Sheremet I.A.
Doctor of Engineering science,
Professor, member of the European
Academy of Natural Sciences e.V.

Grammatical codings

Special Edition:
Hannover Annual
Vol. 1, 2012

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

Hannover, 2012

Sheremet I.A.
Doctor of Engineering science,
Professor, member of the European
Academy of Natural Sciences e.V.

Grammatical codings

Hannover, 2012

Special Edition:
Hannover Annual
Vol. 1, 2012

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

ISBN 978-3-942944-19-9

Annotation

This work is devoted to reviewing of grammatical approach to coding, within the framework of which the methods of construction of effective codes and corresponding coding-decoding algorithms are being developed and applied not to a set of all words in the initial alphabet, but to subsets of this set – to the languages, specified by means of their finite representations – grammars. Essential reduction of constructed codes redundancy and, accordingly, of their cost is ensured as a result.

Three families of grammatical codes – alphabetic (static), automata (dynamic) and structural – are considered. For each of these families algorithms of the irredundant codes construction, coding of initial messages (words of formal languages) and decoding of binary sequences are given. The methods of block decoding, which ensure its essential acceleration, are described here.

The work is intended for specialists in the field of the information theory, theoretical and applied computer science, databases, and for students and post-graduate students of corresponding specialities.

Printed by **NG Verlag (Viatcheslav Demidov Inhaber)**
Berlin
Tel.: 030/4442460
Fax: 030/44739165
E-mail: SlavaDemidov@t-online.de
Internet: www.ng-verlag.de

Content

Introduction 5

1. Grammatical Interpretation of Coding 6

2. Codes, Decipherable Over Context-Free Languages 6

3. Construction of Codes,
   Effective Over Stochastic Context-Free Languages 14

4. Effective Grammatical
   and Generalized-Prefix Codes Interrelations 21

5. Block Decoding. 23

6. Optimization of Syntax Analysis of Coded Messages. 28

7. Parallel Syntax Analysis of Coded Messages. 32

Conclusion 37

Bibliography 38

Introduction

Irredundant codes of variable length, which ensure maximum degree of compression, are being constructed, proceeding from aposteriory possibility of their application to any word in the primary alphabet. At the same time, messages are transmitted through communication channels of computer networks and are accumulated in the memory of their elements far not arbitrary, but satisfying to certain semantic-syntactical restrictions, i.e. this messages are strictly formalized. Therefore, the irredundant codes synthesized by known methods [2], with reference to the streams of messages circulating in networks, possess, as a rule, considerable redundancy, though, from positions of coding of arbitrary words in the primary alphabet, these codes, certainly, are optimal.
In the present monography, the so-called grammatical interpretation of coding, based on structure description of coded messages by means of formal grammars, and construction of coding-decoding procedures on the basis of generating and parsing of words of formal languages, is considered. Within this interpretation synthesis of irredundant codes is performed by means of stochastic grammars, formed from grammars, describing the structure of messages, adding the probability information [3] to them.
Because of difficulties of general grammars practical application for specifying the languages, it is possible without a damage to generality of results to be restricted by context-free grammars, representing convenient and rather powerful tool for characterization of sets of the formalized messages. Grammatic interpretation of coding allows to synthesize coding-decoding automata by means of minimum correction of the parsers, which are being constructed according to grammars by known methods [4]. In this connection two constituents of decoding process – localization of information in sense [1] and decoding itself are integrated into the uniform process of syntax analysis of the word in the binary alphabet (a message code), one of "by-effects" of which is consequent inflow of symbols of the coded message on a decoder exit. Coding is performed similarly too.
The author offered the grammatical interpretation of coding in the early eighties. The first publication on this subject in the academic press appeared in 1992 [5] (thus the paper had arrived in editorial office of "Cybernetics" journal in 1987), and in the most expanded form the approach is written up in the monography [6] (item 5.2).

1. Grammatical Interpretation of Coding

Let's consider the set of admissible messages, which are subject to coding, in the form of the language $L(G) = \{w \mid a_0 \overset{*}{\underset{G}{\Rightarrow}} w \ \&\ w \in V^*\}$, where a_0 and V are accordingly an axiom and terminal alphabet of grammar G. Coding device represents the mapping A_K from the set $L(G)$ into a set of all words in the alphabet $\{0,1\}$. Here the lower index K serves for a code designation, i.e. the function, which consecutive application to a word $w \in L(G)$ ensures its coding, that is shaping of binary sequence $A_K(w)$. Accordingly, the decoding device implements the inverse mapping of A_K^{-1} from $\{0,1\}^*$ to $L(G)$. In the general case, various definitions of K are possible, which specify, actually, different classes of codings. Thus, irrespective of a concrete method of definition, the code K should satisfy to the requirement of unambiguity of decoding:

$$(\forall\, w \in L(G))\ A_K^{-1}\,(A_K(w)) = w, \qquad (1)$$

Following [7], the code K, satisfying to the specified requirement, let us name as decipherable over the language $L(G)$. The code K, decipherable over the language $L(G)$ and supposing decoding $A_K(w)$ by means of deterministic analysis without returns, we will name as decipherable with a finite delay over the language $L(G)$.
Let us notice that the code, decipherable with a finite delay over the language L, preserves this feature over any subset $L' \subseteq L$. In the general case, the contrary is incorrect: the code, which is decipherable over the language L, can be not as such, relating to the language $L' \supset L$.
Searching and examination of criteria of unambiguity conditions of decoding, at accomplishment of which the codes are decipherable relating to definite languages, makes the main content of the following item 2.

