Chapter

Multi-Agent Implementation of Filtering Multiset Grammars

Igor Sheremet

Abstract

Chapter is dedicated to the application of multi-agent technology to generation of sets of terminal multisets (*TMS*) defined by filtering multiset grammars (*FMG*). Proposed approach is based on creation of multi-agent system (*MAS*), corresponding to specific *FMG* in such a way, that every rule of *FMG* is represented by independently acting agent. Such *MAS* provides high-parallel generation of *TMS* and may be effectively used in any proper hardware environment. Directions of further development of the proposed approach are discussed.

Keywords: multi-agent systems and technologies, multisets, multiset grammars, filtering multiset grammars, parallel computations

1. Introduction

Filtering multiset grammars (*FMG*) were developed as a result of multiset-based deep integration and convergence of classical mathematical programming and modern knowledge engineering for compact, flexible and natural representation of wide spectrum of combinatorial problems and their solution by application of unified algorithmics [1–6].

One of the advantages of *FMG* is natural parallelism of generation of multisets (*MS*) due to the possibility of independent application of rules to the currently available *MS*. Such feature being supported by appropriate hardware is a very promising background for the effective implementation of *FMG* and hence effective solution of the aforementioned problems. However, the "brutal force" approach, demanding on the extensive parallelism, is not suitable here because of evident cost restrictions. More attractive looks such techniques which would be based on the "branches and bounds" logics providing cut off sets of multisets (*SMS*) which provably do not contain terminal multisets (*TMS*), defined by *FMG*, without their generation.

To develop such perspective approach we propose to apply multi-agent technology (*MAT*) [7–12] as a basis for implementation of the aforementioned techniques. The main idea of the suggested method of *TMS* generation is representation of the generating engine as a multi-agent system (*MAS*), which includes:

N agents, each corresponding to one rule from *FMG* scheme;
one agent corresponding to *FMG* filter;
one supervising agent, accumulating generated *TMS*, satisfying filter.

This *MAS* operates in such a way that every rule is applied (i.e. agent becomes active) as soon as there occurs multiset, matching this rule. By this approach maximally possible degree of parallelism is achieved.

Structure of the chapter is as follows. Section 2 contains main definitions and notions of the multigrammatical framework, necessary for further considerations. Proposed techniques of the multi-agent implementation of multisets generation is introduced in Section 3. Directions of future development of these techniques are discussed in the conclusion.

2. Basic notions and definitions of the multigrammatical framework

Classical theory of sets is based on notion of set as an unordered collection of mutually distinguishable elements. Basic assumption of theory of multisets is that aforementioned collection may contain indistinguishable (identical) elements:

$$v = \left\{ \underbrace{a_1, \ldots, a_1}_{n_1 \text{ times}}, \ldots, \underbrace{a_i, \ldots, a_i}_{n_i \text{ times}}, \ldots, \underbrace{a_m, \ldots, a_m}_{n_m \text{ times}} \right\}. \tag{1}$$

Record (1) is usually represented as

$$v = \{n_1 \cdot a_1, \ldots, n_m \cdot a_m\}, \tag{2}$$

where v is called *multiset*, $n_i \cdot a_i$ – *multiobjects* (MO), a_i – *objects*, n_i– their *multiplicities*, for all $i = 1, \ldots, m$. According to (2), multiset may be considered as a set of multiobjects, and, in fact, multiset $\{1 \cdot a_1, \ldots, 1 \cdot a_m\}$ and set $\{a_1, \ldots, a_m\}$ represent one and the same collection. Set$\{a_1, \ldots, a_m\}$, denoted as $\beta(v)$, is called *basis* of multiset v. Both empty multiset and empty set are denoted as $\{\emptyset\}$.

Zero multiplicity of some object is equivalent to the absence of this object in multiset, i.e.

$$\{n_1 \cdot a_1, \ldots, n_m \cdot a_m, 0 \cdot a_{m+1}\} = \{n_1 \cdot a_1, \ldots, n_m \cdot a_m\}. \tag{3}$$

Fact, that object a *enters MS* v (or *MS* v *includes* object a), is denoted as $a \in v$. Symbol " $\in$ " is also used to denote, that MO $n \cdot a$ enters *MS* v (MSv includes MO $n \cdot a$):$n \cdot a \in v$. Structure of the left operand determines what kind of relation is referred in every particular case. Similarly, symbol " $\notin$ " is used to denote, that object a (multiobject $n \cdot a$) does not enter multiset v. Due to (3), $a \notin v$ and $0 \cdot a \in v$ are equivalent.

Number $|v| = m$ is called *dimensionality* of *MS* v, and number

$$|v| = \sum_{i=1}^{m} n_i, \tag{4}$$

is called *power* of *MS* v.

Two basic relations on multisets – *inclusion* (" $\subseteq$ ") and *strict inclusion* (" $\subset$ ") – are defined as follows.

