We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK
CITATION
INDEX
INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter

Multiset-Based Knowledge Representation for the Assessment and Optimization of Large-Scale Sociotechnical Systems

Igor Sheremet

Abstract

This chapter is dedicated to a new knowledge representation model, providing convergence of classical operations research and modern knowledge engineering. Kernel of the introduced model is the recursively generated multisets, selected according to the predefined restrictions and optimization criteria. Sets of multisets are described by the so-called multiset grammars (MGs), being projection of a conceptual background of well-known string-generating grammars on the multisets universum. Syntax and semantics of MGs and their practice-oriented development—unitary multiset grammars and metagrammars—are considered.

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

Large-scale sociotechnical systems (STS) usually have hierarchical structure, including personnel and various technical devices, which, in turn, consume various material, financial, information resources, as well as energy. As a result, they produce new resources (objects), which are delivered to other similar systems. Main features of such STS are large dimensionality and high volatility of their structures, equipment, consumed/produced objects, and at all, operation logics and dynamics [1–5].

Knowledge and data representation models, used in STS, provide comparatively easy and comfortable management of very large knowledge and data bases with dynamic structures and content [6–10]. These model bases are objects other than matrices, vectors, and graphs, traditionally used in operations research and systems analysis [11–14], and they are much more convenient for practical problem consideration. But, on the other hand, aforementioned models in general case do not incorporate strict theoretical background and fundamental algorithmics, compared with, for example, mathematical programming, which provides strictly optimal solutions for decision-makers. So, practically all decision-support software in the considered STS is based on various heuristics, which correctness and

adequacy are not proved usually in the mathematical sense. As a consequence, quality of the adopted decisions, based on such heuristics, in many cases may be far of optimal.

This chapter is dedicated to a primary survey of the developed knowledge representation model, providing convergence of classical operations research and modern knowledge engineering. This convergence creates new opportunities for complicated problem formalization and solution by integrating best features of mathematical programming (strict optimal solution search in solution space, defined by goal functions and boundary conditions) and constraint programming [15–17] (natural and easily updated top-down representation of logic of the decision-making in various situations). Kernel of the considered model is multisets (MS)—relatively long ago known and in the last 20 years intensively applied object of classical mathematics [18–29]. This background is generalized to the recursively generated, or, for short, recursive multisets (RMS) by introduction of so-called multiset grammars, or, again for short, multigrammars (MGs), which were described by the author in [30, 31]. Last, in turn, are peculiar "projection" of the conceptual basis of classical formal grammars by Chomsky [32, 33], operating strings of symbols, to the multisets universum in such a way, that MGs provide generation of one multiset from another and selection (filtration) of those, which satisfy necessary integral conditions: boundary restrictions and/or optimality criteria.

MGs may be considered as prolog-like constraint programming language for solution of problems in operations research and systems analysis areas. Taking into account relative novelty of the multigrammatical approach and absence of any substantial associations with mathematical constructions presented lower, we introduce main content of the chapter by short informal description of the main elements of this approach in Section 2. Basic formal definitions are presented in Section 3. Section 4 is dedicated to multiset grammars, while Section 5—to detailed consideration of the so-called unitary multigrammars (UMGs) and unitary multimetagrammars (UMMGs), which are main tool of the aforementioned problem formalization and solution.

2. Informal description

Let us consider a company, which consists of director, three departments, and one separate laboratory. This fact may be simply represented as follows:

$$company \rightarrow 1 \cdot director, 3 \cdot department, 1 \cdot laboratory. \tag{1}$$

In this notation, a whole structure of the company, detailed to employee positions, may be described in such a way:

$$department \rightarrow 1 \cdot head - department, 3 \cdot laboratory,$$
$$laboratory \;\; \rightarrow 1 \cdot head - laboratory, 2 \cdot analyst, 3 \cdot assistant. \tag{2}$$

This set of constructions is of the form:

$$a \rightarrow n_1 \cdot a_1, ..., n_m \cdot a_m \tag{3}$$

describes following set, created by multiplying and summarizing quantities of identical positions:

$$\{1 \cdot director, 3 \cdot head - department, 10 \cdot head - laboratory, 20 \cdot analyst, 30 \cdot assistant\}, \tag{4}$$

where $n_i \cdot a_i$ means there are n_i positions of type a_i in this company.

Let us join to the company structure knowledge about employees' month salary, represented in the same unified manner:

$$
\begin{aligned}
director &\rightarrow 10000 \cdot eur, \\
head\text{-}department &\rightarrow 5000 \cdot eur, \\
head\text{-}laboratory &\rightarrow 3000 \cdot eur, \\
analyst &\rightarrow 1500 \cdot eur, \\
assistant &\rightarrow 500 \cdot eur.
\end{aligned} \tag{5}
$$

After applying to the joined set of constructions just the same multiplying-summarizing procedure, we may obtain resulting set containing the only element {100,000·*eur*}, which defines company's total financial resource, necessary for employees' provision a month.

Presented knowledge representation concerns systems analysis, that is, obtaining integral parameters of the system given its structure and local parameters.

Consider more sophisticated task-relating systems design and concerning development of company structure given its integral parameters. Goal is to determine rational quantity of departments and laboratories in the department, as well as quantities of analysts and assistants in one laboratory. Total salary is no more than 120,000 eur, quantity of analysts in one laboratory may be from 1 to 3, while corresponding quantity of assistants may be from 2 to 6. Total quantity of employees must be maximal. There may be three different variants of company structure: (1) three departments and one laboratory; (2) two departments and three laboratories; and (3) four departments. Corresponding set of constructions is as follows:

$$
company \rightarrow 1 \cdot director, 3 \cdot department, 1 \cdot laboratory, \tag{6}
$$

$$
company \rightarrow 1 \cdot director, 2 \cdot department, 1 \cdot laboratory, \tag{7}
$$

$$
company \rightarrow 1 \cdot director, 4 \cdot department, \tag{8}
$$

$$
department \rightarrow 1 \cdot head\text{-}department, m \cdot laboratory, \tag{9}
$$

$$
laboratory \rightarrow 1 \cdot head\text{-}laboratory, n \cdot analyst, l \cdot assistant. \tag{10}
$$

