Adv Syst Sci Appl 2020; **04**:70–82
Published online at https://ijassa.ipu.ru.

Multi-Valued Neural Networks II: A Robot Group Control

Dmitry Maximov[1]*

[1] *V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, Russia*

Abstract: For the new concept of a multi-valued neural network introduced earlier, an analogue of the T-norm in fuzzy mathematics is considered. In the multi-valued neural network, all variables are elements of the lattice of linguistic variables, i.e., they are all only partially-ordered. The lattice operations are used to build the network output by inputs. However, a lattice elements' multiplication may also be used to determine such operations in the case when not all of them are allowed by the lattice construction. In this paper, the lattice is assumed to be residuated, and the residual construction gives the analogue of a T-norm. The lattice elements' multiplication determines the implication which is used, together with other lattice operations, in output determining of the neural network. Though, such a construction determines a multi-valued associative memory similar to the Brouwer lattice case considered earlier, this variant is more natural to use in the Kohonen-like networks that we demonstrate with the example of a robot group management.

Keywords: multi-valued neural networks, fuzzy neural networks, robot group management, associative memory, linguistic variable lattice, system tasks' lattice

1. INTRODUCTION

The concept of multi-valued neural networks was introduced in [1], [2], and "Multi-Valued Neural Networks I" by D. Maximov, V. I. Goncharenko, and Yu. S. Legovich. It continues a line of research to assess the system state using elements of a lattice. The lattice may be a lattice of sets as in [3], [4] or, even, a lattice of graphs of system state configurations [5], [6].

Such an approach is used in different tasks when the situation is estimated by a set of linguistic variables along with degrees of significance/certainty of the values of these variables. Usually, in fuzzy systems, confidence levels take values in a numerical interval. However, the degree of fuzziness is determined by experts. In [7] it was demonstrated that it is not necessary to use numbers: linguistic variables (but already partially-ordered), which do not require a mandatory numerical evaluation, can again serve as an estimate. Such assessments are determined by the situation itself and do not need an expert opinion or, at least, the experts assessment of the situation is greatly facilitated.

In this case, to compare the valuations and obtain control solutions in [7], the concepts of multi-valued logic are used in which the scale of truth values is a Brouwer lattice of a general form. Such scales of truth values generalize a linearly ordered scale (in particular, in fuzzy logic) and naturally define the implication as a lattice operation.

In the case of a complicated lattice, the calculation of implications becomes a laborious task. In [1] and [2], it is proposed to use a multi-valued neural network to calculate implications and a multi-valued associative memory to obtain a control solution quickly. The

*Corresponding author: jhanjaa@ipu.ru, dmmax@inbox.ru associative memory generalizes a fuzzy one of [8] to the case of using multi-valued logic operations instead of fuzzy ones.

In "Multi-Valued Neural Networks I" the results of [8] on fuzzy associative memory with thresholds to the case of multi-valued one were expanded: the authors introduced the concept of such a network, investigated the properties of the associative memory and gave a generalization of the learning algorithm of fuzzy associative memory with thresholds to the multi-valued case. The inputs, outputs and connection weights of the network are linguistic variables (partially ordered), not numbers, which, however, made it possible to use such a network for processing control and diagnostic information of complex dynamic objects.

However, a Brouwer lattice was used as the scale of truth value in both these cases, which restricts application possibilities. In [8], a general fuzzy associative memory with a T-norm is also considered. In residuated lattices, properties in the definition of the T-norm coincide with the same as in the residuated construction definition. In this paper we use this fact to expand the results of [8] to associative memories with variables taking values in residuated lattices that may not be Brouwerian in this case. We investigate the properties of such a multi-valued associative memory and give a generalization of the learning algorithm of fuzzy associative memory to the multi-valued case similarly to "Multi-Valued Neural Networks I" and [1]. However, the application of such a construction is natural in Kohonen-like neural networks rather than in associative memories due to the residuals' intuitive meaning. We demonstrate such an application — with the example of a robot group management.

2. BACKGROUNDS

Feedforward fuzzy neural networks in which internal operations are based on fuzzy operations of joins and meets $\vee - \wedge$, were proposed in [9]. Such networks are called $\vee - \wedge$ fuzzy associative memory, and fuzzy information in them represented by the elements of the interval [0, 1]. We will use elements of a general lattice in multi-valued associative memory instead of elements of the interval [0, 1].

2.1. Lattices

[10]

Definition 2.1:
*A **lattice** is a partially-ordered set having, for any two elements, their exact upper bound or join $\vee$ (sup, max) and the exact lower bound or meet $\wedge$ (inf, min).*

Definition 2.2:
*The **exact upper bound** of a subset $X \subseteq P$ of a partially-ordered set P is the smallest P-element a, larger than all the elements of X: $min(a) \in P : \ a \geq x, \ \forall x \in P$.*

Definition 2.3:
*The **exact lower bound** is dually defined as the largest P-element, smaller than all the elements of X.*

Definition 2.4:
*A **complete lattice** is a lattice in which any two subsets have a join and a meet. This means that in a non-empty complete lattice there is the largest "$\top$" and the smallest "0" elements.*

If we take such a lattice as a scale of truth values in a multi-valued logic, then the largest element will correspond to complete truth (true), the smallest to complete falsehood (false), and intermediate elements will correspond to partial truth in the same way as the elements of the segment [0,1] evaluate partial truth in fuzzy logic.

In logics with such a scale of truth values, implication can be determined by multiplying lattice elements, or internally, only from lattice operations.

Definition 2.5:
*Lattice elements, from which all the others are obtained by join and meet operations are called **generators** of the lattice.*

Definition 2.6:
*A lattice is called **atomic** if every two of its generators have null meets.*

Definition 2.7:
*A **Brower lattice** is the lattice that has internal implications.*

Definition 2.8:
*In such a lattice, the **implication** $c = a \Rightarrow b$ is defined as the largest $c:\ a \wedge b = a \wedge c$.*

Definition 2.9:
*The implication $\neg a = a \Rightarrow 0$ is called the **pseudo-complement** of a.*

Distribution laws for join and meet are satisfied in Brouwer lattices. The converse is true only for finite lattices.

