Chapter

# Unitary Multiset Grammars an Metagrammars Algorithmics and Application

*Igor Sheremet*

## Abstract

The chapter is dedicated to the algorithmics of unitary multiset grammars and metagrammars. Their application to some actual problems from the area of large-scale sociotechnical systems (STS) assessment and optimization is also considered: estimation of capabilities of the producing STS; amounts of resources, necessary to such STS for various orders completion; assessment of STS sustainability/vulnerability to various destructive impacts (natural disasters, technogenic catastrophes, mutual sanctions, etc.); and STS profit maximization, as well as works optimal distribution among non-antagonistic competing STS, operating in the market economy.

**Keywords:** systems analysis, operations research, knowledge engineering, digital economy, multisets recursive multisets, multiset grammars, unitary multiset grammars and multimetagrammars, sociotechnical systems assessment and optimization

## 1. Introduction

Unitary multiset grammars (UMG) and multimetagrammars (UMMG) are knowledge representation model, providing convergence of classical operations research and modern knowledge engineering. The main area of UMG/UMMG application is assessment and optimization of large-scale sociotechnical systems (STS). Syntax and semantics of multigrammars are described in the first part of this work, being separate chapter of this book. Section 2 of this chapter contains primary description of UMG/UMMG-improved algorithmics, providing generation of terminal multisets (TMS), reduced by unperspective branches cutoff at the maximal early steps of generation. Such branches do not lead to the TMS, satisfying all conditions, entering filter of UMG/UMMG. Section 3 is dedicated to UMG/UMMG application to some actual problems from the STS assessment area (estimation of producing STS capabilities and resources, necessary to such systems for various orders completion, as well as assessment of STS sustainability and vulnerability to various destructive impacts, such as natural disasters, technogenic catastrophes, mutual sanctions, etc.). In Section 4, optimization problems, related to STS, are considered (their profit maximization and works' optimal distribution among non-antagonistic competing STS in the market economy). Conclusion contains list of directions of further development of multigrammatical approach.

## 2. Algorithmics of unitary multigrammars and multimetagrammars

Let us begin from **filtering unitary multigrammars** (FUMG).

From the computational complexity point of view, definition (58) from the first part of this work may be without loss of generated TMS transformed to

$$V_{(i+1)} = V_{(i)} \cup \left( \bigcup_{\substack{v \in V_{(i)} \\ \langle a \to n_1 \cdot a_1, \cdots, n_m \cdot a_m \rangle \in R}} \bigcup_{n \cdot a \exists \in v} \{v - \{n \cdot a\} + n * \{n_1 \cdot a_1, \cdots, n_m \cdot a_m\}\} \right) \quad (1)$$

where $\exists \in$ means selection of any one multiobject $n \cdot a$ from the multiset $v$ instead of repeating such selection for all multiobjects $n \cdot a \in v$ . This provides essential reduction of the computational complexity of TMS generation [1] and is basic for all algorithms, described lower in this section.

As may be seen, sufficiently valuable part of multisets, generated by FUMG unitary rules (UR) application, may be eliminated after few generation steps, because all the following steps do not lead to TMS, satisfying FUMG filter boundary conditions, or have no opportunity for further optimization over terminal multisets, generated earlier, if concerning optimizing conditions. So essence of general approach, which is described further, is to apply filter to every new generated multiset (not only terminal) and to cut off those multisets, which are not perspective in the aforementioned sense. Thus we apply well known and widely used in operations research "branches and bounds" scheme to TMS generation. Of course, filter application to generated nonterminal multisets cannot be identical to filter application to terminal multisets; that is why some additional considerations are necessary.

Let us take the definition of TMS generation logic (57)–(61) from the first part of the work as a basis and construct rather simple and transparent procedure-function terminal multisets generation (*TMSG*), providing reduced generation of set of terminal multisets, defined by FUMG.

We shall use the following variables in the *TMSG* body:

1. $v$, which value is current generated multiset.

2. $R$, which value is set of unitary rules (FUMG scheme), applied to multisets in order to generate new multisets.

3. $F$, which value is FUMG filter used for selection of terminal multisets to the resulting set $V$.

4. $V$, accumulating terminal multisets, satisfying filter $F$, while generation.

5. $FT$, which value is set of triples $<a, opt, l>$, each corresponding to optimizing condition $a = opt \in F$, where $a$ is object, $opt \in \{\max, \min\}$, and $l$ is current value of object $a$ multiplicity obtained after previous generation steps.

6. Couples $<a, w>$ and $<a, c>$, which are representations of unitary rule $a \leftarrow n_1 \cdot a_1, ..., n_m \cdot a_m$, where $w$ as well as $c$ is multiset $\{n_1 \cdot a_1, ..., n_m \cdot a_m\}$.

In *TMSG* body, $F$, $FT$, and $V$ are global variables, which are available from all subfunction calls while generation is executed. Note $F$ is read-only variable, while $V$ and $FT$ are updated (read-write) variables. All other variables are local and are used in the area of their functions in such a way that every new function call operates its own values of these variables.

*TMSG* body is the following:

```
TMSG: procedure (v, R, F) returns (V);
        variables F, V, FT global; variables v, R local;
        $v := \{\varnothing\}$; FT $:= \{\varnothing\}$;
        /* initial values of optimized multiplicities settings */
        do $a = opt \in F$;
        case opt:
            {min : if $k \le a \le k' \in F$
                        then $FT : \cup \{\langle a, \min, k' \rangle\}$;
                        else $FT : \cup \{\langle a, \min, \mathrm{MAX} \rangle\}$;
             max : if $k \le a \le k' \in F$
                        then $FT : \cup \{\langle a, \max, k \rangle\}$;
                        else $FT : \cup \{\langle a, \max, 0 \rangle\}$;
            };
        end F;
        /*main part: generation function G call */
        call G(v, R);
         /*function G body*/
        G: procedure (v, R);
           if v is terminal multiset
             then {call FILTER(v);
               return;};
        /*v is non-terminal multiset, and following operators provide selection of
        unperspective multisets and redundant generation cut-off */
        do $n \cdot a \in v$ where a is terminal object;
           if $k \le a \le k' \in F$ & $n > k'$ /* n already exceeds higher bound */
             then return;
           if $\langle a, \min, l \rangle \in FT$
             then if $l < n$ /* current minimized multiplicity of object a is already
              lower than n, which cannot decrease */
                        then return;
         end v;
         /* branch is perspective, so new multisets are generated */
        select $n \cdot a \exists \in v$ where a is non-terminal object;
        do $\langle a, w \rangle \in R$ ; /* all non-terminal object a alternatives */
           call $G(v - \{n \cdot a\} + n * w, R)$;
         end aw;
         end;
end G;
FILTER: procedure (v); /* generated TMS v filtration */
variables v, x local;
        do $n \cdot a \in v$ ;
           if $k \le a \le k' \in F$
             then if $(n < k) \vee (n > k')$ /* n is out of $[k, k']$ */
                   then return;
           if $\langle a, \min, l \rangle \in FT$
             then if $l < n$ /* n is greater than already stored min value */
                   then return;
           if $\langle a, \max, l \rangle \in FT$
             then if $l > n$ /* n is less than already stored max value */
```

```
                then return;
        end $na$;
        /* correction min/max values by new terminal multiset*/
        $x := 0$; /* flag "no values corrected" */
        do $\langle a, opt, l \rangle \in FT$
            do $n \cdot a \in v$ ;
                if $l \neq n$ /* at least one value corrected */
                    then $\{FT : -\{\langle a, opt, l \rangle\} \cup \{\langle a, opt, n \rangle\}$; /*replacement*/$x := 1\}$;
                    /*flag reset*/
            end $na$;
        end opt;
        if $x$=0 /* no values corrected */
            then $V : \cup \{v\}$; /* one more TMS added to the accumulated set */
            else $V := \{v\}$; /* replacement of earlier accumulated set */
        end $FILTER$;
    end $TMSG$
```

Let us comment on the represented procedure-function *TMSG*.