MS v is *included* to *MS* v', if

$$(\forall n \cdot a \in v)(\exists n' \cdot a \in v')\ n \leq n', \tag{5}$$

i.e. for every MO $n \cdot a$ entering *MS* v there exists $MOn' \cdot a$, which multiplicity n' is not less than n. There may be also $n' \cdot a' \in v'$ such that $a' \notin v$ (as seen, this does not contradict (5), because$0 \cdot a' \in v$ and $0 < n'$).

If $v \subseteq v'$ and $v \neq v'$, then *MS* v is *strictly included* to *MS* v', that is denoted $v \subset v'$.

MS v, which is included to *MS* v', is called *submultiset* of *MS* v'; *MS* v, which is strictly included to *MS* v', is called *strict submultiset* of *MS* v'.

Let us illustrate introduced notions by the following example, where, as in all other examples, having place in this chapter, objects will be represented by strings in brackets.

Example 1. Let

$$v = \{1 \cdot (eur), 5 \cdot (usd), 12 \cdot (rur)\},$$
$$v' = \{6 \cdot (eur), 5 \cdot (usd), 12 \cdot (rur)\}.$$

As seen, according to (5), $v \subseteq v'$ and $v \subset v'$.■

Three basic operations on multisets are *multiplication by a constant*, *addition*, and *subtraction*, which are denoted by bold symbols $*$, $+$ and $-$ respectively. Semantics of these operations is defined by use of the well known set-theoretical operations (join and intersection), as well as arithmetic operations on integer numbers:

$$n * \{n_1 \cdot a_1, \ldots, n_m \cdot a_m\} = \{(n \times n_1) \cdot a_1, \ldots, (n \times n_m) \cdot a_m\}, \tag{6}$$

where "$\times$" denotes multiplication of integer numbers;

$$v + v' = \bigcup_{\substack{a \in \beta(v) \cup \beta(v') \\ n \cdot a \epsilon v \\ n' \cdot a \in v'}} \{(n + n') \cdot a\}, \tag{7}$$

$$v - v' = \bigcup_{\substack{a \in \beta(v) \cup \beta(v') \\ n \cdot a \epsilon v \\ n' \cdot a \in v' \\ n > n'}} \{(n - n') \cdot a\}. \tag{8}$$

Along with these operations, we shall use set-theoretical operations on multisets – *join* and *intersection*, – denoted respectively by bold symbols $\cup$ and $\cap$, different from $\cup$ and $\cap$:

$$v \cup v' = \bigcup_{\substack{a \in \beta(v) \cup \beta(v') \\ n \cdot a \epsilon v \\ n' \cdot a \in v'}} \{\max\{n, n'\} \cdot a\}, \tag{9}$$

$$v \cap v' = \bigcup_{\substack{a \in \beta(v) \cup \beta(v') \\ n \cdot a \epsilon v \\ n' \cdot a \in v'}} \{\min\{n, n'\} \cdot a\}, \tag{10}$$

We have used equivalence between $a \notin v$ and $0 \cdot a \in v$ in (9).

Example 2. Let v and v' be as in **Example 1**. Then

$$3 * v = \{3 \cdot (eur), 15 \cdot (usd), 36 \cdot (eur)\},$$
$$v + v' = \{7 \cdot (eur), 10 \cdot (usd), 24 \cdot (rur)\},$$
$$v - v' = \{\emptyset\},$$
$$v' - v = \{5 \cdot (eur)\},$$

$$v \cup v' = \{6 \cdot (eur), 5 \cdot (usd), 12 \cdot (rur)\},$$
$$v \cap v' = \{1 \cdot (eur), 5 \cdot (usd), 12 \cdot (rur)\}.\blacksquare$$

Common feature of all described operations, which are known from theory of multisets [13, 14], is that their operands are multisets and integer numbers. Unlike them, following operation called "*filtration*" applies set of multisets (*SMS*) as the first operand and so called "*filter*", being set of *conditions*, as the second operand. Filtration is denoted as $V \downarrow F$, where V is filtrated *SMS*, F is filter, and "$\downarrow$" – symbol of operation.

Conditions entering filter F may be boundary and optimizing.

Boundary conditions (*BC*) are recorded as $a\theta n$, where $\theta \in \{>, <, \geq, \leq, =\}$. Multiset v satisfies *BC* $a\theta n$, if $m \cdot a \in v$ and $m\theta n$ is true (as everywhere, $a \notin v$ is equivalent to $0 \cdot a \in v$).

