Data and Knowledge Bases with Incomplete Information in a "Set of Strings" Framework

Igor Sheremet

***Abstract*— Incompleteness of information about monitored objects is immanent property of many information processing systems. That is why databases with incomplete information (DBI) modeling is constantly actual area of the theoretical computer science as well as foundation of efficiency of DBI-centered software and hardware. This article describes an approach to such modeling, the main feature of which is representation of the DBI as a finite set of so-called incomplete (N-) facts being sentential forms (SF) of context-free grammar, which set of rules form metadatabase (MDB). Relation of the mutual informativity on the N-facts set is defined, and set-theoretical, mathematical and operational semantics of the DBI data manipulation language (DML) are described, as well as generalized model of the associative storage and access supporting N-facts handling. Augmented Post systems being knowledge representation model associated with mentioned DBI are introduced.**

***Index Terms*— databases, incomplete information modeling, context-free grammar, sentential form, data manipulation language, associative storage and access to incomplete information, augmented Post systems, logical inference with incomplete information.**

I. INTRODUCTION

Incompleteness of information about the objects of control is immanent property of any information processing systems, especially those which operate in rapidly changing conditions requiring regularly modified (corrected) processing logic. Such systems are usually implemented within the network-centric paradigm [1], [2], and their control centers are processing flows of information from various sensors located in various areas and monitoring objects in various physical fields. Any sensor because of its location and technical capabilities can precisely recognize only a small part of the mentioned objects parameters, so fragments of information, which are transmitted from sensors to control centers, are principally incomplete. That's why databases, which are maintained and used by such systems and accumulate information about the state of the monitored objects, must be initially oriented to the entering, integration, storage, and update of incomplete information as usual. Such databases are called databases with incomplete information and may be used online (OLAP) as well as offline (Data Mining) – the last for actualization of the processing logics of the first. Let's consider two examples to illustrate the said above.

One of the most actual DBI applications during last time becomes Deep Packet Inspection/Deep Packet Processing (DPI/DPP) [3]–[5], which main task is early recognition of signatures of cyberattacks preparation and performance, by OLAP of computer networks perimeters traffic. The analyzed packets contain enciphered structural components usually, so both online and offline analysis of such traffic is based on the accumulation, update, and logical analysis of the stored incomplete information, i.e. DBI handling. DPI/DPP is core technology for Intrusion Detection/Prevention, Data Leakage Prevention as well as other essential cybersecurity technologies [6]-[10].

The second example of DBI use are systems monitoring real car traffic and revealing autos, which are on the police register by reasons of the road accidents, unpaid penalties, stealing and so on. This function implementation is based on the processing of autos sign images, which are transmitted to the processing centers from the video cameras located along the roads and the streets. By reasons of the bad weather, fog, symbols distortion (accidental or criminal), non-optimal mutual location of car and video camera, some structural components of the autos signs may be indefinite. That's why entering information flow contains incomplete information, and databases, which accumulate such information, are DBI.

Examples may follow, but there is a reason to pay attention to one more very essential obstacle. The information fragments, which enter processing, storage, and update in DBI, in general case are strings of symbols (bits) of the unpredicted structure. This structure partly is known before OLAP implementing and usually is corrected while Data Mining. This obstacle makes extremely sophisticated, if implementable, such fragments processing and accumulating by relational/post-relational DBMS–centered software as well as software based on the standard relational toolkits, which theoretical kernel are indefinite ("null", "unknown", "imprecise", "disjunctive", "probabilistic", "possibilistic" etc.) values of the attributes in the records (elements of relations) [11]–[24].

In these conditions the alternative approach to DBI modeling is necessary. This approach must comprehend the listed and many similar applications, and create theoretical basis for sophisticated analytical processing with dynamic, time-varying logic.

Article presented is dedicated to one of such approaches worked out within the so-called "Set-of-Strings" Framework (SSF). This framework is not model nor metamodel of data; rather it is a set of basic representations, mathematical relations and algorithms for strict formal description of the key aspects of the data and knowledge processing in various environments and modes (OLAP as well as Data Mining). SSF was suggested and developed by the author in [25]-[28], and during the past time have been soft- and hardware-implemented in various areas, one of the up-to-date being mentioned cybersecurity.

SSF kernel is integration of the best features of classical string-generating formal grammars proposed by N.Chomsky for natural language syntax description [29] and widely used

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

in modern computer science [30], as well as Post systems proposed by E.Post in [31]. The last are interpreted lower as string-operating logical calculus similar to Horn clauses well-known now due to its procedural interpretation [32], [33], which had lead to Prolog programming language and its various dialects [34]-[38].

By reason that SSF isn't well-known to wide circle of specialists, the second section of the article is dedicated to the short description of its main elements. In the third section the basics of information incompleteness modeling within the SSF are described. The fourth section is dedicated to the key elements of the incomplete (N-facts) fusion. Main content of the fifth section is N-facts processing algorithmics, while the sixth section is dedicated to DBI associative storage modeling and redundant search elimination while access operations execution. The seventh section describes augmented Post systems (APS) and incomplete information modeling in the frames of this knowledge representation.

II. BASIC ELEMENTS OF THE SET-OF-STRINGS FRAMEWORK

The starting point of the SSF is the representation of the database as a finite set of strings (we shall call it "set-of-strings DB", or SDB):

$$W_t = \{w_1, \dots, w_{m(t)}\} \subset V^*, \quad (1)$$

where W_t means DB at the discrete time moment t, and V^* is a set of all strings in the initial (terminal) alphabet V. The structure of the DB elements $w_i \in W_t$ named facts is determined by metadatabase, which is denoted by D_t and will be considered lower.

The couple

$$\Theta_t =< W_t, D_t > \quad (2)$$

is named data storage (DS). Data storage is in the correct state, if $W_t \in \boldsymbol{W}(D_t)$, where $\boldsymbol{W}(D_t)$ is a set of all correct databases, defined by the MDB.

Access message to DS is the triple

$$\omega_t =< o, c, x >, \quad (3)$$

where o is operation, which execution is the purpose of the access (insert, delete, update, query), c – DS component (DB, MDB), and x – content of access, i.e. query body, or elements (facts), which are inserted or deleted. It is supposed, that the answer (reply) to the access is obtained by the user at the moment $t+1$ next to t, and when $c = \mathrm{DB}$, it is the finite set A_{t+1} (in the case when $c = \mathrm{MDB}$ the answer is denoted A^D_{t+1}, and it is also finite set).

All possible triples of the form (3) create the DS manipulation language (DSML).

There are four types of semantics of the DSML within SSF: set-theoretical (S), mathematical (M), operational (O) and implementational (I).

S-semantics is formal definition of functions of the DBMS, which is processing accesses to DS, in the set theory language and is invariant to the specific DSMLs. It is used as a framework for development of DSML as such, i.e. its sentences.

M-semantics corresponds to S-semantics, but is described up to MDB and DSML models, which are based on some well-known and understandable mathematical constructions.

O-semantics corresponds to M-semantics, but unlike the last it is defined algorithmically. Algorithm of DSML messages processing should produce result by the finite number of steps in full accordance with expressions of M-semantics, or detect the impossibility of it due to the errors in the access message.

I-semantics is based on O-semantics and corresponds to the last one in the matter of the result of access message interpretation. However, kernel of O-semantics definition are considerations of algorithmical solvability; at the same time, while defining I-semantics, special attention is given to efficient implementation, i.e. minimization of average time of processing access messages (or, at more abstract level, minimization of computational complexity of the corresponding algorithm).

Consider set-theoretical semantics of the DSML sublanguage providing access to DB, which, according to the canonical data bases terminology, will be addressed lower as data manipulation language (DML). Basic DML S-semantics supposition is that content (substantial component) of the access message w_t is finite description of the set $I_t \subseteq V^*$, which corresponds to the user's knowledge about problem area segment he (or she) is interested (is aware of). DML S-semantics expressions connect W_{t+1} (database after operation execution) and A_{t+1} (answer to the access) with the set I_t and set W_t (database at the moment of access to DB). Starting from the ordinary sense of the operations over DB, we have following expressions (basic SSF equations):

$$W_{t+1} = W_t \cup I_t, \quad (4)$$

$$A_{t+1} = W_{t+1} - W_t, \quad (5)$$

for insertion (speaking more precisely, inclusion),

$$W_{t+1} = W_t - I_t, \quad (6)$$

$$A_{t+1} = W_t - W_{t+1}, \quad (7)$$

for deletion (exclusion), and

$$W_{t+1} = W_t, \quad (8)$$

$$A_{t+1} = W_t \cap I_t, \quad (9)$$

for query (everywhere " $-$ " is set-theoretical subtraction). In the expression (4) I_t postulates facts, which are actual in the problem area since moment t (here for W_t finiteness I_t must be finite too). In the expression (5) the answer contains "new" facts which "extend" DB W_t (facts, which already took place in the DB, are not included to A_{t+1}). In (7), on the contrary, facts, which are really excluded from DB, take place in the answer: I_t postulates facts, which are not actual since moment t in the problem area. Expression (8) reflects that DB remains constant after the query processing. The I_t set in (9) contains all facts, which actuality check is the purpose of the

query. At all the equations fact w once included to DB is staying actual until it is deleted from the database. In the expression (6) deletion mentioned takes place. Both in (6) and (9) I_t may be infinite.

