International Journal of Computer Engineering and Information Technology

VOL. 8, NO. 7, July 2016, 106–114
Available online at: www.ijceit.org
E-ISSN 2412-8856 (Online)

Optimizing Multiset Metagrammars. Formal Definitions

Igor Sheremet[1] and Igor Zhukov[2]

[1] Financial University under the Government of Russian Federation, Moscow, Russian Federation

[2] National Research Nuclear University Moscow Engineering Physical Institute, Moscow, Russian Federation

[1]iasheremet@fa.ru, [2]i.zhukov@inbox.ru

ABSTRACT

A multiset-based approach to optimization problems formalizing and solving is considered. Optimizing multiset metagrammars (OMMG) are described as specific knowledge representation model developed precisely for this purposes. OMMG provide convergence of classical optimization theory and modern knowledge engineering combining the best features of mathematical and logical programming. Paper is dedicated to the formal definition of OMMG syntax and semantics as well as to the multigrammatical representation of the most well-known classical optimization problems.

Keywords: *Multisets, Multiset grammars and metagrammars, Optimizing multiset metagrammars, Multigrammatical representation of optimization problems, Unconventional programming paradigms, Knowledge representation, Mathematical programing, Logical programming.*

1. INTRODUCTION

Multisets (MS) for many years are one of the most interesting and promising mathematical objects which application to various areas of modern computer science is investigated thoroughly [1-13]. One of the most useful directions of multisets processing theory and technologies development is rule-based multiset programming paradigm [14-16], which ideology is exploited widely in DNA and membrane computing [17-19] as well as in Gamma-like programming languages [10,20,21], and chemical programming, which is operating macrosets, or generalized multisets [22-25].

As for the state of the art practical applications, the most valuable are constraint multiset grammars (CMSG) developed mainly for visual objects (first of all hand-written symbols) recognition [26-28]. CSMG inherit key features of constraints logical programming [29-31], and are convenient tool for the description of planar objects with complicated recursive structure.

Author's contribution to the area considered are so called optimizing multiset metagrammars (OMMG) [32, 33] developed for problem solving in system analysis and optimization area (first of all, analysis and construction of the large-scale sociotechnical systems like business structures, manufacturing facilities, macroeconomic objects etc.) From the practical point of view OMMG are specific knowledge representation model providing convergence of classical optimization theory and modern knowledge engineering. OMMG integrate basic features of mathematical programming (optimal solutions search in solutions space which is strictly defined by goal functions and constraints) and logical programming (natural and easily modified top-down recursive representation of complex objects and processed as well as systems operation logic).

Presented paper is dedicated to further development of OMMG pragmatics by constructing multigrammatical representation (formulation) of classical optimization problems.

The first part of the paper contains formal definitions of OMMG family and its various subsets, while main content of the second part is mentioned representations description and investigation in order to establish some standard techniques useful for practical problems solving and OMMG application hardware implementation.

2. BASIC NOTIONS AND DEFINITIONS

Theory of multisets background is notion of multiset which is understood mainly as set of so called multiobjects composed of indistinguishable elements (objects). Multiobject containing n objects of type a is denoted as $n \cdot a$, where n is called multiplicity of object a ($\cdot$ is composing symbol). Record

$$v = \{n_1 \cdot a_1, \dots, n_m \cdot a_m\} \quad (1)$$

means that multiset v contains n_1 objects of type a_1, …, n_m objects of type a_m. We shall use one and the same symbol $\in$ for denotation of object and multiobject entering multiset v, so $a_i \in v$ and $n_i \cdot a_i \in v$ means that object a_i

enters v as well as multiobject $n_i \cdot a_i$. From the substantial point of view set $\{a_1, \dots, a_m\}$ and multiset $\{1 \cdot a_1, \dots, 1 \cdot a_m\}$ are equivalent as well as object a and multiobject $1 \cdot a$. Empty multiset and empty set are denoted as $\{\emptyset\}$. If object a multiplicity is zero, it is equivalent to the absence of a in the multiset v, what is recorded, as usual, $a \notin v$.

We shall say that multiset

$$v = \{n_1 \cdot a_1, \dots, n_m \cdot a_m\} \quad (2)$$

is included to multiset

$$v' = \{n'_1 \cdot a'_1, \dots, n'_{m'} \cdot a'_{m'}\} \quad (3)$$

what is denoted as $v \subseteq v'$, if

$$(\forall n \cdot a \in v)\ (\exists n' \cdot a \in v')\ \ n \le n', \quad (4)$$

and v is strictly included to v', what is denoted $v \subset v'$, if $v \subseteq v'$ and $v \ne v'$.

We shall use small "a" with lower indices as well as small "a", "b" etc. for objects denotation; multisets will be denoted by small "v" with or without indices.

Example 1. Multiset $v = \{3 \cdot a, 4 \cdot b, 1 \cdot c\}$ is strictly included to multiset $v' = \{4 \cdot a, 4 \cdot b, 1 \cdot c, 1 \cdot d\}$.

There are five basic operations on multisets: addition, subtraction, multiplication by integer number, join and intersection.

Result of v and v' multisets addition denoted as bold symbol "**+**" is multiset

$$\begin{aligned} & v \mathbf{+} v' \\ & = \Big(\bigcup_{\substack{a \in \{a_1,\dots,a_m\} \cap \{a'_1,\dots,a'_{m'}\} \\ n \cdot a \in v \\ n' \cdot a \in v'}} \{(n + n') \cdot a\} \Big) \cup \\ & \cup \Big(\bigcup_{\substack{a \in \{a_1,\dots,a_m\} - \{a'_1,\dots,a'_{m'}\} \\ n \cdot a \in v}} \{n \cdot a\} \Big) \cup \\ & \cup \Big(\bigcup_{\substack{a \in \{a'_1,\dots,a'_{m'}\} - \{a_1,\dots,a_m\} \\ n' \cdot a \in v'}} \{n' \cdot a\} \Big). \end{aligned} \quad (5)$$

Result of v'subtraction from v denoted as bold symbol "**-**" is defined as follows:

$$\begin{aligned} & v \mathbf{-} v' = \Big(\bigcup_{\substack{a \in \{a_1,\dots,a_m\} \cap \{a'_1,\dots,a'_{m'}\} \\ n \cdot a \in v \\ n' \cdot a \in v' \\ n > n'}} \{(n - n') \cdot a\} \Big) \cup \\ & \cup \Big(\bigcup_{\substack{a \in \{a_1,\dots,a_m\} - \{a'_1,\dots,a'_{m'}\} \\ n \cdot a \in v}} \{n \cdot a\} \Big). \end{aligned} \quad (6)$$

As seen, in (5) resulting multiset contains multiobjects of three types: entering both v and v'; entering only v; entering only v'. Multiplicities of the first are sums $n + n'$, where $n \cdot a \in v$ and $n' \cdot a \in v'$; multiplicities of the second and the third are equal to multiplicities of multiobjects $n \cdot a \in v$ and $n' \cdot a \in v'$ correspondingly. In (6) resulting multiset contains multiobjects which multiplicities are calculated by subtraction n' from n, where n is greater than n', and $n \cdot a \in v$, $n' \cdot a \in v'$; also $v - v'$ contains multiobjects $n \cdot a \in v$ such that $a \notin v'$ (as said higher, this is equivalent to their zero multiplicities in v').

