Multigrammatical modelling of neural networks

I.A. Sheremet [1]
[1] *Geophysical Center of Russian Academy of Sciences, 119296, Russia, Moscow, Molodezhnaya St. 3*

Abstract

This paper is dedicated to the proposed techniques of modelling artificial neural networks (*NNs*) by application of the multigrammatical framework. Multigrammatical representations of feed-forward and recurrent *NNs* are described. Application of multiset metagrammars to modelling deep learning of *NNs* of the aforementioned classes is considered. Possible developments of the announced approach are discussed.

Keywords: neural networks, multiset grammars, multiset metagrammars, deep learning.

Citation: Sheremet IA. Multigrammatical Modelling of Neural Networks. Computer Optics 2024; 48 (4): 619-632. DOI: 10.18287/2412-6179-CO-1436.

Introduction

Multiset grammars (*MG*s) and multiset metagrammars (*MMG*s), introduced in [1, 2] and thoroughly described in [3], form a deeply integrated multigrammatical framework (*MGF*) being a mathematical toolkit and a knowledge representation model, originating from the theory of multisets [4] and suitable for solution of a wide spectrum of complicated problems from systems analysis, operations research and digital economy areas.

Investigating real capabilities of this relatively novel formalism, it would be reasonable to compare it by its descriptive power with formal systems already known in the modern computer science. Such comparison was done in [1, 2, 3], where Petri nets, systems of vectors addition-substitution, and augmented Post systems were considered, as well as various classes of problems of the operations research, which interconnections with the MGF were studied; namely, it was demonstrated how specific problems formulated by means of the classical operations research toolkit would be represented by application of the *MGF*.

However, the aforementioned list would be definitely incomplete, if not to consider such widely applied and intensively developed artificial intelligence (*AI*) tool as artificial neural networks (*ANN*s) which for short will be referred below as neural networks (*NN*s).

This paper is dedicated to multigrammatical modelling of neural networks as a way of a comparative study of the *MGF* and *NN*s. Our approach is fully consistent with foundations of mathematical logic, where two classes of entities are considered – axiomatic systems (first order predicate calculus [5], Horn clauses [6, 7], grammars [8, 9], Post systems [10], augmented Post systems [11 – 13] etc.) and devices, usually considered as computational models (Turing machines [14], Petri nets [15], systolic structures [16], cellular automata [17], etc.), and the last enable more or less controlled and/or parallelized application of inference rules to input data for achieving objectives. Artificial neural networks from the early times [18, 19] were announced as a mathematical model of a human brain and today are considered as a flexible computational model with a massive parallelism, so a lot of comprehensive neural-based solutions of various classes of complicated practical problems were designed and continue to be on search: computer vision, speech recognition, natural language understanding, radio signals processing, robots control, healthcare etc. [20 – 27]. The most attractive feature of *NN*s is their relatively simple learning from training data sets (*TDS*s) [28 – 33]. By this feature *NN*s are much more convenient in application than inference-based engines demanding on preparation and maintenance of large knowledge bases by experienced knowledge engineers. At the same time *NN*s are rather vulnerable to criticism on resilience to fluctuations in training data sets: to what extent would change a logic of a trained *NN* operation if some part of a training data set (in the simplest case, the only one training sample) would be replaced? And, at all, how representative is an applied *TDS* to create a neural-based device, applicable to any of situations which may occur in future, and would this device correctly operate in such situations? These and a lot of similar questions arise after acquisition of some experience in application of trained *NN*s, and not by the chance the most advanced scholars express their worry about possibility of the unpredicted behavior of an *NN*-based *AI* [34, 35].

To deal with the whole spectrum of issues associated with *NN*s deep learning and post-training application, it would be rational to consider artificial neural networks not alone but regarding already known objects of mathematical logic and knowledge engineering, and some steps in this direction are already known: neural Turing machines as well as Chomsky grammars being a very peculiar examples [36, 37]. The cited papers demonstrate how NNs may be applied to model classical objects and thus prove their universalism in the sense of the theoretical computer science. Our paper follows the inverse direction: **how axiomatic systems** (namely, *MG*s and *MMG*s) **may be applied in order to model** *NN*s. Such approach may open some new research opportunities unavailable on the current device-based groundwork of *NN*s, and in this quality it may be assessed as some very modest contribution to primary **logical foundations of neural networks**.

As it will be shown below in the sections 2 and 3, any feed-forward neural network (*FNN*) may be represented by a multiset grammar implementing **the same mapping** from a set of possible input collections to a set of possible output collections of this *FNN*, and any recurrent neural network (*RNN*) – by a sequence of *MG*s with one and the same scheme and kernels being sums of a multisets, representing current steps inputs and previous steps feedbacks. As even more inquisitive result, which is described in the section 4, logics of deep learning of any *FNN* may be also modelled on a regular basis by a set of multiset metagrammars with multiplicities-**variables**, representing domains of possible weights of connections between NN nodes, and **any such *MMG* is induced by one training sample**. So a set of possible collections of the aforementioned weights, corresponding the whole training data set, is a result of intersection of similar sets, corresponding separate training samples. The most interesting result of this paper, considered in the section 5, is representation of logics of deep learning of *RNN*s by a sequence of one *MMG*, corresponding to the first step (the first training sample), and n-1 *MG*s, corresponding to the second and following steps (training samples). A set of collections of weights satisfying a whole data set is obtained from a result of the first step by exclusion such collections, which do not correspond at least one of the following n-1 training samples. Finally, possible developments of the considered approach to multigrammatical modelling of NNs are discussed in the section 6. The Appendix contains a short introduction to the *MGF*.

1. Multigrammatical representation of feed-forward neural networks

Any artificial feed-forward neural network [30–33] may be represented by an ***acyclic*** weighted oriented graph which in turn may be represented as a ternary relation

$$G \subset A \times A \times W, \tag{1}$$

where A is a set of nodes (neurons), and W is a set of possible weights of nodes connections (synapses), so $a_i, a_j, w_{ij} \in G$ means that a connection (synapse) from a_i to a_j has a weight w_{ij}. Logics of operation of an *FNN*, along with its topology, is determined by a vector H which components h_i are threshold values, so if a sum of weights of active input connections of a node a_i is greater than h_i then a node a_i is activated, and its output connections obtain the value 1. Such feed-forward propagation is done until an output layer which connections represent a result of operation of an *FNN* G, H. So, in fact, this neural network implements mapping from a set of binary vectors of values of input connections to a set of binary vectors of values of output connections. Let us construct some multiset grammar $S(G,H) = v_0, R$ which would implement the same mapping.

We shall include to a set of objects $\boldsymbol{A}$ of this *MG* as subsets a set A of nodes of an *FNN* G, H, a set $\boldsymbol{A}_{in}$ of its inputs and a set $\boldsymbol{A}_{out}$ of its outputs. Rules of this *MG* will be constructed as follows.

Let $A_i = \{a^i_1, ..., a^i_{m_i}\} \subset A$ be a subset of set of nodes of a considered *FNN*, connected with a node a_i by his output connections; $w^i_1, ..., w^i_m$ be weights of the aforementioned connections; and h_i be an input threshold value of this node. We shall define for any such node a rule

$$\{h_i \cdot a_i\} \to \{w^i_1 \cdot a^i_1, \ldots, w^i_{m_i} \cdot a^i_{m_i}\}, \tag{2}$$

and this operation will be done for all nodes $a \in A$. Inputs will be represented by rules

$$\{1 \cdot \boldsymbol{a}_i\} \to \{1 \cdot a_j\}, \tag{3}$$

where $\boldsymbol{a}_i$ is an input and a_j is a node belonging to an input layer $A_{in} \subset A$ (in a general case one input may be connected to several nodes but a node may be connected with the only input). Outputs will be represented by rules

$$\{h_i \cdot a_i\} \to \{1 \cdot \boldsymbol{a}_j\}, \tag{4}$$

where $\boldsymbol{a}_j$ is an output and $a_i$is a node belonging to an output layer $A_{out} \subset A$ (we also allow that one node may be connected to several outputs but one output may be connected to the only node).

As it is acceptable in the *MGF*, values h_i and w^i_j may be any rational numbers [3].

Statement 1. Let $A_{out} = \{\boldsymbol{a}_{j1}, \ldots, \boldsymbol{a}_{jl}\} \subset \boldsymbol{A}_{out}$ be a set of outputs of an *FNN* G, H activated by a set of inputs $A_{in} = \{\boldsymbol{a}_{j1}, \ldots, \boldsymbol{a}_{jm}\} \subset \boldsymbol{A}_{in}$, and $S(G,H) = 1 * A_{in}, R$ where R is a set of rules defined by (2). Then

$$\{1 * A_{out}\} = \bar{V}_{S(G,H)} \Cap 1 * \boldsymbol{A}_{out}. \tag{5}$$

Proof. As seen, a kernel of an *MG* $S(G,H)$ is a multiset containing multiobjects $1 \cdot \boldsymbol{a}_i$, where $\boldsymbol{a}_i \in \boldsymbol{A}_{in}$. Following semantics of multisets generation, any rule with the left part $\{h_i \cdot a_i\}$ at any generation step would be relevant to a current multiset containing a multiobject $h \cdot a_i$ if and only if $h_i \le h$. From the *NNs* theory point of view application of a relevant rule to a current *MS* is equivalent to activation of a neuron a_i by an input which value h exceeds a threshold value h_i of this neuron. From the other side, the aforementioned value would be a sum of multiplicities accumulated as a result of application of rules having multiobjects $w \cdot a_i$ in their right parts. However, multisets generated by application of a scheme R would contain not only multiobjects which objects enter a set $\boldsymbol{A}_{out}$, but also other multiobjects which multiplicities are not sufficient for relevancy (and thus application) of some one rule. From the other side, multiplicities of objects, entering a set $\boldsymbol{A}_{out}$, may be greater than 1 due to possibility of multiple application of one and the same rules in an *MSs* generation process (this is equivalent to multiple activation of one and the same node, and such feature is in a general case possible in a multigrammatical model of an *NN*).