2. Codes, Decipherable Over Context-Free Languages

Let us define a code K and corresponding to it coding A_K as follows:

$$K : V \rightarrow \{0,1\}^*, \qquad (2)$$
$$A_K(w) = K\,(v_{i_1})\ \ldots\ K(v_{i_m}),$$

where $w = v_{i_1} \ \ldots\ v_{i_m}$, $v_{i_j} \in V$. From expressions (2) it follows that the code of the word in the primary alphabet V is the concatenation of symbols codes, entering this word. Thus, the symbols codes in the course of coding remain constant, i.e. they do not depend on a context, enclosing a symbol. It is accepted to name similar coding **alphabetic** [1].
Let us consider the problem of decoding uniqueness with reference to the given class.
Let we have a code K of the form (2) and unambiguous context-free grammar $G = \langle V, V_N , a_0 , R\rangle$, where a_0 и V, as well as earlier, are an axiom and the terminal alphabet of grammar G, and V_N and R – accordingly, its non-terminal alphabet and a set of generating rules (scheme). We will define grammar $G^K = \langle V^K , V_N^K , a_0^K , R^K\rangle$ as follows:

$$V^K = \{0,1\},$$
$$V_N^K = V_N \cup V, \qquad (3)$$
$$R^K = R \cup \left(\bigcup_{v \in V} \{v \to x\} \right),$$
$$a_0^K = a_0,$$

where $x = K(v)$. For definiteness we suppose that $V \cap \{0,1\} = \varnothing$.
There is a lemma cited without proof, which correctness directly follows from a principle of G^K grammar construction.

L e m m a 1. For *any word* $w \in L(G)$ *there exists the only word*

$w' \in L(G^K)$ *such, that* $w' = A_K(w)$, *and* $w \overset{*}{\underset{G^K}{\Rightarrow}} w'$.

The existence of alphabetic codes undecipherable over to V^*, but decipherable over $L(G)$ is generally possible. Namely, if the code K is such that $A_K(w_0) = A_K(w_1) = \ldots = A_K(w_q)$, where $w_0 \in L(G)$, and all pairwise unequal $w_1, \ldots, w_q$ are not words $L(G)$, so the requirement of decoding unambiguity of a word w_0 is fulfilled, though the code K over the language V^* is undecipherable.

E x a m p l e 1. Let we have a unambiguous context-free grammar $G = \langle V, V_N, a_0, R \rangle$, where $V=\{a, b, c\}$, $V_N = \{A, B\}$, $a_0 = A$, and a set of generating rules includes the following rules:
A → aB,
B → cB,
B → b.
As seen, the grammar G generates the language
$L(G)=\{ac^nb \mid n \geq 0\}$.
Alphabetic code K, such, that
$K(a) = 0$,
$K(b) = 1$,
$K(c) = 0$,
is decipherable over context-free language $L(G)$, though relating to the language $V^*=\{a, b, c\}^*$ this code is undecipherable by reason of
$K(a) = K(c)$.
In particular, two words – ab and cb – of language V^* correspond to the word 01 of language $\{0,1\}^*$, but the only word ab of the language $L(G)$, because cb $\notin L(G)$.■
The general criteria of decipherability in the class of alphabetic codes is defined by the following theorem.

T h e o r e m 1. *Code K is decipherable over the language L(G), where G is unambiguous context-free grammar, if and only if the grammar* G^K *is unambiguous.*

P r o o f . If G^K is unambiguous, so for any $w' \in L(G^K)$ there is a unique derivation tree and, hence, the unique word $w' \in L(G^K)$ such that $w \overset{*}{\underset{G^K}{\Rightarrow}} w'$. That is why we decipher the code K over the language $L(G)$, which was to be proved. Let now we decipher the code K over the language $L(G)$, and grammar G^K is ambiguous. The latter is equivalent to the existence of the word $w' \in L(G^K)$, to which corresponds not less than two various canonical derivation $a_0^K \overset{*}{\underset{G^K}{\Rightarrow}} w'$. So far as G is unambiguous, so for each word $w \in L(G)$ there is the only canonical inference $a_0^K \overset{*}{\underset{G^K}{\Rightarrow}} w$ and, hence, $a_0^K \overset{*}{\underset{G^K}{\Rightarrow}} w$. From this it follows that two different canonical derivation can take place only in that case when there are two words w_1 and w_2 of the language *L(G)* such that $w_1 \underset{G^K}{\overset{*}{\Rightarrow}} w'$ and $w_2 \underset{G^K}{\overset{*}{\Rightarrow}} w'$. It means that two various words of the language *L(G)* have identical codes, that is the code *K* is undecipherable over *L(G),* that contradicts to the theorem statement and invalidates the supposition about ambiguity of the grammar G^K. Hence, under a condition of the code *K* decipherability over the language *L(G)*, the grammar *G* is unambiguous, which was to be proved.■

However, because of unsolvability of the problem of unambiguity of context-free grammars recognition [4], the criteria of the theorem 1 is not constructive. In this connection, the recognition problem of alphabetic code decipherability over context-free (CF) language is also algorithmically unsolvable [7]. We will be limited therefore to searching the criteria of alphabetic codes decipherability with reference to such classes of context-free grammars, which, on the one hand, are obviously unambiguous, and on the other – the problem of recognition of concrete grammar belonging to the fixed class is algorithmically solvable. Such class is context-free grammars, supposing the deterministic analysis without returns (DCF-grammars – deterministic context-free grammars).

Let us consider the criteria of alphabetic codes decipherability with reference to DCF-grammars.

Classical representative of DCF-grammars are *LR*(*k*)-grammars, where for any specified *k* it is possible to define, whether the concrete grammar belongs to the class *LR*(*k*) [4]. It allows to formulate simply enough the required criteria [7]: if grammar *G* and code $K : V \rightarrow \{0, 1\}^*$ are so, that for the specified *k* grammar G^K is *LR*(*k*)-grammar, then the code *K* is decipherable over the language *L(G).*

Decoding is implemented on the basis of the determined bottom-up analysis of sentences in the alphabet {0,1}. Therefore, the code *K*, satisfying to the mentioned statement, is decipherable with a finite delay over the language *L(G).* The only addition to traditional algorithm of *LR*(*k*)-analyzer consists in recording of grammar G^K non-terminals from *V* alphabet in an output string immediately after their occurrence in a parser's stack [7]. Coding of words $w \in L(G)$ by means of alphabetic codes does not differ from the usual.