2.2. Residuated Lattices

[11] In non-distributive lattices, the implication cannot be defined. However, we may introduce a multiplication of the lattice elements and use it to define an external implication.

Definition 2.10:
*A **residuated lattice** is an algebra $(L, \vee, \wedge, \cdot, 1, \rightarrow, \leftarrow)$ satisfying the following conditions:*

- $(L, \vee, \wedge)$ *is a lattice;*
- $(L, \cdot, 1)$ *is a monoid;*
- $(\rightarrow, \leftarrow)$ *is a pare of residuals of the operation $\cdot$, that means*

$$\forall x, y \in L : x \cdot y \leqslant z \Leftrightarrow y \leqslant x \rightarrow z \Leftrightarrow x \leqslant z \leftarrow y$$

In this case, the operation $\cdot$ is order preserving in each argument and for all $a, b \in L$ both the sets $\{y \in L | a \cdot y \leqslant b\}$ and $\{x \in L | x \cdot a \leqslant b\}$ each contains a greatest element ($a \rightarrow b$ and $b \leftarrow a$ respectively).

The monoid multiplication $\cdot$ is distributive over $\vee$:

$$x \cdot (y \vee z) = (x \cdot y) \vee (x \cdot z).$$

Also, $x \cdot 0 = 0 \cdot x = 0$. A special case of residuated lattices is a Heyting algebra, when the monoid multiplication coincides with $\wedge$.

In non-commutative monoids, residuals $\rightarrow$ and $\leftarrow$ can be understood as having a temporal quality: $x \cdot y \leqslant z$ means "x *then* y *entails* z," $y \leqslant x \rightarrow z$ means "y *estimates the transition had* x *then* z," and $x \leqslant z \leftarrow y$ means "x *estimates the opportunity* ***if-ever*** y *then* z." You may think about x, y, and z as *bet*, *win*, and *rich* correspondingly (Wikipedia). Though, the temporal quality refers to residuals, we refer the same meaning to any multiplication $x \cdot w \leqslant z$. However, only the residual y estimates the transition $x \rightarrow z$.

Definition 2.11:
*A residuated lattice A is **cancellative** if it satisfies the equations $x \cdot y = x \cdot z \Longrightarrow y = z$ and $y \cdot x = z \cdot x \Longrightarrow y = z$*
[12].

A residuated lattice is cancellative if, and only if, it satisfies the equations $x \rightarrow xy = y$ and $yx \leftarrow x = y$
[12].

Definition 2.12:
*A residuated lattice A is said to be **integrally closed** if it satisfies the equations $x \cdot y \leqslant x \Longrightarrow y \leqslant 1$ and $y \cdot x \leqslant x \Longrightarrow y \leqslant 1$, or equivalently, the equations $x \rightarrow x = 1$ and $x \leftarrow x = 1$ [13].*

Every cancellative residuated lattice is an integrally closed one, and any upper or lower bounded integrally closed residuated lattice L is **integral**, i.e., $a \leqslant 1, \ \forall a \in L$ [13]. Therefore, a finite calcellative lattice is integral.

We will assume that the lattices used in a multi-valued neural network are complete, finite and residuated.

We can compare the definition of a residuated lattice and the definition of a fuzzy operator and a T-norm
[14], [8]:

Definition 2.13:
*We call the mapping $T: \ [0,1] \rightarrow [0,1]$ a **fuzzy operator**, if the following conditions hold:*

- $T(0,0) = 0, T(1,1) = 1$;
- *If* $a, b, c, d \in [0,1]$, *then* $a \leqslant c, \ b \leqslant d, \ \Rightarrow \ T(a,b) \leqslant T(c,d)$;
- $\forall a, b \in [0,1], \ T(a,b) = T(b,a)$;
- $\forall a, b, c \in [0,1], \ T(T(a,b),c) = T(a,T(b,c))$.

*If T is a fuzzy operator, and $\forall a \in [0,1], \ T(a,1) = a$, we call T a **T-norm**. For $a, b \in [0,1]$, let us define $a \rightarrow b \in [0,1] : \ a \rightarrow b = sup\{x \in [0,1] | T(a,x) \leqslant b\}$.*

We see that the latter definition coincides with the residual construction definition in the case of a commutative monoid and $L = [0,1]$. Excluding the condition of commutativity and the requirement $L = [0,1]$, we obtain the construction of a complete and residuated lattice (we demand also its finiteness).

3. MULTI-VALUED $\vee - *$ ASSOCIATIVE MEMORY

As in the case of fuzzy neural networks [8], a multiplication of variables is used together with lattice operations in $\vee - *$ multi-valued associative memory, where $*$ stands for the multiplication. Let us suppose, an input signal is $x \in L^n$, and an output signal is $y \in L^m$, where L is the residuated lattice used. In this case, the input-output relationship in a two-layer multi-valued associative memory can be written as $\mathbf{y} = \mathbf{x} \circledast \mathbf{W}$, where $\circledast$ stands for the $\vee - *$ composition operation, and $\mathbf{W} = (w_{ij})_{n \times m} \in \mu_{n \times m}$ is the $n \times m$ matrix of the weights of connections with elements from L, $i \in N = \{1...n\}, \ j \in M = \{1...m\}$ (Fig. 3.1). In [1], several of the simplest composition variants have been suggested for $\vee - \wedge$ multi-valued associative memory and the next two of them are considered:

$$y_j = \bigvee_i \{x_i \wedge w_{ij}\}; \tag{3.1}$$

$$y_j = \bigwedge_i \{x_i \Rightarrow w_{ij}\}. \tag{3.2}$$

In the theory of the simplest fuzzy associative memory, only option (3.1) is considered. More complex models use a combination of $\vee$ and t-norms
[8], [15] and a combination with implication [15]. However, they all use a linear number segment as a set of variables, unlike this paper where a general non-number lattice with multiplying elements is used.