However, due to the fact, that a scheme *R* of an *MG* v_0, R includes no more than one rule with the left part containing a multiobject a_i, a set $\bar{V}_{S(G,H)}$ generated by this *MG*, will contain the only *TMS*. Hence, a set $\bar{V}_{S(G,H)} \Cap 1 * A_{out}$ in any case will contain the only terminal multiset which multiobjects are $1 \cdot \boldsymbol{a}_i$, where $\boldsymbol{a}_i \in A_{out}$. Returning to a modelled neural network, it means that this *TMS* represents nothing but a set of activated output nodes of this *NN*, i.e. A_{out}. ■

This statement confirms that a ***constructed FMG implements the same mapping*** from a set of values of inputs to a set of values of outputs ***as an initial NN***.

Due to application of the most general basic graph representation of *FNNs* all the said is true for **any particular case of such neural networks**.

Example 1. Consider the multi-layer perceptron depicted in Fig. 1. This *FNN* contains the input layer (nodes a_1, a_2, a_3), the hidden layer (nodes b_1, b_2, b_3, b_4), and the output layer (nodes c_1, c_2).

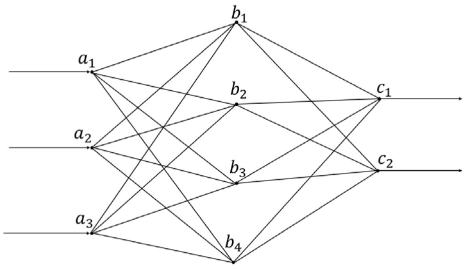

Fig. 1. Modelled multi-layer perceptron's topology

Tab. 1. Input layer of modelled multi-layer perceptron

a_1	b_1	b_2	b_3	b_4
	1	3	2	4
a_2	b_1	b_2	b_3	b_4
	1	5	1	1
a_3	b_1	b_2	b_3	b_4
	2	1	3	4

Tab. 2. Hidden layer of modelled multi-layer perceptron

b_1	c_1	c_2
	3	2
b_2	c_1	c_2
	1	1
b_3	c_1	c_2
	2	4
b_4	c_1	c_2
	3	1

Tab. 3. Threshold values of modelled multi-layer perceptron

a_1	a_2	a_3	b_1	b_2	b_3	b_4	c_1	c_2
1	1	1	4	3	3	4	15	6

Weights of connections from the nodes entering the input layer to the nodes entering the hidden layer are presented in Table 1., the same data regarding the hidden and the output layers respectively – in Table 2., and the threshold values of all nodes – in Table 3. We use here integer values of weights for shortening a record; anyone who would desire to use rational numbers may apply 0. *n* instead of *n* (for example, 0.5 instead of 5).

According to (2)–(4), this *FNN* may be represented by the scheme *R* containing following rules, which names are divided from their bodies by the delimiter ":":

$$r_1 : \{1 \cdot \boldsymbol{a}_1\} \to \{1 \cdot a_1\},$$
$$r_2 : \{1 \cdot \boldsymbol{a}_2\} \to \{1 \cdot a_2\},$$
$$r_3 : \{1 \cdot \boldsymbol{a}_3\} \to \{1 \cdot a_3\},$$
$$r_4 : \{1 \cdot a_1\} \to \{1 \cdot b_1, 3 \cdot b_2, 2 \cdot b_3, 4 \cdot b_4\},$$
$$r_5 : \{1 \cdot a_2\} \to \{1 \cdot b_1, 5 \cdot b_2, 1 \cdot b_3, 1 \cdot b_4\},$$
$$r_6 : \{1 \cdot a_3\} \to \{2 \cdot b_1, 1 \cdot b_2, 3 \cdot b_3, 4 \cdot b_4\},$$
$$r_7 : \{4 \cdot b_1\} \to \{3 \cdot c_1, 2 \cdot c_2\},$$
$$r_8 : \{3 \cdot b_2\} \to \{1 \cdot c_1, 1 \cdot c_2\},$$
$$r_9 : \{3 \cdot b_3\} \to \{2 \cdot c_1, 4 \cdot c_2\},$$
$$r_{10} : \{4 \cdot b_4\} \to \{3 \cdot c_1, 1 \cdot c_2\},$$
$$r_{11} : \{15 \cdot c_1\} \to \{1 \cdot \boldsymbol{a}_4\},$$
$$r_{12} : \{6 \cdot c_2\} \to \{1 \cdot \boldsymbol{a}_5\}.$$

Let the input nodes $\boldsymbol{a}_1$ and $\boldsymbol{a}_3$ are activated. Then the multiset grammar

$$S(G,H) = \{1 \cdot \boldsymbol{a}_1, 1 \cdot \boldsymbol{a}_3\}, R,$$

is equivalent to this *FNN*. Consider generation of the set of terminal multisets by application of the scheme of this *MG* to its kernel:

$$\{1 \cdot \boldsymbol{a}_1, 1 \cdot \boldsymbol{a}_3\} \overset{r_1}{\Rightarrow}$$
$$\{1 \cdot a_1, 1 \cdot \boldsymbol{a}_3\} \overset{r_3}{\Rightarrow}$$
$$\{1 \cdot a_1, 1 \cdot a_3\} \overset{r_4}{\Rightarrow}$$
$$\{1 \cdot b_1, 3 \cdot b_2, 2 \cdot b_3, 4 \cdot b_4, 1 \cdot a_3\} \overset{r_6}{\Rightarrow}$$
$$\{3 \cdot b_1, 4 \cdot b_2, 5 \cdot b_3, 8 \cdot b_4\} \overset{r_8}{\Rightarrow}$$
$$\{3 \cdot b_1, 1 \cdot b_2, 5 \cdot b_3, 8 \cdot b_4, 1 \cdot c_1, 1 \cdot c_2\} \overset{r_9}{\Rightarrow}$$
$$\{3 \cdot b_1, 1 \cdot b_2, 2 \cdot b_3, 8 \cdot b_4, 3 \cdot c_1, 5 \cdot c_2\} \overset{r_{10}}{\Rightarrow}$$
$$\{3 \cdot b_1, 1 \cdot b_2, 2 \cdot b_3, 4 \cdot b_4, 6 \cdot c_1, 6 \cdot c_2\} \overset{r_{10}}{\Rightarrow}$$
$$\{3 \cdot b_1, 1 \cdot b_2, 2 \cdot b_3, 4 \cdot b_4, 6 \cdot c_1, 6 \cdot c_2\} \overset{r_{10}}{\Rightarrow}$$
$$\{3 \cdot b_1, 1 \cdot b_2, 2 \cdot b_3, 9 \cdot c_1, 7 \cdot c_2\} \overset{r_{12}}{\Rightarrow}$$
$$\{3 \cdot b_1, 1 \cdot b_2, 2 \cdot b_3, 9 \cdot c_1, 1 \cdot c_2, 1 \cdot \boldsymbol{a}_5\}$$
$$\{\{3 \cdot b_1, 1 \cdot b_2, 2 \cdot b_3, 9 \cdot c_1, 1 \cdot c_2, 1 \cdot \boldsymbol{a}_5\}\} \Cap$$
$$(1 * \{\boldsymbol{a}_4, \boldsymbol{a}_5\}) = \{\{0 \cdot \boldsymbol{a}_4, 1 \cdot \boldsymbol{a}_5\}\}.$$

As seen, the output $\boldsymbol{a}_5$ of the considered *FNN*, activated by inputs $\boldsymbol{a}_1$ and $\boldsymbol{a}_3$, will be activated as a response.■

We have applied the only sequence of rules because the order of their selection at any current step of generation is immaterial. A concerned reader may apply any other sequence to confirm the identity of obtained results.

Let us consider now the most general and powerful class of *NNs*, namely, recurrent neural networks.

2. Multigrammatical representation of recurrent neural networks

Any artificial recurrent neural network may be represented by an weighted oriented graph which has at least one cycle, i.e. a sequence of connections starting and finishing at some node of this graph [31, 32, 33]. Topology of any such network similarly to *FNNs* may be represented as a ternary relation (1), but an attempt to apply the same technique of multigrammatical representation of neural networks as above would fail because dynamics of *RNN* operation presumes not a single input-output step (finite impulse response, *FIR*) but “upon a time” operation including a sequence of such steps (infinite impulse response, *IIR*) in such a way that outputs of some neurons being a result of a previous step serve as inputs of neurons of an *NN* at the current step, and thus such outputs work as some kind of memory. To model such operation, we shall propose below some more sophisticated technique.

Consider a rule $\{h_i \cdot a_i\} \to \{w_1^i \cdot a_1^i, \ldots, w_{m_i}^i \cdot a_{m_i}^i\}$, where some a_j^i is an object representing a node belonging to some of previous layers already passed inside a current step of forward propagation initiated by activation of input nodes of a network. If we would try to model operation of such network without any changes regarding the *FNN* case, then we would obtain a cyclic multiset grammar generating in a general case an infinite set of responses given one input set. To avoid such senseless result we shall replace every object a_j^i entering the right part of some rule and possessing a described above feature by a new object $\boldsymbol{a}_j^i$ distinguished from a_j^i and from all other objects. A set of such new objects called ***feedback objects*** will be denoted $\boldsymbol{A}_+$.

