ISBN 978-3-00-034815-0

I.A. Sheremet

Doctor of Engineering science, professor,member of the European Academy of Natural Sciences e.V.,

Word Equations on Context-Free Languages

Special Edition:

Hannover Annual

Vol.1, 2011

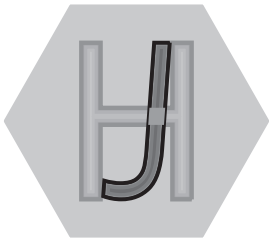
Series: Applied computer science

Printed by decision of European Academy of Natural Sciences e.V.,

Hannover 2011

I.A. Sheremet
Doctor of Engineering science,
professor, member of the EANS

Word Equations on Context-Free Languages



Special Edition:
Hannover Annual
Vol.1, 2011
Series: Applied computer science

Printed by decision of European
Academy of Natural Sciences e.V.,

ISBN 978-3-00-034815-0

Hannover 2011

Europäische Akademie der Naturwissenschaften
Gegründet 2002

30161 Hannover
Husarenstr. 20
E-mail: vladimirt2007@googlemail.com

Auflage: 100 St.

Abstracts

The article introduces the concept of word equation on context-free (CF) language and the finite representation (unifier) of the set of solutions for this type of equation. The article offers an algorithms for constructing the minimal unifier of word equation on CF language, where the last is described by the unambiguous acyclic CF grammar. The single and multiple variable instances cases are investigated as well as ambiguous and cyclical CF grammars cases of the designed mathematical apparatus. The incorrectness of J.A.Robinson's unification algorithm and resolution procedure in general case is proved. Linear and non-linear systems of word equations on CF languages are proposed and investigated. Some possible applications of the proposed concept to data and knowledge mathematical modelling as well as to genetics and social engineering are considered.

About the author

Sheremet Igor Anatolyevich, born in 1956, Doctor of Engineering Science (1993), Professor (1998), Professor at N.E. Bauman Moscow State Technical University. Member of Russian Academy of Natural Sciences and European Academy of Natural Sciences. L.Euler Medal Awards by European Academy of Natural Sciences for the fundamental results in data and knowledge bases mathematical modelling, systems analysis and information theory. P.L.Kapitsa (Nobel Prize winner) Medal Awards by International Association of the Authors of Scientific Discoveries for the monograph "Intelligent Software Medias for Computerized Information Processing Systems".

Contents

1. Word Equations

Assume some alphabet $V=\{\upsilon_1,...,\upsilon_m\}$, with the set of strings in this alphabet denoted by V^*, and the set of non-empty strings denoted by V^+. String $s\in(V\cup\Gamma)^+$ is called a term with Γ being the set of variables, and the substitution is called the set

$$d=\{\gamma_1\to u_1,...,\gamma_n\to u_n\}, \tag{1}$$

where $\gamma_1,...,\gamma_n\in\Gamma$ are variables, "$\to$" is the substitution sign and $u_1,..u_n\in V^*$ are the strings in the alphabet V. Term *s*[*d*] is the result of substitution *d* in term *s;* term *s*[*d*] is obtained from s by the means of substituting variables $\gamma_i\in\{\gamma_1,...,\gamma_n\}$ by strings u_i which correspond to these variables as per (1). If

$$s=u_1\gamma_{i_1}u_2...u_j\gamma_{i_j}u_{j+1}...u_l\gamma_{i_l}u_{l+1}, \tag{2}$$

then

$$s[d]=u_1\overline{u}_{i_1}u_2...u_j\overline{u}_{i_j}u_{j+1}...u_l\overline{u}_{i_l}u_{l+1}, \tag{3}$$

where

$$\overline{u}_{i_j}=\begin{cases}u_{i_j}, if\ \gamma_{i_j}\in\{\gamma_1,...,\gamma_n\}\\ \gamma_{i_j}\quad otherwise\end{cases} \tag{4}$$

Substitution *d* is called terminal substitution relative to term $s\in(V\cup\Gamma)^*-V^*$, if

$$s[d]\in V^*, \tag{5}$$

i.e. if the result of this substitution in term s is represented by a word in alphabet V. In this case, obviously, *s* and *d* are such that $\{\gamma_{i_1},...,\gamma_{i_l}\}\subseteq\{\gamma_1,...,\gamma_n\}$.

From [1-3] a word equation is the structure

$$s=s', \tag{6}$$

where s and s' are terms such that term ss' contains at least one variable and "=" is the equality sign. Terminal substitution *d* with respect to terms s and s' is the solution to equation (6), when being applied to these terms, (6) becomes an identity:

$$s[d]\equiv s'[d], \tag{7}$$

where "≡" is the identity sign.

A set of solutions to (6) denoted by $D[s=s']$ is a set of terminal substitutions relative to terms s and s'

$$D[s=s']=\{d\mid s[d]\equiv s'[d]\}. \tag{8}$$

Obviously, in general $D[s=s']$ is infinite; therefore there is no algorithm for building this set in a finite set of steps.

Given the above and depending on the practical purpose of this equation, one may take two different approaches, which are conventionally called computational and analytical.

Computational, or existential, approach is limited to construct the only one solution from the set $D[s = s']$ by the means of common algorithmics [1,2,4-6].

The finite representation of the family $D[s = s']$ is constructed under analytical, or universal, approach; this finite representation is the substitution

$$d = \{\gamma_1 \to \bar{s}_1, ..., \gamma_n \to \bar{s}_n\}, \tag{9}$$

where $\bar{s}_i \in (V \cup \Gamma)^*$ are terms which in general contain variables; the substitution is such that the following holds as per the above,

$$s[d] \equiv s'[d]. \tag{10}$$

The substitution *d* is called unifying substitution or the unifier. In the strict sense the lack of constraints for variables in terms $\bar{s}_1, ..., \bar{s}_n$ makes the set of unifiers unlimited. Therefore the concept of the smallest general unifier (SGU) is introduced; by substituting the terms from the set $(V \cup \Gamma)^*$ for the variables in terms $\bar{s}_1, ..., \bar{s}_n$ so that (10) becomes identify one may obtain any other unifier.

The construction of SGU defined by (9) is called extended or string unification [1,3] unlike the unification proposed by J.A.Robinson [7], which serves as the basis for the resolution procedure providing the logical inference in various applied versions of first order predicate logics, and in particular in Horn clauses being the parent language of logical programming [8,9].

However result [3] applies to string unification, and according to this result the problem of building the SGU set of terms s and s' in general does not have an algorithmic solution.

On the other hand the attempts to design such an algorithm for the useful and applied modifications of word equations are quite practicable.

The word equations on context free languages discussed in this article represent one of such modifications.

2. Main Concepts and Definitions

Context-free language $L(G) \subseteq V^*$ is generated by the means of CF grammar

$$G = < V, A, \alpha_0, R >, \tag{11}$$

where V and $A = \{\alpha_0, \alpha_1, ..., \alpha_n\}$ are correspondingly terminal and non-terminal alphabets, $\alpha_0 \in A$ is grammar axiom and *R* is the set of rules in the form $\alpha \to \beta$ under which $\alpha \in A$, $\beta \in (V \cup A)^*$; this set is also called the scheme of the grammar, or, for short, scheme [10]. The application of the rule $\alpha \to \beta$ to the string $x = x_1 \alpha x_2$ involves the substitution of the string β for the non-terminal symbol ("non-terminal") α which results in the string $x' = x_1 \beta x_2$, and is generally expressed as $x \Rightarrow x'$ and is called direct generation. Note that the use of the same symbol "$\to$" within the conceptual framework of the word equations

and CF grammars is quite natural, since in both cases the symbols (variables, non-terminals) *de facto* are substituted by the corresponding strings. Crucial semantic difference between seemingly identical structures is that when a substitution is applied to a term (similarly as when it is applied to a word equation) then all the variable instances in the term (terms) are substituted by the only one corresponding string; however, when there are several instances of non-terminal α in string $x \in (V \cup A)^+$ and when the scheme R contains several rules $\alpha \to \beta_1,...,\alpha \to \beta_m$ then each non-terminal instance may be substituted by any string β_i, $i=1,..., m$.

"Generation" relation within the set of strings in alphabet $V \cup A$ is transitive reflexive closure of the direct generation relation and is denoted by $\Rightarrow^*$. Furthermore $x \Rightarrow^* x'$, if either $x = x'$ or $x \Rightarrow x'$, or there is a string $x'' \in (V \cup A)^*$ such that $x \Rightarrow x''$, $x'' \Rightarrow^* x'$.

String $w \in V^*$ is a word in language $L(G)$, if $\alpha_0 \Rightarrow^* w$, therefore

$$L(G) = \{w \mid \alpha_0 \Rightarrow^* w \,\&\, w \in V^*\}. \tag{12}$$

String $x \in (V \cup A)^*$, generated (derived) from the axiom α_0 of CF grammar G, is called the sentential form (SF) of this grammar, and a set of SFs is hereafter denoted by *SF(G)*:

$$SF(G) = \{x \mid \alpha_0 \Rightarrow^* x\}. \tag{13}$$

Therefore, the word $w \in L(G)$ represents sentential form free of non-terminals, and $L(G) \subset SF(G)$.

Assume the generation under which x substitutes for the left non-terminal of all non-terminals in SF, i.e. the rule $\alpha \to \beta$ involves transformation $x = u\alpha x'$ to $u\beta x'$, where $u \in V^*, x' \in (V \cup A)^*$; this type of generation is called canonical generation. Concurrently CF grammar G is called unambiguous if there is only one canonical generation for each string w in language *L(G)*. Alternatively, ambiguous CF grammar G allows for the existence of at least one string in CF language *L(G)* with more than one canonical generation.

CF grammar which allows for $x \Rightarrow^+ x$ generation is called cyclical. Alternatively, CF grammar which makes this type of generations impossible is called acyclic.

Example 1. Assume, CF grammar G with non-terminal alphabet $A = \{\alpha_0, \alpha_1, \alpha_2\}$, terminal alphabet $V = \{a\}$ and scheme $R = \{\alpha_0 \to \alpha_1, \alpha_0 \to \alpha_2, \alpha_1 \to a, \alpha_2 \to a\}$ is ambiguous, since the word a is assigned a couple of canonical generations: $\alpha_0 \Rightarrow \alpha_1 \Rightarrow a$ and $\alpha_0 \Rightarrow \alpha_2 \Rightarrow a$. Moreover, G is acyclic, but grammar G' with the same V and A, and scheme $R' = R \cup \{\alpha_1 \to \alpha_2, \alpha_2 \to \alpha_1\}$ is cyclical, since it enables generation $\alpha_1 \Rightarrow^+ \alpha_1$ (similar to $\alpha_2 \Rightarrow^+ \alpha_2$). ■

The problem of the recognition of the cyclicity of CF grammars is algorithmically solvable; however, the problem of the recognition of their ambiguity is non-solvable [10].

The following construction is a word equation on CF language *L(G)* generated by CF grammar $G = < V, A, \alpha_0, R >$:

$$< s = s', \delta >, \tag{14}$$

where the first component in this sequence is called the kernel and has syntax similar to a word equation; $\delta = \{\gamma_1 \to \beta_1, ..., \gamma_l \to \beta_l\}$ is the suffix, and it defines sets of the values for variables $\gamma_1, ..., \gamma_l$, which have place in terms s and s', by the means of strings $\beta_1, ..., \beta_l$ in alphabet $V \cup A$. Equation (14) may be read as "$s = s'$, where δ". The set of the values for variable γ_i is a set of strings in terminal alphabet V which are generated by grammar G from string β_i (denoted by symbol V which is different from V):

$$\mathrm{V}(\gamma_i, \delta) = \left\{u \mid \gamma_i \to \beta_i \in \delta \,\&\, \beta_i \Rightarrow^* u \,\&\, u \in V^*\right\}. \tag{15}$$

Suffix δ defines a set of substitutions which are denoted by $\sum_\delta$ and which are considered terminal substitutions relative to terms s and s':

$$\sum\nolimits_\delta = \bigcup_{\substack{u_1 \in \mathrm{V}(\gamma_1, \delta) \\ \cdots \\ u_l \in \mathrm{V}(\gamma_l, \delta)}} \{\{\gamma_1 \to u_1, ..., \gamma_l \to u_l\}\}, \tag{16}$$

such that strings $s[d]$ and $s'[d]$ in alphabet V correspond to each term substitution $d \in \sum_\delta$. If terminal substitution d is such that

$$s[d] \equiv s'[d], \tag{17}$$

then it is called the solution to equation (14). Therefore the family of solutions to a word equation in CF language $L(G)$ $< s = s', \delta >$ is the following set:

$$D[s = s', \delta] = \left\{ d \mid d \in \sum\nolimits_\delta \,\&\, s[d] \equiv s'[d] \right\}. \tag{18}$$

By default only so-called correct equations are considered; strings $s[\delta]$ and $s[\delta']$ in alphabet $V \cup A$ within these equations are sentential forms of grammar G. The fulfillment of these conditions ensures that

$$\mathrm{V}(s, \delta) = \left\{w \mid s[\delta] \Rightarrow^* w \,\&\, w \in V^*\right\} \subseteq L(G), \tag{19}$$

$$\mathrm{V}(s', \delta) = \left\{w \mid s'[\delta] \Rightarrow^* w \,\&\, w \in V^*\right\} \subseteq L(G), \tag{20}$$

where the set of strings determined by term $s(s')$ and suffix δ are denoted with the same structure which is used for denoting a set of values for the variable γ_i in (15). One may easily verify that this type of generalization is quite acceptable, since γ_i is a term and, under (19)-(20), $\mathrm{V}(\gamma_i, \beta) = \left\{w \mid \gamma_i[\beta] \Rightarrow^* w \,\&\, w \in V^*\right\} = \left\{w \mid \gamma_i[\beta_i] \Rightarrow^* w \,\&\, w \in V^*\right\} = \left\{w \mid \beta_i \Rightarrow^* w \,\&\, w \in V^*\right\}$, which corresponds to (15).

