Sent for publication to *Neurocomputing*, March 2004

Holographic reduced representation in artificial intelligence and cognitive science

Vladimír Kvasnička and Jiří Pospíchal

Faculty of Computer Science and Information Technologies
Slovak Technical University
812 37 Bratislava, Slovakia

1. Introduction

A modern view of the relation between brain and mind is based on the neuroscience paradigm [3], according to which the architecture of the brain is determined by connections between neurons, their inhibitory or excitatory character and also by the strength of the connections. Human brain displays a great plasticity, synapses are perpetually formed (but also deleted) during a learning process. It can be stated, that *an ability of brain to perform not only cognitive activities, but also to serve as memory and control center for our motoric activities, is fully encoded by its architecture.* The metaphor of a human brain as a computer should be therefore formulated in such a way, that a computer is a *parallel distributed computer* (containing many billions of neurons, elementary processors interconnected into a complex neural network). A program in such a parallel computer is directly encoded in the architecture of the neural network, i.e. human brain is a single-purpose parallel computer represented by a neural network, which can not be reprogrammed without a change of its architecture.

It follows from the above general statement that the mind with the brain creates one integral unit, which is characterized by a complementary *dualism*. The mind is in this approach understood as a program carried out by the brain, while this program is specified by architecture of the distributed neural network representing the brain. The brain and the mind are two different aspects of the same object:

(1) When talking about the brain, we have in mind a „hardware“ structure, biologically determined by neurons and their synaptic connections (formally represented by a neural network), on the other hand.

(2) When talking about the mind, we have in mind cognitive and other similar activities of the brain, which are carried out on a symbolic level, where the transformation of symbolic information is processed on the basis of (simple) rules.

A complementary dualism between brain and mind causes certain difficulties in the interpretation of cognitive activities of mind. A purely neural approach to the interpretation of cognitive activities of mind focuses on the search of neural correlates of neural activities and cognitive activities (connectionism). The application of the neural paradigm for the interpretation of symbolic cognitive activities has a „side effect“ in „dissolving“ of these activities in their microscopic description, symbols quasi „disappear“ in the detailed description of activities of neurons, strengths of synaptic connections etc. On the other side, the absolute acceptance of symbolic paradigm in interpretation of cognitive activities of mind

(cognitivism), ignoring of the fact, that mind is thoroughly embedded in brain, leads to a conceptual sterility, to an effort to explain cognitive activities of human mind only in the phenomenological terms derived from the concept of symbol. It leads to symbolic constructs (methods, algorithms etc.), for which there usually does not exist any experimental support in neuroscience. The goal of this paper is to highlight an alternative approach, which may overcome the gap between the connectionist and cognitivistic approach in the description and interpretation of cognitive activities of the human brain [10-13]. We shall show, that the application of a distributed representation allows to integrate connectionism and cognitivism, where mental representations (symbols) are specified by distributed patterns of neural activities, while over these distributed patterns we can introduce formal algebraic operations, which not only allow to mathematically model cognitive operations, but also allow to simulate processes of storage and retrieval of information from memory.

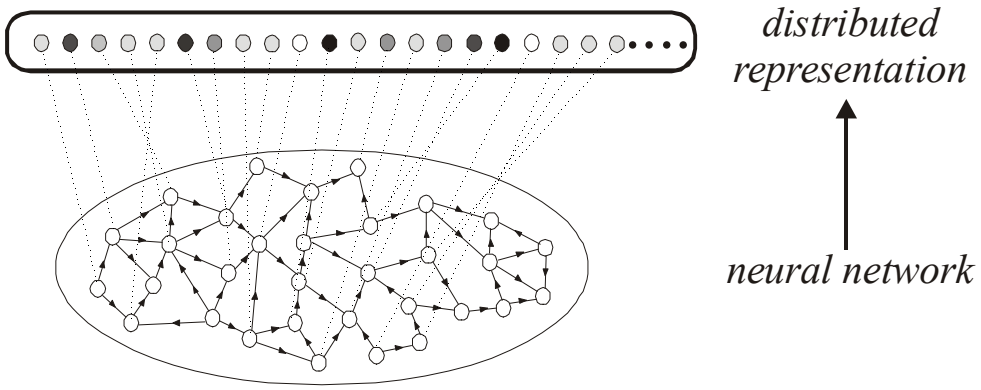

Figure 1. A visualization of the transition from neural network to distributed representation. The state of neural network in the time *t* is given by the activities of single neurons, which are determined by the activities in the previous time *t*-1 and by weight coefficients of single oriented connections. Using a certain abstraction, these activities can be ordered into one big one-dimensional array (vector) of real numbers (their size is determined by the level of gray of the corresponding component – neuron). In the distributed representation the architecture of the neural network is ignored, i.e. two distributed representations must be understood as totally independent without mutual relations, their incidental connections derived from neural network are completely ignored. New unary and binary operations are introduced in the distributed representation, which enable to create new distributed representations from the original ones.

We shall turn our attention to a nontraditional style of performing calculation by using distributed patterns. This approach is substantially different from classical numeric and symbolic computations and it is a suitable model tool for understanding of global properties of neural networks. We shall show, that such a „neurocomputing“ is based on extensive randomly created patterns (represented by multidimensional vectors with random entries), see fig. 1. This approach, which basic principles were formulated already at the end of sixties [2,4,5,9,14], was crowned by a series of works by Tony Plate [7-9] about „*holographic reduced representation*“ (HRR). We shall show which types of computation can be implemented in this approach and whether they help us to understand the processes in the brain during cognitive activities. Our addition to the development of HRR consists in its application to modeling of cognitive processes of reasoning by application of rules modus ponens a modus tollens. Kanerva [16-18] in the middle of nineties proposed a certain alternative to HRR, which is based on randomly generated binary vectors.

2. A mathematical formulation of holographic representation

The aim of this chapter is a presentation of basic properties of a holographic representation, which was developed by A. Plate [7-9]. Its basic notion is a *conceptual vector*, which is represented by an *n*-dimensional vector

$$\boldsymbol{a} \in R^n \Rightarrow \boldsymbol{a} = \left(a_0, a_2, ..., a_{n-1}\right) \tag{1}$$

where its components are random numbers with a standard normal distribution

$$a_i = N\left(0, 1/n\right) \quad \forall \in \left\{0, 1, ..., n-1\right\} \tag{2}$$

where $N(0,1/n)$ is a random number with a mean equal to 0 and a standard deviation $1/n$.

Over conceptual vectors there is defined a binary operation „c*onvolution*“, which assigns to a couple of vectors a third vector, $\otimes \quad : \quad R^n \times R^n \rightarrow R^n$, or

$$\boldsymbol{c} = \boldsymbol{a} \otimes \boldsymbol{b} \tag{3}$$

The components of the resulting vector $c = \left(c_0, c_1, ..., c_{n-1}\right)$ are determined by a formula

$$c_i = \sum_{j=0}^{n-1} a_j b_{[i-j]} \quad \left(i = 0, 1, ..., n-1\right) \tag{4}$$

where the index in the square brackets, [*k*], is defined using a *modulo n* operation as follows[1]

$$k' = k \ mod \ n \tag{5a}$$

$$\left[k\right] = \begin{cases} k' & \left(if \ k' \geq 0\right) \\ n + k' & \left(if \ k' < 0\right) \end{cases} \tag{5b}$$

Since it is one of the basic notions of the holographic representation, we shall give a table of values [*k*] for $-4 \leq k \leq 4$ and $n=4$.

k	*k*'	[*k*]
-4	0	0
-3	-3	1
-2	-2	2
-1	-1	3
0	0	0
1	1	1
2	2	2
3	3	3
4	0	0

[1] A standard definition of an arithmetic operation *k modulo n* is determined as a remainder after integer division by a number *n*. It is necessary to comment, that the used definition of the operation *k modulo n* is different from this standard definition for negative numbers *k*. While the standard definition provides a result with a negative value, if the result is negative in our definition, than it is transformed by adding *n* to it.

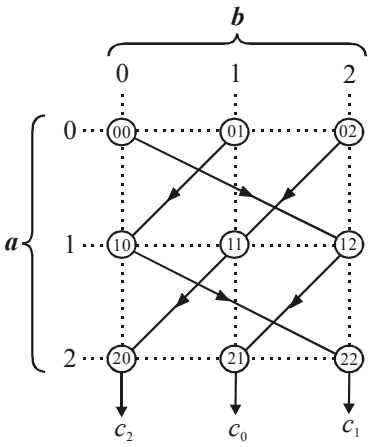

Figure 2. Convolution of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ for n=3 (see (6)).

A convolution of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ for n=3 has the following form (see fig. 2)

$$\begin{aligned} c_0 &= a_0 b_0 + a_1 b_2 + a_2 b_1 \\ c_1 &= a_0 b_1 + a_1 b_0 + a_2 b_2 \\ c_2 &= a_0 b_2 + a_1 b_1 + a_2 b_0 \end{aligned} \tag{6}$$

The convolution satisfies the following properties:

(1) commutativity, $\boldsymbol{a} \otimes \boldsymbol{b} = \boldsymbol{b} \otimes \boldsymbol{a}$

(2) associativity, $(\boldsymbol{a} \otimes \boldsymbol{b}) \otimes \boldsymbol{c} = \boldsymbol{a} \otimes (\boldsymbol{b} \otimes \boldsymbol{c})$

(3) distributiveness, $\boldsymbol{a} \otimes (\alpha \boldsymbol{b} + \beta \boldsymbol{c}) = \alpha (\boldsymbol{a} \otimes \boldsymbol{b}) + \beta (\boldsymbol{a} \otimes \boldsymbol{c})$

(4) an existence of a unit vector, $\mathbf{1} \otimes \boldsymbol{a} = \boldsymbol{a} \quad (\mathbf{1} = (1,0,...,0))$

The convolution can be also expressed by a *circulant matrix* [1]

$$\left.\begin{aligned} c_0 &= a_0 b_0 + a_1 b_2 + a_2 b_1 \\ c_1 &= a_0 b_1 + a_1 b_0 + a_2 b_2 \\ c_2 &= a_0 b_2 + a_1 b_1 + a_2 b_0 \end{aligned}\right\} \Rightarrow \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \underbrace{\begin{pmatrix} a_0 & a_2 & a_1 \\ a_1 & a_0 & a_2 \\ a_2 & a_1 & a_0 \end{pmatrix}}_{circ(a)} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} \Rightarrow \boldsymbol{a} \otimes \boldsymbol{b} = circ(\boldsymbol{a})\boldsymbol{b} \tag{7}$$

This specific example is generalized for an arbitrary dimension n as follows

$$\boldsymbol{c} = \boldsymbol{a} \otimes \boldsymbol{b} \Leftrightarrow \boldsymbol{c} = circ(\boldsymbol{a})\boldsymbol{b} \Leftrightarrow circ(\boldsymbol{a}) = \begin{pmatrix} a_0 & a_{n-1} & .. & a_2 & a_1 \\ a_1 & a_0 & .. & a_3 & a_2 \\ .. & .. & .. & .. & .. \\ a_{n-2} & a_{n-3} & .. & a_0 & a_{n-1} \\ a_{n-1} & a_{n-2} & .. & a_1 & a_0 \end{pmatrix} \tag{8}$$

where the general circulant matrix has its elements

$$\left(circ(a)\right)_{ij} = a_{[i-j]} \tag{9}$$

The circulant matrix has the following properties

$$circ(\boldsymbol{a} \otimes \boldsymbol{b}) = circ(\boldsymbol{a})\, circ(\boldsymbol{b}) \tag{10}$$

and since the convolution is a commutative operation, then circulant matrices are mutually commutative

$$circ(\boldsymbol{a})\, circ(\boldsymbol{b}) = circ(\boldsymbol{b})\, circ(\boldsymbol{a}) \tag{11}$$

Let $\boldsymbol{X}$ is an inverse matrix to a circulant matrix $circ(\boldsymbol{a})$

$$\boldsymbol{X}circ\left(\boldsymbol{a}\right)=circ\left(\boldsymbol{a}\right)\boldsymbol{X}=\boldsymbol{E} \tag{12}$$

Its alternative form is

$$\boldsymbol{X}=circ^{-1}\left(\boldsymbol{a}\right) \tag{13}$$

Let a^{-1} be an *inverse vector* to the vector a, $\boldsymbol{a}^{-1}\otimes\boldsymbol{a}=\boldsymbol{1}=\left(1,0,...,0,0\right)$, then assuming that the circulant matrix is regular, $\left|circ\left(\boldsymbol{a}\right)\right|\neq 0$, it follows

$$circ^{-1}\left(\boldsymbol{a}\right)=circ\left(\boldsymbol{a}^{-1}\right) \tag{14}$$

Let us define a unary operation *involution* (see fig. 3)

$$\left(\ \right)^{*}\ :\ R^{n}\rightarrow R^{n} \tag{15}$$

by a formula

$$\boldsymbol{b}=\boldsymbol{a}^{*}=\left(a_{[0]},a_{[-1]},...,a_{[-n+2]},a_{[-n+1]}\right) \tag{16}$$

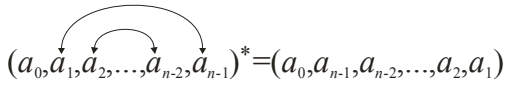

Figure 3. Visualization of the unary operation of involution.

The operation of involution satisfies the equations

$$\left(\boldsymbol{a}+\boldsymbol{b}\right)^{*}=\boldsymbol{a}^{*}+\boldsymbol{b}^{*} \tag{17a}$$

$$\left(\boldsymbol{a}\otimes\boldsymbol{b}\right)^{*}=\boldsymbol{a}^{*}\otimes\boldsymbol{b}^{*} \tag{17b}$$

$$\left(\boldsymbol{a}\otimes\boldsymbol{b}^{*}\right)\cdot\boldsymbol{c}=\boldsymbol{a}\cdot\left(\boldsymbol{b}\otimes\boldsymbol{c}\right) \tag{17c}$$

$$\boldsymbol{a}^{**}=\boldsymbol{a} \tag{17d}$$

$$circ\left(a^{*}\right)=circ^{T}\left(a\right) \tag{17e}$$

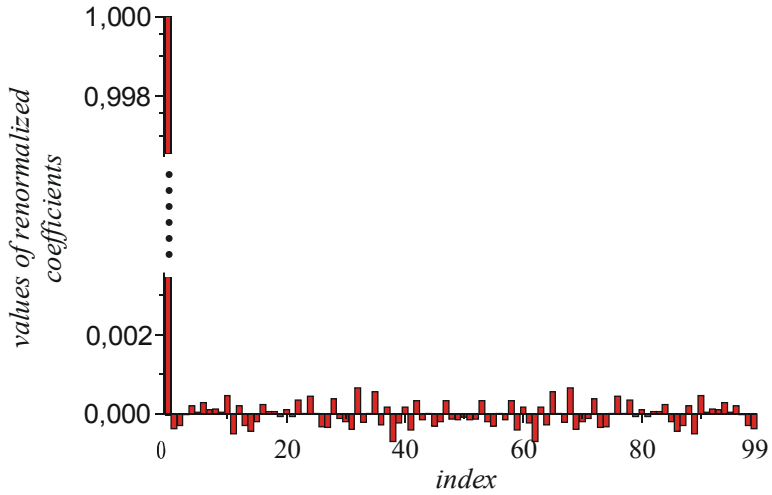

Figure 4. A histogram plot of single components of $\boldsymbol{c}^{*}\otimes\boldsymbol{c}$, where $\boldsymbol{c}$ is randomly generated conceptual vector for n=100. We see from this plot that an absolute value of the „first“ component $\left(\boldsymbol{c}^{*}\otimes\boldsymbol{c}\right)_{0}$ is greater about two-three orders than absolute values of remaining components $\left(\boldsymbol{c}^{*}\otimes\boldsymbol{c}\right)_{i}$, for $i\geq 1$. It means that the product $\boldsymbol{c}^{*}\otimes\boldsymbol{c}$ after a proper normalization plays approximately a role of unit vector $\left(\boldsymbol{c}\cdot\boldsymbol{c}\right)^{-1}\left(\boldsymbol{c}^{*}\otimes\boldsymbol{c}\right)\approx\boldsymbol{1}=\left(1,0,...,0,0\right)$.