A multiset grammar $S(G,H) = v_{in}, R$, which is, due to the aforementioned transformation, acyclic and non-alternating, enables generation of a response v_{out} of a network at a ***current*** step in accordance with the Statement 1. At the same time objects, being elements of a set $\boldsymbol{A}_+$ and entering a set

$$\bar{V}_{v_{in},R},$$

enable transfer of information, obtained at a current step, to the ***next*** step of operation of a considered *NN*, and this is implemented by application of a new *MG*

$$S'^{(G,H)} = v'_{in} + v_+, R, \qquad (6)$$

where v'_{in} is an input coming at the next step, and

$$\{v_+\} = \bar{V}_{v_{in},R} \Cap \boldsymbol{N}{*}\boldsymbol{A}_+. \qquad (7)$$

(By this technique we can select from a set of *TMSs* generated by an *MG* v_{in}, R all multiobjects which objects enter a set $\boldsymbol{A}_+$ because, let us remind, for any multiplicity n $\min\{n, \boldsymbol{N}\} = n$). We shall call a multiset v_+ a ***feedback***. As seen from (6) – (7), a generation step modelling the next step of operation of a considered *NN* starts from a kernel including new input information as well as information obtained at the previous step; the last may be applied as soon as multiobjects, representing nodes belonging to a recurrently activated layer, would appear in a generated multiset with multiplicities sufficient for application of proper rules (thus, for activation of corresponding nodes). Naturally, in a general case a number of such recurrently activated layers may be greater than one.

Now, generalizing (6) – (7), we may write equations determining a result of the *i*-th step of operation of a considered *NN*. It is obtained by application of an *MG*

$$S_i(G,H) = v_{in}^i + v_+^{i-1}, R, \qquad (8)$$

so

$$\{v_{out}^i\} = \bar{V}_{S_i(G,H)} \Cap \{1{*}\boldsymbol{A}_{out}\}. \qquad (9)$$

$$\{v_+^i\} = \bar{V}_{S_i(G,H)} \Cap \{\boldsymbol{N}{*}\boldsymbol{A}_+\}. \qquad (10)$$

As seen, v_+^{i-1} in (8) really works as a memory, enabling transfer of results, obtained at previous steps, to a current step.

Statement 2. An infinite impulse response of any artificial recurrent neural network G, H may be represented by application of a sequence of multiset grammars determined by (8) – (10). ■

A special proof of this statement is redundant because of the detailed description of logics of construction of the aforementioned sequence.

Example 2. Consider the recurrent neural network depicted in Fig. 2. This *NN* contains the input layer (nodes a_1, a_2, a_3, two hidden layers (the first – nodes b_1, b_2, b_3 – and the second – nodes c_1, c_2), as well as the output layer (nodes d_1, d_2).

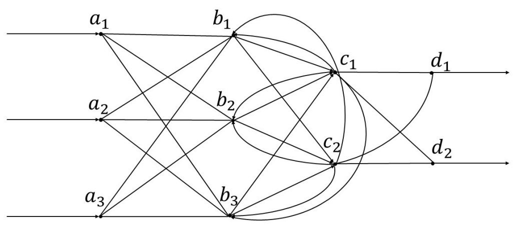

Fig. 2. Modelled recurrent neural network’s topology

Tab. 4. Input layer of modelled recurrent neural network

a_1	b_1	b_2	b_3
	3	2	1
a_2	b_1	b_2	b_3
	1	4	3
a_3	b_1	b_2	b_3
	2	1	4

Tab. 5. The first hidden layer of modelled recurrent neural network

b_1	c_1	c_2
	1	3
b_2	c_1	c_2
	2	2
b_3	c_1	c_2
	4	1

Tab. 6. The second hidden layer of modelled recurrent neural network

c_1	d_1	d_2	b_1	b_2	b_3
	1	2	3	2	4
c_2	d_1	d_2	b_1	b_2	b_3
	2	3	1	1	3

Tab. 7. Threshold values of modelled recurrent neural network

a_1	a_2	a_3	b_1	b_2	b_3	c_1	c_2	d_1	d_2
1	1	1	3	4	5	8	4	11	9

As seen, this *NN* is recurrent because outputs of the nodes c_1, c_2 of the second hidden layer are connected to the nodes b_1, b_2, b_3 of the first hidden layer. Weights of connections from the nodes entering the input layer to the nodes entering the first hidden layer are presented in Table 4., similar information regarding the second hidden layer and the output layer – in Tab. 6, and the threshold values of all nodes – in Tab. 7.

According to $(6)-(10)$, this *RNN* may be represented by the scheme *R* containing following rules (here for simplification of a record we shall use $\boldsymbol{b}_i$ for denotation of feedback objects):

$$r_1 : \{1\cdot\boldsymbol{a}_1\} \to \{1\cdot a_1\},$$
$$r_2 : \{1\cdot\boldsymbol{a}_2\} \to \{1\cdot a_2\},$$
$$r_3 : \{1\cdot\boldsymbol{a}_3\} \to \{1\cdot a_3\},$$
$$r_4 : \{1\cdot a_1\} \to \{3\cdot b_1, 2\cdot b_2, 1\cdot b_3\},$$
$$r_5 : \{1\cdot a_2\} \to \{1\cdot b_1, 4\cdot b_2, 3\cdot b_3\},$$
$$r_6 : \{1\cdot a_3\} \to \{2\cdot b_1, 1\cdot b_2, 4\cdot b_3\},$$
$$r_7 : \{3\cdot b_1\} \to \{1\cdot c_1, 3\cdot c_2\},$$
$$r_8 : \{4\cdot b_2\} \to \{2\cdot c_1, 2\cdot c_2\},$$
$$r_9 : \{5\cdot b_3\} \to \{4\cdot c_1, 1\cdot c_2\},$$
$$r_{10} : \{8\cdot c_1\} \to \{1\cdot d_1, 2\cdot d_2, 3\cdot\boldsymbol{b}_1, 2\cdot\boldsymbol{b}_2, 4\cdot\boldsymbol{b}_3\},$$
$$r_{11} : \{4\cdot c_2\} \to \{2\cdot d_1, 3\cdot d_2, 1\cdot\boldsymbol{b}_1, 1\cdot\boldsymbol{b}_2, 3\cdot\boldsymbol{b}_3\},$$
$$r_{12} : \{12\cdot d_1\} \to \{1\cdot\boldsymbol{a}_4\},$$
$$r_{13} : \{11\cdot d_2\} \to \{1\cdot\boldsymbol{a}_5\}.$$

Let the input nodes $\boldsymbol{a}_2$ and $\boldsymbol{a}_3$ are activated at the first step of the *RNN* operation. Then the multiset grammar

$$S_1(G,H) = \{1\cdot\boldsymbol{a}_2, 1\cdot\boldsymbol{a}_3\}, R\},$$

models this first step enabling generation of the set of terminal multisets by application of the scheme of this *MG* to its kernel:

$$\{1\cdot\boldsymbol{a}_2, 1\cdot\boldsymbol{a}_3\} \overset{r_2}{\Rightarrow}$$
$$\{1\cdot a_2, 1\cdot\boldsymbol{a}_3\} \overset{r_3}{\Rightarrow}$$
$$\{1\cdot a_2, 1\cdot a_3\} \overset{r_5}{\Rightarrow}$$
$$\{1\cdot b_1, 4\cdot b_2, 3\cdot b_3, 1\cdot a_3\} \overset{r_6}{\Rightarrow}$$
$$\{5\cdot b_2, 7\cdot b_3, 4\cdot c_1, 1\cdot c_2\} \overset{r_7}{\Rightarrow}$$
$$\{1\cdot b_2, 7\cdot b_3, 6\cdot c_1, 3\cdot c_2\} \overset{r_9}{\Rightarrow}$$
$$\{1\cdot b_2, 2\cdot b_3, 10\cdot c_1, 4\cdot c_2\} \overset{r_{10}}{\Rightarrow}$$
$$\{1\cdot b_2, 2\cdot b_3, 2\cdot c_1, 4\cdot c_2, 1\cdot d_1, 2\cdot d_2, 3\cdot\boldsymbol{b}_1, 2\cdot\boldsymbol{b}_2, 4\cdot\boldsymbol{b}_3\} \overset{r_{11}}{\Rightarrow}$$
$$\{1\cdot b_2, 2\cdot b_3, 2\cdot c_1, 3\cdot d_1, 5\cdot d_2, 4\cdot\boldsymbol{b}_1, 3\cdot\boldsymbol{b}_2, 7\cdot\boldsymbol{b}_3\} \overset{r_{12}}{\Rightarrow}$$
$$\{1\cdot b_2, 2\cdot b_3, 2\cdot c_1, 1\cdot d_1, 5\cdot d_2, 4\cdot\boldsymbol{b}_1, 3\cdot\boldsymbol{b}_2, 7\cdot\boldsymbol{b}_3, 1\cdot\boldsymbol{a}_4\}.$$

So, according to $(9)-(10)$,

$$v^1_{out} = \{1\cdot\boldsymbol{a}_4, 0\cdot\boldsymbol{a}_5\},$$
$$v^1_{+} = \{4\cdot\boldsymbol{b}_1, 3\cdot\boldsymbol{b}_2, 7\cdot\boldsymbol{b}_3\}.$$

As seen, the output $\boldsymbol{a}_4$ of the considered recurrent *NN*, activated by inputs $\boldsymbol{a}_2$ and $\boldsymbol{a}_3$, will be activated as a response after the first step of its operation, and the feedback of the first step will be $\{4\cdot\boldsymbol{b}_1, 3\cdot\boldsymbol{b}_2, 7\cdot\boldsymbol{b}_3\}$.