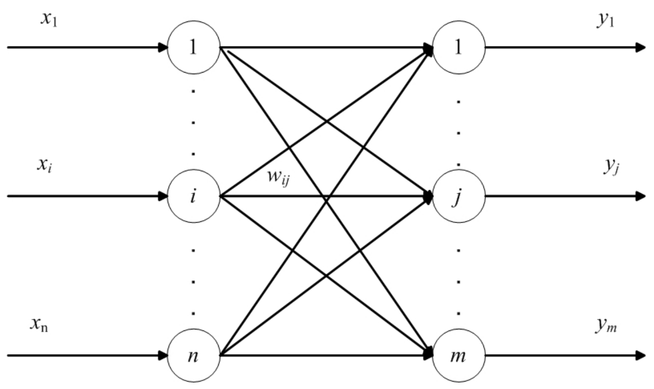

Fig. 3.1. Bilayer associative memory

Thus, we consider the composition 3.1, but with the monoid multiplication $*$ instead of $\wedge$ (Fig. 3.1), as in [8], however with all variables taking values in the residuated lattice L:

$$y_j = \bigvee_i \{x_i * w_{ij}\}. \tag{3.3}$$

In the vector notation, it can be written as:

$$\mathbf{y} = \mathbf{x} \circledast \mathbf{W}. \tag{3.4}$$

We denote $(X,Y) = \{\mathbf{x}^k, \mathbf{y}^k \mid k \in P\},\ P = \{1...p\}$ — the family of pairs multi-valued patterns with $\mathbf{x}^k = (x_1^k, ..., x_n^k)$ and $\mathbf{y}^k = (y_1^k, ..., y_m^k)$.

We establish the connection weight matrix $W^* = (w_{ij}^*)_{n\times m}$:

$$w_{ij}^* = \bigwedge_{k\in P} (x_i^k \to y_j^k). \tag{3.5}$$

The definition generalizes the similar one from [8].
Then, let us define the sets [8]:

$$S_{ij}(\mathbf{W}^*, X, Y) = \{k \in P \mid y_j^k \leqslant x_i^k * w_{ij}^*\}.$$

$$M^w = \{\mathbf{W} \in \mu_{n\times m} \mid \forall k \in P,\ \mathbf{x}^k \circledast \mathbf{W} = \mathbf{y}^k\}.$$

This is the set of the pattern pair family and the connection matrix, which satisfy the equation 3.3.

The next property follows from the residual operation definition and holds both in fuzzy [8] and residuated [11] cases:

$$a * (a \to b) \leqslant b. \tag{3.6}$$

Also, the following properties hold in residuated lattices [11]:

$$c \to (\bigwedge Y) = \bigwedge\{c \to y | y \in Y\}; \tag{3.7}$$

$$b \to (a \to c) = (a * b) \to c. \tag{3.8}$$

The property (3.6) is used in the following theorem proof exactly in the same way as in [8] due to the fact that the T-norm is a special case of the residuated lattice construction. However, such a theorem in [8] contains a necessity condition in point (iii) which does not hold in the case of a general non-chain lattice. In the general case, it may be $\vee_i x_i = y$, though $\forall x_i < y$ that contradicts the necessity.

Theorem 1:
Let $\mathbf{W}^* = (w^*_{ij})_{n\times m} \in \mu_{n\times m} \in L^m$. *Then:*

(i) $\forall k \in P, \mathbf{x}^k \circledast \mathbf{W}^* \subset \mathbf{y}^k$, *and if the matrix* **W** *satisfies:* $\forall k \in P, \mathbf{x}^k \circledast \mathbf{W} \subset \mathbf{y}^k$, *then,* $\mathbf{W} \subset \mathbf{W}^*$*;*
(ii) If $M^w \neq \emptyset$, *then,* $\mathbf{W}^* \in M^w$, *and* $\forall \mathbf{W} \in M^w$, $\mathbf{W} \subset \mathbf{W}^*$, *i.e.,* $\forall i \in N,\ j \in M,\ w_{ij} \leqslant w^*_{ij}$*;*
(iii) The set $M^{wcd} \neq \emptyset$ *if* $\forall j \in M,\ \bigcup_{i\in N} S_{ij}(\mathbf{W}^*, X, Y) = P$.

Theorem 1 tells us that $\mathbf{W}^*$ is the largest solution of the equation (3.4), if such a solution exists, and we will use this fact in the next section when we go down from the largest lattice element to **W** in a learning algorithm.

3.1. Learning Algorithm

The iteration scheme for learning the connection weight matrix **W** of the multi-valued associative memory is similar to [8] with changes as in [1] and “Multi-Valued Neural Networks I”. It generalizes the dynamic δ-learning algorithm of fuzzy associative memory, introduced in [16], [17], [18] for different fuzzy cases, to the $\vee - *$ multi-valued case. However, we are bound by atomic cancellative (hence, integrally closed) residuated lattices in the following algorithm, since Theorem 2 is proved only in this case. Hence, $1 = \top$ in the lattice used since it is finite.

Thus, we get an **Algorithm** of w_{ij} iteration for $i \in N$ and $j \in M$:

Step 1. Initialization: for $i \in N, j \in M$ let us put $w_{ij}(0) = \top,\ t = 0$;

Step 2. Let $\mathbf{W}(t) = (w_{ij}(t))$;

Step 3. Let us calculate the resulting output: $\mathbf{y} = \mathbf{x} \circledast \mathbf{W}$, i.e., $\forall k \in P,\ j \in M,\ y^k_j(t) = \bigvee_i [x^k_i * w_{ij}(t)]$.

Step 4. Weights selection.