As seen, it contains prefix, which provides *V* and *FT* variable values initialization. *FT* value is set of triples, each corresponding to one optimizing condition, entering filter *F*. If there is boundary condition $k \le a \le k' \in F$, then initial value of object *a* multiplicity would be $k'$ in the case condition is $a = \min$, and *k* otherwise. Both values correspond to the worst cases. If there is not any boundary condition with the same object *a*, where $a = opt \in F$, then, obviously, the worst case for $a = \min$ is MAX (the largest possible multiplicity for the considered problem), while for $a = \max$, it is 0. After prefix execution, there is unique operation, returning result of recursive procedure *G* call with input values *v* and *R*, passed without any changes from *TMSG* call itself.

Procedure *G* is core of the described algorithm; it implements the main part of generation and consists of three sections.

First section corresponds to that case, when *v* is terminal multiset, and all that is necessary here is to apply FUMG filter to *v*, what is really done by procedure *FILTER* call with *v* input data. After this call, processing of TMS *v* is terminated.

If *v* is not terminal multiset, it is clear that *v* contains one or more nonterminal objects, which may be used for generation continuation. The last is performed by the second section of *G* in such a way that multiset *v* is checked, where it is perspective in the above sense or it may be eliminated from generation, because all TMS, generated from *v*, would not satisfy *F*. For this purpose, all terminal multiobjects are checked by two selection criteria:

1. If in terminal multiobject $n \cdot a$ multiplicity *n* already exceeds upper bound k′ of boundary condition $k \le a \le k' \in F$ . (it's obvious that while following generation steps, object *a* multiplicity would only increase or in the utmost case remain unchangeable).

2. If mentioned multiplicity is greater than value *l* already having place in the triple $\langle a, \min, l \rangle \in FT$ (again it's obvious that the following steps would not decrease this multiplicity, so all TMS generated from *V* would not satisfy optimizing filter $a = \min$).

(There may be more sophisticated and efficient criteria for earlier recognition and cutting off unperspective generation branches [2, 3], but chapter volume limits make their description impossible). If one of the checked conditions is not satisfied, further generation from multiset *v* is terminated by return from *G* without any operation.

The third section of $G$ corresponds to nonterminal multiset, which was not eliminated, being perspective, so generation is continued by all possible branches, corresponding to unitary rules with the same head, in full accordance with UMG semantics and improvement (1), by $G$ recursive calls with new input data.

Function *FILTER* with unique input (multiset $v$) implements check of all conditions, having place in filter $F$. This is done by the first **do-end** loop for all terminal multiobjects, entering $v$. If one of these checks failed, return from *FILTER* is performed without any additional actions. If all checks were successful, second section is executed. It begins from the installation of flag variable $x$ to 0 value; that means no min/max values in the accumulating variable $FT$ were replaced (i.e., $v$ has no multiobjects $n \cdot a$ with value $n$ more or less over already having place in $FT$). After that, **do-end** loop for all $FT$ elements is executed. If multiplicity $n$ of object $a$ in TMS $v$ is not equal to value $l$ in the considered element $\langle a, opt, l \rangle \in FT$; that means $n$ is less (when $opt = \min$) or greater (when $opt = \max$) than $l$, so $l$ must be replaced by $n$, and flag $x$ must get value 1 (at least one replacement was done). The third section of function *FILTER* operates according to variable $x$ value. If $x = 0$ (i.e., all optimized multiplicities in $v$ are equal to already obtained in the previous generation steps), then $v$ is joined to the resulting set $V$ as new element. If $x = 1$ (i.e., at least one multiplicity was replaced, so $v$ is "better" than earlier created and stored terminal multisets), then previous value of $V$ is replaced by one-element set $\{v\}$.

As seen, the described algorithm due to its simplicity may be implemented easily on every available software/hardware environment. Correctness of this algorithm is confirmed by the following statement [2, 3].

**Statement**. Let $S = \langle a_0, R, F \rangle$, and $TMSG$ $(\{1 \cdot a_0\}, R, F)$ is result of $TMSG$ call. Then

$$TMSG(\{1 \cdot a_0\}, R, F) = \overline{V}_S. \quad \blacksquare \tag{2}$$

(Note $TMSG$ operates only elementary boundary conditions $a\rho n$, not EBC $a\rho a'$, neither CBC. $TMSG$ generalization is not associated with any difficulties).

Let us describe now the main idea of **algorithmics of generation sets of TMS, defined by unitary multimetagrammars**.

As shown in [2, 3], all multisets, generated by any UMMG, have form

$$v = \left\{ C_{a_{i_1}} \cdot a_{i_1}, \dots, C_{a_{i_m}} \cdot a_{i_m} \right\}, \tag{3}$$

where every $C_{a_{i_j}}$ is so-called variables-containing multiplicity (VCM), being polynom of variables-multiplicities, having places in unitary metarules, used while generation of multiset $v$. In the general case, object $a$ VCM is

$$C_a = n_i^a + \sum_{i=1}^{N_{C_a}} n_i^a \cdot \left(\gamma_1^i\right)^{l_{i_1}^a} \cdot \dots \cdot \left(\gamma_{m_i}^i\right)^{l_{im_i}^a}, \tag{4}$$

where $N_{C_a}$ is number of monoms, each being product of all occurrences of variables-multiplicities and constants-multiplicities, having places in one generation branch, leading to object $a$.