Now let at the second step of the *RNN* operation the input nodes $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ be activated. Then the multiset grammar

$$S_2(G,H) = \{1\cdot\boldsymbol{a}_1, 1\cdot\boldsymbol{a}_2, 4\cdot b_1, 3\cdot b_2, 7\cdot b_3\}, R,$$

models this second step enabling generation of the set of terminal multisets by application of the scheme of this *MG* to its kernel:

$$\{1\cdot\boldsymbol{a}_1, 1\cdot\boldsymbol{a}_2, 4\cdot b_1, 3\cdot b_2, 7\cdot b_3\} \overset{r_1}{\Rightarrow}$$
$$\{1\cdot a_1, 1\cdot\boldsymbol{a}_2, 4\cdot b_1, 3\cdot b_2, 7\cdot b_3\} \overset{r_2}{\Rightarrow}$$
$$\{1\cdot a_1, 1\cdot a_2, 4\cdot b_1, 3\cdot b_2, 7\cdot b_3\} \overset{r_4}{\Rightarrow}$$
$$\{1\cdot a_2, 7\cdot b_1, 5\cdot b_2, 8\cdot b_3\} \overset{r_5}{\Rightarrow}$$
$$\{8\cdot b_1, 9\cdot b_2, 11\cdot b_3\} \overset{r_7}{\Rightarrow}$$
$$\{5\cdot b_1, 9\cdot b_2, 11\cdot b_3, 1\cdot c_1, 3\cdot c_2\} \overset{r_7}{\Rightarrow}$$
$$\{2\cdot b_1, 9\cdot b_2, 11\cdot b_3, 2\cdot c_1, 6\cdot c_2\} \overset{r_8}{\Rightarrow}$$
$$\{2\cdot b_1, 5\cdot b_2, 11\cdot b_3, 4\cdot c_1, 8\cdot c_2\} \overset{r_8}{\Rightarrow}$$
$$\{2\cdot b_1, 1\cdot b_2, 11\cdot b_3, 6\cdot c_1, 10\cdot c_2\} \overset{r_9}{\Rightarrow}$$
$$\{2\cdot b_1, 1\cdot b_2, 6\cdot b_3, 10\cdot c_1, 11\cdot c_2\} \overset{r_9}{\Rightarrow}$$
$$\{2\cdot b_1, 1\cdot b_2, 1\cdot b_3, 14\cdot c_1, 12\cdot c_2\} \overset{r_{10}}{\Rightarrow}$$

$$\{2\cdot b_1,1\cdot b_2,1\cdot b_3,6\cdot c_1,12\cdot c_2,1\cdot d_1,2\cdot d_2,3\cdot \boldsymbol{b}_1,2\cdot \boldsymbol{b}_2,4\cdot \boldsymbol{b}_3\}\overset{r_1}{\Rightarrow}$$
$$\{2\cdot b_1,1\cdot b_2,1\cdot b_3,6\cdot c_1,8\cdot c_2,3\cdot d_1,5\cdot d_2,4\cdot \boldsymbol{b}_1,3\cdot \boldsymbol{b}_2,7\cdot \boldsymbol{b}_3\}\overset{r_1}{\Rightarrow}$$
$$\{2\cdot b_1,1\cdot b_2,1\cdot b_3,6\cdot c_1,4\cdot c_2,5\cdot d_1,8\cdot d_2,5\cdot \boldsymbol{b}_1,4\cdot \boldsymbol{b}_2,10\cdot \boldsymbol{b}_3\}\overset{r_1}{\Rightarrow}$$
$$\{2\cdot b_1,1\cdot b_2,1\cdot b_3,6\cdot c_1,7\cdot d_1,6\cdot \boldsymbol{b}_1,5\cdot \boldsymbol{b}_2,13\cdot \boldsymbol{b}_3,1\cdot \boldsymbol{a}_5\},$$

so

$$v^2_{out}=\{\{0\cdot \boldsymbol{a}_4,1\cdot \boldsymbol{a}_5\}\},$$
$$v^2_{+}=\{6\cdot \boldsymbol{b}_1,5\cdot \boldsymbol{b}_2,13\cdot \boldsymbol{b}_3\}.$$

As seen, the output $\boldsymbol{a}_5$ of the considered recurrent *NN*, activated by inputs $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, will be activated as a response after the second step of its operation. The feedback of the second step will be $\{6\cdot \boldsymbol{b}_1, 5\cdot \boldsymbol{b}_2, 13\cdot \boldsymbol{b}_3\}$. All following steps are executed in the same manner.■

Now we shall consider key issues of multigrammatical modelling of *NNs* deep learning. A basic tool for this topic will be multiset metagrammars. Let us begin from feed-forward neural networks.

3. Multigrammatical modelling of feed-forward neural networks deep learning

We shall consider deep learning issues on the background of the above introduced representation of an *NN* as a couple G, H. Namely, instead of a weight we shall mark any connection by a unique variable, and a task will be, starting from a training data set (*TDS*) being a set of training samples *input*, *output*, to assign to these variables such weights which would satisfy this *TDS*, i.e. all training samples entering this set.

We shall represent a trained *FNN* by a scheme *R* of a multiset metagrammar including metarules

$$\{h_i\cdot a_i\}\to\{\gamma^i_1\cdot a^i_1,\dots,\gamma^i_{m_i}\cdot a^i_{m_i}\}. \tag{11}$$

where γ^i_j , i=1,...,m_i, is a variable which domain is $\lceil 0, N^i_j\rceil$, and N^i_j is a maximal possible weight of a connection from an output of a node a_i to an input of a node a^i_j (by this we assume that in a general case any connection may have its own maximal possible weight). In the below considerations we shall operate a total set $\{\gamma_1,...,\gamma_M\}$ of variables entering a scheme *R* and identified by lower indices ($M=|G|$ is a number of edges of a graph *G*, i.e. a number of weights of connections in a modelled *NN*). Until it will be considered a general case, we assume, that threshold values of nodes are constants h_i.

Let a *TDS* be $\boldsymbol{T}=\{\boldsymbol{A}^1_{in},\boldsymbol{A}^1_{out},\dots,\boldsymbol{A}^n_{in},\boldsymbol{A}^n_{out}\}$, where $\boldsymbol{A}^i_{in}$ is a set of input nodes activating a considered trained *FNN* whilst $\boldsymbol{A}^i_{out}$ is a set of its output nodes activated as a response of this network to an input $\boldsymbol{A}^i_{in}$. We shall put in compliance with a training sample $\boldsymbol{A}^i_{in},\boldsymbol{A}^i_{out}$ a multiset metagrammar

$$S_i(G,H)=1{*}\boldsymbol{A}^i_{in},R,F, \tag{12}$$

where

$$F=\left(\bigcup_{k=1}^{|G|}\{0\le\gamma_k\le N_k\}\right). \tag{13}$$

As everywhere above, a kernel of this *MMG* is a multiset containing multiobjects of the form $1\cdot a$ each such *MO* corresponding to one activated input of a considered trained *NN*. A filter of this *MMG* is a set of variables declarations $0\le\gamma_k\le N_k$ each defining a domain $[0,N_k]$ of a variable γ_k.

Evidently, a collection of weights of connections of this *NN* represented by a multiset $w=\{\overline{n_1}\cdot\gamma_1,\dots,\overline{n_M}\cdot\gamma_M\}\subset v$, where

$$v\in\overline{V}_{S_i(G,H)},$$

satisfies a training sample $\boldsymbol{A}^i_{in},\boldsymbol{A}^i_{out}$, if

$$\overline{V}_{S_i(G,H)}\Cap 1{*}\boldsymbol{A}_{out}=\{1{*}\boldsymbol{A}^i_{out}\}. \tag{14}$$

In this case a whole set W_i of multisets, representing collections of weights of connections of a considered trained *NN*, satisfying a training sample $\boldsymbol{A}^i_{in},\boldsymbol{A}^i_{out}$, is nothing but

$$W_i=\overline{V}_{S_i(G,H)}\Cap \boldsymbol{N}{*}\boldsymbol{\Gamma}. \tag{15}$$

Now the task is to select from this set all *TMSs* which satisfy not only this one training sample but all training data set $\boldsymbol{T}$. Evidently, a set $\boldsymbol{W_T}$ of collections of weights of connections of a trained *NN* satisfying ***all*** TDS, i.e. all training samples $\boldsymbol{A}^1_{in},\boldsymbol{A}^1_{out},\dots,\boldsymbol{A}^n_{in},\boldsymbol{A}^n_{out}$, would be a result of intersection of all sets $W_1,\dots, W_n$:

$$\boldsymbol{W_T}=\bigcap_{i=1}^{n}W_i \tag{16}$$

Let us note once more, that a multiset $\{\overline{n_1}\cdot\gamma_1,\dots,\overline{n_M}\cdot\gamma_M\}\in\boldsymbol{W_T}$ defines that the 1-th, …, the *M*-th connections of a trained *NN* would have weights $\overline{n_1},\dots,\overline{n_M}$ respectively. In a general case there may be $|\boldsymbol{W_T}|\ge 1$ collections of weights satisfying a *TDS*. At the same time it is possible that $W_T=\{\varnothing\}$, that means a *TDS* is contradictory regarding a considered *FNN*, and some additional technique would be developed and applied to such case.

Until now it was presumed that threshold values of nodes were constants h_i. However, generalization of this case on a priori unknown threshold values is quite simple. It is sufficient to represent a trained *NN* by a scheme *R* of a multiset metagrammar including metarules

$$\{\gamma^i_0\cdot a_i\}\to\{\gamma^i_1\cdot a^i_1,\dots,\gamma^i_{m_i}\cdot a^i_{m_i}\}. \tag{17}$$

where variables γ^i_j, i=1,...,m, have the same sense as in (11) whilst γ^i_0 is a variable which domain is $\lceil 0, N^i_0\rceil$, and N^i_0 is a maximal possible threshold value of an *i*-th node. All the rest considerations are the same as above.

Statement 3. A result of deep learning of any feed-forward neural network G, H by a training data set $\boldsymbol{T} = \{\boldsymbol{A}^1_{in}, \boldsymbol{A}^1_{out}, \ldots, \boldsymbol{A}^n_{in}, \boldsymbol{A}^n_{out}\}$ may be represented by a set $\boldsymbol{W_T}$ determined by (16).■

A proof of this statement is unnecessary due to the above detailed description of accumulation of a set $\boldsymbol{W_T}$.

All the said was associated with feed-forward neural networks. Let us consider now a general case of recurrent NNs.

