Chapter

Augmented Post Systems: Syntax, Semantics, and Applications

Igor Sheremet

Abstract

Augmented Post systems (APS) are string-operating Prolog-like knowledge representation, affiliated with the "Set of Strings" Framework (SSF). APS descriptive and logical inference capabilities provide natural integration of Big Data with online analytic processing. This chapter is dedicated to strict formal definition of APS syntax, mathematical and operational semantics, and to its most valuable implementational issues, as well as to APS application to Big Data, Internet of Things, cyberphysical industry, and cybersecurity areas.

Keywords: "Set of Strings" framework, augmented Post systems, string data bases, data and knowledge engineering, Big Data, Internet of Things, cybersecurity

1. Introduction

The background of "set of strings" Framework (SSF) [1–4], developed for adequate modeling of Big Data, is the representation of database as an updated Set of Strings, whose structure is defined by the current metadatabase (MDB), being a set of context-free generation rules in the sense of N. Chomsky [5].

Augmented Post systems (APS), considered in this chapter, are result of deep integration in the one toolkit of two well-known formalisms—classical string-generating formal grammars and string-operating logical systems, proposed by E. Post [6].

By this, the main features of APS are:

1. Deep connectivity with "Set of Strings" representation of databases.

2. Deductive capabilities, similar to those, which are inherent to various deductive extensions of the relational model of data, as well as to Prolog and Datalog [7–12].

The second feature provides selection of not only data, presenting in string databases (SDB) in the explicit form, but also data, which may be constructed (derived) from the explicit data by means of logical inference.

This chapter is dedicated to the formal definition and substantial description of APS and their applications. The terminology and abbreviations are the same, as have been used when describing SSF.

Section 2 is dedicated to the definition of syntax and mathematical semantics of APS, and Section 3 to their operational semantics. Procedural connection to APS is

described in Section 4, and flow-processing APS (FAPS), based on the aforementioned connection, are described in Section 5. Implementation issues and applications of APS family are considered in Section 6. The conclusion contains proposals on further development of SSF.

2. Syntax and mathematical semantics of the augmented Post systems

The **augmented Post system** $P=<S, D>$ is the composition of set S of the so-called augmented, or string (S-), productions, and set D of context-free generating rules, being metadatabase. Following the terminology of knowledge engineering, we shall call P also as *APS-represented knowledge base* (for short, APS KB).

S-production $\sigma = <q, d>$ is couple, the first component of which, named *body*, has the form of postproduction:

$$s_0 \leftarrow s_1, ..., s_m, \tag{1}$$

where $m \geq 0$ and $s_0, s_1, ..., s_m$ are terms.

Term s_0 is called *header*, while terms $s_1, ..., s_m$ are called *conditions*. The set of conditions is unordered, i.e., S-productions:

$$<s_0 \leftarrow s_{i_1}, ..., s_{i_m}, d>, \tag{2}$$

and

$$<s_0 \leftarrow s_{j_1}, ..., s_{j_m}, d>, \tag{3}$$

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

$$<s_0 \leftarrow , d>, \tag{4}$$

is called *S-axiom*.

Component d of S-production $\sigma = <q, d>$ is set of rules $\gamma \rightarrow \beta$ and is called here *variables declaration*.

Rule $\gamma \rightarrow \beta \in d$ is called *variable declaration of* γ, which defines $\mathrm{V}(\gamma, \beta)$ set of possible values (domain) of this variable:

$$\mathrm{V}(\gamma, \beta) = \{u \mid \beta \overset{*}{\underset{G}{\Longrightarrow}} u \& u \in V^*\}, \tag{5}$$

where G is CF-grammar, corresponding to metadatabase D. It is essential that there is one and only one variable declaration $\gamma \rightarrow \beta$ for every variable γ, having place in the body of S-production. S-production σ, which body $s_0 \leftarrow s_1, ..., s_m$ has no any variables, i.e.,

$$s_i \in V^*, \tag{6}$$

for all $i = 0, 1, ..., m$, is called *concrete S-production* (for short *CS-production*). CS-production:

$$< s_0 \leftarrow , d >, \tag{7}$$

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

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

$$\overline{\sigma} = \bigcup_{\delta \in \overline{d}} < s_0[\delta] \leftarrow s_1[\delta], ..., s_m[\delta], \{\varnothing\} >, \tag{8}$$

$$\overline{d} = \bigcup_{V(u_1,\gamma_1,\beta_1)} ... \bigcup_{V(u_k,\gamma_k,\beta_k)} \{\{\gamma_1 \rightarrow u_1, ..., \gamma_k \rightarrow u_k\}\}, \tag{9}$$

where $s_i[\delta]$ are strings in terminal alphabet V, which are created by the replacement of variables γ_1, ..., γ_k by strings u_1, ..., u_k in the same alphabet, respectively; all occurrences of one and the same variable γ_i in all terms s_0, s_1, ..., s_m are replaced by one and the same string u_i.

If

$$\begin{gathered}(\forall < w_0 \leftarrow w_1, ..., w_m, \{\varnothing\} > \in \overline{\sigma}) \\ \{w_0, w_1, ..., w_m\} \subseteq L(G),\end{gathered} \tag{10}$$

then S-production $\sigma \in S$ is *correct to* MDB D. This means that all terms of every CS-production from set $\overline{\sigma}$ are words of context-free language $L(G)$, and where G is context-free grammar with set of generating rules D.

APS KB $P = < S, D >$ is *correct*, if all S-productions $\sigma \in S$ are correct to metadatabase D.

Notion of the *APS KB extensional*, i.e., set of facts, which may be derived from the knowledge base by means of logical inference, is defined as follows.

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

$$\overline{S} = \bigcup_{\sigma \in S} \overline{\sigma}, \tag{11}$$

i.e., $\overline{S}$ is a set of all CS-productions defined by all S-productions of this knowledge base in the sense (8)–(9).