If filter $F$ of UMMG contains boundary conditions

$$\begin{gathered} k_1 \le a_{i_1} \le k_1', \\ \dots. \\ k_l \le a_{i_l} \le k_l' \end{gathered} \tag{5}$$

as well as optimizing conditions

$$\begin{aligned} a_{j_1} &= opt_1, \\ &\dots. \\ a_{j_t} &= opt_t, \end{aligned} \tag{6}$$

they induce $l$ inequalities

$$\begin{aligned} k_1 \le C_{a_{i_1}} &\le k'_1, \\ &\dots. \\ k_l \le C_{a_{i_l}} &\le k'_l, \end{aligned} \tag{7}$$

and $t$ goal functions

$$\begin{aligned} C_{a_{j_1}} &\to opt_{j_1}, \\ &\dots. \\ C_{a_{j_t}} &\to opt_{j_t}. \end{aligned} \tag{8}$$

Domain of every variable $\gamma$, having place in polynoms $C_{a_{i_1}}, \dots, C_{a_{i_l}}, C_{j_1}, \dots, C_{j_t}$, is defined by boundary condition

$$k_\gamma \le \gamma \le k'_\gamma \in F. \tag{9}$$

As seen from (3)–(9), set of terminal multisets, generated in the UMMG case, corresponds to set of solutions of multicriterial problem of discrete polynomial programming. There are well-known approaches to such problems' consideration [4, 5], but their common feature is they provide search of any one of the solutions, not all multi-element solutions set, if it exists. Proposed UMMG TMS generation algorithmics [2, 3] is initially oriented to UMMG semantic precise implementation and combines mixed computation and interval analysis techniques [6–9] with global optimization based on theory [10–13]. Aforementioned algorithmics provides multidirectional reduction of redundant generation branches by procedure, similar to *TMSG*, and extended by splitting of intervals, defined by boundary conditions, which describe variable domains, to subintervals, until the last become points. Every such step is accompanied by the estimation of lower and upper bounds of VCMs, containing variable, which current domain is splitted, so if both bounds of at least one VCM are out of interval, defined by corresponding object multiplicity bounds, having place in UMMG filter, then created interval is eliminated, and generation branch is terminated.

As *TMSG*, another powerful tool of unperspective branches early recognition and cutoff is comparison of mentioned lower and upper bound estimates with already obtained values of optimized multiplicities. If corresponding optimizing condition is $a$ = min, and current value of object $a$ multiplicity, obtained as a result of previous steps execution, is $n$, then when lower bound estimate of VCM of $a$ is $\overline{n}$, and already $\overline{n}>n$, so further generation by this branch, which leads only to growth of VCM (or it remains unchangeable in the best case), is senseless, and branch may be terminated. Similarly, if optimizing condition is $a$ = max, and current value of corresponding multiplicity is $n$, while VCM upper bound estimate is $\overline{n}' < n$, then further generation by this branch will not lead to object $a$ multiplicity increase, and branch may be terminated.

VCM generation is based on unified representation of polynoms in (4) form as sets of multisets: $C_a$ is represented as

$$v_a = \left\{ \left\{ n_0^a \cdot \overline{\gamma}_0 \right\}, \left\{ n_1^a \cdot \overline{\gamma}_0, e_{1_1}^a \cdot \overline{\gamma}_1^1, \dots, e_{1_{m_1}}^a \cdot \overline{\gamma}_{m_1}^1 \right\}, \dots, \left\{ n_k^a \cdot \overline{\gamma}_0, e_{k_1}^a \cdot \overline{\gamma}_1^k, \dots, e_{k_{m_k}}^a \cdot \overline{\gamma}_{m_k}^k \right\} \right\}, \tag{10}$$

where $k = N_{C_a}$, $\overline{\gamma}_j^i$ are objects, corresponding to variables, while $\overline{\gamma}_0$ is fictive object, corresponding to constants, having place in polynom. For example, polynom

$$C_a = 5 + 3 \cdot (\gamma_1)^2 \cdot (\gamma_2)^4 + (\gamma_2)^5 \cdot \gamma_3 \tag{11}$$

is represented by multiset

$$v_a = \{\{5 \cdot \overline{\gamma}_0\}, \{3 \cdot \overline{\gamma}_0, 2 \cdot \overline{\gamma}_1, 4 \cdot \overline{\gamma}_3\}, \{1 \cdot \overline{\gamma}_0, 5 \cdot \overline{\gamma}_2, 1 \cdot \overline{\gamma}_3\}\}. \tag{12}$$

This representation is sufficiently flexible, and it is the basis of implementation of mixed computation in the multisets case; the core of this implementation is polynoms multiplication and addition.

More detailed description of algorithmics, providing efficient generation of sets of terminal multisets, defined by unitary multimetagrammars, needs separate survey.

Implementation issues, related with the proposed knowledge representation model, are described in [1, 14].

However, presented formal definitions of syntax, semantics, and algorithmics of UMG/UMMG are, in our opinion, sufficient for consideration of their pragmatics, that is, their application to various practical problems.

## 3. Assessment of the producing sociotechnical systems

Multigrammatical paradigm and UMG/UMMG toolkit are sufficiently general and simple to formalize and solve a lot of practical problems from various areas of operations research and systems analysis. Techniques, shortly described in Section 2 of the first part of this work, is one of the many possible to apply. Some more examples from hierarchical sociotechnical systems assessment and design concerned reader may find in [2, 3], where one may find also description of multigrammatical emulation of well-known classical problems of optimization theory: shortest path, traveling salesman, maximal flow, maximal pair matching, optimal assignments problems, and transport problem as well as integer linear programming problem. (Note that in [2, 3], there is also analysis of interconnections between multigrammars' family and known computational models, such as Petri nets, vectors addition, substitution systems, etc.).

Lower in this section, we shall consider problems, associated with the producing (manufacturing) STS, being most complicated for modeling.

Let us introduce the following **structural interpretation of unitary rules**.

We shall understand UR

$$a \to n_1 \cdot a_1, \dots, n_m \cdot a_m \tag{13}$$

as follows: object $a$ consists of $n_1$ objects $a_1$, …, $n_m$ objects $a_m$ .

In turn, **technological interpretation of unitary rules** is generalization of the structural one and is as follows: production of one object (unit of resource) $a$ requires $n_1$ objects (units of resource) $a_1$, …, $n_m$ objects (units of resource) $a_m$. One may consider (13) as a black box, representing producing device or manufacturing facility (factory, plant, etc.), containing a lot of such devices, working cooperatively. Set $R$ of such URs represents technological base (TB) of some social group, possessing represented by this set producing (manufacturing) equipment. If $R$ contains $l > 1$ URs with identical head and different bodies, this means that one and the same object may be produced in $l$ various ways (by $l$ various devices, or by one and the same device, but by various methods, or by $l$ various facilities). Objects, having places in URs, may be manufactured devices, their blocks, spare parts, chips, pieces of connecting cables, various measured resources involved (necessary amounts of electrical energy, liquids, solid materials, etc. "down to ore"), as well as time and money. Manufactured devices may be, in turn, manufacturing ("means of production") and may be used further in production processes/chains.

Unitary multigrammars provide most natural "top-down" way of formal description of technological base of arbitrary producing STS, as well as deep structure of manufactured objects of any level of structural complexity (obviously, these two entities are interconnected closely). "Additivity" of multigrammatical knowledge bases (KB), being consequence of their "granularity" (due to unitary rules and metarules as knowledge representation atoms), provides creating and updating KB in near real time. Since now we shall use notations "multigrammatical knowledge base" and "scheme of UMG/UMMG" as synonyms.

