4.3. Triplet code of amino acids.

According to the table of the codes of amino acids, its synonymous partition is not Cartesian. Namely, let there be three alphabets:

$$\mathbf{A}_1 = \{A, T, C, G\}, \mathbf{A}_2 = \{A, T, C, G\}, \mathbf{A}_3 = \{(A, T, C, G), (A, G), (T, C), (C), (A, T, G)\}$$

for the first, the second and third position of code word respectively. In the first two alphabets the letters coincide with the letters of the basic alphabet, in the third alphabet the subsets (A,T,C,G), (A,G), (T,C), (C), (A,T,G) are the quotient letters. These subsets have non-empty intersections, so that this alphabet corresponds to irregular Cartesian synonymous partition (see Example 2.2.3). In the proposed alphabets it is almost always possible to write down the synonymous classes of amino acids as one word. Thus, for example, the entire synonymous class *proline* is $G \times G \times (A,T,C,G)$, and the entire synonymous class *glutamine* is $G \times T \times (T,C)$. However, *arginine* consists of two words of the synonymous Cartesian partitions:

$$G \times C \times (A, T, C, G) \cup T \times C \times (T, C),$$

intersection of which is empty. Same is also true for *leucine* and *serine*:

$$\begin{aligned} &A \times A \times (A, T, C, G) \cup G \times A \times (T, C), \\ &A \times G \times (A, T, C, G) \cup T \times C \times (A, G) \end{aligned}$$

respectively. The synonymous classes of remaining amino acids are written as one word in the given alphabets (Synonymous class stop-codon also consists of two quotient words $A \times T \times (T, G) \cup A \times C \times T$). Thus, amino-acid synonymous partition can be represented in the form of the direct union of several Cartesian partitions. In order to verify the existence of a dense sequence it is necessary to take one word from each synonymous class and to proceed according to the proposed scheme. That is, to build an appropriate bipartite graph and to analyze the possibility of the existence of an Euler cycle in it. All amino acids will be represented in such a graph, but some of them will be represented by different quotient words. Further, it is possible to verify the existence of a dense sequence in each case and to find all such sequences. In this case the method is effective, since it requires to verify 8 possible combinations, if stop-codon is not included, or 16 otherwise. Figure 12 depicts the bipartite graph G for this combination of the quotient words. Here σ^- is the left set of vertices and σ^+ is the right set of vertices. All arcs of words are drawn and numbers on the arcs correspond to the list of amino acids (Table 2). Of course, only one version of the amino acids that can be represented in two versions is included in graph. The second version is shown near the name of the corresponding amino acid in Table 2.

Let us remove arcs of words and leave only in-degrees of vertices of the set σ^+ and out-degrees of vertices of the set σ^- for convenience in further analysis. Blue color denotes the in-degrees and the out-degrees of vertices. Let us draw only the arcs of recovery on the obtained graph. Using definition of Z -family introduced in Section 2.3, let us define the table of correspondence as follows: *an arc between the two-letter words exists if a word from the left set of the graph is contained in the quotient word on the right that is considered as a Cartesian product of the components of its sets* (see Figure 13).

Let us demonstrate that no set of multiplicities can convert this graph into the Euler graph. First of all let us note that vertex TT of the set σ^- and vertices