Let $F_{\leq} = \{bc_1, \ldots, bc_k\}$ be a filter, containing only boundary conditions. Then

$$V \downarrow F_{\leq} = \bigcap_{i=1}^{k} (V \downarrow \{bc_i\}). \tag{11}$$

Example 3. Let $V = \{v_1, v_2, v_3\}$, where

$$v_1 = \{4 \cdot (eur), 3 \cdot (rur)\},$$
$$v_2 = \{8 \cdot (usd), 10 \cdot (rur)\},$$
$$v_3 = \{3 \cdot (eur), 19 \cdot (usd), 17 \cdot (rur)\},$$

and $F_{\leq} = \{(usd) > 3, (rur) \leq 12\}$. Then

$$V \downarrow F_{\leq} = (V \downarrow \{(usd) > 3\}) \cap (V \downarrow \{(rur) \leq 12\}) = \blacksquare$$
$$= \{v_2, v_3\} \cap \{v_1, v_2\} = \{v_2\}.$$

Optimizing conditions (*OC*) are recorded as $a = opt$, where $opt \in \{max, min\}$. Multiset $v \in V$ satisfies *OC* $a = max$, if $n \cdot a \in v$ and all multisets $v' \in V - \{v\}$ satisfy boundary condition $a \leq n$. Similarly, multiset $v \in V$ satisfies *OC* $a = min$, if $n \cdot a \in v$ and all multisets $v' \in V - \{v\}$ satisfy boundary condition $a \geq n$.

Let $F_{opt} = \{oc_1, \ldots, oc_l\}$ be filter, containing only optimizing conditions. Then, similarly to (6),

$$V \downarrow F_{opt} = \bigcap_{i=1}^{l} (V \downarrow \{oc_i\}). \tag{12}$$

Example 4. Let V be the same as in Example 3, and $F_{opt} = \{(usd) = max, (rur) = min\}$. Then

$$V \downarrow F_{opt} = (v \downarrow \{(usd) = max)\}) \cap (v \downarrow \{(rur) = min\}) = \{v_3\} \cap \{v_1\} = \{\emptyset\}.\blacksquare$$

If filter F contains both boundary and optimizing conditions, i.e.

$$F = F_{\leq} \cup F_{opt}, \tag{13}$$

then

$$V \downarrow F = (V \downarrow F_{\leq}) \downarrow F_{opt}, \tag{14}$$

i.e. result of filtering set of multisets V by filter F is obtained by application of boundary subfilter $F_{\leq}$ to V, after what resulting *SMS* is filtered by optimizing subfilter F_{opt}.

Application of filters, containing boundary and optimizing conditions, is illustrated by **Figure 1**.

Considered operations on multisets and their sets make it possible to define syntax and semantics of family of multiset grammars.

Background of this family is notion of *multiset grammar* as a couple $S = \langle v_0, R \rangle$, where v_0 called *kernel* is multiset, and R called *scheme* is finite set of so called rules. Set of all objects used in kernel and scheme of *MG* S is denoted as A_S.

Rule $r \in R$ is a construction

$$v \to v', \tag{15}$$

where multisets v and v' are called respectively *left part* and *right part* of the rule r, and "$\to$" is divider. The only restriction on left and right parts of the rule is$v \neq \{\emptyset\}$.

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

$$\boldsymbol{v}' = \boldsymbol{v} - v + v'. \tag{16}$$

Speaking informally, (16) defines, that if left part of the rule, i.e. multiset v, is included to *MS* $\boldsymbol{v}$, then v is replaced by right part of this rule, i.e. multiset v'. Result of application of rule r to multiset $\boldsymbol{v}$ is denoted as

$$\boldsymbol{v} \overset{r}{\Rightarrow} \boldsymbol{v}', \tag{17}$$

and it is said, that *MS* $\boldsymbol{v}'$ is generated from *MS* $\boldsymbol{v}$ by application of rule r. If left part v is not included to *MS* $\boldsymbol{v}$, result of application r to $\boldsymbol{v}$ is empty *MS* $\{\emptyset\}$.

Set of multisets, generated by application of multigrammar $S = \langle v_0, R \rangle$, or, just the same, *defined by this multigrammar*, is recursively created as follows:

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

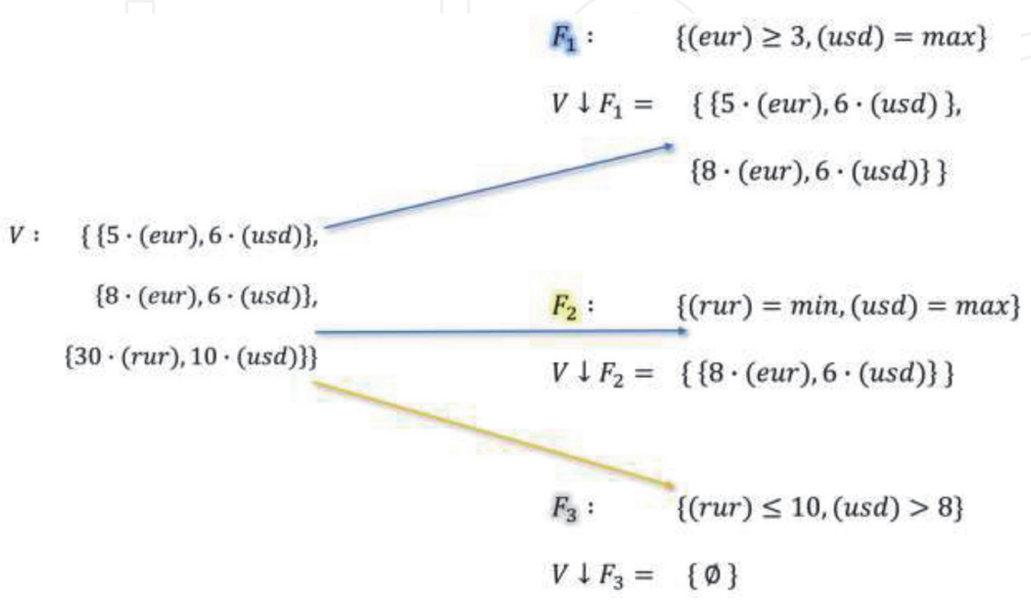

Figure 1.
Filters application.

$$V_{(i+1)} = V_{(i)} \bigcup \left(\bigcup_{\boldsymbol{v} \in V_{(i)}} \bigcup_{r \in R} \left\{ \boldsymbol{v}' \middle| \boldsymbol{v} \overset{r}{\Rightarrow} \boldsymbol{v}' \right\} \right), \tag{19}$$

$$V_S = V_{(\infty)}. \tag{20}$$

As seen, V_S includes all multisets, which may be generated from *MS* v_0 by sequential application of rules $r \in R$, and V_S is fixed point of the sequence $V_{(0)}, V_{(1)}, \ldots, V_{(i)}, \ldots,$ so

$$V_S = \bigcup_{i=0}^{\infty} V_{(i)}. \tag{21}$$

In general case V_S may be infinite.

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

$$\boldsymbol{v} \overset{R}{\Rightarrow} \boldsymbol{v}', \tag{22}$$

and, if so,

$$V_S = \left\{ \boldsymbol{v} \middle| v_0 \overset{R}{\Rightarrow} \boldsymbol{v} \right\}. \tag{23}$$

Multiset $\boldsymbol{v} \in V_S$ is called *terminal multiset* (*TMS*), if

$$(\forall r \in R)\ \boldsymbol{v} \overset{r}{\Rightarrow} \{\emptyset\}, \tag{24}$$

i.e. no any rule $r \in R$ may be applied to this multiset. Set of terminal multisets (*STMS*) defined by multiset grammar S is denoted $\overline{V_S}$. Evidently,

$$\overline{V_S} \subseteq V_S. \tag{25}$$

Example 5. Let $S = \langle v_0, R \rangle$, where kernel

$$v_0 = \{3 \cdot (eur), 6 \cdot (usd), 5 \cdot (rur)\},$$

and scheme $R = \{r_1, r_2\}$, where r_1 is

$$\{2 \cdot (eur)\} \rightarrow \{4 \cdot (usd)\},$$

and r_2 is

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

According to (18)–(19),

$V_{(0)} = \{\{3 \cdot (eur), 6 \cdot (usd), 5 \cdot (rur)\}\},$
$V_{(1)} = V_{(0)} \cdot \cup \{\{1 \cdot (eur), 10 \cdot (usd), 5 \cdot (rur)\}\}, \{5 \cdot (eur), 4 \cdot (usd), 2 \cdot (rur)\},$
$V_{(2)} = V_{(1)} \cup \{\{3 \cdot (eur), 8 \cdot (usd), 2 \cdot (rur)\}\},$
...

As seen, this *MG* provides generation of all possible collections of Euros, US dollars, and Russian rubles, which may be obtained from the initial collection v_0 by

sequential currency exchanges, which parameters are fixed by rules r_1 and r_2 (2 Euros may be exchanged to 4 US dollars, 2 US dollars and 3 Russian rubles may be exchanged to 2 Euros).■

Generalized scheme of application of multiset grammars is presented at **Figure 2**.

Due to the fact, that multiobjects contain both numerical and symbolic components (multiplicities and object names), generation of multisets provides knowledge-driven numerical computation, that creates a lot of new opportunities for simple formalizing and effective solution of various sophisticated practical problems with hard combinatorial background. To implement such opportunities, so called filtering multiset grammars were proposed in [1, 2]. *FMG* are such generalization of *MG*, that integrate two basic concepts—generation of set of multisets and selection from it such *MS*, that satisfy some logical conditions, joined to filter.

Filtering multiset grammar is a triple $S = \langle v_0, R, F \rangle$, where v_0 and R are, as above, kernel and scheme, while F is a filter, including boundary and optimizing conditions, defining multisets, which would be selected from the set of *TMS*, generated by *MG* $\langle v\_0, R \rangle$, i.e.