**Example 1.** Consider car, consisting of body, engine, transmission, four wheels, and fuel cistern. Car body, in turn, consists of frame, front and back glasses, engine cover, baggage place, and two first and two second doors. Engine includes motor, cooling system, and accumulator. All the said may be represented by the following set of unitary rules in the structural interpretation:

$$\begin{aligned}
&car \to 1\cdot body,\, 1\cdot engine,\, 1\cdot transmission,\, 4\cdot wheel,\, 1\cdot fuel\text{-}cisterm,\\
&body \to 1\cdot frame,\, 1\cdot front\text{-}glass,\, 1\cdot back\text{-}glass,\, 1\cdot engine\text{-}cover,\, 1\cdot baggage\text{-}place,\\
&\qquad 2\cdot first\text{-}door,\, 2\cdot back\text{-}door,\\
&engine \to 1\cdot motor,\, 1\cdot cooling\text{-}system,\, 1\cdot accumulator. \quad \blacksquare
\end{aligned}$$

As it is easy to see, all terminal objects may become nonterminal after joining to this set of new URs, detailing them down to undivided spare parts. If to follow technological interpretation, then every UR reflects assembling operation, implemented by corresponding segment of manufacturing facility: one such segment is assembling car of the listed components, another segment - body, etc.

To take into account cost of any operation executed (obviously, it is "*added value*" *in K. Marx terminology*), as well as time interval necessary for its execution, it is sufficient to include to UR bodies multiobjects like $n\cdot e$ and $m\cdot t$, where $e$ and $t$ are fixed-name objects being cost and time measurement units (e.g., *usd* and *sec*). There may be recalculation of both to another unit by including to the KB additional rules, reflecting currencies interrelations and different time scales, for example,

$$eur \to 1.15\cdot usd, \tag{14}$$

$$hour \to 60\cdot mnt, \tag{15}$$

$$mnt \to 60\cdot sec \tag{16}$$

(rational multiplicities' appearance along with integer ones, considered higher, does not bring any principal transformations and difficulties into MG semantics and algorithmics [2, 3]). Both cost and time may be defined in "compound" units, that is, UR body may contain three multiobjects, $3\cdot hour$, $23\cdot \min$, $15\cdot \sec$, that after application of URs (14)–(16) will be transformed to one multiobject, $10953\cdot \sec$.

Considering time intervals description in unitary rules, we must take into account that, unlike cost, time is not fully additive resource, because producing devices may operate in parallel. That's why time is additive resource only regarding separate device, and typical form of "local time" description is multiobject $n \cdot \langle t - x \rangle$, where $x$ may be manufacturing device name as well as assembled technical object name, while $t$ is time measurement unit (here angle brackets are used for syntactical unambiguity).

**Example 2.** Let us consider the following cost and time parameters, implanted to URs from example 1. Let cost of one car assembling is 3000 USD; body, 2000 USD; and engine, 6000 USD; time interval necessary for one car assembling is 28 minutes, body, 21 minutes; and engine, 63 minutes. Then URs from Example 1 may be rewritten in the following way:

$$\begin{aligned}
&car \to 28 \cdot \langle mnt - car \rangle, 3000 \cdot usd, 1 \cdot body, 1 \cdot engine, 1 \cdot transmission, 4 \cdot wheel,\\
&\quad 1 \cdot fuel\text{-}cisterm,\\
&body \to 21 \cdot \langle mnt - body \rangle, 2000 \cdot usd, 1 \cdot front\text{-}glass, 1 \cdot back\text{-}glass, 1 \cdot engine\text{-}cover, 1 \cdot baggage\text{-}place,\\
&\quad 2 \cdot first\text{-}door, 2 \cdot back\text{-}door,\\
&engine \to 60 \cdot \langle mnt - engine \rangle, 6000 \cdot usd, 1 \cdot motor, 1 \cdot cooling\text{-}system, 1 \cdot accumulator. \quad \blacksquare
\end{aligned}$$

If we have knowledge base, prepared as shown above, then it may be used to estimate resources amounts, necessary to complete any order by means of technological base, defined by $R$. Order may be represented as multiset

$$q = \{m_1 \cdot b_1, \ldots, m_k \cdot b_k\}, \tag{17}$$

which means customer needs $m_1$ objects $b_1$, …, $m_k$ objects $b_k$.

Then, as it is easy to see, resources collection, necessary to complete this order, is terminal multiset $\overline{v}$ being element of set of TMS $\overline{V}_{S_q}$, where

$$S_q = \langle a_q, R_q \rangle, \tag{18}$$

$$R_q = R \cup \{\langle a_q \to m_1 \cdot b_1, \ldots, m_k \cdot b_k \rangle\} \tag{19}$$

(unitary rule, having place in (19), is in angle brackets for unambiguity). If unitary multigrammar $S_q$ generates one-element SMS, that is, $|\overline{V}_{S_q}| = 1$, there is unique variant of aforementioned resources collection. Otherwise, $|\overline{V}_{S_q}|>1$, and there are various ways of some objects' assembling, each consuming its own resources collection.

**Example 3.** Let $q = \{3 \cdot car\}$. Then $R_q$ consists of all URs from Example 2 and unitary rule

$$order \to 3 \cdot car, \tag{20}$$

that is, order is to assemble three cars. According to (17)–(19),

$$\begin{aligned}
\overline{V}_{S_q} = \{\{&84 \cdot \langle mnt\text{-}car \rangle, 63 \cdot \langle mnt\text{-}body \rangle, 180 \cdot \langle mnt\text{-}engine \rangle, 27000 \cdot usd,\\
&3 \cdot transmission, 12 \cdot wheel, 3 \cdot fuel\text{-}cistern, 3 \cdot front\text{-}glass, 3 \cdot back\text{-}glass,\\
&3 \cdot engine\text{-cover}, 3 \cdot baggage\text{-place}, 6 \cdot \text{first-door}, 6 \cdot \text{back-door},\\
&3 \cdot motor, 3 \cdot cooling\text{-}system, 3 \cdot accumulator\}\}.
\end{aligned}$$

That means order completion requires spare parts from external suppliers as well as money and time for assembling segments of manufacturing facility in amounts, being multiplicities of corresponding objects, having places in $\overline{V}_{S_q}$. There is the only one variant of resources set necessary for order completion. ■

If it is necessary to evaluate (estimate) total cost of order completion, then it is sufficient to join to the KB $R$ unitary rules, defining costs of all necessary spare parts and other external ("outsourced") resources. These URs may have form $a \to n \cdot e$, where $a$ is terminal object from $R$, while $e$ is monetary unit, used for cost calculation. As may be seen, if one would eliminate from URs time-defining multiobjects, then $e$ becomes unique terminal object of the created scheme $R'$, and set of terminal multisets generated by UMG $S' = \left\langle a_q, R'_q \right\rangle$ would contain one-element TMS of form $\{\overline{n} \cdot e\}$, where $\overline{n}$ is total cost of order completion, corresponding to one of its variants (as was said higher, there may be more than one such variant).