Example 1. Consider database accumulating data from the distributed network of the ecological sensors, which are monitoring areas, where they are located. Let

$$W_t = \{AREA\ GREEN\ FOREST\ NORMAL\ AT\ 15.00,\ AREA\ LITTLE\ LAKE\ NORMAL\ AT\ 15.03,\ AREA\ HIGHLAND\ SMOKED\ AT\ 15.10\}, \quad (10)$$

that means two monitored areas are at normal state while one area is smoked. (Pay attention to the free structure of facts and arbitrary natural language use due to the basic "set of strings" assumption). If there occurs information from the sensor monitoring green forest, that at 15.10 this area is also smoked, inclusion

$$W_{t+1} = W_t \cup \{AREA\ GREEN\ FOREST\ SMOKED\ AT\ 15.10\} \quad (11)$$

is executed. When at this moment $t+1$ user accesses DB with query, which purpose is to get information about all smoked areas, the infinite set I_t may be as following:

$$I_{t+1} = \{AREA\ A\ SMOKED\ AT\ 00.00, \ldots, AREA\ A\ SMOKED\ AT\ 23.59, \ldots, AREA\ AA\ SMOKED\ AT\ 00.00, \ldots, AREA\ AA\ SMOKED\ AT\ 23.59, \ldots, AREA\ Z\ SMOKED\ AT\ 00.00, \ldots, AREA\ Z\ SMOKED\ AT\ 23.59, \ldots\} \quad (12)$$

The answer to the query is

$$A_{t+2} = W_{t+1} \cap I_{t+1} = \{AREA\ HIGHLAND\ SMOKED\ AT\ 15.10, AREA\ GREEN\ FOREST\ SMOKED\ AT\ 15.30\}. \quad (13)$$

In the expression (12) names of areas are all strings in the alphabet $V = \{A, \ldots, Z, 0, \ldots, 9, ., \}$, so

$$I_{t+1} = \{AREA\} \cdot V^* \cdot \{SMOKED\ AT\} \cdot \{00, \ldots, 23\} \cdot \{.\} \cdot \{00, \ldots, 59\}. \blacksquare \quad (14)$$

Note that definitions (4)–(9) are not unique. For example, in the inclusion definition elements of I_t set, having place in the DB at moment t, may be included to the answer

$$A_{t+1} = W_t \cap I_t, \quad (15)$$

as well as the answer may be defined as

$$A_{t+1} = \{FACT"\} \cdot (W_t \cap I_t) \cdot \{"ALREADY\ PRESENTS\ IN\ DATABASE\} \cup \{FACT"\} \cdot (W_{t+1} - W_t) \cdot \{"IS\ INCLUDED\ TO\ DATABASE\}. \quad (16)$$

So, according to (16), the answer to the access may be as follows:

$$A_{t+1} = \{FACT\ "AREA\ GREEN\ FOREST\ SMOKED\ AT\ 15.10"\ ALREADY\ PRESENTS\ IN\ DATABASE,\ FACT\ "AREA\ HIGHLAND\ SMOKED\ AT\ 15.30"\ IS\ INCLUDED\ TO\ DATABASE\}. \quad (17)$$

Note also, that equations (4)–(9) correspond mostly to the closed world interpretation, which postulates that the absence of the fact in the DB is equivalent to its absence in the reality.

While accesses to the DB metadatabase remains constant, i.e. $D_{t+1} = D_t$.

Let's consider DML mathematical and operational semantics.

DML M-semantics is based on the MDB D_t representation as set of generating rules of the context-free (CF) grammar in the sense of N.Chomsky [29]. Every such rule is the construction $\alpha \to \beta$, where α is the non-terminal symbol ("non-terminal" shortly) while β is the string containing symbols of terminal alphabet V and non-terminals, which are the names of the structural components of the facts $w \in W_t$. The initial non-terminal ("axiom" in N.Chomsky terminology) of the context-free grammar G_t, which set of the rules ("scheme") is D_t, is denoted as α_0 and from the substantial point of view is "fact". In this frames

$$G_t = < V, N_t, \alpha_0, D_t >, \quad (18)$$

where

$$N_t = \{\alpha \mid \alpha \to \beta \in D_t\} \quad (19)$$

is set of the non-terminals ("non-terminal alphabet") of the CF grammar G_t.

Database W_t is named correct to the metadatabase D_t if

$$W_t \subseteq L(G_t), \quad (20)$$

i.e. facts, having place in the DB, are words of the CF language $L(G_t)$. In other notation,

$$(\forall w \in W_t)\ \alpha_0 \overset{*}{\underset{G_t}{\Rightarrow}} w, \quad (21)$$

i.e. any fact w in the DB W_t is generated by (derived in) the CF grammar G_t. Notation $\overset{*}{\underset{G_t}{\Rightarrow}}$ is used to show generativity (or derivability) of one string in alphabet $V \cup N_t$ from another.

Example 2. Consider metadatabase D_t containing following rules (non-terminals are in the angle brackets as in Backus-Naur notation):

$< fact > \to AREA\ < name\ of\ area >< state >\ AT < time >,$
$< name\ of\ area > \to < text >,$
$< state > \to NORMAL,$
$< state > \to SMOKED,$
$< time > \to < hours >.< minutes >,$
$< hours > \to < 0\ to\ 1 >< 0\ to\ 9 >,$
$< hours > \to 2 < 0\ to\ 3 >,$
$< 0\ to\ 1 > \to 0,$
$< 0\ to\ 1 > \to 1,$
$< 0\ to\ 9 > \to 0,$

...
$< 0\ to\ 9 > \rightarrow 9,$
$< 0\ to\ 3 > \rightarrow 0,$
...
$< 0\ to\ 3 > \rightarrow 3,$
$< minutes > \rightarrow < 0\ to\ 5 >< 0\ to\ 9 >,$
$< 0\ to\ 5 > \rightarrow 0,$
...
$< 0\ to\ 5 > \rightarrow 5,$
$< text > \rightarrow < symbol >,$
$< text > \rightarrow < symbol >< text >,$
$< symbol > \rightarrow A,$
...
$< symbol > \rightarrow Z,$
$< symbol > \rightarrow 0,$
...
$< symbol > \rightarrow 9,$
$< symbol > \rightarrow \_.$

Database

$$W_t = \{\, AREA\ A\ W\ SMOKED\ AT\ 15.10,\ AREA\ E\ NORMAL\ AT\ 23.59 \,\}$$

is correct to this MDB, while database

$$W_t = \{AREA\ A\ W\ NORMAL\}$$

is incorrect.■

As may be seen, the classical formal grammars suggested by N.Chomsky as a tool for description of syntax structures of the sentences of natural languages [29] are used here in such a way, that the structures of DB facts are described by the metadatabase, which itself is the rules set of CF grammar. In this application CF grammars are simple, understandable and quite general model of data description language (DDL). Note also, that MDB itself is "set-of-strings" database, which elements are string representations of the rules $\alpha \rightarrow \beta$ (the only difference is that its alphabet includes angle brackets and arrow additionally to V). This unifies DSML sublanguages for DB and MDB.

As to the data manipulation language, we shell use representation of the substantial part *x* of any access message $\omega_t =< o, \mathrm{DB}, x >$, as a sentential form of grammar G_t, so that

$$I_t = \{w \mid x \stackrel{*}{\Rightarrow} w \,\&\, w \in V^*\} \qquad (22)$$

(for short, $\stackrel{*}{\Rightarrow}$ is used instead of $\underset{G_t}{\overset{*}{\Longrightarrow}}$).

The set of all sentential forms of the grammar G_t will be denoted lower as $SF(G_t)$:

$$SF(G_t) = \{y \mid \alpha_0 \stackrel{*}{\Rightarrow} y\}. \qquad (23)$$

As may be seen, the answer to the query with substantial part $x \in SF(G_t)$ is set of strings of the language $L(G_t)$, derived from the sentential form *x* and being facts of DB W_t:

$$A_{t+1} = \{w \mid w \in W_t \,\&\, x \stackrel{*}{\Rightarrow} w\}. \qquad (24)$$

For the DB correctness, inclusion definition is rewritten in a following way:

$$W_{t+1} = W_t \cup \{w \mid w \in I_t \,\&\, w \in L(G_t)\}, \qquad (25)$$

while deletion by the access $W_t =< delete, \mathrm{DB}, x >$ is defined as

$$W_{t+1} = W_t - \{w \mid x \stackrel{*}{\Rightarrow} w \,\&\, w \in W_t\}. \qquad (26)$$

Example 3. The substantial part of query to the DB from examples 1, 2 , which purpose is information about smoked areas, looks like

$$x = AREA < name\ of\ area > \ SMOKED\ AT \ < time >.\blacksquare$$

It is obvious, that "SF–like" query language do not express selection conditions which define boundary restrictions for numeric fields of records in usual relational databases. However, this necessary feature may be added easily to this language by means of ordering set of rules with the similar left part (non-terminal) in a way described in [28]. This allows to define any necessary linear order on the set of words generated from every non–terminal of the CF grammar and to use such orders for selecting facts from the databases.

More sophisticated queries may be constructed by boolean expressions of the atomary units $x \in SF(G_t)$, while relationally complete query languages implementation needs transfer to the knowledge segment of the SSF based on the so called augmented Post systems [25], [28].

Relational databases SSF representation and operation techniques, as well as pre- and post-relational DB modeling, are described in details in [28].

Operational semantics of the DML is quite evident: it is based on the sequential sorting of facts $w \in W_t$ with the check of $x \stackrel{*}{\Rightarrow} w$ derivability, which is algorithmically solvable for CF grammars.

Various types of implementational semantics correspond to various fixed types of CF grammars (for example, so-called page databases with key access which support hypertext storage and online processing [26], [28] and are SSF unified representation of the today web-interface portals and twitter-like databases).

III. SET-OF-STRINGS DATABASES WITH INCOMPLETE INFORMATION

For the limited size of the article let us begin the consideration of the SSF modeling of databases with incomplete information directly from the mathematical semantics of the corresponding data manipulation language (denoted as DMLI).