Constructions, defining employees' salary, and other aforementioned restrictions are as follows (for definiteness, let us take that quantity of laboratories in one department does not exceed five):

$$
director \rightarrow 1 \cdot employee, 10000 \cdot eur, \tag{11}
$$

$$
head\text{-}department \rightarrow 1 \cdot employee, \ \ 5000 \cdot eur, \tag{12}
$$

$$
head\text{-}laboratory \rightarrow 1 \cdot employee, \ \ 3000 \cdot eur, \tag{13}
$$

$$
analyst \rightarrow 1 \cdot employee, \ \ 1500 \cdot eur, \tag{14}
$$

$$
assistant \rightarrow 1 \cdot employee, \ \ 500 \cdot eur, \tag{15}
$$

$$
employee = \max, \tag{16}
$$

$$
eur \leq 120000, \tag{17}
$$

$$
1 \leq m \leq 5, \tag{18}
$$

$$
1 \leq n \leq 3, \tag{19}
$$

$$
1 \leq l \leq 6. \tag{20}
$$

As seen, along with already introduced "detailing" constructions, there are additional constructions, defining sets of values of variables, having places in the first ones, as well as conditions, determining optimization criterion (there may be several such criteria), and bounds of quantities of some objects in the resulting sets. Evidently, due to presence of alternatives in the description of company structure (there are three such alternatives) and variables in some of "detailing" constructions, there may be more than one resulting set like Eq. (4). These sets are of the form $\{x \cdot employee, y \cdot eur\}$, where x is the quantity of employees, while y—total salary, corresponding to this variant. Conditions (16)–(20) provide selection of those sets, which satisfy them in the described sense. In general, Eqs. (16)–(20) may be interpreted as a query, determining subset of all possible variants, described by Eqs. (6)–(15).

To "mark" "detailing" constructions, used while resulting set creation, one can add to their "bodies" elements like $1 \cdot variant\text{-}i$, for example,

$$company \rightarrow 1 \cdot variant\text{-}1, 1 \cdot director, 3 \cdot department, 1 \cdot laboratory, \tag{21}$$

$$company \rightarrow 1 \cdot variant\text{-}2, 1 \cdot director, 2 \cdot department, 3 \cdot laboratory, \tag{22}$$

$$company \rightarrow 1 \cdot variant\text{-}3, 1 \cdot director, 4 \cdot department. \tag{23}$$

If so, then resulting sets will be of the form:

$$\{1 \cdot variant\text{-}i, x \cdot employee, y \cdot eur\}. \tag{24}$$

To implant to these sets values of variables, it is sufficient to represent them in resulting sets in "usual" form $j \cdot v$, where v is variable and j is its value, so considered example will lead us to sets like:

$$\{1 \cdot variant\text{-}i, x \cdot employee, y \cdot eur, i \cdot m, j \cdot n, k \cdot l\}. \tag{25}$$

As seen, shortly introduced by this example knowledge and query representation language, being easy to understand and to use, allows formalization of multicriterial optimization problems, for years associated with mathematical programming. On the other hand, "detailing" constructions have form of productions (rules), far and wide used in knowledge engineering and being common background of prolog-like declarative (nonprocedural) knowledge representation [34–36]. As will be shown lower, such constructions may be used not only for structuring, but in many other cases, enabling description of various systems behavior and interaction, as well as their mutual impacts. For such reasons, this informally described technique is taken as a basis for the description of the developed mathematical toolkit considered thoroughly in the following sections.

3. Basic operations on multisets

Classical set theory is based on the concept of set as unordered assembly of elements, different from one another. Theory of multisets assumes presence of equal ("indistinguishable") elements:

$$v = \left\{ \underbrace{a_1, ..., a_1}_{n_1 \ times}, ..., \underbrace{a_i, ..., a_i}_{n_i \ times}, ..., \underbrace{a_m, ..., a_m}_{n_m \ times} \right\} \tag{26}$$

Expression (26) is recorded as:

$$v = \{n_1 \cdot a_1, ..., n_m \cdot a_m\}, \tag{27}$$

where v is called multiset, $a_1, ..., a_m$—objects, $n_1, ..., n_m$—multiplicities of these objects, and $n_1 \cdot a_1, ..., n_m \cdot a_m$—multiobjects. Following Eq. (27), one may consider v as set of multiobjects; also, from substantial point of view, set $\{a_1, ..., a_m\}$ and multiset $\{1 \cdot a_1, ..., 1 \cdot a_m\}$ are equivalent. Empty multiset, as well as empty set, is designated as $\{\varnothing\}$. Multiplicity of object may be zero, what is equivalent to absence of this object in the multiset:

$$\{n_1 \cdot a_1, ..., n_m \cdot a_m, 0 \cdot a_{m+1}\} = \{n_1 \cdot a_1, ..., n_m \cdot a_m\}. \tag{28}$$

Fact that object a or multiobject $n \cdot a$ belongs to multiset v ("enters v") is designated by one and the same symbol $\in$: $a \in v$, $n \cdot a \in v$. Set $\beta(v) = \{a | n \cdot a \in v\}$ is called basis of multiset v.

There are five main operations on multisets, used lower: join, intersection, addition, subtraction, and multiplication by constant [26, 27].

Consider two multisets:

$$\begin{aligned} v &= \{n_1 \cdot a_1, ..., n_m \cdot a_m\}, \\ v' &= \{n'_1 \cdot a'_1, ..., n'_{m'} \cdot a'_{m'}\}. \end{aligned} \tag{29}$$

Result of their join (recorded as $\cup$) is multiset.

$$
\begin{aligned}
v \bigcup v' = &\left(\bigcup_{\substack{a \in \{a_1, ..., a_m\} \cap \{a'_1, ..., a'_{m'}\} \\ n \cdot a \in v \\ n' \cdot a \in v}} \{\max(n, n') \cdot a\} \right) \cup \\
&\cup \left(\bigcup_{\substack{a \in \{a_1, ..., a'_{m'}\} - \{a'_1, ..., a'_{m'}\} \\ n \cdot a \in v}} \{n \cdot a\} \right) \cup \\
&\cup \left(\bigcup_{\substack{a' \in \{a'_1, ..., a'_m\} - \{a_1, ..., a_m\} \\ n' \cdot a' \in v}} \{n' \cdot a'\} \right),
\end{aligned}
\tag{30}
$$

where $\cup$, $\cap$ and $-$ designate operations of set-theoretical join, intersection, and subtraction of two sets correspondingly, while $\bigcup$ designates operation of set-theoretical join of sets determined by underwritten conditions.