**Example 4.** Let us add to the KB from Example 3 the following unitary rules, defining prices of car the outsourced elements:

$$\begin{aligned} &transmission \to 100 \cdot usd, \\ &wheel \quad\quad\;\; \to 50 \cdot usd, \\ &\qquad\qquad\quad\;\; \ldots \\ &accumulator \to 10 \cdot usd. \end{aligned}$$

As seen, new UMG provides generation of one-element set

$$\{\{84 \cdot \langle mnt\text{-}car \rangle, 63 \cdot \langle mnt\text{-}body \rangle, 180 \cdot \langle mnt\text{-}engine \rangle, X \cdot usd\}\},$$

where $X$ is total cost of order completion, that is, three car assembling.

If multiobjects $28 \cdot \langle mnt\text{-}car \rangle$, $21 \cdot \langle mnt\text{-}body \rangle$ and $60 \cdot \langle mnt\text{-}engine \rangle$ are eliminated from UR set presented in Example 10, then obtained UMG will generate one-element set $\{\{X \cdot usd\}\}$. ■

In practice every STS, which is capable to complete input orders and manufacture some output production, possesses usually not only technological base but also resource base (RB). For this reason, **assessment of STS capability to orders completion** requires comparison of two resources collections—being in system's ownership and necessary. In multigrammatical paradigm, this problem is solved quite simply.

Let $R$ be technological base, while multiset $\overline{v} = \{\overline{m}_1 \cdot \overline{b}_1, \ldots, \overline{m}_k \cdot \overline{b}_k\}$ is resource base of the system, that is, STS possesses $\overline{m}_1$ objects $\overline{b}_1$, ..., $\overline{m}_k$ objects $\overline{b}_k$. As it is easy to see, order $q = \{m_1 \cdot b_1, \ldots, m_l \cdot b_l\}$ may be completed by this system, if there exists multiset $v \in \overline{V}_{S_q}$ such that

$$v \subseteq \overline{v}, \tag{21}$$

that is, collection of resources, belonging to the STS, is sufficient for order $q$ completion by one of the implementable ways. In this case order completion is possible. Otherwise, that is, if there is no one multiset $v \in \overline{V}$ satisfying (21), then system is not able to complete $q$.

**Example 5.** Let KB from Example 2 be STS technological base, while

$$\begin{aligned} \overline{v} = \{&6 \cdot transmission, 10 \cdot wheel, 2 \cdot fuel\text{-}cistern, 7 \cdot front\text{-}glass, \\ &9 \cdot back\text{-}glass, 180 \cdot \langle mnt\text{-}car \rangle, 240 \cdot \langle mnt\text{-}body \rangle, 600 \cdot \langle mnt\text{-}engine \rangle, \\ &1000000 \cdot eur, 500000 \cdot usd\} \end{aligned}$$

is its resource base. As may be seen, this resource base is not sufficient for order $q = \{3 \cdot car\}$ from Example 11 completion, because condition (21) is not satisfied: there are objects, entering all $v \in \overline{V}_{S_q}$, which do not enter $\overline{v}$ at all (*motor, accumulator*, etc.). At the same time multiplicities of some terminal objects are less than it is necessary for order $q$ completion $\left(2 \cdot fuel\text{-}cistern \in \overline{v} \text{ while } 3 \cdot fuel\text{-}cistern \in v \in \overline{V}_{S_q}\right)$. ■

After recognition of system's inability to complete order $q$, there may be two questions:

1. What amount of resources must be acquired by the system to complete the order?

2. What part of order may be completed, given resources, owned by STS?

The answer to the first question is obvious: if $v \in \overline{V}_{S_q}$, and $v$ is not subset of $\overline{v}$, then additional amount of resources, necessary for order completion, is $\overline{v}–v$. Thus variants of necessary resources acquisition are elements of set of terminal multisets

$$\Delta\overline{V}_{S_q} = \left\{\overline{v}-v | v \in \overline{V}_{S_q}\right\}. \tag{22}$$

The answer to the second question concerned reader may find in [1], where the so-called reverse multigrammars are used for this problem solution.

One more area of useful application of UMG/UMMG is **assessment of sociotechnical systems sustainability/vulnerability to various destructive impacts,** which was in details considered in [1]. The main result of this work is as follows.

Let $R_q$ be scheme, corresponding to technological base of the system and order $q$ in the sense (17)–(19); $\overline{v}$ is total resource of this system (resource base, joined with multiset representation of technological base, as it was suggested in [1]), and $\Delta\overline{v}$ is impact, destructing some part of the aforementioned total resource. We shall call STS with technological base $R$ and total resource $\overline{v}$ sustainable to impact $\Delta\overline{v}$ while completing order $q$, if

$$\left(\exists v \in \overline{V}_{S_q}\right) v \subseteq \overline{v} - \Delta\overline{v}. \tag{23}$$

That is, despite impact there is at least one way of order completion. Otherwise, if there is no one TMS $v \in \overline{V}_{S_q}$, satisfying (23), considered STS is vulnerable to impact $\Delta\overline{v}$.

## 4. Optimization problems

Another important issue to be discussed here is **profit optimization in production economy** (POPE). We shall consider it not only because of its practical value but also in order to illustrate techniques of UMG/UMMG application to the multigrammatical representation and solution of classical optimization problems.

POPE is formulated as follows [15]. Let there be $n$ products, manufactured by STS, $x_i$ is amount of $i$-th product, and $c_i$ is its price. Profit, which producing STS would obtain, is

$$C = \sum_{i=1}^{m} c_i \cdot x_i. \tag{24}$$

To produce one unit of $i$-th product, the system needs respective resources, namely, $a_{ij}$ units of the $j$-th resource, where $j = 1, \ldots, n$, and $n$ is total number of different resources, consumed by the system for production. There are $b_1, \ldots, b_n$ amount of resources available, and the problem is to maximize profit $C$ under restrictions

$$\sum_{i=1}^{m} a_{ij}x_i \leq b_j, \tag{25}$$

where $j = 1, \ldots, n$.

This problem may be represented by unitary multiset metagrammar $S = <q, R, F>$, which scheme $R$ contains one unitary metarule

$$q \to x_1 \cdot u_1, \ldots, x_m \cdot u_m, \tag{26}$$

where $x_1, \ldots, x_m$ are variables, and $m$ unitary rules

$$u_i \to ai_1 \cdot e_1, \ldots, ai_n \cdot e_n, c_i \cdot e, \tag{27}$$

where $i = 1, \ldots, m$, and $a_{ij}$ is nonzero amount of $j$-th resource, needed for $i$-th product, while $c_i$ is its price. Filter $F$ contains $n$ boundary conditions

$$e_j \leq b_j, \tag{28}$$

where $j = 1, \ldots, n$, as well as one optimizing condition

$$e = \max, \tag{29}$$

along with $m$ variable declarations

$$0 \leq x_i \leq M_i, \tag{30}$$

where $i = 1, \ldots, m$ and $M_i$ is maximal amount of $i$-th product, which may be manufactured by the system. As seen, terminal objects $e_1, \ldots, e_n$ are measurement units of corresponding resources, while terminal object $e$ is price measurement unit. Nonterminal objects $u_1, \ldots, u_m$ represent products, and UMR (26) along with optimizing condition (29) represents order, while $n$ URs (27) represent STS technological base. STS resource base, as it was introduced higher, is represented by boundary conditions (28).