We prove that an involution $\boldsymbol{c}^*$ is approximately equal to an inverse vector $\boldsymbol{c}^{-1}$, $\boldsymbol{c}^* \otimes \boldsymbol{c} \approx \mathbf{1}$. Let us study the i-th component of convolution $\boldsymbol{c}^* \otimes \boldsymbol{c}$

$$\left(\boldsymbol{c}^* \otimes \boldsymbol{c}\right)_i = \sum_{k=0}^{n-1} c_k^* c_{[i-k]} = \sum_{k=0}^{n-1} c_{[-k]} c_{[i-k]}$$
$$= \begin{cases} \boldsymbol{c} \cdot \boldsymbol{c} & \left(\text{for } i = 0\right) \\ \sum_{k=0}^{n-1} c_{[-k]} c_{[i-k]} & \left(\text{for } i > 0\right) \end{cases} \tag{18}$$

The zero-component of convolution $\left(\boldsymbol{c}^* \otimes \boldsymbol{c}\right)_0$ corresponds to a scalar product $\boldsymbol{c} \cdot \boldsymbol{c}$ expressed as a sum of positive "diagonal" terms c_i^2, whereas other components $\left(\boldsymbol{c}^* \otimes \boldsymbol{c}\right)_i$, for $i \geq 1$, are determined by sums of "nondiagonal" terms $c_i c_j$ with fully random signs. This observation has an important consequence that $\left(\boldsymbol{c}^* \otimes \boldsymbol{c}\right)_0$ is much greater than absolute values of remaining components $\left(\boldsymbol{c}^* \otimes \boldsymbol{c}\right)_i$, for $i \geq 1$, then we proved that $\boldsymbol{c}^* \otimes \boldsymbol{c} \approx \mathbf{1}$, which was to be proved (see Fig. 4).

One of the basic aspects of the holographic representation is the possibility of reconstruction of the original components, which were used for construction of convolution of two vectors. This possibility is very important, since it allows us to decode the original information from the complex conceptual vectors. *Reconstruction of* $\boldsymbol{x}$ from $\boldsymbol{c} \otimes \boldsymbol{x}$ is based on the above proved formula $\boldsymbol{c}^* \otimes \boldsymbol{c} \approx \mathbf{1}$

$$\tilde{\boldsymbol{x}} = \boldsymbol{c}^* \otimes \left(\boldsymbol{c} \otimes \boldsymbol{x}\right) = \left(\boldsymbol{c}^* \otimes \boldsymbol{c}\right) \otimes \boldsymbol{x} \approx \frac{1}{\boldsymbol{c} \cdot \boldsymbol{c}} \mathbf{1} \otimes \boldsymbol{x} = \frac{1}{\boldsymbol{c} \cdot \boldsymbol{c}} \boldsymbol{x} \tag{19}$$

according to which the convolution $\boldsymbol{c}^*$ with the vector $\boldsymbol{c} \otimes \boldsymbol{x}$ produces the vector $\tilde{\boldsymbol{x}}$, which is similar to the original vector $\boldsymbol{x}$, $\tilde{\boldsymbol{x}} \approx \boldsymbol{x}$. This result can be reformulated in the form

$$\frac{1}{\left(\boldsymbol{c} \cdot \boldsymbol{c}\right)} \tilde{\boldsymbol{x}} = \begin{pmatrix} x_0 \\ x_1 + \eta_1 \\ \ldots\ldots\ldots\ldots \\ x_{n-1} + \eta_{n-1} \end{pmatrix} = \boldsymbol{x} + \boldsymbol{\eta} \tag{20}$$

where $\boldsymbol{\eta}$ is interpreted as a random noise with a normal distribution with a zero mean and a standard deviation much smaller than $\boldsymbol{x}$.

The *overlap* of the resulting vector $\tilde{\boldsymbol{x}}$ with the original vector $\boldsymbol{x}$ is determined from a scalar product by

$$-1 \leq overlap\left(\boldsymbol{x}, \tilde{\boldsymbol{x}}\right) = \frac{\boldsymbol{x} \cdot \tilde{\boldsymbol{x}}}{|\boldsymbol{x}||\tilde{\boldsymbol{x}}|} \leq 1 \tag{21}$$

where the inequalities result directly from the Schwartz's inequality from linear algebra. The more this value is close to its maximum value, the more similar[2] are the vectors $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$.

In the fig. 5 a histogram of overlaps is shown for the product $\boldsymbol{c} \otimes \boldsymbol{x}$, containing a couple of randomly generated different conceptual vectors $\boldsymbol{c}$ and $\boldsymbol{x}$ of the dimension n=1000. It is evident from the figure, that the most common overlap between $\tilde{\boldsymbol{x}} = \boldsymbol{c}^* \otimes \boldsymbol{c} \otimes \boldsymbol{x}$ and $\boldsymbol{x}$ is around 0.7, from which follows, that the vectors $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$ are similar, $\tilde{\boldsymbol{x}} \approx \boldsymbol{x}$.

[2] In the case, that the overlap value approaches -1, then the vectors $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$ are also similar, even though they have opposite orientation (they are anticolinear).

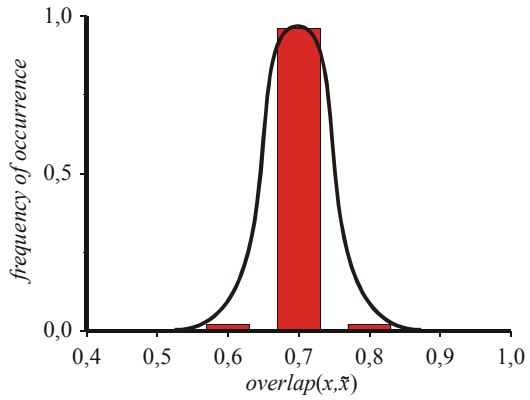

Figure 5. A histogram of overlaps between vectors $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$ (of dimensions n=1000) has highest frequency around 0.7, from which follows, that the vectors $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$ are similar.

Let's turn our attention to the second possibility of the verification of the formula (20b) with the application of the approach called the „*superposition memory*". Let us have a set containing $p+q$ randomly generated conceptual vectors, $X=\{\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_p,\boldsymbol{x}_{p+1},...,\boldsymbol{x}_{p+q}\}$, while $p<q$. Using the first p vectors from X allows us to define a *memory vector* $\boldsymbol{t}$ as their sum

$$\boldsymbol{t}=\sum_{i=1}^{p}\boldsymbol{x}_i \tag{22}$$

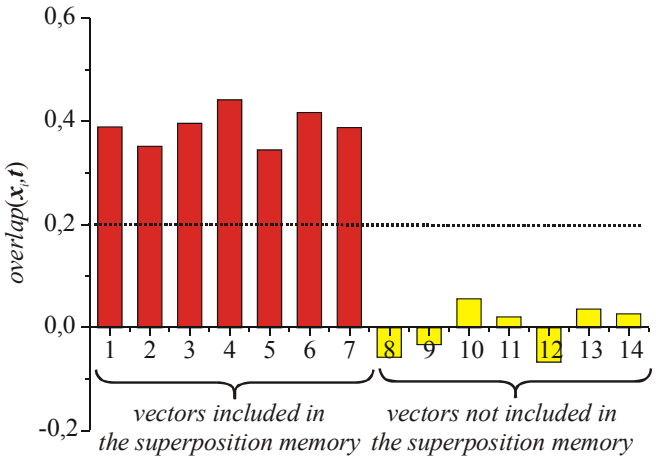

Figure 6. Illustration of the superposition memory for the first 7 vectors of the set X, which contains 14 randomly generated conceptual vectors of the dimension n=1000. The threshold value ϑ can be in this case set to 0.2.

The vector $\boldsymbol{t}$ represents a superposition memory, which by a simple additive way contains vector from the set X. The decision, whether some vector $\boldsymbol{x}\in X$ is contained in $\boldsymbol{t}$ must be based on the value of the overlap (21)

$$overlap(\boldsymbol{x},\boldsymbol{t})=\frac{\boldsymbol{x}\cdot\boldsymbol{t}}{|\boldsymbol{x}||\boldsymbol{t}|} \tag{23}$$

If this value is greater than a predefined threshold value, $overlap(\boldsymbol{x},\boldsymbol{t})\geq\vartheta$, then the vector $\boldsymbol{x}$ is included in the superposition memory $\boldsymbol{t}$, in the opposite case, if $overlap(\boldsymbol{x},\boldsymbol{t})<\vartheta$, then the vector $\boldsymbol{x}$ is not included in $\boldsymbol{t}$ (see fig. 6).

In the above illustrative example (see fig. 6) we used such a method for determination of conceptual vectors, which can „appear“ in some other different complex conceptual vector (which can be the result of complicated previous calculations – transformations). The used method is called „*clean-up“* and it is specified as follows: Let us have a set of vectors $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ and some vector $\boldsymbol{t}$. We face the decision, whether the memory vector (trace) $\boldsymbol{t}$ contains a superposition component, which is similar (or which is not similar) to some vector from the set X. This problem can be solved by calculating so called overlap (23), formally

$$\boldsymbol{x} \approx \boldsymbol{t} = \begin{cases} yes & \left(overlap(\boldsymbol{x}, \boldsymbol{t}) \geq \vartheta\right) \\ no & \left(overlap(\boldsymbol{x}, \boldsymbol{t}) < \vartheta\right) \end{cases} \tag{24}$$

where ϑ is a chosen *threshold value* of acceptance of the size of the overlap as the positive answer. The result of this cleaning-up process is a subset of vectors

$$X(\boldsymbol{t}) = \{\boldsymbol{x} \in X; \boldsymbol{x} \approx \boldsymbol{t}\} \subseteq X \tag{25}$$

We can put the question also in a rather different form, which is, whether the memory vector $\boldsymbol{t}$ is similar to any of the vectors from the set X? The answer to this more general question shall be decided from the maximum value of the overlap

$$overlap(\boldsymbol{t}, X) = \max_{x \in X} overlap(\boldsymbol{t}, \boldsymbol{x}) \tag{26}$$

Then we can rewrite (24) in the form

$$\boldsymbol{x} \approx X = \begin{cases} yes & \left(overlap(\boldsymbol{x}, X) \geq \vartheta\right) \\ no & \left(overlap(\boldsymbol{x}, X) < \vartheta\right) \end{cases} \tag{27}$$

3. Associative memory

The construction of the associative memory belongs to the main results of the holographic reduced representation, which can be further generalized by so called chunking. Let us have a set of conceptual vectors $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ and a training set $A_{train} = \{\boldsymbol{c}_i / \boldsymbol{x}_i ; i = 1, 2, ..., m\}$, which contains $m<n$ associated couples of conceptual vectors $\boldsymbol{c}_i / \boldsymbol{x}_i$, where $\boldsymbol{c}_i$ is the *input* to the associative memory (*cue*) and $\boldsymbol{x}_i$ is the output from the memory. Let's create a memory vector $\boldsymbol{t}$ representing *the associative memory* created from the training set A_{train}

$$\boldsymbol{t} = \boldsymbol{c}_1 \otimes \boldsymbol{x}_1 + ... + \boldsymbol{c}_m \otimes \boldsymbol{x}_m = \sum_{i=1}^{m} \boldsymbol{c}_i \otimes \boldsymbol{x}_i \tag{28}$$

Let us suppose, that we know in advance only the inputs $\boldsymbol{c}_i$ to the associative memory, we do not know the possible outputs from the set $X_{train} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_m\}$. The response of the associative memory to the input - clue $\boldsymbol{c}_i$ is determined by the process of „*clearing-up*“ represented by the formula (27). In the first step we shall calculate the vector $\tilde{\boldsymbol{x}}_i = \boldsymbol{c}_i^* \otimes \boldsymbol{t}$, then by a process based on the maximum value of the overlap we shall find whether $\tilde{\boldsymbol{x}}_i \approx \boldsymbol{x}_i \in X$

$$overlap(\tilde{\boldsymbol{x}}_i, X) = \max_{x \in X_{train}} overlap(\tilde{\boldsymbol{x}}_i, \boldsymbol{x}) \tag{29}$$

The associative memory will be illustrated by the following two examples.

1st example

This example uses only the training set $A_{train} = \{\boldsymbol{c}_i / \boldsymbol{x}_i ; i = 1, 2, ..., m\}$, which is randomly generated for m=8, while the dimension of conceptual vectors is n=1000. For each associated

couple $\boldsymbol{c}_i/\boldsymbol{x}_i$ there are calculated $\boldsymbol{t}_i = \boldsymbol{c}_i \otimes \boldsymbol{x}_i$. The values of $overlap\left(\boldsymbol{c}_i^* \otimes \boldsymbol{t}_i, \boldsymbol{x}_j\right)$ are presented in the table

	1	2	3	4	5	6	7	8
1	0.71703	-0.01820	0.01452	0.02776	-0.01488	-0.01922	-0.02442	0.01358
2	-0.03998	0.73804	0.01510	0.01430	0.00276	0.02346	-0.00545	-0.01626
3	-0.02757	-0.01736	0.64667	0.00474	-0.11580	-0.00812	0.01476	0.00379
4	0.00785	0.00374	-0.01899	0.68728	-0.15340	0.00005	-0.00561	0.00136
5	-0.00466	0.00426	-0.01831	-0.00827	0.70767	0.04175	-0.03384	-0.00668
6	-0.01467	0.02522	-0.01403	-0.01316	-0.03000	0.71444	0.00078	-0.00526
7	0.02966	0.00892	-0.00301	-0.00358	0.01285	0.00971	0.70790	0.01816
8	-0.00344	-0.01080	0.00843	-0.01871	0.00324	-0.02629	0.00851	0.58957

It is evident from the table, that the overlaps are sufficiently great just for diagonal values, while the non-diagonal overlaps are smaller by an order of magnitude. We can therefore unambiguously decide from the overlap, whether $\boldsymbol{c}_i^* \otimes \boldsymbol{t}_i \approx \boldsymbol{x}_i$ is associated with the cue $\mathbf{c}_i$.