Result of v, v' multisets intersection (recorded as $\bigcap$) is multiset.

$$
v \bigcap v' = \left(\bigcup_{\substack{a \in \{a_1, ..., a_m\} \cap \{a'_1, ..., a'_{m'}\} \\ n \cdot a \in v \\ n' \cdot a \in v}} \{\min(n, n') \cdot a\} \right). \tag{31}
$$

Result of v, v' multisets addition (recorded as bold $+$) is multiset.

$$v+v' = \left(\bigcup_{\substack{a \in \{a_1, ..., a_m\} \cap \{a'_1, ..., a'_{m'}\} \\ n \cdot a \in v \\ n' \cdot a \in v}} \{(n+n') \cdot a\} \right) \cup$$

$$\cup \left(\bigcup_{\substack{a \in \{a_1, ..., a_m\} - \{a'_1, ..., a'_{m'}\} \\ n \cdot a \in v}} \{n \cdot a\} \right) \cup \qquad (32)$$

$$\cup \left(\bigcup_{\substack{a' \in \{a'_1, ..., a'_{m'}\} - \{a_1, ..., a_m\} \\ n' \cdot a \in v'}} \{n' \cdot a\} \right).$$

Result of v' multiset subtraction from v multiset (recorded as bold $-$) is multiset.

$$v-v' = \left(\bigcup_{\substack{a \in \{a_1, ..., a_m\} \cap \{a'_1, ..., a'_{m'}\} \\ n \cdot a \in v \\ n' \cdot a \in v' \\ n > n'}} \{(n-n') \cdot a\} \right) \cup \qquad (33)$$

$$\cup \left(\bigcup_{\substack{a \in \{a_1, ..., a_m\} - \{a'_1, ..., a'_{m'}\} \\ n \cdot a \in v}} \{n \cdot a\} \right).$$

At last, result of v multiset multiplication by integer number n (recorded as $v * n$) is multiset.

$$v * n = \{(n \times n_1) \cdot a_1, ..., (n \times n_m) \cdot a_m\} \qquad (34)$$

(here integers' usual multiplication is recorded as $\times$)

There are also two basic relations on multisets: inclusion ($\subseteq$) and strict inclusion ($\subset$).

Multiset v is included to multiset v', that is, $v \subseteq v'$, if

$$(\forall n \cdot a \in v)[((\exists n' \cdot a \in v') \, n \leq n')], \qquad (35)$$

and multiset v is strictly included to multiset v', that is, $v \subset v'$, if $v \subseteq v'$ & $v \neq v'$.

Example 1. Let $v_1 = \{3 \bullet analyst, 2 \bullet assistant\}$, $v_2 = \{4 \bullet assistant, 1 \bullet director\}$, then

$$v_1 \cup v_2 = \{3 \cdot analyst, 4 \cdot assistant, 1 \cdot director\},$$

$$v_1 \cap v_2 = \{2 \cdot assistant\},$$

$$v_1 + v_2 = \{3 \cdot analyst, 6 \cdot assistant, 1 \cdot director\},$$

$$v_2 - v_1 = \{2 \cdot assistant\},$$

$$v_1 * 2 = \{6 \cdot analyst, 4 \cdot assistant\},$$

$$\{1 \cdot analyst, 2 \cdot assistant\} \subset v_1,$$

$$\{4 \cdot assistant, 1 \cdot director\} \subseteq v_2. \quad \blacksquare$$

All defined operations are known from widespread sources (e.g., aforementioned [26, 27]). At the same time, filtering operations, defined lower, operate sets of multisets (SMS), creating subsets of these sets by selection of multisets, which satisfy some conditions, being operands of these operations.

There are two types of conditions: boundary and optimizing.

Boundary condition may be elementary or concatenated (for short, "chain"). Elementary boundary condition (EBC) may have one of the following forms:

$$n\rho a, \tag{36}$$

$$a\rho n, \tag{37}$$

$$a\rho a', \tag{38}$$

where a and a' are the objects, n is the integer number, and $\rho \in \{<, =, \leq\}$. Chain boundary condition (CBC) is constructed from elementary by writing them sequentially:

$$e_1\rho_1 e_2\rho_2 ... e_i\rho_i e_{i+1} ... e_m\rho_m e_{m+1}, \tag{39}$$

where $e_1, ..., e_{m+1}$ are the objects or nonnegative integers, while $\rho_1, ..., \rho_m$ are the symbols of relations $(<, =, \leq)$.

EBC semantics is following. Let V be set of multisets, and $v \in V$. Multiset v satisfies EBC $\overline{n}\rho a$, if $n \cdot a \in V$, and $\overline{n}\rho n$ is true. Similarly, v satisfies EBC $a\rho\overline{n}$, if $\overline{n}\rho n$ is also true. At last, v satisfies EBC $a\rho a'$, if $n \cdot a \in v$, $n' \cdot a' \in v$, and $n\rho n'$ is true. There is one addition to all listed definitions, concerning particular case, when $n \cdot a \notin v$ $(n' \cdot a \notin v)$, which is equivalent to $n = 0$ $(n' = 0)$.

CBC semantics is defined as follows. CBC (39) is replaced by CBC sequence

$$e_1\rho_1 e_2, e_2\rho_2 e_3, ..., e_i\rho_i e_{i+1}, ..., e_m\rho_m e_{m+1}, \tag{40}$$

and $v \in V$ is considered satisfying CBC (39), if it satisfies all EBC having place in Eq. (40).

Result of application of boundary condition b to SMS V is recorded as $V \downarrow b$.

Example 2. Let $V = \{v_1, v_2\}$, where $v_1 = \{5 \cdot analyst, 3 \cdot assistant, 1 \cdot director\}$, $v_2 = \{2 \cdot assistant, 4 \cdot director, 3 \cdot employee\}$, and boundary conditions are $2 \leq analyst \leq 4$, $assistant < employee$, $1 \leq director \leq assistant \leq 3$, and $analyst = assistant < 5$. **Table 1** contains result of application of listed boundary conditions to V.∎

Optimizing condition has form $a = opt$, where a is the object, and $opt \in \{min, max\}$. Semantics of this construction is following. Multiset $v \in V$ satisfies condition $a = min$, if for every $v' \in V$, such that $v \neq v'$, multiplicity n in multiobject $n \cdot a \in v$ is not greater, than multiplicity n' in multiobject $n' \cdot a \in v'$, that is, $n \leq n'$. Similarly, $v \in V$ satisfies condition $a = max$, if for every $v' \in V$, such that $v \neq v'$, multiplicity n in multiobject $n \cdot a \in v$ is not less, than multiplicity n' in

condition	V ↓ condition
$2 \leq analyst \leq 4$	$\{v_2\}$
$assistant < employee$	$\{v_2\}$
$1 \leq director \leq assistant \leq 3$	$\{v_1\}$
$analyst = assistant < 5$	$\{\varnothing\}$

Table 1.
Results of application of boundary conditions.

multiobject $n' \cdot a \in v'$, that is, $n \geq n'$. If $a \notin v$ ($a' \notin v$), we consider $n \cdot a \in v$ ($n' \cdot a \in v$), where $n = 0$ ($n' = 0$).