Example 2. Consider CF grammar G with non-terminal alphabet $A = \{\alpha_0, \alpha_1, \alpha_2\}$, terminal alphabet $V = \{0,1\}$ and scheme $R = \{\alpha_0 \to \alpha_1, \alpha_1 \to \alpha_2\alpha_1\alpha_2, \alpha_1 \to 0, \alpha_2 \to 1\}$. Obviously $L(G) = \{1^n 01^n \mid n \geq 0\}$. Consider the following word equation in CF language *L(G)*:

$$< 11\gamma_1 = \gamma_2 1, \ \{\gamma_1 \to \alpha_1 11, \ \gamma_2 \to 11\alpha_1 1\} >. \tag{21}$$

For this equation

$$\mathrm{V}(\gamma_1, \delta) = \left\{u \mid \alpha_1 11 \Rightarrow^* u \,\&\, u \in \{0,1\}^*\right\} = \left\{1^n 01^{n+2} \mid n \geq 0\right\}, \tag{22}$$

$$\mathrm{V}(\gamma_2,\delta)=\left\{u \mid 11\alpha_1 1 \Rightarrow^{*} u \,\&\, u\in\{0,1\}^{*}\right\}=\left\{1^{n+2}01^{n+1} \mid n\geq 0\right\}. \quad (23)$$

Therefore,

$$\mathrm{V}(s,\delta)=\mathrm{V}(11\gamma_1,\delta)=\left\{1^{n+2}01^{n+2} \mid n\geq 0\right\}\subseteq L(G), \quad (24)$$

$$\mathrm{V}(s',\delta)=\mathrm{V}(\gamma_2 1,\delta)=\left\{1^{n+2}01^{n+2} \mid n\geq 0\right\}\subseteq L(G), \quad (25)$$

$$\sum\nolimits_{\delta}=\left\{\left\{\gamma_1\to 1^{n}01^{n+2},\gamma_2\to 1^{m+2}01^{m+1}\right\} \mid n\geq 0\,\&\, m\geq 0\right\}. \quad (26)$$

Terminal substitution $d=\{\gamma_1\to 10111,\gamma_2\to 111011\}$ is one of the solutions to (21), since $s[d]$=1110111, $s'[d]$=1110111, and therefore $s[d]\equiv s'[d]$.

The set of solutions to (21) is

$$D[s=s',\delta]=\left\{\left\{\gamma_1\to 1^{n}01^{n+2},\gamma_2\to 1^{n+2}01^{n+1}\right\} \mid n\geq 0\right\}. \quad (27)$$

Furthermore, the set of solutions to the word equation

$$11\gamma_1=\gamma_2 1, \quad (28)$$

which is the kernel of equation (21), is

$$D[s=s']=\left\{\left\{\gamma_1\to u1,\gamma_2\to 11\ u\right\} \mid u\in\{0,1\}^{*}\right\}. \blacksquare \quad (29)$$

As we can see, syntax difference between the word equation on CF language (14) and the word equation (6) is the suffix added to the later; as a result the sets of values for variables γ_i presenting in terms s and s' are the sets $\mathrm{V}(\gamma_i,\delta)$, but not V^{*} as in (6). Moreover $s[d]$ and $s'[d]$ in (6) may be any strings in alphabet V while in (14) they may only be the words in language $L(G)$; as a result (14) is called a word equation "on CF language". Further, for short, we shall call it simply "equation".

Having made these refining remarks let us now consider the algorithmic framework for constructing a finite representation of the family of solutions of (14), i.e. set $D[s=s',\delta]$.

3. The Form of the Finite Representation of a Set of Solutions of the Equation

The problem of constructing any type of finite representation $D[s=s',\delta]$ is twofold. The first task addresses the issue of specifying the form of this representation, and the second task concerns the development of the algorithmic complex which will construct this representation in finite set of steps.

First, consider the easier issue. Assume x and x' are a couple of the sentential forms of unambiguous and acyclic grammar *G*. Each of these sentential forms naturally defines a set of strings of language *L(G)* generated (derived) from this SF:

$$W_x=\left\{w \mid x\Rightarrow^{*} w \,\&\, w\in V^{*}\right\}, \quad (30)$$

$$W_{x'}=\left\{w \mid x'\Rightarrow^{*} w \,\&\, w\in V^{*}\right\}, \quad (31)$$

and therefore sentential form *x(x')* is the finite representation of the set $W_x(W_{x'})$. The following lemma holds (presented without proof):

Lemma 1. If

$$W=W_x\cap W_{x'}\neq\{\varnothing\}, \quad (32)$$

then there is a sentential form y such that

$$W = \left\{ w \mid x \Rightarrow^* y \;\&\; x' \Rightarrow^* y \;\&\; y \Rightarrow^* w \;\&\; w \in V^* \right\}. \blacksquare \tag{33}$$

Consequently the intersection of sets W_x and $W_{x'}$ is a set of words of *L(G)* generated from sentential form *y,* which itself is generated from SF x and x' simultaneously.

Example 3. Consider CF grammar *G* from Example 2 and both of its sentential forms $x = 1\alpha_1\alpha_2$ and $x' = \alpha_2 0\alpha_2$. Here $W_x = \left\{ 1^n 01^n \mid n \geq 1 \right\}$, $W_{x'} = \{101\}$, $W = W_x \cap W_{x'} = \{101\}$. There are two sentential forms generated from x and x', and both generate the word 101: $y = 10\alpha_2$ and $y = 101$. ■

Obviously SF *y* in (33) is the finite representation of the non-empty intersection of sets W_x and $W_{x'}$, which is infinite in general.

This finding lays at the basis of the approach for constructing the finite representation of the set $D[s = s', \delta]$.

We shall be restricted by the case of the single instances of variables in equation (14) or, which is the same, in term $s's$ (or ss').

Obviously, if

$$W = \mathrm{V}(s, \delta) \cap \mathrm{V}(s', \delta) = \{\varnothing\} \tag{34}$$

then the equation $< s = s', \delta >$ does not have a solution, i.e.

$$D[s = s', \delta] = \{\varnothing\}, \tag{35}$$

and if

$$W = \mathrm{V}(s, \delta) \cap \mathrm{V}(s', \delta) \neq \{\varnothing\} \tag{36}$$

and since $s[d]$ and $s'[d]$ are the sentential forms of grammar *G*, then according to Lemma 1 the finite representation of the set *W* is SF *y* generated (derived) from $s[d]$ and $s'[d]$ simultaneously. Consequently, the finite representation of the set $D[s = s', \delta]$ is the set

$$\bar{\delta} = \left\{ \gamma_1 \to \bar{\beta}_1, ..., \gamma_l \to \bar{\beta}_l \right\}, \tag{37}$$

such that

$$W = \left\{ w \mid s[\bar{\delta}] = s'[\bar{\delta}] = y \;\&\; y \Rightarrow^* w \;\&\; w \in V^* \right\}. \tag{38}$$

It is easy to verify that $\bar{\beta}_1, ..., \bar{\beta}_l$ are the strings in alphabet $V \cup A$, generated from strings $\beta_1, ..., \beta_l$ accordingly by derivations $s[d] \Rightarrow^* y$ and $s'[d] \Rightarrow^* y$.

The set $\bar{\delta}$ will be named the unifier of equation (14) for short.

Example 4. Consider CF grammar *G* generating atomic formulas ("atoms") of the predicate logic language, which corresponds to the elements of triadic relations $P_i \subseteq B \times C \times E$, *i=1,...,m*, and which are written as $p : b / c / e$, where *p* is the name of the relation, and *b,c,e* are the values from the sets *B,C* and *E* accordingly; ":" and "/" are the separators. The scheme of this grammar may look the following way:

$$
\begin{aligned}
&\langle atom\rangle \to \langle P\rangle : \langle B\rangle / \langle C\rangle / \langle E\rangle \\
&\langle P\rangle \to p_1, \\
&\ldots \\
&\langle P\rangle \to p_m, \\
&\langle B\rangle \to b_1, \\
&\ldots \\
&\langle B\rangle \to b_n, \\
&\langle C\rangle \to c_1, \\
&\ldots \\
&\langle C\rangle \to c_l, \\
&\langle E\rangle \to e_1, \\
&\ldots \\
&\langle E\rangle \to e_k,
\end{aligned}
\tag{39}
$$

where "⟨" and "⟩" are metalinguistic brackets and they are different from angle brackets used for lists, $A = \{\langle atom\rangle, \langle P\rangle, \langle B\rangle, \langle C\rangle, \langle E\rangle\}$, the axiom of this grammar is $\langle atom\rangle$, and $p_1,\ldots,p_m,b_1,\ldots,b_n,c_1,\ldots,c_l,e_1,\ldots,e_k$ are strings in terminal alphabet *V,* which do not contain separators ":" and "/". It follows from (39) that *G* is unambiguous and acyclic.

The word equation on CF language *L(G)* is the following:

$$< p_1 : x / y / e_1 = z : t / c_2 / s,\ \{x \to \langle B\rangle, y \to \langle C\rangle, z \to \langle P\rangle, t \to \langle B\rangle, s \to \langle E\rangle\} > . \tag{40}$$

Obviously, the equation is correct. According to (37), the finite representation of the set of solutions to this equation will be the following set

$$\bar{\delta} = \{x \to \langle B\rangle, y \to c_2, z \to p_1, t \to \langle B\rangle, s \to e_1\}. \tag{41}$$

Here

$$s[\delta] = p_1 : \langle B\rangle / \langle C\rangle / e_1, \tag{42}$$

$$s'[\delta] = \langle P\rangle : \langle B\rangle / c_2 / \langle E\rangle, \tag{43}$$

$$s[\bar{\delta}] = s'[\bar{\delta}] = p_1 : \langle B\rangle / c_2 / e_1 . \blacksquare \tag{44}$$

Generally, there may be more than one unifier to equation (14); it relies on the possibility of the sentential forms of grammar *G* which are connected to SF *y* in (38) by means of "generation" relation. Moreover, both $y' \Rightarrow^* y$ and $y \Rightarrow^* y''$ are possible, and (38) will apply to both y' and y''. As a result the next paragraph introduces the concept of the minimal unifier similar to SGU within the resolution procedure; the algorithmic framework presented hereafter facilitates the construction of minimal unifiers for equations considered.

4. The Construction of Minimal Unifier of Equation with Single Instances of the Variables

Assume that $G=<V, A, \alpha_0, R>$ is still unambiguous acyclic CF grammar. The following algorithm for constructing the unifier makes use of Lemma 2 known from [11]:

Lemma 2. Set *SF(G)* of the sentential forms of unambiguous acyclic CF grammar *G* is partially ordered by the relation $\Rightarrow^*$.■

Moreover, set *SF(G)* contains maximal element α_0, since for every SF $x \in SF(G)$ $\alpha_0 \Rightarrow^* x$. There is a set of upper bounds $\overline{X}$ for each subset $X \subseteq SF(G)$, and this set of upper bounds is such that for each element $\bar{x} \in \overline{X}$ and each element $x \in X$ the following holds: $\bar{x} \Rightarrow^* x$. Naturally, there is minimal upper bound $\sup_G X$ for set *X,* and this minimal upper bound is the sentential form of grammar *G* such that for any other SF $\bar{x} \in \overline{X}$ the following holds: $\bar{x} \Rightarrow^* \sup_G X$. In general subset $x \in SF(G)$ may have a set of its lower bounds $\underline{X}$, and for each element of this subset $\underline{x} \in \underline{X}$ and for each element $x \in X$ $x \Rightarrow^* \underline{x}$. Moreover, there may be maximal lower bound $\inf_G X$ among all lower bounds, and this maximal lower bound will be the sentential form such that for any other SF $\underline{x} \in \underline{X}$ the following will hold: $\inf_G X \Rightarrow^* \underline{x}$.

Example 5. Consider grammar *G* from Example 4 and set $X = \{\langle P\rangle : b_1 / \langle C\rangle / e_2, \langle P\rangle : \langle B\rangle / c_1 / e_2, p_2 : \langle B\rangle / c_1 / e_2\}$, for which

$$\sup_G X = \langle P\rangle : \langle B\rangle / \langle C\rangle / e_2, \tag{45}$$

$$\inf_G X = p_2 : b_1 / c_1 / e_2. \blacksquare \tag{46}$$

Getting back to the issue presented in paragraph 3 dealing with the problem of the finite representation of the intersection of two sets W_x and $W_{x'}$ represented by sentential forms x and x', notice, that SF $y = \inf_G \{x, x'\}$ is minimal among all these representations since all other representations are derivatives of SF $y = \inf_G \{x, x'\}$: any other representation is either SF y' which is generated from *y*, or finite set consisting from SF $y'_1, ..., y'_k$, $k \geq 1$, generated from SF y and such that

$$\bigcup_{i=1}^{k} W_{y'_i} = W_y. \tag{47}$$

Having said the above, we will refer to the finite representation of the set of sentential forms $X \subseteq L(G)$ as to its minimal finite representation. In the same sense, the unifier of equation (14) corresponding to SF $y = \inf_G \{s[\bar{\delta}], s'[\bar{\delta}]\}$ will also be minimal unifier, therefore in subsequent discussion we will refer exactly to this minimal unifier.

As part of the next comment to the concepts and definitions introduced, we would like to note that conceptually this framework is one of a number of possible approaches to the refinement and the development of algorithmic framework to define the concept “the quantity of information” [12].

Notably, if the sentential forms of unambiguous acyclic CF grammar *G* are "constructive objects" by A.N. Kolmogorov (and the grammar itself is “the method of

programming" according to the definitions in [12]), then constructive object $x' \in SF(G)$ is more complex compared to constructive object $x \in SF(G)$, if $x \Rightarrow^{+} x'$, i.e. x' is generated from x in grammar G (when $x \Rightarrow^{*} x'$, object x' is at least as complex as x, since either $x = x'$ or $x \Rightarrow^{+} x'$). The length of the "program" implementing the generation of x' from x (in [12] the length is denoted by *l(p)*), will naturally be the number of applications of the rules of CF grammar within the derivation $x \Rightarrow^{*} x'$ (derivation length of $x \Rightarrow^{*} x'$). Along with term "relative complexity" we shall use of term "relative informativity" as in [11], and therefore object x' is more informative compared to object x (contains more information), if $x \Rightarrow^{+} x'$. Given that, $\sup_G X$ is the object with the highest informativity compared to other objects which are less informative compared to objects from set *X*.

Example 6. Speaking of grammar *G* from Example 4, object $x' = \langle P \rangle : b_1 / \langle C \rangle / e_2$ is more informative compared to object $x = \langle P \rangle : \langle B \rangle / \langle C \rangle / e_2$ since $x \Rightarrow^{*} x'$. Obviously it is the result of x' containing string b_1 in alphabet *V*, which is generated from non-terminal $\langle B \rangle$. ■

The assumption of unambiguity and acyclicity properties of CF grammar *G* and the assumption of single instances of all the variables in equation $< s = s', \delta >$ makes use of the following algorithm for constructing the minimal unifier of that equation:

begin;
if $y = \inf_G \{ s[\delta], s'[\delta] \}$ exists
then do;
construction of generation $s \Rightarrow^{*} y$ in CF grammar
$\overline{G} = < V, A \cup \{\gamma_1, .., \gamma_l\}, \alpha_0, R \cup \delta >$;
construction of substitution $\delta_S = \{ \gamma \to \overline{\beta} \mid s = u_1 \gamma u_2 \ \& \ y = u_1 \overline{\beta} u_2 \ \& \ \gamma \in \Gamma \}$;
construction of generation $s' \Rightarrow^{*} y$ in CF grammar $\overline{G}$;
construction of substitution $\delta_{S'} = \{ \gamma \to \overline{\beta} \mid s' = u_1' \gamma u_2' \ \& \ y = u_1' \overline{\beta} u_2' \ \& \ \gamma \in \Gamma \}$;
return $(\delta_S \cup \delta_{S'})$;
end;
else **return** $(\{\varnothing\})$;
end.