Denotation 1:
From this place, we denote by $\{y^k_j\}$ *the set of generators contained in the lattice element* y^k_j*:* $\{g_l | g_l \leqslant y^k_j\}$ *and by* $\{w\}_{ij}$ *the matrix of such sets. The matrix elements are the sets of generators of the weight matrix elements.*

Matrices w_{ij} of weights and $\{w\}_{ij}$ are one-to-one correspondent to each other in distributive lattices. In non-distributive lattices, not one set $\{w\}_{ij}$ may correspond to one w_{ij}. In this case, we take $max\{w\}_{ij}$ as corresponding set to w_{ij}.The **set difference** will be denoted by a **minus** sign. Then,

$$\{w\}_{ij}(t+1) = \bigcap_k \begin{cases} \{w\}_{ij}(t) - sym(\{y^k_j(t)\} - \{y^k_j\}), \\ \qquad if\ k:\ x^k_i * w_{ij}(t) > y^k_j; \\ \{w\}_{ij}(t),\ otherwise. \end{cases} \tag{3.9}$$

Here operation sym means the symmetrized set difference:

$$sym(A - B) = \begin{cases} A - B,\ A \cap B \subset A; \\ B - A,\ A \cap B = A. \end{cases} \tag{3.10}$$

Step 5. For $i \in N,\ j \in M$, let us check $\{w\}_{ij}(t+1) = \{w\}_{ij}(t)$? If this is true, then the Algorithm stops, otherwise $t = t + 1$ and goes to Step 2.

Theorem 2:
Let the matrix sequence $\{\mathbf{W}(t) \mid t = 1, 2...\}$ is obtained by the learning Algorithm. Then,

(a) $\{\mathbf{W}(t) \mid t = 1, 2...\}$ *is non-increasing sequence;*
(b) $\{\mathbf{W}(t) \mid t = 1, 2...\}$ *converges;*
(c) $\{\mathbf{W}(t) \mid t = 1, 2...\}$ *converges to* $\mathbf{W} \subseteq \mathbf{W}^*$*, where* w^*_{ij} *is defined in (3.5).*

Proof
(a) For $i \in N$, $j \in M$, $k \in P$ we get from (3.9) that $\{w\}_{ij}(t+1) \leqslant \{w\}_{ij}(t)$. Therefore, for $i \in N$, $j \in M$: $\mathbf{W}(t+1) \subset \mathbf{W}(t)$. Thus, $\{\mathbf{W}(t) \mid t = 1, 2...\}$ is a non-increasing sequence.

(b) $\{\mathbf{W}(t) \mid t = 1, 2...\}$ converges, since the sequence $w_{ij}(t)$ is bounded below by the smallest lattice element 0 $\forall t = 1, 2....$

(c) Let us prove $\forall t \in \{0, 1, ...\},\ w_{ij}(t) \leqslant w^*_{ij}$. If $t = 0$, then $w_{ij}(0) = \top \equiv 1$. If $w_{ij}(t) = \top$ for $t = 1, 2, ...,$ then $x^k_i * w_{ij}(t) \leqslant y^k_j$, or $w_{ij}(0) = \top$ is a decision, by (3.9). By the definition of the residual, $\forall k \in P,\ x^k_i \to y^k_j = \top$, or $w_{ij}(0) = w^*_{ij}$ by Theorem 1. Therefore, $w^*_{ij} = \top \equiv 1 = w_{ij}(t),\ \forall t \in \{0, 1, ...\}$.

Let us suppose $\lim_t \{w_{ij}\}(t) \leqslant w^*_{ij}$. Let t_0 is the first step such that $w_{ij}(t_0) \leqslant w^*_{ij}$. Then, $x^k_i * w_{ij}(t_0) \leqslant x^k_i * w^*_{ij} = x^k_i * \bigwedge_{k' \in P}(x^{k'}_i \to y^{k'}_j) \leqslant x^k_i * (x^k_i \to y^k_j) \leqslant y^k_j$ by (3.6). Thus, by (3.9), we get $w_{ij}(t_0+1) = w_{ij}(t_0) \leqslant w^*_{ij}$. Let $\lim_t w_{ij}(t) = w_{ij}(t_0) = l_{ij} \leqslant w^*_{ij}$. Then,

$$\lim_t y^k_j(t) = \bigvee_{i \in N} (x^k_i * l_{ij}) \leqslant y^k_j. \tag{3.11}$$

Therefore, by (3.9) and (3.11), we get:

$$\begin{aligned} &\lim_t \{w_{ij}\}(t) = \{l_{ij}\} = \\ &\qquad = \bigcap_k [\lim_t \{w_{ij}\}(t) - sym(\{\lim_t y^k_j(t)\} - \{y^k_j\})] = \\ &\qquad\qquad = \bigcap_k [\{l_{ij}\} - (\{y^k_j\} - \{\lim_t y^k_j(t)\})]. \end{aligned} \tag{3.12}$$

Thus, $\lim_t y^k_j(t) = y^k_j$ for $\forall k \in P$, and $l_{ij} = w_{ij}(t_0) = \lim_t w_{ij}(t) \leqslant w^*_{ij}$ is a decision of (3.4), or $\{l_{ij}\} \bigcap (\{y^k_j\} - \{\lim_t y^k_j(t)\}) = \emptyset$. The latter case means, at least, that $\{l_{ij}\}$ is incomparable with $(\{y^k_j\} - \{\lim_t y^k_j(t)\})$. Thus, $\{y^k_j\} \bigcap \{\lim_t y^k_j(t)\} \supseteq \{y^k_j\} \bigcap \{l_{ij}\}$. Then, $\{y^k_j\} \bigcap \{\lim_t y^k_j(t)\} = \{y^k_j\} \bigcap \{l_{ij}\}$, since $x^k_i * l_{ij} \leqslant l_{ij}$ in integral lattices, and, therefore,

$$\lim_t y^k_j(t) = \bigvee_{i \in N} (x^k_i * l_{ij}) \leqslant l_{ij}. \tag{3.13}$$

Then, let us suppose $\lim_t \{w_{ij}\}(t) = l_{ij} > x^k_i \to y^k_j$ for $\forall k \in P$, hence $l_{ij} > w^*_{ij}$. Then, we get by the definition of the residual: $x^k_i * l_{ij} > y^k_j$ and, therefore,

$$\lim_t y^k_j(t) = \bigvee_{i \in N} (x^k_i * l_{ij}) > y^k_j. \tag{3.14}$$

Hence, we obtain similarly (3.12) with (3.13) that $\lim_t y^k_j(t) = y^k_j$ that contradicts (3.14). Thus, $\lim_t \{w_{ij}\}(t) = l_{ij} \not> x^k_i \to y^k_j$ for $\forall k \in P$ that means $\lim_t \{w_{ij}\}(t) \leqslant \{w^*_{ij}\}$, or $\lim_t \{w_{ij}\}(t)$ is incomparable with $\{w^*_{ij}\}$.