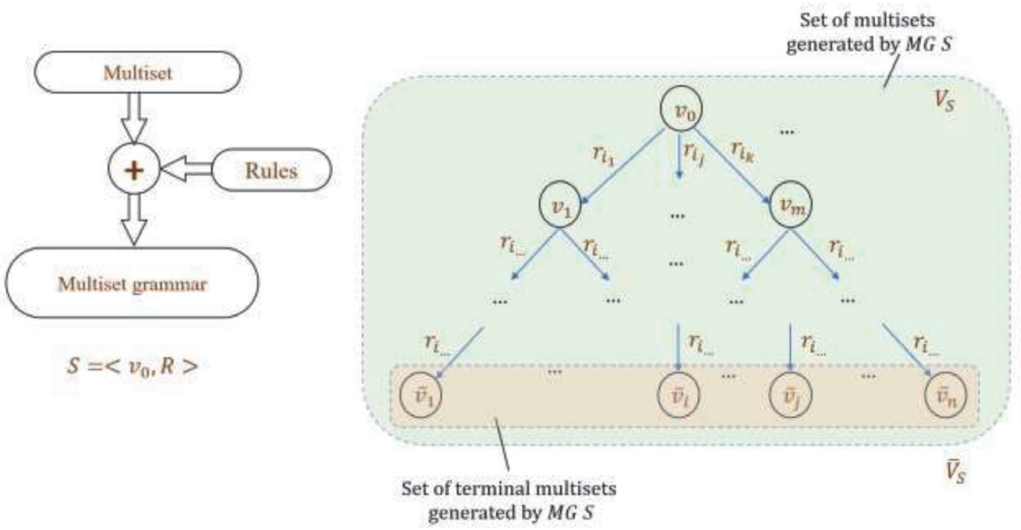

Figure 2.
Application of multiset grammars.

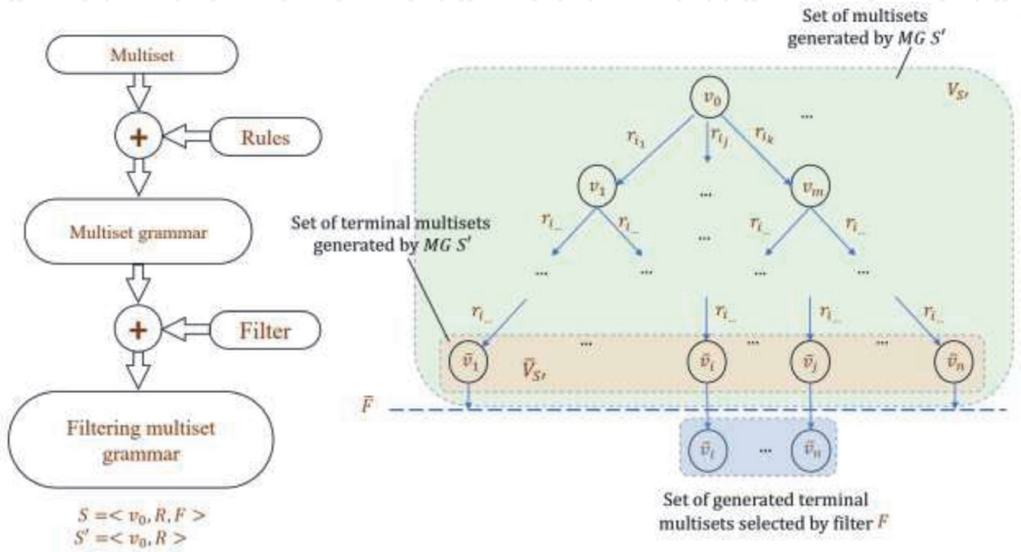

Figure 3.
Application of filtering multiset grammars.

$$V_s = \overline{V}_{\langle v_0,R\rangle} \downarrow F. \tag{26}$$

Verbally, V_s is subset of $\overline{V}_{\langle v_0,R\rangle}$, which includes only such elements of this set, that satisfy filter F. Generalized scheme of application of filtering multiset grammars is presented at **Figure 3**.

After description of syntax and semantics of filtering multiset grammars we may move to their implementation issues, which background are multi-agent technologies.

3. Basic techniques of the multi-agent implementation of multisets generation

Due to granularity and natural internal parallelism of multigrammatical representation it is perspective to try to use multi-agent paradigm as a background for implementation of the *FMG* application engine (*AE*), providing generation of multisets.

We shall use scheme, depicted at **Figure 4**, as a basis for primary multi-agent implementation of *FMG AE*. Proposed multi-agent system implementing *FMG* $S = \langle v_0, R, F\rangle$, contains $l = |R|$ agents $r_1, \ldots, r_l$, each corresponding to one rule from set R (everywhere below we shall denote rules and implementing them agents by the same symbols) ; one agent F^*, implementing filter F; one agent G^*, providing supervision on other agents interaction (in fact, implementing ubiquitous generation by application of rules to multisets, generated at previous steps). Agent G^* uses storage V_S for accumulation of all generated terminal multisets, satisfying filter F. Also G^* operates storage V, containing current set of generated multisets, which are not yet transferred to other agents. Initial state of V is $\{v_0\}$. Agents communicate via network N.