Obviously, in those cases when $\inf_G \{ s[\delta], s'[\delta] \}$ exists, then grammar $\overline{G}$ is constructed from grammar *G,* and the set of non-terminals in grammar $\overline{G}$ along with all $\alpha \in A$ includes all the variables which present in the suffix of the equation (they become non-terminals of $\overline{G}$, which is acceptable since they enter the equation only once), and the scheme is set-theoretical union of scheme *R* and the suffix. As a result the generation mechanism in CF grammars may be used for constructing substitutions δ_S and $\delta_{S'}$ which are formed by means of including all rules $\gamma \to \beta$ for all the variables (non-terminals) which have place in terms *s* and s', and all the sub- strings β of string *y*, which are derived from these variables (non-terminals) within generations $s \Rightarrow^{*} y$ and $s' \Rightarrow^{*} y$ in them. In the result of algorithm operation one will get substitution which was obtained by

the union of δ_S and $\delta_{S'}$. If $\inf_G\{s[\delta], s'[\delta]\}$ does not exist, then, obviously, the equation has no solution.

Example 7. Apply the algorithm described above to the equation from Example 4:

$$< p_1 : x / y / e_1 = z : t / c_2 / s, \{x \to \langle B \rangle, y \to \langle C \rangle, z \to \langle P \rangle, t \to \langle B \rangle, s \to \langle E \rangle\} > .$$

First construct $y = \inf_G\{s[\delta], s'[\delta]\} = p_1 : \langle B \rangle / c_2 / e_1$, then from generation $p_1 : x / y / e_1 \Rightarrow^* p_1 : \langle B \rangle / c_2 / e_1$ obtain substitution $\delta_s = \{x \to \langle B \rangle, y \to c_2\}$, and from generation $z : t / c_2 / s \Rightarrow^* p_1 : \langle B \rangle / c_2 / e_1$ obtain substitution $\delta_{s'} = \{z \to p_1, t \to \langle B \rangle, s \to e_1\}$.

The result:

$$\delta_S \cup \delta_{S'} = \{x \to \langle B \rangle, y \to c_2, z \to p_1, t \to \langle B \rangle, s \to e_1\}. \blacksquare$$

The following Theorem 1 proves the fact that as a result of algorithm operation given $\inf_G\{s[\delta], s'[\delta]\}$ we obtain the minimal unifier of the equation $< s = s', \delta >$, hereafter denoted by $U[s = s', \delta]$.

Theorem 1.

$$D[s = s', \delta] = \bigcup_{\substack{u_1 \in V(\gamma_1, \delta_s \cup \delta_{s'}) \\ \dots \\ u_l \in V(\gamma_l, \delta_s \cup \delta_{s'})}} \{\{\gamma_1 \to u_1, \dots, \gamma_l \to u_l\}\} \tag{48}$$

Proof. Denote the right-hand side of (48) by $Д$, and set $\delta_s \cup \delta_{s'} = \{\gamma_1 \to \overline{\beta}_1, \dots, \gamma_l \to \overline{\beta}_l\}$ denote by $\overline{\delta}$. Assume, that $D[s = s', \delta] \neq Д$. The assumption is only possible when either $D[s = s', \delta] - Д \neq \{\varnothing\}$, or $Д - D[s = s', \delta] \neq \{\varnothing\}$. The existence of substitution $d \in D[s = s', \delta]$ which is not an element of set $Д$ is equivalent to the existence of word $w = s[d] = s'[d]$ which is not generated from SF $y = \inf_G\{s[\delta], s'[\delta]\}$. However, under Lemma 1 there are no words which belong to the intersection of sets $W_{s[d]}$ and $W_{s'[d]}$, and are not generated from SF *y*. On the other hand, the existence of substitution $d \in Д$ which is not an element of $D[s = s', \delta]$ is equivalent to the existence of word $w = s[d] = s'[d]$ generated from SF $y = \inf_G\{s[\delta], s'[\delta]\}$ and not belonging to the intersection $W_{s[d]}$ and $W_{s'[d]}$ which is again impossible as per Lemma 1. Therefore the assumption about the inequality of sets $D[s = s', \delta]$ and $Д$ is false. $\blacksquare$

There is apparent relation between unifier $U[s = s', \delta] = \{\gamma_1 \to \overline{\beta}_1, \dots, \gamma_l \to \overline{\beta}_l\}$ and the set of solutions $D[s = s', \delta]$ of the equation $< s = s', \delta >$:

$$D[s = s', \delta] = \{\{\gamma_1 \to u_1, \dots, \gamma_l \to u_l\} \mid \overline{\beta}_1 \Rightarrow^* u_1 \,\&\, \dots \,\&\, \overline{\beta}_1 \Rightarrow^* u_l \,\&\, \{u_1, \dots, u_l\} \subset V^*\}. \tag{49}$$

It is noticeable that the key element of the algorithm is the construction of the maximal lower bound of the set consisting of two sentential forms of unambiguous acyclic CF grammar. The corresponding algorithm is presented in the following paragraph.

5. Building the Maximal Lower Bound of the Set of Sentential Forms of Unambiguous Acyclic CF Grammar

Let us have set $\{x,x'\}\subseteq SF(G)$. Practical criteria evaluating existence of $\inf_G\{x,x'\}$ is defined by the following Theorem 2 [11].

Theorem 2. If $G=<V,A,\alpha_0,R>$ is unambiguous acyclic CF grammar, while x and x' are its sentential forms, then $\inf_G\{x,x'\}$ exists only if $y=\sup_G\{x,x'\}$ does not contain non-terminal $\alpha\in A$, such that $y=z\alpha z', z, z'\in(V\cup A)^*$, and set R includes two rules $\alpha\to\beta$ and $\alpha\to\beta'$, such that $z\beta z'\Rightarrow^* x$ and $z\beta' z'\Rightarrow^* y$. ■

The essence of this Theorem is illustrated by

Example 8. Let us have grammar G from Example 4 and sentential forms $x=\langle P\rangle:b_1/\langle C\rangle/e_1$, $x'=\langle P\rangle:b_2/c_1/e_1$. One may easily construct $y=\sup_G\{x,x'\}=\langle P\rangle:\langle B\rangle/\langle C\rangle/e_1$. This SF contains non-terminal $\langle B\rangle$ that results in scheme R including rules $\langle B\rangle\to b_1$ and $\langle B\rangle\to b_2$, so that $y=z\langle B\rangle z'$; $z=\langle P\rangle:$; $z'=/\langle C\rangle/e_1$; so from $zb_1z'\Rightarrow^* x$, $zb_2z'\Rightarrow^* x'$ we know that $\inf_G\{x,x'\}$ does not exist. ■

$\inf_G\{x,x'\}$ constructing logics is defined by recursive function I from three arguments; the first two are sub-strings of strings x and x', while the third one is sub-string of string $\sup_G\{x,x'\}$:

$$I(x_1, x_2, z)=u\cdot I(\overline{\beta}_1, \overline{\beta}_2, \alpha)\cdot I(z_1, z_2, \overline{z}), \tag{50}$$

if $x_1=u\overline{\beta}_1z_1$, $x_2=u\overline{\beta}_2z_2$, $z=u\alpha\overline{z}$, $\alpha\Rightarrow^*\overline{\beta}_1$, $\alpha\Rightarrow^*\overline{\beta}_2$, and

$$I(\Delta,\Delta,\Delta)=\Delta, \tag{51}$$

where $u\in V^*, \{\overline{\beta}_1, z_1, \overline{\beta}_2, z_2, z\}\subseteq(V\cup A)^*, \alpha\in A$, and Δ is an empty string.

Evidently, function I is constructed in the manner implementing synchronous left-to-right parsing of strings x, x' and $\sup_G\{x,x'\}$; during this process the resulting string part accumulated throughout previous steps gradually includes: string u in alphabet V that is prefix of the remaining parts x, x' and $\sup_G\{x,x'\}$; result $I(\overline{\beta},\overline{\beta}',\alpha)$, where $\overline{\beta}$ and $\overline{\beta}'$ are sub-strings of x and x' following u and derived from non-terminal α located directly after u in $\sup_G\{x,x'\}$; and the result of application of the same function I to sub-strings x, x' and $\sup_G\{x,x'\}$ following $\overline{\beta},\overline{\beta}'$ and α accordingly. If all the three arguments I are empty strings, parsing is completed, and the empty string obtained in accordance with (51) concatenates with the accumulated result without changing it and terminates recursion at the same time.

Theorem 3. If $\inf_G\{x,x'\}$ exists, then

$$\inf_G\{x,x'\}=I(x,x',\sup_G\{x,x'\}). \tag{52}$$

Proof. According to the definition of the maximal lower bound, $y=\inf_G\{x,x'\}$ is a sentential form generated in grammar G simultaneously from x and x', and with this

$y \Rightarrow^* \bar{y}$ is attributed to any other SF $\bar{y}$ fulfilling these conditions. Both implies from (50) where $I(\bar{\beta}_1, \bar{\beta}_2, \alpha)$ plays the key role. The definition of the minimal upper bound results in $\alpha \Rightarrow^* \bar{\beta}_1$, $\alpha \Rightarrow^* \bar{\beta}_2$, therefore, $\bar{\beta}_1 \neq \bar{\beta}_2$ (otherwise $\bar{\beta}_1$ and $\bar{\beta}_2$ would have been taken up by $\sup_G \{x, x'\}$). Pursuant to theorem 2 and assumption on the existence of $\inf_G \{x, x'\}$, only two variants are possible in which $\bar{\beta}_1$ and $\bar{\beta}_2$ are unequal — $\bar{\beta}_1 = \alpha, \bar{\beta}_2 \neq \alpha$ and $\bar{\beta}_2 = \alpha, \bar{\beta}_1 \neq \alpha$. In both cases the resulting string includes string $\inf_G \{\bar{\beta}_1, \bar{\beta}_2\}$, generated from non-terminal α, or, in other words, not equal to this non-terminal. Recursive application of (50) results in $I(x, x', \sup_G \{x, x'\})$ being exactly $\inf_G \{x, x'\}$. ■

$\inf_G \{x, x'\}$ construction logics and its relations with $\sup_G \{x, x'\}$ are shown on Fig.1 and in Example 9.

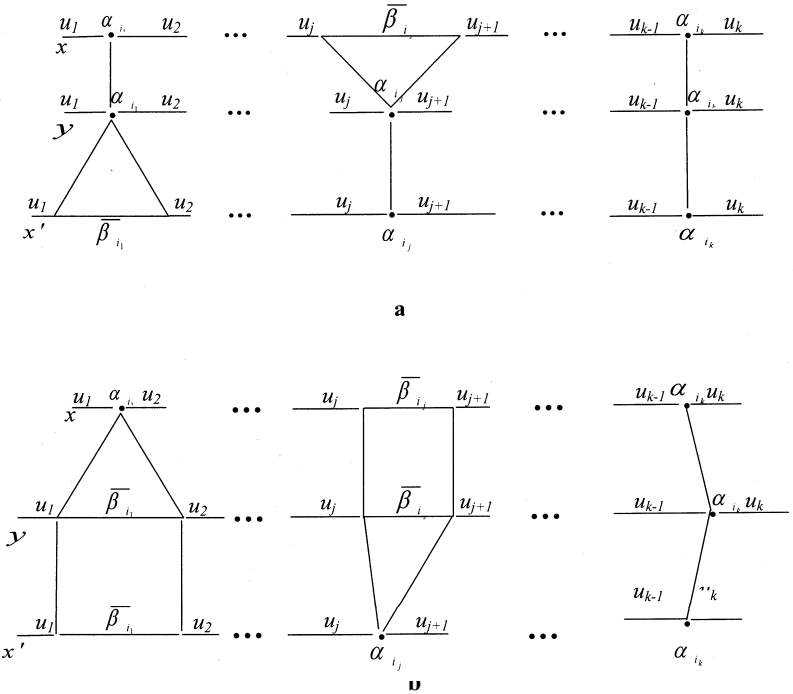


Fig.1. Relations between x, x', $\sup_G \{x, x'\}$ and $\inf_G \{x, x'\}$: a) $y = \sup_G \{x, x'\}$; b) $y = \inf_G \{x, x'\}$.

Fig.1 shows that non-terminal α_{i_1} presenting in $\sup_G\{x, x'\}$ while constructing $\inf_G\{x, x'\}$ is replaced by string $\overline{\beta}_{i_1}$ derived from α_{i_1} in the derivation $\sup_G\{x, x'\} \Rightarrow^* x'$; non-terminal α_{i_j}, similarly, is replaced by string $\overline{\beta}_{i_j}$ derived from it in the derivation $\sup_G\{x, x'\} \Rightarrow^* x'$;, and, finally, non-terminal α_{i_k} is added to $\inf_G\{x, x'\}$ as is. Thus, x' is more informative than x and $\sup_G\{x, x'\}$ "in the sense" α_{i_1}; x is more informative than $\sup_G\{x, x'\}$ and x' "in the sense" α_{i_j}; while as per α_{i_k} x, x' and $\sup_G\{x, x'\}$ are equally informative.

Example 9. Assuming we have grammar G from Example 4 and sentential forms $x = p_1 : b_1 / \langle C \rangle / \langle E \rangle, x' = \langle P \rangle : b_1 / c_2 / \langle E \rangle$. While

$$\sup_G\{x, x'\} = \langle P \rangle : b_1 / \langle C \rangle / \langle E \rangle. \tag{53}$$

In accordance with definition of function *I*,

$$\begin{aligned}
&\inf_G\{x, x'\} = \Delta \cdot I(p_1, \langle P \rangle, \langle P \rangle) \cdot I(: b_1 / \langle C \rangle / \langle E \rangle, : b_1 / c_2 / \langle E \rangle),\\
&I(p_1, \langle P \rangle, \langle P \rangle) = p_1,\\
&I(: b_1 / \langle C \rangle / \langle E \rangle, : b_1 / c_2 / \langle E \rangle) =\\
&: b_1 / \cdot I(\langle C \rangle, c_2, \langle C \rangle) \cdot I(/\langle E \rangle), / \langle E \rangle, / \langle E \rangle),\\
&I(\langle C \rangle, c_2, \langle C \rangle) = c_2,\\
&I(/\langle E \rangle, / \langle E \rangle, / \langle E \rangle) = / \cdot I(\langle E \rangle, \langle E \rangle, \langle E \rangle) \cdot I(\Delta, \Delta, \Delta),\\
&I(\langle C \rangle, \langle C \rangle, \langle C \rangle) = \langle C \rangle,\\
&I(\Delta, \Delta, \Delta) = \Delta.
\end{aligned} \tag{54}$$

So,

$$\inf_G\{x, x'\} = p_1 : / b_1 / c_2 / \langle E \rangle. \tag{55}$$

Here x is more informative than x' as per non-terminal $\langle P \rangle$, x' is more informative than x as per non-terminal $\langle C \rangle$, and as per non-terminal $\langle E \rangle$ both the SFs are equally informative. ■

The only uncovered entry of the offered algorithmic complex is constructing of $\sup_G\{x, x'\}$. This entry is defined by recursive function S from three arguments being sentential forms of unambiguous acyclic CF grammar $G = \langle V, A, \alpha_0, R \rangle$:

$$S(x_1, x_2, z) = \begin{cases} S(x_1, x_2, z_1 \beta z_2), \ \textit{if} \ (\exists \alpha \in A)\, z = z_1 \alpha z_2 \ \& \ (\exists \alpha \to \beta \in R) \\ \qquad\qquad z_1 \beta z_2 \Rightarrow^* x_1 \ \& \ z_1 \beta z_2 \Rightarrow^* x_2 \\ z \qquad\qquad \textit{otherwise..} \end{cases} \tag{56}$$

It is evident that generation of SF that generates both x_1 and x_2 is implemented at every recursion step as long as it is possible through application of some rule $\alpha \to \beta$ to non-terminal α having place in the third argument. The SF that breaks the continuity of such generation is $\sup_G\{x, x'\}$. It is clear that the order of selection of the next non-terminal for generator is non-essential.