As may be seen, set of POPE solutions is

$$\overline{V_S} = \{v_1, \ldots, v_t\}, \tag{31}$$

where

$$\{l_1 \cdot e_1, \ldots, l_n \cdot e_n, C \cdot e, p_1 \cdot \overline{x}_1, \ldots, p_m \cdot \overline{x}_m\} \in \overline{V_S} \tag{32}$$

means that maximal profit is $C$ and it is gotten when STS produces $p_1$ units of the first product, ..., $p_m$ units of the $m$-th product. In general case

$$|\overline{V_S}| > 1, \tag{33}$$

so there may be $t>1$ solutions of the specific POPE.

**Example 6.** Let POPE is

$$2x_1 + 3x_2 \to \max$$

under restrictions

$$3x_1 + 2x_2 \leq 10$$
$$5x_1 + 3x_2 \leq 18.$$

That means STS is producing two products, which prices are 2 and 3, respectively, and there is resource base, containing 10 units of the first resource and 18 units of the second. To produce one unit of the first product, STS needs 3 units of the first resource and 5 units of the second, while producing of one unit of the second product needs 2 units of the first resource and 3 units of the second resource.

According to (26)–(30), scheme $R$ of the corresponding UMMG $S = <q, R, F>$ includes UMR

$$q \to x_1 \cdot u_1, x_2 \cdot u_2,$$

as well as two URs:

$$u_1 \to 3 \cdot e_1, 5 \cdot e_1, 2 \cdot e,$$
$$u_2 \to 2 \cdot e_1, 3 \cdot e_2, 3 \cdot e.$$

Filter $F$ contains two boundary conditions

$$e_1 \leq 10,$$
$$e_2 \leq 18,$$

one optimizing condition

$$e = \max,$$

as well as two variables declarations:

$$0 \leq x_1 \leq 10,$$
$$0 \leq x_2 \leq 10,$$

where 10 is maximal amount of any product, which may be produced by STS.
As seen,

$$\overline{V}_S = \{\{10 \cdot e_1, 16 \cdot e_2, 10 \cdot \text{e}, 2 \cdot \overline{x}_1, 2 \cdot \overline{x}_2\}\},$$

which means STS would get maximal profit of 10 units, producing 2 units of both products and, spending for that purpose, 10 units of the first resource and 16 units of the second resource. ■

Let us note that classical matrix–vector POPE modeling is limiting set of the considered cases to the simplest two-level structures of the manufactured objects ("object-component"), represented by unitary rules like (27). In practice, all such objects have much more complicated, multilevel heterogeneous hierarchical structure, that is clearly illustrated by the previous Examples 1–3, concerning car manufacturing.

UMMG application provides natural representation of the POPE problem in the most general formulation. Namely, it is sufficient to join to set of URs, describing technological base of the STS, the only UMR like (26). Similarly, to represent resource base of the STS, filter of the created UMMG would contain boundary conditions like (28); it is important that absence of some resource $e$ in the RB must be represented by boundary condition $e = 0$; otherwise any generated TMS, containing multiobject $n \cdot e$, where $n > 0$, may enter one of the solutions, and this contradicts reality. The goal of STS is defined by the optimizing condition like (29), and domains of variables, having place in the aforementioned UMR, are defined by variable declarations like (30). If it is necessary to maximize profit, taking into account expenses for some resources acquisition, it is very convenient to use representation of prices of the acquired resources, as it was illustrated by Example 4, but with negative multiplicities (such techniques are described in [2, 3]), that is, as to URs from Example 4,

$$\begin{aligned} &\textit{transmission} \rightarrow \text{-}100 \cdot \textit{usd}, \\ &\textit{wheel} \qquad\quad\;\; \rightarrow \text{-}50 \cdot \textit{usd}, \\ &\qquad\qquad\quad\;\; \dots \\ &\textit{accumulator} \rightarrow \text{-}10 \cdot \textit{usd}. \end{aligned}$$

To cut off generated TMS with nonpositive multiplicities of object $e$ (such TMS correspond to unprofitable variants), it is sufficient to join to UMMG filter one more boundary condition $e > 0$. However, the use of negative multiplicities leads to some corrections in UMG/UMMG algorithmics, which would be considered separately.

All the said higher in this section is very close to the Leontief model and other "input–output" models of mathematic economy, developed on the matrix–vector algebra basis [16]. As may be seen, transfer to the UMG/UMMG basis makes such modeling much more flexible and closer to the reality. That is why we consider multigrammatical paradigm as very perspective for the development of various issues in the future digital economy [17, 18], first of all, planning and scheduling in the cyberphysical industry, integrated with deeply robotized logistics [19, 20]. However, application of the described here approach to the core areas of digital economy (Industry 4.0) needs separate publications.

Concerning implementation issues, it would be aptly to say that multisets processing is very promising area for application of non-conventional computing paradigms [21, 22].

Now let us spend some place of this section for **multiset modeling of competitions and works distribution in the concurrent environment**, typical for market economy.

The main tool of the last is the so-called variative unitary multigrammars and multimetagrammars, which schemes include unitary rules (metarules) with the same head and different bodies. UMG/UMMG variativity provides representation of coexistence of various subjects able to complete one and the same order. We assume these subjects are non-antagonistic, that is, they all are ready to execute any part of the total work.

There may be at least three possible approaches to multigrammatical modeling of competitions:

1. "The winner takes it all."

2. Splitting order among various subjects.

3. "The winner coalition takes it all" (combination of two previous).

Let us consider the *first* approach.

Let $R$ be set of unitary rules containing, among others, $k$ URs with one and the same head and different bodies:

$$\begin{gathered} a \leftarrow n_1^1 \cdot a_1^1, \dots, n_{m_1}^1 \cdot a_{m_1}^1, \\ \dots \\ a \leftarrow n_1^k \cdot a_1^k, \dots, n_{m_k}^k \cdot a_{m_k}^k. \end{gathered} \tag{34}$$

where $i$-th alternative corresponds to $i$-th subject of considered technological base, able to produce object $a$, consuming for that purpose

$n_1^i$ objects $a_1^i, \ldots, n_{m_i}^i$ objects $a_{m_i}^i$. To simplify and unify recognition of variant implemented, let us introduce $k$ terminal objects $s_1, \ldots, s_k$, being names of corresponding subjects, and replace set (34) by

$$\begin{gathered} a \leftarrow 1 \cdot s_1, n_1^1 \cdot a_1^1, \ldots, n_{m_1}^1 \cdot a_{m_1}^1, \\ \ldots \\ a \leftarrow 1 \cdot s_k, n_1^k \cdot a_1^k, \ldots, n_{m_k}^k \cdot a_{m_k}^k. \end{gathered} \tag{35}$$