2nd example

In this illustrative example we shall use the training set $A_{train} = \{\boldsymbol{c}_i/\boldsymbol{x}_i\}$, generated for m=10 associated couples – vectors of dimension n=1000. This memory is represented by a memory vector $\boldsymbol{t} = \boldsymbol{c}_1 \otimes \boldsymbol{x}_1 + ... + \boldsymbol{c}_m \otimes \boldsymbol{x}_m$. The following table shows 20 experiments of „clean up“, where we used with a 50% probability as an associative entry a vector $\boldsymbol{c}_i$ from the training set or a randomly generated conceptual vector. The table contains maximal values of overlaps (29), by which we can unambiguously determine, whether the used input has an associated counterpart in the training set.

#	max. overlap	Input index	index of output with max.overlap
1	0.311	6	6
2	0.047	rand. gener.	nonexistent
3	0.383	5	5
4	0.373	10	10
5	0.316	3	3
6	0.397	4	4
7	0.074	rand. gener.	nonexistent
8	0.065	rand. gener.	nonexistent
9	0.069	rand. gener.	nonexistent
10	0.039	rand. gener.	nonexistent
11	0.344	7	7
12	0.402	8	8
13	0.032	rand. gener.	nonexistent
14	0.073	rand. gener.	nonexistent
15	0.017	rand. gener.	nonexistent
16	0.004	rand. gener.	nonexistent
17	0.033	rand. gener.	nonexistent
18	0.056	rand. gener.	nonexistent
19	0.373	10	10
20	0.037	rand. gener.	nonexistent

It follows from the table, that the associative memory with the clean up process is unambiguously identifying, the values of a maximum overlap for conceptual vectors well specify the existence (or nonexistence) of corresponding associative outputs.

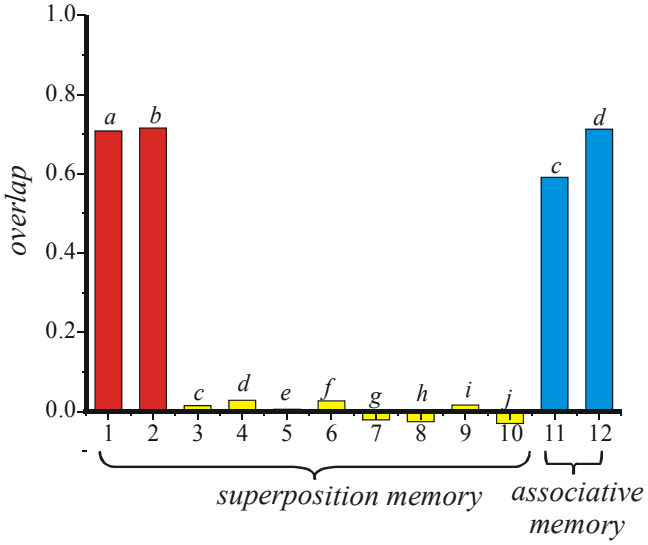

Figure 7. Illustration of analysis of a combination of superposition memory and associative memory. In the first step we carried out the process of clean up, by which we found, that the memory contains only two items – atomic vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, which in the next step serve as an input for further analysis of the associative memory, where we found as outputs the vectors $\boldsymbol{c}$ and $\boldsymbol{d}$. During the clean up process we moreover verified with a negative result, whether the superposition memory contains also further vectors $\boldsymbol{c},\boldsymbol{d},...,\boldsymbol{i},\boldsymbol{j}$. With a great probability we can therefore decide, that the memory vector $\boldsymbol{t}$ is the combination of the superposition and associative memory $\boldsymbol{t} = \boldsymbol{a} + \boldsymbol{b} + \boldsymbol{a} \otimes \boldsymbol{c} + \boldsymbol{b} \otimes \boldsymbol{d}$. The dimension of the used vectors is n=1000.

Combination of superposition memory and associative memory

We shall show, that also a combination of can be superposition and associative memory reliably analyzed, which will prove in our further applications as an advantageous feature of the holographic representation. Let us presume, that we have 10 conceptual vectors $\boldsymbol{a},\boldsymbol{b},\boldsymbol{c},\boldsymbol{d},...,\boldsymbol{i},\boldsymbol{j}$, and from the first four we shall construct a combination of a superposition and associative memory as follows

$$\boldsymbol{t} = \boldsymbol{a} + \boldsymbol{b} + \boldsymbol{a} \otimes \boldsymbol{c} + \boldsymbol{b} \otimes \boldsymbol{d} \tag{30}$$

By the clean up procedure we can find out, that the vector $\boldsymbol{t}$ contains as its parts the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, which we shall in the following step remove from the vector $\boldsymbol{t}$

$$\boldsymbol{t}' = \boldsymbol{t} - \boldsymbol{a} - \boldsymbol{b} = \boldsymbol{a} \otimes \boldsymbol{c} + \boldsymbol{b} \otimes \boldsymbol{d} \tag{31}$$

From the remaining superposition part we can find out by the application of the analysis (as from the associative memory) that it contains two couples $\boldsymbol{a}/\boldsymbol{c}$ and $\boldsymbol{b}/\boldsymbol{d}$, see fig. 7.

4. Sequence of symbols

The construction of the associative memory does not allow storing of structured data, the aim of this chapter is to show, that a holographic distributed representation is able to process a linear sequence of symbols, which are represented by a sequence of conceptual vectors.

To concretize our thoughts, let us study a sequence of 6 conceptual vectors of the dimension n=1000

$$sequence = \{\boldsymbol{a} \to \boldsymbol{b} \to \boldsymbol{c} \to \boldsymbol{d} \to \boldsymbol{e} \to \boldsymbol{f}\} \tag{32}$$

For these vectors we shall construct a memory vector

$$\boldsymbol{t}_0 = \boldsymbol{a} + \boldsymbol{a} \otimes \boldsymbol{b} + \boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c} + \boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c} \otimes \boldsymbol{d} + \boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c} \otimes \boldsymbol{d} \otimes \boldsymbol{e} + \boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c} \otimes \boldsymbol{d} \otimes \boldsymbol{e} \otimes \boldsymbol{f} \tag{33}$$

We know, that this vector contains the sequence of vectors coded by (33), but we do not know, which vectors these are and in what order. We shall show, that by the clean up

procedure we can from the vector $\boldsymbol{t}_0$ reconstruct the original sequence (32) step by step using the following procedure:

1. step: $\boldsymbol{a} = clean\_up(\boldsymbol{t}_0)$, $\boldsymbol{t}_1 := \boldsymbol{t}_0 - \boldsymbol{a}$,

$\tilde{\boldsymbol{t}}_1 := \boldsymbol{a}^* \otimes \boldsymbol{t}_1$,

2. step: $\boldsymbol{b} = clean\_up(\tilde{\boldsymbol{t}}_1)$, $\boldsymbol{t}_2 := \boldsymbol{t}_1 - \boldsymbol{a} \otimes \boldsymbol{b}$,

$\tilde{\boldsymbol{t}}_2 := (\boldsymbol{a} \otimes \boldsymbol{b})^* \otimes \boldsymbol{t}_2$,

3. step: $\boldsymbol{c} = clean\_up(\tilde{\boldsymbol{t}}_2)$, $\boldsymbol{t}_3 := \boldsymbol{t}_2 - \boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c}$,

$\tilde{\boldsymbol{t}}_3 := (\boldsymbol{y}_1 \otimes \boldsymbol{y}_2 \otimes \boldsymbol{y}_3)^* \otimes \boldsymbol{t}_3$,

4. step: $\boldsymbol{d} = clean\_up(\tilde{\boldsymbol{t}}_3)$, $\boldsymbol{t}_4 := \boldsymbol{t}_3 - \boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c} \otimes \boldsymbol{d}$,

$\tilde{\boldsymbol{t}}_4 := (\boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c} \otimes \boldsymbol{d})^* \otimes \boldsymbol{t}_4$,

5. step: $\boldsymbol{e} = clean\_up(\tilde{\boldsymbol{t}}_4)$, $\boldsymbol{t}_5 := \boldsymbol{t}_4 - \boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c} \otimes \boldsymbol{d} \otimes \boldsymbol{e}$,

$\tilde{\boldsymbol{t}}_5 := (\boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c} \otimes \boldsymbol{d} \otimes \boldsymbol{e})^* \otimes \boldsymbol{t}_5$,

6. step: $\boldsymbol{f} = clean\_up(\tilde{\boldsymbol{t}}_5)$.

The function *clean_up*(·) carries out the clean up process for the given vector t with respect to the set of vectors $X = \{\boldsymbol{a}, \boldsymbol{b}, ..., \boldsymbol{f}, \boldsymbol{g}, \boldsymbol{h}, ...\}$. The single steps of the reconstruction of the sequence of conceptual vectors – symbols are shown in the fig. 8, from which follows, that the process of the reconstruction of a sequence of symbols rather quickly degrades, already for the sixth vector the overlap is smaller than 0.2.

The sequence of symbols can be coded also by an associative memory, where the vector of the entry $\boldsymbol{c}_i$ specifies *i*th position of the given symbol. The above mentioned illustrative example is represented by a memory vector

$$\boldsymbol{t} = \boldsymbol{c}_1 \otimes \boldsymbol{a} + \boldsymbol{c}_2 \otimes \boldsymbol{b} + \boldsymbol{c}_3 \otimes \boldsymbol{c} + \boldsymbol{c}_4 \otimes \boldsymbol{d} + \boldsymbol{c}_5 \otimes \boldsymbol{e} + \boldsymbol{c}_6 \otimes \boldsymbol{f} \tag{34}$$

The recognition of this sequence consists in the search of the associate vector to the input vector $\boldsymbol{c}_i$, by application of the clean up process there can be constructed a „training set“

$$A_{train} = \{\boldsymbol{c}_1/\boldsymbol{a}, \boldsymbol{c}_2/\boldsymbol{b}, \boldsymbol{c}_3/\boldsymbol{c}, \boldsymbol{c}_4/\boldsymbol{d}, \boldsymbol{c}_5/\boldsymbol{e}, \boldsymbol{c}_6/\boldsymbol{f}\} \tag{35}$$

which unambiguously specifies the sequence of vectors. The advantage of such a technique is in the accuracy of recognition, which does not degrade so fast as during the original procedure based on the memory vector (33).

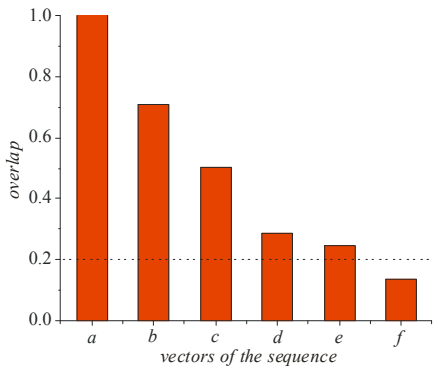

Figure 8. The overlap for single vectors with a sequence from (30), which were obtained by the reconstruction from the vector $\boldsymbol{t}_0$. It is apparent from the figure, that a degradation of reconstruction relatively quickly appears, already the sixths vector $\boldsymbol{f}$ is reconstructed with a probability smaller than 0.20.

The associative approach to the implementation of the memory for a sequence of symbols (represented by atomic vectors $\boldsymbol{a}$, $\boldsymbol{b}$,) can be easily modified into the form of so called „*stack memory*“. Associative entries $\boldsymbol{c_i}$ are determined as follows

$$\boldsymbol{c}_i = \boldsymbol{p}^i \quad (i = 1,2,...) \tag{36}$$

where $\boldsymbol{p}^i$ is the *i*th (convolutive) power of randomly generated conceptual vector $\boldsymbol{p}$, $\boldsymbol{p}^i = \boldsymbol{p}^{i-1} \otimes \boldsymbol{p}$. Then the memory vector (32) has a form

$$\boldsymbol{t} = \boldsymbol{p} \otimes \boldsymbol{x}_1 + \boldsymbol{p}^2 \otimes \boldsymbol{x}_2 + \boldsymbol{p}^3 \otimes \boldsymbol{x}_3 + ... + \boldsymbol{p}^n \otimes \boldsymbol{x}_n \tag{37}$$

where $\boldsymbol{x}_i$ are single items from memory $\{\boldsymbol{a}, \boldsymbol{b},...\}$. Such an interpreted associative memory for the sequence of symbols is called the „*stack memory*“, with the help of power entries $\boldsymbol{p}^i$ we can easily change their contents, see fig. 9. Over this memory we can define three different operations, by which we can change its contents:

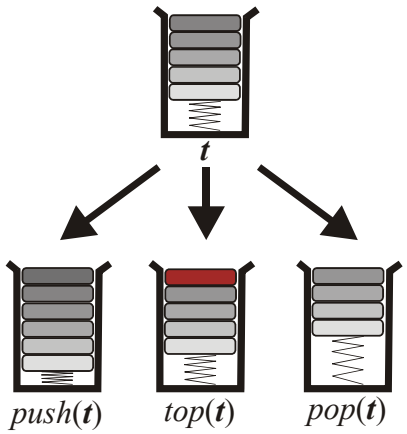

Figure 9. Three possible operators for the stack memory represented by a vector.

(1) $push(\boldsymbol{t}, \boldsymbol{x}) = \boldsymbol{p} \otimes \boldsymbol{x} + \boldsymbol{p} \otimes \boldsymbol{t}$, the new item $\boldsymbol{x}$ is placed to the top of the stack.

(2) $top(\boldsymbol{t}) = clean\_up(\boldsymbol{p}^* \otimes \boldsymbol{t})$, recognizes the top item in the stack.

(3) $pop(\boldsymbol{t}) = \boldsymbol{p}^{-1} \otimes \boldsymbol{t} - top(\boldsymbol{t})$, removes the top item from the stack.

The most problematic is the third operation, which removes the top item from the stack. The correct implementation needs an application of the exact inverse vector $\boldsymbol{p}^{-1}$, the approximation of this inverse vector by involution, $\boldsymbol{p}^{-1} \approx \boldsymbol{p}^*$, leads to a fast degradation of the stack memory.

5. Memory chunks

Memory chunks help to overcome the problems with a degradation of memory for a sequence of symbols (see chap. 4). Let us have a set of conceptual vectors $S = \{\boldsymbol{a}, \boldsymbol{b}, ..., \boldsymbol{k}, \boldsymbol{l}, ...\}$, this set shall be divided into disjoint subsets - *chunks*

$$S = S_1 \cup S_2 \cup S_3 \cup S_4 \cup ... \qquad (S_i \cup S_j = \varnothing, \text{pre } i \neq j) \tag{38}$$

Let's study a set $S = \{a, b, c, d, e, f, g, h\}$, its decomposition into chunks looks as follows (see fig. 10)

$$S_1 = \{a,b,c\},\ S_2 = \{d,e\},\ S_3 = \{f\},\ \text{and}\ S_4 = \{g,h\} \tag{39}$$

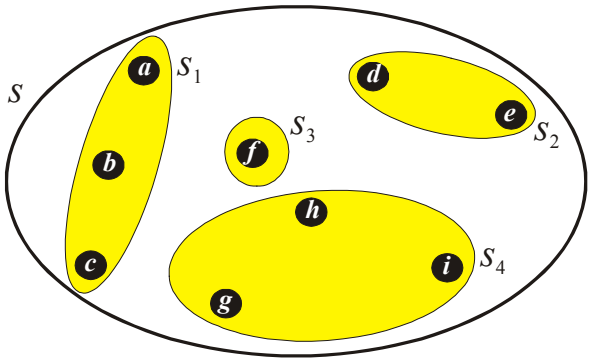

Figure 10. Illustration of the decomposition of the set *S* into disjoint chunks, see (38).