Theorem 4. For sentential forms x and x' of unambiguous acyclic CF grammar $G=<V, A, \alpha_0, R>$

$$\sup_G \{x, x'\} = S(x, x', \alpha_0). \tag{57}$$

Proof is evident. ■

Corollary 1. Operations $\inf_G$ and $\sup_G$ are commutative and associative. ■

Example 10. As applied to grammar G from Example 4 and objects x and x' from Example 9

$$\sup_G\{x, x'\} = S(x, x', \langle atom\rangle) = S(x, x', \langle P\rangle : \langle B\rangle / \langle C\rangle / \langle E\rangle) =$$
$$= S(x, x', \langle P\rangle : b_1 / \langle C\rangle / \langle E\rangle) = \langle P\rangle : b_1 / \langle C\rangle / \langle E\rangle.$$ ■

In general the logics underlying the analyzed algorithmic complex can be illustrated in the manner shown on Fig.2.

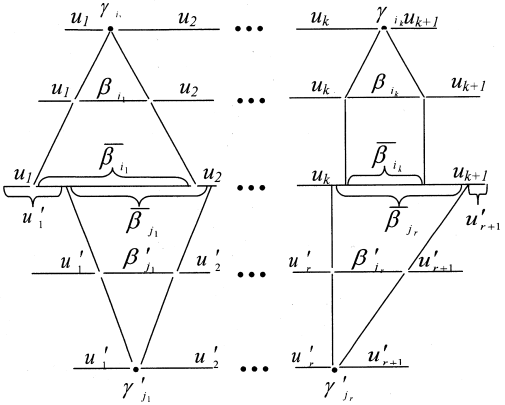


Fig.2. Relations between elements of equation $<s = s', \delta>$ and its unifier ($\delta = \{\gamma_1 \to \beta_1, ..., \gamma_l \to \beta_l\}$, $\overline{\delta} = \{\gamma_1 \to \overline{\beta}_1, ..., \gamma_l \to \overline{\beta}_l\}$)

Note that algorithmics of solution of "classic" word equations [1, 2, 4] is similar to the described one. It is based on left-side parsing of terms s and s' in the process of which their single-symbol prefixes are compared, and depending on the variant ($<v,v'>$, $<v,\gamma'>$, $<\gamma,v'>$ or $<\gamma,\gamma'>$) either termination is executed (at $v \neq v'$) or one or two residual equations are remained (the latter variant corresponds to the one when both the terms start with variables, i.e. $\langle\gamma,\gamma'\rangle$, which is processed by two alternatives $\gamma' = \gamma\gamma''$ and $\gamma = \gamma'\gamma''$ with introduction of additional variable γ''). As a result there is a tree with initial equation $s = s'$ at its root and residual equations at its other nodes. Herewith, each path from its root to the terminal node (leave) where generation of a new equation is impossible corresponds to a term similar to $\inf_G\{s[\delta], s[\delta']\}$ in its sense.

A deeper analysis of relations between word equations on CF languages and "classic" word equations (i.e., generally speaking, on language V^*, where sets of values of variables in these equations also are V^*) is beyond the scopes of this article.

With this we finish our analysis of mathematical apparatus of solution of word equations on CF languages for the cases of the variables single instances to equation and unambiguous acyclic CF grammars and proceed to examination of possibilities of generalization of the obtained results on general cases.

6. Constructing the Minimal Unifier of Equation with Multiple Instances of Variables

Assume we have equation $< s = s', \delta >$ with unlimited number of instances of variables in term $s's$.

It is essential that when some variable γ enters term $s's$ more than once, but $\gamma \to \beta \in \delta$ is such that $\beta \in V^*$, then multiple instance of γ is equivalent to its absence, and initial equation $< s = s', \delta >$ is equivalent to equation $< s[\{\gamma \to \beta\}] = s'[\{\gamma \to \beta\}], \delta >$. Such variables are “hidden” constants in their essence.

Having generalized this observation, we can exclude from consideration such multiple instances of some variable γ, for which $\gamma \to \beta \in \delta$ and set $\mathrm{V}(\gamma, \delta)$ is finite. If initial equation $< s = s', \delta >$ has such variable, the latter is equivalent to $r = |V(\gamma, \delta)|$ of equations

$$\begin{gathered} < s[\{\gamma \to u_1\}] = s'[\{\gamma \to u_1\}], \delta - \{\gamma \to \beta\} >, \\ \ldots \\ < s[\{\gamma \to u_r\}] = s'[\{\gamma \to u_r\}], \delta - \{\gamma \to \beta\} >, \end{gathered} \tag{58}$$

with $\mathrm{V}(\gamma, \delta) = \{u_1, .., u_r\}$. It is evident that none of these equations contains variable γ in terms s and s'. If initial equation $< s = s', \delta >$ contains q similar variables $\gamma_{i_1}, ..., \gamma_{i_q}$, it is equivalent to

$$\prod_{j=1}^{q} |\mathrm{V}(\gamma_{i_j}, \delta)| \tag{59}$$

of equations of type

$$< s[\gamma_{i_1} \to u_1, .., \gamma_{i_q} \to u_q] = s'[\gamma_{i_1} \to u_1, .., \gamma_{i_q} \to u_q], \bar{\delta} >, \tag{60}$$

with

$$\begin{gathered} \bar{\delta} = \delta - \{\gamma_{i_1} \to \beta_{i_q}, ..., \gamma_{i_q} \to \beta_{i_q}\}, \\ u_1 \in \mathrm{V}(\gamma_{i_1}, \delta), \\ \ldots \\ u_q \in \mathrm{V}(\gamma_{i_q}, \delta). \end{gathered} \tag{61}$$

From now on we will consider only such equations with multiple instances of variables where all variables corresponding to “hidden” constants were pre-excluded using

the described techniques, which means that for each of the remaining variables γ in term $s's$ at least one set $\mathrm{V}(\gamma,\delta)$ is infinite.

Let us assign a new unique variable to each instance of variable $\gamma_i \in \{\gamma_1,...,\gamma_l\}$, which will result in term $s's$ having k_i of unique variables of type $\gamma_i^{(1)},...,\gamma_i^{(k_i)}$ (superscript denotes the instance number), with each of this variables presents in this term (i.e. in the equation) once. We will execute such operation with regard to all variables included into term $s's$ several times, thus obtaining a new equation

$$< s_* = s'_*, \delta_* >, \tag{62}$$

having

$$\sum_{i=1}^{l} k_i \tag{63}$$

of variables with single instance (if variable γ_i is included into $s's$ once then $k_i = 1$), while suffix δ_* instead of one element of form $\gamma_i \to \beta_i$ includes k_i of elements of form $\gamma_i^{(1)} \to \beta_i,..., \gamma_i^{(k_i)} \to \beta_i, i = 1,..., l.$

We will assign the algorithm described in paragraphs 4-5 to equation (62) and construct unifier

$$\overline{\delta}_* = \bigcup_{i=1}^{l} \bigcup_{j=1}^{k_i} \{\gamma_i^{(j)} \to \overline{\beta}_i^{(j)}\} \tag{64}$$

The question arises how these unique variables can be converted back to their "cloned" originals, i.e. multiple instances. This problem is resolved by Lemma 3 obtained by deepening of Lemma 2 and given without evident proof.

Lemma 3. Assuming that x is non-empty string in alphabet $V \cup A$, and

$$SF_x(G) = \{x' \mid x \Rightarrow^* x'\} \tag{65}$$

is the set of all strings in alphabet $V \cup A$ generated from string x in unambiguous acyclic grammar $G = < V, A, \alpha_0, R >$. Then set $SF_x(G)$ is partially ordered in relation to $\Rightarrow^*$, and $\sup_G SF_x(G) = x$. ■

It is clear that the set of solutions of initial equation $D[s = s', \delta]$ can include only such substitutions that in connection with to all instances $\gamma_i^{(1)},..., \gamma_i^{(k_i)}$ of variable γ_i contain rules with the same right part:

$$\gamma_i^{(1)} \to u,..., \gamma_i^{(k_i)} \to u, \tag{66}$$

with $u \in V^*$. Therefore,

$$\overline{\delta} = \bigcup_{i=1}^{l} \{\gamma_i \to \overline{\beta}_i \mid \overline{\beta}_i = \inf_G \{\overline{\beta}_i^{(1)},..., \overline{\beta}_i^{(k_i)}\}\}, \tag{67}$$

where the maximal lower bound is related to the set of $SF_{\beta_i}(G)$ objects derived in grammar G from β_i. If some i has no $\inf_G \{\overline{\beta}_i^{(1)},..., \overline{\beta}_i^{(k_i)}\}$, the initial equation has no solutions.

The described algorithm is illustrated by the example below.

Example 11. Let us consider we have grammar G from Example 4 with the scheme augmented by rule

$$\langle B\rangle \to \langle C\rangle, \tag{68}$$

that enables instance of strings of type $p:c/c/e$ along with strings of type $p:b/c/e$ into language $L(G)$, which, in its turn, is equivalent to

$$P_i \subseteq (B \cup C)\times C \times E. \tag{69}$$

Equation

$$< p_1 : x/c_2/e_1 = z : y/x/s,\ \{x \to \langle B\rangle, z \to \langle P\rangle, y \to \langle P\rangle, s \to \langle E\rangle\} > \tag{70}$$

after "cloning" of variable x looks like that –

$$< p_1 : x^{(1)}/c_2/e_1 = z : y/x^{(2)}/s, \{x^{(1)} \to \langle B\rangle, x^{(2)} \to \langle B\rangle, z \to \langle P\rangle, y \to \langle B\rangle\} >. \tag{71}$$

Its unifier is

$$\bar{\delta}_* = \{x^{(1)} \to \langle B\rangle, z \to p_1, y \to \langle B\rangle, x^{(2)} \to c_2, s \to e_1\} \tag{72}$$

Consequently, unifier of the initial equation is

$$\bar{\delta} = \{x \to c_2, z \to p_1, y \to \langle B\rangle, s \to e_1\}, \tag{73}$$

due to $\inf_G \{\langle B\rangle, c_2\} = c_2$, as $\langle B\rangle \Rightarrow^* c_2$.

Supposing equation (70) was related to the language generated by the initial grammar from Example 4, it would have no solutions, because in this case $\inf_G \{\langle B\rangle, c_2\}$ would not exist (non-terminal $\langle B\rangle$ generates strings $b_1, ..., b_n$, but not c_2). ■

At the first glance it looks like the end, but the issue is much deeper, because due to the semantics of variables (substitution of all instances of one variable with one and the same string) $W_{s[\delta]}$ and $W_{s'[\delta]}$ are subsets of sets $W_{s_*[\delta_*]}$ and $W_{s'_*[\delta_*]}$, therefore:

$$\begin{aligned} W_{s[\delta]} &\subseteq W_{s_*[\delta_*]} \\ W_{s'[\delta]} &\subseteq W_{s'_*[\delta_*]} \end{aligned}, \tag{74}$$

which is illustrated by the following example.

Example 12. Let us consider grammar G from Example 11 and equation

$$< p_1 : x/x/e_1 = z : y/x/s,\ \{x \to \langle B\rangle, z \to \langle P\rangle, y \to \langle B\rangle, s \to \langle E\rangle\} >, \tag{75}$$

with the only difference from equation (70) from the previous Example 11 being that the left part of its kernel contains two instances of variable x. One can easily see that

$$W_{s[\delta]} = \{p_1 : c/c/e_1 \mid c \in C\}, \tag{76}$$

$$W_{s_*[\delta_*]} = \{p_1 : h/c/e_1 \mid h \in B \cup C \,\&\, c \in C\}, \tag{77}$$

i.e. in (77) any combinations of h and c are permitted, while (76) permits only the combinations containing $h = c$. This confirms (74), and besides,

$$|W_{s[\delta]}| = |C|, \tag{78}$$

$$|W_{s_*[\delta_*]}| = |B \cup C| \times |C|. \blacksquare \tag{79}$$

The discovered fact means that in general even if some i from (68) contains $\inf_G\{\overline{\beta}_i^{(1)},...,\overline{\beta}_i^{(k_i)}\}$, then considering (74) it does not mean that

$$\bigcap_{j=1}^{k_i} W_{\overline{\beta}_i^{(j)}} \neq \{\varnothing\}, \tag{80}$$

i.e. that the unifier of initial equation $< s = s', \delta >$ exists. After all

$$W_{s_*[\overline{\delta}]} \cap W_{s'_*[\overline{\delta}]} \neq \{\varnothing\} \tag{81}$$

does not mean that this is true for their subsets, and in general it is possible that

$$W_{s[\overline{\delta}]} \cap W_{s'[\overline{\delta}]} = \{\varnothing\}. \tag{82}$$

To exclude this possibility, the solution of the initial equation, but with new suffix $\overline{\delta}$, is apparently required:

$$< s = s', \overline{\delta} >. \tag{83}$$

If this equation has unifier $\overline{\delta}$, i.e.

$$U[s = s', \overline{\delta}] = \overline{\delta}, \tag{84}$$

then $\overline{\delta}$ is indeed the unifier for initial equation $< s = s', \delta >$. If it is not so, i.e. equation (83) has unifier $\overline{\delta}^{(1)} \neq \overline{\delta}$, then we obtain the following equation as a result

$$< s = s', \overline{\delta}^{(1)} >. \tag{85}$$

The equation requires new solution with the purpose of constructing unifier $\overline{\delta}^{(2)}$ that is to be checked whether it contains $\overline{\delta}^{(1)} = \overline{\delta}^{(2)}$, i.e.

$$U[s = s', \overline{\delta}^{(1)}] = \overline{\delta}^{(1)}, \tag{86}$$

and execute the described actions up to i, so that

$$U[s = s', \overline{\delta}^{(i)}] = \overline{\delta}^{(i)}. \tag{87}$$

In [11] such iterative process is called “variables description clarification”. The answer to the question whether this process can be terminated, which means whether word equations with multiple instances of variables on languages generated by unambiguous acyclic CF grammars can be algorithmically solved, would be negative. Considering crucial importance of this question we will give a corresponding Theorem 5 with substantiation that is quite specific in illustrating the “nature” of the process under analysis.