Let $q = \{n \cdot a\}$, and $R_q$ is set of URs, containing UR

$$a_q \rightarrow n \cdot a, \tag{36}$$

$k$ URs (35), and all unitary rules from set $R$, excluding (35). Consider UMG $S_q = \langle a_q, R_q \rangle$. As seen, $\overline{V}_{S_q}$ is set of terminal multisets, each corresponding to some variant of order $q$ completion. If we establish filter $F_q$ to select one and only one element of $\overline{V}_{S_q}$, transforming UMG $S_q$ to FUMG $S_q = \langle a_q, R_q, F_q \rangle$, this element will be

$$\left\{n \cdot s_i, m_1^i \cdot \overline{b}_1^i, \ldots, m_{l_i}^i \cdot \overline{b}_{l_i}^i\right\}, \tag{37}$$

where multiobject $n \cdot s_i$ corresponds to $n$ operation cycles of subject $s_i$ during order $q$ completion and every multiobject $m_j^i \cdot \overline{b}_j^i$ corresponds to $m_j^i$ terminal objects $\overline{b}_j^i$, required for these cycles' implementation. In this context, filter $F_q$ may contain boundary conditions, defining required resources limits, as well as optimizing conditions, defining some of terminal object $\overline{b}_j^i$ multiplicities as minimal (e.g., cost, electrical energy, or fuel consumed) or maximal (e.g., some integral parameters of quality of produced objects or given services). If subject $s_i$ becomes the only winner, it takes all order to complete.

**Example 7.** Let us consider following unitary rules:

$$\begin{aligned} &car \rightarrow 1 \cdot first, 30 \cdot \langle mnt\text{-}car \rangle, 2800 \cdot usd, 1 \cdot accessories - set, \\ &car \rightarrow 1 \cdot \text{second}, 40 \cdot \langle mnt\text{-}car \rangle, 2700 \cdot usd, 1 \cdot accessories - set, \\ &car \rightarrow 1 \cdot third, 45 \cdot \langle mnt\text{-}car \rangle, 2500 \cdot usd, 1 \cdot accessories - set. \end{aligned}$$

These URs being joined with URs, detailing nonterminal object *accessories set* from $R$, describe competition of three car manufacturers (first, second, and third), assembling cars from one and the same accessories but differing by the time spent for this operation and its cost. If we consider FUMG $S_q = \langle a_q, R, F \rangle$, where $q = \{3 \cdot car\}$ and $F = \{usd = \min\}$, then

$$\overline{V}_{S_q} = \{\{1 \cdot third, X \cdot usd, 135 \cdot \langle mnt \cdot car \rangle, \ldots\}\},$$

which corresponds to the choice of the third manufacturer. If $F = \{usd = \min, \langle mnt\text{-}car \rangle = \min\}$, then

$$\overline{V}_{S_q} = \{\varnothing\},$$

because no one of the possible order executors is optimal by time and cost simultaneously. ■

As it is well known from practice, approach, described higher, may be not rational from various points of view, especially, when capabilities of no one of the competitors (subjects $s_1, ..., s_k$) are not sufficient for the whole order completion. In this case, more rational may be the *second* approach when order is splitted between non-antagonistic (cooperating) competitors in such a way that the total amount of objects produced is distributed among subjects $s_1, ..., s_k$. This techniques may be modeled by unitary multimetagrammar $S_q$, which scheme $R_q$ includes unitary metarule

$$a_q \to \gamma_1 \cdot b_1, ..., \gamma_k \cdot b_k, \tag{38}$$

and unitary rules

$$\begin{aligned} b_1 &\to 1 \cdot a, n_1^1 \cdot a_1^1, ..., n_{l_1}^1 \cdot a_{l_1}^1, \\ &\dots \\ b_k &\to 1 \cdot a, n_1^k \cdot a_1^k, ..., n_{l_k}^k \cdot a_{l_k}^k, \end{aligned} \tag{39}$$

as well as all URs, having place in the scheme, constructed higher by application of the first approach, excluding (34). Filter $F_q$ contains boundary condition

$$a = n, \tag{40}$$

which is directly induced by order $q = \{n \cdot a\}$ and boundary conditions, defining domains of variables $\gamma_1, ..., \gamma_k$:

$$0 \le \gamma_i \le n. \tag{41}$$

As seen, terminal multisets, generated by UMMG $S = \langle a_q, R_q, F_q \rangle$, have form

$$\left\{ n \cdot a, n_1 \cdot \bar{\gamma}_1, ..., n_k \cdot \bar{\gamma}_k, m_1^i \cdot \bar{b}_1^i, ..., m_{l_i}^i \cdot \bar{b}_{l_i}^i \right\}, \tag{42}$$

where, according to (38)–(40),

$$\sum_{i=1}^{k} n_i = n, \tag{43}$$

and thus values $n_1, ..., n_k$ are parts of order $q = \{n \cdot a\}$, distributed among subjects $s_1, ..., s_k$, respectively: $s_1$ will produce $n_1$ objects $a$, $s_2 - n_2$ objects $a$, up to $s_k$, which will produce $n_k$ objects $a$.

**Example 8.** Let us transform set $R$ from Example 7 to the following:

$$\begin{aligned} &\textit{order} &&\to \gamma_1 \cdot \textit{first}, \gamma_2 \cdot \text{second}, \gamma_3 \cdot \textit{third}, \\ &\textit{first} &&\to 1 \cdot \textit{car}, 2800 \cdot \textit{usd}, 2800 \cdot \textit{usd}\text{-}1, 1 \cdot \textit{accessories-set}, \\ &\text{second} &&\to 1 \cdot \textit{car}, 2500 \cdot \textit{usd}, 2500 \cdot \textit{usd}\text{-}2, 1 \cdot \textit{accessories-set}, \\ &\textit{third} &&\to 1 \cdot \textit{car}, 2200 \cdot \textit{usd}, 2200 \cdot \textit{usd}\text{-}3, 1 \cdot \textit{accessories-set}. \end{aligned}$$

If order $q = \{10 \cdot car\}$, then $F_q = \{car = 10, 0 \le \gamma_1 \le 10, 0 \le \gamma_2 \le 10, 0 \le \gamma_3 \le 10\} \cup F'_q$, where $F'_q$ may contain boundary and optimizing conditions, selecting terminal multisets, for example, $F'_q = \{usd = \min, usd\text{-}1 \ge 2800, usd\text{-}2 \ge 5000, usd\text{-}3 \ge 6600\}$. According to such $F_q$, set of terminal multisets, generated by UMMG $S_q = \langle a_q, R_q, F_q \rangle$, may contain element of the form $\{10 \cdot car, 25600 \cdot usd, 14000 \cdot usd\text{-}1, 5000 \cdot usd\text{-}2, 6600 \cdot usd\text{-}3, \ 5 \cdot \gamma_1, 2 \cdot \gamma_2, 3 \cdot \gamma_3, ...\}$, which corresponds to splitting order $q$ in such a way that five cars would be assembled by the first manufacturer, two by the second, and three by the third. ■

Let us underline once more that time is not additive resource in relation to parallel processes; it is additive only regarding one device (manufacturing unit). Consideration of multisets with time-containing multiobjects is separate direction of the multigrammatical approach and needs special tool, which is called temporal multiset grammars (TMG), announced in [1] . This branch concerns problems, addressed by the classical theory of scheduling [23, 24].