Define

$$W_{(0)} = \{w \mid < w \leftarrow , \{\varnothing\} > \in \overline{S}\}, \tag{12}$$

$$W_{(i+1)} = W_{(i)} \cup \left(\bigcup_{\substack{< w_0 \leftarrow w_1, ..., w_m, \{\varnothing\} > \in \overline{S} \\ w_1 \in W_{(i)} \\ ... \\ w_m \in W_{(i)}}} \{w_0\} \right), \tag{13}$$

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

$$Ex(P) = W_{(\infty)}. \tag{14}$$

Evidently, due to P correctness,

$$Ex(P) \subseteq L(G), \tag{15}$$

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

$$W_{(i)} = W_{(i+1)} \tag{16}$$

If we consider the notion of extensional of APS KB from the linguistic point of view, then $Ex(P)$ is a language in alphabet V, which, according to (15), is sublanguage of $L(G)$. At the same time, from the point of view of knowledge engineering, $Ex(P)$ is the join of set of facts $w \in W_{(0)}$, which are known explicitly (they are called lower *ground facts*), and set of facts, which are derived from ground facts and/or another derived facts. As it is easy to see, set of ground facts $W_{(0)}$ is nothing else, than SDB, while S-productions with nonempty sets of conditions form *APS KB intensional*, providing the aforementioned inference.

Now we can define semantics of language of queries to APS KB.

Set-theoretical (S-) semantics of this language is similar to S-semantics of SDB query languages:

$$A = \overline{W} \cap I \tag{17}$$

where $\overline{W}$ is APS KB extensional and I is set of facts, which actuality check is the purpose of the query.

Here we shall use the simplest query language, being set of couples $< s, d >$, where s is term and d is its variable declaration.

Mathematical (M-) semantics of this language is based on (14) and the following evident equation:

$$I = Ex(< \{ < s \leftarrow, d > \}, D >) \tag{18}$$

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

Example 1. Consider metadatabase from Example 2 of the chapter of this book, describing SSF. We shall add to it the following three rules:

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

Facts like *SENSOR NN AT* 17.00 − *NORMAL* contain information about sensors, which gather and send to the fusion center the data about the state of the atmosphere in the surrounding area. Facts like
SENSOR NN LOCATED AT AREA Y…Y contain information about sensors' location.

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

$$\sigma_1 : < AREA\ a\ IS\ s\ AT\ t \leftarrow$$
$$SENSOR\ e\ AT\ t - s,$$
$$SENSOR\ e\ LOCATED\ AT\ AREA\ a,$$
$$\{a \rightarrow < name\ of\ area >, s \rightarrow < state >, e \rightarrow < i >,$$
$$t \rightarrow < time > \} >,$$

$$\sigma_2 : <SENSOR\ AX\ AT\ 16.00 - NORMAL \leftarrow, \{\varnothing\}>,$$
$$\sigma_3 : <SENSOR\ FY\ AT\ 11.30 - SMOKED \leftarrow, \{\varnothing\}>,$$
$$\sigma_4 : <SENSOR\ AX\ LOCATED\ AT\ AREA\ HIGHLANDS \leftarrow, \{\varnothing\}>,$$
$$\sigma_5 : <SENSOR\ FY\ LOCATED\ AT\ AREA\ GREEN\ VALLEY \leftarrow, \{\varnothing\}>.$$

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

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

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

while query, whose purpose is to get information about sensor *FY* location, may look like

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

Evidently, this knowledge base extensional is

$$\{AREA\ HIGHLANDS\ IS\ AT\ NORMAL\ STATE\ AT\ 16.00,$$
$$AREA\ GREEN\ VALLEY\ IS\ SMOKED\ AT\ 11.30,$$
$$SENSOR\ AX\ AT\ 16.00 - NORMAL,$$
$$SENFOR\ FY\ AT\ 11.30 - SMOKED,$$
$$SENSOR\ AX\ LOCATED\ AT\ AREA\ HIGHLANDS,$$
$$SENSOR\ FY\ LOCATED\ AT\ AREA\ GREEN\ VALLEY\}.\blacksquare.$$

3. Operational semantics of APS

The background of operational (O-) semantics of the considered query language is the so-called S-unification, which is generalization of well-known unification, introduced by J. Robinson in [7] and used in the resolution procedures, developed for the first-order predicate logic. *(In fact, S-unification is a strict generalized definition of heuristically described unification, which, as it was shown in [1], is incorrect in the case of multiple occurrences of at least one variable to one of the unified terms.)*

We shall consider basic concept of S-unification and then describe operational semantics of used query language, corresponding to (11–18) and defining main features of the controlled logical inference of answers to queries to APS KB.

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

$$\sigma_i : <s_0^i \leftarrow s_1^i, ..., s_{m_i}^i, d_i> \tag{19}$$

As may be seen easily, the 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

$$W_{\sigma_i}^{s,d} = Ex(<\{<s \leftarrow, d>\}, D>) \cap$$
$$Ex\left(<\left\{<s_0^i \leftarrow, d_i>\right\}, D>\right) \neq \{\varnothing\} \tag{20}$$

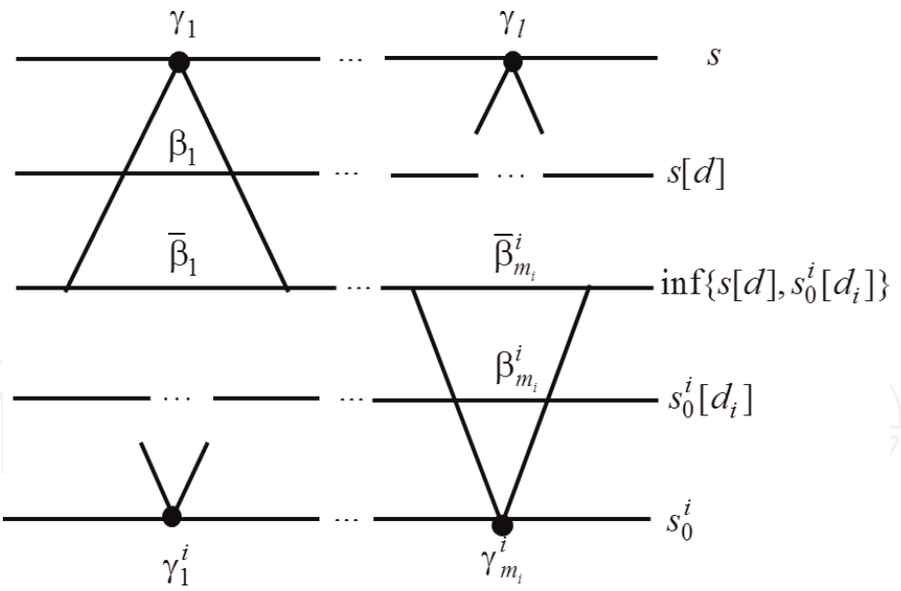

Figure 1.
Graphical illustration of S-unification.