Described multi-agent system is operating according to the definition of mathematical semantics of *FMG* (15)–(26). Set of messages, which are circulating between the agents, is represented in **Table 1**.

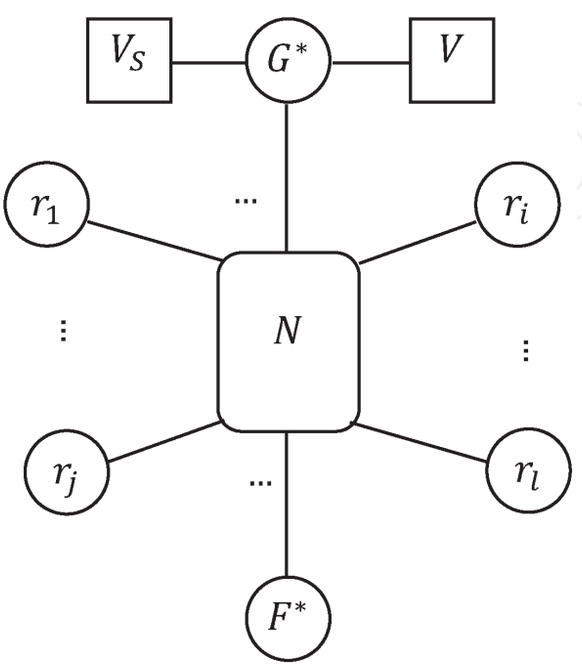

Figure 4.
Multi-agent system implementing FMG.

N	Sender	Receiver	Message	Comment
1	G^*	r_i	$\langle j, v\rangle$	j – number of *MS*,$v \neq \{\emptyset\}$
2	r_i	G^*	$\langle j, v'\rangle$	v' – result of application of rule r_i to *MS* v
3	r_i	G^*	$\langle j, \{\emptyset\}\rangle$	Rule r_i is not applicable to v
4	G^*	$\langle j, v\rangle$,	$\langle j, v\rangle$	As 1
5	F^*	G^*	$\langle j, 1\rangle$	jth *MS* satisfies filter F
6	F^*	G^*	$\langle j, 0\rangle$	jth *MS* does not satisfy filter F

Table 1.
Set of MAS messages.

All messages are couples, the first components of which are numbers of multisets, processed by the agents, and every *MS* has its unique number. Assigning numbers to multisets is performed by agent G^*. Current maximal number is denoted J. Also agent G^* uses variable Z, which value is set of couples $\langle j, n\rangle$, where n is number of agents r_i which until current moment have not sent to G^* messages with results of application of corresponding rule to jth *MS*.

Agent G^* sends couple $\langle j, v\rangle$ to all agents r_i. After this, G^* joins to the current value of variable Z couple $\langle j, l\rangle$. Every agent r_i, receiving message $\langle j, v\rangle$ from agent G^*, tries to apply v to rule r_i. If application is possible, r_i sends to G^* message j, v', where v' is result of application of rule r_i to *MS* v. Otherwise r_i sends to G^* message $\langle j, \{\emptyset\}\rangle$. Agent G^*, receiving message $\langle j, v\rangle$, where $v \neq \{\emptyset\}$, assigns J new value $J + 1$, and sends message $\langle J, v\rangle$ to all agents. Also couple $\langle J, l\rangle$ is joined to Z, and couple $\langle j, q\rangle \in Z$ is eliminated from set Z, that means at least one rule was applied to jth *MS*, so this multiset is non-terminal. If agent G^* receives message $\langle j, \{\emptyset\}\rangle$ from r_i, that means rule r_i was not applied to jth multiset, and $\langle j, q\rangle \in Z$ is replaced by $\langle j, q - 1\rangle$. If now $\langle j, 0\rangle \in Z$, that means no one rule was applied to jth *MS*, that's why it is terminal, and, according to *FMG* semantics, it must be filtered. So agent G^* sends couple $\langle j, v\rangle$ to agent F^*, which provides testing, whether v satisfies boundary subfilter $F_{\leq} \subseteq F$. If testing is successful, agent F^* sends to agent G^* message $\langle j, 1\rangle$, otherwise—message $\langle j, 0\rangle$. In the case $\langle j, 1\rangle$ couple $\langle j, v\rangle$ is joined to the current value of variable V_s. After no active agents r_i remain, agent G^* applies optimizing subfilter $F_{opt} \subseteq F$ to the aforementioned current value V_s, eliminating from it all multisets, which do not satisfy F_{opt}. Final value of variable V_s is exactly set of terminal multisets, defined by *FMG* $S = \langle v_0, R, F\rangle$.