Fig. 12. The bipartite graph G with all arcs of words for amino acids

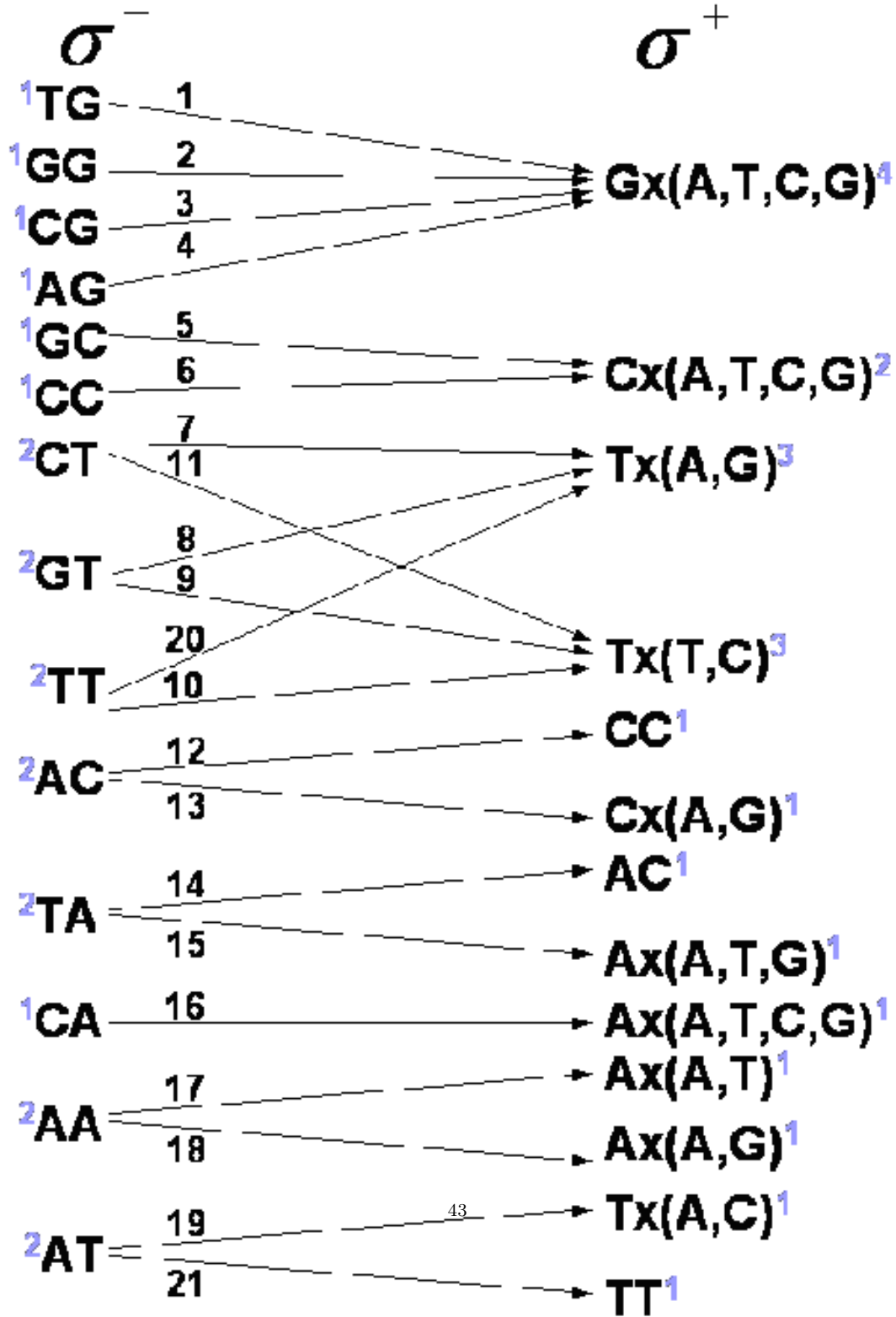


Fig. 13. The bipartite graph G with all arcs of recovery for amino acids

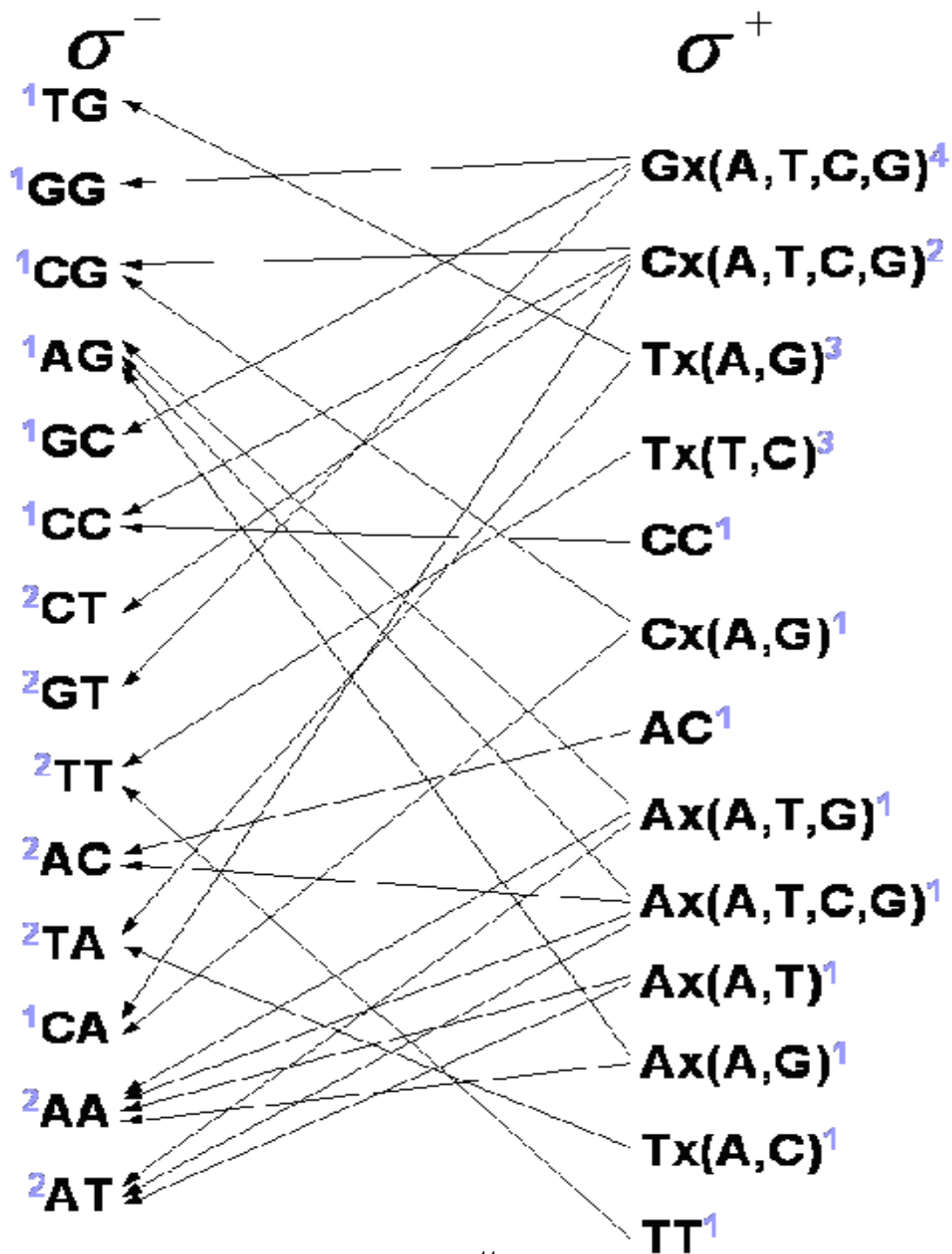
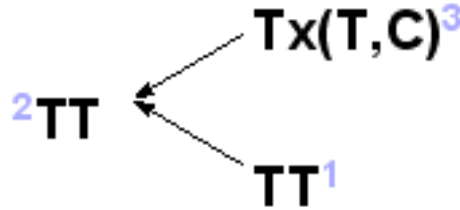


Table 2. Numerated list of amino acids

1 threonine	8 histidine	15 isoleucine
2 proline	9 glutamine	16 valine
3 alanine	10 lysine	17 leucine $GxAx(T,C)$
4 serine $TxCx(A,G)$	11 glutamate	18 phenylalanine
5 arginine $TxCx(T,C)$	12 thryptophan	19 tyrosine
6 glycine	13 cysteine	20 asparagine
7 aspartate	14 methionine	21 stop ATT

$T \times (T,C)$, TT of the set σ^+ form a graph that is not connected with the remaining vertices:

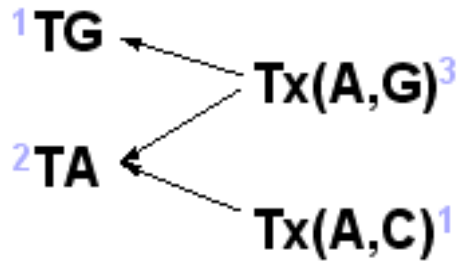


Assuming multiplicities of the arcs in this graph x and y (arcs $(TT, T \times (T,C))$, (TT, TT) respectively), we obtain the following equalities:

$$x + y = 2; x = 3; y = 1,$$

which are, obviously, contradictory. However, if, for example, $x = 3$, and $y = 0$, then sets σ^- and σ^+ each have one vertex (TT) of degree 1 and the entire graph G can have an Euler path under condition that the in-degree and out-degree of each remaining vertex are equal. Arrangement of the multiplicities on the remaining arcs of the graph does not depend on the multiplicities of the arcs of the examined subgraph.

Consider now another isolated subgraph in the graph G :



It is obvious that all vertices of this subgraph cannot have equal in-degree and out-degree for any given multiplicities of arcs. Therefore, there is no such distribution of multiplicities on this graph that the graph would have an Euler path. Thus, given combination of the amino acids representatives cannot form a dense sequence.

Let us select now the word $T \times C \times (T, C)$ as a representative of the synonymous class *arginine*. In the obtained graph G with arcs of words on Figure 14 all notation is the same as in the previous figure.

Let us build graph G in the following way. As earlier, let us remove the arcs of words and let us leave the in-degrees and the out-degrees of vertices. In the obtained graph let us draw the arcs of recovery; not all but only those necessary for the balance of vertices, with the non-zero multiplicity. Naturally, our analysis was involved with all arcs of recovery. Also for simplification of the figure of graph G we changed the positions of vertices in both sets of vertices, σ^- and σ^+ . Pluses denote two vertices, where the in-degree is not equal to out-degree. For the rest of the vertices their in-degree is equal to their out-degree, therefore there is an Euler path in this graph (see Figure 15) and, in this case, this is the necessary condition for the existence of a dense sequence. Our computer calculations showed that such sequences do exist and on the basis of the described technology we obtained entire set of dense sequences, which contain exactly one word from each synonymous group.

Acknowledgements We thank A. Korol, E.N. Trifonov, and A. Vainstein for discussions and critical comments on the paper.

Bibliography

- [1] E.N. Trifonov, Genetic Sequences as Product of Compression by Inclusive Superposition of Many Codes, *Molecular Biology*, Vol. 31, No. 4, 1997, p. 647-654
- [2] O. Peleg, V. Kirzhner, E. Trifonov, A. Bolshoy. Overlapping messages and survivability. *J Mol Evol.* 2004 Oct;59(4): 7-520
- [3] E. Coward. Shufflet: shuffling sequences while conserving the k-let counts. *Bioinformatics. Applications note*, Vol.15, no.12 1999, p. 1058-1059.
- [4] S.F. Altschul and B.W. Erickson. Significance of Nucleotide Sequence Alignments: A Method for Random Sequence Permutation That Preserves Dinucleotide and Codon Usage. *Mol.Biol.Evol.* 2(6): 526- 538, 1985
- [5] M. Lothaire, Combinatorics on Words, *Encyclopedia of Mathematics and its Applications*, Vol. 17, Section: Algebra, 1983
- [6] M. Hall, Combinatorial Theory, Ginn, Blaisdell, 1976
- [7] L.R. Ford, Jr. and D.R. Fulkerson. Flows in Networks. Princeton University Press, Princeton, NJ, 1962.
- [8] E. Ukkonen. Approximate String Matching with q-grams and Maximal Matches. *Theoretical Computer Science*, vol. 92: 191-211, 1992
- [9] P.A. Pevzner. DNA Physical Mapping and Alternating Eulerian Cycles in Colored Graphs. *Algorithmica*, vol. 13: 77-105, 1995.