This condition is equivalent to the existence of solution of word equation on context-free language (WECFL):

$$\left[s = s^i_o, d \cup d_i\right], \tag{21}$$

(for correctness, sets of variables of terms s and s^i_o would not intersect, that is, simply implemented by variables renaming, if this condition is not satisfied).

Result of WECFL (21) solution is the new declaration of variables of S-production σ_i, "concreted" in accordance with query $< s, d >$. This concretization will be denoted as $d_i\left(\frac{s^i_o}{s, d}\right)$; it is obtained by replacing variable declarations β^i_j of term s^i_0 by new ones $\overline{\beta}^i_j$, which are more informative ("concrete," "precise"), regarding initial declarations in d_i. This concretization is adequate to query $< s, d >$ in such a way, that CS-productions, whose headers may enter the answer to this query, may be created at the following steps of inference.

Logics of S-unification are illustrated in **Figure 1**, followed by Example 2.

Example 2. Consider query $< s, d >$, where

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

and S-production σ_1 from Example 10, where

$$s^1_0 = AREA\ a\ IS\ s\ AT\ t,$$
$$d_1 = \{a \rightarrow < name\ of\ area >, s \rightarrow < state >, e \rightarrow < i >, t \rightarrow < time >\}.$$

According to (21),

$$s^1_0[d_1] = AREA < name\ of\ area > IS < state > AT < time >,$$
$$s[d] = AREA < name\ of\ area > IS\ NORMAL\ AT < time >,$$
$$d_1\left(\frac{s^1_0}{s, d}\right) = d_1 - \{s \rightarrow < state >\} \cup \{s \rightarrow NORMAL\} =$$

$$= \{a \rightarrow <name\ of\ area>, s \rightarrow NORMAL, e \rightarrow <i>,$$

$$t \rightarrow <time>\}.\blacksquare$$

S-unification provides opportunity of answer 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 [1, 2]. Here we shall describe more natural and understandable **procedural representation** of the considered O-semantics, which fully corresponds to the aforementioned axiomatics.

This representation contains two recursively interconnected algorithmically defined functions.

The first of them Q has two input variables s and d, which values represent query $<s, d>$, and one output variable A, which value, returned after Q termination, is the answer to this query, as defined by (17).

The second function RB has also two input variables, φ and d, which values represent, respectively, the so-called residual body RB of S-production (subset of body of S-production, including terms, which until current call were not used for the derivation of new queries) and variable declaration, obtained after previous steps of logical inference. Function RB returns set $\mathcal{D}$of variable declarations, each corresponding to one element of answer to the query $<s, d>$, where $s \in \varphi$ is the next term, extracted from RB. By this, input variables declaration d is made "more precise."

Text of function Q is as follows:

```
1  Q : function(s, d) returns (A);
2      variables((s, s0) term, (d, d', δ, δ') declaration, φ set of terms,
3                A set of words initial({∅})) local;
4      variables(S set of productions, D set of context − free rules) global;
5      do <s0, φ, d'> ∈ S;
6          δ:=d'(s0/(s, d));
7          if δ ≠ ∇
8            then do δ' ∈ RB(φ, δ);
9                    A : ∪{s0[δ']};
10                 end δ';
11     end S;
12     end Q
```

As seen, lines 2–4 contain declarations of variables, used lower in the function body. It is essential that there are two variables declared as global, namely, S (set of S-productions) and D (metadatabase). Nevertheless the last one is not used in the body, it is presumed that MDB is used inside S-unification (line 6). The initial value of local variable A is an empty set.

The main part of Q is loop (lines 5–11) on S-productions, entering set S and represented in the form of triples $<s_0, \varphi, d'>$, where s_0 is header; φ is body of S-production, being set of terms; and d' is its variable declaration. The first operator inside loop (line 6) implements S-unification of query $<s, d>$ and couple $<s_0, d'>$. If this S-unification is successful (i.e., its result is not a special value ∇), then loop on elements of set of variable declarations, obtained by function RB, is executed (lines 8–10). The values of input variables of function RB are φ (set of terms, entering body of the current S-production) and δ (result of the

aforementioned S-unification). Each δ', being set of variable declarations of the form $\gamma \to u$, where $u \in V^*$, is used for making facts $s_0[\delta']$ by substitution right parts of those declarations to term s_0, and all such facts are joined to the output set A (line 9).

Text of function *RB*, in turn, is as follows:

```
1  RB : function(φ, d) returns (𝒟);
2       variables(φ set of terms, d declaration, s term, w fact,
3                 𝒟 set of declarations initial({∅}) local;
4       variables(D set of context−free rules) global;
5       if φ = {∅}
6         then 𝒟:={d};
7         else do s ∈ φ;
8                 do w ∈ Q(s, d);
9                    𝒟 : ∪RB(φ − {s}, d(s / (w, {∅}))));
10                end w;
11        end s;
12      end RB
```

Here lines 2–4, as in the text of Q, are declarations of variables, used in the *RB* body. *RB* execution begins by check, whether set φ is empty (line 5): if so, search of terms of S-production body is already finished (or, as partial case, it is empty because S-production is axiom), and the returned value is one-element set $\{d\}$, where d is the same, as input value. Otherwise loop on terms, entering residual body, is executed (lines 7–11). Every selected term s is used in query $<s, d>$ to KB <*S,D*>, which is processed by recursive call $Q\ (s,d)$, and loop on all facts, entering answer to this query, is executed (lines 8–10). The body of this loop contains single operator (line 9), providing accumulation of set $\mathcal{D}$ as a result of recursive call of the same function *RB* with the first input variable being $\varphi - \{s\}$ (i.e., set of terms of the body of the processed S-production, reduced by extraction of the processed term s) and the second input variable value being $d\left(\frac{s}{w, \{\varnothing\}}\right)$ (i.e., S-production variables declaration, made "more precise" according to result of S-unification of query $<s, d>$ and fact w).