Results of v multiplication by integer number n, what is denoted as $v \ast n$, is multiset:

$$v \ast n = \{(n \times n_1) \cdot a_1, \dots, (n \times n_m) \cdot a_m\}, \quad (7)$$

i.e. multiplicity n_i of every object a_i is multiplied by n (here $\times$ denotes usual multiplication of integer numbers).

Multisets v and v' may be joined (denoted $v \cup v'$) and intersected (denoted $v \cap v'$):

$$\begin{aligned} & v \cup v' \\ & = \Big(\bigcup_{\substack{a \in \{a_1,\dots,a_m\} \cap \{a'_1,\dots,a'_{m'}\} \\ n \cdot a \in v \\ n' \cdot a \in v'}} \{max(n, n') \cdot a\} \Big) \\ & \cup \\ & \cup \Big(\bigcup_{\substack{a \in \{a_1,\dots,a_m\} - \{a'_1,\dots,a'_{m'}\} \\ n \cdot a \in v}} \{n \cdot a\} \Big) \cup \\ & \cup \Big(\bigcup_{\substack{a \in \{a'_1,\dots,a'_{m'}\} - \{a_1,\dots,a_m\} \\ n' \cdot a \in v'}} \{n' \cdot a'\} \Big). \end{aligned} \quad (8)$$

$$\begin{aligned} & v \cap v' \\ & = \bigcup_{\substack{a \in \{a_1,\dots,a_m\} \cap \{a'_1,\dots,a'_{m'}\} \\ n \cdot a \in v \\ n' \cdot a \in v'}} \{min(n, n') \cdot a\} \end{aligned} \quad (9)$$

Record (8) defines that join of v and v' contains all objects, which have place in at least one of those multisets, and multiplicity of each such object is maximal of multiplicities of corresponding objects from v and v' (if object belongs to only one of the multiset, and its multiplicity is n, then $max(n, 0) = n$). Record (9) is similar, but intersection of v and v' contains only those objects that enter both multisets, and multiplicity of each such object is minimal from two.

Example 2. Let $v = \{3 \cdot a, 2 \cdot b, 1 \cdot d\}$, $v' = \{1 \cdot a, 2 \cdot c, 2 \cdot d\}$. Then

$v \mathbf{+} v' = \{4 \cdot a, 2 \cdot b, 2 \cdot c, 3 \cdot d\}$,

$v \mathbf{-} v' = \{2 \cdot a, 2 \cdot b\}$,

$3 \ast v' = \{9 \cdot a, 6 \cdot b, 3 \cdot d\}$,

$v \cup v' = \{3 \cdot a, 2 \cdot b, 2 \cdot c, 2 \cdot d\}$,

$v \cap v' = \{1 \cdot a, 1 \cdot d\}$.

Multisets may be used for generating another multisets by means of multiset grammars. By analogy with classical grammars operating strings of symbols [34,35], we shall define multigrammar as a couple

$$S =< v_o, R >, \quad (10)$$

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

$$v \to v', \quad (11)$$

where v and v', called left and right parts of the rule correspondingly, are multisets, and $v \neq \{\emptyset\}$. Semantics of rule is defined as follows. Let $\bar{v}$ be multiset. We shall say that rule (11) is applicable to $\bar{v}$, if

$$v \subseteq \bar{v}. \quad (12)$$

Then result of application is multiset

$$\bar{v}' = \bar{v} - v + v', \quad (13)$$

i.e. if $\bar{v}$ includes v, then v is replaced by v'. This operation is called generation step.

Generation step providing generation of MS $\bar{v}'$ from MS $\bar{v}$ by application of rule $r \in R$ is denoted as

$$\bar{v} \overset{r}{\Rightarrow} \bar{v}', \quad (14)$$

while fact, that MS $\bar{v}'$ is generated from MS $\bar{v}$ by any (including empty) sequence of generation steps, called also "generation chain", will be recorded as

$$\bar{v} \overset{R}{\Rightarrow} \bar{v}', \quad (15)$$

or, if the only MG is considered, then as in classical grammars,

$$\bar{v} \overset{*}{\Rightarrow} \bar{v}'. \quad (16)$$

Set of multisets generated by MG $S =< v_o, R >$ is denoted V_s and is strictly defined as follows:

$$V_s = \{v \mid v_0 \overset{R}{\Rightarrow} v\}. \quad (17)$$

MS v is called terminal multiset (TMS) if there is no one rule $r \in R$ which is applicable to v. Set of terminal sets (STMS) will be denoted $\bar{V}_s$. Obviously, $\bar{V}_s \subseteq V_s$.

Iterative representation of MG semantics, i.e. SMS V_s generated by MG $S =< v_o, R >$, is following:

$$V_{(0)} = \{v_0\}, \quad (18)$$

$$V_{(i+1)} = V_{(i)} \cup \left(\bigcup_{\bar{v} \in V_{(i)}} \bigcup_{r \in R} \pi(\bar{v}, r) \right), \quad (19)$$

$$V_S = V_{(\infty)}, \quad (20)$$

where

$$\pi(\bar{v}, v \to v') = \begin{cases} \{\bar{v} - v + v'\}, \text{ if } v \subseteq \bar{v}, \\ \{\emptyset\} \text{ otherwise.} \end{cases} \quad (21)$$

As seen, (21) implements application of rule $v \to v'$ as defined by (12) – (13).

SMS V_s is fixed point of the described process, i.e. $V_s = V_{(i)}$ with $i \to \infty$. If for some finite i $V_{(i)} = V_{(i+1)}$, then $V_S = V_{(i)}$, and V_s is also finite. For TMS v

$$\pi(v, r) = \{\emptyset\} \quad (22)$$

for all rules $r \in R$, i.e. no one multiset may be generated from TMS, and

$$\overline{V}_S = \{v \mid v \in V_S \,\&\, (\forall\, r \in R)\ \pi(v, r) = \{\emptyset\}\}. \quad (23)$$

By analogy with classical string-generating grammars, multiset grammars may be general, or context-sensitive (CS), and context-free (CF). In the last, left parts of all rules $r \in R$ are multisets of a form $\{1 \cdot a\}$ while rules in CS MG contain arbitrary left and right parts (the only limitation already mentioned is that left part must be non-empty).