Filter is join of boundary $F_{\leq}$ and optimizing F_{opt} subfilters:

$$F = F_{\leq} \cup F_{opt}, \tag{41}$$

where $F_{\leq}$ is set of boundary conditions, and F_{opt} is set of optimizing conditions. Result of filtration of set of multisets V by filter F is denoted as $V \downarrow F$ and is defined by expression

$$V \downarrow F = (V \downarrow F_{\leq}) \downarrow F_{opt} \tag{42}$$

where

$$F_{\leq} = \{c_1, \dots, c_k\}, \tag{43}$$

$$F_{opt} = \{opt_1, \dots, opt_l\}, \tag{44}$$

$$V \downarrow F_{\leq} = \bigcap_{i=1}^{k} (V \downarrow c_i) = V', \tag{45}$$

$$V' \downarrow F_{opt} = \bigcap_{j=1}^{l} \left(V' \downarrow opt_j \right), \tag{46}$$

and $c_1, \dots, c_k$ are EBC. As seen, set V is filtered by boundary conditions, so there are selected multisets, satisfying all of these conditions, and intermediate result V' is then filtered by optimizing conditions, so, that multisets, satisfying all of them, are included to the final result.

Example 3. Consider set $V = \{v_1, v_2, v_3, v_4\}$, where

$$v_1 = \{3 \cdot analyst, 2 \cdot assistant, 2 \cdot employee\},$$
$$v_2 = \{6 \cdot assistant, 2 \cdot director\},$$
$$v_3 = \{1 \cdot analyst, 3 \cdot assistant, 5 \cdot director, 2 \cdot employee\},$$
$$v_4 = \{1 \cdot analyst, 2 \cdot assistant, 2 \cdot employee\}.$$

Let $F = \{1 \leq analyst \leq 3, 2 \leq director \leq employee, analyst = \min, assistant = \max\}$. Then, according to Eqs. (41)–(46),

$$V \downarrow F = (V \downarrow F_{\leq}) \downarrow F_{opt},$$

where

$$F_{\leq} = \{1 \leq analyst \leq 3, 2 \leq director \leq employee\},$$
$$F_{opt} = \{assistant = \min, employee = \max\}.$$

Filtration is performed as follows:

$$V \downarrow \{1 \leq analyst \leq 3\} = \{v_1, v_3, v_4\},$$
$$V \downarrow \{2 \leq director \leq 4\} = \{v_1, v_2, v_4\},$$
$$V \downarrow F_{\leq} = \{v_1, v_4\},$$
$$\{v_1, v_4\} \downarrow \{assistant = \min\} = \{v_1\},$$
$$\{v_1, v_4\} \downarrow \{employee = \max\} = \{v_1, v_4\},$$
$$V \downarrow F = \{v_1\} \cap \{v_1, v_4\} = \{v_1\}. \quad \blacksquare$$

Due to commutativity of set-theoretic join and intersection operations, filtration inside subfilters may be executed in the arbitrary order.

4. Multiset grammars

As mentioned higher, multiset grammars are tool, providing generation of one multisets from another, or, what is the same, generation sets of multisets.

By analogy with classical grammars, operating strings of symbols [32, 33], we shall define multigrammar as a couple.

$$S = \langle v_0, R \rangle, \tag{47}$$

where v_0 is a multiset called kernel, while R, called scheme, is finite set of the so-called rules, which are used for generation of new multisets from already generated. Rule has the form:

$$v \rightarrow v', \tag{48}$$

where v (left part of the rule) and v' (right part of the rule) are multisets, and $v \neq \{\varnothing\}$. Semantics of rule is as follows. Let $\bar{v}$ be multiset; with that we shall speak, that rule (48) is applicable to $\bar{v}$, if $v \subseteq \bar{v}$, and result of its application is a multiset.

$$\bar{v}' = \bar{v} - v + v'. \tag{49}$$

Speaking informally, if $\bar{v}$ includes v, then the last is replaced by v'. Application of rule $r \in R$ to multiset $\bar{v}$ is denoted as $\bar{v} \overset{r}{\Rightarrow} \bar{v}'$, and any sequence $\bar{v} \overset{r}{\Rightarrow} ... \overset{r'}{\Rightarrow} \bar{v}'$ is called generation chain.

Set of multisets, defined by MGs $S = \langle v_0, R \rangle$, is denoted as V_S. Iterative representation of MG semantics, that is, SMS V_S generation by application of MG S, is the following:

$$V_{(0)} = \{v_0\}, \tag{50}$$

$$V_{(i+1)} = V_{(i)} \cup \left(\bigcup_{\bar{v} \in V_{(i)}} \bigcup_{r \in R} \pi(\bar{v}, r) \right), \tag{51}$$

$$V_S = V_{(\infty)}, \tag{52}$$

where

$$\pi(\bar{v}, v \rightarrow v') = \begin{cases} \{\bar{v} - v + v'\}, \text{ if } v \subseteq \bar{v}, \\ \{\varnothing\} \text{ otherwise.} \end{cases} \tag{53}$$

As seen, function (53) implements application of rule $v \rightarrow v'$ to multiset $\bar{v}$ as described higher. As a result of $i + 1$-th step of generation, new SMS is formed by application of all rules $r \in R$ to all multisets $\bar{v} \in V_{(i)}$, and it is joined to SMS $V_{(i)}$. If multiset $\bar{v'}$ is generated from multiset $\bar{v}$ by some sequence of such steps, it is denoted as $\bar{v} \overset{*}{\Rightarrow} \bar{v'}$.

V_S is fixed point of the described process, that is, $V_S = V_{(i)}$, where $i \rightarrow \infty$. If for some finite i $V_{(i)} = V_{(i+1)}$, then $V_S = V_{(i)}$, and V_S is finite. In the introduced notation,

$$V_S = \left\{ v | v_0 \overset{*}{\Rightarrow} v \right\}. \tag{54}$$

V_S includes subset $\overline{V}_s \subseteq V_s$ of the so-called terminal multisets (TMS) $v \in \overline{V}_s$ such that $\pi(v, r) = \{\varnothing\}$ for all $r \in R$, that is, no one multiset may be generated from terminal multiset. Set $\overline{V}_S$ is called final; final set consists of terminal multisets only.