As it is easy to see, execution of loop $s \in \varphi$ provides generation of $n!$ sequences of terms of body of S-production, i.e., all possible permutations of these terms.

Let $Q(s, d)$ be finite result of function Q application to the query $<s, d>$ and APS KB <*S,D*>. There is theorem on the correctness of the described operational semantics of the augmented Post systems [2].

Theorem 1. $Q(s, d) = Ex(S, D) \cap Ex(<\{s \leftarrow\}, \{d\}>)$.

In fact, Q and *RB* collaborative operation provides top-down derivation of the set of CS-productions, involved in creation of non-ground facts, which enter the answer to the query $<s, d>$, in full accordance with bottom-up definition of M-semantics of APS. (Ground facts are simply selected by one-step application of Q and *RB*.)

As seen from this description, logical inference is implemented in a wavelike manner. "Direct waves," including generated queries, which contain "more and more concrete" variables declarations, are "spreading" until S-axioms are reached. This leads to the creation of CS-productions, and "back waves" start until a new set of facts, being answer to some intermediate (or, finally, initial) query, is assembled. Both direct and back waves meet one another and "interfere," producing new queries. As may be seen, APS operational semantics corresponds to the *universal*

mode of inference (all facts, being goal of the query, are derived), unlike *existential* mode (anyone fact from the possibly multifact set), which is inherent to known resolution procedures. By this, full accordance of APS O-semantics to its S- and M-semantics is achieved.

4. Procedural connection to APS KB

One of the most important features of any knowledge representation model are embedded tools of the procedural connection, providing interaction of "rigid" (non-updated by knowledge engineers) software modules with inference process and data exchange between inference engine and the aforementioned modules.

Background of the **procedural connection to APS knowledge bases** are *program (P-) productions*.

P-production is couple $<s_0 \Leftarrow p, d>$, where term s_0 is header and set d variable declaration; s_0 and d are declarative component of the procedural connection. Symbol $'' \Leftarrow ''$ is the divider, distinct from $'' \leftarrow ''$, and p is the name of the connected program (software module), which has its own extensional $W_p \subseteq Ex$ $(<\{<s_0 \leftarrow, d>\}, D>) \subseteq L(G)$.

Couple $<s_0, d>$ is called *cover of the connected program* p. In general case one and the same program may be connected to APS KB by several P-productions with different covers.

Example 3. Program named MULT for multiplication of integer numbers may be connected to knowledge base by P-production:

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

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

Let Π be the set of names of programs, connected to APS KB. The extensional of this KB will differ from (12) to (14) by the only feature; instead of (12) the following is used:

$$W_{(0)} = \left(\bigcup_{p \in \Pi} W_p\right) \cup \{w | \{w \leftarrow, \{\varphi\}> \in \overline{S}\}, \tag{22}$$

i.e., $W_{(0)}$ is a join of extensionals of all connected programs and set of ground facts, being heads of CS-axioms. Until general case will be discussed, we shall consider programs with the so-called static extentionals, whose basic feature is $W_p = \text{const}$ for all period of query processing.

Concerning operational semantics, it is sufficient to extend function Q by operators, providing call of the "perspective" connected programs via unified application programs interface and joining resulting sets to the accumulated answers. The extension is as follows:

12 **do** $<s_0, p, d'> \in S$;

13 $\delta := d'\left(\frac{s_0}{s, d}\right)$;

14 **if** $\delta \neq \nabla$

15 **then** $A : \cup p(s_0, \delta)$;

16 **end** S;

As seen, after search on S-production set (lines 5–11), similar search on P-production set would be executed (lines 12–16). If S-unification of query $< s, d >$ and program p cover $< s_0, d' >$ is successful, program call is executed, and its result —set of words, denoted $p(s_0, \delta)$—is joined to the accumulated set A. This fully corresponds to (22).

As seen, search on the set S is nondeterministic (there is no any predefined order on this set, except S-productions processing before P-productions; but in general case, this ordering, of course, is not mandatory). This obstacle creates some difficulties in implementation of calls of the connected programs, when input data are less informative, than it is necessary for computation (e.g., MULT$(a * b = c, \{a \rightarrow < integer >, b \rightarrow 4, c \rightarrow < integer >\})$. By this, result of such call may be infinite set, even if there would be an opportunity to implement this operation using tools for N-facts processing, developed in [1–4].

To avoid such difficulty, it would be reasonable to implant to S- and P-productions some information, which may cut off "dangerous" branches.

There is a simple tool for logical inference control within APS knowledge representation. Namely, variables, which the declarations after S-unification would have right parts, consisting only of terminal symbols (i.e., there would be 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. Such variables are called *complete*. In example 3 declarations of variables a and b would be $a \dot{\rightarrow} <$ integer $>$, $b \dot{\rightarrow} <$ integer $>$, so query

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

while inference will result in program MULT call, while query $< x * a = s, \{x \rightarrow < integer >, a \rightarrow 135, s \rightarrow < integer >\} >$ will not, because of information incompleteness of the declaration of x, which is a complete variable. Negative result of S-unification in the case of such incompleteness is, as higher, denoted by symbol ∇.

Let us consider now the general case, where extensionals of programs, connected to APS KB, are not static, i.e., may vary while answer derivation. The simplest examples of such kind of programs are database management systems.

To connect DBMS, operating key-addressed databases, it is sufficient to join to APS KB P-production:

$$< ?k = d \Leftarrow DBMS, \{k \dot{\rightarrow} < key >, d \rightarrow < data >\} >$$

for queries (variable k, denoting key, is complete), as well as

$$< k{:=}d \Leftarrow DBMS, \{k \dot{\rightarrow} < key >, \dot{d} < data >\} >$$

for update operations (both variables k and d, corresponding to key and data, are complete).