DMLI is based on the same MDB representation in the form of the D_t set of the context-free grammar rules $\alpha \rightarrow \beta$. Database with incomplete information denoted as X_t is finite

set of CF grammar G_t sentential forms, called incomplete (N-) facts in this application:

$$X_t = \{x_1, \dots, x_{m(t)}\} \subseteq SF(G_t). \quad (27)$$

Example 4. Consider MDB from the example 2, and corresponding DBI

$$X_t = \{AREA\ LONELY\ TREES\ NORMAL\ AT\ 12.01,$$
$$AREA\ LONELY\ TREES\ < state > AT\ 13. < minutes >,$$
$$AREA < name\ of\ area > SMOKED\ AT\ 14.30\}.$$

As seen, DBI contains three N-facts, first of which is usual fact in the sense (4)–(9). Second N-fact corresponds to the information about the same object of monitoring but with unknown state of area and time, which is in the interval 13.01-13.59. Third N-fact contains information about unknown area, which was smoked at 14.30. One may suppose, that two last N-facts inclusion to DBI is consequence of the sensors and/or communication hardware malfunction. ■

Consider equations describing DBI update and query processing.

We shell suppose without loss of generality that CF-grammar G_t corresponding to the MDB D_t is acyclic and unambiguous in the sense [30]. Under these suppositions set $SF(G_t)$ is partly ordered by the relation $\overset{*}{\underset{G_t}{\Rightarrow}}$: it is reflexive ($x \overset{*}{\underset{G_t}{\Rightarrow}} x$), antisymmetric (when $x \overset{*}{\underset{G_t}{\Rightarrow}} y$ and $y \overset{*}{\underset{G_t}{\Rightarrow}} x$ then $x = y$) and transitive (if $x \overset{*}{\underset{G_t}{\Rightarrow}} y$ and $y \overset{*}{\underset{G_t}{\Rightarrow}} z$ then $x \overset{*}{\underset{G_t}{\Rightarrow}} z$).

There is maximal element of the set $SF(G_t)$ – it is axiom α_0 (“fact”) because for every $\in SF(G_t)$ $\alpha \overset{*}{\underset{G_t}{\Rightarrow}} x$. For every subset $X \subseteq SF(G_t)$ there exists set of its upper bounds – sentential forms (“N-facts”) $y \in SF(G_t)$ such that $y \overset{*}{\underset{G_t}{\Rightarrow}} x$ for all $x \in X$, and minimal (least) upper bound $\sup X$ such that for every other upper bound y from the mentioned set the relation $y \overset{*}{\underset{G_t}{\Rightarrow}} \sup X$ is true. For some $X \subseteq SF(G_t)$ there may exist set of lower bounds – sentential forms (“N-facts”) $y \in SF(G_t)$ such that $x \overset{*}{\underset{G_t}{\Rightarrow}} y$ for all $x \in X$ – and maximal lower bound $\inf X$ such that for every other lower bound $\inf X \overset{*}{\underset{G_t}{\Rightarrow}} y$ is true.

Example 5. For DBI from the example 4 $\inf X_t$ does not exist, but

$$\sup X_t = AREA < name\ of\ area >< state >$$
$$AT\ 1 < 0\ to\ 9 > . < minutes > .$$

At the same time

$$\inf \{AREA\ LONELY\ TREES\ < state > AT\ 12.30,$$
$$AREA\ < name\ of\ area > SMOKED\ AT$$
$$12. < minutes >\}$$
$$= AREA\ LONELY\ TREES\ SMOKED\ AT\ 12.30. ■$$

Starting from the said higher, we shall follow the substantial interpretation of the relation $\overset{*}{\underset{G_t}{\Rightarrow}}$ of the mutual derivability of the sentential forms of the G_t CF grammar as the relation of the mutual informativity of incomplete facts $x \in SF(G_t)$. In the frames of this interpretation $x \overset{*}{\underset{G_t}{\Rightarrow}} x'$ means that N-fact x' is not less informative in comparison with N-fact x (if $x \overset{+}{\underset{G_t}{\Rightarrow}} x'$ then x' is more informative and is called concretization of x). If $X \subseteq X_t$, then $\sup X$ is N-fact, which is maximally informative N-fact in comparison with all N-facts, which are less informative, than all N-facts from the set X. At the same time N-fact $\inf X$, if it exists, is minimally informative N-fact in comparison with all N-facts, which are more informative, than all N-facts from the set X. (This interpretation fits to A. Kolmogorov’s algorithmic theory of information basic postulates, i.e. constructive objects mutual complexity [39]).

Let us consider DBI $X_t = \{x_1, \dots, x_{m(t)}\} \subseteq SF(G_t)$, in relation to which it is reasonable to suppose that it contains maximally informative N-facts, which only may be acquired by the system. In this case existence of N-fact $x \in X_t$ excludes existence of N-fact $x' \in X_t$, such that $x' \overset{*}{\underset{G_t}{\Rightarrow}} x$, because, obviously, x' is redundant here. We shall call DBI X_t, which does not contain such redundant N-facts, “non-redundant DBI”. We shall consider only non-redundant databases with incomplete information lower, if the contrary is not said specially.

After this assumption M-semantics of the non-redundant DBI update (or result of N-fact $x \in SF(G_t)$ inclusion) is described by the following expression:

$$X_{t+1} = X_t \cup \{x\} - \{y \mid y \in X_t \ \&\ (x \overset{*}{\underset{G_t}{\Rightarrow}} y \vee y \overset{*}{\underset{G_t}{\Rightarrow}} x)\}. \quad (28)$$

In other words, inclusion of N-fact x to DBI results in deletion from DBI all N-facts more or less informative in comparison with x. Both are necessary for DBI non-redundancy maintaining. The “novelty” of N-fact x, which reflects state of problem area at the last moment t, serves the grounds for storing x, while N-facts “compatible by informativity” with x eliminating from DBI. The answer to N-fact inclusion to DBI must contain N-facts eliminated from the database (more or less informative in comparison with x):

$$A_{t+1} = X_t - X_{t+1}. \quad (29)$$

Example 6. If N-fact $x = AREA\ GREEN\ FOREST\ SMOKED\ AT\ 14.30$ is included to DBI X_t from the example 4, then

$$X_{t+1} = \{AREA\ LONELY\ TREES\ NORMAL\ AT\ 12.01,$$

$AREA\ LONELY\ TREES < state >$
$AT\ 13. < minutes >,$
$AREA\ GREEN\ FOREST\ SMOKED\ AT\ 14.30\},$

because

$$AREA < name\ of\ area > SMOKED\ AT\ 14.30 \overset{*}{\underset{G_t}{\Rightarrow}} AREA\ GREEN\ FOREST\ SMOKED\ AT\ 14.30. \blacksquare$$

As to query M-semantics, there may by two different versions, which run out of the new DBI features in comparison with DB.

The first version is direct generalization of (24):

$$A^y_{t+1} = \{x \mid x \in X_t \ \&\ y \overset{\bullet}{\Rightarrow} x\}, \tag{30}$$

where A^y_{t+1} is the answer to the query with content y. As seen, all N-facts from the DBI X_t, which are no less informative than x, are included to the answer.

The second version is more precise and practically useful. It is based on the purpose of the query as check of the possibility of facts more informative, than N-facts having place in the DBI X_t. Returning to S-semantics, we may postulate that DBI X_t contains N-fact x, and the query content y is such, that

$$(\exists w \in L(G_t))\ x \overset{*}{\underset{G_t}{\Rightarrow}} w\ \&\ y \overset{*}{\underset{G_t}{\Rightarrow}} w, \tag{31}$$

so, while x is not concretization of y, it is sensible to include x to the answer because there may be facts $w \in L(G_t)$, which are both x and y concretizations. The set of such facts is intersection $W_x \cap W_y$, where

$$W_x = \{w \mid x \overset{*}{\underset{G_t}{\Rightarrow}} w\}, \tag{32}$$

$$W_y = \{w \mid y \overset{*}{\underset{G_t}{\Rightarrow}} w\}. \tag{33}$$

The intersection finite representation is, obviously, maximal lower bound of the $\{x, y\}$ set. For this reason, answer to the query with y content may be the set

$$\underline{A}^y_{t+1} = \{x \mid x \in X_t \ \&\ \exists \inf\{x, y\}\}, \tag{34}$$

and, as alternative,

$$\bar{\bar{A}}^y_{t+1} = \{\inf\{x, y\} \mid x \in X_t \ \&\ \exists \inf\{x, y\}\}. \tag{35}$$

Example 7. Consider DBI X_t from the example 4, and the query with content $y = AREA < name\ of\ area > SMOKED\ AT\ < time >$. Purpose of this query is to get information about all areas smoked. According to (30), (34) and (35)

$A^y_{t+1} = \{AREA < name\ of\ area > SMOKED\ AT\ 14.30\},$
$\underline{A}^y_{t+1} = \{AREA\ LONELY TREES < state > AT$
$13. < minutes >,$
$AREA < name\ of\ area > SMOKED\ AT\ 14.30\},$
$\bar{\bar{A}}^y_{t+1} = \{AREA\ LONELY TREES\ SMOKED\ AT$
$13. < minutes >,$
$AREA < name\ of\ area > SMOKED\ AT\ 14.30\}. \blacksquare$

From here, possessing minimal toolkit for maintaining and processing databases with incomplete information in the "Set-of-Strings" Framework, we may consider some key aspects of N-facts fusion.

IV. INCOMPLETE FACTS FUSION

N-facts fusion may be implemented in two versions: preprocessing and postprocessing.

In the first case (while OLAP usually) the "assembling" of the fragmentary data entering from various sensors is executed in order to create maximally informative N-fact, which is directed to the following processing. Every of the mentioned sensors surveys the monitored objects in such a way, that one part of their state parameters is known more precisely while another less precisely. The "assembled" N-fact must contain all information having place in the entering N-facts in the supposition they do not contradict one another. If the last occurs, then some extra processing is needed.

At the second case (while Data Mining usually) DBI subset selected by the query is checked for the possibility of the maximally informative N-fact "assembled" from N-facts selected. This N-fact, if it exists, accumulates all information, which is contained in the mentioned N-facts selected.

Consider both cases.