However, $\lim_t\{w_{ij}\}(t)$ may not be incomparable with w^*_{ij}. Indeed, let t_0 is the first step such that $w_{ij}(t_0)$ is incomparable with w^*_{ij}. Then, if $x^k_i * w_{ij}(t_0)$ is incomparable with y^k_j, we get $\lim_t\{w_{ij}\}(t) = l_{ij} = w_{ij}(t_0)$ by (3.9), and $\lim_t y^k_j(t) = \bigvee_{i\in N}(x^k_i * l_{ij})$ is incomparable with y^k_j or $\lim_t y^k_j(t) = \bigvee_{i\in N}(x^k_i * l_{ij}) > y^k_j$. However, the latter case contradicts (3.12) with (3.10) similarly to (3.14) and (3.13). The same holds if $x^k_i * w_{ij}(t_0) > y^k_j$, also similarly to (3.14) and (3.13).

Let $\lim_t y^k_j(t) = \bigvee_{i\in N}(x^k_i * l_{ij})$ is incomparable with y^k_j. However, (3.12) and (3.10) demand then $\{\lim_t y^k_j(t)\} - \{y^k_j\} = \emptyset$ since $(x^k_i * l_{ij}) \leqslant l_{ij}$, and, hence, $\{\lim_t y^k_j(t)\} \subseteq \{l_{ij}\}$.

Now, let us $\lim_t\{w_{ij}\}(t) = l_{ij}$ is incomparable with w^*_{ij}, and $x^k_i * l_{ij} \leqslant y^k_j$. Then, $x^k_i * l_{ij}$ is also incomparable with $x^k_i * w^*_{ij}$, since the lattice is cancellative (otherwise, e.g., $x^k_i * l_{ij} \leqslant x^k_i * w^*_{ij} \Longrightarrow l_{ij} \leqslant x^k_i \to x^k_i * w^*_{ij} = w^*_{ij}$). Hence, it should be $x^k_i * l_{ij} < y^k_j$ and $x^k_i * w^*_{ij} < y^k_j$, otherwise, they are comparable.

Let us consider $l_{ij} \to w^*_{ij} = l_{ij} \to \bigwedge_k x^k_i \to y^k_j$. Thus, $l_{ij} \to w^*_{ij} = \bigwedge_k l_{ij} \to (x^k_i \to y^k_j) = \bigwedge_k (x^k_i * l_{ij}) \to y^k_j$ by (3.7) and (3.8). Therefore, $(x^k_i * l_{ij}) \to y^k_j = 1$ in integral lattices, since $(x^k_i * l_{ij}) < y^k_j$
[13]. Hence, $l_{ij} \to w^*_{ij} = 1$, and $l_{ij} \leqslant w^*_{ij}$ which contradicts their incomparability.

Thus, we obtain the unique result: $\lim_t w_{ij}(t) \leqslant w^*_{ij}$. □

4. EXAMPLE OF A ROBOT GROUP MANAGEMENT

Such a network may be used as an associative memory or pattern classifier in the same manner as $\vee - \wedge$ networks in [2] and "Multi-Valued Neural Networks I". It may be useful in the case of a non-distributive lattice of linguistic variables. However, a systematic method to introduce the monoid multiplication in such a lattice is absent, and we might use heuristics in every particular case. Moreover, the residual intuition treats the multiplication as action: we had something in the past, have something else now, that entails a third thing as a result of the operation. All these reasons lead us to the fact that the networks are more suitable for using, not with patterns (their storing or classifying), but in some leader output detecting.

Thus, we consider here the problem of task distribution in a group of janitor robots. The group is fulfilling a set of tasks. Then, a new task arises. Robots must decide which one (or several) of them will perform the new task. Such a problem was considered in [5] based on linear logic. However, the structure of linear logic is rather intricate and difficult to implement programmatically . The residuated construction is much simpler and allows the task lattice to be determined more adequately.

Let us consider a group of robots that can perform the following tasks: x_1 — trash search, e — taking out the trash, x_3 — sawing of the found garbage when it cannot be taken out, x_2 — the stupor: it is the state of a robot when it tries, but cannot perform a regular task, e.g., the robot cannot pick up the trash, or cannot saw it, or can not move.

All these main tasks have subtasks (Fig. 4.2): trash search x_1 includes d_1 — a move, and d_4 — a video camera operation; trash removing e includes d_1 and d_3 — grabbing of the debris; sawing the garbage x_3 includes d_3 and d_2 — sawing which is not depicted for simplicity. The stupor x_2 includes only the grip: a robot has grabbed a thing, but can do nothing with it. Elements x_2, x_3, d_1, and d_4 may be taken as generators. All others are their meets and joins. Such tasks are, e.g., C_{42} — a robot in a stupor operates with a video camera, or C_{43} — the same during sawing. In this case, we consider the joined tasks as fulfilling in parallel. However, in other cases, we may consider joined tasks as performed sequentially. The lattice in Fig. 4.2 differs from the similar one in [5]: now, we include all joined tasks