4. Multigrammatical modelling of recurrent neural networks deep learning

We shall assume that a complete training data set $\boldsymbol{T} = \{\boldsymbol{A}^1_{in}, \boldsymbol{A}^1_{out}, \ldots, \boldsymbol{A}^n_{in}, \boldsymbol{A}^n_{out}\}$, where n is a number of steps of operation of a trained recurrent neural network, is applied to it in such a way that the i-th training sample is associated with the i-th step of operation of this *RNN*. In order to make a proposed technique of modelling deep learning of *RNNs* more natural, compact and understandable we shall use not analytic, as above, but algorithmic representation of its logic. The last will be represented by a function *RNNDL* (*R*ecurrent *N*eural *N*etwork *D*eep *L*earning) which inputs are a *TDS* $\boldsymbol{T}$, a set R of metarules representing a trained *RNN*, and a filter F containing declarations of multiplicities-variables having place in the aforementioned metarules. An output of this function is a set $\boldsymbol{W_T}$ of satisfying a *TDS* $\boldsymbol{T}$ multiset-represented collections of weights of connections of a trained *NN*, i.e. satisfying all training samples $\boldsymbol{A}^1_{in}, \boldsymbol{A}^1_{out}, \ldots, \boldsymbol{A}^n_{in}, \boldsymbol{A}^n_{out}$ entering this *TDS*.

We shall use in the text of the function *RNNDL* a filter $\boldsymbol{F}_i$ (in line 3 $\boldsymbol{F}_1$ as a particular case), which is defined as follows:

$$\boldsymbol{F}_i = F \cup \left(\bigcup_{a \in \boldsymbol{A}^i_{out}} \{a \geq 1\} \right) \cup \left(\bigcup_{a \in A - \boldsymbol{A}^i_{out}} \{a = 0\} \right) \qquad (18)$$

As may be seen, such filter enables selection of terminal multisets satisfying a set $\boldsymbol{A}^i_{out}$ of activated outputs determined by the i-th training sample (we use $a \geq 1$ not $a = 1$ because of the possibility of multiple activations of outputs, which was discussed above in the section 3).

Also we shall use following variables in the text of the function *RNNDL*:

- v which current value is a terminal multiset generated by an *MMG* $1{*}\boldsymbol{A}^1_{in}, R, \boldsymbol{F}_1$;
- $\boldsymbol{R}$ which current value is a set of rules created by substitution of values of variables, entering a set v, to metarules entering a set R;
- v_+ which current value is a feedback created as a result of the current step (in line 5 – from the first to the second step, and in line 9 – from the i-th to the $i+1$-th, where $i = 2, \ldots, n-1$);
- $\boldsymbol{v}$ which current value is a terminal multiset generated by a filtering multiset grammar $1{*}\boldsymbol{A}^i_{in} + v_+, \boldsymbol{R}, \boldsymbol{F}_i$.

The text of the function *RNNDL* is as follows.

```
1 RNNDL: function ($\boldsymbol{T}, R, F$) returns ($\boldsymbol{W_T}$);
2     $\boldsymbol{W_T} := \{\varnothing\}$;
3     do $v \in \bar{V}_{1*\boldsymbol{A}^1_{in}, R, \boldsymbol{F}_1}$;
4         $\boldsymbol{R} := R$¤$(v \cap \boldsymbol{N}{*}\boldsymbol{\Gamma})$;
5         $v_+ := v \cap \boldsymbol{N}{*}\boldsymbol{A}_+$;
6         do $i \in [2, n]$;
7             if $\exists \boldsymbol{v} \in \bar{V}_{1*\boldsymbol{A}^i_{in}+v_+, \boldsymbol{R}, \boldsymbol{F}_i}$;
8                 then $v_+ := v \cap \boldsymbol{N}{*}\boldsymbol{A}_+$;
9                 else goto $v$#;
10        end $i$;
11        $\boldsymbol{W_T} : \cup \{v \cap \boldsymbol{N}{*}\boldsymbol{\Gamma}\}$;
12    $v$#:;
13    end $v$;
14    return ($\boldsymbol{W_T}$);
15    end RNNDL
```

Let us comment this text in short.

At the very beginning (line 2) the variable $\boldsymbol{W_T}$ receives the initial value $\{\varnothing\}$. Accumulation of a multiset-represented set of collections of weights of connections of a trained *NN* is done inside the loop (lines $3-17$) on the variable v which values, as it was mentioned above, are terminal multisets generated by application of a multiset metagrammar $1{*}\boldsymbol{A}^1_{in}, R, \boldsymbol{F}_1$. Its kernel $1{*}\boldsymbol{A}^1_{in}$ is a multiset representation of a set of inputs activating a trained *RNN* at the first step of training; its scheme R includes metarules like (17) representing this *RNN*; and its filter $\boldsymbol{F}_1$ determines terminal multisets, satisfying a set of activated outputs $\boldsymbol{A}^1_{out}$ belonging to the first training sample. According to (14), every current value of the variable v includes a multiset $\{\overline{n_1} \cdot \gamma_1, \ldots, \overline{n_M} \cdot \gamma_M\}$ representing a collection of weights of *RNN* connections satisfying a filter $\boldsymbol{F}_1$. (Evidently, in a general case a number of *TMSs* entering a set $\bar{V}_{1*\boldsymbol{A}^1_{in}, R, \boldsymbol{F}_1}$ may be greater than 1). Because the aforementioned collection of weights must satisfy ***all*** training samples, a sequence of operations is executed inside the inner loop (lines $6-10$) on the variable i, which values are numbers of training steps (from the second to the n-th). These operations enable check whether a current collection of weights, induced by a current value of the variable v, satisfies the current i-th training sample. A collection is included to the set $\boldsymbol{W_T}$ (line 11) only in the case when results of all executed checks are successful. Otherwise the external loop on values of the variable v continues (this is done by the jump operator **goto** v#, line 9); thus, a current "unsuccessful" collection of weights is not included to $\boldsymbol{W_T}$.

It would be useful some additional clarifying comments to this general description.

As it was already said, a set of ***rules*** denoted $\boldsymbol{R}$ is created by substitution of values of variables entering a set v to ***metarules*** entering a set R (line 4). This operation is denoted ¤ and is defined as follows:

$$R \text{¤} \left\{ \overline{n_1} \cdot \overline{\gamma_1}, \dots, \overline{n_M} \cdot \overline{\gamma_M} \right\} = R \circ \overline{n_1}, \dots, \overline{n_M} \,, \tag{19}$$

where operation $\circ$ is defined by (A.27), and, evidently, if

$$\left\{ \overline{n_1} \cdot \overline{\gamma_1}, \dots, \overline{n_M} \cdot \overline{\gamma_M} \right\} \subset v \,, \tag{20}$$

then

$$\left\{ \overline{n_1} \cdot \overline{\gamma_1}, \dots, \overline{n_M} \cdot \overline{\gamma_M} \right\} = v \cap \boldsymbol{N}^* \boldsymbol{\Gamma} \,. \tag{21}$$

Since a multiset v includes a multiset v_+ being a feedback, the next operator (line 5) enables selection of this submultiset by intersection of v and $\boldsymbol{N}^*\boldsymbol{A}_+$, where, remind, $\boldsymbol{A}_+$ is a set of feedback objects.

Every step of a loop on training samples (lines 6 – 10) from the second to the n-th includes the only conditional (**if-then-else**) operator. If a set $\overline{V}_{1*\boldsymbol{A}^i_{in}+v_+,\boldsymbol{R},\boldsymbol{F}_i}$ is non-empty (this means that a generated terminal multiset $\boldsymbol{v}$ includes a set of multiobject representations of activated outputs coinciding a set of outputs of a training sample $\boldsymbol{A}^i_{out}$) then a new feedback $\boldsymbol{v}_+$ to be used at the next step of the loop is created (line 8) by selecting from the *MS* $\boldsymbol{v}$ multiobjects containing feedback objects (this is done by intersection of *MSs* $\boldsymbol{v}$ and $\boldsymbol{N}^*\boldsymbol{A}_+$). Otherwise (i.e. no one *TMS* is generated because a set of multiobject representations of activated outputs summarized with a feedback v_+ do not respond a set of outputs of a training sample $\boldsymbol{A}^i_{out}$) a loop on numbers of training steps for the current multiset v is broken by the **goto** $v\#$ operator, and this jump enables continuation of the loop on the values of the variable v. If the loop on numbers of training steps (or, just the same, training samples) is successfully finished, this means that the current set of variables values (weights of connections of the trained *RNN*) satisfies the *TDS* and thus would be included to the resulting set $\boldsymbol{W_T}$ (line 12). After finishing the loop on v a final value of this variable is returned (line 14). Evidently, if no one successful execution of the loop on numbers of training steps had place, then $\boldsymbol{W_T}$ would remain the empty set.

Statement 4. A result of deep learning of any recurrent neural network G, H by a training data set $\boldsymbol{T} = \left\{ \boldsymbol{A}^1_{in}, \boldsymbol{A}^1_{out}, \dots, \boldsymbol{A}^n_{in}, \boldsymbol{A}^n_{out} \right\}$ may be represented by a set $\boldsymbol{W_T}$ created by application of the function *RNNDL*.■

A proof of this statement is redundant due to the above detailed description of the function *RNNDL*.

So the case of *RNNs* deep learning is also covered without principal difficulties by the proper application of the multigrammatical framework.

5. Discussion

There are at least three possible directions of the *MGF* application to modelling not only artificial but **biological** neural networks (*BNN*s) with capturing some new features. The **first** one may cover physical growth of a brain from a baby to an adult and concomitant increasing of cognitive abilities of a human being which may be modelled by application of self-generating multiset grammars [3]. The **second** is based on a possibility of combining in *MG* rules ordinary for artificial *NNs* topological-weights information with information on resources needed for brain operation and circulating inside a brain. The **third** is rather close to the second and concerns studying resilience of *BNNs* to possible destructive impacts, blocking some parts of a brain or/and depriving it some of necessary resources. Techniques developed for this task regarding resilience of sociotechnological systems [3,38] looks quite relevant for the beginning of this research.