Preprocessing. Let us have the set of N-facts $X = \{x_1, \dots, x_m\} \in SF(G_t)$, corresponding to one and the same monitored object. This set at the current step of the OLAP system operating process enters this system. From the substantial point of view it is obvious, that N-fact containing all information, which is contained in all N-facts from this set, is $\underline{x} = \inf X$, if maximal lower bound of the set X does exist. Set X in this case is named "non-contradictory" (if $\inf X$ does not exist – respectively, "contradictory"). Various aspects of the contradictory sets of N-facts processing are object of the separate publication. From the general point of view, it's obvious that if contradictions exist, the maximum one can make while "assembling", is to include to the resulting N-fact $\bar{x}$ all fragments of information from N-facts $x_1, \dots, x_m$, which do not contradict one another. Formally, in this case $\bar{x} = \sup X$.

Example 8. Consider $X = \{x_1, x_2, x_3, x_4\}$, where

$x_1 = CAR < Ford\ or\ Bentley > COLOUR$
$< white\ or\ gray > NUMBER\ MN < l >< f >< f >,$
$x_2 = CAR\ FORD\ COLOUR\ WHITE\ NUMBER\ < l >$
$< l >< l >< f > 6$
,
$x_3 = CAR\ FORD\ COLOUR < colour > NUMBER\ <$
$l > NX1 < f >$
,
$x_4 = CAR\ < brand > COLOUR < colour >$
$NUMBER\ M < l > X < f >< f >$
.

Metadatabase relating this set of N-facts include following rules:

$< fact >\rightarrow CAR < brand > COLOUR < colour >$
$NUMBER \ < l >< l >< l >< f >< f >,$
$< brand >\rightarrow < Ford\ or\ Bentley >,$
$< brand >\rightarrow < BMW\ or\ Audi >,$
$< Ford\ or\ Bentley >\rightarrow FORD,$
$< Ford\ or\ Bentley >\rightarrow BENTLEY,$
$< BMW\ or\ Audi >\rightarrow BMW,$
$< BMW\ or\ Audi >\rightarrow AUDI,$
$< colour >\rightarrow < white\ or\ gray >,$
$< colour >\rightarrow < black\ or\ brown >,$
$< white\ or\ gray >\rightarrow WHITE,$
$< white\ or\ gray >\rightarrow GRAY,$
$< black\ or\ brown >\rightarrow BLACK,$
$< black\ or\ brown >\rightarrow BROWN,$
$< l >\rightarrow A,$
...
$< l >\rightarrow Z,$
$< f >\rightarrow 0,$
...
$< f >\rightarrow 9.$

The set contains N-facts obtained from interrogation of witnesses to a road traffic accident, committed by car driver fled the scene. The first witness (x_1) asserts that car was white or gray, Ford or Bentley, while alphabetical part of its number begun from M and N letters. The second witness (x_2) thinks, that car was white Ford, and the last figure of its number is 6. The third witness (x_3) saw, that car was Ford, but two last letters of the alphabetic part of its number where N and X while the first figure was 1. The fourth witness (x_4) knows neither car brand, nor its colour, but he is sure the alphabetic part of car number begins from M and ends by X.

The result of preprocessing is

$$\inf X = CAR\ FORD\ COLOUR\ WHITE\ NUMBER\ MNX16.$$

That means accident probably committed by the driver of the white Ford number MNX 16. ■

Postprocessing. Independently from the M-semantics version (30), (34) or (35), reply to y query to DBI is some set $A = \{x_1, \dots, x_m\}$ of N-facts. To "assemble" from this set maximally informative N-fact, which contains all information, which takes place in N-facts from the A set, it is sufficient to create N-fact $\underline{x} = \inf A$. If $\underline{x}$ does exist (A is non-contradictory), then $\underline{x}$ itself is necessary N-fact. Otherwise, i.e. if A is contradictory, according to the preprocessing mode, it is sufficient to create N-fact $\overline{x} = \sup A$, which contains all information, that is "common" to all N-facts, which belong to A.

Example 9. Consider set $X = \{x_1, x_2, x_3, x_4'\}$, where x_1, x_2, x_3 are from the example 8, and

$$x_4' = CAR\ BENTLEY\ COLOUR < colour > NUMBER\ M \\ < l >< l > 46.$$

As seen, $\inf A$ does not exist, while

$$\sup X = CAR < brand > COLOUR\ WHITE\ NUMBER \\ MNX < f > 6,$$

that corresponds to the white car of unknown brand, alphabetic part MNX and numeric part of the car number with last figure 6 and first unknown. ■

The described key elements of the N-facts fusion are based on the creation of minimal upper and maximal lower bounds of "two-N-facts" set. Consider algorithmics of these operations. It is kernel of the DMLI operational semantics.

V. KEY ALGORITHMS OF INCOMPLETE FACTS SETS PROCESSING

Let us begin from formulating criterion for the recognition if $\inf\{x, y\}$ exists, where $x, y \in SF(G)$, and G is unambiguous and acyclic CF grammar.

Theorem 1 [25]. Let x and y are sentential forms of the unambiguous and acyclic CF grammar $G =< V, N, \alpha_0, R >$. Then $\inf\{x, y\}$ exists if and only if $\sup\{x, y\}$ does not contain non-terminal $\alpha \in N$ such, that $\sup\{x, y\} = z_1 \alpha z_2$, where $z_1, z_2 \in (V \cup N)^*$, and R includes two rules $\alpha \rightarrow \beta$ and $\alpha \rightarrow \beta'$ such, that

$$\beta \neq \beta', \tag{36}$$
$$z_1 \beta z_2 \overset{*}{\Rightarrow} x, \tag{37}$$
$$z_1 \beta' z_2 \overset{*}{\Rightarrow} y. \blacksquare \tag{38}$$

This criterion may be reformulated to the simpler one, which is more convenient for the practical use.

Let $\sup\{x, y\} = w_1 \alpha_{i_1} w_2 \dots \alpha_{i_j} \dots w_m \alpha_{i_m} w_{m+1}$, where $w_i \in V^*, \alpha_{i_j} \in N$. Let us denote A_x set of non-terminals numbers, as they take place in SF $\sup\{x, y\}$, such, that $j \in A_x$, if

$$x = x_1 \alpha_{i_j} x_2, \tag{39}$$
$$w_1 \alpha_{i_1} w_2 \dots w_{j-1} \alpha_{i_{j-1}} w_j \overset{*}{\Rightarrow} x_1, \tag{40}$$
$$w_{j+1} \alpha_{i_{j+1}} w_{j+2} \dots w_m \alpha_{i_m} w_{m+1} \overset{*}{\Rightarrow} x_2. \tag{41}$$

In other words, $\sup\{x, y\}$ and x are "equiinformative" "in the sense" of j-th non-terminal in SF $\sup\{x, y\}$, because this non-terminal isn't used in the generation (derivation) of the SF x from $\sup\{x, y\}$. A_x is the set of all such numbers. By analogy A_y is defined.

Example 10. As may be seen from the example 8,

$$\sup\{x, y\} = CAR < Ford\ or\ Bentley > COLOUR \\ < white\ or\ gray > NUMBER < l >< l >< l > \\ < f >< f >,$$
$$A_{x_1} = \{1,2,5,6,7\},\ A_{x_2} = \{3,4,5,6\}. \blacksquare$$

The practical criterion for $\inf\{x, y\}$ existence recognition gives the following theorem.

Theorem 2 [25]. Let

$\sup\{x,y\} = w_1\alpha_{i_1}w_2 \dots w_m\alpha_{i_m}w_{m+1}$.

Then $\inf\{x,y\}$ exists if and only if $A_x \cup A_y = \{1,\dots,m\}$.■

In the example 10 $A_x \cup A_y = \{1,\dots,7\}$, so $\inf\{x_1,x_2\}$ does exist.

The correct algorithm creating $\inf\{x,y\}$, when it exists, is the following.

INF: **function** (x,y) **returns** ( );
  $z \coloneqq \mathrm{SUP}(x,y)$;
  do $j = 1,\dots,M(z)$;
    if $j \in Ax(z)$;
    then substitute to z instead of α_{i_j} string y', which is derived from α_{i_j} in derivation $z \overset{\bullet}{\Rightarrow} y$;
    else substitute to z instead of α_{i_j} string x', which is derived from α_{i_j} in derivation $z \overset{\bullet}{\Rightarrow} x$;
  end;
  return (z);
end INF

In the body of INF variables x and y are input sentential forms (N-facts), $Ax(z)$ is denotation of set A_x of the SF $z = \sup\{x,y\}$, while $M(z)$ is set of non-terminal position numbers in the SF z. Function SUP provides derivation of $\sup\{x,y\}$:

SUP: **function** (x, y) **returns** ( );
  return $(\mathrm{SUP\#}(x,y,\alpha_0))$;
  SUP#: **function** (x,y,t) **returns** ( );
    if
      $(\exists a \in (V \cup N)^*)(\exists b \in (V \cup N)^*)(\exists \alpha \to \beta \in R)$
      $t = a\alpha b \;\&\; a\beta b \overset{\bullet}{\Rightarrow} x \;\&\; a\beta b \overset{\bullet}{\Rightarrow} y$
      then return $(\mathrm{SUP\#}(x,y,a\beta b))$;
      else return (t);
  end SUP#;
end SUP

As seen from the subfunction SUP# body, all N-facts, which are input parameters in the initial (from the function SUP body) and following calls, are upper bounds of the set $\{x,y\}$, and that N-fact, which cannot be "concreted" "to the x and y directions" simultaneously, is the minimal upper bound of this set.

Graphical illustration of the essence of described algorithms is at Fig. 1a. Here

$x = u_1\bar{u}_1u_2 \dots u_i\alpha_{j_i}u_{i+1} \dots u_n\bar{u}_nu_{n+1}$,
$x' = u_1\alpha_{i_1}u_2 \dots u_i\bar{u}'_iu_{i+1} \dots u_n\bar{u}'_nu_{n+1}$,
so $\sup\{x,x'\} = u_1\alpha_{j_1}u_2 \dots u_i\alpha_{j_i}u_{i+1} \dots u_n\alpha_{j_n}u_{n+1}$,
while at Fig. 1b $\inf\{x,x'\} = u_1\bar{u}_1u_2 \dots u_i\bar{u}_iu_{i+1} \dots u_{n+1}$,
where $u_i \in (V \cup N)^+, \bar{u}_i \in (V \cup N)^+, \alpha_{j_i} \in N$.