The third possible case of competitions ("the winner coalition takes it all") may be considered by the concerned reader on his (her) own.

## 5. Conclusion

Presented primary survey of multigrammatical knowledge representation along with brief consideration of its possible applications is, of course, only a background for future development, which most valuable directions may be:

1. MG/UMG/UMMG extension by features, necessary for "single-time-scale" modeling of manufacturing and logistical processes and their optimal control, that is critically needed for the developed digital economy (Industry 4.0)

2. Development of algorithmics for local correction of solutions (generated sets of TMS) while UMG/UMMG local correction in the sense [25], which is necessary for the aforementioned control in hard real-time and highly volatile environment

3. Further development of MG/UMG/UMMG improved algorithmics and its software/hardware implementation in high-parallel general-purpose computing environments

4. Development of specialized high-parallel computing environments, initially oriented to MG/UMG/UMMG algorithmics implementation

5. Development of quantum, neural and molecular algorithmics for MG/UMG/UMMG toolkit implementation in corresponding computer environments

6. MG/UMG/UMMG pragmatics expansion to new problem areas and convergence with other known knowledge/data engineering paradigms (first of all, multiagent systems [15, 26, 27])

Some of the listed directions are already developed by the author and his colleagues; some are waiting their time, being targeted to the creation of unified framework for the intellectual (knowledge-based) digital economy. This way is leading us to the Big Knowledge paradigm being generalization of the Big Data one, which is already everyday reality. The author will be glad, if this paper will be of any interest for some scholars working in the related areas.

## Acknowledgements

The author is grateful to Prof. Fred Roberts for useful discussions and support, and to Prof. Jeffrey Ullman, whose useful remarks on the primary version of this

work contributed to its essential upgrade. A significant incentive for the development of the proposed approach was its positive assessment by Prof. Noam Chomsky, which early works on syntactic structures formed a conceptual background of the described mathematical toolkit.

## Author details

Igor Sheremet
Financial University under the Government of Russian Federation, Moscow, Russia

*Address all correspondence to: sheremet@rfbr.ru

## References

[1] Sheremet I. Multiset analysis of consequences of natural disasters impacts on large-scale industrial systems. Data Science Journal. **17**(4): 1-17. DOI: 10.5334/dsj-2018-004

[2] Sheremet IA. Recursive Multisets and Their Applications. Moscow: Nauka; 2010. p. 293. (In Russian)

[3] Sheremet IA. Recursive Multisets and Their Applications. Berlin: NG Verlag; 2011. p. 249

[4] Lasdon SL. Optimization Theory for Large Systems. NY: Dover Publications; 2013. p. 560

[5] Hemmecke R, Koppe M, Lee J, Weismantel R. Nonlinear Integer Programming. In: 50 Years of Integer Programming 1958–2008: The Early Years and State-of-the-Art Surveys. NY: Springer-Verlag; 2009. pp. 1-57. DOI: 10.1007/978-3-540-68279-0-15

[6] Ershov AP. On the partial computation principle. Information Processing Letters. 1977;**2**(2):38-41

[7] Bjorner D, Ershov AP, Jones ND, editors. Partial Evaluation and Mixed Computation. North Holland: Amsterdam; 1988. p. 625

[8] Itkin VE. An algebra of mixed computation. Theoretical Computer Science. 1991;**90**(1):81-93

[9] Lloyd JW, Shepherdson JC. Partial evaluation in logic programming. Journal of Logic Programing. 1991;**11** (3–4):217-242. DOI: 10.1016/0743-1066 (91)90027–M

[10] Hansen E. Global optimization using interval analysis—The one-dimensional case. Journal of Optimization Theory and Applications. 1979;**29**:331-334

[11] Hansen E. Global Optimization Using Interval Analysis. NY: Marcel Dekker; 1992. p. 284

[12] Hansen E, Walster GW. Global Optimization Using Interval Analysis. NY: Marcel Dekker; 2004. p. 530

[13] Fiedler M, Nedoma J, Ramik J, Rohn J, Zimmerman K. Linear Optimization Problems with Inexact Data. NY: Springer; 2006. p. 227

[14] Sheremet IA. Augmented Post Systems: The Mathematical Framework for Knowledge and Data Engineering in Network-Centric Environment. Berlin: EANS; 2013. p. 395

[15] Shoham Y, Leyton-Brown K. Multiagent Systems. Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge, UK: Cambridge University Press; 2009. p. 532

[16] Bjerkholt O, Kuzz HD. Introduction: The history of input-output analysis, Leontief's path and alternative tracks. Economic Systems Research. 2006; **18**(4):331-333. DOI: 10.1080/ 09535310601030850

[17] Schwab K. The Fourth Industrial Revolution. What It Means and How to Respond. Foreign Affairs. December 12, 2015. Available from: https://foreignaffairs.com/articles/2015-12-12/fourth-industrial-revolution

[18] Baller S, Dutta S, Lanvin B, editors. The Global Information Technology Report 2016. Innovating in the Digital Economy. Geneve: WEF and INSEAD; 2016. p. 62

[19] Colombo AW, Bangemann T, Karnouskos T, Delsing J, Stluka P, Harrison R, et al. Industrial Cloud-Based Cyber-Physical Systems:

The IMC-AESOP Approach. NY: Springer Verlag; 2014. p. 245

[20] Lee J, Bagheri B, Hung-An K. A cyber-physical systems architecture for Industry 4.0-based manufacturing systems. Manufacturing Letters. 2015;**3**: 18-23. DOI: 10.1016/j.mfglet.2014.12.001

[21] Montanaro A. Quantum algorithms: An overview. npj Quantum Information. 2015;**2**:15023. DOI: 10.1038/npjqi.2015.23

[22] Farhi E, Goldstone J, Gutmann S. A quantum approximate optimization algorithm applied to a bonded occurrence constraint problem. In: Quantum Physics. Report MIT-CTP/ 4628. NY: Cornell University, 2015. arXiv: 1412.6062

[23] Conway RW, Maxwell WL, Miller LW. Theory of Scheduling. Mineola, NY: Dover Publications; 2003. p. 304

[24] Leung JY-T, editor. Handbook of Scheduling: Algorithms, Models, and Performance Analysis. NY: Chapman & Hall/CRC; 2004. p. 1224

[25] Sheremet IA. Intelligent Software Environments for Computerized Information Processing Systems. Moscow: Nauka; 1994. p. 544. (In Russian)

[26] Wooldridge M. An Introduction to Multi-Agent Systems. Chichester, England: John Wiley & Sons; 2009. p. 460

[27] Salamon T. Design of Agent-Based Models. Repin-Zivolin: Brukner Publishing; 2011. p. 220