Investigation of the aforementioned opportunities as well as further expansion of techniques of *NNs* multigrammatical modelling on a greater number of classes of neural networks will be a subject of future publications on this topic. No doubt, the three mentioned directions would be extremely useful for the development and enhancement of the *MGF* itself. And, naturally, the most advancing seem attempts of *NN*-based implementation of the *MGF*.

The author is grateful to Acad. Victor Soifer and Prof. Fred Roberts for a long-term support, as well as to the reviewer whose useful advices have definitely contributed to the quality of the final version of the presented paper.

Acronyms

AI artificial intelligence
ANN artificial neural network
APS augmented Post system
BC boundary condition
BNN biological neural network
CBC chain boundary condition
IIR infinite impulse response
FIR finite impulse response
FMG filtering multiset grammar
FNN feed-forward neural network
MG multiset grammar
MGF multigrammatical framework
MMG multiset metagrammar
MO multiobject
MS multiset
MV multiplicity-variable
NN neural network
RNN recurrent neural network
RNNDL recurrent neural network deep learning
STMS set of terminal multisets
TDS training data set
TMS terminal multiset

References

[1] Sheremet IA. Recursive multisets and their applications [In Russian]. Moscow: "Nauka" Publisher; 2010.
[2] Sheremet I.A. Recursive multisets and their applications. European Academy of Natural Sciences; 2011. ISBN: 9783942944120.
[3] Sheremet IA. Multigrammatical framework for knowledge-based digital economy. Cham: Springer Nature Switzerland AG; 2022. ISBN: 978-3-031-13857-7.
[4] Petrovskiy AB. Theory of measured sets and multisets [In Russian]. Moscow: "Nauka" Publisher; 2018.

[5] Ben-Ari M. Mathematical logic for computer science. London: Springer-Verlag; 2012.
[6] Kowalski R.A. Algorithm = Logic + Control. Comm ACM 1979; 22(7): 424-436.
[7] Lee KD. Foundations of programming languages. 2nd ed. Cham: Springer International Publishing AG; 2017.
[8] Davis MD, Sigal R, Weyuker EJ. Computability, complexity, and languages: Fundamentals of theoretical computer science. 2nd ed. Boston: Academic Press; 1994.
[9] Chomsky N. Syntactic Structures. The Hague: Mouton de Gruyter; 2005.
[10] Post EL. Formal reductions of the general combinatorial problem. Am J Math 1943; 65: 197-215.
[11] Sheremet IA. Intelligent software environments for information processing systems [In Russian]. Moscow: "Nauka" Publisher; 1994.
[12] Sheremet IA. Augmented post systems: The mathematical framework for data and knowledge engineering in network-centric environment. Berlin: EANS; 2013.
[13] Sheremet I. Augmented post systems: Syntax, semantics, and applications. In Book: Sud K, Erdogmus P, Kadry S, eds. Introduction to data science and machine learning. Ch 11. London: IntechOpen; 2020. DOI: 10.5772/intechopen.86207.
[14] Hopcroft JE, Motwani R, Ullman JD. Introduction to automata theory, languages, and computation. 2nd ed. Boston: Addison–Wesley; 2001.
[15] David R, Alla H. Discrete, continuous, and hybrid Petri Nets. Berlin, Heidelberg: Springer-Verlag; 2010.
[16] Ullman JD. Computational aspects of VLSI. New York, NY: W H Freeman & Co; 1984.
[17] Schiff JL. Cellular automata: A discrete view of the world. Hoboken, NJ: John Wiley & Sons Inc; 2011.
[18] McCulloch W, Pitts W. A logical calculus of the ideas immanent in nervous activity. Bull Math Biol 1943; 7: 115-133.
[19] Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. Psychol Rev 1958; 65(6): 386.
[20] Siegelmann HT, Sontag ED. Analog computation via neural networks. Theor Comput Sci 1994; 131(2): 331-360.
[21] Bragin AD, Spitsyn VG. Motor imagery recognition in electroencephalograms using convolutional neural networks. Computer Optics 2020; 44(3): 482-487. DOI: 10.18287/2412-6179-CO-669.
[22] Kalinina MO, Nikolaev PL. Book spine recognition with the use of deep neural networks. Computer Optics 2020; 44(6): 968-977. DOI: 10.18287/2412-6179-CO-731.
[23] Firsov N, Podlipnov V, Ivliev N, Nikolaev P, Mashkov S, Ishkin P, Skidanov R, Nikonorov A. Neural network-aided classification of hyperspectral vegetation images with a training sample generated using an adaptive vegetation index. Computer Optics 2021; 45(6): 887-896. DOI: 10.18287/2412-6179-CO-1038.
[24] Acemoglu D. Redesigning AI. MIT Press; 2021.
[25] Andriyanov NA, Dementiev VE, Tashlinskiy AG. Detection of objects in the images: from likelihood relationships towards scalable and efficient neural networks. Computer Optics 2022; 46(1): 139-159. DOI: 10.18287/2412-6179-CO-922.
[26] Arlazarov VV, Andreeva EI, Bulatov KB, Nikolaev DP, Petrova OO, Savelev BI, Slavin OA. Document image analysis and recognition: a survey. Computer Optics 2022; 46(4): 567-589. DOI: 10.18287/2412-6179-CO-1020.
[27] Agafonova YD, Gaidel AV, Zelter PM, Kapishnikov AV, Kuznetsov AV, Surovtsev EN, Nikonorov AV. Joint analysis of radiological reports and CT images for automatic validation of pathological brain conditions. Computer Optics 2023; 47(1): 152-159. DOI: 10.18287/2412- 6179-CO-1201.
[28] Sallans B, Hinton G. Reinforcement learning with factored states and actions. J Mach Learn Res 2004; 5: 1063-1088.
[29] Salakhutdinov R, Hinton G. Semantic hashing. Int J Approx Reasoning 2009; 50(7): 969-978.
[30] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015; 521: 436-444.
[31] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016.
[32] Nielsen M. Neural networks and deep learning. Determination Press; 2019.
[33] Aggarwal CC. Neural networks and deep learning. Cham: Springer International Publishing AG; 2018.
[34] Bommasani R, Hudson DA, Adeli E, et al. On the opportunities and risks of foundation models. arXiv Preprint. 2022. Source: <https://arxiv.org/abs/2108.07258>.
[35] Metz C. "The Godfather of A.I." leaves google and warns of danger ahead. The New York Times 2023, 1 May.
[36] Graves A, Wayne G, Danihelka I. Neural turing machines. arXiv Preprint. 2014. Source: <https://arxiv.org/abs/1410.5401>.
[37] Ackerman J, Cybenko G. A survey of neural networks and formal languages. arXiv Preprint. 2020. Source: <https://arxiv.org/abs/2006.01338>.
[38] Sheremet I. Application of the multigrammatical framework to the assessment of resilience and recoverability of large-scale industrial systems. In Book: Roberts FS, Sheremet IA, eds. Resilience in the digital age. Ch 2. Cham: Springer Nature Switzerland AG; 2021: 16-34. DOI: 10.1007/978-3-030-70370-7_2.

Appendix A. Filtering multiset grammars and multiset metagrammars

A ***multiset*** (*MS*) is defined as a collection of ***multiobjects*** (*MOs*) consisting of indistinguishable elements (***objects***) [3,4]. A ***multiobject*** containing n objects a is denoted $n \cdot a$, and n is called a ***multiplicity*** of an object a. (Below we shall use small symbols "a", "b", "c" etc. with or without lower indices for objects denotation; multisets will be denoted by small "*v*" with or without indices, as well).

A record

$$v = \{n_1 \cdot a_1, \ldots, n_m \cdot a_m\} \tag{A.1}$$

means that an *MS* v contains n_1 objects a_1, …, n_m objects a_m. We shall use a symbol "$\in$" for denotation that an object or an multiobject enters a multiset v, so $a_i \in v$ means that an object a_i enters an *MS* v as well as $n_i \cdot a_i \in v$ means that a multiobject $n_i \cdot a_i$ enters this *MS*. From the substantial point of view a set $\{a_1,\ldots,a_m\}$ and a multiset $\{1 \cdot a_1,\ldots,1 \cdot a_m\}$ are equivalent, as well as an object a and a multiobject $1 \cdot a$. The empty multiset and the empty set both are denoted as $\{\varnothing\}$.

If a multiplicity of an object a is zero, it is equivalent to an absence of a in a multiset v, what is written, as usual, a v. If an *MS* v is determined by (A.1), then a set $\beta(v)=\{a1,\dots,a_m\}$ is called a ***basis*** of a multiset v.

Here in this paper we shall use two known relations (inclusion $\subseteq$ and strict inclusion $\subset$) and five known operations on multisets [3] – addition +, subtraction –, multiplication by a constant *, join $\cup$ and intersection $\cap$ – as well as two new operations, which will be applied below to multigrammatical modelling of neural networks.

Operation of ***multiset creation*** from a set is similar to the operation of multiplication of a multiset by a constant, but one of its operands is a set not multiset, so we shall denote this operation by the same symbol *. It would be clear what namely operation is applied from what namely operand is used – a set or a multiset. (Such way of operations denotation is common for modern programming languages). Semantics of multisets creation is as follows:

$$\{a_1,\dots,a_m\}*n=n*\{a_1,\dots,a_m\}=\{n_1\cdot a_1,\dots,n_m\cdot a_m\}. \tag{A.2}$$

Evidently, for any set A and any integer number n it is true $\beta(n*A)=A$.