As one can see, due to $\bar{u}_n \neq \bar{u}'_n$ at Fig. 1a $\inf\{x,x'\}$ does not exist. On the contrary, due to the lack of such $\bar{u}_n, \bar{u}'_n$ at Fig. 1b $\inf\{x,x'\}$ does exist.

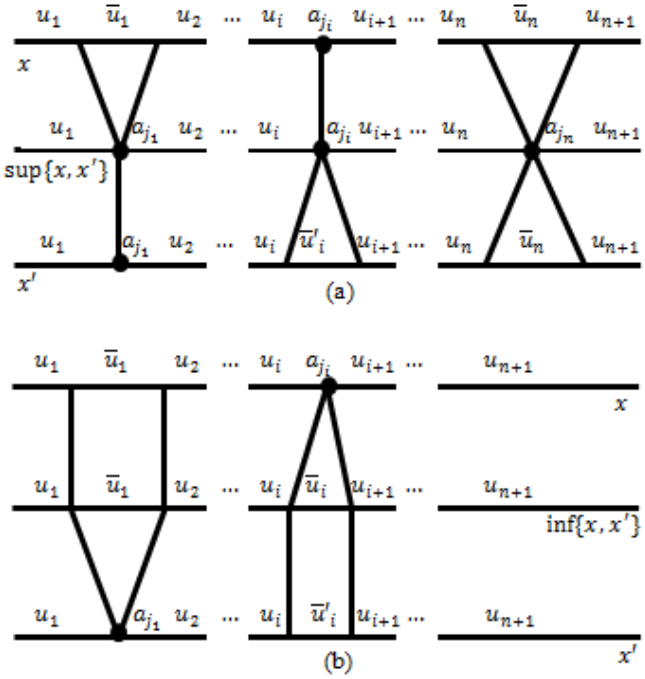

Fig. 1. Graphical illustration: (a) $\sup\{x,x'\}$; (b) $\inf\{x,x'\}$.

Example 11. Consider x_2 and x_3 from example 8. By direct execution of SUP and INF functions

$\sup\{x_2,x_3\} = CAR\ FORD\ COLOUR <$
$colour > NUMBER < l >< l >< l >< f >$
$< f >$

$\inf\{x_2,x_3\} = CAR\ FORD\ COLOUR\ WHITE\ NUMBER$
$< l > NX16$. ■

Thus, we have algorithmics for two-element sets of N-facts processing. Consider main features of the similar algorithmics for set consisting of arbitrary number of N-facts.

Let $X = \{x_1,\dots,x_m\} \in SF(G)$, where $m > 2$.

Lemma 1. For every sequence $< i_1,\dots,i_m >$ such that $\{i_1,\dots,i_m\} = \{1,\dots,m\}$,

$$\sup X = \sup\{x_{i_1},\sup\{x_{i_1},\dots,\sup\{x_{i_{m-1}},x_{i_m}\}\dots\}\}, \qquad (42)$$

and, if $\inf X$ exists,

$$\inf X = \inf\{x_{i_1},\inf\{x_{i_1},\dots,\inf\{x_{i_{m-1}},x_{i_m}\}\dots\}\}. \qquad (43)$$

Proof. As shown higher, $SF(G)$ is partially ordered set. For this kind of sets operations $\sup X$ and $\inf X$, where X is finite subset of $SF(G)$, are commutative and associative, that's why (42) and (43) are true. ■

The main problem of the described approach implementation is minimization of the redundant search while manipulating DBI. Let us look through basic elements of this problem solving techniques.

VI. BASIC ELEMENTS OF THE SET-OF-STRINGS ASSOCIATIVE STORAGE AND ACCESS

The main idea of eliminating redundant search in the set-of-strings databases (including databases with incomplete

information) is to use so-called SF-tree, which structure is induced by the mutual informativity of N-facts relation.

Let us begin from database $W \subset V^*$, which metadatabase is D (index t for simplicity is removed), and $G =< V, N, \alpha_0, D >$.

SF-tree of database W is denoted $\tau(W)$ and possesses the following features:

1) root of the tree is α_0;
2) terminal nodes are facts $w \in W$;
3) internal (non-terminal) nodes are sentential forms of G grammar;
4) for each non-terminal node $x \in SF(G)$ and its every closest descendant (son) x' condition $x \overset{+}{\underset{G}{\Rightarrow}} x'$ is true;
5) number of sons of every non-terminal node does not exceed value

$$m = 1 + \max_{\alpha \in N}\{l \mid \{\alpha \to \beta_1, \dots, \alpha \to \beta_l\} \subseteq D\}, \quad (44)$$

i.e. maximal number of alternatives of non-terminal symbols plus one (by transforming scheme D to bialternative mode [27], [28] one may fix $m = 3$).

The opportunity of the search reduction by SF-tree $\tau(W)$ runs out from the following theorem.

Theorem 3 [28]. If CF grammar $G =< V, N, \alpha_0, D >$ is unambiguous and acyclic, x and y are its sentential forms, and $\inf\{x, y\}$ does not exist, then for every $x' \in SF(G)$ such, that $x \overset{+}{\underset{G}{\Rightarrow}} x'$, $\inf\{x', y\}$ does not exist too. ■

This allows, while access with substantial part y processing, to eliminate from search every subtree of tree $\tau(W)$, which root x is such, that $\inf\{x, y\}$ does not exist, because due to theorem 3 and feature 4 of $\tau(W)$ it is clear, that all terminal nodes of the subtree (i.e. facts $w \in W$ such, that $x \overset{+}{\underset{G}{\Rightarrow}} w$) $\inf\{x, w\}$ does not exist too. The last is equivalent to false of condition $y \overset{\cdot}{\Rightarrow} w$, which is necessary for inclusion of fact $w \in W$ to the answer.

Accumulation of the answer to the query by SF-tree search ("SDB index navigation") is implemented by recursive function SRCHK with two arguments, first of which is y (substantial part of the query) and the second one is x_0 (root of the searched subtree). Function SRCHK returns set of facts, entering W, which are located at terminal nodes of the mentioned subtree and are derived from x_0. Here evident equality $\inf\{y, x_0\} = x_0$, if $y \overset{\cdot}{\Rightarrow} x_0$, is used. In the SRCHK text body variable X_0 is the set of x_0 sons.

```
SRCHK: function (y, x0) returns ( );
  a := {∅};   /*initial value of a accumulating variable */
   if x0 is terminal node
        then a := {x0};
        else do x' ∈ X0;
                if ∃ inf{y, x'}
                then a:∪ SRCHK(y, x');
              end x';
        return (a);
   end SRCHK
```

Application of this function to SDB W and query y is implemented by call $\mathrm{SRCHK}(y, \alpha_0)$, where, remember, α_0 (axiom of the CF grammar G with scheme D) is root of the tree $\tau(W)$.

Example 12. Let

W_t = {*AREA LONELY TREES NORMAL AT* 12.00,
AREA LONELY TREES NORMAL AT 13.30,
AREA GREEN FOREST SMOKED AT 14.50,
AREA GREEN FOREST SMOKED AT 15.30}

and metadatabase D from the example 2. Possible SF-tree $\tau(W)$ may be shown at Fig.2.

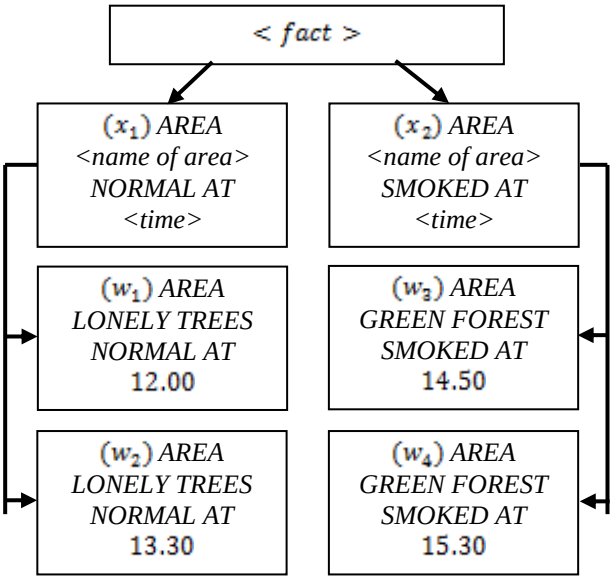

Fig.2. SF-tree $\tau(W)$

Let $y = AREA \ < name\ of\ area > \ SMOKED\ AT \ < time >$ (purpose of the query is to get all information about smoked areas). The search is performed as follows:

(step 1) $\inf\{y, x_1\}$ does not exist (so w_1 and w_2 being the sons of x_1 are not processed);

(step 2) $\inf\{y, x_2\}$ exists (so subtree with root x_2 is processed);

(step 3) $\inf\{y, w_3\}$ exists (fact w_3 is included to the answer);

(step 4) $\inf\{y, w_4\}$ exists (fact w_4 is included to the answer as well). ■

As seen from the example 12, both direct search and search by the SDB index navigation are executed by four steps, but with the growth of the database volume the computational complexity of the navigation runs to the logarithmic area (of course, if $\tau(W)$ "corresponds" to queries structure [28]).

Generalization of SF-trees to the BDI case is trivial – N-facts (being sentential forms of CF-grammar, which scheme is metadatabase) in terminal nodes of the tree are permitted. Essentially, that if we follow M-semantics (34), no corrections to the SRCHK text are needed. If we use (35), then the assignment operator "**then** $a := \{x_0\}$" is replaced by "**then** $a:\{\inf\{y, x_0\}\}$".