It is obvious that the decipherability criteria, similar to mentioned above, is correct for any known class of DCF-grammars, for example *G(k), LL(k)* etc.

In the course of alphabetic coding, the symbols codes remain constant. In [8] it is offered, instead of compression of database element by means of Huffman code, constructed for *V* alphabet of symbols, used in elements of DB, to compress fields of file entries by means of codes, constructed for subsets of *V*, characteristic for these fields or their types (digital, letter and etc.). The effect is reached because potencies of subsets, as a rule, notably less than $|V|$. Here that fact is most important that various code combinations can correspond to the same symbol, which is taking place in values of different fields. It puts the described method beyond the framework of alphabetic coding, for which static correspondence of a code combination to a symbol, which does not vary depending on a context, is characteristic. Drawing analogy between the method [8] and automatic coding in the sense of [1], it is easy to notice that alphabetic coding can be generalized in such a way that codes of symbols depending on analyzer DCF-language condition could vary. Because of similar generalization we come to **automata coding** of DCF-languages. As for any such language there exists *LR*(1)-grammar [4] generating it, it is enough to consider automata coding for a case of *LR*(1)-grammars.

Let we have *LR*(1)-grammar *G* and the mapping $\Psi: S \times V \rightarrow \{Y, N\}$, corresponding to function of actions of *LR*(1)-analyzer of language *L(G)*, where *S* is a set of the analyzer conditions, *V* – as before is the primary (terminal) alphabet; $\Psi(s, v) = N$ if the symbol *v* is inadmissible in *s* condition, and $\Psi(s, v) = Y$ in the other case (the analyzer action - transposition, convolution or admission). We will define the code *K* as follows:

$$K: S \times V \rightarrow \{0,1\}^*, \tag{4}$$

at that the function *K* is defined only on pairs $<s, v>$, for which $\Psi(s, v)=Y$. The binary sequence *K*(*s*, *v*) represents a symbol code *v* in a state of *LR*(1)-analyzer *s*. Automata coding of sentences of *LR*(1)-

language *L(G)* is defined as follows:

$$A_K(w) = K(s_{i_1}, v_{j_1}) \dots K(s_{i_m}, v_{j_m}), \qquad (5)$$

where $w = v_{j_1} \dots v_{j_m}$, a $s_{i_1}, \dots, s_{i_m}$ is the sequence of underwent in the process of parsing the word *w* conditions, in which the symbols $v_{j_1} \dots v_{j_m}$ are analyzed for the first time, accordingly. Here $i_1 = 0$.

The definition (5) straightforwardly assigns an algorithm of automata coding of words $w \in L(G)$. In its basis the algorithm of *LR*(1)-analyzer of language *L(G)* lies, in the course of which work, at the moment of passage to the analysis of the recurrent symbol *v* in the condition of *s*, the code combination *K(s, v)* is written in the output string.

For research of a problem of unambiguity of binary sequences decoding of the form $A_K(w)$, let us define $|S|$ of so-called partial codes $K_{(i)}: V_{(i)} \to \{0,1\}^*$, where $V_{(i)} = \{v \mid \Psi(s_i, v) = Y\}$ is a set of symbols, admitted in the condition s_i, and for any $v \in V_i$ $K_i(v) = K(s_i, v)$. Naturally, partial codes are defined only for those conditions, for which $V_{(i)} \neq \varnothing$. A set of such conditions, named encoding conditions, are designated lower through *S'*. The sufficient condition of context code decipherability relating to *LR*(1)-language is defined by the following theorem.

T h e o r e m 2. *The automata code* $K: S \times V \to \{0,1\}^*$ *is decipherable over the language L(G), where G is LR*(1)*-grammar, if all partial codes* $K_{(i)}: V_{(i)} \to \{0,1\}^*$, $s_i \in S'$, *a r e d e c i p h e r a b l e o v e r t h e l a n g u a g e* V^*.

P r o o f . Let we have an initial condition $s_0 \in S$, when the parse's stack is empty, and there is binary sequence $z \in \{0,1\}^*$ at the input of decoding device. As the code $K_{(0)}$ we decipher in relation to V^*, the first symbol $v_{i_1} \in V$ is univalently decoded. Thus $K_{(0)}(v_{i_1}) = z_1$, where z_1 is prefix *z*. As *L(G)* is *LR*(1)-language, the presence of v_{j_1} symbol is enough to define univalently the new state of s_{i_2}, in which the analysis of the next symbol of a string of the alphabet *V* is necessary. Let now we have some random state of s_{i_2}, and at the input of decoding device we have the rest part of *z*. While reasoning analogous, by induction, we come to the conclusion that whatever state was the *LR*(1)-analyzer, the recurrent symbol $v_{j_k} \in V$ and the state of $s_{i_{k+1}} \in S$ are defined univalently. Thus, as for any *i* $K_i(v) = K(s_i, v)$, the result of decoding of binary sequence $A_K(w)$ is the word $w \in L(G)$, which was to be proved.■

The technique of the theorem 2 proof, in essence, defines algorithm of decoding with reference to the automata codes, satisfying to the formulated criterion of decipherability. The key feature of this algorithm is that on every step of *LR*(1)-analyzer's work, when the analysis of the recurrent symbol $v_{i_k} \in V$ is necessary, the decoding of prefix of the rest part of input binary sequence is performed with the help of the code $K_{(i_k)}: V_{(i_k)} \to \{0,1\}^*$, where s_{i_k} is the analyzer's current condition.

Owing to existence of the described algorithm, the automata codes, satisfying to the statement of theorem 2, are decipherable with a finite delay over the language *L(G)*.