Example 3. Consider context-free multigrammar

$S =< \{1 \cdot a, 2 \cdot b\}, \{r_1, r_2, r_3, r_4\} >$,

where

r_1 is $\{1 \cdot a\} \to \{3 \cdot b, 2 \cdot c\}$,

r_2 is $\{1 \cdot a\} \to \{1 \cdot b, 3 \cdot c\}$,

r_3 is $\{1 \cdot b\} \to \{2 \cdot c\}$,

r_4 is $\{1 \cdot b\} \to \{1 \cdot c\}$,

According to (18) – (19), there are following generation chains:

$\{1 \cdot a, 2 \cdot b\} \overset{r_1}{\Rightarrow} \{5 \cdot b, 2 \cdot c\} \overset{r_3}{\Rightarrow} \{12 \cdot c\}$,

$\{1 \cdot a, 2 \cdot b\} \overset{r_1}{\Rightarrow} \{5 \cdot b, 2 \cdot c\} \overset{r_3}{\Rightarrow} \{7 \cdot c\}$,

$\{1 \cdot a, 2 \cdot b\} \overset{r_2}{\Rightarrow} \{3 \cdot b, 3 \cdot c\} \overset{r_4}{\Rightarrow} \{9 \cdot c\}$,

$\{1 \cdot a, 2 \cdot b\} \overset{r_2}{\Rightarrow} \{3 \cdot b, 3 \cdot c\} \overset{r_4}{\Rightarrow} \{6 \cdot c\}$,

and

$V_s = \{\{1 \cdot b, 2 \cdot c\}, \{5 \cdot b, 2 \cdot c\}, \{3 \cdot b, 3 \cdot c\}, \{6 \cdot c\}, \{7 \cdot c\}, \{9 \cdot c\}, \{12 \cdot c\}\}$,

$\bar{V}_s = \{\{6 \cdot c\}, \{7 \cdot c\}, \{9 \cdot c\}, \{12 \cdot c\}\}$.

3. UNITARY MULTIGRAMMARS

One practice-oriented and useful modification of CF multigrammars are so called unitary multigrammars (UMG) having slightly different syntax and semantics of basic constructions.

UMG is couple $S =< a_o, R >$, where a_o is called title object, and R is, as higher, scheme, being here set of so called unitary rules (UR) having form

$$a \to n_1 \cdot a_1, \ldots, n_m \cdot a_m, \quad (24)$$

where a is called head while unordered list of multiobjects $n_1 \cdot a_1, \ldots, n_m \cdot a_m$ is called body of the unitary rule ("unordered" means all permutations of $n_1 \cdot a_1, \ldots, n_m \cdot a_m$ represent one and the same body, or $n_1 \cdot a_1, \ldots, n_m \cdot a_m$ is interpreted as multiset $\{n_1 \cdot a_1, \ldots, n_m \cdot a_m\}$).

Application of unitary rule (24) to multiset $\bar{v}$ containing multiobject $n \cdot a$ is, as higher, called generation step, and is defined by the following equality not contradicting to (12) – (13) and (21):

$$\bar{v}' = \bar{v} - \{n \cdot a\} + n * \{n_1 \cdot a_1, \dots, n_m \cdot a_m\}, \quad (25)$$

what corresponds to n times application of CF rule

$$\{1 \cdot a\} \to \{n_1 \cdot a_1, \dots, n_m \cdot a_m\}. \quad (26)$$

However, record is not the only difference between context-free and unitary multigrammars. Being applied to practical problems, unitary rules are most frequently interpreted as descriptions of complex objects through their elements (which, in turn, may be decomposed, and so on until non-divided entities): (24) means that object (of type) a consists of n_1 objects (of type) $a_1, \dots, n_m$ objects (of type) a_m. So if there are $m > 1$ alternative ways of object a structuring – for example, along with (24) there is one more UR with head a –

$$a \to n'_1 \cdot a'_1, \dots, n'_{m'} \cdot a'_{m'}, \quad (27)$$

and we apply unitary rule (24) once in the generation chain, then for all following generation steps, if they occur, we shall use the same UR (24). For short, object a structure is postulated constant within all generation chain.

As seen, UMG semantics is based on "variable-like" interpretation of objects, which differ by this feature from non-terminals in classical CF string-generating grammars, which semantics allows arbitrary alternatives of any non-terminal substitution at every generation step where this non-terminal may be replaced.

Iterative representation of UMG $S =< a_o, R >$ semantics is following:

$$X_{(0)} = \{< \{1 \cdot a_0\}, R >\}, \quad (28)$$

$$X_{(i+1)} = \bigcup_{<\bar{v},\bar{R}> \in X_{(i)}} \bigcup_{r \in \bar{R}} \{\bar{\pi}(\bar{v}, \bar{R}, r)\}, \quad (29)$$

$$V_S = \{\bar{v} \mid < \bar{v}, \bar{R} > \in X_{(\infty)}\}, \quad (30)$$

where

$$\bar{\pi}(\bar{v}, \bar{R}, r) = \quad (31)$$

$$= \begin{cases} < \bar{v} - \{n \cdot a\} + n * \{n_1 \cdot a_1, \dots, n_m \cdot a_m \\ \{\emptyset\} \text{ otherwise,} \end{cases}$$

$$alt(a \to w) = \{a \to w' \mid a \to w' \in \bar{R} \;\&\; w' \neq w\}, \quad (32)$$

where w and w' designate $n_1 \cdot a_1, \dots, n_m \cdot a_m$ and $n'_1 \cdot a'_1, \dots, n'_{m'} \cdot a'_{m'}$ correspondingly, and r is unitary rule $a \to n_1 \cdot a_1, \dots, n_m \cdot a_m$. Inequality $w \neq w'$ in (32) is interpreted as if w and w' were multisets.

In (28) – (30) $X_{(i)}$ is set of couples of the form $< \bar{v}, \bar{R} >$, where $\bar{v}$ is multiset created by already executed generation steps, and $\bar{R}$ is subset of R scheme including all unitary rules which may be applied to $\bar{v}$ at the next such step. As seen, difference between (21) and (31) is that in the last case $a \to w$ application to $\bar{v}$ implemented by $\bar{\pi}$ results not only in substitution of n objects a having place in $\bar{v}$ by $n \times n_1$ objects a_1, …, $n \times n_m$ objects a_m, but also in elimination from $\bar{R}$, which is current R subset of applicable unitary rules, all URs $a \to w'$ with the same head a and alternative bodies $w' \neq w$. This prevents from applying $a \to w'$ in future generation steps, that strictly corresponds to URs semantics verbally described higher; so object a will be "detailed" by the same only UR $a \to w$, as at the executed step. As seen, result of function *alt* application is set of all URs with the same head a and bodies different from w having place in the argument $a \to w$.

Lower we shall designate by $\beta(v)$ set of all objects having place in multiset v (it is called multiset basis [11, 12]):

$$\beta(v) = \{a \mid a \in v\}. \quad (33)$$

By A_s we shall designate set of all objects having place in UMG $S =< a_0, R >$, while by $\overline{A_s}$ – set of all terminal objects having place in S, i.e. objects which are not heads of unitary rules $r \in R$ and present only in URs bodies. As seen,

$$\overline{A_s} \subseteq A_s. \quad (34)$$

Multiset $v \in V_s$ such that all objects of v are terminal is, as higher, called terminal multiset; this notion fully corresponds to the defined one in relation to multigrammars. Formally,

$$\overline{V_s} = \{v | v \in V_s \,\&\, \beta(v) \subseteq \overline{A_s}\} \quad (35)$$

Let us present an example illustrating main semantical feature of unitary multigrammars in comparison with context-free multigrammars.