Consideration of SF-trees correction while facts (N-facts) inclusion and deletion, as well as of the variety of these trees modifications (ΔSF-trees, multidimensional SF- and ΔSF-trees etc.) developed to compress the volume of the stored data and to minimize computational complexity of the

search are out of the presented work coverage; the interested reader may find corresponding content in [27], [28].

VII. AUGMENTED POST SYSTEMS AND APS INCOMPLETE INFORMATION MODELING

Described elements of DBI modeling illustrate new opportunities, which are not available by classical relational and post-relational approaches. Much more opportunities are provided by the SSF basic knowledge representation model – mentioned above augmented Post systems, various of modifications of which provide OLAP of the incomplete information as well as Data Mining [25], [27], [28].

Most efficient knowledge-based systems (KBS), implemented within the SSF, are operating SDB with fixed CF grammars: for example, mentioned higher SDB, which elements (facts) have the form of variable-length pages accessible by their first strings, that allows common hypertext information representation and provides fast information processing because of no necessity of arbitrary CF grammar parsing of the entering flow. Up-to-date APS KBS are able to operate efficiently in many practical applications one of the most useful being DPI/DPP mentioned in the introduction.

Let us consider main elements of the APS knowledge representation.

Augmented Post systems were constructed from two predecessors - Post systems and formal grammars - in order to obtain formalism which, first of all, would be deeply integrated with set-of-strings database representation, and secondly, would possess deductive features comparable with Prolog and various relational model deductive extensions, providing selection of not only explicit data, but also data, which may be obtained from the last by logical inference.

The augmented Post system, or APS–represented knowledge base (for short, APS KB), P is couple $< S, D >$, where D is metadatabase in the sense of (18)-(19), while S is a set of so-called augmented, or string (S-), productions. S-production $\sigma =< q, d >$ is couple, the first component of which, named body, has the form

$$s_0 \leftarrow s_1, \dots, s_m, \tag{45}$$

where $m \geq 0$ and $s_0, s_1, \dots, s_m$ are Post terms (for short, "terms" lower), i.e. non-empty strings of symbols of the terminal alphabet V of the MDB D and variables. Universum of variables is denoted below as Γ, and for $i = 0,1, \dots, m$,

$$s_i \in (V \cup \Gamma)^+. \tag{46}$$

Term s_0 is called header while terms $s_1, \dots, s_m$ are called conditions. Set of conditions is unordered, i.e. *S*-productions

$$< s_0 \leftarrow s_{i_1}, \dots, s_{i_m}, d > \tag{47}$$

and

$$< s_0 \leftarrow s_{j_1}, \dots, s_{j_m}, d >, \tag{48}$$

where $\{i_1, \dots, i_m\} = \{j_1, \dots, j_m\} = \{1, \dots, m\}$ are identical. The mentioned set of conditions may be empty in general case, i.e. $m = 0$. S-production with the empty conditions set, looking like

$$< s_0 \leftarrow, d >, \tag{49}$$

is called S-axiom.

Component d in the S-production $\sigma =< q, d >$ is called variables declaration and is set of generation rules of the form

$$\gamma \rightarrow \beta, \tag{50}$$

where $\gamma \in \Gamma$ is variable presenting in at least one of the terms $s_0, s_1, \dots, s_m$, and $\beta \in (V \cup N)^*$, where N is non-terminal alphabet of the metadabase D. Rule (50) is called variable γ declaration, which defines $V(\gamma)$ set of permitted values of this variable:

$$V(\gamma) = \{u \mid \beta \overset{*}{\underset{G}{\Rightarrow}} u \ \& \ u \in V^*\}, \tag{51}$$

where G is CF-grammar, corresponding to metadabase D in sense (18). It is essential, that there is one and only one declaration $\gamma \rightarrow \beta$ for every variable γ having place in the q body of S-production $\sigma =< q, d >$. S-production σ, which body $s_0 \leftarrow s_1, \dots, s_m$ has not variables, i.e.

$$s_i \in V^*, \tag{52}$$

for all $i = 0,1, \dots, m$, is called concrete S-production (for short CS-production). Evidently, in this case $d = \{\emptyset\}$. CS-production

$$< s_0 \leftarrow, d >, \tag{53}$$

such, that $s_0 \in V^+, d = \{\emptyset\}$, is called CS-axiom.

S-production $\sigma =< q, d >$ defines set of CS-productions $\overline{\sigma}$ in the following way:

$$\overline{\sigma} = \bigcup_{\delta \in \overline{d}} < s_0[\delta] \leftarrow s_1[\delta], \dots, s_m[\delta], \{\emptyset\} >, \tag{54}$$

$$\overline{d} = \bigcup_{u_1 \in V(\gamma_1)} \dots \bigcup_{u_k \in V(\gamma_k)} \{\{\gamma_1 \rightarrow u_1, \dots, \gamma_k \rightarrow u_k\}\}, \tag{55}$$

where $s_0[\delta]$ are words in alphabet V, which are obtained by the substitution of strings $u_1, \dots, u_k$ instead of variables $\gamma_1, \dots, \gamma_k$ having place in terms $s_0, s_1, \dots, s_k$ in such a way, that all occurrences of the variable γ_i in all terms of the q body are replaced by one and the same string u_i.

S-production $\sigma \in S$ is called correct to MDB D if

$$(\forall < w_0 \leftarrow w_1, \dots, w_m, \{\emptyset\} > \in \overline{\sigma}) \ \{w_0, w_1, \dots, w_m\} \subseteq L(G), \tag{56}$$

i.e. all terms of every CS-production from set $\overline{\sigma}$ are words of language $L(G)$, where CF grammar G corresponds to metadabase D in sense (18). APS KB $P =< S, D >$ is called correct, if all S-productions $\sigma \in S$ are correct to MDB D.

Let us define the notion of the APS KB extensional.

Consider correct APS KB $P =< S, D >$. Let

$$\overline{S} = \bigcup_{\sigma \in S} \overline{\sigma}, \tag{57}$$

i.e. $\overline{S}$ is a set of all CS-productions defined by all S-productions of this knowledge base in the sense (54) - (55).

Define

$$W_{(0)} = \{w \mid < w \leftarrow, \{\emptyset\} > \in \overline{S}\}, \tag{58}$$

$$W_{(i+1)} = W_{(i)} \cup \Big(\bigcup_{\substack{< w_0 \leftarrow w_1, \dots, w_m, \{\emptyset\} > \in \overline{S} \\ w_1 \in W_{(i)} \\ \dots \\ w_m \in W_{(i)}}} \{w_0\} \Big), \tag{59}$$

and APS KB $P =< S, D >$ extensional $Ex(P)$, i.e. set of facts defined by this knowledge base, is fixed point of the sequence $W_{(0)}, \dots, W_{(i)}, W_{(i+1)}, \dots$, which is infinite in general case:

$$Ex(P) = W_{(\infty)}. \tag{60}$$

Evidently, due to *P* correctness,

$$Ex(P) \subseteq L(G), \tag{61}$$

and $Ex(P)$ is finite, if their exists i such, that

$$W_{(i)} = W_{(i+1)}. \tag{62}$$

From the linguistical point of view $Ex(P)$ is language in alphabet V, defined by APS P and being sublanguage of $L(G)$. From the SSF knowledge/data engineering point of view $Ex(P)$ is set of facts, which are either known explicitly, i.e. $w \in W_{(0)}$ (they are called ground facts) and either are derived from ground facts and/or other derived facts. It is evident, that set of ground facts $W_{(0)}$ is set-of-strings database, while set of S-productions with non-empty conditions sets is APS KB intensional providing the mentioned derivation.

Now we can define semantics of queries to APS KB.

Set-theoretical semantics of query to APS KB is similar to (9):

$$A = \overline{W} \cap I, \tag{63}$$

where $\overline{W}$ is APS KB extensional, and I is set of facts, which actuality check is purpose of the query (for simplicity lower indexes t and $t+1$ having place in (9) are omitted here).

The simplest query language, harmonized with similar SDB query language considered higher, may be set of couples $< s, d >$, where s is term, and d is its variables declaration, and

$$I = Ex(< \{< s \leftarrow, d >\}, D >). \tag{64}$$

Here, $< \{< s \leftarrow, d >\}, D >$ is correct APS KB, which set of S-productions – selection criterion – contains one S-axiom, defining set of facts, which may belong to $Ex(P)$.

Example 13. Consider metadatabase from Example 2 adding to it following three rules:

$< fact > \rightarrow SENSOR < i > AT < time > - < state >,$
$< i > \rightarrow < symbol >< symbol >,$
$< fact > \rightarrow SENSOR < i > LOCATED\ AT\ AREA$
$< name\ of\ area >.$

Facts like $SENSOR\ XX\ AT\ 12.30 - NORMAL$ contain information about sensors, which gather and send to the fusion centre data about state of the atmosphere in the surrounding area. Facts like $SENSOR\ XX\ LOCATED\ AT\ AREA\ Y...Y$ contain information about sensors location.

Consider APS knowledge base $P =< S, D >$, which MDB D was described higher, and S contains following S-productions:

$\sigma_1 :< AREA\ a\ s\ AT\ t \leftarrow$
$SENSOR\ e\ AT\ t - s,$
$SENSOR\ e\ LOCATED\ AT\ AREA\ a,$
$\{a \rightarrow < name\ of\ area >, s \rightarrow < state >, e \rightarrow < i >,$
$t \rightarrow < time >\} >,$
$\sigma_2 :< SENSOR\ AL\ AT\ 15.00 - NORMAL \leftarrow, \{\emptyset\} >,$
$\sigma_3 :< SENSOR\ XF\ AT\ 12.30 - SMOKED \leftarrow, \{\emptyset\} >,$
$\sigma_4 :<$
$SENSOR\ AL\ LOCATED\ AT\ AREA\ HIGHLANDS \leftarrow,$
$\{\emptyset\} >$
,
$\sigma_5 :< SENSOR\ XF\ LOCATED\ AT\ AREA\ GREEN\ FOREST$
$\leftarrow, \{\emptyset\} >.$

As seen, $\sigma_2 - \sigma_5$ are concrete S-productions, and $W_{(0)}$ set, corresponding to this APS KB, consists of ground facts being headers of these CS-productions.