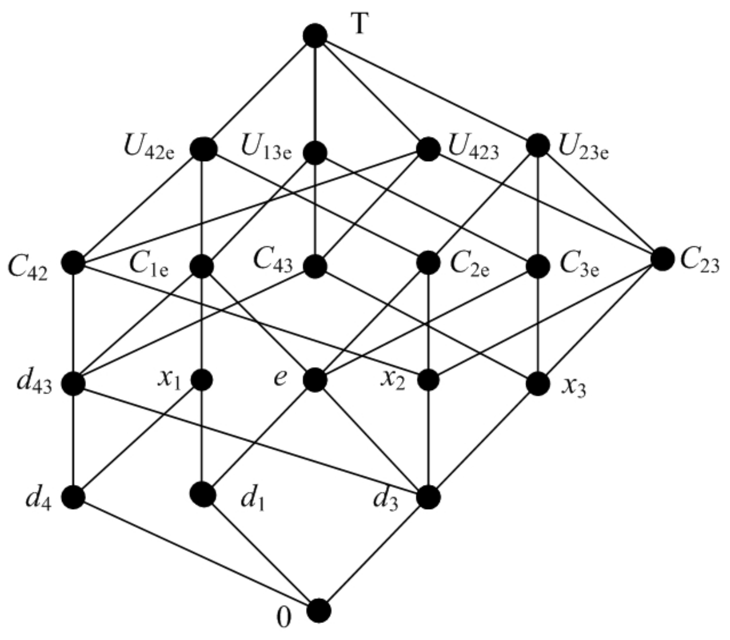

Fig. 4.2. A lattice of system task estimations

which have meanings. However, including all of them may be unnecessary. Then, the lattice may be non-distributive.

Thus, we consider, as in previous papers, the task lattice as the lattice of linguistic variables denoting tasks. Though, the tasks may also be associated with the sets of operations required for a task to be performed. As usually, the lattice partial order orders lattice elements by their values: the higher an element lies in the lattice diagram Fig. 4.2, the greater the value it has. Thus, maximal active system behaviour has the top value $\top$, and activity absent has the bottom value 0. These values estimate transitions $a \to b = c$: the value c estimates the transition from the task a to the task b. Or, once more, "had a then c (arising) entails b."

4.1. Monoid Definition

There is no a general method to define the monoid operation in a residuated lattice. Every time we should use some heuristics. In our case, we demand $C_{42} = 1$, since we want to change the stupor x_2 into the grip d_3 if we obtain the stupor in two consequent iterations: $x_2 x_2 = d_3$[†], though in other cases, e.g., $d_1 d_1 = d_1$[‡]. Therefore, it may be $x_2 < 1$. Subtasks d_3 and d_4 arising should not usually change the current task. Thus, we consider $C_{42} = 1$ as the only *right* neutral element of the monoid multiplication (correspondingly, we consider only the right multiplication and left residuals $x \to y$), since one can hardly give equally meaningful meaning to a left neutral element.

Then, we want $ex_2 = d_3$, i.e., if we get stupor during debris pick up because the robot cannot lift the trash, only the grab remains from the trash remove. Therefore, $d_1 x_2 \vee d_3 x_2 = d_3$, and, hence, $d_1 x_2 = d_3$ or $d_1 x_2 = 0$, and $d_3 x_2 = 0$ or $d_3 x_2 = d_3$. We chose $d_1 x_2 = 0$ (that is evident: the stupor during the move entails an activity cessation) and $d_3 x_2 = d_3$, because it should be for all lattice elements $x1 = x$: $d_1 1 = d_1 x_2 \vee d_1 d_3 \vee d_1 d_4 = d_1$, and, therefore,

[†]We omit the multiplication sign in this section for simplicity.

[‡]We may also demand $x_2 x_2 = 0$ — stupor changes to activity absent.

$d_1x_2 \neq d_3$[§]. Thus, $d_1d_3 = 0$ and $x_2d_3 = d_3$ by monotonicity, and, hence, $d_1d_4 = d_1$. Also, we demand $x_1x_2 = d_4$, i.e., if a robot cannot move during a debris searching, only camera functioning remains. Then, $d_4x_2 = d_4$ — the stupor does not influence camera functioning. Similarly, $x_3x_2 = d_3$ and $x_1d_3 = d_4d_3 = d_4$.

Then, evidently, $x_1d_1 = x_1$, $d_4d_1 = x_1$ (since, a move does not cancel the previous camera operation), $x_1d_4 = x_1$ (since, the robot may stop during the search), and $d_4x_1 = x_1x_1 = x_1$. Hence, $x_1e = x_1$, and $ex_1 = d_3x_1 = e$ since we suppose $d_3d_1 = d_1$ and $d_3d_4 = d_3$ — from $d_31 = d_3$. Thus, e.g., $C_{42} = 1 = d_4 \to d_4$ and $C_{1e} = x_1 \to x_1$. Note, that the robot in search does not normally switch to trash removing, as in ants' colonies, because $x_1e = x_1$ — "had x_1, now e arises, that entails x_1". This result was obtained in [3] [3] from the linear logic structure definition, while here the conclusion is the consequence of our multiplication operation choice.

Now, let us $x_2d_1 = d_1$. Therefore, $x_2x_1 = C_{2e}$, because $x_2d_4 = x_2$ from $x_21 = x_2$. Then, let it be $x_3d_1 = d_1$ and $x_3d_4 = x_3$ — from $x_31 = x_3$ and evidently $x_3d_3 = d_3$. Then, $x_3x_1 = C_{3e}$.

At least, let us consider x_3 multiplication: $d_1x_3 = x_3,\ d_3x_3 = x_3,\ d_4x_3 = C_{43}$. The first two are evident, in the latter case we establish that sawing does not cancel the camera operation. Also, we suppose $x_2x_3 = \top$ since we do not know if the stupor can switch to sawing at this place, or transition to any task (or again to the stupor) is possible. Then, $U_{23e} = x_2 \to \top$.