E x a m p l e 2. We will illustrate the decoding logic in a class of automata codes with reference to the automata language *L*, described by a finite automata represented in fig. 1.

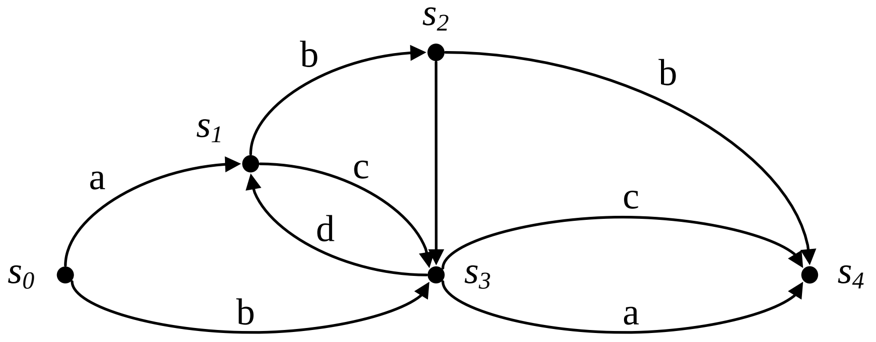

Fig. 1. Finite automata, describing the language *L*,
the words of which are coded by automata code *K*

The following decipherable over the language *L* automata code corresponds to this automata:

State s	Symbol v	Code $K(s, v)$
s_0	a	0
	b	1
s_1	b	0
	c	1
s_2	b	0
	d	1
s_3	a	0
	c	10
	d	11

In a functional notation:
$K(s_0, \text{a}) = 0$, $K(s_0, \text{b}) = 1$,
$K(s_1, \text{b}) = 0$, $K(s_1, \text{c}) = 1$,
$K(s_2, \text{b}) = 0$, $K(s_2, \text{d}) = 1$,
$K(s_3, \text{a}) = 0$, $K(s_3, \text{c}) = 10$, $K(s_3, \text{d}) = 11$. ■
Both considered classes of codings are based on forming of codes of words
$w \in L(G)$ in the course of their symbol-by-symbol parsing. We will notice that each such word is univalently defined by its derivation tree in the grammar *G*. This allows to code their derivation trees instead of words. Because the derivation tree of the formalized message from the substantial point of view is the representation of its syntactic construction (structure), this kind of coding is called **structural**.
Let us pass to formal description of **structural codes** from the point of view of coding theory.
Let us put in correspondence to generating rules of unambiguous grammar *G* binary code combinations, i.e. define the function

$$K: R \rightarrow \{0,1\}^*. \qquad (6)$$

Let

$$x_0 \xrightarrow{(k_0, r_{i_0})} x_1 \xrightarrow{(k_1, r_{i_1})} \ldots \xrightarrow{(k_{n-1}, r_{i_{n-1}})} x_n \qquad (7)$$

is the derivation of word *w* in grammar *G*, where $x_0 = a_0$, $x_n = w$, and each sentential form $x_j \in (V \cup V_N)^*$ ($j = 1, \ldots, n$) is obtained from sentential form x_{j-1} by using of a generating rule $r_{i_j} \in R$ to k_j (to the left) nonterminal SF x_{j-1} (it is supposed that the mentioned nonterminal coincides with the left part of a rule).

Two variants of the structural codings, corresponding to the fixed or variable order of the world derivation tree parsing, are possible.

The fixed order is defined by some prescribed function $D\colon (V \cup V_N)^* \to \mathbf{N}$, where $\mathbf{N}$ is a set of positive integral numbers. This function puts in correspondence to each string in the alphabet $V \cup V_N$ the sequential number of one of non-terminals, entering it (at their indexing in string from left to right), which then is chosen for application a generating rule to it, continuing the conclusion.

In this connection (7) is written down in the form

$$x_0 \xrightarrow{(D(x_0), r_{i_0})} x_1 \xrightarrow{(D(x_1), r_{i_1})} \ldots \xrightarrow{(D(x_{n-1}), r_{i_{n-1}})} x_n \qquad (8)$$

In particular, having defined $D(x) = 1$ for any *x*, we will get canonical parsing, and having defined $D(x) = N_x$, where N_x is the number of non-terminals in the string *x*, is right-hand top-down parsing, and etc. We will lay emphasis that, while speaking of the sequential number of a nonterminals in a string, we mean “nonterminal-position” (for example, at $D(aaAbABC) = 3$, the third nonterminal is *B*, but not *C*).

E x a m p l e 3. Let us consider a unambiguous context-free grammar $G = \langle V, V_N, a_0, R\rangle$ from an example 2 and the function K: $R = \{0,1\}^*$ such that

[r_1] $K(\mathrm{A} \to \mathrm{aB}) = 0$,
[r_2] $K(\mathrm{B} \to \mathrm{cB}) = 0$,
[r_3] $K(\mathrm{B} \to \mathrm{b}) = 1$.

In this connection the word acb $\in L(G)$ can be generated as a result of the following derivation:

$\mathrm{A} \xrightarrow{(1,1)} \mathrm{aB} \xrightarrow{(1,2)} \mathrm{acB} \xrightarrow{(1,3)} \mathrm{acb}$. ■

The structural code with the fixed order of parsing of derivation trees of the encoded words (for brevity SCF) we will name the pair $\langle K, D\rangle$, which defines the coding of a word *w* with the conclusion (8) as follows:

$$A_{K,D}(w) = K(r_{i_1}) \ldots K(r_{i_{n-1}}). \qquad (9)$$

From (9) it follows that word coding of $w \in L(G)$ by the SCF $\langle K, D\rangle$ is reduced to parsing of derivation tree *w*, prescribed by function *D*, on each *j* step of which the binary line $K(r_{i_1})$ is written to the right of the earlier received part of this word code.