Query, which purpose is to get information about smoked areas, may be as follows:

$$< AREA\ z\ SMOKED\ AT\ t, \begin{Bmatrix} z \rightarrow < name\ of\ area >, \\ t \rightarrow < time > \end{Bmatrix} >,$$

while query, which purpose is to get information about sensor XF location, may look like

$< SENSOR\ XF\ LOCATED\ AT\ AREA\ v,$
$\{v \rightarrow < name\ of\ area >\} >.$

Evidently, this knowledge base extensional is

$\{AREA\ HIGHLANDS\ NORMAL\ AT\ 15.00,$
$AREA\ GREEN\ FOREST\ SMOKED\ AT\ 12.30,$
$SENSOR\ AL\ AT\ 15.00 - NORMAL,$
$SENFOR\ XF\ AT\ 12.30 - SMOKED,$
$SENSOR\ AL\ LOCATED\ AT\ AREA\ HIGHLANDS,$
$SENSOR\ XF\ LOCATED\ AT\ AREA\ GREEN\ FOREST\}.$■

Expressions (57)–(64) form mathematical semantics of the considered APS KB simple query language.

Kernel of this language operational semantics is so-called S-unification [25], [28], which place in the SSF is similar to well-known unification by J.Robinson used in the first order predicate logic resolution procedures [40]. We shall consider basic concept of S-unification and then describe shortly axiomatic system providing controlled logical inference of answers to APS KB queries.

Let $< s, d >$ be the query and $< s_0^i, d_i >$ is couple formed from S-production

$$\sigma_i: < s_0^i \leftarrow s_1^i, \dots, s_{m_i}^i, d_i >. \quad (65)$$

As may be seen easily, answer to $< s, d >$ may contain headers of the CS-productions being concretizations of σ_i, i.e. belonging to set $\overline{\sigma}_i$, if set

$$W_{\sigma_i}^{s,d} = Ex(< \{< s \leftarrow, d >\}, D >) \cap Ex(< \{< s_0^i \leftarrow, d_i >\}, D >) \neq \{\emptyset\}. \quad (66)$$

Let us suppose for simplicity, that term s has no more than one occurrence of every variable; just the same let us suppose to term s_0^i. Lower, $\inf$ denotation relates to CF-grammar G corresponding to metadabase D of APS KB $P =< S, D >$.

Lemma 2. $W_{\sigma_i}^{s,d} \neq \{\emptyset\}$ if and only SF $x = \inf\{s[d], s_0^i[d_i]\}$ exists.

Proof. If there exists fact $w \in W_{\sigma_i}^{s,d}$, then in turn exists x. If x does exist then $W_{\sigma_i}^{s,d} \neq \{\emptyset\}$. Both run out from correctness of the APS KB.■

It is evident, that SF x corresponds to new, more precise (informative, concrete) S-production σ_i variables declaration $\overline{d}_i$, which is constructed as follows:

$$s_0^i = u_1^i \gamma_1^i u_2^i \dots u_{m_i}^i \gamma_{m_i}^i u_{m_i+1}^i \overset{*}{\underset{G}{\Rightarrow}} u_1^i \beta_1^i u_2^i \dots u_{m_i}^i \beta_{m_i}^i u_{m_i+1}^i \overset{*}{\underset{G}{\Rightarrow}} u_1^i \bar{\beta}_1^i u_2^i \dots u_{m_i}^i \bar{\beta}_{m_i}^i u_{m_i+1}^i = x, \quad (67)$$

and

$$\overline{d}_i = \{\gamma_1^i \to \bar{\beta}_1^i, \dots, \gamma_{m_i}^i \to \bar{\beta}_{m_i}^i\}, \quad (68)$$

$$G_i =< V, N_i, \underset{m_i}{\alpha_0}, D_i > \quad (69)$$

$$D_i = D \cup (\bigcup_{k=1}^{} \{\gamma_k^i \to \beta_k^i\}), \quad (70)$$

$$N_i = N \cup \{\gamma_1^i, \dots, \gamma_{m_i}^i\}. \quad (71)$$

As seen, G_i is CF grammar, obtained from $< V, N, \alpha_0, D >$ by variables $\gamma_1^i, \dots, \gamma_{m_i}^i$, having place in term s_0^i, joining to the non-terminal set N, and generating rules $\gamma_1^i \to \beta_1^i, \dots, \gamma_{m_i}^i \to \beta_{m_i}^i$ joining to the scheme D of the grammar G. Because of unique occurrences of variables in term s_0^i they are non-terminals of G_i in strict sense of this notion. As a result, $\overline{d}_i$ is obtained by replacing s_0^i variables declarations by new ones, which are more concrete (precise, informative) in comparison with initial declarations having place in d_i, and, that is very important, this concretization corresponds to query $< s, d >$ in such a way, that set of CS-productions, which headers may belong to the answer to this query, may be constructed at the following steps of inference.

Sense of the operation described is illustrated by Fig. 3 and Example 13.

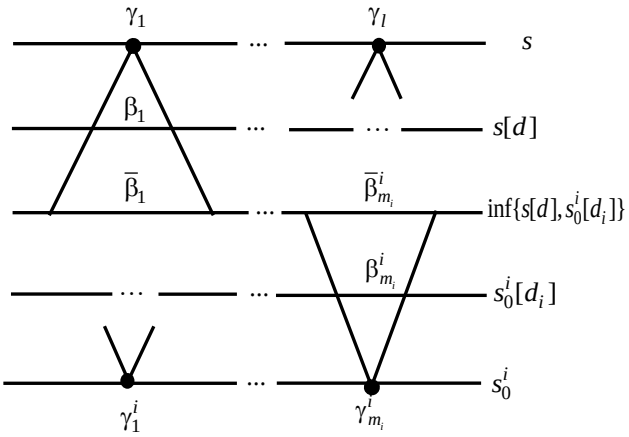

Fig. 3. Graphical illustration of S-unification.

Example 13. Consider query $< s, d >$, where

$$s = AREA\ e\ NORMAL\ AT\ t,$$
$$d = \{e \to < name\ of\ area >, t \to < time >\},$$

and S-production σ_1 from the example 13, where

$$s_0^1 = AREA\ a\ s\ AT\ t,$$
$$d_1 = \{a \to < name\ of\ area >, s \to < state >, e \to < i >, t \to < time >\}.$$

According to (67) – (71),

$$s_0^1[d_1] = AREA < name\ of\ area >< state > AT < time >,$$
$$s[d] = AREA < name\ of\ area > NORMAL\ AT < time >,$$
$$\overline{d}_1 = d_1 - \{s \to < state >\} \cup \{s \to NORMAL\} =$$
$$= \{a \to < name\ of\ area >, s \to NORMAL, e \to < i >, t \to < time >\}.■$$

S-unification provides opportunity of answers derivation by controlled logical inference, which is implemented by the axiomatic system, which contains only three inference rules – top-down successful S-resolution, top-down unsuccessful S-resolution and bottom-up S-resolution [28]. While inference processб waves of new queries with more and more precise variables declarations are generated until they reach S-axioms; after that CS-productions are formed, and back waves start until another facts set is assembled. These waves "meet and interfere", so inference process is bidirectional. Main difference between well-known first order predicate logic resolution procedures and APS axiomatics is that the last provides not existential (i.e. anyone fact of answer set), but universal inference mode (all facts needed) in full accordance with S- and M-semantics defined by (63) – (71). All the details, from the multiple variables occurrences case to procedural connection to APS KB providing calls of various software modules and complexes (up to DBMS and various external online data sources drivers) while answer derivation, concerned reader may find at [25], [28], as well as various APS flow-oriented dialects constructed for the specific classes of problems efficient handling. The last are hard real time network-centric systems implementation and application, multiagent systemы design, programming languages syntax and semantics strict formal definition,

natural languages processing, computer-aided design, technical systems simulation etc.

Basic tool of the procedural connection to APS KB are so called program (P-) productions of the form $< s_0 \Leftarrow p, d >$, where s_0 and d have the same sense as higher, being declarative component of the connection, while p is connected program name, and symbol " $\Leftarrow$ " is divider distinct from " $\leftarrow$ ". Each program p has its own extensional $W_p \subseteq L(G)$, which is subset of $Ex(< \{< s \leftarrow, d >\}, D >)$.

Example 14. Program named MULT for multiplication of integer numbers may be connected to knowledge base by P-production

$$< a * b = c \Leftarrow \text{MULT}, \{ a \rightarrow < integer >, b \rightarrow < integer >, c \rightarrow < integer >\} >$$

.

MULT extensional is infinite set containing strings like $1 * 1 = 1,\ 0 * 5 = 0,\ 311 * -1 = -311$ etc. ■

Programs may be time-dependent (with dynamic extensionals) and time-invariant (with static extensionals, as in the example above). The first are various DBMS and hardware drivers (usually sensors and actors in network-centric environments), while the last are various operations on data (numerical, graphical, textual etc.) usually.

There is simple tool for logical inference control within APS knowledge representation. Namely, variables, which declarations after S-unification must have right parts, consisting only of terminal symbols (i.e. there is no incomplete information, associated with non-terminals, inside these declarations), are marked by point over arrow in the initial descriptions having place in P- and S-productions. That concerns variables a and b in the example above, which declarations would be $a \dot{\rightarrow} < integer >, b \dot{\rightarrow} < integer >$, so query

$$< c * e = f, \{ c \rightarrow 1, e \rightarrow 4, f \rightarrow < integer >\} >$$

processing while inference will result in program MULT call, while query

$$< x * a = s, \{x \rightarrow < integer >, a \rightarrow 135, s \rightarrow < integer >\} >$$

processing will not because of information incompleteness of the declaration of x variable.