Example 1.

$$\{a,b,c\}*3=3*\{a,b,c\}=\{3\cdot a,3\cdot b,3\cdot c\}.$$ ■

Operation of ***intersection*** of a set of multisets ***by a set*** is denoted and is defined as follows:

$$\{v_1,\dots,v_n\}\Cap v=\{v_1\cap v,\dots,v_n\cap v\}. \tag{A.3}$$

This operation is very convenient for selecting sub-multisets from multisets. Due to definition of multisets intersection [3] as

$$v\cap v'=\bigcup_{\substack{n\cdot a\in v\\ n'\cdot a\in v'\\ a\in\beta(v)\cap\beta(v')}}\{\min\{n,n'\}\cdot a\} \tag{A.4}$$

it is quite simple to select objects, entering a set x, from a multiset v by applying

$$v\cap(1*x), \tag{A.5}$$

or, taking into account priority of operations (multiplication has a greater priority than intersection), without brackets

$$v\cap(1*x). \tag{A.6}$$

In this case, due to $\min\{n,1\}=n$ for any non-zero integer multiplicity n, a result will be $\{1\cdot a_1,\dots,1\cdot a_m\}$, where

$$\{a_1,\dots,a_m\}=\beta(v)\cap\beta(x). \tag{A.7}$$

From the other side, not more difficult is to select multiobjects, which objects enter some predefined set, along with their multiplicities. For this purpose a constant $\boldsymbol{N}$ representing some maximal value, which is greater than any multiplicity in a concrete implementation of the multiset algebra, may be used. In this case

$$v\cap\boldsymbol{N}*x \tag{A.8}$$

would be applied, and due to $\min\{n,\boldsymbol{N}\}=n$ for any multiplicity n, a result will be $\{n_1\cdot a_1,\dots,n_m\cdot a_m\}$, where $x=\{a_1,\dots,a_m\}$.

Example 2. To check whether objects a and b enter multisets belonging to a set

$$V=\{\{2\cdot a,3\cdot c\},\{4\cdot b,8\cdot c\}\},$$

operation $V\Cap 1*\{a,b\}$ may be used, and the result will be $\{\{1\cdot a\},\{1\cdot b\}\}$.

To select from this set of multisets their submultisets, which basis is $\{a,c\}$, it is sufficient to apply operation $v\Cap N*\{a,c\}$, obtaining $\{\{2\cdot a\},\{3\cdot c\}\}$. ■

A ***multiset grammar*** is a couple $S=v_0,R$, where a multiset v_0 is named a ***kernel***, and a finite set of ***rules*** R is named a ***scheme***. A set of all objects used in a kernel and a scheme of an *MG S* is denoted A_S.

A rule $r\in R$ is a construction

$$v\to v', \tag{A.9}$$

where multisets v and v' are named respectively the ***left part*** ($v\neq\{\varnothing\}$) and the ***right part*** of a rule r, and "$\to$" is a divider.

If $v\subseteq\overline{v}$, then a result of application of a rule r to a multiset $\overline{v}$ is a multiset

$$\overline{v}' = \overline{v} - v + v' \tag{A.10}$$

Speaking informally, (10) defines, that if the left part of a rule, i.e. a multiset *v*, is included to an *MS* $\overline{v}$, then *v* is replaced by the right part of this rule, i.e. a multiset *v'*. In this case a rule *r* is called ***relevant to a multiset*** $\overline{v}$. A result of application of a rule *r* to a multiset $\overline{v}$ is denoted as

$$\overline{v} \overset{r}{\Rightarrow} \overline{v}', \tag{A.11}$$

and it is said, that an *MS* $\overline{v}'$ is ***generated*** from an *MS* $\overline{v}$ by application of a rule *r*.

A set of multisets**, generated by application of a multigrammar S*= v_0, R, or, just the same, ***determined by this multigrammar, is recursively created as follows:

$$V_{(0)} = \{v_0\}, \tag{A.12}$$

$$V_{(i+1)} = V_{(i)} \cup \left(\bigcup_{\overline{v} \in V_{(i)}} \bigcup_{r \in R} \left\{ \overline{v}' \,\middle|\, \overline{v} \overset{r}{\Rightarrow} \overline{v}' \right\} \right), \tag{A.13}$$

$$V_S = V_{(\infty)}. \tag{A.14}$$

As seen, a set V_S includes all multisets, which may be generated from an *MS* v_0 by the sequential application of rules $r \in R$, and V_S is a fixed point of a sequence

$$V_{(0)}, V_{(1)}, \ldots, V_{(i)}, \ldots,$$

so

$$V_S = \bigcup_{i=0}^{\infty} V_{(i)}. \tag{A.15}$$

In a general case a set V_S may be infinite.

If an *MS* $\overline{v}'$ may be generated from an *MS* $\overline{v}$ by application of some sequence (chain) of rules entering a scheme *R*, it is denoted as

$$\overline{v} \overset{R}{\Rightarrow} \overline{v}', \tag{A.16}$$

and, if so, then

$$V_S = \left\{ \overline{v} \,\middle|\, v_0 \overset{R}{\Rightarrow} \overline{v} \right\}. \tag{A.17}$$

A multiset $\overline{v} \in V_S$ is called a ***terminal multiset*** (*TMS*), if no one rule $r \in R$ is relevant to this multiset, i.e.

$$(\forall (v \to v') \in R)\ \neg (v \subseteq \overline{v}). \tag{A.18}$$

A set of terminal multisets (*STMS*), determined by a multiset grammar *S*, is denoted $\overline{V_S}$. Any *STMS* is a subset of a set of all multisets defined by an *MG S*:

$$\overline{V_S} \subseteq V_S. \tag{A.19}$$

Example 3. Let the *MG S*= v_0, R, where the kernel is

$$v_0 = \{3 \cdot (rur), 4 \cdot (usd), 2 \cdot (eur)\},$$

and the scheme $R = \{r_1, r_2\}$, where, in turn, the rule r_1 is

$$\{3 \cdot (eur)\} \to \{4 \cdot (usd)\},$$

and the rule r_2 is

$$\{2 \cdot (usd), 3 \cdot (rur)\} \to \{2 \cdot (eur)\}.$$

As seen, the initial collection of currencies includes 3 Russian Rubles, 4 US Dollars, and 2 Euros; the scheme of this *MG* represents actual regulations for currencies exchange (3 Euros may be exchanged to 4 US Dollars, 2 US Dollars and 3 Russian Rubles may be exchanged to 2 Euros).

In accordance with definitions (1) – (9),

$$V_{(0)}=\left\{\left\{2\cdot(eur),4\cdot(usd),3\cdot(rur)\right\}\right\},$$

$$V_{(1)}=V_{(0)}\cdot\left\{\left\{2\cdot(usd),4\cdot(eur)\right\}\right\},$$

$$V_{(2)}=V_{(1)}\cdot\left\{\left\{6\cdot(usd),1\cdot(eur)\right\}\right\}=V_S.$$

As seen, this *MG* enables generation of all possible collections of Euros, US dollars, and Russian rubles, which may be obtained from the initial collection v_0 by sequential currency exchanges, which parameters are fixed by regulations represented by the rules r_1 and r_2. A set of terminal multisets $\overline{V_S}$ generated by this MG contains the only *TMS* {1·(*eur*), 6·(*usd*)}.■

Filtering multiset grammars (*FMGs*) are such extension of *MGs*, which semantics presumes two sequential operations – generation of a set of terminal multisets by application of a scheme to a kernel (step 1), and selection from it such *TMSs*, that satisfy some restrictions (conditions), expressed by a so called filter (step 2).

A ***filter*** is a set of conditions, and a multiset satisfies a filter if it satisfies all conditions entering it. Conditions may be boundary and optimizing. Here in this paper we shall use only boundary conditions (*BCs*).

A ***boundary condition*** is recorded as $a\theta n$ or $n\theta a$, where $\theta\in\{>,<,\geq,\leq,=\}$. A multiset v satisfies a *BC* $a\theta n$, if $m\cdot a\in v$ and $m\theta n$ is true, and satisfies a *BC* $n\theta a$, if $m\cdot a\in v$ and $n\theta m$ is true (in both cases $a\notin v$ is equivalent to $0\cdot a\in v$). In a general case so called ***chain boundary conditions*** (*CBCs*) recorded as $n\theta a\theta' n'$ may be applied, any of such *BCs* being equivalent to two boundary conditions: $n\theta a$ and $a\theta' n'$. A result of application of a filter F to a set of multisets V is denoted $V\downarrow F$.

Example 2. Let us consider the set of multisets

$$V=\left\{\begin{array}{c}\left\{1\cdot(eur),6\cdot(usd)\right\},\\ \left\{4\cdot(usd),3\cdot(rur)\right\},\\ \left\{7\cdot(eur),1\cdot(usd),5\cdot(rur)\right\}\end{array}\right\},$$

and the filter

$$F=\left\{(usd)<5,(rur)\geq 3\right\}.$$

Then

$$V\downarrow F=\left(V\downarrow\left\{(usd)<5\right\}\right)\cap\left(V\downarrow\left\{(rur)\geq 3\right\}\right)=\left\{\left\{4\cdot(usd),3\cdot(rur)\right\},\left\{7\cdot(eur),1\cdot(usd),5\cdot(rur)\right\}\right\}.\qquad■$$

A ***filtering multiset grammar*** is a triple $S=v_0,R,F$, where v_0 and R are, as above, a kernel and a scheme, while F is a filter, including conditions, defining multisets, which would be selected from a set of *TMSs*, generated by an *MG* v_0,R, i.e.

$$\overline{V}_s=\overline{V}_{v_0,R}\downarrow F. \qquad (A.20)$$

Verbally, $\overline{V}_s$ is a subset of $\overline{V}_{v_0,R}$, which includes only such elements of this set, which satisfy a filter F.