Thus, we obtain the following basic definitions from the consideration above, though, our choice was sometimes not unique and, therefore, may be changed:

$d_1d_1 = d_1$	$d_3d_3 = d_3$	$d_4d_4 = d_4$	$x_2x_2 = d_3$ *or* 0	$x_3x_3 = x_3$
$d_1d_3 = 0$	$d_3d_1 = d_1$	$d_4d_3 = d_4$	$x_2d_1 = d_1$	$x_3d_1 = d_1$
$d_1d_4 = d_1$	$d_3d_4 = d_3$	$d_4d_1 = x_1$	$x_2d_4 = x_2$	$x_3d_4 = x_3$
$d_1x_2 = 0$	$d_3x_2 = d_3$ *or* 0	$d_4x_2 = d_4$	$x_2d_3 = d_3$	$x_3x_2 = d_3$
$d_1x_3 = x_3$	$d_3x_3 = x_3$	$d_4x_3 = C_{43}$	$x_2x_3 = \top$	$x_3d_3 = d_3$

It is easy to verify that the (right) monoid definition satisfies these equations.

4.2. Neural Network Operation

A variant of the Kohonen network may be used to determine the leading output or outputs in the problem of task distribution in a robot group. The Kohonen net [19] is a bilayer neural network in which the output layer consists of n neurons with m inputs in each neuron. The output of the j-th neuron is obtained according to the following formula:

$$y_j = d_j + \sum_{i \in [0;m]} w_{ij}x_i,$$

where $\mathbf{x} = (x_1, ..., x_m)$ is an input vector, and d_j is a threshold. The network operates by the rule "winner takes it all" which determines the leading output. Residuated lattice operations may be used in the formula as in $\vee - *$ ((3.3)) and $\vee - \wedge$ [2], [1] multi-valued associative memories. Here, we consider only a variant without a threshold, with $w_{ij} = 0$ if $i \neq j$, and with a unique input x (this is the arising task in the robot group). The formula for neuron outputs is also different then:

$$y_i = T_i \vee x \geqslant T_i(T_i \to (T_i \vee x)) \tag{4.15}$$

In the expression, T_i are tasks which the i-th robot supposes to perform (its intentions), and $T_i \vee x$ denotes the i-th robot intention to fulfil task x together with the other robot tasks.

[§] If we chose $d_3x_2 = 0$, then $ex_2 = 0$ that is also possible.

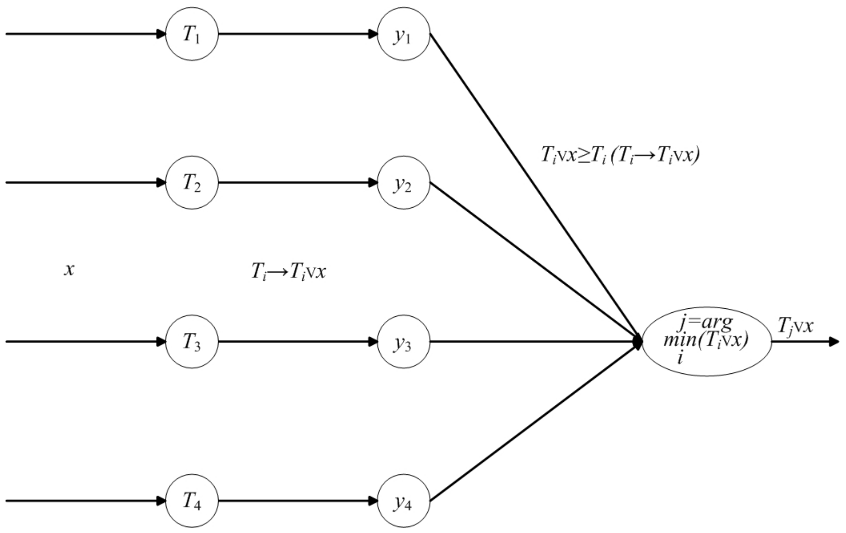

Fig. 4.3. The neural network topology for the system task choice

The right part of (4.15) is a consequent of (3.3) and corresponds to the usual logic rule: $[a \wedge (a \Rightarrow b)] \Rightarrow b$. That means we obtain b if we have a, and a entails b. The formula has the same meaning in the residuated case: $ac \leqslant b$, where $c \leqslant a \to b$. Hence, we take $T_i \vee x$ as an output and we might have an extremal principal to choose the necessary output. We take here $min_i[T_i \vee x]$ as the principal, because it is naturally to chose a robot with a minimal set of tasks which it is supposed to perform. Clearly, we take several such outputs in the absence of the minimum.

Thus, we consider the neural network depicted in Fig. 4.3, in which the input variable x is an arising task in the robot group which is fulfilling tasks T_i at the current time. An arising task may be here x_1, e, and x_3, since stupor x_2 refers to only one concrete robot, but not to all of them. We might choose now the necessary output for different input cases and tasks performed. Here, we will use an extended form for outputs instead of (4.15): $T_i x \leqslant T_i \vee x$ ("there was T_i, now x arises, entails $T_i \vee x$"), in the case of $x \leqslant T_i \to (T_i \vee x)$.

Let us suppose a standard situation: robots are looking for a trash and remove it. Then, if a search task x_1 arises, some of the robots already performing x_1 will include this new task into their intention set. Indeed, $x_1 x_1 = x_1$ is the minimal intention of joined tasks. A robot fulfilling subtask d_4 of x_1 (the camera operation) may also include the new task into its intentions, since $d_4 x_1 = x_1$. Such a state d_4 may happen if a searching robot falls into a stupor: $x_1 x_2 = d_4$. A robot moving without any other task (subtask d_1, e.g., which return after the trash remove) may also switch to this new one, since $d_1 x_1 = d_1 < x_1$.

If a task e of taking out the trash arises, then it will be performed by a robot already removing the debris: $ee = e$ is the minimal value in the standard situation. Though, a moving robot (in d_1) or a grabbing one (in d_3) may also switch to taking out: $d_1 e = d_1 < e$, $d_3 e = e$. The latter case arises after sawing when only grabbing remains. Thus, sawn-off pieces may be removed by the same robot that has sawn them off, or one robot can saw, and another remove the sawn-off pieces.