Theorem 5 [11]. There exists word equation in language $L(G)$, where G is unambiguous acyclic CF grammar, with multiple instances of variables,

$$< s = s', \delta >, \tag{88}$$

that with any i

$$U[s = s', \overline{\delta}^{(i)}] \neq \overline{\delta}^{(i)}. \tag{89}$$

Proof. Assuming we have the following CF grammar that describes a set of atomary predicate logics language formulas with the single literal P — binary relation symbol — and two functors F and H — unary function symbols:

$$
\begin{aligned}
&\langle atom\rangle \to P(\langle term\rangle,\langle term\rangle),\\
&\langle term\rangle \to \langle functor\rangle(\langle term\rangle),\\
&\langle functor\rangle \to F,\\
&\langle functor\rangle \to H,\\
&\langle term\rangle \to \langle functor\rangle(\langle term\rangle),\\
&\langle term\rangle \to \langle \text{constant}\rangle,\\
&\langle constant\rangle \to A,\\
&\quad ...\\
&\langle constant\rangle \to Z.
\end{aligned}
\tag{90}
$$

Let us consider equation

$$< P(H(x),x) = P(y,F(y)),\ \{x \to \langle term\rangle, y \to \langle term\rangle\} >, \tag{91}$$

that includes each of variables *x* and *y* twice.

Having executed all the steps described above we will obtain (non-terminal $\langle term\rangle$ for short is replaced by symbol *"T"*):

(step 1)

$$
\begin{aligned}
&s[\delta] = P(H(T),T),\\
&s'[\delta] = P(T,F(T)),\\
&\inf_G\{s[\delta],s'[\delta]\} = P(H(T),F(T)),\\
&\bar{\delta}_* = \{x^{(1)} \to T, x^{(2)} \to F(T), y^{(1)} \to H(T), y^{(2)} \to T\},\\
&\bar{\delta} = \{x \to \inf_G\{F(T),T\}, y \to \inf_G\{T,F(T)\}\} = \{x \to F(T), y \to H(T)\},
\end{aligned}
\tag{92}
$$

(step 2)

$$
\begin{aligned}
&s[\bar{\delta}] = P(H(F(T)),F(T)),\\
&s'[\bar{\delta}] = P(H(T),F(H(T))),\\
&\inf_G\{s[\bar{\delta}],s'[\bar{\delta}]\} = P(H(F(T)),F(H(T))),\\
&\bar{\delta}^{(1)}_* = \{x^{(1)} \to F(T), x^{(2)} \to F(H(T)), y^{(1)} \to H(T), y^{(2)} \to H(F(T))\},\\
&\bar{\delta}^{(1)} = \{x \to \inf_G\{F(T),F(H(T))\}, y \to \inf_G\{H(T),H(F(T))\}\} =\\
&= \{x \to F(H(T)), y \to H(F(T))\},\\
&\qquad\qquad \dots
\end{aligned}
\tag{93}
$$

(step *i*+1)

$$\inf_G\{s[\bar{\delta}^{(i)}],s'[\bar{\delta}^{(i)}]\} = P(\underbrace{H(F(...(H(F}_{i+1\ functors}(T)))...)), \underbrace{F(H(...(F(H}_{i+1\ functors}(T)))...))) \tag{94}$$

and, consequently,

$$\bar{\delta}^{(i+1)} = \{x \to F(H(...F(H(F(T)...)),\ y \to H(F(...H(F(H(T)...)\} \neq \bar{\delta}^{(i)},$$

where right part of each rules also has $i+1$ of reciprocate functors and starts with *F* and *H*. Therefore, induction step shows that the described iteration process of equation (91) solution is infinite, *quod erat demonstrandum*. ■

The given example has both auxiliary and individual significance.

Corollary 2. Robinson's unification algorithm with multiple instances of variable in being unificated atomic formulas is incorrect in general. ■

Indeed, Robinson's unification algorithm as it is described in the original [7] and posterior applied works like [8,9] is an intuitive heuristic procedure which enables the presence of similar variables in unified atoms, but processing of this situation is limited by one step of the iteration process described above that in general may lead to improper conclusion (i.e. resolution of irresolvable formulas through contrary atoms of form $P(...)$ and $\neg P(...)$ with multiple inputs of instances). In particular, having applied Robinson's algorithm to a pair of contrary atoms $< P(H(x),x), \neg P(y,F(y)) >$, with the pair corresponding to equation (91), we will obtain unifier $U = \{H(x)/y, x/F(y)\}$ that, at the first glance, indeed unifies it by transforming it into $< P(y,x), \neg P(y,x) >$ by "reciprocal" substitution of terms with variables. But having executed "direct" substitution of variables with terms we will obtain pairs $< P(H(F(y)),F(y)), \neg P(y,F(y)) >$ and $< P(H(x),x), \neg P(H(x),F(H(y))) >$, none of which can be applied to resolution. While the whole essence of unification is acquisition of variables substitution which are to be applied in the consequent conclusion exactly in the variant of "direct" substitution.

Corollary 3. Resolution procedure with multiple instances of variables into atomic formulas is incorrect in general. ■

The wide look on the Robinson's unification made by F.Baader in his article "Unification Theory" [1] and other authors of this workshop, of course, make the "naive" unification essentially more strict, but like earlier works they operate by "mappings" from one atom to another (however complicating this basis by various associated identities forming so-called equational theories). Our approach is based on the quite different formalization of unificated objects and unification itself. So the notions "incorrect algorithm" and "incorrect procedure" used higher must be interpreted only in the sense of the mentioned formalization.

Note that while proving Theorem 5 we significantly used multiple instances of a variable into one of the terms of equation (88) kernel. Let us now analyze a case when variables are included into any of terms s and s' once, and several times into the equation. One may easily see that each variable can be included into equation not more than twice — once into each of the terms. Such equations where terms s and s' contain only single instances of variables hereinafter referred to as linear equations, as opposed to non-linear equations that allow for multiple instances of variables into the given terms.

Theorem 6. The problem of resolution of linear word equation on language *L(G)*, where *G* is unambiguous acyclic CF grammar, is algorithmically solvable.

Proof. As each variable is included into term s once, it is semantically equivalent to non-terminal, so

$$W_{s[\delta]} = W_{s_*[\delta_*]}. \tag{95}$$

This fact also points out to

$$W_{s'[\delta]} = W_{s'_*[\delta_*]}. \tag{96}$$

Thus,

$$W_{s[\delta]} \cap W_{s'[\delta]} = W_{s_*[\delta_*]} \cap W_{s'_*[\delta_*]}, \tag{97}$$

which makes algorithm of constructing of equation $\langle s = s', \delta \rangle$ unifier with multiple instances of variables described at the beginning of this Paragraph 6 and demonstrated by Example 11 correct in this case. ■

7. Assessment of Constructing the Minimal Unifiers of Linear Equation on Languages Defined by the Means of Cyclic and Ambiguous CF Grammars

We remind you that cyclic grammar allows for $x \Rightarrow^+ x$ derivation. Suppose that CF grammar G is not reducing, i.e. it does not contain rules of the form $\alpha \to \Delta$, where Δ is the empty string (otherwise G transforms into grammar G' which does not contain these rules and is equivalent to G in the sense that $L(G') = L(G)$ [10]). Furthermore $x \Rightarrow^+ x$ is possible only with at least one non-terminal α, for which $\alpha \Rightarrow^+ \alpha$, so it enables the cycling of recursive function *S* denoted by formula (56). In fact, having construct on some step of recursion object *z* which contains non-terminal α, we will continue deriving infinite set of steps, since there is always some subsequent non-terminal for its continuation from the chain $\alpha \Rightarrow \alpha^{(1)} \Rightarrow ... \Rightarrow \alpha^{(k)} \Rightarrow \alpha$ which corresponds to $\{\alpha \to \alpha^{(1)}, ..., \alpha^{(k-1)} \to \alpha^{(k)}, \alpha^{(k)} \to \alpha\} \subseteq R$. Here all the objects obtained in this way ($z_1 \alpha z_2, z_1 \alpha^{(1)} z_2, ..., z_1 \alpha^{(k-1)} z_2, z_1 \alpha^{(k)} z_2$) will be formally the minimal upper bounds of set $\{x, x'\}$. One may easily verify that this situation results from the fact that the set of SFs of unambiguous cyclic CF grammar does not take the form of partially ordered set – it enables generations of $\alpha \Rightarrow^+ \alpha'$ and $\alpha' \Rightarrow^+ \alpha$ (using the language of numerical mathematics, $a > a'$ and $a' > a$ simultaneously). Therefore, we shall be restricted by statement that the suggested apparatus itself is not applicable to unambiguous cyclic CF grammars, although in terms of "informational" point of view the problem does not have any signs of fatality (it is easy to reconstruct function *S* by "curing" it from cycling).

Regarding acyclic CF grammars in general, including ambiguous CF grammars, the following theorem exists.

Theorem 7 [11]. The problem of assessing the existence of at least one lower bound of set $\{x, y\}$, where *x* and *y* are sentential forms in acyclic CF grammar, in general does not have an algorithmic solution.

Proof of the theorem [11] is based on the fact that in case of resolvability of the problem of assessing the existence of $\inf_G\{x, y\}$, the problem of defining at least one word $w \in L(G)$ which has two different canonical derivations in grammar G would also have a solution. It, in its turn, is equivalent to resolvability of the problem of clarifying CF grammar unambiguous properties which does not have an algorithmic solution [10]. ■

Since the algorithm for constructing $\inf_G\{s[\delta], s'[\delta]\}$ serves as the basis for all discussed algorithms for constructing various equation unifiers, the obtained result implies the following corollary.

Corollary 4. The problem of constructing the linear word equation unifier $< s = s', \delta >$ in the language defined by the means of acyclic CF grammar in general does not have an algorithmic solution. ■

Thus, the application area of the designed apparatus is restricted to linear word equations on languages generated by unambiguous acyclic CF grammars; however this restriction is practically insignificant.

8. Linear Equation Systems

By linear word equation system on language $L(G)$ generated by unambiguous acyclic CF grammar G we shall mean the following structure

$$< s_1 = s_1', ..., s_m = s_m', \delta >, \tag{98}$$

where $m \geq 2$, $s_1, s_1', .., s_m, s_m', \delta$ and equality sign denote the same as in (14); δ is called the suffix as before and is preceded by equation set called kernel. Only correct equation systems will be considered, in which

$$\{s_1[\delta], s_1'[\delta], ..., s_m[\delta], s_m'[\delta]\} \subseteq L(G). \tag{99}$$

Term substitution d with respect to terms $s_1, s_1', .., s_m, s_m'$ is the solution to correct equation system (98); it transforms equations $s_1 = s_1', .., s_m = s_m'$ into identities:

$$\begin{gathered} s_1[d] \equiv s_1'[d], \\ ... \\ s_m[d] \equiv s_m'[d]. \end{gathered} \tag{100}$$

Correspondingly, the set of solutions to equation system (98), in which

$$\delta = \{\gamma_1 \to \beta_1, ..., \gamma_l \to \beta_l\}, \tag{101}$$

is the following set

$$D[s_1 = s_1', ..., s_m = s_m', \delta] = \{d \mid d \in \sum\nolimits_{\delta} \& \, s_1[d] \equiv s_1'[d] \& ... \& \, s_m[d] \equiv s_m'[d]\}, \tag{102}$$

where $\sum_{\delta}$ is defined by equation (16).

By analogy with (37)-(38), the finite representation (unifier) of the set of the solutions to equation system (98) is the set

$$\overline{\delta} = U[s_1 = s_1', ..., s_m = s_m', \delta] = \{\gamma_1 \to \overline{\beta}_1, ..., \gamma_l \to \overline{\beta}_l\} \tag{103}$$

such that

$$\begin{gathered} s_1[\overline{\delta}] \equiv s_1'[\overline{\delta}], \\ ... \\ s_m[\overline{\delta}] \equiv s_m'[\overline{\delta}]. \end{gathered} \tag{104}$$

Furthermore,

$$D[s_1 = s_1', ..., s_m = s_m', \delta] = \{d \mid d \in \sum\nolimits_{\overline{\delta}} \}. \tag{105}$$

Equation system with at least one solution, i.e. such that

$$D[s_1 = s_1', ..., s_m = s_m', \delta] \neq \{\varnothing\}, \tag{106}$$

is called compatible, and the equation system with no solution, for which

$$D[s_1 = s_1',...,s_m = s_m', \delta] = \{\varnothing\}, \tag{107}$$

is called incompatible.

Assuming all equations which are included in kernel of the system (98) are linear, a rather evident algorithm for constructing its unifier will be considered.

The algorithm is represented by cycle with body involving two steps. During the initial step m equations which form the system are solved independently of one another; it results in formation of m unifiers of these equations:

$$\begin{gathered} U[s_1 = s_1', \delta] = \{\gamma_1 \to \overline{\beta}_1^1,..., \gamma_l \to \overline{\beta}_l^1\}, \\ ... \\ U[s_i = s_i', \delta] = \{\gamma_1 \to \overline{\beta}_1^i,..., \gamma_l \to \overline{\beta}_l^i\}, \\ ... \\ U[s_m = s_m', \delta] = \{\gamma_1 \to \overline{\beta}_1^m,..., \gamma_l \to \overline{\beta}_l^m\}. \end{gathered} \tag{108}$$

The unifier of equation system (98) is built during the second step:

$$\overline{\delta} = \{\gamma_1 \to \overline{\beta}_1,..., \gamma_l \to \overline{\beta}_l\}, \tag{109}$$

where

$$\begin{gathered} \overline{\beta}_1 = \inf_G\{\overline{\beta}_1^1,..,\overline{\beta}_1^m\}, \\ ... \\ \overline{\beta}_l = \inf_G\{\overline{\beta}_l^1,..,\overline{\beta}_l^m\}. \end{gathered} \tag{110}$$

If at least one of m equation unifiers in (108) or one of l maximal lower bounds in (110) does not exist, the system is incompatible. If all the enumerated objects exist, recycling applies to system

$$< s_1 = s_1',...,s_m = s_m', \overline{\delta} >. \tag{111}$$

Algorithm stops, if *j*-th recycling results in

$$U[s_1 = s_1',...,s_m = s_m', \overline{\delta}^{(j)}] = \overline{\delta}^{(j)}, \tag{112}$$

thus, $\overline{\delta}^{(j)}$ is the minimal unifier.