DBMS, operating databases with symmetric access, may be connected in a following way. In order to minimize redundant search while queries processing, some substrings of facts may be declared indexed (used for creation and maintenance of dynamic search trees, as it is described in [2, 3]). So some covers of the relational DBMS, corresponding to some subsets of DB, may contain complete variables, with declarations like $\dot{\rightarrow} u$, where $\rightarrow u$ defines domain of the indexed attribute of the relation. Covers, providing inclusion to SDB, contain only complete variables. (If DBMS operate SDB with incomplete information, any variable may be incomplete.)

5. Flow-processing APS

Described approach to the procedural connection makes easy integration of the distributed hardware components, linked by the computer network, into a single system with the single operation process. This may be done by procedural connection of the corresponding hardware drivers and implementation of the so-called flow-processing augmented Post systems.

FAPS KB includes along with S- and P-productions also the so-called flow (F-) productions having the form of

$$< s_0 \rightarrow s_1, \ldots, s_m, s_{m+1}, d > \tag{23}$$

where terms s_0, called activator, and s_{m+1}, called actor, contain only complete variables and symbol "→" (inverse to "←", used in S-production) divides activator and body, including, along with actor, conditions $s_1, \ldots, s_m$. This structure is rather usual as well as operational semantics of F-productions, which is defined by function F, similar to Q and RB, and interconnected with them into integrated algorithmics.

We assume that F-productions are represented in the knowledge base as couples $< s_0, \varphi, s, d >$, where s_0 is activator, φ is set of terms-conditions, and s is actor, while d is variable declaration. By this, APS KB contains three types of objects, corresponding to three types of productions:

1. $< s_0, \varphi, d >$, whereφ is set of terms (S-productions);

2. $< s_0, p, d >$, where p is string, being name of the connected program (P-productions);

3. $< s_0, \varphi, s, d >$ (F-productions).

All these objects are accumulated to set S, which, along with metadatabase D, present in the bodies of functions Q, RB, and F as global variable.

Function F is as follows:

```
1  F : function(w);
2     variables((s0, s') term, (d, δ, δ') declaration, w word, φ set of terms) local;
3     variables(S set of productions, D set of context−free rules) global;
4     do < s0, φ, s, d > ∈ S;
5           δ:=d'(s0 / w, {∅});
6           if δ ≠ ∇
7              then do δ' ∈ RB(φ, δ);
8                       F(s[δ']);
9                    end δ';
10    end S;
11    terminate;
12    end F
```

As seen, the search on F-productions set (lines 4–10) provides selection of such F-productions, which are activated by input message w. The set of conditions of

every activated F-production is interpreted as residual body of S-production (lines 7–9). Every variable declaration δ', obtained as a result of this interpretation, is used for the creation of new message by substitution of δ' to actor *s*. This message is used as input value for function *F* recursive application. So the wave of messages, triggered by initial message, is generated, modeling well-known *blackboard architecture*. This wave propagation of any programs (including DBMS, software/hardware drivers, providing activation and operation of various devices, as well as networking middleware) may be applied. Operator **terminate** (line 11) stops *F* execution, so no return to the parent call is performed.

Various practice-oriented dialects of APS, providing various effective technologies of non-procedural programming (multiactivating, triggers, terms with negators, decision tables, etc.), were developed; also described operational semantics of APS and its extensions were redefined for highly parallel hardware environment [1, 2]. Taking into account nondeterministic nature of operational semantics of APS family, some additional tools for logical inference control were introduced in [1, 2].

Let us consider now the main features of APS application to the most advanced areas—Big Data, Internet of Things, cyberphysical industry, and cybersecurity—forming the future digital economy information infrastructure (DEII) [13, 14]. In all aforementioned applications, APS knowledge representation plays interconnecting and flexifying role, providing fast integration of various heterogeneous systems and fast adaptation of their operation logic to the highly volatile environment. In fact, APS simplify to the maximally possible level the most complicated problem of *interoperability* of any a priori developed systems; APS KB is nothing but "glue," which in the simplest regular way integrates them together.

6. Implementational issues and applications of APS

DEII is the result of integration of three basic paradigms: network-centricity, Big Data, and Internet of Things (**Figure 2**).

Network-centricity provides interconnection of any subjects of digital economy (DE), integrating human society and technosphere into global technosocium, operating as a global megasystem for the mankind wealth and prosperity. Big Data provides storage of great amounts of data from multiple heterogeneous sources and

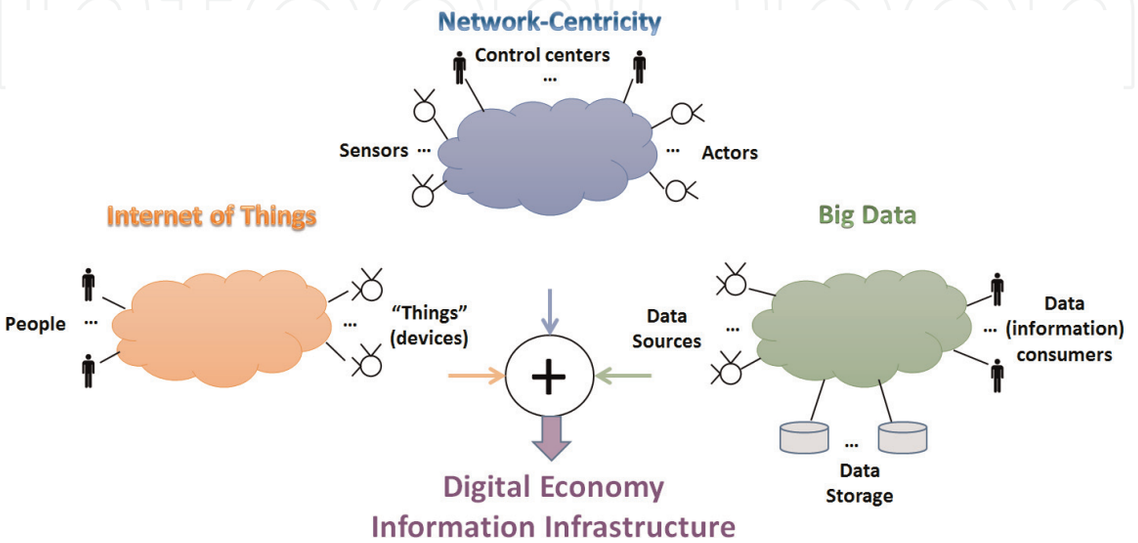

Figure 2.
Basic paradigms of digital economy information infrastructure.