As may be seen, proposed multi-agent system provides generation and filtration of multisets by parallel operation of all agents, entering this *MAS*.

However, there are some evident bottlenecks, limiting speed of multisets generation. Most essential of such bottlenecks are massive transmissions of multiply repeated sets of multiobjects via *MAS* communication network. To reduce such transmission it is possible not to send *MO* to agents r_i such, that corresponding them rules are applicable to multisets, having place in the storage V, and this applicability may be recognized directly by agent G^*. If such opportunity may be implemented, only those agents r_i, which, possibly, may apply corresponding rules to current *MS*, would receive it. By this measure, traffic on *MAS* communication network may be reduced sharply.

To implement proposed logics, we shall introduce auxiliary database L, which elements would be couples $\langle a, \{\langle i_1, n_1\rangle, \ldots, \langle i_k, n_k\rangle\}\rangle$, where a is name of object, and in couples $\langle i, n\rangle$, entering the set, which is second component of the couple, integer i is number of the rule, which left part contains multiobject $n \cdot a$.

Example 6. Let scheme $R = \{r_1, r_2\}$, where r_1 is

$$\{9 \cdot (eur)\} \rightarrow \{10 \cdot (usd)\},$$

and r_2 is

$$\{5 \cdot (eur), 3 \cdot (rur)\} \rightarrow \{7 \cdot (usd)\}.$$

Then $L = \{ <(eur), \{ <1, 9>, <2, 5> \} >, <(rur), \{ <2, 3> \} > \}$.■

Database L has such internal organization, that there exists associative index, providing direct selection of couple $\langle a, w \rangle$ for object name a. To reduce search in the selected list of couples $\langle i, n \rangle$, it may be created as ordered by increase of multiplicities n. Let $v = \{n_1 \cdot a_1, \ldots, n_m \cdot a_m\}$ be a current multiset processed by agent G^*, and $Q = \{\langle a_1, \leq n_1 \rangle, \ldots, \langle a_m, \leq n_m \rangle\}$ is set of queries to database L, each providing selection of couples $\left\langle a_i, \left\{ j_1^i, \ldots, j_{k_i}^i \right\} \right\rangle$, such that $n_p^i \leq n_i$. It is clear, that only in this case rules $r_{j_1^i}, \ldots, r_{j_{k_i}^i}$ may have opportunity to be applied to *MS* v, and, totally, only those rules, in which all multiobjects from their left parts have multiplicities hot greater than those of the same objects having place in v. So there is an evident criterion for selection of rules, which may be applicable to the current multiset v.

Statement 1. Let $\left\{ \langle a_{i_1}, N_1 \rangle, \ldots, \left\langle a_{i_p}, N_p \right\rangle \right\}$ be a set, selected from database L by query $Q = \{\langle a_1, \leq n_1 \rangle, \ldots, \langle a_m, \leq n_m \rangle\}$, corresponding to multiset $v = \{n_1 \cdot a_1, \ldots, n_m \cdot a_m\}$, and $\{n_{j_1} \cdot a_{j_1}, \ldots, n_{j_s} \cdot a_{j_s}\}$ be the left part of rule r. Then r may be applicable to v, if

$$\{a_{j_1}, \ldots, a_{j_s}\} \subseteq \left\{a_{j_1}, \ldots, a_{j_p}\right\}.\blacksquare \tag{27}$$

As seen, proposed associative organization of set of left parts of rules entering scheme R provides fast selection of sets of rules, which may be applied to the current multiset.

Let us consider further enhancement of *FMG* application engine, based on the multi-agent technology.

First of all, it is evident, that it is not necessary to send all multiobjects of multiset v, to which ruler_i is applicable, to agent r_i, because replacement of left part of this rule by its right part is local operation, regarding in general case relatively small number of multiobjects in the processed multiset v, while all the rest *MO* remain unchanged. So it is sufficient to send to agent r_i only tuple $\langle f_1 n_{i_1}, \ldots, f_t n_{i_t} \rangle$ of multiplicities of objects $a_{i_1}, \ldots, a_{i_t}$, in *MS* v, such that tuple $A = \langle a_{i_1}, \ldots, a_{i_t} \rangle$ is ordered lexicographically set of objects, having place in both left and right parts of rule r, and signs f_i before multiplicities $n_{i_j}$of objects a_{i_j}, having place in left side of rule r, are "–", while all other are "+". Receiving this tuple, agent r provides computation of tuple $\langle \boldsymbol{n_{i_1}}, \ldots, \boldsymbol{n_{i_t}} \rangle$, where $\boldsymbol{n_j} = n_j + \left(f_j n_{i_j} \right), j = 1, \ldots, t$. (There may be particular case, when some objects do enter both left and right parts but this singularity is simply handled by positioning to the jth place of tuple A number $-n + n'$, where n is multiplicity of object a_{i_j} in the left part, and n'–its multiplicity in the right part of rule r).