Example 13. Assume grammar *G* from Example 1, and the system consisting of two equations

$$\begin{gathered} < p_1 : x / y / e_1 = z : y / c_2 / s, \\ z : b_2 / c_2 / s = p_1 : x / y / t, \\ \{x \to \langle B \rangle, y \to \langle C \rangle, z \to \langle P \rangle, s \to \langle E \rangle, t \to \langle E \rangle\} >. \end{gathered} \tag{113}$$

From (108), the following is obtained:

$$\begin{gathered} U[s_1 = s_1', \delta] = \{x \to \langle B \rangle, y \to c_2, z \to p_1, s \to e_1, t \to \langle E \rangle\}, \\ U[s_1 = s_2', \delta] = \{x \to b_2, y \to c_2, z \to p_1, s \to \langle E \rangle, t \to \langle E \rangle\}. \end{gathered} \tag{114}$$

Then, pursuant to (110),

$$\begin{aligned}
&\inf_G\{\langle B\rangle, b_2\} = b_2,\\
&\inf_G\{c_2, c_2\} = c_2,\\
&\inf_G\{p_1, p_1\} = p_1,\\
&\inf_G\{e_1, \langle E\rangle\} = e_1,\\
&\inf_G\{\langle E\rangle, \langle E\rangle\} = \langle E\rangle.
\end{aligned} \tag{115}$$

Repeating the cycle in respect to

$$\bar{\delta}^{(1)} = \{x \to b_2,\ y \to c_2,\ z \to p_1,\ s \to e_1,\ t \to \langle E\rangle\}. \tag{116}$$

the following is obtained

$$\begin{aligned}
&U[s_1 = s_1', \bar{\delta}] = \bar{\delta},\\
&U[s_2 = s_2', \bar{\delta}] = \{x \to b_2,\ y \to c_2,\ z \to p_1,\ s \to e_1,\ t \to e_1\},
\end{aligned} \tag{117}$$

thereafter

$$\bar{\delta}^{(2)} = \{x \to b_2,\ y \to c_2,\ z \to p_1,\ s \to e_1,\ t \to e_1\}. \tag{118}$$

Here cycle repeating makes no sense, since all basic substitutions are terms, so that $\bar{\delta}^{(2)}$ is the minimal unifier of equation system (113). ■

Since all equations forming the system are linear, the algorithm for its solution always stops in a finite set of steps.

There is another algorithm for solution of linear equation system. It is of the same "outer shell" (cycle before the fulfillment of condition (112)). However cycle body contributes to successive solution of these equations so that for the suffix the next *i*-th equation has unifier constructed from solution of (*i*-1) equation:

$$\begin{aligned}
&\bar{\delta}_1 = U[s_1 = s_1', \bar{\delta}_0],\\
&\bar{\delta}_2 = U[s_2 = s_2', \bar{\delta}_1],\\
&\qquad\cdots\\
&\bar{\delta}_i = U[s_i = s_i', \bar{\delta}_{i-1}],\\
&\qquad\cdots\\
&\bar{\delta}_m = U[s_m = s_m', \bar{\delta}_{m-1}].
\end{aligned} \tag{119}$$

During the first repetition of outer cycle $\bar{\delta}_0$ there is the suffix of solvable equation system δ, during next repetitions it is unifier $\bar{\delta}_m$ obtained from the preceding one.

If while solving the system the construction of unifier of the next *i*-th equation $< s_i = s_i', \bar{\delta}_{i-1} >$ is impossible, the system is incompatible. If, during some repetition of the cycle, $\bar{\delta}_m = \bar{\delta}_0$, then

$$\bar{\delta}_m = \bar{\delta}_0 = U[s_1 = s_1', ..., s_m = s_m', \delta]. \tag{120}$$

Equation solving procedure is of immediate interest. It follows from the subsequent Theorem 8 presented without proof, that all $m!$ variants of ordering the equations, forming kernel of the system (98) are equivalent in the sense that if $U[s_1 = s_1', ..., s_m = s_m', \delta]$ exists, it will be built in all variants. If the system is incompatible, then similarly this fact will also be traced in all variants.

Theorem 8. If linear equation system $< s_1 = s'_1,...,s_m = s'_m, \delta >$ is compatible, then

$$U[s_1 = s'_1,...,s_m = s'_m, \delta] = U[s_{i_1} = s'_{i_1},...,s_{i_m} = s'_{i_m}, \delta] \quad (121)$$

for any sequence $< i_1,...,i_m > \neq < 1,...,m >$. ■

Apart from the suggested methods there is another method of solving linear equation system based on reduction of system to one non-linear equation, and then it is resolved according to Chapter 6.

We shall construct the CF grammar G' given below based on the original unambiguous acyclic CF grammar $G = < V, A, \alpha_0, R >$, which defines language *L(G),* system (98) is solved on:

$$G' = < V \cup \{*\}, A \cup \{\alpha'_0\}, \alpha'_0, R \cup \{\alpha'_0 \to \alpha_0 * \alpha_0 * ... * \alpha_0\} >, \quad (122)$$

in which the number of instances of non-terminal α_0 (axiom of the original grammar *G*) in the right part of rule added to scheme *R* is equal to the number of equations in system (98), i.e. *m* (therefore, the number of instance of "*" separator, added to the term alphabet V, is *m*-1).

Then let us consider the following word equation on CF language *L* (G')

$$< s_1 * s_2 * ... * s_m = s'_1 * s'_2 * ... * s'_m, \delta >, \quad (123)$$

with unifier, if such exists, which obviously is the same as the unifier for the solvable equation system; alternatively, the absence of unifier (123) is equivalent to incompatibility of system. This equation is non-linear, since each variable can be included into its left and right parts $m > 1$ times. However, one may easily verify that the problem of constructing non-linear equation unifiers of this type is algorithmically solvable, and the algorithm from Chapter 6 provides its solution in a finite set of steps.

9. Non-linear Equation Systems

Non-linear equation system

$$< s_1 = s'_1, \ldots, s_m = s'_m, \delta > \quad (124)$$

allows for at least one non-linear equation kernel in which at least one of terms s_i or s'_i has more than one instance of at least one of the variables.

Since according to Theorem 5, the problem of constructing non-linear equation unifier generally does not have an algorithmic solution, the problem of constructing the unifier of non-linear equation system is also non-solvable. However, analysis suggests that there is a subset of these systems family, which is of practical interest, since the problem is solvable for it.

The following observation forms the basis of suggested algorithm for building unifier of non-linear equation system. If under sequential solution of equations, forming the system, according to (119) terms s_i and/or s'_i in subsequent equation

$$< s_i = s'_i, \bar{\delta}_{i-1} > \quad (125)$$

contain more than one variable input γ, but basic substitution (rule)

$$\gamma \to \overline{\beta} \in \overline{\delta}_{i-1} \tag{126}$$

is such that $\overline{\beta} \in V^*$, and this condition holds for all variables of this type, then obviously the equation (125) is in fact linear – if $\inf_G \{s_i[\overline{\delta}_{i-1}], s_i'[\overline{\delta}_{i-1}]\}$ exists, then $\overline{\delta}_i$ will contain the same basic substitution $\gamma \to \overline{\beta}$. In other words, (126) is equivalent to lack of variables similar to γ in equation (125), since in terms of form $s_i[\{\gamma \to \overline{\beta}\}]$ and $s_i'[\{\gamma \to \overline{\beta}\}]$ all γ instances are absent — their places are occupied by constant $\overline{\beta}$ (the string in the alphabet V).

Therefore, if sequence (119) is formed not randomly, but in such a manner that by the time of solution of subsequent equation all variables which make it non-linear have already become "hidden" constants (see Chapter 6) obtained from solutions of preceding equations, then under this approach the unifier of equation system can be built even if the system contains non-linear equations.

Let us now consider the suggested algorithmic framework which provides estimation of solvability of non-linear equation system.

Let us introduce recursive function Ψ from two arguments q and $\widetilde{\delta}$. The first is the sequence of numbers of scanned and solved equations of system (124) which results in unifier expressed by the latter. This sequence takes the form of the string $q = j_1 / \ldots / j_r$, where $j_1, \ldots, j_r$ are equation numbers and "/" is the separator. The result of $\Psi(\Delta, \delta)$, where Δ is the empty string, δ is the suffix of solved system, which corresponds to the initial state of its solution (none of the equations is scanned, therefore obtained unifier is the mentioned suffix), is the set $Q = \{q_1, \ldots, q_p\}$, in which *i*-th element is one of the possible sequences of equation selection $q_i = j_1^i / \ldots / j_m^i$, which contributes to the constructing the unifier of the solved system. The set $\{j_1^i, \ldots, j_r^i\}$ is denoted by $\overline{q}_i$, and Ω denotes the set which contains the so-called pseudo-linear equation where all variables included in equation terms repeatedly are "hidden" constants. This set also contains linear equations originally contained in the kernel of the solvable system. If $q = \Delta$, then $\overline{q} = \{\varnothing\}$.

The definition of function Ψ takes the following form:

$$\Psi(q, \widetilde{\delta}) = \begin{cases} \bigcup\limits_{\substack{i \in \{1,\ldots,m\} - \overline{q} \\ <s_i = \overline{s}_i, \widetilde{\delta} > \in \Omega}} \Psi(q / i, U[s_i = s_i', \widetilde{\delta}]), \text{if } \overline{q} \neq \{1,\ldots,m\} \\ \{q\} \qquad\qquad \text{otherwise.} \end{cases} \tag{127}$$

According to this definition, the set of immediate subsequent steps is formed during each subsequent step of calculation which corresponds to q and $\widetilde{\delta}$. Each of these steps corresponds to one of the remaining equations which is either linear or pseudo-linear. For every equation with number *i* there are arguments of recursive call of function Ψ. The former is the input sequence q with number *i* attached through the separator on the right, and the latter takes the form of unifier of the equation $< s_i = s_i', \widetilde{\delta} >$.

Therefore, the recursive process denoted by (127) provides complete enumeration of all possible sequences of scanned equations of the system (124) by terminating the sequences, in which the subsequent selected equation is neither linear, nor pseudo-linear, or has the form of one or the other equation, but does not have a solution (unifier). The

exhaustion of all equations is the common condition of recursion termination (the second alternative (127) corresponding to $\bar{q}=\{1,\ldots,m\}$). If the application of function Ψ to initial arguments Δ and δ results in termination of all recursion branches, then depending on termination reasons it testifies that the system is incompatible (all "gaps" are resulted from the lack of solution in selected linear/pseudo-linear equations), or the system does not belong to the class of non-linear equation systems which can be solved by sequential linearization. If $\Psi(\Delta,\delta)$ results in non-empty set Q, then any element of this set $q=j_i/\ldots/j_m$ can be selected for application to the solved system of the following form

$$<s_{j_1}=s'_{j_1},\ldots,s_{j_m}=s'_{j_m},\delta>, \tag{128}$$

which is outlined in Chapter 8 in algorithm of sequential solution of linear equation system by means of cycling (119). Moreover, cycle can start with unifier $\tilde{\delta}$ (i.e. the first scanning of all equations already took place during q constructing).

Example 14. Consider grammar G from Example 11 and the system consisting of two equations

$$\begin{aligned}&<p_1:x/x/e_1=z:y/y/s,\\&z:c_2/c_2/s=p_1:x/y/t,\\&\{x\to\langle B\rangle,\ y\to\langle C\rangle,z\to\langle P\rangle,s\to\langle E\rangle,t\to\langle E\rangle\}>.\end{aligned} \tag{129}$$

As we can see, the first equation is non-linear, therefore sequence 1/2 is impossible. Equation 2 in sequence 2/1 is linear. Having solved it, we obtain:

$$\tilde{\delta}=\{x\to c_2,y\to c_2,z\to p_1,s\to\langle E\rangle,t\to\langle E\rangle\}. \tag{130}$$

The first equation, with variable x which is twice included in its left part, and variable y which is twice included in its right part, having obtained suffix $\tilde{\delta}$ with terminal substitutions $x\to c_2$ and $y\to c_2$ corresponding to x and y, becomes pseudo-linear. Having solved it, we obtain

$$\tilde{\delta}=\{x\to e_2,y\to c_2,z\to p_1,s\to e_1,t\to\langle E\rangle\}. \tag{131}$$

After another "run" of system (129) in sequence 2/1, we obtain unifier

$$\tilde{\tilde{\delta}}=\{x\to c_2,y\to c_2,z\to p_1,s\to e_1,t\to e_1\},$$

which is the solution of the system. ■

The suggested method can be generalized when, after the solution of $i-1$ preceding equations in i-th non-linear equation with multiple instances of variable γ in one of the terms, the unifier includes substitution

$$\gamma\to\bar{\beta}\in\bar{\delta}_{i-1}, \tag{132}$$

such that $\bar{\beta}$ contains non-terminals, but in alphabet V the set of words generated from $\bar{\beta}$ in grammar G is finite. Since the problem of recognizing the finiteness of the set of words, derived from non-terminal of CF grammar, has an algorithmical solution, the problem of

recognizing the finiteness of the set $V(\gamma,\overline{\delta}_{i-1})$ is also solvable; the set can be built by the means of direct generation. Furthermore the equation

$$< s_i = s_i', \overline{\delta}_{i-1} > \tag{133}$$

is equivalent to $|V(\gamma,\overline{\delta}_{i-1})|$ equations of the following type

$$< s_i = s_i',\ \overline{\delta}_{i-1} - \{\gamma \to \overline{\beta}\} \cup \{\gamma \to u\} >, \tag{134}$$

where $u \in V(\gamma,\overline{\delta}_{i-1})$.

Having fulfilled the discribed procedure with similar variables $\gamma_{k_1},\dots,\gamma_{k_r}$, which have multiple instances in terms s_i and s_i', we obtain

$$\prod_{j=1}^{r} |V(\gamma_{k_j},\overline{\delta}_{i-1})| \tag{135}$$

pseudo-linear equations of form (130) (in assume that there are no other variables with multiple instances in terms s_i and s_i' in equation (130)). The described algorithmic framework can be successfully applied to such equations.

We will not go into peculiarities of the suggested class of equations. Let us turn our attention to its applications.

10. **Applications**

The class of mathematical objects and algorithms associated with them suggested in this article develop the approach to modelling data bases (DB) and knowledge bases (KB) [13-19] designed by the author in 1979-1984 that make up the basis of *Intelligent Software Medias* (ISM) Theory described in full in [11].

Within the framework of this approach, the data base in the moment of time *t* is defined as a set of strings ("facts") W_t and the metadata base (MDB) is in the form of scheme D_t of CF grammar $G_t = < V, A, \alpha_0, D_t >$. Data base W_t is correct if $W_t \subseteq L(G_t)$. The data base with incomplete information (IDB) is defined as set of incomplete (I-) facts $X_t \subseteq SF(G_t)$, so the presence of non-terminal α in some I-fact $x \in X_t$ corresponds to the incomplete information "as per α". I-fact x' is more informative than the I-fact *x*, if $x \Rightarrow^* x'$ (in grammar G_t). IDB X_t is non-contradictory if it lacks two I-facts x and x', one of which is more informative than the other.