Chunks are represented by a vector, which represents a sequence of chunks $\{\boldsymbol{s}_1 \to \boldsymbol{s}_2 \to \boldsymbol{s}_3 \to \boldsymbol{s}_3\}$

$$\boldsymbol{t} = \boldsymbol{s}_1 + \boldsymbol{s}_1 \otimes \boldsymbol{s}_2 + \boldsymbol{s}_1 \otimes \boldsymbol{s}_2 \otimes \boldsymbol{s}_3 + \boldsymbol{s}_1 \otimes \boldsymbol{s}_2 \otimes \boldsymbol{s}_3 \otimes \boldsymbol{s}_4 \tag{40b}$$

while single chunks are defined by corresponding sequences of vectors (see fig. 11)

$$\boldsymbol{s}_1 = \boldsymbol{a} + \boldsymbol{a} \otimes \boldsymbol{b} + \boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c} \tag{40c}$$

$$\boldsymbol{s}_2 = \boldsymbol{d} + \boldsymbol{d} \otimes \boldsymbol{e} \tag{40d}$$

$$\boldsymbol{s}_3 = \boldsymbol{f} \tag{40e}$$

$$\boldsymbol{s}_4 = \boldsymbol{g} + \boldsymbol{g} \otimes \boldsymbol{h} \tag{40f}$$

The processing of the memory chunks can be divided into two steps:

1. step – by a clean up process we shall identify chunks contained in $\boldsymbol{t}$ (we presume, that the clean up process has the set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ from the end of the 2nd chapter, where this process was specified, enlarged also by chunks $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \boldsymbol{s}_4$, i.e. in our illustrative example $X = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, \boldsymbol{f}, \boldsymbol{g}, \boldsymbol{h}, \boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \boldsymbol{s}_4\}$).

2. step – the identified chunks are further processed by the clean up technique.

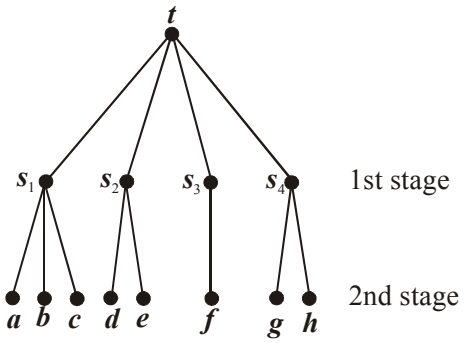

Figure 11. The chunking of 8 conceptual vectors onto 4 chunks. In the 1st stage the clean up process identifies the chunks, which are then in the 2nd step further analyzed up to vectors describing elementary concepts.

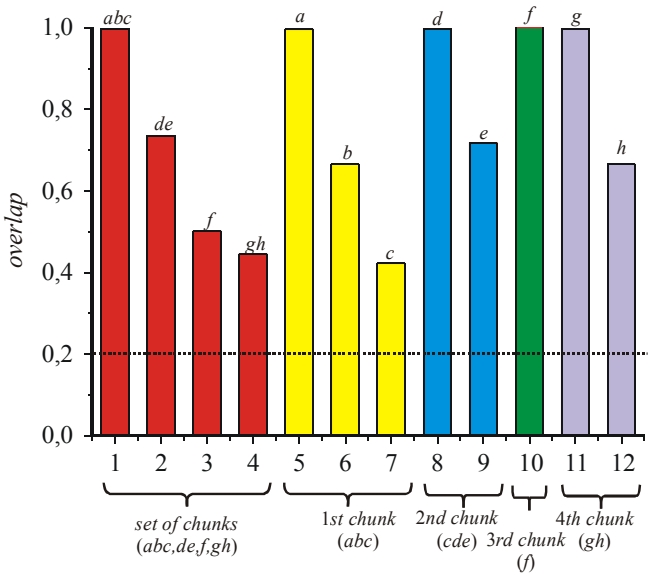

Figure 12. The representation of the two-step clean up process, where in the first step the chunks are identified, while in the second step there are identified the vectors corresponding to atomic concepts from the already identified chunks.

The result of the two-step process of clean up is shown in the fig. 11. It is evident from this figure, that in the case of a long sequence of conceptual vectors a fast degradation during the clean up process can be partially overcome by the chunking of concepts onto chunks, which are at the highest level separately coded.

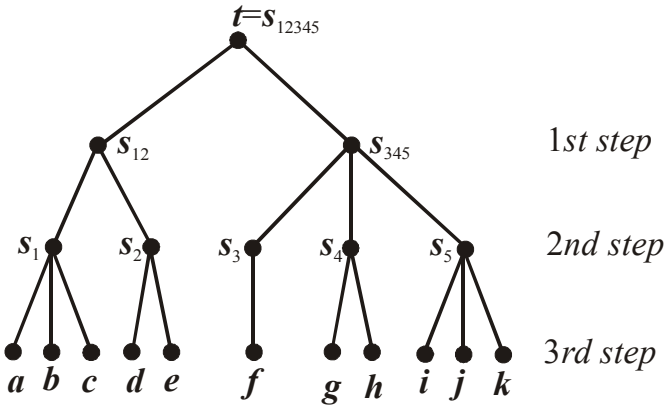

Figure 13. Illustration of chunks of higher order, where some of the used chunks are composed from simpler chunks.

The chunking method can be generalized, so that the chunks of higher order are created, i.e. chunks composed of chunks, see fig. 12, where single chunks are defined as follows (the used vectors of concepts $\boldsymbol{a}$, $\boldsymbol{b}$, ..., $\boldsymbol{c}$ have the dimension n=1000)

$$\boldsymbol{s}_1 = \boldsymbol{a} + \boldsymbol{a} \otimes \boldsymbol{b} + \boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c} \tag{41a}$$

$$\boldsymbol{s}_2 = \boldsymbol{d} + \boldsymbol{d} \otimes \boldsymbol{e} \tag{41b}$$

$$\boldsymbol{s}_3 = \boldsymbol{f} \tag{41c}$$

$$\boldsymbol{s}_4 = \boldsymbol{g} + \boldsymbol{g} \otimes \boldsymbol{h} \tag{41d}$$

$$\boldsymbol{s}_5 = \boldsymbol{i} + \boldsymbol{i} \otimes \boldsymbol{j} + \boldsymbol{i} \otimes \boldsymbol{j} \otimes \boldsymbol{k} \tag{41e}$$

$$\boldsymbol{s}_{12} = \boldsymbol{s}_1 + \boldsymbol{s}_1 \otimes \boldsymbol{s}_2 \tag{41f}$$

$$\boldsymbol{s}_{345} = \boldsymbol{s}_3 + \boldsymbol{s}_3 \otimes \boldsymbol{s}_4 + \boldsymbol{s}_3 \otimes \boldsymbol{s}_4 \otimes \boldsymbol{s}_5 \tag{41g}$$

$$\boldsymbol{t} = \boldsymbol{s}_{12345} = \boldsymbol{s}_{12} + \boldsymbol{s}_{12} \otimes \boldsymbol{s}_{345} \tag{41h}$$

The process of clean up of the chunked memory trace $\boldsymbol{t}$ specified by (41h) is shown in the fig. 13. In the 1st step we analyze the trace $\boldsymbol{t}$, its analysis tells us, that the trace $\boldsymbol{t}$ contains two chunks $\boldsymbol{s}_{12}$ a $\boldsymbol{s}_{345}$. In the second step we analyze chunks from the previous first step, and we recognize their contents as chunks $\boldsymbol{s}_1$, $\boldsymbol{s}_2$,..., $\boldsymbol{s}_5$. In the last third step we analyze chunks from the previous step, which already contain atomic conceptual vectors $\boldsymbol{a}$, $\boldsymbol{b}$, ..., $\boldsymbol{k}$. The overlaps of the resulting vectors in the clean up process are shown at fig. 14.

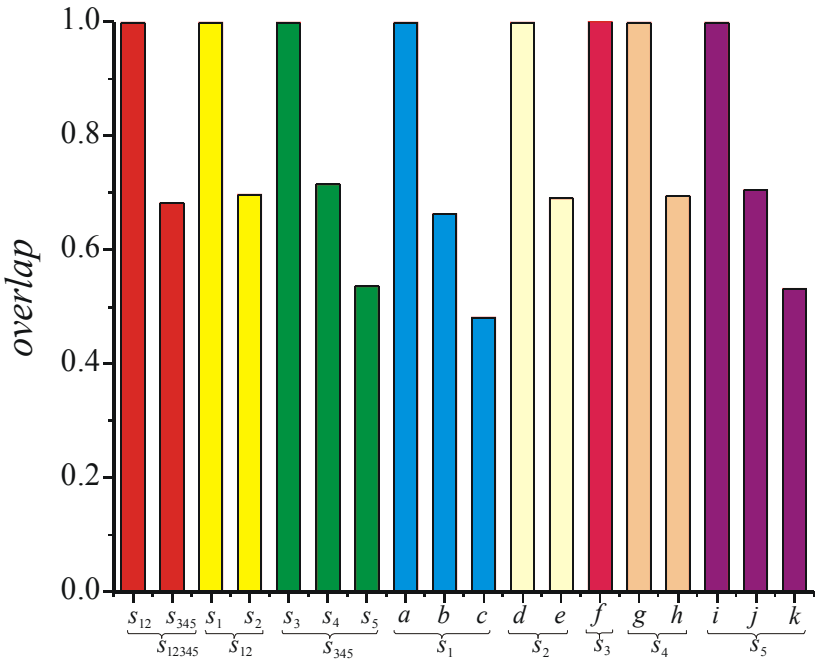

Figure 14. Illustration of a 3-step clean up of the chunked trace $\boldsymbol{t}$ specified by the formula (41h). In the first step the trace $\boldsymbol{t}$ is analyzed, it is found, that it contains two chunks $\boldsymbol{s}_{12}$ and $\boldsymbol{s}_{345}$. In the second step the two chunks from the previous step are analyzed, and found to contain chunks $\boldsymbol{s}_1$, $\boldsymbol{s}_2$, ..., $\boldsymbol{s}_5$. In the last third step the 5 chunks identified in the previous step are successively analyzed. They contain the vectors $\boldsymbol{a}$, $\boldsymbol{b}$, ..., $\boldsymbol{k}$.

The presented illustrative examples show, that the memory chunk approach represents an effective way to overcome fast degradation of the original version of a successive analysis of the vector (33). By combining of several conceptual vectors into a chunk, we shall gain a simple opportunity to extend our ability to analyze correctly greater sets of conceptual vectors. The chunking process can have several hierarchic levels, which removes the limits from our ability to store and recall conceptual vectors.

6. Coding of relations

Holographic reduced representation can serve also as a suitable means for encoding relations (predicates). Let us study a binary relation $P(x,y)$, when the Pascal code is used, this relation is formally specified by the head

$$function\ P\left(x : type_1; y : type_2\right) : type_3 \tag{42}$$

The single arguments of the relation are specified by the types $type_1$ and $type_2$, which specify the domain, over which are these variables defined; similarly also the relation P itself is understood as a function, which domain of values is specified by the type $type_3$. In many cases the domain of variables and also the domain of the relation itself are equal to each other;

therefore their specifications can be omitted, which substantially reduces the holographic representation of relations. The reduced form of relation (42) looks as follows

$$function\ P(x;y) \tag{43}$$

where we know in advance the type of variables x, y, and also the type of the relation P itself. The holographic representation of the relation (42) can have the following form

$$\begin{aligned} \boldsymbol{t} &= \boldsymbol{P} + \boldsymbol{variable}_1 + \boldsymbol{variable}_2 + \\ &\boldsymbol{P} \otimes \left(\boldsymbol{type}_3 + \boldsymbol{variable}_1 \otimes \left(\boldsymbol{x} + \boldsymbol{type}_1\right) + \boldsymbol{variable}_2 \otimes \left(\boldsymbol{y} + \boldsymbol{type}_2\right)\right) \end{aligned} \tag{44}$$

Their decoding is carried out step-by-step. In the first step we use the clean up procedure to recognize the name (identifier) of the relation $\boldsymbol{P}$ and also the names (identifiers) of its variables $\boldsymbol{x}$ and $\boldsymbol{y}$. In the second step we identify the type $\boldsymbol{type}_3$ of the relation $\boldsymbol{P}$, in the last, third step we use previous results to identify variables $\boldsymbol{x}$, $\boldsymbol{y}$ and also their types $\boldsymbol{type}_1$ and $\boldsymbol{type}_2$. In many cases the representation of the relation $P(x,y)$ is satisfactory in the following simplified form (see (43))

$$\boldsymbol{t} = \boldsymbol{P} + \boldsymbol{variable}_1 \otimes \boldsymbol{x} + \boldsymbol{variable}_2 \otimes \boldsymbol{y} \tag{45}$$

The chosen method of the holographic representation of relation can be easily generalized also for more complex (higher order) relations, where the variables are predicates as well, e.g. $P\left(x,Q\left(y,z\right)\right)$, where the „inner" predicate Q is characterized by

$$function\ Q\left(y:type_3;\ z:type_4\right):type_5 \tag{46}$$

In order to create a higher order relation $P\left(x,Q\left(y,z\right)\right)$, we must presume a type compatibility of the second variable of the relation P and of the type of relation Q, i.e. $type_2$=$type_4$. In the simplified approach, where all the types are the same, it is not necessary to distinguish the types of single variables and the relations themselves. A simplified holographic representation of relation (46) has the following form

$$\boldsymbol{t'} = \boldsymbol{Q} + \boldsymbol{variable}_1 \otimes \boldsymbol{y} + \boldsymbol{variable}_2 \otimes \boldsymbol{z} \tag{47}$$

By exchanging the representation (47) for the variable $\boldsymbol{y}$ in the representation (45) we get the following resulting representation of the higher order relation $P\left(x,Q\left(y,z\right)\right)$

$$\begin{aligned} \boldsymbol{t} &= \boldsymbol{P} + \boldsymbol{variable}_1 \otimes \boldsymbol{x} + \boldsymbol{variable}_2 \otimes \left(\boldsymbol{Q} + \boldsymbol{variable}_3 \otimes \boldsymbol{y} + \boldsymbol{variable}_4 \otimes \boldsymbol{z}\right) \\ &= \boldsymbol{P} + \boldsymbol{variable}_1 \otimes \boldsymbol{x} + \boldsymbol{variable}_2 \otimes \boldsymbol{Q} + \\ &\quad \boldsymbol{variable}_2 \otimes \boldsymbol{variable}_3 \otimes \boldsymbol{y} + \boldsymbol{variable}_2 \otimes \boldsymbol{variable}_4 \otimes \boldsymbol{z} \end{aligned} \tag{48}$$

1st illustrative example –a similarity between geometric figures

In the figure 15 there are presented 48=6×8 geometric patterns, which contain either in horizontal or in vertical settings two objects, which moreover can be of two sizes, small and big. Let us mark holographic representations of corresponding atomic concepts as follows:

Objects: ***tr*** (triangle), ***sq*** (square), ***ci***(circle), ***st*** (star)
Unary relations: ***sm*** (small), ***lg*** (large)
Binary relations: ***hor*** (horizontal), ***ver*** (vertical)
Variables: ***ver_var*$_1$** (1st variable for binary relation ***ver***), ***ver_var*$_1$** (2nd variable for binary relation ***ver***), ***hor_ver*$_1$** (1st variable for binary relation ***hor***), ***hor_ver*$_2$** (2nd variable for binary relation ***hor***)

Single figures from z fig. 15 are characterized by relations given in the following table.

row	specification
1	***ver***(***lg***(***x***),***lg***(***y***))
2	***hor***(***lg***(***x***),***lg***(***y***))
3	***hor***(***sm***(***x***),***lg***(***y***)) and ***hor***(***lg***(***x***),***sm***(***y***))
4	***ver***(***sm***(***x***),***lg***(***y***)) and ***ver***(***lg***(***x***),***sm***(***y***))
5	***ver***(***sm***(***x***),***sm***(***y***))
6	***hor***(***sm***(***x***),***sm***(***y***))

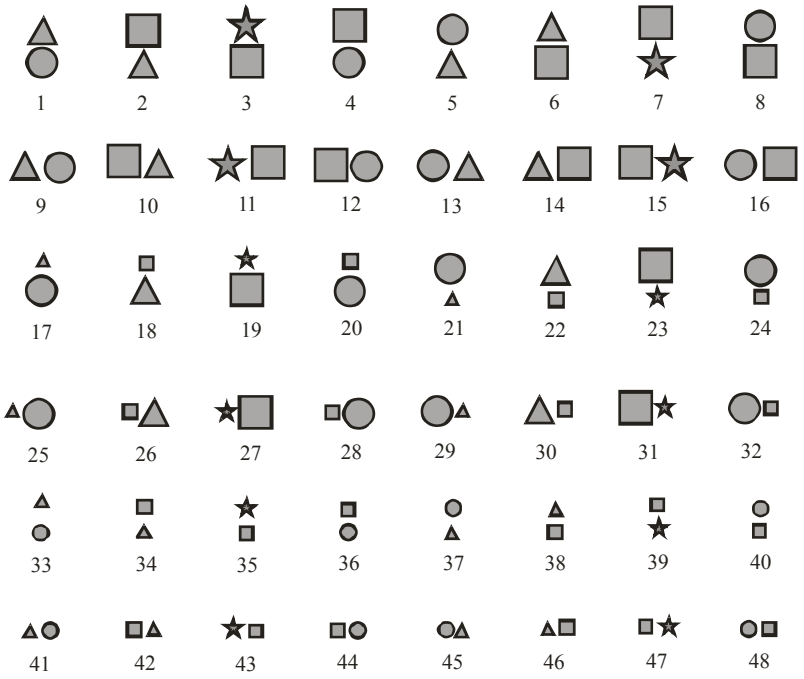

Figure 15. A set of 48 similar figures, which contain two objects, placed either next to each other, or above each other and the objects are either small or big. Every column contains a couple of similar objects, which differ only by their placement or size.

Holographic representations of single cases from this table have the following form (compare with the equation (45)).

$$
\begin{aligned}
\boldsymbol{t}_{1,x,y} &= \boldsymbol{ver} + \left\langle \boldsymbol{ver\_var}_1 \otimes \boldsymbol{lg} \otimes \boldsymbol{x} + \boldsymbol{ver\_var}_2 \otimes \boldsymbol{lg} \otimes \boldsymbol{y} \right\rangle \\
\boldsymbol{t}_{2,x,y} &= \boldsymbol{hor} + \left\langle \boldsymbol{hor\_var}_1 \otimes \boldsymbol{lg} \otimes \boldsymbol{x} + \boldsymbol{hor\_var}_2 \otimes \boldsymbol{lg} \otimes \boldsymbol{y} \right\rangle \\
\boldsymbol{t}_{3,x,y} &= \begin{cases} \boldsymbol{ver} + \left\langle \boldsymbol{ver\_var}_1 \otimes \boldsymbol{lg} \otimes \boldsymbol{x} + \boldsymbol{ver\_var}_2 \otimes \boldsymbol{sm} \otimes \boldsymbol{y} \right\rangle \\ \boldsymbol{ver} + \left\langle \boldsymbol{ver\_var}_1 \otimes \boldsymbol{sm} \otimes \boldsymbol{x} + \boldsymbol{ver\_var}_2 \otimes \boldsymbol{lg} \otimes \boldsymbol{y} \right\rangle \end{cases} \\
\boldsymbol{t}_{4,x,y} &= \begin{cases} \boldsymbol{hor} + \left\langle \boldsymbol{hor\_var}_1 \otimes \boldsymbol{lg} \otimes \boldsymbol{x} + \boldsymbol{hor\_var}_2 \otimes \boldsymbol{sm} \otimes \boldsymbol{y} \right\rangle \\ \boldsymbol{hor} + \left\langle \boldsymbol{hor\_var}_1 \otimes \boldsymbol{sm} \otimes \boldsymbol{x} + \boldsymbol{hor\_var}_2 \otimes \boldsymbol{lg} \otimes \boldsymbol{y} \right\rangle \end{cases} \\
\boldsymbol{t}_{5,x,y} &= \boldsymbol{ver} + \left\langle \boldsymbol{ver\_var}_1 \otimes \boldsymbol{sm} \otimes \boldsymbol{x} + \boldsymbol{ver\_var}_2 \otimes \boldsymbol{sm} \otimes \boldsymbol{y} \right\rangle \\
\boldsymbol{t}_{6,x,y} &= \boldsymbol{hor} + \left\langle \boldsymbol{hor\_var}_1 \otimes \boldsymbol{sm} \otimes \boldsymbol{x} + \boldsymbol{hor\_var}_2 \otimes \boldsymbol{sm} \otimes \boldsymbol{y} \right\rangle
\end{aligned}
\tag{49}
$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are holographic representations of single objects (***tr***, ***sq***, ***ci***, ***st***) and the bracket $\langle \boldsymbol{u} \rangle$ indicates, that the vector $\boldsymbol{u}$ is normalized. The similarity between single figures is determined by the overlap of their holographic representations

$$similarity\left(X,X'\right)=overlap\left(\boldsymbol{t},\boldsymbol{t}'\right) \tag{50}$$

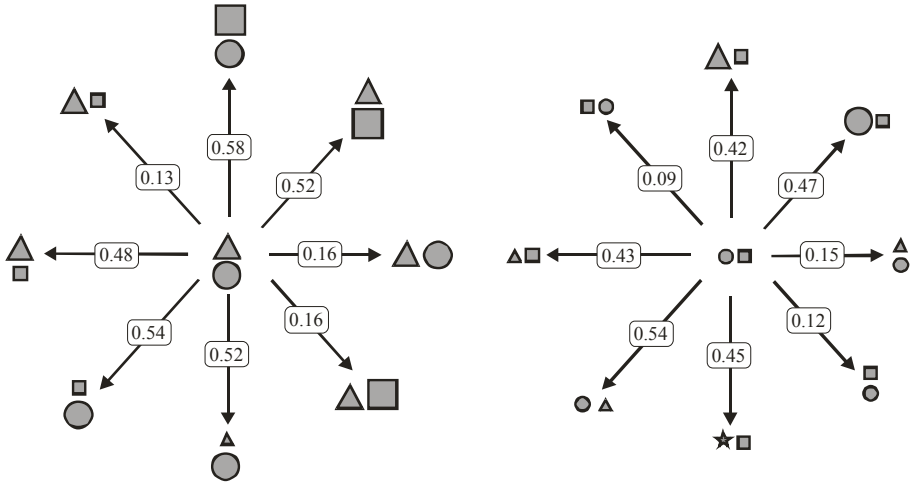

Figure 16. Illustrative presentation of similar figures for two chosen figures 1 and 48 (see fig. 16). Single arrows are marked by the overlap between the figures calculated by formula (50).

The obtained results are shown in the fig. 16. The dominant feature controlling similarity value is the horizontal or vertical arrangement of objects. The overlap (i.e. also the similarity) between two figures, which have different arrangement is usually smaller than 0.1.

In general, holographic reduced representation allows fairly simple determination of similarity of objects specified by a predicate structure (42) or by its generalization through further nested predicates (see (48)). This possibility opens new horizons on future developments in fundamental methods of search for similar objects or analogies, which are considered very difficult problems for artificial intelligence requiring special symbolic techniques [7].

2nd illustrative example – similarity between binary numbers

We shall study similarity between binary numbers of the length 3, which are represented by a sequence $\left(\alpha_1\alpha_2\alpha_3\right)\in\left\{0,1\right\}^3$. This number can be understood as an ordered triple of binary symbols, which are in the distributed representation represented as follows (see (33))

$$\boldsymbol{t}_{(\alpha_1\alpha_2\alpha_3)}=\boldsymbol{t}_{\alpha_1}+\boldsymbol{t}_{\alpha_1}\otimes\boldsymbol{t}_{\alpha_2}+\boldsymbol{t}_{\alpha_1}\otimes\boldsymbol{t}_{\alpha_2}\otimes\boldsymbol{t}_{\alpha_3} \tag{51a}$$

$$\boldsymbol{t}_{\alpha}=\begin{cases}\boldsymbol{zero} & \left(for\ \alpha=0\right)\\ \boldsymbol{one} & \left(for\ \alpha=1\right)\end{cases} \tag{51b}$$

where ***zero*** and ***one*** are distributed representations of numbers ‘0’ or ‘1’. For example, a binary number (101) is represented as follows

$$\boldsymbol{t}_{(101)}=\boldsymbol{one}+\boldsymbol{one}\otimes\boldsymbol{zero}+\boldsymbol{one}\otimes\boldsymbol{zero}\otimes\boldsymbol{one} \tag{52}$$

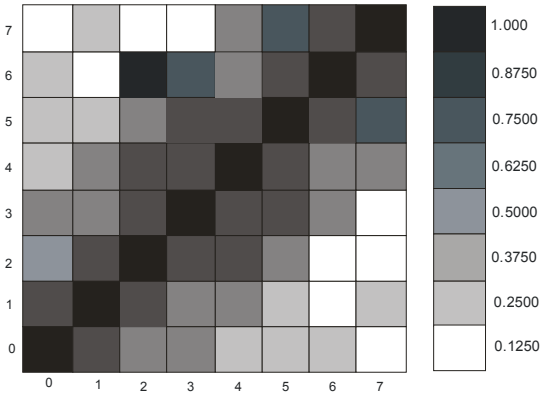

Figure 17. Graphic representation of similarity between distributed representations of binary numbers, the size of overlap between couples is proportional to the brightness of the corresponding square area (the areas on the skew diagonal are the brightest ones). The darkest areas are in the bottom lower corner and in the left upper corner. It corresponds to the fact that these areas are assigned to representations of maximally distant couples of numbers.

The similarity between single representations is reflected also by a similarity between corresponding binary numbers; the representation of two close binary numbers is inversely proportional to their distance (e.g. to the absolute value of their difference). In the following table there are given the similarities between representations of binary numbers, which were calculated from their overlap (23).

	000	001	010	011	100	101	110	111
000	1.00	0.74	0.50	0.46	0.21	0.16	0.14	0.11
001	0.74	1.00	0.71	0.42	0.45	0.15	0.12	0.15
010	0.50	0.71	1.00	0.75	0.70	0.43	0.09	0.08
011	0.46	0.42	0.75	1.00	0.46	0.70	0.37	0.04
100	0.21	0.45	0.70	0.46	1.00	0.74	0.43	0.39
101	0.16	0.15	0.43	0.70	0.74	1.00	0.70	0.35
110	0.14	0.12	0.09	0.37	0.43	0.70	1.00	0.69
111	0.11	0.15	0.08	0.04	0.39	0.35	0.69	1.00

The maximum similarity is between couples of representations assigned to two neighboring integers, minimum similarity of two representations occurs, if the corresponding numbers have a maximum distance, which equals 7. These values from the table are graphically represented in the fig. 17.

This simple illustrative example shows, that in the framework of the holographic distributed representation one can use (at least potentially) associative representations of the type (28), where associative cues correspond to numbers. It means that in this distributed approach there exists a possibility of associative simulation of an arbitrary function, which substantially increases the potential of the method to be used universally.

7. Reasoning by modus ponens and modus tollens

Simulation of reasoning processes (inference) belongs to the basic problems, which are repeatedly solved in artificial intelligence and cognitive science [15]. Fodor‘s critique of connectionism [19] was based precisely on the brash conclusion, that artificial neural networks are not able to simulate higher cognitive activities, which are purported to be an exclusive domain of the classical symbolic approach. This Fodor’s opinion was proved to be incorrect, further development of theory of neural networks showed, that connectionism is

a universal computational tool, which does not have limits of applicability, it does not have domains of inapplicability, which would be forbidden for it. Of course, it can transpire, that in some domains its application is extremely cumbersome and exceedingly complicated, that there exist other approaches, which in the given domain provide substantially simpler and direct solution, than the one provided by neural networks.

In this chapter we shall show a possibility of representation of two basic modes of deductive reasoning of modal logic,

$$\begin{array}{l} p \Rightarrow q \\ \underline{p\quad\quad} \\ q \end{array} \qquad \text{and} \qquad \begin{array}{l} p \Rightarrow q \\ \underline{\overline{q}\quad\quad} \\ \overline{p} \end{array} \tag{53}$$

which are called *modus ponens* resp. *modus tollens*. These modes of reasoning are equivalent to the following tautologies of the predicate logic

$$\big((p \Rightarrow q) \wedge p\big) \Rightarrow q \tag{54a}$$

$$\big((p \Rightarrow q) \wedge \overline{q}\big) \Rightarrow \overline{p} \tag{54b}$$

Implication $'\Rightarrow'$ can be understood as a binary relation, which can be in holographic distribution represented like this (see formula (47))

$$\boldsymbol{t}_{p\Rightarrow q} = \boldsymbol{op} \otimes \boldsymbol{impl} + \boldsymbol{var}_1 \otimes \boldsymbol{p} + \boldsymbol{var}_2 \otimes \boldsymbol{q} \tag{55}$$

which contains a sum of three parts, the first part specifies the type of relation (implication), the second and third parts specify the first (antecedent) resp. the second (consequent) variable of the relation of implication. This conceptual vector representing relation of implication can be transformed as follows

$$\tilde{\boldsymbol{t}}_{p\Rightarrow q} = \boldsymbol{t}_{p\Rightarrow q} \otimes \boldsymbol{T} \tag{56a}$$

where

$$\boldsymbol{T} = \boldsymbol{var}_1^* \otimes \boldsymbol{p}^* \otimes \boldsymbol{p}^* \otimes \boldsymbol{q} + \boldsymbol{var}_2^* \otimes \boldsymbol{q}^* \otimes \overline{\boldsymbol{q}}^* \otimes \overline{\boldsymbol{p}} \tag{56b}$$

The transformed representation of implication is represented by a sum of two associated couples

$$\tilde{\boldsymbol{t}}_{p\Rightarrow q} \approx \boldsymbol{p}^* \otimes \boldsymbol{q} + \overline{\boldsymbol{q}}^* \otimes \overline{\boldsymbol{p}} \tag{57}$$

which gives the holographic representation of the rules *modus ponens* and *modus tollens*

$$\boldsymbol{p} \otimes \tilde{\boldsymbol{t}}_{p\Rightarrow q} \approx \boldsymbol{q} \tag{58a}$$

$$\overline{\boldsymbol{q}} \otimes \tilde{\boldsymbol{t}}_{p\Rightarrow q} \approx \overline{\boldsymbol{p}} \tag{58b}$$

The first formula (58a) can be understood as a holographic representation of modus ponens (see (53) and (54a)), while the other formula is a holographic representation of modus tollens (see (53) and (54b)).