use of these data for rational everyday operation of the aforementioned megasystem and its permanent improvement. The Internet of Things, whose more precise and correct name would be Internet of Devices, provides creation, development, and operation of the aforementioned future global technosphere, including cyberphysical industry, based on additive manufacturing technologies, deeply robotized smart logistics, smart energy generation and delivery systems, and life resources infrastructure (food, water, living houses, etc.), as well as digital banking and finance infrastructure. All these segments, containing many billions of interconnected devices, provide implementation of such ambitious initiatives as Smart City and Smart Nation, inevitably leading to the Smart World as a well-understood and achievable goal of mankind evolution.

The most complicated problem to be solved for DEII creation and development is providing its flexibility, i.e., rapid correction of operation logic of various DEII elements and sets of elements to the constantly occurring changes of the environment, as well as to changes in our knowledge about nature, human society, and technosphere. It is evident that sufficient flexibility of DEII may be achieved only on the background of knowledge engineering, leading to the knowledge-based digital economy.

As shown in **Figure 3**, every subject of the DEII (no matter, human, or device), being associated with unique address of the address space of global information infrastructure, is supported by local knowledge base (LKB), applied by knowledge interpreter-corrector (KIC) for processing of input information flow, as well as local database, used and updated while aforementioned processing. Here LKB may contain not only "soft" component, i.e., rules, defining logic of input messages processing, but also "firm" component, i.e., "rigid" (nonmodified) software modules and systems (up to DBMS clients and servers), connected to LKB by the common interface, and called while rules interpretation (their sets are marked CP, e.g. Connected Programs, with lower indices).

The described approach may be effectively implemented, if there would be some unified data item, providing unified representation of operation logic of any system, joining any set of DE subjects.

The possible practical outcome of the SSF is the creation of toolkit, providing the described higher approach to DEII implementation upon **string as the aforementioned unified data item**. This outcome in the integrated form is placed in

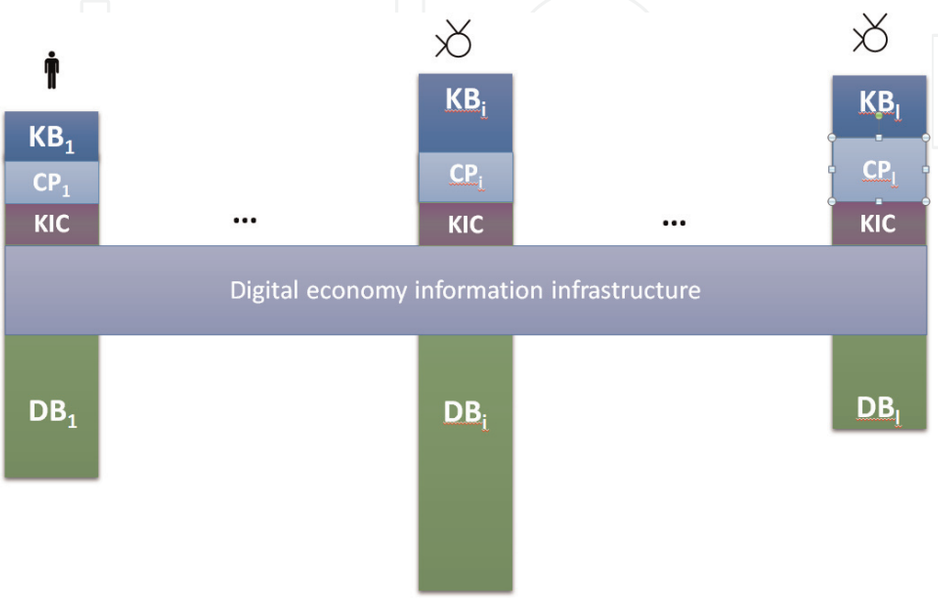

Figure 3.
Linearized representation of the knowledge-based digital economy information infrastructure.

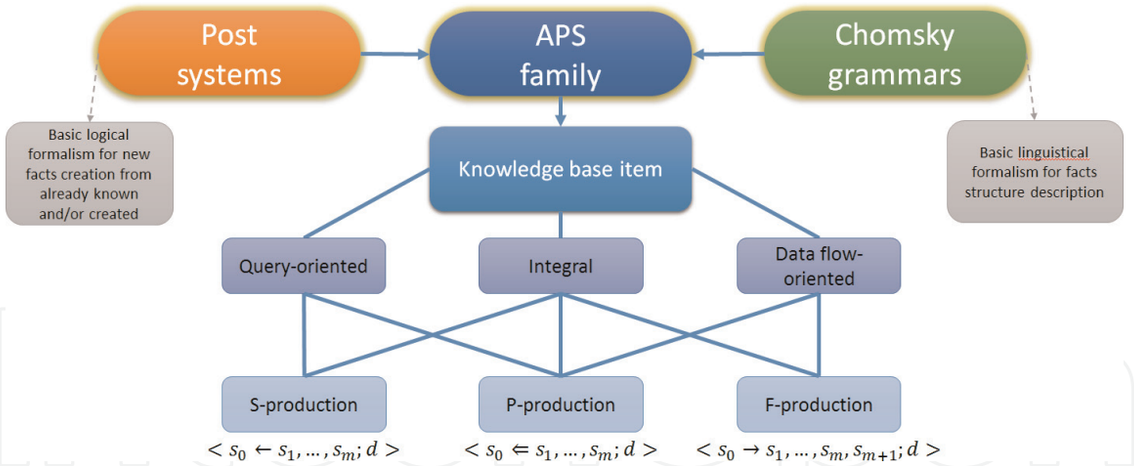

Figure 4.
Integrated representation of APS family of knowledge representation models.