Example 15. Assume we have metadata base D_t which describes the following structure of facts from the field of ecological monitoring of some territory:

$$\begin{aligned}
&\langle fact\rangle \to AREA\ \langle r\rangle\ AT\ \langle moment\ T\rangle\ \langle condition\rangle\\
&\langle condition\rangle \to NORMAL\\
&\langle condition\rangle \to SMOKED\\
&\langle r\rangle \to \langle area\ code\rangle\\
&\langle area\ code\rangle \to \langle number\rangle\langle number\rangle\\
&\langle moment\ T\rangle \to \langle t\rangle\\
&\langle t\rangle \to \langle time\rangle\\
&\langle time\rangle \to \langle hours\rangle.\langle minutes\rangle\\
&\langle hours\rangle \to \langle number\rangle\langle number\rangle\\
&\langle minutes\rangle \to \langle number\rangle\langle number\rangle\\
&\langle number\rangle \to\ 0\\
&\quad ...\\
&\langle number\rangle \to\ 9.
\end{aligned} \tag{136}$$

The correct data base W_t which accumulates messages from ecological sensors of the monitored area may look the following way:

$$\begin{aligned}
&\{AREA\ 12\ \ AT\ \ 12.00\ \ NORMAL,\\
&AREA\ 16\ \ AT\ \ 13.45\ \ NORMAL,\\
&AREA\ 12\ \ AT\ \ 14.10\ \ NORMAL,\\
&AREA\ 31\ \ AT\ \ 9.00\ \ SMOKED\}.
\end{aligned} \tag{137}$$

If the informational nucleus of the environmental monitoring system is data base with incomplete information, X_t in a certain moment *t* may be the following:

$$\begin{aligned}
&\{AREA\ 15\ \ AT\ \ 12.00\ \ \langle condition\rangle,\\
&AREA\ 56\ \ AT\ \ 13.45\ \ NORMAL,\\
&AREA\ 42\ \ AT\ \ 18.\langle minutes\rangle\ \ NORMAL,\\
&AREA\ \langle r\rangle\ \ AT\ \ 18.40\ \ SMOKED\}.
\end{aligned} \tag{138}$$

In this IDB the first I-fact states the absence of information about the condition of area 15 at 12.00; the third I-fact contains the information that in some certain period of time from 18.00 to 18.59 area 42 was normal; the fourth I-fact contains the information that at 18.42 one of the monitored areas was smoke filled. Even so, IDB X_t is non-contradictory in the mentioned sense. If in the moment t we have I-fact $x = AREA\ \ 42\ \ AT\ \ 18.40\ \ SMOKED$, IDB $X_{t+1} = X_t \cup \{x\}$ will be contradictory, because

$$AREA\ \langle r \rangle\ \ AT\ \ 18.40\ \ SMOKED \Rightarrow^* AREA\ \ 12\ \ AT\ \ 18.40\ \ SMOKED, \tag{139}$$

so the entered I-fact is more informative that the one in IDB. ■

There are two queries protolanguage to the DB: sentential and termal.

In the first one, the query is SF x of the grammar G_t and the answer A_{t+1} is a set of facts derived from x:

$$A_{t+1} = W_t \cap SF_x(G_t) = \{w \mid w \in W_t \ \& \ x \Rightarrow^* w\}. \tag{140}$$

In termal protolanguage the query is sequence $< s, \delta >$ where s is term and δ is suffix in the terminology of this article. Even so the answer is a set of facts $w \in W_t$, for which the word equation in the CF language $L(G_t)$ $< s = w, \delta >$ has the solution below:

$$A_{t+1} = \bigcup_{w \in W_t} \{D[s = w, \delta]\} \tag{141}$$

Applying to IDBs, the answer to the query $y \in SF(G_t)$ is set

$$A_{t+1} = \{x \mid x \in X_t \ \& \ \exists \inf_{G_t} \{x, y\}\}, \tag{142}$$

and the answer to the query $< s, \delta >$ is set

$$A_{t+1} = \{x \mid x \in X_t \ \& \ \exists U[\gamma = s, \delta \cup \{\gamma \to x\}]\}, \tag{143}$$

where the auxiliary variable γ is not included in term s.

Example 16. The answer to the query $x = AREA\ \langle r \rangle\ \ AT\ \ \langle t \rangle\ \ SMOKED$ to DB W_t from Example 15 (purpose of the query is selection of facts about the areas where ecological sensors registered smoke) will be set

$$A_{t+1} = \{AREA\ \ 31\ \ AT\ \ 9.00\ \ SMOKED\}. \tag{144}$$

The answer to the query $< s, \delta > = < a\ \ SMOKED, \{a \to AREA\ \langle r \rangle\ \ B\ \ \langle t \rangle\} >$ to the same DB will be the same set.

The answer to the query $< s, \delta >$ to IDB X_t from example 15 according to (142)-(143) will be the set

$$A_{t+1} = \{AREA\ \ 15\ \ AT\ \ 12.00\ \ \langle condition\rangle,$$
$$AREA\ \ \langle r\rangle\ \ AT\ \ 18.40\ \ SMOKED\}.\ \blacksquare \qquad (145)$$

Protolanguage serves as a basis for creating query languages (in a wider sense – data manipulation languages) of particular data base management systems (DBMS).

As the protolanguage for knowledge representation the Post systems are used being a "string-oriented" analog of Horn clauses whose practical appliance in creating knowledge based systems and intelligent interfaces of the relational DBMS is well-known.

The knowledge base as generalization of data base is a pair $< S_t, D_t >$, where S_t is a set of productions of form $< s_0 \leftarrow s_1,...,s_m;\delta >$, and $s_0, s_1,...,s_m$ are terms, δ is the suffix, moreover, $\{s_0[\delta],...,s_m[\delta]\} \subseteq SF(G_t)$. Even so, the data base as a factual part of knowledge data base is a subset of set S_t which includes productions of form $< s_0 \leftarrow; d >$. While operating the incomplete information in all productions $s_0[\delta] \in SF(G_t) - V^*$ is acceptable, i.e. non-terminals present in SF $s_0[\delta]$.

The extensional of knowledge base $< S, D >$ (index t is omitted to simplify the record) denoted by $Ex(S,D)$, is a set of facts (strings) provable in Post system without variables intentionally equivalent to this data base. Even so the production $\sigma = < s_0 \leftarrow s_1,..., s_m; \delta >$ is intentionally equivalent to a set of Post productions without variables $\bar{\sigma}$ in the sense that

$$\bar{\sigma} = \bigcup_{d \in \delta} \{< s_0[d] \leftarrow s_1[d], ..., s_m[d] >\}, \qquad (146)$$

where Post production is in angular brackets.

Example 17. Assume we have metadata base D_t from example 15 where new rules are additionally included:

$$\begin{aligned} &\langle fact\rangle \rightarrow SENSOR\ \ \langle i\rangle\ \ AT\ \ \langle moment\ T\rangle - SMOKE \\ &\langle fact\rangle \rightarrow SENSOR\ \ \langle i\rangle\ \ AT\ \ AREA\ \ \langle r\rangle \\ &\langle i\rangle \rightarrow \langle number\rangle\langle number\rangle, \end{aligned} \qquad (147)$$

which determine the structure of facts about the ecological sensors indications and its position in the controlled areas. In a set of productions S_t of data base $< S_t, D_t >$ we include the unique production:

$$
\begin{array}{l}
< AREA\ \ r\ \ AT\ \ t\ \ SMOKED \leftarrow \\
SENSOR\ \ i\ \ AT\ \ t - SMOKE, \\
SENSOR\ \ i\ \ AT\ \ AREA\ \ r; \\
\{r \to \langle r \rangle, \\
t \to \langle time \rangle, \\
i \to \langle i \rangle\} > .
\end{array} \tag{148}
$$

The usage of symbols r and i for denotation of variables of production and non-terminals of metadata bases do not bring any ambiguity because the variables have their place in left parts, while non-terminals are located in the right parts. Besides, the record of non-terminals requires using metalinguistic brackets. Meanwhile, the variable operating area is the production where it was used, whereas non-terminal operating area are all KB productions.

The presence of the facts about sensors and their indications in the data base corresponds to the availability of the following productions in set S_t:

$$
\begin{array}{l}
< SENSOR\ \ 11\ \ AT\ \ AREA\ \ 22 \leftarrow, \{\varnothing\} >, \\
< SENSOR\ \ 16\ \ AT\ \ AREA\ \ 22 \leftarrow, \{\varnothing\} >, \\
< SENSOR\ \ 18\ \ AT\ \ AREA\ \ 35 \leftarrow, \{\varnothing\} >, \\
< SENSOR\ \ 11\ \ AT\ \ 15.00\ \ - SMOKE \leftarrow, \{\varnothing\} >, \\
< SENSOR\ \ 18\ \ AT\ \ 16.00\ \ - SMOKE \leftarrow, \{\varnothing\} > .
\end{array} \tag{149}
$$

We can easily see that according to (149) the KB extensional includes the following two elements:

$$
\begin{array}{l}
AREA\ \ 22\ \ AT\ \ 15.00\ \ SMOKED, \\
AREA\ \ 35\ \ AT\ \ 16.00\ \ SMOKED,
\end{array} \tag{150}
$$

and elements

$$
\begin{array}{c}
SENSOR\ \ 11\ \ AT\ \ AREA\ \ 22, \\
\ldots \\
SENSOR\ \ 18\ \ AT\ \ 16.00\ \ - SMOKE,
\end{array} \tag{151}
$$

that correspond to the productions from (149). ■

The answer to the query $< s, \delta >$ is a set of elements of extensional $w \in Ex(S, D),$, i.e. the facts derived from KB $< s, \delta >$, for which equation $< s = w, \delta >$ has the solution:

$$
A_{t+1} = \bigcup_{w \in Ex(S,D)} \{D[s = w, \delta]\} . \tag{152}
$$

So the answer to query $< a \ \ SMOKED, \{a \to AREA \langle r \rangle \ AT \ \langle t \rangle\} >$ to the KB from example 17 will be the set of two elements (150).

As it can be seen, the string-oriented representation of information allows direct use of natural language (NL) to operate DB and KB. If there is predicate (predicate-actant) representation that form the core of relational data bases and their various derivatives, it requires the creation and maintenance of rather difficult-to-use natural language interfaces characterized by a limited subsets of the lexicon, syntax, semantics and pragmatics of NL with its own knowledge representation models (KRM). The string-oriented representation does not require substantiational time to create even polylanguage systems to communicate with the DB and the KB due to the unified and easy-to-use KRM.

[11] gives different axiomatics of the inference of responses to queries to the KB, including knowledge bases of the so-called procedural connection (extension), which provides in a process of inference a call of "hard" (non-modifiable) programs (for example DBMS), as well as a number of issues related to the need for well-formed and efficient software/hardware implementation (including through the parallel inference) on very large data and knowledge bases with regard to different network ISM infrastructures.

The key element of the described axiomatics is the so-called *S*-unification which is the solution of equation $< s = s_0^i, \delta \cup \delta_i >$ where $< s, \delta >$ is query (initial or derived, i.e. formed during the inference process), while s_0^i and δ_i is a head and a suffix of *i*-th production of the KB; meanwhile, a set of variables that have their place in terms s and s_0^i do not intersect.

The primary tool of reducting the inference computational complexity (including access to data bases), i.e. complexity of mass solutions of equation like $< s = w, \delta >$ and $< s = s_0^i, \delta \cup \delta_i >$, are the so-called *SF*-trees. Non-terminal nodes of the *SF*-tree are sentential forms of the CF grammar G_t, the root is its axiom α_0, and leaves are the objects which require direct solution of these equations (elements of the DB/IDB and the heads of productions). The backbone and the effector feature of *SF*-tree which enables cutting off its subtrees without their direct scanning, is condition $x \Rightarrow^* x'$ that satisfies the node x and each of its descendants x'. Therefore, when $\inf_{G_i} \{x, s[\delta]\}$ does not exist (it means the corresponding equation has no solutions), this condition is executed for any descendant of the node x, which allows excluding from scanning all the subtree roots x.

The development of the *SF*-trees are the so-called ΔSF -trees which exclude the possibility of multiple duplication of the SF's symbols by means of storage in every node – direct descendant – of only the so-called increment relative to its direct ancestor.

The so-called grammar approach to coding, which provides constructing codes on stochastic CF languages and on this basis much more effective than conventional Huffman codes constructed under the assumption on the admissibility of all words in the alphabet V , is associated with the apparatus of inference optimization and data base management developed in [11,18]. For all the proposed classes of codes, algorithms of block decoding were developed based on the transformation of CF grammar encoded words to bilateral type that provide an increase in decoding speed by K times if compared to the bit-decoding, where K being the length of the block that can be selected based on the amount of memory available for allocation of the decoding tables.

The basis for the practical application of the developed mathematical apparatus was the so-called page data representation (notice that "up-to-bottom", "left-to-right" linearized page is also string), which allowed, in particular, in 1986 implementation of the so-called page DBMS on mainframe and personal computers, being in fact the first network web-server, and in 1990 implementation of an intelligent software media that provided the distributed intelligent processing of the streams of messages (pages) at the rate of their income from geographically separated sensors and computer terminals in accordance with the current state of KB, i.e. to create, in modern terminology, a distributed multiagent system of the OLAP class (On-Line Analytic Processing) with a unified web-interface. In this case, unlike in T. Berners-Lee paradigm [20, 21], for creating and maintaining paged data bases (meaning web-sites in the modern sense) it did not require a metalanguage like HTML (HyperText Markup Language). This greatly simplified the process nowadays referred to as web-mastering making it available to any user of network information resources organized in the form of harmonically associated personal and public DB updated in real time by ISM in accordance with the current logic of processing of incoming messages. This logic could, if necessary, evolve to adapt to dynamic external conditions through rapid introduction of local changes in the distributed network KB by knowledge engineers without the involvement of the ISM designers.

Thus, the development of versatile mathematical apparatus based on the universal string-oriented representation of data, which goes back to the algorithmic approach in information theory, enabled not only to compile correctly the earlier data model and knowledge-based relational (predicate-actant) representation [8, 9, 22], but in a short

period of time to create a practical and effective tools, while their counterparts began to appear 8-10 years after the first page DBMSs and ISMs were actually put into operation.