Example 4. Let $S = \langle v_0, R \rangle$, where $v_0 = \{3 \cdot eur, 4 \cdot usd\}$,
$R = \{r_1, r_2\}$, where
r_1 is $\{2 \cdot eur, 1 \cdot usd\} \rightarrow \{1 \cdot eur, 1 \cdot gbp\}$,
r_2 is $\{1 \cdot eur, 2 \cdot usd\} \rightarrow \{1 \cdot eur, 2 \cdot gbp\}$.
As seen,

$$v_0 = \{3 \cdot eur, 4 \cdot usd\} \overset{r_1}{\Rightarrow} \{2 \cdot eur, 3 \cdot usd, 1 \cdot gbp\} \overset{r_1}{\Rightarrow} \{1 \cdot eur, 2 \cdot usd, 2 \cdot gbp\} \overset{r_2}{\Rightarrow} \{1 \cdot usd, 4 \cdot gbp\},$$

$$\{2 \cdot eur, 3 \cdot usd, 1 \cdot gbp\} \overset{r_2}{\Rightarrow} \{1 \cdot a_1, 2 \cdot usd, 3 \cdot gbp\} \overset{r_2}{\Rightarrow} \{1 \cdot usd, 5 \cdot gbp\},$$

$$\{3 \cdot eur, 4 \cdot usd\} \overset{r_2}{\Rightarrow} \{2 \cdot eur, 3 \cdot usd, 2 \cdot gbp\} \overset{r_1}{\Rightarrow} \{1 \cdot eur, 2 \cdot usd, 3 \cdot gbp\} \overset{r_2}{\Rightarrow} \{1 \cdot usd, 5 \cdot gbp\},$$

$$\{2 \cdot eur, 3 \cdot usd, 2 \cdot gbp\} \overset{r_2}{\Rightarrow} \{1 \cdot eur, 2 \cdot usd, 4 \cdot gbp\} \overset{r_2}{\Rightarrow} \{1 \cdot usd, 6 \cdot gbp\}$$

(for short, identical parts of different generation chains are omitted). So

$$\overline{V}_s = \{\{1 \cdot eur, 4 \cdot gbp\}, \{1 \cdot usd, 5 \cdot gbp\}, \{1 \cdot usd, 6 \cdot gbp\}\} \quad \blacksquare.$$

By analogy with classical string-generating grammars, multigrammars may be context-sensitive and context-free (CF). In the last one, left parts of all rules have form $\{1 \cdot a\}$, while in the first, there are no any limitations on both parts of rules, excluding, that left part must be nonempty multiset.

5. Unitary multiset grammars and metagrammars

Start point for unitary multigrammars (UMGs), developed on the considered basis, is simplified representation of CF rules: instead of

$$\{1 \cdot a\} \rightarrow \{n_1 \cdot a_1, ..., n_m \cdot a_m\} \tag{55}$$

they are written as:

$$a \rightarrow n_1 \cdot a_1, ..., n_m \cdot a_m. \tag{56}$$

Construction (56) is called unitary rule (UR), object a—its head, and unordered sequence (list) $n_1 \cdot a_1, ..., n_m \cdot a_m$—its body.

Let us consider UMG formal definition and illustrating example. Unitary multigrammar is couple $S = \langle a_0, R \rangle$, where a is the so-called title object, and R, as in multigrammars, is scheme—set of unitary rules (56).

Iterative representation of UMG semantics, i.e., generation of SMS V_S, where $S = \langle a_0, R \rangle$, is following:

$$V_{(0)} = \{1 \cdot a_0\}, \tag{57}$$

$$V_{(i+1)} = V_{(i)} \cup \bigcup_{\overline{v} \in V_{(i)}} \bigcup_{r \in R} \{\overline{\pi}(\overline{v}, r)\}, \tag{58}$$

$$V_S = V_{(\infty)}, \tag{59}$$

$$\overline{V}_S = \{\overline{v} | \overline{v} \in V_S \ \& \ \beta(\overline{v}) \subseteq \overline{A}_s\}, \tag{60}$$

where

$$\overline{\pi}(\overline{v}, \langle a \to n_1 \cdot a_1, ..., n_m \cdot a_m \rangle) = \begin{cases} \overline{v} - \{n \cdot a\} + n * \{n_1 \cdot a_1, ..., n_m \cdot a_m\}, \text{ if } n \cdot a \in \overline{v}, \\ \{\varnothing\} \text{ otherwise.} \end{cases}$$

$$\tag{61}$$

Here, $\overline{A}_s$ is set of the so-called terminal objects, such that $a \in \overline{A}_s$, if and only if R does not include URs, which head is a (i.e., a has place only in the UR bodies). $\overline{A}_s$ is subset of set A_s of all objects, having places in scheme R of UMG S. Multiset, generated by UMG S, all objects of which are terminal, is also called terminal multiset (as seen, this notion of TMS does not contradict to the defined higher regarding MGs). In Eq. (61), UR $a \to n_1 \cdot a_1, \cdots, n_m \cdot a_m$ is written in the angle brackets for unambiguity.

As seen, Eq. (59) defines V_S—set of all multisets, generated by UMG S,—while Eq. (60) by condition $\beta(\overline{v}) \subseteq \overline{A}_s$ provides selection of $\overline{V}_S$—set of terminal multisets (STMS)—from V_S.

Example 5. Consider unitary multigrammar $S = \langle company, R \rangle$, where R includes following unitary rules:

$$company \to 3 \cdot group, 2 \cdot analyst,$$
$$company \to 3 \cdot analyst,$$
$$group \to 1 \cdot analyst,$$
$$group \to 2 \cdot analyst.$$

According to Eqs. (57)–(61),

$$V_S = \{\{1 \cdot company\}, \{3 \cdot group, 2 \cdot analyst\}, \{3 \cdot analyst\}, \{5 \cdot analyst\}, \{8 \cdot analyst\}\},$$
$$\overline{V}_S = \{\{3 \cdot analyst\}, \{5 \cdot analyst\}, \{8 \cdot analyst\}\}. \blacksquare$$

Filtering unitary multigrammars (FUMGs) are UMG generalization, providing generation of terminal multisets and selection those of them, which satisfy conditions, assembled to filters, which were described higher in Section 3.

FUMGs are triple $S = \langle a_0, R, F \rangle$, where a_0, R, and F have the same sense, as above, and set of terminal multisets, generated by S, is defined as follows:

$$\overline{V}_S = \overline{V}_{\langle a_0, R \rangle} \downarrow F, \tag{62}$$

that is, set of terminal multisets, generated by S, is result of filtering STMS, generated by UMGs $\langle a_0, R \rangle$, by filter F.

Example 6. Let $S = \langle company, R, F \rangle$, where R is as in Example 5, while $F = \{analyst > 3, analyst = \min\}$. Then, according to Eqs. (57)–(62),

$$\overline{V}_S = (\{\{3 \cdot analyst\}, \{5 \cdot a_2\}analyst, \{8 \cdot analyst\}\} \downarrow \{analyst > 3\}) \downarrow \{analyst = \min\}$$
$$= (\{\{5 \cdot analyst\}, \{8 \cdot analyst\}\}) \downarrow \{analyst = \min\} = \{\{5 \cdot analyst\}\}. \blacksquare$$

Filtering unitary multigrammars are, in turn, basis for unitary multiset metagrammars, or, for short, multimetagrammars, which are considered at all the rest part of the chapter and are main toolkit for the description and solution of the optimization problems, mentioned in the introduction.

Unitary multimetagrammar S is, as higher, triple $\langle a_0, R, F \rangle$, where a_0 is the title object, and R is the scheme, containing unitary rules and so-called unitary metarules (UMR), while F is the filter. Consider UMMG syntax and semantics.