Figure 4, where APS family of string-operation knowledge representation models is explicated.

The operational semantics of the simplest practically used model from this family, whose background is FAPS, is shown in **Figure 5**.

As seen, input messages, which are transported to the local subject of DEII by means of network infrastructure, are recorded on the external blackboard (in fact, there are two blackboards used: external, for messages from the external sources, and internal, for messages generated while current external message processing). Every next message (string) w is navigated by special associative index (binary/ternary tree) to the activators s_o^i. Selected activators are used for solving the corresponding WECFL (to simplify variable declarations and accelerate messages parsing, the so-called ultragrammars were introduced in [1, 2]), and if solutions do exist, they are transferred to the rest part of the F-production, in which the terms may provide *access to databases with symmetric access (DBSA) or key-addressed databases (KADB)* SQL-like or NoSQL queries and updates; *check database integrity* before updates (by applying constraints defined by the corresponding sets of productions); *activate devices* by their drivers, connected to LKB by proper P-productions;

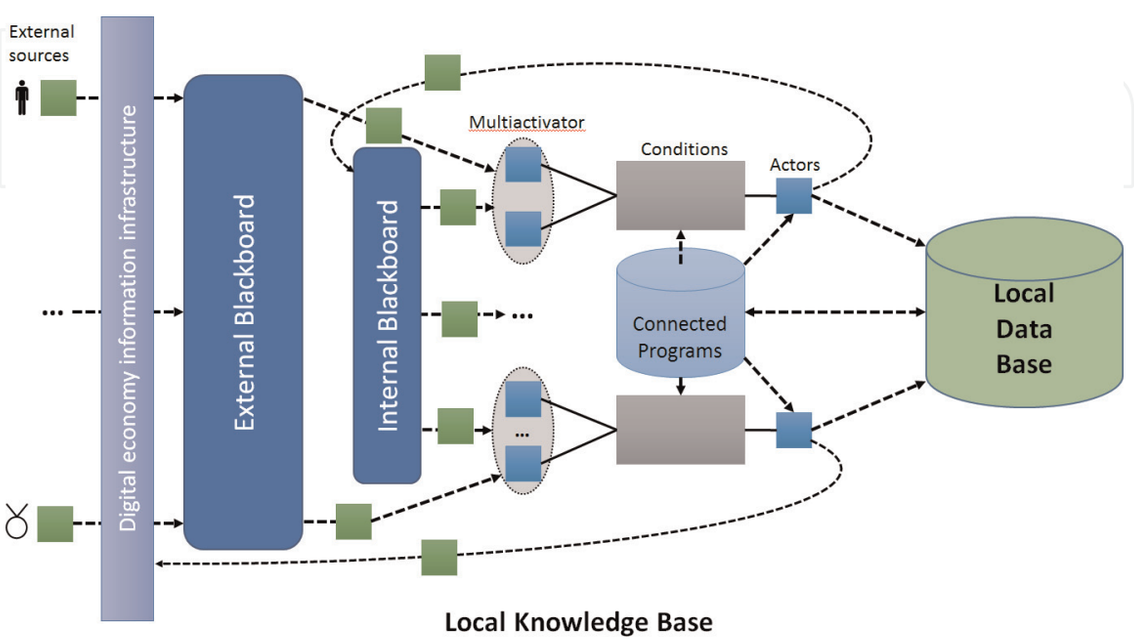

Figure 5.
Basic operation semantics of multiactivated flow APS.

perform some nontrivial *processing by connected "firm" software products*; and also *send messages*, which are created strings to the internal blackboard or, by lower-level communication software, to another local subject (i.e., their external blackboards).

As seen, described dialect of APS family provides flexible implementation of operations, necessary for Big Data and Internet of Things, as well as its cyberphysical manufacturing segment, i.e., industrial Internet of Things. In the simplest case, strings, being transferred to 3D printers by their driver's calls, may be STL files containing layer descriptions of the printed material objects [15].

The main feature, which is, in fact, core for all APS family, is "additivity" of the local knowledge bases. Namely, the occurrence of any new device or human, generating earlier unknown and unprocessed messages, is supported by addition to LKB of subjects (again devices or humans), communicating with source of such messages, new elements with activators, providing initiation of their processing. Such occurrence results, finally, in creation and sending new messages to another subjects. Such "additivity" may be called "vertical" (new types of messages are supported by the addition of new F-productions with new activators to LKB). However, horizontal "additivity" is also supported, when new F-productions with proper activators are added for internal messages, which are sent by some modules of LKB while processing external messages. By this, internal messages processing becomes "deeper." As seen, FAPS provide sufficiently regular and refined technology of the distributed system software development and debugging.

Let us underline that unlike well-known tools from the object programming area, which also support interconnection between modules by messages and use blackboard for such processing, in the case of flow APS, new message receivers are not known, and thus, there is no opportunity for its determination in the text of the program module. By this, APS-based knowledge engineering approach to software making principally differs from object programming as well as any other approaches from the more or less procedural programming.

Let us pay some attention to such important area of flow APS application as cybersecurity, which is critical for DEII normal operation.

Dynamically extended spectrum and complexity of cyberattacks, implementing today advanced persistent threats (APT), have led to the necessity of development of more and more sophisticated tools for early recognition and prevention of APT. The most efficient of such tools are based on the security information and event management (SIEM) paradigm [16–19], whose background technologies are deep packets inspection/deep packets processing (DPI/DPP) [20, 21] and data leakage prevention (DLP) [22, 23]. The first operates flows of bit packets, providing their fusion in order to recognize elaborately covered signatures of known cyberattacks, while the second operates usually traffic from the application level of the OSI model. However, both operate strings (no matter, bit, symbol, or combined), and by this reason the described higher FAPS are the perfect tool for the SIEM implementation, especially at security operation centers (SOC), providing fusion of the primary data, passing to SOC from large amount of software and hardware traffic sensors from the security perimeter of the protected system. (Such sensors may be also effectively implemented on the FAPS background.) It is very important that SIEM is solidly based on three pillars: Big Data, which are stored fragments of the network traffic, caught at many network links in a 24×7 regime; data mining; and knowledge engineering. This area is the hardest known case of data mining and knowledge-based technologies application: cybersecurity knowledge engineers, by analyzing accumulated enormous volumes of primary data items, must quickly understand signatures of the prepared or already performed cyberattacks, developed by the smartest people from the global cybercrime and states special services teams; and rapidly correct LKBs or extend them by new sets of productions,

providing early recognition and neutralizing such attacks. This combination of very large volumes of data and knowledge with hard real-time regime of SOC operation (e.g., SOC of the Russia largest financial group Sberbank neutralizes daily over 14,000 cyberattacks from all over the world [24]) makes cybersecurity application of data and knowledge engineering the most important from both practical and theoretical points of view.