Example 4. Consider UMG $S =< a, R >$, where R contains following unitary rules (recorded for unambiguity in angle brackets):

$r_1 = \langle a \to 2 \cdot b, 3 \cdot c \rangle$, $r_2 = \langle a \to 3 \cdot b, 2 \cdot c \rangle$,
$r_3 = \langle b \to 1 \cdot d \rangle$, $r_4 = \langle b \to 2 \cdot d \rangle$,
$r_5 = \langle c \to 1 \cdot b, 2 \cdot d \rangle$, $r_6 = \langle c \to 3 \cdot b, 1 \cdot d \rangle$.

There are following generation chains:

$a \overset{r_1}{\Rightarrow} \{2 \cdot b, 3 \cdot c\} \overset{r_3}{\Rightarrow} \{3 \cdot c, 2 \cdot d\} \overset{r_5}{\Rightarrow} \{3 \cdot b, 6 \cdot d\} \overset{r_3}{\Rightarrow} \{9 \cdot d\}$,
$a \overset{r_1}{\Rightarrow} \{2 \cdot b, 3 \cdot c\} \overset{r_3}{\Rightarrow} \{3 \cdot c, 2 \cdot d\} \overset{r_6}{\Rightarrow} \{9 \cdot b, 3 \cdot d\} \overset{r_3}{\Rightarrow} \{12 \cdot d\}$,
$a \overset{r_1}{\Rightarrow} \{2 \cdot b, 3 \cdot c\} \overset{r_4}{\Rightarrow} \{3 \cdot c, 4 \cdot d\} \overset{r_5}{\Rightarrow} \{3 \cdot b, 10 \cdot d\} \overset{r_4}{\Rightarrow} \{16 \cdot d\}$,
$a \overset{r_1}{\Rightarrow} \{2 \cdot b, 3 \cdot c\} \overset{r_4}{\Rightarrow} \{3 \cdot c, 4 \cdot d\} \overset{r_6}{\Rightarrow} \{9 \cdot b, 7 \cdot d\} \overset{r_4}{\Rightarrow} \{25 \cdot d\}$,

etc. (generation chains beginning from r_2 unitary rule may be constructed by the concerned reader independently). As seen, there is no one generation chain which contains r_3 and r_4 simultaneously: application of r_3 at the second generation step excludes application of r_4 at the fourth one, and vice versa.

4. FILTERS AND FILTERING UNITARY MULTIGRAMMARS

Filters are constructions which provide selection of subsets of sets of generated TMS.

One may consider filters as query language with special features used for selection of solutions from solutions space.

Filters consist from atomary constructions called conditions. There are two types of conditions: boundary

and optimizing. The first, in turn, may be elementary and chain.

Elementary boundary condition (EBC) may have form apn, npa and apa', where a and a' are objects, n is integer number, and $\rho \in \{<, =, \leq\}$ is symbol of relation. EBC semantics is following.

Let V be set of multisets. Multiset $v \in V$ satisfies EBC $\bar{n}\rho a$, if $n \cdot a \in v$, and $\bar{n}\rho n$ is true. Similarly, $v \in V$ satisfies EBC $a\rho\overline{n}$, if $n\rho\bar{n}$ is true. And, at last, $v \in V$ satisfies EBC $a\rho a'$, if $\overline{n} \cdot a \in v$, $\overline{n}' \cdot a' \in v$ and $\bar{n}\rho\bar{n}'$ is true.

Example 5. Let $V = \{v_1, v_2, v_3\}$, where
$v_1 = \{3 \cdot a, 2 \cdot b, 4 \cdot c\}$,
$v_2 = \{1 \cdot a, 3 \cdot b\}$,
$v_3 = \{1 \cdot b, 5 \cdot c\}$.

Then result of application of EBC $a < 2$ to v is set $\{v_2, v_3\}$ (note, that v_3 is included to this set because, as said higher, absence of object in the multiset is equivalent to zero multiplicity of this object). If EBC is $b > 3$, then result is empty set, if $c = 5$, then result is $\{v_3\}$; if $b > c$, then result is $\{v_2\}$.

Optimizing condition has form $a = opt$, where a is object, and $opt \in \{min, max\}$.

Multiset $v \in V$ satisfies optimizing 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 that multiplicity n' in multiobject $n' \cdot a \in v'$, i.e. $n \leq n'$.

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 multiobject $n' \cdot a \in v'$, i.e. $n \geq n'$. As higher, case $a \notin v$ ($a' \notin v$) is equivalent to $0 \cdot a \in v$ ($0 \cdot a' \in v'$).

Example 6. Let V be the same as in the previous example, and optimizing condition is $a = max$. Then result of its application to v is set $\{v_1\}$. If optimizing condition is $b = min$, then result is $\{v_3\}$.

Filter F is set of conditions and may be represented as join of boundary $F_{\leq}$ and optimizing F_{opt} subfilters:

$$F = F_{\leq} \cup F_{opt}, \tag{36}$$

where $F_{\leq}$ is set of EBC and F_{opt} is set of optimizing conditions.

Result of filtration of MS V by filter F is denoted $V \downarrow F$ and is defined as follows:

$$V \downarrow F = (V \downarrow F_{\leq}) \downarrow F_{opt}, \tag{37}$$

i.e. V is filtered by boundary subfilter, and obtained result is filtrated by optimizing subfilter:

$$F_{\leq} = \{bc_1, \dots, bc_k\}, \tag{38}$$

$$F_{opt} = \{opt_1, \dots, opt_l\}, \tag{39}$$

$$v \downarrow F_{\leq} = \bigcap_{i=1}^{k} (v \downarrow \{bc_i\}) = v', \tag{40}$$

$$v' \downarrow F_{opt} = \bigcap_{j=1}^{l} (v' \downarrow \{opt_j\}). \tag{41}$$

Example 7. Let V is the same as in two previous examples, and filter
$F = \{b > 1, b \leq 3, c = min\}$.
Then

$$V \downarrow F = (V \downarrow F_{\leq}) \downarrow F_{opt} = (V \downarrow \{b > 1, b \leq 3\}) \downarrow \{c = min\} = (\{v_1, v_2\} \cap \{v_1, v_2, v_3\}) \downarrow \{c = min\} = \{v_1, v_2\} \downarrow \{c = min\} = \{v_2\},$$

because $c \notin v_2$, so object c multiplicity is zero.

Note that due to commutativity of set-theoretical join and intersection operations filtration inside subfilters may be executed in arbitrary order.

Let us now define syntax and semantics of so called filtering multigrammars (FMG) and filtering unitary multigrammars (FUMG).

FMG $S =< v_0, R, F >$, where v_0 and R have the same sense as in MG, and F is filter, defines set of terminal multisets in the following way:

$$\overline{V_S} = \overline{V_{S'}} \downarrow F, \tag{42}$$

where $S' =< v_0, R >$; i.e. STMS, generated by MG S', is filtered by F, and result is defined as generated by FMG S.

Similarly, FUMG $S =< a_0, R, F >$, where a_0 and R have the same sense as in UMG, and F is filter, defines set of terminal multisets as follows:

$$\overline{V_S} = \overline{V_{S'}} \downarrow F \tag{43}$$

where $S' =< a_0, R >$.