If a task of garbage sawing x_3 arises, then it will be performed by one of robots already fulfilling sawing ($x_3 x_3 = x_3$), or by a robot in grabbing: $d_3 x_3 = x_3$ — these values are minimal. The latter situation arises when the robot picking up the debris can not lift it: $ex_2 = d_3$ — only the grabbing remains. If the robot cannot saw the trash, it switches to grabbing: $x_3 x_2 = d_3$. However, in this situation, no regular task can be performed, and collective removing is required: $d_3 e = e$, where the task e is a part of the collective action. But this decision is external to the neural network application. Also, such an external decision is needed if the robot cannot grab the trash: $ex_2 = d_3$, then, $d_3x_2 = d_3$, and we obtain in two iterations $x_2x_2 = d_3$ or $x_2x_2 = 0$.

Thus, we see that the system of robots behaves quite reasonable, and the behaviour is due only to our choice of basic multiplications in the residuated lattice.

5. CONCLUSION

A new concept of multi-valued neural networks with variables and weights taking values in a residuated lattice was introduced in this study. The concept continues a row of studies in which the state of a system is estimated not by numbers, but by elements of a partially ordered set, namely, the lattice. In our case the lattice is finite and residuated, and it may consist of some linguistic variables. Such an approach can facilitate the assessment of the situation by experts in cases requiring their participation.

We have expanded the results on fuzzy neural networks to such a $\vee - *$ multi-valued case in which variables may take values in a non-distributive lattice. We found out the conditions under which it is possible to store in the multi-valued associative memory given pairs of network variable patterns. We also gave the learning algorithm that generalizes the fuzzy one for the multi-valued case. However, the algorithm is suitable in the special case of the residuated lattice: it should be integral.

Such a network may be used as an associative memory or a pattern classifier like the $\vee - \wedge$ multi-valued neural network. However, here, we gave the example of such a Kohonen-like network using it in a janitor robot group management.

REFERENCES

1. Maximov D. (2020) Making a decision on the management of a group of unmanned aerial vehicles by using multi-valued networks, *in Russian Materials of the 13th International Conference 'Management of Large-Scale System Development' (MLSD'2020)*, Moscow, Trapeznikov Institute of Control Science Russian Academy of Science.
2. Maximov D. (2020) Multi-Valued Neural Networks and their Use in Decision Making on the Management of a Group of Unmanned Vehicles, *Proceedings of the 13th International Conference "Management of Large-Scale System Development" (MLSD)*, IEEE, 1–5. https://ieeexplore.ieee.org/document/9247800.
3. Maximov D. Yu., Legovich YU. S., Ryvkin S. (2017) How the Structure of System Problems Influences System Behavior, How the Structure of System Problems Influences System Behavior, *Automation and Remote Control*, **78**(4), 689–699.
4. Maximov D. (2019) An Optimal Itinerary Generation in a Configuration Space of Large Intellectual Agent Groups with Linear Logic, *Advances in Systems Science and Applications*, **19**(4), 79–86. https://ijassa.ipu.ru/index.php/ijassa/article/view/829/513
5. Maximov D., Ryvkin S. (2017) Systems smart effects as the consequence of the systems complexity, *Proc. 17th International Conf. on Smart Technologies (IEEE EUROCON 2017, Ohrid)*, Ohrid, IEEE, 576–582.
6. Maximov D., Ryvkin S. (2019) Multi-valued logic in graph transformation theory and self-adaptive systems, *Annals of Mathematics and Artificial Intelligence*, **87**(4), 395–408.
7. Maximov D. (2019) Control in a Group of Unmanned Aerial Vehicles Based on Multi-Valued Logic, *Proc. of the 12th International Conference 'Management of Large-Scale System Development' (MLSD'2019)*, Providence, IEEE, 1–5. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8911092
8. Liu P., Li H. (2004) Fuzzy neural network theory and application, *Series in Machine Perception and Artificial Intelligence*,**59**, London: World Scientific Publishing Co. Pte.

Ltd.
9. Kosko B. (1987) Fuzzy associative memories, *Fuzzy Expert Systems Reading*, MA: Addison-Weley.
10. Birkhoff G. (1967) *Lattice Theory*, Rhode Island: Providence.
11. Blount K. , Tsinakis C. (2003) The structure of residuated lattices, *Int. J. Algebra Comput.* (13), 437–461.
12. Bahls P., Cole J., Galatos N., Jipsen P., Tsinakis C. (2003) Cancellative residuated lattices, *Algebr. Univ.*, **50**(1), 83–106.
13. Gil-Férez J., Lauridsen F. M., Metcalfe G. (2019) Integrally closed residuated lattices, *Stud. Logica*, 1–24. https://doi.org/10.1007/s11225-019-09888-9
14. Meneganti M., Saviello F. S., Tagliaferri R. (1998) Fuzzy neural networks for classification and detection of anomalies, *IEEE Trans. on Neural Networks*, (9), 848–861.
15. Sussner P., Valle M. E. (2006) Implicative Fuzzy Associative Memories, *EEE Transactions on Fuzzy Systems*, **14**(6), 793–807.
16. Li X. Z., Ruan D. (1997) Novel neural algorithms based on fuzzy δ rules for solving fuzzy relation equations: Part I, *Fuzzy Sets and Systems*, **90**, 11–23.
17. Li X. Z., Ruan D. (1999) Novel neural algorithms based on fuzzy δ rules for solving fuzzy relation equations: Part I, *Fuzzy Sets and Systems*, **103**, 473–486.
18. Li X. Z., Ruan D. (2000) Novel neural algorithms based on fuzzy δ rules for solving fuzzy relation equations: Part I, *Fuzzy Sets and Systems*, **109**, 355–362.
19. Kohonen T. (1989) *Self-Organizing Maps*, Berlin-New-York: Springer-Verlag.