A similar result can be obtained also by an alternative approach, which is based on the disjunctive form of implication

$$(p \Rightarrow q) \equiv (\overline{p} \vee q) \tag{59}$$

The distributed representation of implication in this alternative form can be expressed by

$$\boldsymbol{t}_{\overline{p}\vee q} = \boldsymbol{op} \otimes \boldsymbol{disj} + \boldsymbol{var}_1 \otimes \overline{\boldsymbol{p}} + \boldsymbol{var}_2 \otimes \boldsymbol{q} \tag{60}$$

By a transformation of this representation we can get (see (55))

$$\tilde{\boldsymbol{t}}_{\overline{p}\vee q} = \boldsymbol{t}_{\overline{p}\vee q} \otimes \boldsymbol{T} \approx \overline{\boldsymbol{p}} \otimes \overline{\boldsymbol{q}}^* + \boldsymbol{q} \otimes \boldsymbol{p}^* \tag{61a}$$

where

$$\boldsymbol{T} = \boldsymbol{var}_1^* \overline{\boldsymbol{q}}^* + \boldsymbol{var}_2^* \boldsymbol{p}^* \tag{61a}$$

This transformation is much simpler than the one in the previous case (56b). The rules modus ponens and modus tollens are now realized by formulas similar to (58a-b). Moreover, we get also the following two „rules“

$$\boldsymbol{q}^* \otimes \tilde{\boldsymbol{t}}_{\bar{p}\vee q} \approx \boldsymbol{p}^* \tag{62a}$$

$$\bar{\boldsymbol{p}}^* \otimes \tilde{\boldsymbol{t}}_{\bar{p}\vee q} \approx \bar{\boldsymbol{q}}^* \tag{62b}$$

which remind us of the well known fallacies

$$\begin{array}{l} p \Rightarrow q \\ q \\ \hline p \end{array} \quad \text{a} \quad \begin{array}{l} p \Rightarrow q \\ \bar{p} \\ \hline \bar{q} \end{array} \tag{63}$$

that are known as „*affirming the consequent*“ resp. „*denying the antecedent*“. This fault is caused by the fact, that the transformed representations of implications $\tilde{\boldsymbol{t}}_{p\Rightarrow q}$ and $\tilde{\boldsymbol{t}}_{\bar{p}\vee q}$ are not identical, the representation $\tilde{\boldsymbol{t}}_{\bar{p}\vee q}$ leads to unexpected results (63), which represent erroneous modes of reasoning (which are however often used by people without knowledge of principles of logic).

8. Predicate logic

We shall further deal with a simple form of predicate logic, which is based on unary predicates, $P(x)$, where the distributed representation has a form (see chapter 6)

$$\boldsymbol{t}_{P(x)} = \boldsymbol{pred} \otimes \boldsymbol{P} + \boldsymbol{pred\_var} \otimes \boldsymbol{x} \tag{64}$$

The connection of this predicate with the universal quantifier, $(\forall x)P(x)$, can be represented in the following way

$$\boldsymbol{t}_{(\forall x)} = \boldsymbol{uni\_quant} \otimes \boldsymbol{uni} + \boldsymbol{uni\_quant\_var} \otimes \boldsymbol{x} \tag{65a}$$

$$\boldsymbol{t}_{(\forall x)P(x)} = \boldsymbol{t}_{(\forall x)} + \boldsymbol{t}_{(\forall x)} \otimes \left(\boldsymbol{pred} \otimes \boldsymbol{P} + \boldsymbol{pred\_var} \otimes \boldsymbol{x} \right) \tag{65b}$$

Both conceptual vectors $\boldsymbol{t}_{P(x)}$ a $\boldsymbol{t}_{(\forall x)P(x)}$ can be recognized and extracted by a clean up procedure.

This process is unnecessarily complicated for our purposes of further study of reasoning processes in the framework of predicate logic and their distributed representation; the application of the conceptual vector $\boldsymbol{t}_{(\forall x)}$ for the representation of the symbol $(\forall x)$ basically only unnecessarily complicates the process of analysis of composed conceptual vectors containing as a constituent $\boldsymbol{t}_{(\forall x)}$. We shall therefore cease to use the symbol $(\forall x)$ explicitly, its meaning will be substituted by usage of a „*universal variable*“ x, i.e. predicate $P(x)$ containing the universal variable x is interpreted as $(\forall x)P(x)$, we can therefore with a certain caution use a „formula“ $(\forall x)P(x) \equiv P(x)$.

In the predicate logic there exists a rule of universal instantiation, which concretizes a predicate with a universal quantifier onto a predicate with a concrete variable *a,* $(\forall x)P(x) \Rightarrow P(a)$, which is a result of a simple tautology of propositional logic $\big((p \wedge q) \wedge p\big) \Rightarrow q$. With an application of the universal variable x we shall rewrite this concretization into a simpler form

$$P(x) \Rightarrow P(a) \tag{66}$$

We shall construct a distributed representation of this universal instantiation of a simple unary replicator by a transformation vector $\boldsymbol{T}$, see equations (56-58). The distributed representation (66) looks as follows

$$\boldsymbol{t}_{P(a)} \approx \boldsymbol{t}_{P(x)} \otimes \boldsymbol{T} \tag{67}$$

where $\boldsymbol{T} = \boldsymbol{x}^* \otimes \boldsymbol{a}$, then

$$\boldsymbol{t}_{P(a)} \approx \boldsymbol{t}_{P(x)} \otimes \boldsymbol{x}^* \otimes \boldsymbol{a} \tag{68}$$

We can recapitulate our thoughts in saying that distributed representation of the universal instantiation is concretized by a transformation vector $\boldsymbol{T}$, which helps to substitute a universal variable $\boldsymbol{x}$ by a „specified“ variable a.

We shall use this simplified representation of quantified predicates to study so called *generalized modus ponens* and *generalized modus tollens*

$$\frac{(\forall x)\big(P(x) \Rightarrow Q(x)\big) \quad P(a)}{Q(a)} \quad \text{a} \quad \frac{(\forall x)\big(P(x) \Rightarrow Q(x)\big) \quad \bar{Q}(a)}{\bar{P}(a)} \tag{69a}$$

or in a simplified form using a universal variable x

$$\frac{P(x) \Rightarrow Q(x) \quad P(a)}{Q(a)} \quad \text{a} \quad \frac{P(x) \Rightarrow Q(x) \quad \bar{Q}(a)}{\bar{P}(a)} \tag{69b}$$

These generalized schemes of deductive reasoning follow directly from their standard sentential form (53) and concretization (63). The distributed representation of the main (top) premise of these rules has a form

$$\begin{aligned} \boldsymbol{t}_{P(x)\Rightarrow Q(x)} = \boldsymbol{op} \otimes \boldsymbol{impl} + \boldsymbol{var}_1 \otimes \big(\boldsymbol{pred} \otimes \boldsymbol{P} + \boldsymbol{pred\_var} \otimes \boldsymbol{x}\big) + \\ \boldsymbol{var}_2 \otimes \big(\boldsymbol{pred} \otimes \boldsymbol{Q} + \boldsymbol{pred\_var} \otimes \boldsymbol{y}\big) \end{aligned} \tag{70}$$

The concretization of implication $P(x) \Rightarrow Q(x)$ onto $P(a) \Rightarrow Q(b)$, can be formally expressed by an implication (see (66))

$$\big(P(x) \Rightarrow Q(x)\big) \Rightarrow \big(P(a) \Rightarrow Q(a)\big) \tag{71}$$

where the right hand side has the following distributed representation

$$\boldsymbol{t}_{P(a)\Rightarrow Q(a)} = \boldsymbol{op} \otimes \boldsymbol{impl} + \boldsymbol{var}_1 \otimes \big(\boldsymbol{P} + \boldsymbol{a}\big) + \boldsymbol{var}_2 \otimes \big(\boldsymbol{Q} + \boldsymbol{b}\big) \tag{72}$$

Similarly as in the introductory illustrative example (see (67)), this transfer expressed by an implication (71) can be in a distributed representation written by the following transformation

$$\tilde{\boldsymbol{t}}_{P(a)\Rightarrow Q(a)} = \boldsymbol{t}_{P(a)\Rightarrow Q(a)} \otimes \boldsymbol{T} \approx \big(\boldsymbol{P} + \boldsymbol{a}\big)^* \otimes \big(\boldsymbol{Q} + \boldsymbol{b}\big) + \big(\bar{\boldsymbol{Q}} + \boldsymbol{b}\big)^* \otimes \big(\bar{\boldsymbol{P}} + \boldsymbol{a}\big) \tag{73}$$

where the new transformed distributed representation $\tilde{\boldsymbol{t}}_{P(a)\Rightarrow Q(a)}$ satisfies the formulas, which represent the rules (69) modus ponens resp. modus tollens

$$\big(\boldsymbol{P} + \boldsymbol{a}\big) \otimes \tilde{\boldsymbol{t}}_{P(a)\Rightarrow Q(a)} \approx \big(\boldsymbol{Q} + \boldsymbol{b}\big) \tag{74a}$$

$$\big(\bar{\boldsymbol{Q}} + \boldsymbol{b}\big) \otimes \tilde{\boldsymbol{t}}_{P(a)\Rightarrow Q(a)} \approx \big(\bar{\boldsymbol{P}} + \boldsymbol{a}\big) \tag{74b}$$

Illustrative example – modeling of reflexive reasoning

In this illustrative example we shall show, that the holographic distributed representation provides formal tools, which can be used to simulate the reasoning process based on generalized modus ponens (69). This process was widely studied by Shastri a Ajjanagadde

[15] by the connectionist system called SHRUTI, which was able to simulate reflexive reasoning based on predicate logic. Similar results are achieved also by a formalism of holographic distributed representation.

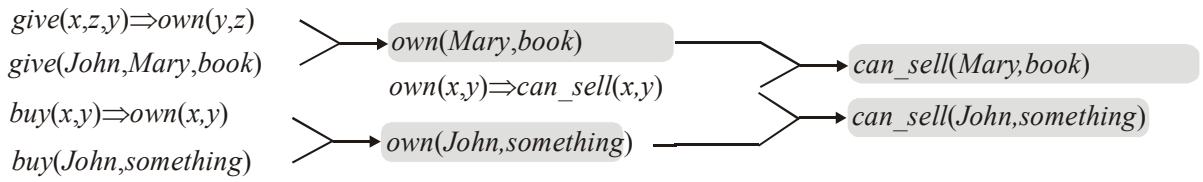

Figure 18. Illustration of an application of the generalized rule modus ponens (66) for deduction or knowledge discovery (marked by gray shading and also by incoming arrows) from implications (1-3) and from input facts (a-c), marked by outgoing arrows.

Let us have a formal system containing three general rules (see fig.18):

(1) $give(x,y,z) \Rightarrow own(y,z)$, *type x* : donor; *type y* : acceptor; *type z* : object,

(2) $buy(y,z) \Rightarrow own(y,z)$, *type y* : buyer; *type z* : object,

(3) $own(y,z) \Rightarrow can\_sell(y,z)$, *type y* : owner; *type z* : object,

and three observations (facts)

(a) $give(John,Mary,book)$,

(b) $own(Mary,book)$

(c) $buy(John,something)$.

What are the deductive conclusions of this system? The results are shown in the figure 18, we shall now deduce them with an application of distributed representation based on conceptual vectors and operations over them.

Let us analyze the first generalized modus ponens from the figure 18

$$\frac{give(x,y,z) \Rightarrow own(y,z) \qquad give(John,Mary,book)}{own(Mary,book)} \tag{75}$$

With an application of the approach described by (70-74) we can realize this scheme of reasoning by a representation of conceptual vectors, its single items (going top down) are represented as follows

$$\boldsymbol{t}_1^{(1)} = \boldsymbol{op} \otimes \boldsymbol{impl} + \boldsymbol{var}_1 \otimes \boldsymbol{t}_{dat'(x,y,x)} + \boldsymbol{var}_2 \otimes \boldsymbol{t}_{vlastnit'(y,x)} \tag{76a}$$

$$\boldsymbol{t}_2^{(1)} = \boldsymbol{t}_{give(John,Mary,book)} = \boldsymbol{give} + \boldsymbol{give\_var}_1 \otimes \boldsymbol{John} + \boldsymbol{give\_var}_2 \otimes \boldsymbol{Mary} + \boldsymbol{give\_var}_3 \otimes \boldsymbol{book} \tag{76b}$$

$$\boldsymbol{t}_3^{(1)} = t_{own(Mary,book)} = \boldsymbol{own} + \boldsymbol{own\_var}_1 \otimes \boldsymbol{Mary} + \boldsymbol{own\_var}_2 \otimes \boldsymbol{book} \tag{76c}$$

where conceptual vectors $\boldsymbol{t}_{give(x,y,x)}$ and $\boldsymbol{t}_{own(y,x)}$ are constructed analogically as in (45). In the first step we must carry out a concretization of the implication $give(x,y,z) \Rightarrow own(y,z)$, so that the general variables *x*, *y*, *z* are substituted by concrete variables *John*, *Mary*, *book*. This concretization is carried out by a transformation $\boldsymbol{T}$, which is specified by formulas (67-68)

$$\hat{\boldsymbol{t}}_1^{(1)} \approx \boldsymbol{t}_1^{(1)} \otimes \boldsymbol{T} \tag{77}$$

where

$$\hat{\boldsymbol{t}}_1^{(1)} = \boldsymbol{op} \otimes \boldsymbol{impl} + \boldsymbol{var}_1 \otimes \boldsymbol{t}_{give(John,Mary,book)} + \boldsymbol{var}_2 \otimes \boldsymbol{t}_{own(Mary,book)} \tag{78a}$$

$$\boldsymbol{T} = \boldsymbol{t}_1^{*(1)} \otimes \hat{\boldsymbol{t}}_1^{(1)} \tag{78b}$$

Thus concretized representation $give(John,Mary,book) \Rightarrow own(Mary,book)$ is in the next step applicable for modus ponens carried out by formulas (73-74)

$$\tilde{\boldsymbol{t}}_1^{(1)} \approx \hat{\boldsymbol{t}}_1^{(1)} \otimes \boldsymbol{T}' \tag{79a}$$

where the resulting conceptual vector $\tilde{\boldsymbol{t}}_1^{(1)}$ already represents modus ponens, i.e. the next formula holds

$$\boldsymbol{t}_{give(John,Mary,book)} \otimes \tilde{\boldsymbol{t}}_1^{(1)} \approx \boldsymbol{t}_{own(Mary,book)} \tag{80}$$

The other three generalized modus ponens from the figure 18 can be realized in a similar way.