Example 8. Let $S =< a, R, F >$, where $R = \{r_1, r_2, r_3, r_4\}$, and
$r_1 =< a \to 3 \cdot b, 2 \cdot c >$, $r_2 =< a \to 2 \cdot b, 5 \cdot c >$,
$r_3 =< b \to 2 \cdot c, 1 \cdot d >$, $r_4 =< b \to 1 \cdot c, 3 \cdot d >$,
and $F = \{c \leq 8, d > 3, c = max, d = min\}$. Then
$S' =< a, R >$,
$\overline{V_{S'}} = \{\{8 \cdot c, 3 \cdot d\}, \{5 \cdot c, 9 \cdot d\}, \{9 \cdot c, 2 \cdot d\}, \{7 \cdot c, 6 \cdot d\}\}$,

$$\overline{V_S} = \overline{V_{S'}} \downarrow F = (\overline{V_{S'}} \downarrow \{c \leq 8, d > 2\}) \downarrow \{c = max, d = min\} = \{\{8 \cdot c, 3 \cdot d\}, \{5 \cdot c, 9 \cdot d\}, \{7 \cdot c, 6 \cdot d\}\} \downarrow \{c = max, d = min\} = \{\{8 \cdot c, 3 \cdot d\}\}.$$

Type equation here.

There may be more sophisticated boundary subfilters containing not only elementary but also so called chain boundary condition (CBC). Each CBC is constructed from EBC by writing them sequentially:

$$e_1\rho_1 e_2\rho_2 \dots e_i\rho_i e_{i+1} \dots e_m\rho_m e_{m+1}, \tag{44}$$

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

CBC semantics is defined as follows. CBC (44) is replaced by boundary filter

$$\{e_1\rho_1 e_2, e_2\rho_2 e_3, \dots, e_i\rho_i e_{i+1}, \dots, e_m\rho_m e_{m+1}\}, \tag{45}$$

and multiset $v \in V$ satisfies CBC (43) if and only if it satisfies (45).

Example 9. Let $V =< v_1, v_2 >$, where
$v_1 = \{5 \cdot a, 3 \cdot b, 1 \cdot c\}$,
$v_2 = \{2 \cdot a, 4 \cdot c, 3 \cdot d\}$,
and chain boundary conditions are $2 \le a \le 4$, $1 \le c \le b \le 3$ and $a = b < 5$.

Table 1 contains results of application of listed boundary conditions to V.

Table 1

bc	$V \downarrow \{bc\}$
$2 \le a \le 4$	$\{v_2\}$
$1 \le c \le b \le 3$	$\{v_1\}$
$a = b < 5$	$\{\emptyset\}$

Moreover, boundary subfilters may be sets of logical expressions consisting of boundary conditions and logical operations ($\wedge, \vee$ and $\neg$) composed, if necessary, by brackets to any level of complexity. Similar structure may have optimizing subfilters. By this filters language becomes close to relational database query languages [36,37]. Similar approach to the integer programming in considered in [38].

Having in mind introduced notions let us move to the most general and flexible tool discussed in this paper.

5. UNITARY MULTISET METAGRAMMARS

Unitary multigrammars are useful for the description of objects which structure variants number is relatively little, and this variety may be simply reflected by explicit representation of alternative structures as unitary rules with the same head and different bodies. But in practically interest cases this number may be such, that explicit representation of sets of alternatives for all objects, which structure if apriori unknown, if really impossible. So there is necessary such UMG extension which would allow compact description of arbitrary number of alternative variants of object structure. This extension is called unitary multiset metagrammars (multimetagrammars), or, for short, UMMG.

Unitary multiset metagrammar S is triple $< a_0, R, F >$, where a_0 and F are, as higher, title object and filter correspondingly, while R is scheme containing unitary rules and so called unitary metarules (UMR).

Unitary metarule has form

$$a \to \mu_1 \cdot a_1, \dots, \mu_m \cdot a_m, \tag{46}$$

where μ_i is positive integer number, as in sections 2-3, or variable $\gamma \in \Gamma$, where Γ is universum of variables. When μ_i is variable then it is called multiplicity-variable (MV). As seen, unitary rule is partial case of unitary metarule, which multiplicities $\mu_1, \dots, \mu_m$ having place in (46) are constants. As in (24), object a is called head, while unordered (in the same sense as in UR definition) list $\mu_1 \cdot a_1, \dots, \mu_m \cdot a_m$ – body of UMR.

Filter F of UMMG $S =< a_0, R, F >$ is set of conditions, which may be boundary and optimizing as in UMG, i.e. having form $a\rho n, n\rho a, a\rho a'$ and $a = opt$, where $\rho \in \{<, =, \le\}$, $opt \in \{min, max\}$. At the same time F contains chain boundary conditions of the form

$$n \le \gamma \le n', \tag{47}$$

which contain variables not objects ($n \ge 0$); there is one and only one CBC (47) for every variable γ having place in unitary metarules entering scheme R. This CBC is called variable declaration and has quite evident semantics.

If F includes subfilter $F_\Gamma = \{n_1 \le \gamma_1 \le n_1', \dots, n_l \le \gamma_l \le n_l'\}$ containing all variables declarations, then every tuple $\langle \overline{n_1}, \dots, \overline{n_l} \rangle$ such that $\overline{n_1} \in [n_1, n_1'], \dots, \overline{n_l} \in [n_l, n_l']$ provides creation of one unitary multigrammar by substitution of $\overline{n_1}, \dots, \overline{n_l}$ to all unitary metarules having place in R scheme instead of multiplities-variables being in their bodies. Unitary rules entering R are transferred to new scheme denoted $R \circ \langle \overline{n_1}, \dots, \overline{n_l} \rangle$ without any transformations. Every such UMG generates set of terminal multisets by the help of filter $F = F - F_\Gamma$, which contains all "usual" (without variables) conditions (both boundary and optimizing). As may be seen from this short informal description, UMMG $S = \langle a_o, R, F \rangle$ is simple unified tool for compact representation of set of FUMG denoted S^*: for practically valuable problems number of such FUMG, i.e. S^* cardinality, may reach extremely large numbers. It is easy to see that this number is not greater than product of all variables domains cardinalities:

$$|S^*| \le \prod_{j=1}^{l} \left(n_j' - n_j + 1\right) \tag{48}$$

(if there are no identical UMG after variables values substitution then (48) becomes equality).

Let us now give strict definition of unitary multiset metagrammars semantics. (From the described it is obvious nature of "multiset metagrammar" notion – as "metalanguage" is used for description of another language, so "multiset metagrammar" is "unitary-like" multigrammar used for description of other UMG by means of unitary metarules, variables-multiplicities as well as by boundary conditions defining their domains).