Unitary metarule has the form:

$$a \rightarrow \mu_1 \cdot a_1, ..., \mu_m \cdot a_m, \tag{63}$$

where μ_i is the positive integer number, as in UMGs/FUMGs, or variable $\gamma \in \Gamma$, where Γ is the universum of variables. When μ_i is $\gamma_i \in \Gamma$, then it is called multiplicity-variable (MV). As seen, unitary rule is the simplest particular case of unitary metarule with all multiplicities $\mu_1, ..., \mu_m$ being constants. As in URs, object a in Eq. (55) is called head, while $\mu_1 \cdot a_1, ..., \mu_m \cdot a_m$—body of metarule.

Filter F is set of conditions, which may be of the following forms:

$$n \leq a \leq n', \tag{64}$$

$$a = opt, \tag{65}$$

$$n \leq \gamma \leq n', \tag{66}$$

where $opt \in \{\max, \min\}$. As seen, boundary condition (64) and optimizing condition (65) are the same, as in FUMG filters, while boundary condition (65), called variable declaration, defines set of values (domain) of variable γ and is denoted lower as $N(\gamma)$. If F includes subfilter $F_\Gamma = \{n_1 \leq \gamma_1 \leq n'_1, ..., n_l \leq \gamma_l \leq n'_l\}$, containing boundary conditions of form (66), then every combination of variable values $\overline{n}_1 \in N(\gamma_1), ..., \overline{n}_l \in N(\gamma_l)$ provides creation of one unitary multigrammar by substitution of $\overline{n}_1, ..., \overline{n}_l$ to all unitary metarules, having places in scheme R, instead of multiplicities-variables being in their bodies; unitary rules, already having place in R, are transferred to new scheme, denoted $R \circ \langle \overline{n}_1, ..., \overline{n}_l \rangle$, without any transformations. Every such UMGs generates set of terminal multisets, after what all these STMS are joined, and resulting set is filtered by filter $F = F - F_\Gamma$, containing all "FUMG-like" conditions (From the described, it is obvious nature of "multimetagrammar" notion—in mathematical logic, or "metamathematics," "metalanguage" is language, used for description of another language, so "multimetagrammar" is "unitary-like" multigrammar, used for description of other unitary multigrammars by means of unitary metarules, variables-multiplicities, and boundary conditions, defining their domains.). As may be seen from this informal description, UMMGs are simple unified tool for compact representation of sets of FUMGs (for practically valuable problems, containing very large numbers of elements—millions and greater).

Coming back to Section 2, one can see that Eqs. (6)–(20) are set of elements of unitary metamultigrammar: Eqs. (6)–(8) and Eqs. (11)–(15) are unitary rules, Eqs. (9)–(10) are unitary metarules, Eq. (16) is optimizing condition, Eq. (17) is boundary condition, while Eqs. (18)–(20) are variable declarations. As seen,

$$N(m) = \{1, 2, 3, 4, 5\}, \tag{67}$$

$$N(n) = \{1, 2, 3\}, \tag{68}$$

$$N(l) = \{1, 2, 3, 4, 5, 6\}, \tag{69}$$

so, this one UMMG, consisting of 15 lines, replaces $5 \times 3 \times 5 = 75$ filtering unitary multigrammars, each scheme consisting of 10 lines.

Let us now give strict definition of unitary multimetagrammar notion. UMMG $S = \langle a_0, R, F \rangle$ defines set of terminal multisets $\overline{V}_S$ in such a way:

$$\overline{V}_S = \left(\bigcup_{\overline{S} \in S^*} \overline{V}_{\overline{S}} \right) \downarrow F, \tag{70}$$

$$S^* = \bigcup_{\gamma_1 \in n_1}^{n_1'} \cdots \bigcup_{\gamma_l \in n_l}^{n_l'} \{ \langle a_0, R \circ \langle \gamma_1, ..., \gamma_l \rangle \rangle \}, \tag{71}$$

$$R \circ \langle \overline{n}_1, ..., \overline{n}_l \rangle = \{ r \circ \langle \overline{n}_1, ..., \overline{n}_l \rangle | r \in R \}, \tag{72}$$

$$F = F - F_\Gamma, \tag{73}$$

$$F_\Gamma = \bigcup_{i=1}^{l} \{ n_i \leq \gamma_i \leq n_i' \}, \tag{74}$$

and, at last, if r is $a \leftarrow \mu_1 \cdot a_1, ..., \mu_m \cdot a_m$, then $r \circ \langle \overline{n}_1, ..., \overline{n}_l \rangle$ is unitary rule.

$$a \leftarrow (\mu_1 \circ \langle \overline{n}_1, ..., \overline{n}_l \rangle) \cdot a_1, ..., (\mu_m \circ \langle \overline{n}_1, ..., \overline{n}_l \rangle) \cdot a_m, \tag{75}$$

where

$$\mu_i \circ \langle \overline{n}_1, ..., \overline{n}_l \rangle = \begin{cases} \mu_i, & \text{if } \mu_i \in N, \\ \overline{n}_j, & \text{if } \mu_i \text{ is } \gamma_j \in \Gamma. \end{cases} \tag{76}$$

As seen, according to Eqs. (75) and (76), all multiplicities-variables of unitary metarule $a \leftarrow \mu_1 \cdot a_1, ..., \mu_m \cdot a_m$ are replaced by their corresponding values from the tuple $\langle \overline{n}_1, ..., \overline{n}_l \rangle$, while all multiplicities-constants (elements of positive integer numbers set N) remain unchanged. Evidently, if all $\mu_1, ..., \mu_m$ are constants, that is, if unitary metarule is UR, it remains unchanged.

Let us note, that multiplicities-variables area of actuality is whole UMMG scheme, that is, if there are $n > 1$ occurrences of one and the same variable γ in different unitary metarules (and, of course, in one and the same unitary metarule), they all are substituted by one and the same value from the applied sequence $\langle \overline{n}_1, ..., \overline{n}_l \rangle$.