E x a m p l e 4. Let us consider a unambiguous context-free grammar *G* and a function *K* from an example 3. We will define a structural code with the fixed order of parsing of derivation trees of words in the form of a pair $\langle K, D\rangle$, where $D(x) = 1$ for any $x \in (V \cup V_N)^* - V_N$. Then the code of the word acb $\in L(G)$ will be $A_{K,D}(acb) = K(r_1) \cdot K(r_2) \cdot K(r_3) = 001$.■

With a variable order of parsing, the decision about the number of non-terminal, to which the rule r_{i_j} is applied, it is applied directly after the generation of SF x_{i_j} ; so the sequence of numbers $\langle k_0, k_1, \ldots, k_{n-1}\rangle$ should be known to the decoder, and for this purpose in its turn, it should be coded and included in a code of word. Let the numbers $\langle k_0, \ldots, k_{n-1}\rangle$ be coded by the code E: $N \to \{0,1\}^*$.

The structural code with a variable order of parsing of derivation trees of words (for brevity SCV) we will name the pair <*K*, *E*>, which defines coding of the word *w* with the derivation (7) as follows:

$$A_{K,E}(w) = E(k_0)K(r_{i_0}) \ldots E(k_{n-1})K(r_{i_{n-1}}) \qquad (10)$$

As seen, coding of the word $w \in L(G)$ by <*K*, *E*> consists in fulfillment of semisteps of the number k_j coding of recurrent non-terminal, chosen for continuation of the derivation and coding of the generating rule r_{i_j}, which determines an alternative of this non-terminal. Both code combinations are written to the right of the formed on previous steps part of the code of *w*.

E x a m p l e 5. Let us consider the unambiguous context-free grammar *G* of example 3. We will define that numbers of non-terminals (non-terminal-positions) in sentential forms, generated by derivation, are coded by two-bit numbers (thus a supposition is used that the greatest value of the number of non-terminal-position in the course of coding is 3, or in binary notation 11). Then the word code acb $\in L(G)$ will be the sequence

$A_{K,E}(\text{acb}) = 01 \cdot K(\text{A} \rightarrow \text{aB}) \cdot 01 \cdot K(\text{B} \rightarrow \text{cB}) \cdot 01 \cdot K(\text{B} \rightarrow \text{b}) = 010010011.$■

The form SCV (structural code with variable order), which corresponds to bialternative grammars, is more convenient for practical applications. Thus a derivation tree of the word $w \in L(G)$ is univalently defined by sequence $<k_0, b_0, k_1, b_1, ..., k_{n-1}, b_{n-l}>$, where $k_j \in N$, $b_j \in \{0,1\}$. Transferring b_j to signs of numbers k_j (0 corresponds to plus, 1 corresponds to minus) and designating the numbers received by such a way through k_j', we will receive the sequence $<k_0', ..., k_{n-1}'>$, also univalently defining a derivation tree of *w*. As a result, SCV is reduced, in essence, to forming of integral numbers sequences.

E x a m p l e 6. The sequence of integral numbers coding a derivation tree of a word acb $\in L(G)$ from an example 5, looks like <1,1,-1> (the last integral number is negative because $K(\text{B} \rightarrow \text{b}) = 1$). ■

It is necessary to notice that in general case the grammar *G* can be ambiguous due to the word $w \in L(G)$ can have several codes by number of derivation trees *w* in the grammar *G*. That is why the structural coding in its any version represents many-valued mapping from *L(G)* in {0,1} , that is the function $A_K : L(G) \rightarrow 2^{\{0,1\}^*}$.

For simplicity, we will consider lower only unambiguous grammars, what from the practical point of view is not at all essential restriction.

Let we have function $K: R \rightarrow \{0,1\}^*$ and the grammar $G = <V, V_N, a_0, R>$. Let us define the grammar $G^{CK} = <V^{CK}, V_N^{CK}, a_0^{CK}, R^{CK}>$ as follows:

$$\begin{aligned} &V^{CK} = \{0,1\}, \\ &V_N^{CK} = V \cup V_N, \qquad (11) \\ &R^{CK} = \left(\bigcup_{a \rightarrow \beta \in R} \{a \rightarrow z\beta\} \right) \cup \left(\bigcup_{v \in V} \{v \rightarrow \Delta\} \right) \\ &a_0^{CK} = a_0, \end{aligned}$$

where $z = K(a \rightarrow \beta)$, Δ – empty string and, as everywhere before $V \cap \{0,1\} = \varnothing$. Besides, we will fix a canonical order of derivation trees parsing. The decipherability criteria of structural codes with this parsing order is defined by the following theorem, which is similar on sense to the theorem 1.

T h e o r e m 3. *SCF <K, D>*, where $K: R \rightarrow \{0,1\}^*$ and $(\forall x \in (V \cup V_N)^* - V^*)\ D(x) = 1$, *is decipherable over the language L(G), where G is context-free grammar, if the grammar G^{CK} is unambiguous.*

P r o o f . On account of theorem for each word $w' \in L(G^{CK})$ there exists the only one canonical derivation $a_0^{CK} \overset{*}{\underset{G^{CK}}{\Rightarrow}} w'$ and, in such a way, the only one sentential form $x = y_1 v_{i_1} y_2 \ldots y_q v_{i_j} y_{q+1}$,

where $v_{i_j} \in V$, $y_j \in \{0,1\}^*$. Thus $w' = y_1 y_2 \ldots y_q y_{q+1}$. From here it follows the existence of the only one word $w = v_{i_1} \ldots v_{i_q}$ such that $w' = A_K(w)$ and, by this means, the code $<K, D>$ decipherable over $L(G)$.■
As it is easy to see, the statement of the theorem 3 is correct for any SCF and SCV.
As well as in case of alphabetic codes, the universal criteria of structural codes decipherability over context-free languages is non-constructive because of algorithmic indecidability of problem of context-free grammars unambiguity recognition. That is why it is necessary to find some sufficient condition of decipherability for practical applications.
Let

$$R = \{a_0 \to \beta^0_1, \ldots, a_0 \to \beta^0_{n_0}, \ldots$$
$$\ldots\ a_i \to \beta^i_1, \ldots, a_i \to \beta^i_{n_0}, \ldots, \ldots \qquad (12)$$
$$\ldots\ a_m \to \beta^m_1, \ldots, a_m \to \beta^m_{n_m}\}.$$

For each group of generating rules $R_{(i)} = \{a \to \beta^i_1, \ldots, a \to \beta^i_{n_i}\}$ with an identical left part a$_i$ we will define a partial code $K_{(i)}$: $R_{(i)} \to \{0,1\}^*$ in such a manner that $K_{(i)}(r) = K(r)$ for any $r \in R_{(i)}$.