UMMG $S = \langle a_o, R, F \rangle$ defines set of terminal multisets $\overline{V_s}$ in such a way:

$$\overline{V_s} = \left(\bigcup_{\overline{s} \in S^*} \overline{V_{\overline{s}}} \right) \downarrow \mathcal{F}, \tag{49}$$

$$S^* = \bigcup_{\overline{n}_1 \in [n_1, n_1']} \dots \bigcup_{\overline{n_l} \in [n_l, n_l']} \{ \langle a_o, R \circ \langle \overline{n_1}, \dots, \overline{n_l} \rangle \rangle \} \tag{50}$$

$$R \circ \langle \overline{n_1}, \dots, \overline{n_l} \rangle = \{ r \circ \langle \overline{n_1}, \dots, \overline{n_l} \rangle | r \in R \}, \quad (51)$$

$$\mathcal{F} = F - F_\Gamma, \quad (52)$$

$$F_\Gamma = \bigcup_{j=1}^{l} \{ n_j \le \gamma_j \le n_j' \}, \quad (53)$$

and, if r is $a \leftarrow \mu_1 \cdot a_1, \dots, \mu_m \cdot a_m$, then $r \circ \langle \overline{n_1}, \dots, \overline{n_l} \rangle$ is unitary rule

$$a \leftarrow (\mu_1 \circ \langle \overline{n_1}, \dots, \overline{n_l} \rangle) \cdot a_1, \dots, (\mu_m \circ \langle \overline{n_1}, \dots, \overline{n_l} \rangle) \cdot a_m, \quad (54)$$

where

$$\mu_i \circ \langle \overline{n_1}, \dots, \overline{n_l} \rangle = \begin{cases} \mu_i, \text{ if } \mu_i \in [1, \infty], \\ \overline{n_i}, \text{ if } \mu_i \in \Gamma. \end{cases} \quad (55)$$

As seen, according to (54) – (55), all multiplicities-variables of UMR $a \leftarrow \mu_1 \cdot a_1, \dots, \mu_m \cdot a_m$ are substituted by their corresponding values from tuple $\langle \overline{n_1}, \dots, \overline{n_l} \rangle$ while all multiplities-constants remain unchanged. Evidently, if all $\mu_1, \dots, \mu_m$ are constants, i.e. UMR is UR, it remains unchanged.

Note, that multiplicities-variables area of actuality is whole R scheme, i.e. if there are $n > 1$ occurrences of one and the same variable γ in different unitary metarules, they all are substituted by one and the same value from the applied tuple $\langle \overline{n_1}, \dots, \overline{n_l} \rangle$.

Example 10. Consider UMMG $S = \langle a, R, F \rangle$, where R contains following three unitary metarules:

$r_1 = \langle a \to 3 \cdot b, \gamma_1 \cdot c \rangle,$
$r_2 = \langle b \to 1 \cdot c, \gamma_2 \cdot d \rangle,$
$r_3 = \langle b \to \gamma_1 \cdot c, \gamma_2 \cdot d \rangle,$

while filter F contains following conditions:

$2 \le c \le 4,$
$1 \le \gamma_1 \le 2,$
$0 \le \gamma_2 \le 1,$
$d = max.$

According to (49) – (55), S defines four UMG:

$S_1 = \langle a, R \circ \langle 1,0 \rangle \rangle,$
$S_2 = \langle a, R \circ \langle 1,1 \rangle \rangle,$
$S_3 = \langle a, R \circ \langle 2,0 \rangle \rangle,$
$S_4 = \langle a, R \circ \langle 2,1 \rangle \rangle,$

which correspond to $4 = 2 \times 2$ sets of two-values domains of γ_1 and γ_2 variables.

As seen,

$R_1 = R \circ \langle 1,0 \rangle = \{ \langle a \to 3 \cdot b, 1 \cdot c \rangle, \langle b \to 1 \cdot c \rangle \},$
$\bar{V}_{\bar{s}_1} = \{ \{ 4 \cdot c \} \},$
$R_2 = R \circ \langle 1,1 \rangle = \{ \langle a \to 3 \cdot b, 1 \cdot c \rangle, \langle b \to 1 \cdot c, 1 \cdot d \rangle \},$
$\bar{V}_{\bar{s}_2} = \{ \{ 4 \cdot c, 3 \cdot d \} \},$
$R_3 = R \circ \langle 2,0 \rangle = \{ \langle a \to 3 \cdot b, 2 \cdot c \rangle, \langle b \to 1 \cdot c \rangle \},$
$\bar{V}_{\bar{s}_3} = \{ \{ 5 \cdot c \} \},$
$R_4 = R \circ \langle 2,1 \rangle = \{ \langle a \to 3 \cdot b, 2 \cdot c \rangle, \langle b \to 1 \cdot c, 1 \cdot d \rangle, \langle b \to 2 \cdot c, 1 \cdot d \rangle \},$
$\bar{V}_{\bar{s}_4} = \{ \{ 5 \cdot c, 3 \cdot d \}, \{ 8 \cdot c, 3 \cdot d \} \}.$

(note, that after variables values substitution there may occur identical URs as in R_1, R_2 and R_3 cases, that's why number of URs in constructed UMG may be less than number of UMRs in initial UMMG). After that,

$\bar{V}_s = (\bar{V}_{\bar{s}_1} \cup \bar{V}_{\bar{s}_2} \cup \bar{V}_{\bar{s}_3} \cup \bar{V}_{\bar{s}_4}) \downarrow \{ 2 \le c \le 4, d = max \} =$
$= (\{ \{ 4 \cdot c \}, \{ 4 \cdot c, 3 \cdot d \}, \{ 5 \cdot c \}, \{ 5 \cdot c, 3 \cdot d \}, \{ 8 \cdot c, 3 \cdot d \} \} \downarrow \{ 2 \le c \le d \}) \downarrow \{ d = max \} = \{ \{ 4 \cdot c \}, \{ 4 \cdot c, 3 \cdot d \} \} \downarrow \{ d = max \} = \{ \{ 4 \cdot c, 3 \cdot d \} \}.$

As seen, UMMG filters contain boundary conditions concerning objects and variables but optimizing conditions concerning only objects. That's why from both theoretical and practical points of view it is reasonable to extend mentioned filters by optimizing conditions relating variables and having form

$$\gamma = opt. \quad (56)$$

This form defines generated TMS optimality through multiplicities-variables values used while these terminal multisets generation.

Semantics of optimizing condition (56) is quite clear: select those TMS which are generated by the help of γ value which is optimal (minimal, maximal) among all other TMS generated by the help of all other values from γ domain. By this verbal description we extend TMS optimality notion from multiplicities-constants to also multiplicities-variables having place in unitary metarules applied while TMS generation.

Formal definition of the sense of (56) verbally described higher is as follows.

Let us introduce l auxiliary terminal objects $\overline{\gamma_1}, \dots, \overline{\gamma_l}$ corresponding to variables $\gamma_1, \dots, \gamma_l$ having place in UMMG $S = \langle a_0, R, F \rangle$, i.e. in unitary metarules and boundary conditions defining multiplicities-variables domains. After that let us add one new unitary metarule

$$\overline{a}_0 \to 1 \cdot a_0, \gamma_1 \cdot \overline{\gamma_1}, \dots, \gamma_l \cdot \overline{\gamma_l} \quad (57)$$

to R scheme thus creating new $\bar{R}$ scheme which contains (57) and all elements (URs and UMRs) of R. After that we shall replace all optimizing conditions of the form $\gamma = opt$, having place in F filter, by $\bar{\gamma} = opt$, thus converting them to the "canonical" object-containing form; let us repeat, that $\bar{\gamma}$ is object not variable and, more, terminal object because there is no any UR or UMR with the head $\bar{\gamma}$ in R. Constructed filter will be denoted lower as $\overline{F}$.