Fig. 14. The bipartite graph G for amino acids; word

$T \times C \times (T, C)$ is selected as a representative of the synonymous class arginine

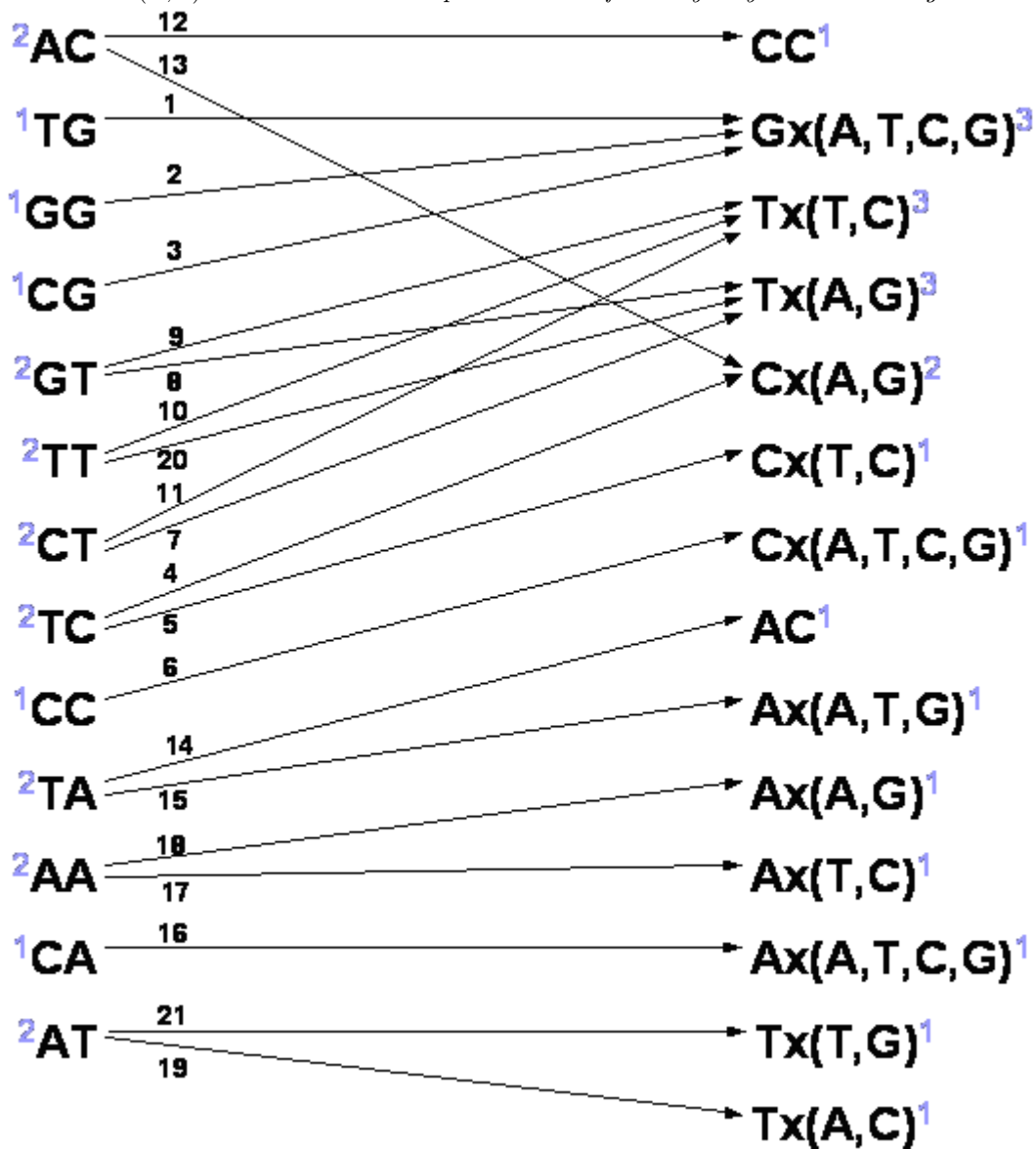


Fig. 15. The bipartite graph G with arcs of recovery; not all but only those necessary for the balance of vertices, with the non-zero multiplicity