Example 7. Let $v = \{5 \cdot (eur), 10 \cdot (usd), 7 \cdot (rur), 18 \cdot (pound)\}$, and r is $\{3 \cdot (eur), 2 \cdot (rur)\} \rightarrow \{5 \cdot (usd)\}$. Because set of lexemes, entering this rule, is ordered lexicographically as $\langle eur, rur, usd, \rangle$ so tuple, sent by agent G^* to agent r, would be$\langle 5, 7, 10, \rangle$ and agent r would sent to agent G^* tuple $\langle 5-3, 7-2, 10+5 \rangle = \langle 2, 5, 15 \rangle$, and, thus, result of $v \overset{r}{\Rightarrow} v'$ would be $v' = \{2 \cdot (eur), 15 \cdot (usd), 5 \cdot (rur),$

$18 \cdot (pound)\}$. As seen, multiplicity of object $(pound)$ is not transferred to agent r, because this object does not enter rule r.■

Proposed techniques provides further reduction of traffic on *MAS* communication network and, thus, total time of generation of *STMS*, defined by *FMG*.

Application of the described *MAS*-based generation of *STMS* is flexible as it is only possible: due to granularity of multigrammatical knowledge representation local corrections of *FMG* by replacement of one rules by another are easily reflected by corresponding replacement of only concerned agents without touching the other. The same may be done with *FMG* filters. Such flexibility provides the simplest implementation of the most practically useful "what-if" regimes of application of *MG*-centered knowledge-based decision support systems.

4. Conclusion

Proposed techniques of application of multi-agent technology to the high-parallel generation of sets of multisets, defined by filtering multiset grammars, provides essential growth of speed of creation of *STMS*. However, there are some evident ways of further enhancement of *FMG* implementation upon this background:

- development of methods of matching constructed *MAS* to real homogeneous or heterogeneous hardware in such a way that delays, caused by information exchange between agents, would be minimal;

- development of more efficient *MAT* application techniques concerning some more particular cases of *MG* – first of all, filtering context-free multiset grammars as well as filtering unitary *MG* and unitary multiset metagrammars;

- design of special-purpose computing environments suitable for direct implementation of *FMG* and other dialects of *MG* family;

- development of *MAT*-based techniques of implementation of algorithmics of unitary multiset metagrammars as most practically useful tool of the *MG* family.

Acknowledgements

Author is grateful to Prof. Fred Roberts and Prof. Alexey Gvishiani for useful discussions.

Author details

Igor Sheremet
Russian Foundation for Basic Research, Moscow, Russia

*Address all correspondence to: sheremet@rfbr.ru

References

[1] Sheremet IA. Recursive Multisets and their Applications. Moscow: Nauka; 2010. p. 292 (in Russian)

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

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

[4] Sheremet I. Multiset-based knowledge representation for the assessment and optimization of large-scale sociotechnical Systems. In: Vizureanu P, editor. Enhanced Expert Systems. London: IntechOpen; 2019. DOI: 10.5772/intechopen.81698. Available from: https://www.intechopen.com/books/enhanced-expert-systems/multiset-based-knowledge-representation-for-the-assessment-and-optimization-of-large-scale-sociotech

[5] Sheremet I. Unitary multiset grammars and metagrammars algorithmics and applications. In: Vizureanu P, editor. Enhanced Expert Systems. London: IntechOpen; 2019. DOI: 10.5772/intechopen.82713. Available from: https://www.intechopen.com/books/enhanced-expert-systems/unitary-multiset-grammars-an-metagrammars-algorithmics-and-application

[6] Sheremet I. Multiset-based assessment of resilience of sociotechnological systems to natural hazards. In: Tiefenbacher J, editor. Natural Hazards—Risks, Exposure, Response, and Resilience. London: IntechOpen; 2019. DOI: 10.5772/intechopen.83508. Available from: https://www.intechopen.com/online-first/multiset-based-assessment-of-resilience-of-sociotechnological-systems-to-natural-hazards

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

[8] Schatten M, Tomicic I, Duric BO. Multi-agent modeling methods for massively multi-player on-line role-playing games. In: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). 2015

[9] Waldrop MM. Free agents: Monumentally complex models are gaming out disaster scenarios with millions of simulated people. Science. 2018;**360**(6385):144-147

[10] Sycara KP. Multiagent systems. AI Magazine. 1998;**19**(2):79-92

[11] Yeoh W, Yokoo M. Distributed problem solving. AI Magazine. 2012; **33**(3):53-65

[12] Van der Hoog S. Deep learning in (and of) agent-based models: A prospectus. 2018. ArXiv: 1706.06302

[13] Petrovskiy AB. Spaces of Sets and Multisets. Moscow: Editorial URSS; 2003. p. 248 (in Russian)

[14] Petrovskiy AB. Theory of Measured Sets and Multisets. Moscow: Nauka; 2018. p. 360 (in Russian)