Example 7. Let us consider UMMG $S = <company, R, F>$, where scheme R contains following three unitary metarules:

$$company \rightarrow 2 \cdot group, \gamma_1 \cdot analyst,$$
$$group \rightarrow 3 \cdot analyst, \gamma_2 \cdot assistant,$$
$$group \rightarrow \gamma_1 \cdot analyst, \gamma_2 \cdot assistant,$$

and filter F includes following conditions, the first being boundary, the second—optimizing, while the last two—variable declarations:

$$2 \leq analyst \leq 6,$$
$$assistant = \min,$$
$$0 \leq \gamma_1 \leq 1,$$
$$2 \leq \gamma_2 \leq 3.$$

According to Eqs. (70)–(76), this UMMG defines four UMGs:

$$S_{0,2} = <company, R\circ <0, 2>>,$$
$$S_{0,3} = <company, R\circ <0, 3>>,$$
$$S_{1,2} = <company, R\circ <1, 2>>,$$
$$S_{1,3} = <company, R\circ <1, 3>>,$$

where

$R\circ <0, 2> = \{ <company \rightarrow 2\cdot group>,$
$\qquad <group \rightarrow 3\cdot analyst, 2\cdot assistant>, <group_1 \rightarrow 2\cdot assistant>\},$
$R\circ <0, 3> = \{ <company \rightarrow 2\cdot group>,$
$\qquad <group \rightarrow 3\cdot analyst, 3\cdot assistant>, <group \rightarrow 3\cdot assistant>\},$
$R\circ <1, 2> = \{ <company \rightarrow 2\cdot group, 1\cdot analyst>,$
$\qquad <group \rightarrow 3\cdot analyst, 2\cdot assistant>, <group \rightarrow 1\cdot analyst, 2\cdot assistant>\},$
$R\circ <1, 3> = \{ <company \rightarrow 2\cdot group, 1\cdot analyst>,$
$\qquad <group \rightarrow 3\cdot analyst, 3\cdot assistant>, <group \rightarrow 1\cdot analyst, 3\cdot assistant>\},$

(URs are represented in angle brackets for unambiguity). These UMGs define, respectively, following sets of terminal multisets:

$$\overline{V}_{S_{0,2}} = \{\{6\cdot analyst, 4\cdot assistant\}, \{4\cdot assistant\}\},$$
$$\overline{V}_{S_{0,3}} = \{\{6\cdot analyst, 6\cdot assistant\}, \{6\cdot assistant\}\},$$
$$\overline{V}_{S_{1,2}} = \{\{7\cdot analyst, 4\cdot assistant\}, \{3\cdot analyst, 4\cdot assistant\}\},$$
$$\overline{V}_{S_{1,3}} = \{\{7\cdot analyst, 6\cdot assistant\}, \{3\cdot analyst, 6\cdot assistant\}\},$$
$$\overline{V}_S = \left(\overline{V}_{S_{0,2}} \cup \overline{V}_{S_{0,3}} \cup \overline{V}_{S_{1,2}} \cup \overline{V}_{S_{1,3}}\right) \downarrow (2 \leq analyst \leq 6) \downarrow \{assistant = \min\}$$
$$= \{6\cdot analyst, 4\cdot assistant\}, \{6\cdot analyst, 6\cdot assistant\}, \{6\cdot assistant\},$$
$$\{3\cdot analyst, 4\cdot assistant\}, \{3\cdot analyst, 6\cdot assistant\} \downarrow \{assistant = \min\}$$
$$= \{\{6\cdot analyst, 4\cdot assistant\}, \{3\cdot analyst, 4\cdot assistant\}\}. \quad \blacksquare$$

As may be seen, Eqs. (64)–(66) define boundary conditions, concerning objects and variables, and optimizing conditions, concerning only objects, that is why from both theoretical and practical points of view, it is reasonable to extend UMMG filters by optimizing conditions, relating variables. By analogy with Eq. (65), such conditions will have the form:
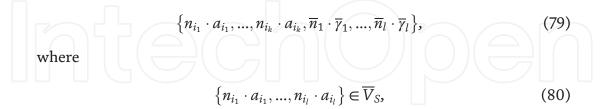
$$\gamma = opt. \tag{77}$$

This form defines optimality of the generated terminal multisets through multiplicity-variable values, used while these TMS generation. Eq. (77) semantics is quite clear: select those TMS, which are generated by the help of value of variable γ, which (value) is minimal (maximal) among all other TMS, generated by γ application. As seen, Eq. (77) extends optimality definition from only multiplicities-constants, having places in TMS, to also multiplicities-variables, having places in unitary metarules, applied while TMS generation.

Most simple formal definition of the verbally described sense of Eq. (77) optimizing condition may be as follows. Let us introduce l auxiliary terminal objects $\overline{\gamma}_1, ..., \overline{\gamma}_l$ corresponding variables $\gamma_1, ..., \gamma_l$, having places in UMMG $S = \langle a_0, R, F \rangle$, i.e., unitary metarules and boundary condition (66). After that, let us add one new unitary metarule:

$$a'_0 \rightarrow 1 \cdot a_0, \gamma_1 \cdot \overline{\gamma}_1, ..., \gamma_k \cdot \overline{\gamma}_l \qquad (78)$$

to scheme R, thus creating scheme R', which contains Eq. (78) and all elements of R, and substituting all optimizing conditions of the form $\gamma = opt$ by $\overline{\gamma} = opt$ in filter F, thus converting them to the "canonical" form (65)—remember, $\overline{\gamma}$ is object not variable and, more, terminal object, because there is no any UR or UMR with head $\overline{\gamma}$ in R. Obtained filter will be denoted as F'.

As seen now, UMMG $S' = \langle a'_0, R', F' \rangle$ generates terminal multisets of the form:

$$\{n_{i_1} \cdot a_{i_1}, ..., n_{i_k} \cdot a_{i_k}, \overline{n}_1 \cdot \overline{\gamma}_1, ..., \overline{n}_l \cdot \overline{\gamma}_l\}, \qquad (79)$$

where

$$\{n_{i_1} \cdot a_{i_1}, ..., n_{i_l} \cdot a_{i_l}\} \in \overline{V}_S, \qquad (80)$$

and TMS (79) will be selected to $\overline{V}_{S'}$, if and only if TMS (80) satisfies all conditions, entering F and concerning terminal objects $a_{i_1}, ..., a_{i_k}$, as well as TMS $\{\overline{n}_1 \cdot \overline{\gamma}_1, ..., \overline{n}_l \cdot \overline{\gamma}_l\}$ satisfies all optimizing conditions of the form $\overline{\gamma}_i = opt \in F'$, corresponding $\gamma_i = opt \in F$..

It is not difficult to define $\overline{V}_S$ by subtracting from all $v' \in \overline{V}_{S'}$ multisets of the form $\{\overline{n}_1 \cdot \overline{\gamma}_1, ..., \overline{n}_l \cdot \overline{\gamma}_l\}$, but from the practical point of view, it is more useful to consider not $\overline{V}_S$ but $\overline{V}_{S'}$ as a result of application of unitary multimetagrammar S: it is clear that all $v' \in \overline{V}_{S'}$ contain values $\overline{n}_1, ..., \overline{n}_l$ of variables $\gamma_1, ..., \gamma_l$ as terminal objects $\overline{\gamma}_1, ..., \overline{\gamma}_l$ multiplicities, which computation is often main purpose of the mentioned application.