As seen now, UMMG $\bar{S} = \langle \overline{a_0}, \bar{R}, \overline{F} \rangle$ generates terminal multisets of the form

$$\left\{ n_{i_1} \cdot a_{i_1}, \dots, n_{i_k} \cdot a_{i_k}, \overline{n_{j_1}} \cdot \overline{\gamma_1}, \dots, \overline{n_{j_l}} \cdot \overline{\gamma_l} \right\}, \quad (58)$$

which are selected to $\bar{V}_{\bar{s}}$, if and only if TMS

$$\left\{ n_{i_1} \cdot a_{i_1}, \dots, n_{i_k} \cdot a_{i_k} \right\} \quad (59)$$

satisfies all conditions entering F and concerning terminal objects having place in R scheme, as well as TMS

$$\left\{ \overline{n_{j_1}} \cdot \overline{\gamma_1}, \dots, \overline{n_{j_l}} \cdot \overline{\gamma_l} \right\} \quad (60)$$

satisfies all optimizing conditions of the form $\overline{\gamma_i} = opt_i$.

It is not difficult to define $\bar{V}_{\bar{s}}$ by subtracting multisets of the form (60) from all TMS $v \in \bar{V}_{\bar{s}}$, but from the practical point of view it is more useful to consider $\bar{V}_{\bar{s}}$ not $\bar{V}_{\bar{s}}$ as a result of S application. It's clear that all TMS $v \in \bar{V}_{\bar{s}}$ contain variables $\gamma_1, \dots, \gamma_l$ values as multiplicities-constants of terminal objects $\overline{\gamma_1}, \dots, \overline{\gamma_l}$, which computation is usually main purpose of the mentioned unitary multimetagrammar application.

Example 11. Consider UMMG $S = \langle a, R, F \rangle$, where R contains the same unitary metarules as in the previous example while filter F contains all conditions from this example and one more optimizing condition $\gamma_1 = max$. Then

$\bar{R} = R \cup \{r_0\}$,

where

$r_0 = \langle \overline{a} \to 1 \cdot a, \gamma_1 \cdot \overline{\gamma_1}, \gamma_2 \cdot \overline{\gamma_2} \rangle$,

and $\bar{S} = \langle \overline{a}, \bar{R}, \overline{F} \rangle$, where

$\overline{F} = F \cup \{\overline{\gamma_1} = max\}$.

According to (49) – (55), UMMG $\bar{S}$ defines four UMMGs

$\bar{S}_1 = \langle a, \bar{R} \circ \langle 1,0 \rangle \rangle$,

$\bar{S}_2 = \langle a, \bar{R} \circ \langle 1,1 \rangle \rangle$,

$\bar{S}_3 = \langle a, \bar{R} \circ \langle 2,0 \rangle \rangle$,

$\bar{S}_4 = \langle a, \bar{R} \circ \langle 2,1 \rangle \rangle$,

which, as in the Example 10, correspond to four sets of variables γ_1 and γ_2 values. Then, as higher,

$$\bar{R}_1 = \bar{R} \circ \langle 1,0 \rangle = \{\langle a_0 \to 1 \cdot a, 1 \cdot \overline{\gamma_1} \rangle, \langle a \to 3 \cdot b, 1 \cdot c \rangle, \langle b \to 1 \cdot c \rangle\},$$

$$\bar{V}_{\bar{s}_1} = \left\{\{4 \cdot c, 1 \cdot \overline{\gamma_1}\}\right\},$$

$$\bar{R}_2 = \bar{R} \circ \langle 1,1 \rangle = \{\langle a_0 \to 1 \cdot a, 1 \cdot \overline{\gamma_1}, 1 \cdot \overline{\gamma_2} \rangle, \langle a \to 3 \cdot b, 1 \cdot c \rangle, \langle b \to 1 \cdot c \rangle\},$$

$$\bar{V}_{\bar{s}_2} = \left\{\{4 \cdot c, 3 \cdot d, 1 \cdot \overline{\gamma_1}, 1 \cdot \overline{\gamma_2}\}\right\},$$

$$\bar{R}_3 = \bar{R} \circ \langle 2,0 \rangle = \{\langle a_0 \to 1 \cdot a, 2 \cdot \overline{\gamma_1} \rangle, \langle a \to 3 \cdot b, 2 \cdot c \rangle, \langle b \to 1 \cdot c \rangle\},$$

$$\bar{V}_{\bar{s}_3} = \left\{\{5 \cdot c, 2 \cdot \overline{\gamma_1}\}\right\},$$

$$\bar{R}_4 = \bar{R} \circ \langle 2,1 \rangle = \{\langle a_0 \to 1 \cdot a, 2 \cdot \overline{\gamma_1}, 1 \cdot \overline{\gamma_2} \rangle, \langle a \to 3 \cdot b, 2 \cdot c \rangle, \langle b \to 1 \cdot c, 1 \cdot d \rangle, \langle \to 2 \cdot c, 1 \cdot d \rangle\},$$

$$\bar{V}_{\bar{s}_4} = \left\{\{5 \cdot c, 3 \cdot d, 2 \cdot \overline{\gamma_1}, 1 \cdot \overline{\gamma_2}\}, \{8 \cdot c, 3 \cdot d, 2 \cdot \overline{\gamma_1}, 1 \cdot \overline{\gamma_2}\}\right\}.$$

After that

$$\bar{V}_{\bar{s}} = \left(\bar{V}_{\bar{s}_1} \cup \bar{V}_{\bar{s}_2} \cup \bar{V}_{\bar{s}_3} \cup \bar{V}_{\bar{s}_4}\right) \downarrow \{2 \le c \le 4, d = max, \overline{\gamma_1} = max\} = \left\{\{4 \cdot c, 1 \cdot \overline{\gamma_1}\}, \{4 \cdot c, 3 \cdot d, 1 \cdot \overline{\gamma_1}, 1 \cdot \overline{\gamma_2}\}\right\} \downarrow \{d = max, \overline{\gamma_1} = max\} = \left\{\{4 \cdot c, 3 \cdot d, 1 \cdot \overline{\gamma_1}, 1 \cdot \overline{\gamma_2}\}\right\},$$

i.e. set of terminal multisets generated by UMMG S contains the only TMS $\{4 \cdot c, 3 \cdot d, 1 \cdot \bar{\gamma}_1, 1 \cdot \bar{\gamma}_2\}$, which corresponds to variables values $\gamma_1 = 1$ and $\gamma_2 = 1$.

As seen, it is very practically useful to consider namely $\bar{V}_{\bar{s}}$ as a result of UMMG $S = \langle a_0, R, F \rangle$ application (even in the cases where R does not contain optimizing conditions of the form $\gamma = opt$). The only inconveniency which attentive reader may notice due to the last example is the absence of multiobjects of the form $n \cdot \overline{\gamma}$, where $n = 0$. This inconvenience, however, may be eliminated simply by imperative inclusion of multiobjects of the form $0 \cdot \overline{\gamma}$ to the generated multiset.