T h e o r e m 4. *SCF $<K, D>$, where $K: R \to \{0,1\}^*$ and $(\forall x \in (V \cup V_N)^* - V^*)\ D(x) = 1$, is decipherable over the language $L(G)$, where G is context-free grammar, if all codes $K_{(i)}$: $R_{(i)} \to \{0,1\}^*$ $(i = 0, \ldots, m)$ are decipherable over the language R.*
P r o o f . Let us show that on condition of partial codes $K_{(i)}$ decipherability over R, the grammar G^{CK} is $LL(k)$-grammar. We will remind that the context-free grammar belongs to the class $LL(k)$, if for any sentential form wax, where $w \in V^*$, $a \in V_N$, $x \in (V \cup V_N)^*$, and any word $w\ y_1 y_2$, where $y_1 \in V^k$, $y_2 \in V^*$, analyzed from top to down string y_1 with the length of k symbols, unambiguously defines an alternative of non-terminal a, which should be applied to analysis continuation.
In grammar G^{CK} the rules $v \to \Delta$ have the only one alternative, that allows us to limit reviewing groups of rules

$$R_{(i)}^{CK} = \{a_i \to z^i_1\, \beta^i_1, \ldots, a_i \to z^i_{n_i}\, \beta^i_{n_i}\}\ (i = 0, \ldots, m). \qquad (13)$$

Let a_i is non-terminal, which alternative should be chosen for continuation of top-down parsing. Set by a theorem statement, the decipherability of code K relative to V defines the possibility of univalent decoding of "symbol" of "alphabet" R, that is from positions of syntax analysis, a univalent choice of concrete alternative $a_i \to z^i_j\, \beta^i_j$. Actually the length of a prefix of not decoded part of entering binary sequence w', the knowledge of which is enough for definition of demanded alternative, is the value $l = \max\{\,|\,z^i_1\,|, \ldots, |\,z^i_{n_i}\,|\,\}$. For the whole grammar G^{CK} the value of minimum prefix length, sufficient for the determined analysis, is the value $k = \max\{l_0, \ldots, l_m\}$. Hence, the string of bits of length k, with which not analyzed part of entering word w' begins, univalently defines an alternative of any non-terminal, necessary for the analysis continuation. Hence, according to definition given above, G^{CK} is $LL(k)$-grammar. By reason of the fact that any $LL(k)$- grammar is univalent, G^{CK} is a univalent grammar and, as it follows from theorem 3, the code $<K, D>$ is decipherable over the language $L(G)$.■
As well as earlier, the statement of the given theorem is easily generalized on any SCF and SCV. As a basis of decoding algorithm there can be put any (not necessarily top-down) algorithm of the context-free languages words parsing, added with the operations, storing of a derivation tree in memory of the decoding device of the analyzed binary sequence. After construction of the mentioned tree all terminal nodes with symbols 0,1 and Δ and all edges, leading to these nodes, are removed from it. It is easy to

see that the sequence of the alphabet V symbols, generated as a result of left-to-right parsing of terminal nodes of the received tree, represents the coded word of language $L(G)$.

E x a m p l e 7. As seen, SCF $<K, D>$, where the function K is defined in example 3, and $D(x) = 1$ for any sentential form x of context-free grammar G from this example, is decipherable over the language $L(G)$, as far as the codes

$K_{(1)}(A \to aB) = 0$,

$K_{(2)}(B \to cB) = 0$, $K_{(2)}(B \to b) = 1$.

are decipherable over the language R.■

Owing to existence of algorithm of the determined parsing of words $A_K(w)$ on the basis of $LL(k)$-analyzer, any structural code $<K, D>$, decipherable over the language $L(G)$, is decipherable with a finite delay over $L(G)$.

3. Construction of Codes, Effective Over the Stochastic Context-Free Languages

Let us use the representation of a set of coded messages in the form of stochastic language (in sense of [3])

$$L(G_p) = \{<w,q> \mid a_0 \overset{*}{\underset{G}{\Rightarrow}} w \ \& \ w \in V^* \ \& \ q = p(w)\}, \tag{14}$$

where G_p is stochastic context-free grammar, describing this language, a_0 is its axiom, and $p(w) \in [0,1]$ is the probability of the word w derivation in the mentioned grammar. Without limiting the generality, let us assume that G_p is correct, that is

$$\sum_{w \in L(G)} p(w) = 1, \tag{15}$$

where $L(G)$ is characteristic language of stochastic language $L(G_p)$, and G is characteristic grammar:

$$L(G) = \{w \mid a_0 \overset{*}{\underset{G}{\Rightarrow}} w \ \& \ w \in V^*\} = \{w \mid (\exists q)<w, q> \in L(G_p)\}. \tag{16}$$

If G is unambiguous, the value $p(w)$ is equal to product of probabilities of application of the generating rules, used in the derivation $a_0 \overset{*}{\underset{G}{\Rightarrow}} w$. Within the framework the considered application $p(w)$, from the conceptual point of view, represents the probability of message w incoming to the coding device input.