7. Conclusion

The described "Set of Strings" Framework is based on integration of three research areas—knowledge/data engineering, theory of grammars, and information theory—which until now are very close but rather isolated from one another. Synergetic effect, which may be the main result of such integration, would be useful for the development of the unified theory, necessary for convergence of Big Data, data mining, artificial intelligence, and Internet of Things. Such convergence is extremely needed for solution of all spectrum of problems of digital economy.

The most valuable directions of the SSF future development may be the following:

1. Metainference and machine learning in APS KB.

2. N-facts concretization in SDB with incomplete information (SDBI) and APS KB, using functional dependencies, associated usually with the relational data model.

3. Contradictory SDBI and APS KB management.

4. SDBI and APS KB with metadatabases, describing two- and three-dimensional objects.

5. SSF ideology and techniques transfer to the numeric data with the help of the SSF-associated theory of recursive multisets and multiset grammars/metagrammars, developed for the solution of problems of planning and scheduling in digital economy [25–28].

6. Hardware implementation of key elements of SSF in the data flow and other high-parallel computer environments.

The author is highly interested in the cooperation with computer scientists and engineers, who may spend some research effort participating in the development of the listed directions and the SSF at all.

Acknowledgements

The author expresses gratitude to the editor for useful advices. The author is grateful to Prof. Noam Chomsky for words of support to the development of the "Set of Strings" Framework. The author is also thankful to Prof. Jeffrey Ullman for useful remarks on the current state of the theory of grammars.

DOI: http://dx.doi.org/10.5772/intechopen.86207

Author details

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

*Address all correspondence to: sheremet@rfbr.ru

References

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

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

[3] Sheremet I. Data and knowledge bases with incomplete information in a "Set of Strings" framework. International Journal of Engineering and Applied Sciences. 2016;**3**(3):90-103

[4] Sheremet IA. Augmented Post systems: String-operating knowledge representation for Big Data and Internet of Things applications. Geoinformatics Research Papers. 2017;**5**:BS1002. DOI: 10.2205/CODATA 2017

[5] Chomsky N. Syntactic Structures. 2nd ed. Berlin–New York: Mouton de Gruyter; 2002. p. 117

[6] Post EL. Formal reductions of the general combinatorial decision problem. American Journal of Mathematics. 1943; **65**:197-215

[7] Robinson JA. A machine-oriented logic based on the resolution principle. Journal of the ACM. 1965;**12**:23-41

[8] Kowalski RA. Algorithm = Logic + Control. Communications of the ACM. 1979;**22**(7):424-436

[9] Nilsson U, Maluszynski J. Logic, Programming and Prolog. 2nd ed. John Wiley & Sons; 2000. p. 282

[10] Cali A, Gottlob G, Lukasiewicz T. A general datalog-based framework for tructable query answering over ontologies. Journal of Web Semantics. 2012;**14**(1):57-83

[11] Miller D, Nadathur G. Programming with Higher-Order Logic. Cambridge University Press; 2012. p. 322

[12] Gallaire H, Minker J, Nicolas J-M. Logic and databases: A deductive approach. ACM Computing Surveys. 1984;**16**(2):153-185

[13] The Global Information Technology Report 2016. In: Baller S, Dutta S, Lanvin B, eds. Innovating in the Digital Economy. INSEAD, 2016. p. 62

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

[15] Eyers DR, Potter AT. Industrial additive manufacturing: A manufacturing systems perspective. Computers in Industry. 2017;**92-93**: 208-218. DOI: 10.1016/j. compind.2017.08.002

[16] Miller DR, Harris S, Harper A, Vandyke S. Security Information and Event Management (SIEM) Implementation. McGraw Hill; 2010. p. 464

[17] Miller DR, Harris S, Harper A, Vandyke S, Blask C. Security Information and Event Management (SIEM) Implementation. McGraw Hill; 2011. p. 465

[18] Sweeny J. Creating Your Own SIEM and Incident Response Toolkit Using Open Source Tools. SANS; 2011. p. 24. Available at: //sans.org/

[19] Limacher M, Kurz U. Security operation centre in action. CryptoMagazine. 2013;(3):10-12

[20] Fuchs C. Implications of Deep Packet Inspection (DPI) Internet

Surveillance for Society. Research Paper. Uppsala University. 2012. p. 125

[21] Bass T. Intrusion detection systems and multisensor data fusion. Communications of the ACM. 2000; **43**(4):99-105

[22] Papadimitriou P, Garcia-Molina H. Data leakage detection. IEEE Transactions on Knowledge and Data Engineering. 2011;**23**:51-63

[23] Shabtai I, Elovici Y, Rokach L. A Survey of Data Leakage Detection and Prevention Solutions. NY: Springer-Verlag; 2012. p. 92

[24] Sheremet IA. Digital economy and cybersecurity of its financial sector. Proceedings of Free Economical Society. 2018;**210**:23-34 (In Russian)

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

[26] Sheremet IA. Multiset approach to the estimation of consequences of natural disaster impacts on industrial systems. Geoinformatics Research Papers. Sochi 2016;**4**:BS4002, DOI: 10.2205/2016 BS01 .

[27] 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

[28] Gvishiani AD, Roberts FS, Sheremet IA. On the assessment of sustainability of distributed sociotechnical systems to natural disasters. Russian Journal of Earth Sciences. 2018;**18**:ES4004. DOI: 10.2205/2018ES000627