Example 8. As may be seen, problem, described in Section 2, is to obtain m quantity of laboratories, as well as n and l quantities of analysts and assistants, respectively, in one laboratory. Although Eqs. (18)–(20) do not contain optimizing conditions of the form $\gamma = opt$, generating TMS like

$$\{100 \cdot employee, 115000 \cdot eur, 3 \cdot m, 2 \cdot n, 5 \cdot l\} \qquad (81)$$

is much more useful than TMS like {100·*employee,* 115,000·*eur*} because of Eq. (81) with greater informativity (here, we use m, n, l instead of $\overline{n}, \overline{m}, \overline{l}$). ∎

So we shall use $\overline{V}_{S'}$ as a result of $S = \langle a_0, R, F \rangle$ unitary multimetagrammar application, even if R does not include variable-containing optimizing conditions.

To finish with syntax and semantics of UMGs/UMMGs, let us note that class of unitary multigrammars is strict subclass of filtering unitary multiset grammars (UMGs ⊂ FUMGs): every UMGs is FUMGs with empty filter. From the other side, FUMGs are strict subclass of unitary multiset metagrammars (UMGs ⊂ FUMGs): every FUMGs is UMMGs without variable multiplicities and corresponding variable declarations inside filter.

UMG/UMMG algorithmics and applications are considered in the separate chapter of this book.

Author details

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

*Address all correspondence to: sheremet@rfbr.ru

IntechOpen

References

[1] Whitten JL, Bentley LD. Introduction to Systems Analysis and Design. New York: McGraw-Hill Irwin; 2006. p. 640

[2] Bentley LD, Whitten JL. Systems Analysis and Design for the Global Enterprise. New York: McGraw-Hill Irwin; 2007. p. 160

[3] Blanchard BS, Fabrycky WJ. Systems Engineering and Analysis. Englewood Cliffs, NJ: Prentice-Hall; 2010. p. 723

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

[5] Tilly S, Rosenblatt HJ. System Analysis and Design. Boston, MA: Cengage Learning; Ebook-dl.com. 2016. p. 572

[6] Sainter P, Oldham K, Larkin A, Murton A, Brimble R. Product knowledge management within knowledge-based engineering system. – In: Proceedings of ASME 2000 Design Engineering Technical Conference. – Baltimore, Maryland: ASME, 2000. pp. 1-8

[7] Akerkar R, Sajja P. Knowledge-Based Systems. Sudbury, MA: Jones and Bartlett Publishers; 2010. p. 350

[8] Kendal SL, Green M. An Introduction to Knowledge Engineering. London: Springer; 2007. p. 300

[9] Pannu A. Artificial intelligence and its application in different areas. International Journal of Engineering and Innovative Technology. 2015;**4**(4):79-84

[10] Cross TB. The Uses of Artificial Intelligence in Business. New York: Prentice Hall; TECHtionary.com. 2017. p. 271

[11] Gass SI, Assad AA. An Annotated Timeline of Operations Research: An Informal History. NY: Kluwer Academic Publishers; 2005. p. 213

[12] Franks B. The Analytics Revolution: How to Improve Your Business by Making Analytics Operational in the Big Data Era. New York: John Wiley & Sons; 2014. p. 307

[13] Hillier SF, Lieberman GJ. Introduction to Operations Research. Boston, MA: McGraw Hill; 2014. p. 1237

[14] Taha HA. Operations Research: An Introduction. London: Pearson; 2016. p. 838

[15] Marriott K, Stucky PG. Programming with Constraints: An Introduction. Cambridge, MA: MIT Press; 2003. p. 420

[16] Apt K. Principles of Constraint Programming. Cambridge, UK: Cambridge University Press; 2003. p. 420

[17] Frunkwirth T, Abdennadher S. Essentials of Constraint Programming. Berlin: Springer Verlag; 2003. p. 398

[18] Lake J. Sets, fuzzy sets, multisets and functions. Journal of the London Mathematical Society. 1976;**12**:323-326

[19] Hickman JL. A note on the concept of multiset. Bulletin of the Australian Mathematical Society. 1980;**22**:211-217. DOI: 10.1017/5000497270000650X

[20] Meyer RK, McRobbie MA. Multisets and relevant implication. I, II. Australasian Journal of Philosophy. 1982;**60**:107-139. DOI: 10.1080/00048408212340551

[21] Banatre J-P, Le Metayer D. Programming by multiset

transformation. Communications of the ACM. 1993;**36**:98-111. DOI: 10.1145/151233.151242

[22] Marriott K. Constraint multiset grammars. In: Proceedings of IEEE Symposium on Visual Languages. IEEE Computer Society Press; 1994. pp. 118-125. DOI: 10.1109/VL.1994.363633

[23] Marriott K. Parsing visual languages with constraint multiset grammars. In: Programming Languages: Implementation, Logic and Programs. Lecture Notes in Computer Science. Vol. 1292. New York: Springer; 1996. p. 419

[24] Marriott K, Meyer B. On the classification of visual languages by grammar hierarchies. Journal of Visual Languages and Computing. 1997;**8**: 375-402. DOI: 10.1006/jvlc.1997.0053

[25] Calude CS, Paun G, Rozenberg G, Salomaa A. Multisets Processing: Mathematical, Computer Science and Molecular Computing Points of View. Lecture Notes in Computer Science. Vol. 2235. NY: Springer; 2001. p. 359. DOI: 10.1007/3-540-45523-X

[26] Petrovsky AB. Main Notions of the Multisets Theory. Moscow: URSS; 2002. p. 80. (In Russian)

[27] Petrovsky AB. Sets and Multisets Spaces. Moscow: URSS; 2003. p. 248. (In Russian)

[28] Singh D, Ibrahim AM, Yohanna T, Singh JN. An overview of applications of multisets. Novi Sad Journal of Mathematics. 2007;**37**:37-92

[29] Red'ko VN, Bui DB, Grishko Yu A. Modern state of multisets theory from the entity point of view. Cybernetics and Systems Analysis. 2015;**51**:171-178

[30] Sheremet I. A. Recursive Multisets and Their Applications. – Moscow: Nauka; 2010. p. 293. (In Russian)

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

[32] Chomsky N. Syntactic Structures. The Hague: Mouton de Gruyter; 2002. p. 118

[33] Meduna A. Formal Languages and Computation: Models and their Application. New York: CRC Press; 2014. p. 233

[34] Wallace M. Constraint logic programming. In: Computational Logic: Logic Programming and Beyond. Lecture Notes in Computer Science. New York: Springer; Vol. 2407. 2002. pp. 512-556

[35] Bratko I. Prolog Programming for Artificial Intelligence. NY: Addison-Wesley; 2012. p. 696

[36] Diaz D. GNU Prolog. www.gprolog.org. 2018. p. 238