Integral efficiency index of messages coding is the average time T_0, spent on transmission of the message from a source to recipient, which is defined by the following way:

$$T_0 = \sum_{w \in L(G)} p(w) \cdot T(w), \tag{17}$$

where $T(w)$ for a case of an ideal data transmission channel is expressed in the form of a sum

$$T(w) = t_c \cdot | w | + t_o \cdot | A_K(w) | + t_d \cdot | A_K(w) |. \tag{18}$$

In (18) expression t_c represents an average time, spent on coding of one symbol of the message**,** but t_o and t_d represent an average time, which is being spent accordingly on transmission and decoding of one bit of the message code. While t_o is the characteristics of channel terminating equipment, t_c and t_d at the specified effectiveness of coding and decoding means are completely defined by peculiarities of a class of codes to which the applied code K belongs.
Through $| w |^*$ we will designate mathematical expectation of length of the language $L(G_p)$ word, and through $| A_K(w) |^*$ – mathematical expectation of length of this word code:

$$| w |^* = \sum_{w \in L(G)} p(w) \cdot | w |, \qquad (19)$$

$$| A_K(w) |^* = \sum_{w \in L(G)} p(w) \cdot | A_K(w) |. \qquad (20)$$

On account of (18)-(20), we will write (17) as follows:

$$T_0 = t_c \cdot | w |^* + (t_o + t_d) \cdot | w |^* \cdot C(G_p, K), \qquad (21)$$

where

$$C(G_p, K) = \frac{|A_K(w)|^*}{| w |^*} . \qquad (22)$$

The value $C(G_p, K)$ represents an average number of bits, spent on coding of one symbol of a word of the language $L(G)$ by means of code K, and with account of existing terminology of coding theory [1] it is called hereafter as cost of the code K over the stochastic language $L(G_p)$. Thereafter everywhere, where it does not lead to misunderstanding, we will name $C(G_p,K)$ for brevity simply the cost of code K.
Class of codes over stochastic language $L(G_p)$ we will name any of the following codes assemblage, defined in item 2:
– alphabetic codes $K: V \to \{0,1\}^*$;
– automata codes $K: S \times V \to \{0,1\}^*$;
– structural codes with the fixed order of parsing of derivation trees of messages $<K, D>$, where K: $R \to \{0,1\}^*$, $D: V \cup V_N) \to N$, with the same D;
– structural codes with a variable order of parsing of derivation trees of messages $<K, D>$, where K: $R \to \{0,1\}^*$, with the same E.
Thus, classes of automata codes are admissible only in a case when G is LR(l)-grammar.
Code K_0, belonging to some class of codes over $L(G_p)$ and decipherable with a final delay over the language $L(G)$, we will name nonredundant over stochastic language $L(G_p)$, if for any other code K, decipherable with a final delay over the language $L(G)$, of the same class

$$C(G_p, K) \geq C(G_p, K_0). \qquad (23)$$

Let us remind that in the theory of coding the cost of code $K: V \to \{0,1\}^*$, corresponding to distribution $p: V \to \{0,1\}$ of probabilities of occurrence of the alphabet V symbols in messages being coded and designated hereafter through $C_p(K)$, is defined as mathematical expectation of length of a code combination:

$$C_p(K) = \sum_{v \in V} p(v) \cdot | K(v) |. \qquad (24)$$

The least cost in sense (24) possess codes, which are synthesized by Huffman method [2].

Let $\bar{p} : V \to \{0,1\}$ is distribution of probabilities of symbols $v \in V$ occurrence in words of the language $L(G_p)$, and $K^H: V \to \{0,1\}^*$ is Huffman code, corresponding to this distribution. Code K, decipherable with a finite delay over the language $L(G)$, we will name effective over stochastic language $L(G_p)$, if

$$C(G_p, K) \geq C_p(K^H). \tag{25}$$

Let us pass directly to consideration of construction methods of codes, effective over stochastic context-free languages. We will begin with the construction of **effective alphabetic codes** of a form $K: V \to \{0,1\}^*$.

Let we have Huffman code $K^H: V \to \{0,1\}^*$, constructed on probabilities distribution $\bar{p} : V \to \{0,1\}$, corresponding to the language $L(G_p)$. The cost of this code $C_{\bar{p}}(K^H)$ can be accepted as an upper bound of costs of required alphabetic codes, effective over $L(G_p)$. The presence of such bound allows to use the following algorithm for construction of irredundant code.

Let be given the number k and for each ensemble of code combinations lengths $l_1, \dots, l_n$, where $n = | V |$, such that

$$\sum_{i=1}^{m} \bar{p}(v_{i_{l_v}}) \cdot l_i < C_{\bar{p}}(K^H), \tag{26}$$

we search through all possible codes $K: V \to \{0,1\}^*$ such that $| K(v_i) | = l_i \ (i=1, \dots, n)$. For each of these codes we define, whether the grammar G^K is $LR(k)$-grammar. As a result it would be found a code K_0, possessing the least cost of all codes, which satisfy to criteria of decipherability [7] for the preset k.
It is clear that, because of $C_p(K^H)$ assignment and non-negativeness of l_i the number of lengths gatherings of code combinations is finite. For this reason, and also owing to algorithmic decidability of recognition of grammars membership to class $LR(k)$ [4], code will be found for a finite number of steps N. However, computational complexity of the described algorithm is extremely high and is estimated by the value

$$O(N) = O(N_K) \cdot \sum_{l_1=0}^{m_1} \dots \sum_{l_n=0}^{m_n} 2^{(\sum_{i=1}^{n} l_i)}, \tag{27}$$