Example 3. Let us consider the *FMG* $S=v_o,R,F$, where the kernel v_o and the scheme R are the same as in the Example 1, and the filter

$$F=\left\{(eur)\geq 2,(usd)\geq 3\right\}.$$

Applying this filter to the set of terminal multisets $\overline{V}_{v_0,R}$ generated by the *MG* v_0,R, one solves a task of selecting such collections, obtained by proper exchange chains, which would contain not less than 2 Euros and not less than 3 US Dollars.

According to (20),

$$V_s=V_{v_0,R}\downarrow F=\{\varnothing\},$$

that means no one collection satisfies the filter F. ■

A ***multiset metagrammar*** S is a triple $\langle v_0,R,F\rangle$, where v_0 and F are, as above, a kernel and a filter respectively, and a scheme R contains, along with rules, also metarules.

A ***metarule*** has the same form as a rule

$$\{\mu_1\cdot a_1,\dots,\mu_m\cdot a_m\}\to\{\mu_1{}'\cdot a_1{}',\dots,\mu_n{}'\cdot a_n{}'\},\tag{A.21}$$

but any μ_i, μ_j' may be not only a positive integer number, as in *MGs*, but also a ***variable*** $\gamma\in\boldsymbol{\Gamma}$, where $\boldsymbol{\Gamma}$ is an universum of variables. If μ_i or μ_j' is a variable, then it is called a ***multiplicity-variable*** (*MV*). As seen, a rule is a partial case of a metarule, which all multiplicities $\mu_1,\dots,\mu_m,\mu_1',\dots,\mu_n{}'$ are constants.

A filter F of an *MMG* $S=\langle v_0,R,F\rangle$ is a set of boundary conditions, as in *FMGs*. At the same time F includes chain boundary conditions of a form

$$n\le\gamma\le n'.\tag{A.22}$$

There is one and only one *CBC* (A.22) for every variable γ having place in at least one metarule entering a scheme R. This *CBC* is called a ***variable declaration*** and determines a set of possible values (domain) of this variable.

If F includes a subfilter

$$F_{\Gamma}=\{n_1\le\gamma_1\le n_1',\ \dots,\ n_l\le\gamma_l\le n_l'\}\tag{A.23}$$

containing all variables declarations, then every tuple $\overline{n_1},\dots,\overline{n_l}$, such that $\overline{n_1}\in[n_1,\ n_1'],\dots,\overline{n_l}\in[n_l,n_l']$, enables creation of one filtering multigrammar by substitution of $n_1,\dots,n_l$ to all metarules, entering a scheme R, instead of multiplicities-variables. Rules, entering R, are transferred to a new scheme denoted

$$R\circ\overline{n_1},\dots,\overline{n_l}\tag{A.24}$$

without any transformations. Every such *FMG* generates a set of terminal multisets by application of a filter $\mathcal{F}=F-F_{\Gamma}$ $\mathcal{F}=F-F_{\Gamma}$, which contains all "usual" conditions, which do not include variables.

Finally, an *MMG* $S=v_0,R,F$ determines a set of terminal multisets $\overline{V_s}$ in such a way:

$$\overline{V_s}=\left(\bigcup_{\overline{s}\in S^*}\overline{V_{\overline{s}}}\right)\downarrow\mathcal{F},\tag{A.25}$$

$$S^*=\bigcup_{\overline{n_1}\in[n_1,n_1']}\cdots\bigcup_{\overline{n_l}\in[n_l,n_l']}\left\{\begin{matrix}v_0+\{\overline{n_1}\cdot\overline{\gamma_1},\dots,\overline{n_l}\cdot\overline{\gamma_l}\},\\ R\circ\overline{n_1},\dots,\overline{n_l}\end{matrix}\right\},\tag{A.26}$$

$$R\circ\overline{n_1},\dots,\overline{n_l}=\{r\circ\overline{n_1},\dots,\overline{n_l}\,|\,r\in R\},\tag{A.27}$$

$$\mathcal{F}=F-F_{\Gamma},\tag{A.28}$$

$$F_{\Gamma}=\bigcup_{j=1}^{l}\{n_j\le\gamma_j\le n_j'\},\tag{A.29}$$

where $\overline{\gamma_1},\dots,\overline{\gamma_l}$ are auxiliary objects, used in such a way, that a multiobject $\overline{n_j}\cdot\overline{\gamma_j}$, $j=1,\dots,l$, represents a value $\overline{n_j}$ of a variable γ_j. As seen, due to (A.25) – (A.29) an *MMG* $S=v_0,R,F$ generates terminal multisets of a form

$$\{n_{i_1}\cdot a_{i_1},\dots,n_{i_k}\cdot a_{i_k},\overline{n_{j_1}}\cdot\overline{\gamma_1},\dots,\overline{n_{j_l}}\cdot\overline{\gamma_l}\},\tag{A.30}$$

and this feature is useful for multigrammatical modelling of deep learning neural networks.

As seen, *MG*s are a multiset-operating analogue of classic string-operating grammars introduced by N. Chomsky [9] whilst *MMGs* due to use of variables are some analogue of Horn clauses operating atomic formulas of the first-order predicate logic [6, 7], as well as Post systems [10] and their generalization – augmented Post systems (*APSs*) [11, 12, 13] – operating strings. A principal feature distinguishing *MMGs* from Horn clauses, Post systems and *APSs* is that in the last three an area of actuality of any variable is a specific clause or production (S-production) to which this variable belongs, whilst in *MMGs* this area is an entire scheme (a set of rules).

Let us illustrate the definition (A.25) – (A.29) by the following example.

Example 4. Consider the *MMG* $S=v_0,R,F$, where $v_0=\{1\cdot a\}$, and R contains three metarules, including multiplicities-variables x and y:

$$r_1:\ \{1\cdot a\}\to\{2\cdot b,x\cdot c\},$$

$$r_2:\ \{1\cdot b,1\cdot c\}\to\{y\cdot d\},$$

$$r_3:\ \{1\cdot b,y\cdot c\}\to\{y\cdot d\},$$

whilst the filter F contains the following conditions:

$c < 2,$

$d \geq 1,$

$1 \leq x \leq 2,$

$0 \leq y \leq 1,$

the last two *CBCs* being the declarations of the variables x and y.

According to (A.25) – (A.29), the scheme S determines four *FMGs*:

$$S_1 = \{1 \cdot a, 1 \cdot \overline{x}\}, R \circ 1, 0,$$

$$S_2 = \{1 \cdot a, 1 \cdot \overline{x}, 1 \cdot \overline{y}\}, R \circ 1, 1,$$

$$S_3 = \{1 \cdot a, 2 \cdot \overline{x}\}, R \circ 2, 0,$$

$$S_4 = \{1 \cdot a, 2 \cdot \overline{x}, 1 \cdot \overline{y}\}, R \circ 2, 1,$$

and the filter $\mathcal{F} = \{c < 2, d \geq 1\}$.

As seen,

$$R_1 = R \circ 1, 0 = \{\{1 \cdot a\} \to \{2 \cdot b, 1 \cdot c\}, \{1 \cdot b, 1 \cdot c\} \to \{\varnothing\},\ \{1 \cdot b\} \to \{\varnothing\}\},$$

$$\overline{V}_{S_1} = \{\{1 \cdot \overline{x}\}, \{1 \cdot c, 1 \cdot \overline{x}\}\},$$

$$R_2 = R \circ 1, 1 = \{\{1 \cdot a\} \to \{2 \cdot b, 1 \cdot c\}, \langle 1 \cdot b, 1 \cdot c\} \to \{1 \cdot d\}\},$$

$$\overline{V}_{S_2} = \{\{1 \cdot b, 1 \cdot \overline{x}, 1 \cdot \overline{y}\}, \{1 \cdot d, 1 \cdot \overline{x}, 1 \cdot \overline{y}\}\},$$

$$R_3 = R \circ 2, 1 = \{\{1 \cdot a\} \to \{2 \cdot b, 2 \cdot c\}, \{1 \cdot b, 1 \cdot c\} \to \{\varnothing\},\ \{1 \cdot b\} \to \{\varnothing\}\},$$

$$\overline{V}_{S_3} = \{\{2 \cdot \overline{x}\}, \{1 \cdot c, 2 \cdot \overline{x}\}, \{2 \cdot c, 2 \cdot \overline{x}\}\},$$

$$R_4 = R \circ 2, 1 = \{\{1 \cdot a\} \to \{2 \cdot b, 2 \cdot c\}, \{1 \cdot b, 1 \cdot c\} \to \{1 \cdot d\}\},$$

$$\overline{V}_{S_4} = \{\{2 \cdot d, 2 \cdot \overline{x}, 1 \cdot \overline{y}\}\}.$$

According to (A.25),

$$\overline{V}_S = \left(\overline{V}_{S_1} \cup \overline{V}_{S_2} \cup \overline{V}_{S_3} \cup \overline{V}_{S_4}\right) \downarrow \{c < 2, d \geq 1\} = \left\{ \begin{array}{c} \{1 \cdot \overline{x}\}, \{1 \cdot c, 1 \cdot \overline{x}\}, \{1 \cdot b, 1 \cdot \overline{x}, 1 \cdot \overline{y}\}, \\ \{1 \cdot d, 1 \cdot \overline{x}, 1 \cdot \overline{y}\}, \{2 \cdot \overline{x}\}, \{1 \cdot c, 2 \cdot \overline{x}\}, \\ \{2 \cdot c, 2 \cdot \overline{x}\}, \{2 \cdot d, 2 \cdot \overline{x}, 1 \cdot \overline{y}\} \end{array} \right\} \blacksquare$$

$$\downarrow \{c < 2, d \geq 1\} = \{\{1 \cdot d, 1 \cdot \overline{x}, 1 \cdot \overline{y}\}, \{2 \cdot d, 2 \cdot \overline{x}, 1 \cdot \overline{y}\}\}.$$

Authors' information

Igor A. Sheremet. Geophysical Center of Russian Academy of Sciences, Moscow, Russia. E-mail: sheremet@rfbr.ru

Received October 04, 2023. The final version – November 01, 2023.