Marked at the beginning of this century transition to the creation of *network-centric robotics* [23-31] gives reason to develop the existing theoretical and practical groundwork to create a new generation of distributed ISMs as the basis for formation system of autonomous mobile networking groups (masses) of multisensory and multi-functional robots. Elements of the developed apparatus can be effectively used to create on-board intelligent robotic systems that provide problem solving in the localization of the observed objects, their spatial and indicative clustering, classification, identification, and, on this basis, continuous monitoring of the environment groups of robots. The limitedness of on-board power margin of robots leads to minimization of the computational complexity of these tasks that are traditionally related to the field of pattern recognition, i.e. developing such algorithms for their solution that would ensure the minimum time function of on-board computing and on this basis would maximize the periods while robots being in an active state. As each of these tasks can be formalized in the form of an equation in the words of the CF language, efficient and universal means of this optimization can be mentioned $SF(\Delta SF)$-trees; ideology and techniques of robotic applications will be discussed in a separate publication.

Another promising direction of applying the described mathematical apparatus can be *modelling of genetic processes*.

If we accept the hypothesis that the genome of a class of organisms is a CF-grammar $G=<V,A,\alpha_0,R>$, and the current state of an organism at moment *t* is its sentential form x_t, then the process of life (development) of an organism is permanent generation $x_t \Rightarrow^* x_{t+1}$ by replacing non-terminals $\alpha \in A$ (developable elements of an organism) on the right parts of rules $\alpha \to \beta \in R$ containing both non-terminals and terminals. The organisms $x_t \in V^*$ in which non-terminals are absent have no opportunities for development, and they cease to exist (die).

Regeneration (self-reproduction) of the elements is possible due to the presence of cyclic rules of form $\alpha \to \alpha\beta$ in scheme R, while their alternative rules $\alpha \to \upsilon$ under certain conditions can lead to termination of regeneration.

Each generator rule $\alpha \to \beta \in R$ is characterized by a minimal amount of energy intake $\Delta E(\alpha \to \beta)$ of non-terminals α required for the direct generation $x_t^1 \alpha x_t^2 \Rightarrow x_t^1 \beta x_t^2$.

Generation of new organisms is made by means of combination of two parent cells x_t and x_t' with creation of cell

$$y_t = \sup_G \{x_t, x_t'\}, \tag{153}$$

which carries all genetic information common for both parents, i.e. genetic properties that are distinctive for both of them. Then it enables generation of cell

$$\bar{y}_t = \inf_G \{x_t, x_t'\}, \tag{154}$$

where each non-terminals α which presents in SF y_t and one of SF x_t, x_t' is replaced by strings $\bar{\beta}$ derived from α in other SF in accordance with (52). As a result, the descendant in the first part of non-terminal α gets genetic information $\bar{\beta}$ possessed by only one parent and not possessed by the other one. If the both parents have the same informativity as per α, the descendant will have the same non-terminal α.

In general, however, $\inf_G \{x_t, x_t'\}$ does not exist due to the fact that some non-terminals $\alpha_{i_1},...,\alpha_{i_r}$ that form a part of y_t have various strings $\bar{\beta}_1 \neq \bar{\beta}_1',...,\bar{\beta}_r \neq \bar{\beta}_r'$ arising from $\alpha_{i_1},...,\alpha_{i_r}$ in derivations $y_t \Rightarrow^* x_t$, $y_t \Rightarrow^* x_t'$. In this situation $\bar{y}_t$ can be formed through the implementation of two operations:

1) replacement of non-terminals α such that $y_t = y_1 \alpha\, y_2$, $x_t = x_1 \alpha\, x_2$, $x_t' = x_1' \bar{\beta}\, x_2'$, $y_1 \Rightarrow^* x_1$, $y_1 \Rightarrow^* x_1'$, $y_2 \Rightarrow^* x_2$, $y_2 \Rightarrow^* x_2'$, with $\bar{\beta}$;

2) replacement of non-terminals α such that $y_t = y_1 \alpha\, y_2$, $x_t = x_1 \bar{\beta}\, x_2$, $x_t' = x_1' \bar{\beta}'\, x_2'$, $\bar{\beta} \neq \bar{\beta}'$, $y_1 \Rightarrow^* x_1$, $y_1 \Rightarrow^* x_1'$, $y_2 \Rightarrow^* x_2$, $y_2 \Rightarrow^* x_2'$, with one of the strings $\bar{\beta}$ or $\bar{\beta}'$.

It is evident that the first operation is fully consistent with logic of constructing $\inf_G \{x, x'\}$, whereas the second in the presence of "genetic selection" implements it in favor of one parent.

Conclusions (creations) $\alpha_0 \Rightarrow^* \bar{y}_t$ are encoded by means of DNA sequences consisting of nucleotide bases. The kernel of coding is, apparently, bialternativity of scheme R (each non-terminal α has two alternatives $\alpha \to \beta, \alpha \to \beta' \in R$), which results from the presence of two types of pairings of the mentioned bases (adenine-thymine and cytosine-guanine) [32] in the DNA.

Artificial change of individual sections of these chains in living cells (*in vivo*) under the influence of various factors explains the changes in the chains of rules that ensure the creation and leads to the emergence and development of genetically modified organisms

$\tilde{y}_t$, some of which may have different internal and external abnormalities due to non-fulfillment of condition $\sup_G\{x,x'\} \Rightarrow^* \tilde{y}_t$ (for that matter $\tilde{y}_t$ is a mutant that has features not characteristic for any of the parents).

Despite the fact that the introduced formalization obviously has hypothetical and descriptive nature, it is distinguished by its original settings from the methods of usage of the grammars apparatus [32, 33] in genetic engineering, so it may result in its creative application.

The constructive application of the results presented in this article may be found also in *social engineering*, especially in its most demanded part – non-directive control of societies [34].

A society in electoral situation may be presented in the form of CF grammar $G =< V, A, \alpha_0, R >$ where α_0 is "society" axiom, $\alpha_1,...,\alpha_m$ are individuals that form it, $V = \{\upsilon_1,...,\upsilon_n\}$ – election variants, so

$$\begin{aligned} R = \{&\alpha_1 \to \upsilon_1,...,\alpha_1 \to \upsilon_n, \\ &... \\ &\alpha_m \to \upsilon_1,...,\alpha_m \to \upsilon_n\}, \end{aligned} \tag{155}$$

i.e. in its initial state $L(G) = V$, and $\alpha_j \to \upsilon_i \in R$ which correspond to *j*-th selection of individuals of *i*-th variant. The purpose of the electoral headquarters supporting the *i*-th variant (for example, *i*-th candidate for elective position) is to turn G into $G_i =< V_i, A_i, \alpha_0, R_i >$, where $L(G_i) = \{\upsilon_i, \Delta\}$, just like before Δ is an empty string, and make it with the help of non-directive methods which exclude the possibility of removing the rules of scheme R. In the adopted formalization any headquarters uses two types of electoral technologies corresponding to the derivation chains

$$\upsilon_j \Rightarrow \upsilon_j^{(1)} \Rightarrow ... \Rightarrow \upsilon_j^{(k_j)} \Rightarrow \upsilon_i \tag{156}$$

and

$$\upsilon_j \Rightarrow \overline{\upsilon}_j^{(1)} \Rightarrow ... \Rightarrow \overline{\upsilon}_j^{(l_j)} \Rightarrow \Delta. \tag{157}$$

Technology (156) – the "positive one" – forms the electorate idea that all positive features that distinguish *j*-th candidate from the others are also possessed by *i*-th candidate. Technology (157) – the "negative one" – forms the electorate idea that the choice of *j*-th candidate is a dead-end, therefore, it is pointless. In this case, greater complexity (informativity) of object υ_i compared to object υ_j (because of $\upsilon_j \Rightarrow^* \upsilon_i$) at the level of

ordinary consciousness is interpreted as a big experience, knowledge and wisdom of *i*-th candidate in comparison to *j*-th candidate. In general, due to (156) *i*-th candidate embodies all the best features possessed the remaining candidates, and thanks to (157) it is completely devoid of their defects.

The essence of the electoral process is to create a mass consciousness of society (it means ultimately, consciousness of each individual), CF $G_i =< V_i, A_i, \alpha_0, R_i >$ such that

$$\begin{aligned} V_i &= \{\upsilon_i\}, \\ A_i &= A \cup (V - \{\upsilon_i\}) \cup \left(\bigcup_{\substack{j=1 \\ j \neq i}}^{n} \left[(\bigcup_{q=1}^{k_j} \{\upsilon_i^{(q)}\}) \cup (\bigcup_{q=1}^{l_i} \{\overline{\upsilon}_j^{(q)}\}) \right] \right), \\ R_i &= R \cup \left(\bigcup_{\substack{j=1 \\ j \neq i}}^{n} \{\upsilon_i \to \upsilon_j^{(1)}, ..., \upsilon_j^{(k_j)} \to \upsilon_i\} \right) \cup \\ &\cup \left(\bigcup_{\substack{j=1 \\ j \neq i}}^{n} \{\upsilon_i \to \overline{\upsilon}_j^{(1)}, ..., \overline{\upsilon}_j^{(l_j)} \to \Delta\} \right). \end{aligned} \tag{158}$$

Formation, as a rule, is performed by gradual, "soft" introduction of separate rules of scheme R_i to the individual conscience, with each rule clearly not contrary to the moral and psychological attitudes typical for different groups of society, therefore not rejected by them. After this, in a relatively short period of time immediately preceded by the election the initiation of "thinking process" is implemented, and voters, having applied the rules of scheme R_i embedded in their minds, by the voting time come to the "right" decision $L(G_i) = \{\upsilon_i, \Delta\}$, where Δ is attributed to all candidates, except for *i*-th candidate.

Naturally, in general, there is a race of several election headquarters; each of them attempts to implement this technology to their own advantage basing on the available resources.

A more detailed discussion of this and other possible applications goes beyond the scope of this article.

The author expresses his sincere appreciation to V.G. Tyminskiy

Bibliography

1. Word Equations and Related Topics. Ed. by K.U. Schulz. — Lecture Notes in Computer Science, Vol.572. — Springer, 1991.
2. Lothaire M. Applied Combinatorics on Words. — Cambridge University Press, 2004.
3. Zikmann Y., Zabo P. Universal Unification and Classification of Equational Theories. — Cybernetic Collection; new series, Issue 21. — M: Mir, 1984.
4. Makanin G.S. Problem of Equation Solvability in a Free Semigroup. — Mathematical Collection, Vol.103, pp.147-236, 1977.
5. Makanin G.S. Equations in a Free Group. — "Izvestia" (Academy of Science, USSR). Mathematical Collection, Vol.46, pp.1199-1273, 1983.
6. Makanin G.S. Solvability of Universal and Positive Theories of a Free Group. — "Izvestia" (Academy of Science, USSR). Mathematical Collection, Vol.48, pp.735-749, 1984.
7. Robinson J. Computer-Oriented Logic Based on a Resolution Principle. — In: Cybernetic Collection, Issue 7. — M: Mir, 1970.
8. Kowalski R. Predicate Logic as a Programming Language. — In: Logic and Databases. Ed. by H.Gallaire and J. Minker. — Plenum Press, 1978.
9. Kowalski R. Logic for Problem Solving. — N.Y.: North Holland, 1979.
10. Aho A., Ullmann J. The Theory of Syntactic Analysis, Translation and Compilation, in 2 volumes. — M: Mir, 1979.
11. Sheremet I.A. Intelligent Software Medias for Computerized Information Processing Systems. — M: Nauka, 1994.
12. Kolmogorov A.N. Three Approaches to Definition of "Information Measure" Concept. — In: The Selectas. Vol.3 "Information Theory and Theory of Algorithms. — M: Nauka, 2005.
13. Bushenkov V.A, Sheremet I.A. Unformalized Database Management System for PCs. — Articles for All-Soviet Union Conference "Problems of Economic Experiments Modelling". — M: VNIIPOU, State Committee for Science and Technology, USSR, 1986.
14. Shaburov E.P., Sheremet I.A. Implementation and Application of Symbolic-Production Intelligent Software Medias. — Control Systems and Computers, 1990, #5.
15. Sheremet I.A. Implementation of Dynamic Hypertexts by Means of Symbolic-Production Intelligent Software Medias. — Articles for All-Soviet Union Workshop Conference "Economics and IT". — Gomel, 1991.
16. Shaburov E.P., Sheremet I.A. Principles of Conveyor Implementation of Symbolic-Production Systems Based on Homogeneous Structures. — Articles for V All-Soviet Union Conference "Homogeneous Computational Structures, Systems and Environments", part I — M.: 1991.
17. Nazarov S.V., Sheremet I.A. Symbolic-production Systems as a Means of Formal Description of Informational Processes. I, II. — Automatics and Telemechanics, 1992, #9, 10.

18. Sheremet I.A. Effective Coding of Formalized Messages. — Cybernetics and System Analysis, 1992, #3.
19. Sheremet I.A. On a Formalism Description for Knowledge Representation. — Programming, 1993, #2.
20. Berners-Lee T., Cailliau R., Luotonen A., Nielsen H.F., Secret A. The World-Wide Web. – Communications of the ACM, Vol. 37 (1994), #8.
21. Berners-Lee T. World-Wide Computer. – Communications of the ACM, Vol. 40 (1997), #2.
22. Codd E.F. Relational Databases: A Practical Foundation for Productivity. – Communications of the ACM, Vol. 25 (1982), #2.
23. Network-Centric Warfare: Creating a Decisive Warfighting Advantage. — Washington, DC: Department of Defense, 2004.
24. Defense Transformation and Network-Centric Systems. Ed. by R. Suresh. — Proc. of SPIE, Vol. 6249, 2006.
25. Mobile Ad-hoc and Sensor Networks XIX. Ed. by J. Cao, I. Stoimenovic, X. Jia, S. K. Das. — Lecture Notes in Computer Science, Vol. 4325. — Springer, 2006.
26. Wireless Sensor Networks. Proc. EWSN 2007. Ed. by K. Langendoen, T. Voigt. — Lecture Notes in Computer Science, Vol. 4373. — Springer, 2007
27. Handbook of Sensor Networks. Algorithms and Architectures. Ed. I. Stoimenovic. — John Wiley AND Sons, 2007.
28. Multimodal Surveillance. Sensors, Algorithms, and Systems. Ed. by Zh. Zhu, T.S. Huang. — Artech House, 2007.
29. Sheremet I.A. Concept of the "Network-Centric Warfare" and its Specific Implementation. — Independent Military Review, 2005, #44.
30. Sheremet I.A. Network Centricity and Robotics: Two Sides of Future Military Systems. — National Defence, 2006, #3.
31. Kaliaev I.A., Sheremet I.A. Military Robotics: Choice of Approach. — Mechatronics, Automation, and Control, 2008, #2.
32. Paun Gh., Rosenberg G., Salomaa A. DNA-Computer. New Computing Model. Translated from English – M: Mir, 2004.
33. Marcus S. Language at the Crossroad of Computation and Biology. – In: Computing with Bio-Molecules. Theory and Experiments. Ed. by Gh. Paun. – Berlin: Springer, 1998.
34. Semashko K.V., Sheremet I.A. Mathematical Modelling of Informational-Psychological Relations in Sociums. Ed. By I.A. Sheremet. – M: Nauka, 2007.