Coming back to main object of this article, let us consider APS KB incomplete information modeling techniques. It is quite simple and fully corresponds to the described higher in application to SDB as well as to already developed for relational databases with Datalog-like extensions [17].

S-production

$$\sigma = < s_0 \leftarrow s_1, \dots, s_m, d > \quad (72)$$

is called complete, if its header s_0 variables set denoted $\Gamma(s_0)$ is subset of set of variables having place in the conditions $s_1, \dots, s_m$:

$$\Gamma(s_0) \subseteq \Gamma(s_1 \dots s_m), \quad (73)$$

where term $s_1 \dots s_m$ is constructed by concatenation of terms $s_1, \dots, s_m$.

On the contrary, if

$$\Gamma(s_0) - \Gamma(s_1, \dots, s_m) \neq \{\emptyset\}, \quad (74)$$

S-production (72) is incomplete.

Incomplete S-axioms $< s_0 \leftarrow, d >$ such, that $s_0[d] \in SF(G) - V^*$, i.e. header s_0 contains at least one variable γ such, that $\gamma \rightarrow \beta \in d$ and $\beta \in SF(G) - V^*$, naturally correspond to N-facts considered higher. This allows to transfer from set-of-strings databases with incomplete information to similar APS knowledge bases, and to generalize SDBI handling techniques to APS KB with incomplete S-productions. On the other hand, shortly described higher SDB search techniques is directly and efficiently applicable to APS KB inference engines implementations, providing minimization of the S- and P-productions headers redundant search and thus minimizing inference computational complexity.

VIII. CONCLUSION

As may be seen, the described approach provides convergence of very close but still stand-alone areas: theory of formal languages, information theory and knowledge/data engineering. Synergy, obtained by this convergence, may be very useful for creation of the unified theory so necessary for practical problems solving.

The most interesting directions of the described approach future development are, to our opinion, the following:

1) information concretization in SDBI and APS KBS with the help of functional dependencies known from the relational databases theory [11,12,20];
2) contradictory SDBI and APS KBS management;
3) SDBI and APS KBS with metadatabases, describing two and three-dimensional objects;
4) SSF ideology and techniques transfer to the numeric data with the help of the SSF-associated theory of recursive multisets and optimizing multiset grammars/metagrammars [26];
5) hardware SSF implementation.

The author is ready to cooperate with all scholars and engineers who will be interested in the listed directions as well as in the SSF at all.

ACKNOWLEDGEMENT

Author extends sincere appreciation to Prof. Noam Chomsky for understanding and words of support to the SSF development.

REFERENCES

[1] Cares J. Distributed Networked Operations: The Foundation of Network Centric Warfare. – iUniverse, Inc. – 2006. – 216 p.
[2] Chang W.Y. Network-Centric Service Oriented Enterprise. – NY:Springer-Verlag. – 2010. – 560 p.
[3] Requirements for Deep Packet Inspection in Next Generation Networks. – Telecommunications Technology Association, 2012. – 97 p.

[4] Fuchs C. Implications of Deep Packet Inspection (DPI) Internet Surveillance for Society. Research Paper. – Uppsala University, July 2012. – 125 p.
[5] Lee Y., Oh J.. Lee J.K., Lee B.G. The Development of Deep Packet Inspection and Its Applications. – In: 3d International Conference on Intelligent Computational Systems (ICICS) Hong Kong, 2013, pp. 5-8.
[6] Bass T. Intrusion Detection Systems and Multisensor Data Fusion. – Communications of the ACM, Vol.43 (2000), No.4, pp.99-105.
[7] Hansen J.V., Lowry P.B., Meservy R., McDonald D. Genetic Programming for Prevention of Cyberterrorism through Dynamic and Evolving Intrusion Detection. – Decision Support Systems, Vol. 43 (2007), No.4, pp.1362-1374.
[8] Papadimitriou P., Garcia-Molina H. Data Leakage Detection. – IEEE Transactions on Knowledge and Data Engineering, Vol. 23 (2011), No.1, pp. 51-63.
[9] Shabtai I., Elovici Y., Rokach L. A Survey of Data Leakage Detection and Prevention Solutions. – NY: Springer-Verlag, 2012. – 92 p.
[10] Mohamed A.A. Design Intrusion Detection System Based on Image Block Matching. – International Journal of Computer and Communication Engineering, Vol.2 (2013), No.5, pp. 605-607.
[11] Connolly T.M., Begg C.E. Database Systems: A Practical Approach to Design, Implementation and Management (5th Edition). – Addison Wesley. – 2009. – 1400p.
[12] Gill P.S. Database Management Systems. – IK International Publishing House. -2011. – 310 p.
[13] Imielinski T., Lipski W. Incomplete Information in Relational Databases. – Journal ACM, Vol.31 (1984), No.4, pp.761-791.
[14] Zaniolo C. Database Relations with Null Values. – Journal of Computer and Systems Sciences, Vol. 28 (1984), No. 2, pp. 27-33.
[15] Winslett M. A Model-Based Approach to Updating Databases with Incomplete Information. – ACM Transactions on Database Systems, Vol.13 (1988), No.2, pp.197-227.
[16] Grahne G. The Problem of Incomplete Information in Relational Databases. – Lecture Notes in Computer Science. Vol. 554. – NY: Springer-Verlag, 1991. – 165 p.
[17] Kong O., Cheng G. On Deductive Databases with Incomplete Information. – ACM Transactions on Information Systems, Vol. 13 (1995), No.3, pp.354-370.
[18] Rafanelli M. Multidimensional Databases: Problems and Solutions. – Idea Group Publishing, 2003. – 443 p.
[19] Klein H.-J. Null Values in Relational Databases and Sure Information Answers. – In: Semantics in Databases. Lecture Notes in Computer Science. Vol. 2582. – NY: Springer–Verlag, 2003, pp. 119-138.
[20] Greco S., Molinaro C., Spezzano F. Incomplete Data and Data Dependencies in Relational Databases. – Morgan & Claypool Publishers, 2012. – 123 p.
[21] Cali A., Gottlob G., Lukasiewicz T. A General Datalog-Based Framework for Tractable Query Answering Over Ontologies. – Journal of Web Semantics, Vol.14 (2012), No.1, pp.57-83.
[22] Gheerbrant A.,Libkin L., Sirangelo C. Naive Evaluation of Queries over Incomplete Databases. – ACM Transactions on Database Systems, Vol.39 (2014), No.4, pp.31:1-31:42.
[23] Arenas M., Barcelo P., Libkin L., Murlak F. Foundations of Data Exchange. – Cambridge University Press, 2014. – 341 p.
[24] Libkin L. SQL's Three-Valued Logic and Certain Answers. – ACM Transactions on Databases Systems, Vol. 41 (2016), No.1, 10.1145/2877206.
[25] Sheremet I.A. Word Equations on Context-Free Languages. – Hannover: EANS, 2011. – 44 p.
[26] Sheremet I.A. Recursive Multisets and Their Applications. – Berlin: NG Verlag, 2011. – 249 p.
[27] Sheremet I.A. Grammatical Codings. – Hannover: EANS, 2012. – 54 p.
[28] Sheremet I.A. Augmented Post Systems: The Mathematical Framework for Data and Knowledge Engineering in Network-Centric Environment. – Berlin: EANS, 2013. – 395 p.
[29] Chomsky N. Syntactic Structures (2d Edition). – Berlin-New York: Mouton de Gruyter. – 2002. – 117 p.
[30] Moll R.N., Arbib N.A., Kfoury A.J. An Introduction to Formal Languages Theory.– NY: Springer - Verlag. – 1988. – 213 p.
[31] Post E. Formal Reductions of the General Combinatorial Decision Problem. – American Journal of Mathematics, Vol.65 (1943), No.2, pp.197-215.
[32] Kowalski R.A. Algorithm = Logic + Control. – Communications of the ACM, Vol.22 (1979), No.7, pp.424-436.
[33] Dantsin E., Eiter T., Gottlob G., Voronkov A. Complexity and Expressive Power of Logic Programming. – ACM Computer Surveys, Vol.33 (2001), No.3, pp.374-425.
[34] Nilsson U., Maluszynski J. Logic, Programming and Prolog. 2d Edition. – John Wiley & Sons, 2000. – 282 p.
[35] Niederlinski A. A Quick and Gentle Guide to Constraints Logic Programming via ECLIPSE (3d Edition). – PKJS, 2011. – 420 p.
[36] Miller D., Nadathur G. Programming with Higher-Order Logic. – Cambridge University Press, 2012. – 322 p.
[37] Bratko I. Prolog Programming for Artificial Intelligence (4th Edition). – Addison Wesley, 2014. – 640 p.
[38] Zhou Neng-Fa, Kjellerstrand H., Fruhman J. Constraint Solving and Planning with Picat. – NY: Springer-Verlag, 2015. – 160 p.
[39] Kolmogorov A.N. Three Approaches to Definition of "Information Measure" Concept. – In: The Selectas. Vol. 3. Information Theory and Theory of Algorithms – Moscow: Nauka, 1987. (In Russian). – 303 p.
[40] J.A.Robinson. A Machine-Oriented Logic Based on the Resolution Principle. – Journal of the ACM, Vol. 12 (1965), No.1, pp.23-41.

Igor Sheremet, born 1956, Doct. Eng. Sci. (1993), Professor (1998). Head of the department of Financial University under the Government of Russian Federation, deputy director for science of Russian Foundation for Basic Research. Works at areas of theoretical and applied computer science, systems analysis, data and knowledge engineering, robotics, computer security. Vice-chairman of Russian Academy of Sciences (RAS) Board on Robotics and Mechatronics, member of RAS Systems Analysis Committee, official member of the Scientific Advisory Committee of International Institute of Applied Systems Analysis (IIASA) at Laxenburg (Austria).