UMMG, which filter contains at least one optimizing condition, is called ***optimizing multimetagrammar***. As may be seen, optimizing multimetagrammars are special form of constraints logic programming language, oriented to systems analysis and optimization applications. Mutual interconnections of different classes of unitary multiset grammars are presented at Fig.1.

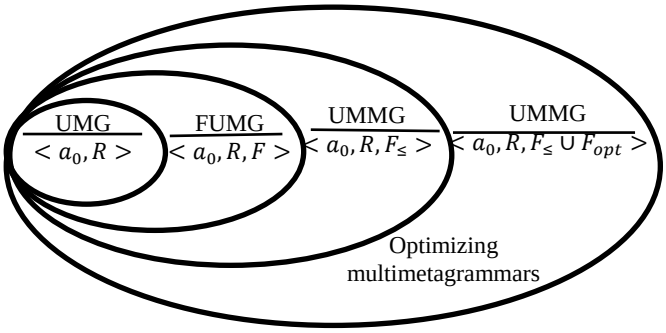

Fig. 1.

OMMG algorithmics as well as OMMG family subsets and formal features (internal alternativity, integrity, cyclicity, finitarity) are described in details in [32,33], while multigrammatical representation of classical optimization problems – in the second part of this paper.

REFERENCES

[1] Bender E.A. Partitions of Multisets. – Discrete Mathematics, Vol.9 (1974), pp. 301-311.

[2] Lake J. Sets, Fuzzy Sets, Multisets and Functions // Journal of London Mathematical Society, Vol.12 (1976), pp. 323-326.

[3] Hickman J. L. A Note on the Concept of Multiset // Bulletin of Australian Mathematical Society, Vol. 22 (1980), pp. 211-217.
[4] Meyer R. K., McRobbie M. A. Multisets and Relevant Implication. I, II. // Australian Journal of Philosophy, Vol. 60 (1982), pp. 107-139, 265-281.
[5] Blizard W. Multiset Theory. – Notre Dame Journal of Formal Logic. – Vol.30 (1989), pp. 36-66.
[6] Martin V. A Geometrical Approach to Multiset Orderings. – Theoretical Computer Science, Vol. 67 (1989), pp. 37-54.
[7] Knuth D.E. Context – Free Multilanguages. – Theoretical Studies in Computer Science – academic Press, 1992, pp. 1-13.
[8] Banatre J.-P., Le Metayer D. Programming by Multiset Transformation. – Communications of the ACM, Vol.36 (1993), No.1, pp. 98-111.
[9] Marriott K., Meyer B. On the Classification of Visual Languages by Grammar Hierarchies. – Journal of Visual Languages and Computing, Vol. 8 (1997), pp. 375-402.
[10] Multisets Processing: Mathematical, Computer Science and Molecular Computing Points of View. Ed. by C.S. Calude, G.Paun, G. Rozenberg, A. Salomaa. – Lecture Notes in Computer Science, Vol. 2235. – Springer – Verlag, 2001.
[11] Petrovskiy A.B. Main notions of Multiset Theory. – Moscow: Editorial URSS, 2002 (in Russian).
[12] Petrovskiy A.B. Spaces of Sets and Multisets. – Moscow: Editorial URSS, 2003 (in Russian).
[13] Singh D., Ibrahim A.M., Yohanna T., Singh J.N. An Overview of the Applications of Multisets. – Novi Sad J. Math., Vol.37 (2007), №2, pp. 73-92.
[14] Krishinamwithy E.W. Rule-Based Multiset Programming Paradigm: Applications. – ACT 0200, Canberra, Australia, 2005.
[15] Soon L. L. E., Cervesato I. Optimized Compilation of Multiset Rewriting with Comprehensions. – Programming Languages and Systems. – Lecture Notes in Computer Science, Vol. 8858. – Springer–Verlag, 2014.
[16] Henglein F. Generic Multiset Programming. – University of Copenhagen, 2015.
[17] Paun G., Rozenberg G., Salomaa A. DNA computing. New Comptutation Paradigm.
[18] Paun Gh. Membrane Computing – An Introduction. – Springer–Verlag, 2002.
[19] The Oxford Handbook of Membrane Computing. Ed. by G. Paun, G. Rozenberg, A. Salomaa. – Oxford University Press, 2010.
[20] Giancarini P., Fogli D., Gaspari M. A Language Based on GAMMA-like Multiset Rewriting. – Extensions of Logic Programming. 5th International Workshop. – Lecture Notes in Computer Science, Vol.1050. – Springer – Verlag, 1996.
[21] Mousan M. et al. Aspects + GAMMA = AspectGAMMA: A Formal Framework for Aspect-Oriented Specification. – www.dre.vanderbilt.edu/~schmidt/pdf/aop-find.pdf
[22] Banatre J.-P., Fradet P., Radenne Y. Principles of Chemical Programming. – Tltctronic Notes in Theoretical Computer Science, 2004.
[23] Banatre J.-P., Fradet P., Radenne Y. Generalized Multisets for Chemical Programming. – Rep. №5743, INRIA, 2005.
[24] Shin S.W. Compiling and Verifying DNA-Based Chemical Reactions Network Implementation. – Californian Institute of Technology, 2011.
[25] Thachuk C. Space and Energy Efficient Molecular Programming. – University of British Columbia, 2012.
[26] Marriott K. Constraint Multiset Grammars. – Proc. IEEE Symposium on Visual Languages. – IEEE Computer Society Press, 1994.
[27] Marriott K., Meyer B. On the Classification of Visual Languages by Grammar Hierarchies. – Journal of Visual Languages and Computing, Vol. 8 (1997), pp. 375-402.
[28] Marriott K. Parsing Visual Languages with Constraint Multiset Grammars. – In: Programming Languages: Implementation, Logic and Programs. – Lecture Notes.
[29] Marriott K., Stucky P.G. Programming with Constraints: An Introduction. – MIT Press, 1998.
[30] Apt K. Principles of Constraints Programming. – Cambridge University Press, 2003.
[31] Frunkwirth T., Abdennadher S. Essentials of Constraints Programming. – Berlin. – Springer–Verlag, 2003.
[32] Sheremet I.A. Recursive Multisets and Their Applications. – Moscow: Nauka, 2010 (in Russian).
[33] Sheremet I.A. Recursive Multisets and Their Applications. – Berlin: EANS, 2011.
[34] Moll R.N., Arbib N.A., Kfoury A.J. An Introduction to Formal Languages Theory. – Springer – Verlag, 1988.
[35] Chomsky N. Syntactic Structures (2d Edition). – Berlin–New-York: Mouton de Gruyter, 2005.
[36] Connolly T.M., Begg C.E. Database Systems: A Practical Approach and Design, Implementation and Management (5th Edition). – Addison Wesley, 2009.
[37] Ullman J. Principles of Database Systems. – IK International Publishing House, 2011.
[38] Williams H.P. Logic and Integer Programming. – Springer, 2009.