Illustrative example –generalization by induction

Let us have a „training“ set composed of sequences of simple unary predicates, which can be interpreted as observations

$$A_{train} = \{m(a_i)\,;\, i = 1,2,...,q\} \tag{81}$$

Our goal will be to generalize these particular predicates into the form with a universal quantificator (or in our simpler formalism, with a universal variable)

$$(\forall x)[m(x)] \equiv m(x) \tag{82}$$

It means, that the single particular cases $m(a_i)$ from the training set (77) are generalized into a formula, which is not directly deducible from them (see fig. 19)

$$m(a_1) \wedge ... \wedge m(a_p) \overset{?}{\Rightarrow} m(x) \tag{83}$$

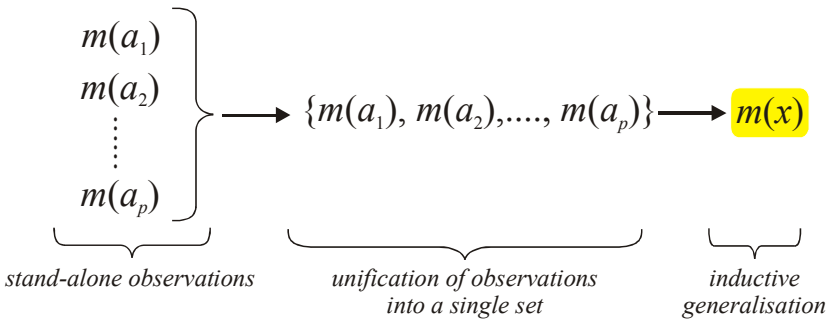

Figure 19. Diagrammatic scheme of inductive generalization, which consists from three stages. In the first stage we have isolated observations $m(a_i)$, which are not related. In the second stage the isolation of single observations is replaced by their unification into a single set, which expresses the fact that observations are not isolated and independent, but they have something in common. In the final, third stage, the unified form of observations is inductively generalized using a universal variable *x*, which represents a class of objects with the same property *m*.

Let each predicate $m(a_i)$ from the training set A_{train} is holographic represented in the following way

$$\boldsymbol{t}_{m(a_i)} = \boldsymbol{rel} \otimes \boldsymbol{m} + \boldsymbol{m\_var} \otimes \boldsymbol{a}_i \tag{84}$$

These represented conceptual vectors can be mutually transformed by transformational vectors $T_{i,j}$

$$\boldsymbol{t}_{m(a_j)} \approx \boldsymbol{t}_{m(a_i)} \otimes \boldsymbol{T}_{i,j} \tag{85a}$$

$$\boldsymbol{T}_{i,j} = \boldsymbol{t}^{*}_{m(a_i)} \otimes \boldsymbol{t}_{m(a_j)} \approx \boldsymbol{a}^{*}_{i} \otimes \boldsymbol{a}_{j} \tag{85b}$$

Let us define a new transformation vector $\tilde{\boldsymbol{T}}_i$

$$\tilde{\boldsymbol{T}}_i = \sum_{j=1}^{q} \boldsymbol{T}_{i,j} = \boldsymbol{a}^{*}_{i} \otimes \boldsymbol{x} \tag{86a}$$

$$\boldsymbol{x} = \boldsymbol{a}_1 + \boldsymbol{a}_2 + ... + \boldsymbol{a}_p \tag{86b}$$

where the conceptual vector $\boldsymbol{x}$ is assigned to the new "generalized" vector, which represents each argument a_i from the training set A_{train} (in our further consideration about induction it is understood as a stand-alone conceptual entity equal to the representations of the original objects $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, ..., $\boldsymbol{a}_p$)

$$\boldsymbol{t}_{m(a_i)} \otimes \tilde{\boldsymbol{T}}_i \approx \boldsymbol{t}_{m(x)} \tag{87}$$

This formula can be interpreted as an inductive generalization, where particular objects a_i from the training set in representations $\boldsymbol{t}_{m(a_i)}$ are substituted by a new object x, that can be interpreted as a new universal object.

9. Formal languages

In Section 4 we have studied an application of holographic distributed representation of sequences of symbols, which are considered as one of the simplest structured data. Let us consider a finite vocabulary $\mathcal{A} = \{a, b, ..., p, q, ...\}$ composed of lower-case letter that are called the terminal symbols. A set of all strings composed of terminal symbol is denoted by $L = \mathcal{A}^{+} = \{a, b, ..., aa, ab, ..., aaa, aab, ..., ppp, ...\}$. This infinite set (but enumerable) set may be divided onto two disjoint nonempty subsets, $L = L_G \cup L_{NG}$, and now we are standing before a problem how to briefly characterize the strings from L_G. For example, we may look for a characteristic function that specifies the subset L_G

$$\eta_G(x) = \begin{cases} 1 & (\text{if } x \in L_G) \\ 0 & (\text{otherwise}) \end{cases} \tag{88}$$

then $L_G = \{x \mid x \in L \wedge \eta_G(x) = 1\}$. This approach of characteristic function is rather formal and specifies, in fact, strings from L_G by their enumeration, i.e. whether a string $x \in L$ belongs or not to the subset L_G.

In general, we may postulate that strings from L_G are hierarchically chunked into two or more components, such that going successively from initially "unchunked" string we arrive at one global chunk. This process may be represented by many different ways (see Fig. 20, where three simple ways of chunking are presented). The approach of rooted tress used in Fig. 20 is one of simplest ways to formalize the above mentioned chunking process of string symbols into hierarchically higher entities, it is simple formalized by making use of the so-called productions (rewriting rules), for an example presented in the first row in Fig. 20 we get

$$\begin{aligned} P_1 &: S \to BC \\ P_2 &: B \to Ac \\ P_3 &: C \to ab \\ P_4 &: A \to ab \end{aligned} \tag{89}$$

where the capital letters are called the nonterminal symbols (loosely speaking, they represent labels of chunks), the capital *S,* called the starting nonterminal (it corresponds to hierarchically top chunk encompassing all terminal symbols from the used string). Applying then productions from (89) we may create successively the string $x=(abcab)$ as follows

$$S \to BC \begin{cases} \to AcC \begin{cases} \to abcC \\ \to Acab \end{cases} \\ \to Bab \ \to \ Acab \end{cases} \to abcab \tag{90}$$

We see that there exist three different ways how to construct the string $x=(abcab)$.

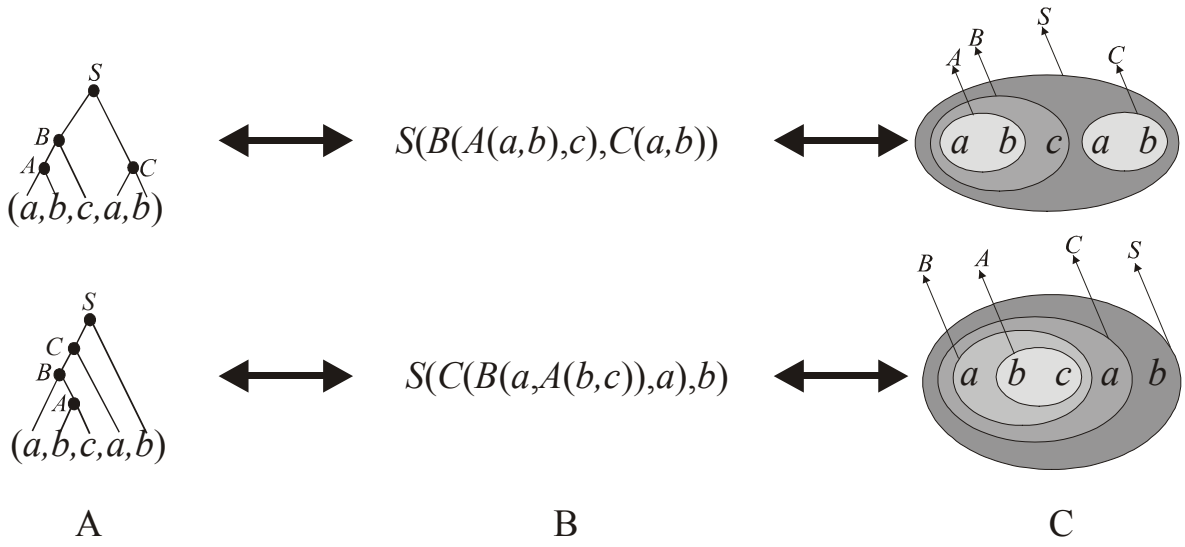

Figure 20. Two illustrative conversions of a sequence (*a,b,c,a,b*) into different „nested" structures: (A) Rooted tree structure that aggregates simple sequence elements into higher-level hierarchic structures, (B) composite two argument functions, and (C) nested subsets encompass one or more symbols such that the resulting structure correspond to an aggregation of symbols.

The production system (89) may be considerably simplified such that we removed all nonterminal symbols that are redundant (in this specific case we may remove the nonterminal *C*, which is identical to *A*.

$$\begin{aligned} P_1 &: \ S \to BA \\ P_2 &: \ B \to Ac \\ P_3 &: \ A \to ab \end{aligned} \tag{91}$$

Moreover, the production system may be extended about one or more productions such that the construction of the original string $x=(abcab)$ is saved, but we may potentially construct many other new different strings $x \in L$, e.g.

$$\begin{aligned} P_1 &: \ S \to BA \\ P_2 &: \ B \to Ac \\ \left.\begin{aligned} P_3 &: \ A \to ab \\ P_4' &: \ A \to Aa \end{aligned}\right\} & P_{34} : \ A \to ab \mid Ab \end{aligned} \tag{92}$$

where new production $P_4' : \ A \to Aa$ introduces a generative feature of the resulting used scheme.

The above considerations may be generalized onto a concept called the *grammar* [xx] (initially introduced by Noam Chomsky [20] almost 50 years ago)specified as follows

$$G = \left(N, T, S, P \right) \tag{93}$$

where N is a set composed of nonterminal symbols (in our above preliminary considerations represented by capital letters), $T \subset \mathcal{A}$ is a set composed of terminal symbols (represented by lower case letters), $S \in N$ is a *starting symbol* especially distinguished from elements of nonterminals *N*, and finally, *P* is a set composed of the so-called *productions* (often called also production rules or rewriting rules)

$$P = \left\{ \alpha \to \beta \right\} \tag{94}$$

where α is a substring composed of terminal and at least one nonterminal symbols and β is a substring composed of terminal or nonterminal symbols. In particular, the set *P* contains a production of the form $S \to \beta$, this production will initialized the procedure of creation of strings that are induced by the grammar *G*. These strings, initialized by the starting symbol *S* and composed entirely of terminal symbols form a subset denoted by L_G, which is called the *language*. It means that the characteristic function (88) is now unambiguously specified by the production set (94) such its elements are (1) composed only of terminal symbols and (2) its creation procedure is initialized the nonterminal starting symbol *S*.

We have seen that each grammatical string $x \in L_G$ may be interpreted by a rooted tree $t\left(x \right)$ called the *derivation tree*, where a top vertex is labeled by the stating symbol *S*, its leafs (terminal vertices) are labeled by terminal symbols, and all other remaining intermediate vertices are labeled by nonterminal symbols (cf. Fig. 20). Derivation trees have an extraordinary position in artificial intelligence and cognitive science [21,22], a semantic meaning of the grammatical string $x \in L_G$ is specified not only by the string itself but also by its derivation tree $t\left(x \right)$. If a string has two or more derivation trees, then we may say that it has more than one different semantic meanings. An actual semantic meaning of $x \in L_G$ will depend on the used derivation tree $t\left(x \right)$, which specifies an actual chunking of string symbols onto hierarchically higher clusters that contribute to the whole semantic meaning by a specific contribution.

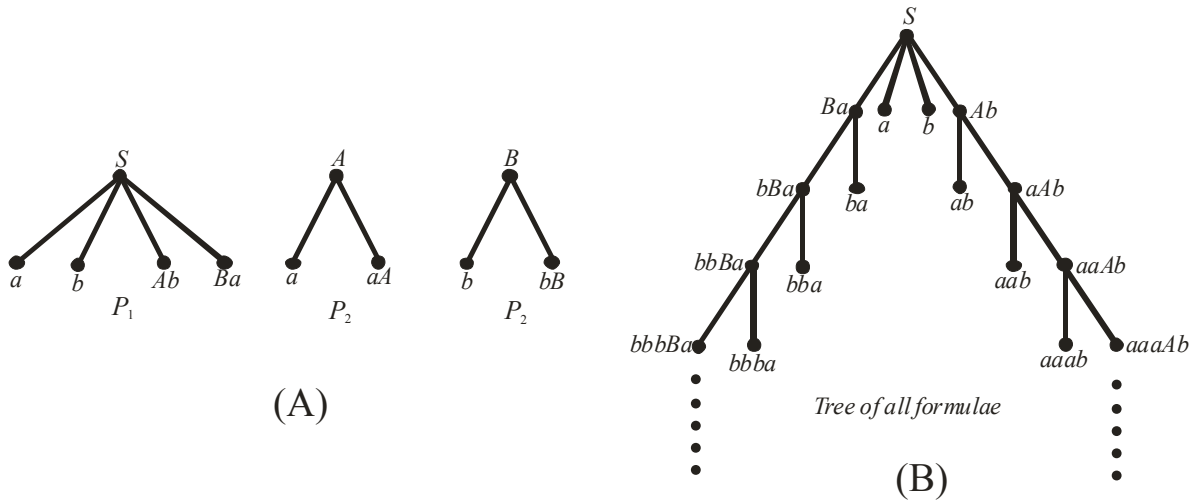

Figure 21. (A) Diagrammatic representation of productions (1).. (B) An infinite tree of all possible solutions – formulae.

Let is consider the following simple grammar $G=\left(N=\{S,A,B\},T=\{a,b\},S,P\right)$ with productions specified by

$$\begin{aligned} P_1 &: S \to a \mid b \mid Ab \mid Ba \\ P_2 &: A \to a \mid aA \\ P_3 &: B \to b \mid bB \end{aligned} \tag{95}$$

with productions diagrammatically visualized by These productions are diagrammatically visualized in Fig. 21, diagram A.

From fig. 21 we may construct all possible formulae that are specified by the productions (95)

$$\begin{aligned} L_G &= \left\{b^n a; n \geq 0\right\} \cup \left\{a^n b; n \geq 0\right\} = \\ &= \{a,b,ab,ba,bba,aab,bbba,aaab,...\} \end{aligned} \tag{96}$$

For each grammatical formula there exists on tree of all formulae an oriented path beginning in the root of tree (labeled by a starting nonterminal *S*) and ending at a leaf evaluated by the respective formula, e.g.

$$\begin{aligned} a \in L_G &\Rightarrow S \to a \\ b \in L_G &\Rightarrow S \to b \\ ab \in L_G &\Rightarrow S \to Ab \to ab \\ ba \in L_G &\Rightarrow S \to Ba \to ba \\ aab \in L_G &\Rightarrow S \to Ab \to aAb \to aab \\ bba \in L_G &\Rightarrow S \to Ba \to bBa \to bba \end{aligned} \tag{97}$$

The corresponding derivation trees are listed in Fig. 22.

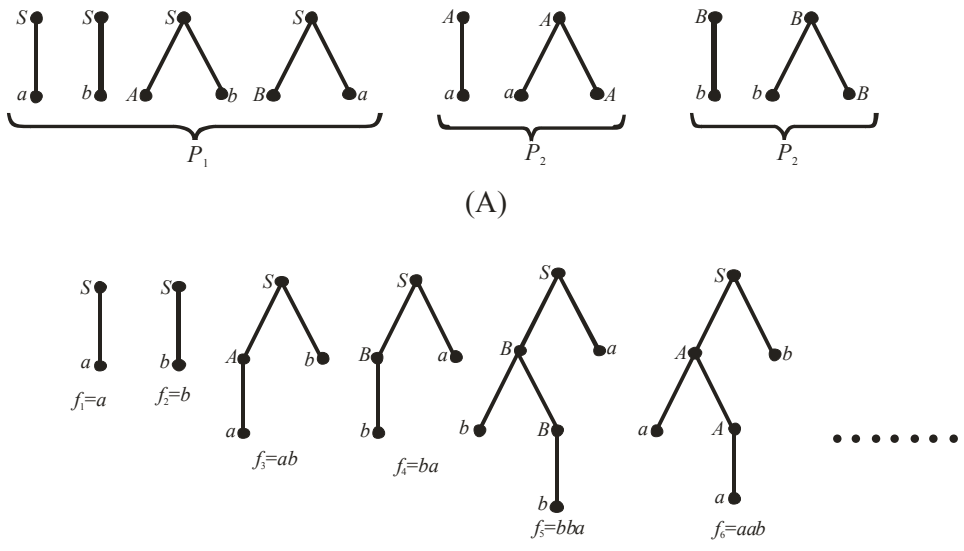

Figure 22. (A) Detailed diagrammatic representation of productions (1). (B) Derivation trees assigned to the first six formulae.

HRR representations of derivation trees in Fig. 22 are (cf. (44))

$$
\begin{aligned}
\boldsymbol{t}_1 &= \boldsymbol{left} \otimes \boldsymbol{a} \\
\boldsymbol{t}_2 &= \boldsymbol{left} \otimes \boldsymbol{b} \\
\boldsymbol{t}_3 &= \boldsymbol{left} \otimes \left(\boldsymbol{A} + \boldsymbol{left} \otimes \boldsymbol{a} \right) + \boldsymbol{right} \otimes \boldsymbol{b} \\
\boldsymbol{t}_4 &= \boldsymbol{left} \otimes \left(\boldsymbol{B} + \boldsymbol{left} \otimes \boldsymbol{b} \right) + \boldsymbol{right} \otimes \boldsymbol{a} \\
\boldsymbol{t}_5 &= \boldsymbol{left} \otimes \left(\boldsymbol{B} + \boldsymbol{left} \otimes \boldsymbol{b} + \boldsymbol{right} \otimes \left(\boldsymbol{B} + \boldsymbol{left} \otimes \boldsymbol{b} \right) \right) + \boldsymbol{right} \otimes \boldsymbol{a} \\
\boldsymbol{t}_6 &= \boldsymbol{left} \otimes \left(\boldsymbol{A} + \boldsymbol{left} \otimes \boldsymbol{a} + \boldsymbol{right} \otimes \left(\boldsymbol{A} + \boldsymbol{left} \otimes \boldsymbol{a} \right) \right) + \boldsymbol{right} \otimes \boldsymbol{b}
\end{aligned}
\tag{98}
$$

where we have omitted the vector representation $\boldsymbol{S}$ of the nonterminal starting symbol, which appears as the first in each representation of derivation trees.

HRR representation of the first grammatical strings (96) are (cf. (34))

$$
\begin{aligned}
\boldsymbol{f}_1 &= \boldsymbol{first} \otimes \boldsymbol{a} \\
\boldsymbol{f}_2 &= \boldsymbol{first} \otimes \boldsymbol{b} \\
\boldsymbol{f}_3 &= \boldsymbol{first} \otimes \boldsymbol{a} + \boldsymbol{second} \otimes \boldsymbol{b} \\
\boldsymbol{f}_4 &= \boldsymbol{first} \otimes \boldsymbol{b} + \boldsymbol{second} \otimes \boldsymbol{a} \\
\boldsymbol{f}_5 &= \boldsymbol{first} \otimes \boldsymbol{b} + \boldsymbol{second} \otimes \boldsymbol{b} + \boldsymbol{third} \otimes \boldsymbol{a} \\
\boldsymbol{f}_6 &= \boldsymbol{first} \otimes \boldsymbol{a} + \boldsymbol{second} \otimes \boldsymbol{a} + \boldsymbol{third} \otimes \boldsymbol{b}
\end{aligned}
\tag{99}
$$

The used holographic representations of derivation trees and string of symbols have a serious shortcoming; different $\boldsymbol{t}$'s have great similarities between them (the same property is also true for vectors $\boldsymbol{t}$'s), see Fig. 23. This fact causes that an immediate application of (28) to form an analogy of associative memory vector $\boldsymbol{T} = \boldsymbol{t}_1 \otimes \boldsymbol{f}_1 + ... + \boldsymbol{t}_6 \otimes \boldsymbol{f}_6$ could not be well performed, since this particular memory vector does not satisfy the following two important properties $\boldsymbol{t}_i^* \otimes \boldsymbol{T} \approx \boldsymbol{t}_i$ and $\boldsymbol{f}_i^* \otimes \boldsymbol{T} \approx \boldsymbol{f}_i$ that are of basic importance for correct recognition of $\boldsymbol{T}$. Therefore we have to use slightly another approach to the construction of associative memory vector $\boldsymbol{T}$ based on the so-called *label vectors*.

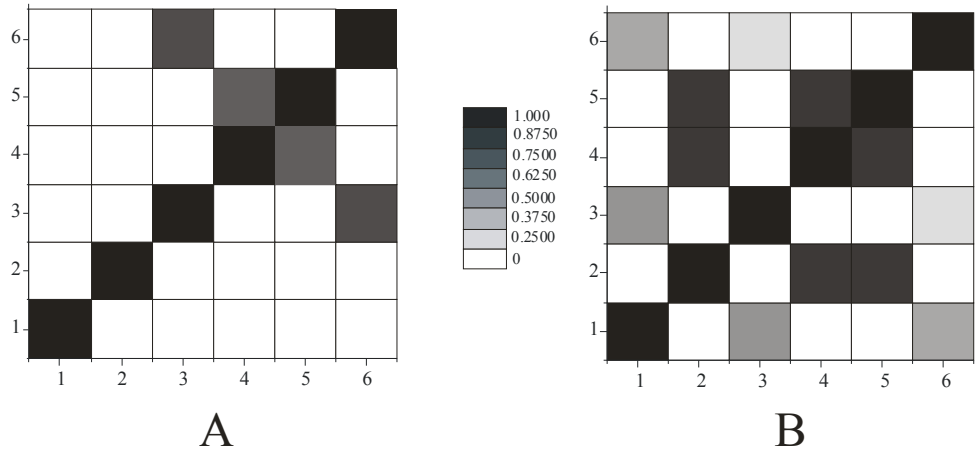

Figure 23. Graphic representation of similarities (A) between holographic distributed representations of derivation trees (98) and (B) between their terminal symbol saequences.

Let us have a set of six randomly generated label vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_6$, which will serve in our forthcoming considerations as labels for distinguishing of vector representations of derivation trees (that may be very similar from the standpoint of their mutual overlap specified by (21)). Applying these "label" vectors we construct the following new modified vectors

$$\tilde{\boldsymbol{t}}_i = \boldsymbol{x}_i \otimes \boldsymbol{t}_i \quad \text{and} \quad \tilde{\boldsymbol{f}}_i = \boldsymbol{x}_i \otimes \boldsymbol{f}_i \tag{100}$$

A transformation vector is determined as follow

$$\boldsymbol{T} = \sum_{i=1}^{6} \tilde{\boldsymbol{t}}_i^* \otimes \tilde{\boldsymbol{f}}_i \tag{101}$$

The modified vectors are specified by

$$\tilde{\boldsymbol{t}}_i \otimes \boldsymbol{T} \approx \tilde{\boldsymbol{f}}_i \quad \text{and} \quad \tilde{\boldsymbol{f}}_i \otimes \boldsymbol{T}^* \approx \tilde{\boldsymbol{t}}_i \tag{102}$$

for i=1,2,…,6. It means that the transformation vector $\boldsymbol{T}^*$ plays a role of parser, which assigns to a „formula" $\boldsymbol{f}$ the respective „derivation tree" $\boldsymbol{t}$. In other words, we have „algebraized" the highly symbolic parsing process of formulae, which assigns derivation trees to formulae, where the trees are very important for a construction of semantic interpretations of the formulae (i.e. the semantic meaning of formulae is specified not only by its terminal symbols but also by its respective derivation tree).

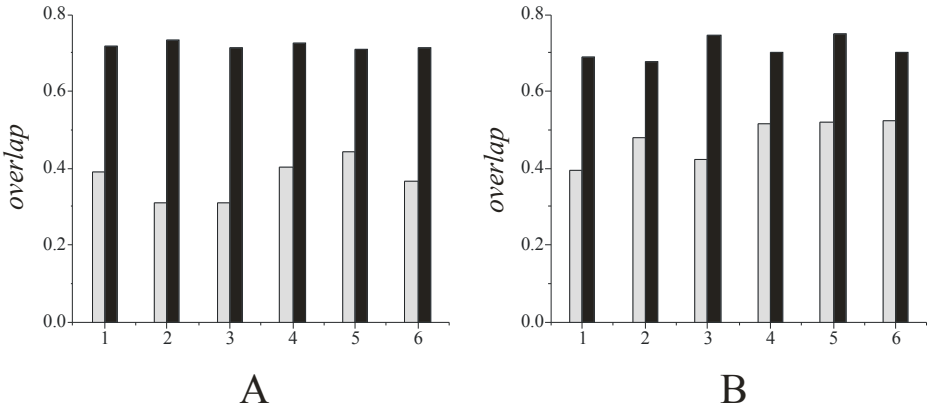

Figure 24. Reconstruction of holographic representations of derivation tress (A) and symbol sequences (B) induced by the grammar (95). The light darkness columns correspond to overlap between reconstructed and original vectors

10. Conclusions

Holographic reduced representation offers new unconventional solution to one of the basic problems of artificial intelligence and cognitive science, which is to find a suitable distributive coding of structured information (sequence of symbols, nested relational structures, etc.). The used distributed representation is based on two binary operations: unary operation „involution" and binary operation „convolution" over a domain of n-dimensional randomly generated conceptual vectors, which elements satisfy normal distribution $N(0,1/n)$. Application of this distributed representation allows us to model various types of associative memory, which are represented by a conceptual vector and also to decode a memory vector, i.e. to determine the conceptual (atomic) vectors it is composed of. Such an analysis of the memory vector is carried out by a clean-up procedure that determines from the overlap of the vectors, which of the vectors is the most similar to the memory vector. We have also described the process of chunking of vectors, which allows us to overcome the undesirable fast degradation of success in retrieval of all the components of the memory vector. The countermeasure against the degradation consists in chunking of several vectors into one, which is then put into the memory vector. Holographic reduced representation allows to measure similarity between two structured concepts by a simple algebraic operation of scalar product of their distributed representations. This fact can be very useful, when we want to model processes, which search through memory to find its similar (analogical) single

components. In the last part of the paper we have demonstrated, that the holographic reduced representation may be used also to model an inference process based on the rules *modus ponens* and *modus tollens*, and moreover, we have demonstrated the effectiveness of the approach for modeling of an inductive generalization process.

Acknowledgments

This work was supported by the grants # 1/0062/03 and # 1/1047/04 of the Scientific Grant Agency of Slovak Republic.

References

[1] Davis, P. J.: *Circulant Matrices*. Chelsea Publishing, New York, 1999.

[2] Gabor, D.: Holographic model for temporal recall. *Nature*, **217** (1968), 1288–1289.

[3] McClelland, J. L. M., Rumelhart, D. E., and the PDP research group (eds.): *Parallel distributed processing: Explorations in the microstructure of cognition*, volumes 1 and 2. The MIT Press, Cambridge, MA (1986).

[4] Metcalfe Eich, J.: Levels of processing, encoding specificity, elaboration, and charm. *Psychological Review*, **92** (1985), 1–38.

[5] Murdock, B. B.: A theory for the storage and retrieval of item and associative information. *Psychological Review*, **89** (1982), 316–338.

[6] Plate, T. A.: *Distributed Representations and Nested Compositional Structure*. Department of Computer Science, University of Toronto. Ph.D. Thesis, 1994.

[7] Plate, T. A.: *Holographic Reduced Representation: Distributed Representation for Cognitive Structures*, CSLI Publications, Stanford, CA, 2003.

[8] Plate, T. A.: Holographic Distributed Representations. *IEEE Transaction on Neural Networks*, **6** (1995), 623-641.

[9] Slack, J. N.: The role of distributed memory in natural language processing. In O'Shea, T. (ed.): *Advances in Artificial Intelligence: Proceedings of the Sixth European Conference on Artificial Intelligence, ECAI-84*. Elsevier Science Publishers, New York, 1984.

[10] Smolensky, P., Legendre, G., Miyata, Y.: *Principles for an Integrated Connectionist/Symbolic Theory of Higher Cognition***.** Technical Report CU-CS-600-92, Department of Computer Science and 92-8, Institute of Cognitive Science. University of Colorado at Boulder, 1992, 75 pages.

[11] Smolensky, P.: On the proper treatment of connectionism. *The Behavioral and Brain Sciences,* **11** (1988), 1-74.

[12] Smolensky, P.: Tensor product variable binding and the representation of symbolic structures in connectionist networks. *Artificial Intelligence,* **46** (1990)., 159-216.

[13] Smolensky, P: Neural and conceptual interpretations of parallel distributed processing models**.** In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models*. MIT Press - Bradford Books, Cambridge, MA, 1986, pp. 390-431.

[14] Willshaw, D. J., Buneman, O. P., and Longuet-Higgins, H. C.: Non-holographic associative memory. *Nature*, **222** (1969), 960–962.

[15] Shastri, L., Ajjanagadde, V.: From simple associations to systematic reasoning: connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences*, **16** (1993), 417-494.

[16] Kanerva, P.: The Spatter Code for Encoding Concepts at Many Levels. In Marinaro, M., Morasso, P. G. (eds.): *ICANN '94, Proceedings of the International Conference on Artificial Neural Networks*. Springer Verlag, London, 1994, pp. 226–229.

[17] Kanerva, P.: Binary Spatter-Coding of Ordered K-tuples. In von der Malsburg, C., von Seelen, W., Vorbruggen, J. C., Sendhoff, B. (eds.): *ICANN'96, Proceedings of the International Conference on Artificial Neural Networks*. Springer Verlag, Berlin, 1996, pp. 869-873.

[18] Kanerva, P.: Analogy as a Basis of Computation, In Uesaka, Y., Kanerva, P., Asoh, H.: *Foundations of real-world intelligence*. CSLI Publications, Stanford, CA, 2001.

[19] Fodor, J.A., Pylyshyn, Z.W.: Connectionism and cognitive architecture: A critical analysis., *Cognition*, **28** (1988) 3-71.

[20] Chomsky, N.: Three models for the description of language. *IEEE Trans. Info. Theory* **2** (1956), 113 - 124.

[21] Johnson – Laird, P.N.: *Mental Models. Toward a Cognitive Science of Language, Inference, and Consciousness*. Cambridge University Press, Cambridge 1983.

[22] Russel,. S., Norvig, P.: *Artificial Intelligence. A Modern Approach*. Prentice Hall, New Jersey 1995.