where $O(N_K)$ is computational complexity of check of grammar G^K membership to class $LR(k)$, $m_1, \dots, m_n$ are maximum values of code combinations lengths ($m_i = [C_p(K^H) / \bar{p}(v_i)]$),$2^{li}$ is the number of various code combinations of the length l_i.
By virtue of the fact that synthesis of irredundant alphabetic codes by a method of simple searching is generally almost impossible, we will be limited to reviewing this problem for a case of stochastic $LR(1)$-grammars.
Let we have $LR(1)$-grammar G and corresponding to it a set of coding states of S' and alphabets $V_{(i)}$, $s_i. \in S'$, specified in item 2. For every $v \in V$ we will construct a set $S_{(v)} \subseteq S'$ of states, in which this symbol can be coded:

$$S_{(v)} = \{s_i \in S' \ \& \ v \in V_{(i)} \}. \tag{28}$$

T h e o r e m 5. *Let $V = V_{(1)} \cup \ldots \cup V_{(t)}$, where $1 \leq t \leq |V|$, a $V(k) \subseteq V$ is a set of symbols, to which the same combination of code $K: V \rightarrow \{0,1\}^*$, corresponds, having the following properties:*

1) *for any v_i^k u v_j^k, belonging to the set $V_{(k)}$, the condition $S_{(v_i^k)} \cap S_{(v_j^k)} = \varnothing$ is fulfilled;*

2) *the code K': $\{v_1^1, \ldots, v_1^t\} \rightarrow \{0,1\}^*$ is such that for all $k = 1, \ldots, t$ $K(v_1^k) = \ldots = K(v_n^k)$, $n_k = |V_{(k)}|$, is decipherable with finite delay over the language V^*.*

Then the code K is deciphered over the LR(1)-language L(G).

P r o o f . We will consider the process of decoding in the form of an aggregation of two sub-processes: decoding of the prefix of the remaining part of entering binary sequence and handling of the received symbol $v \in V$ according to algorithm of *LR*(1)-analyzer of language *L(G).* Let, as a result of performance of several steps, the analyzer is in a state $s \in S$ in which the analysis of a following symbol is necessary. As the code K' is deciphered with a finite delay over the language V^*, as a result of prefix decoding the set $V_{(k)} \subseteq V(k = 1, \ldots, q)$, will be uniquely defined, one of which elements is decoded. As for any two symbols $V_{(k)}$ sets of states in which they are admissible, are not intersected, the state *s* can belong to only one of these sets. This allows to specify univalently the code of what symbol $v \in V_{(k)}$ is the decoded prefix. As the cited verbal proofs are correct for any state of $s \in S$, the code *K*, satisfying to a theorem statement, is decoded over *LR*(1)-language of *L(G)*.■

The criteria of the proved theorem is constructive. There is the elementary algorithm of decipherability recognition, which consists of looking up for each pair of symbols such that $K(v_i) = K(v_j)$, of *i* and *j* table columns of actions of *LR*(1)-analyzer of the language *L(G)*.

The existence of at least one line with not empties *i* and *j* elements testifies of the code indecipherability.

E x a m p l e 8. Let we have the following simplified table of *LR*(1)- analyzer of some *LR*(1)-language *L(G)* with the terminal alphabet {a, b, c}, in which the symbol "+" designates an admissibility of read out symbol of an analyzed line in corresponding state of *LR*(1)-analyzer:

S	a	b	c
s_0	+		+
s_1	+	+	
s_2		+	+
s_3			+

Then sets of states of *LR*(1)-analyzer, in which symbols a, b, c, are admitted, are like that:
$S_{(a)} = \{s_0, s_1\}$,
$S_{(b)} = \{s_1, s_2\}$,
$S_{(c)} = \{s_0, s_2, s_3\}$.

Let thus such alphabetic code *K* takes place, that *K*(a)=0, *K*(b)=0, *K*(c)=1. As *K*(a)=*K*(b)=0, a $S_{(a)} \cap S_{(b)} = \{s_1\} \neq \{\varnothing\}$, code *K* is indecipherable over the language *L(G)*.

At the same time, in case we have *LR*(1)-language *L(G)*, the table *LR*(1)- analyzer of which looks like

S	a	b	c
s_0	+		+
s_1	+		
s_2		+	+
s_3		+	

The alphabetic code K is decipherable over the language $L(G)$, as
$S_{(a)} = \{s_0, s_1\}$,
$S_{(b)} = \{s_2, s_3\}$
and $S_{(a)} \cap S_{(b)} = \{\varnothing\}$.■
However, as it is easy to see, for construction of an irredundant code of the form $K\colon V \to \{0,1\}^*$ on the basis of criterion of the theorem 5 the sorting of all possible partitions of V into non-overlapping sets is necessary. The amount of similar partitions of $N(V)$ is defined by the following expression known from [9]:

$$N(V) = \frac{1}{e}\sum_{i=0}^{\infty}\frac{i^n}{i!} = \sum_{i=0}^{n}\sigma(n,i), \qquad (29)$$

where $n = |V|$, a $\sigma(n, i)$ are the second kind Stirling numbers. For practical V the value of $N(V)$ is inadmissible high, which excludes the possibility of code synthesis by a simple sorting of partitions. In this connection, we will consider the offered method of synthesis, which possesses comprehensible computational complexity and basically followes the methodology of similar problems solution, developed in the theory anti-prefixability [10-12].
Let us construct a graph $\Gamma \subseteq V \times V$ of left-context distinction of the alphabet symbols of V. Two nodes of this graph v_i and v_j are joined by an edge in that and only in that case, if there is the state of $LR(1)$-analyzer of the language $L(G)$, in which both symbols can be coded:

$$\Gamma = \{<v_i, v_j> \mid S_{(v_i)} \cap S_{(v_j)} \neq \varnothing\}. \qquad (30)$$

We will colour the nodes of the graph Γ with the minimum number of paints, that is we will construct the function $\pi\colon V \to \{1, \chi(\Gamma)\}$, where $\chi(\Gamma)$ is chromatic number of this graph. As nodes of Γ are correctly colored, then for any v_i, v_j, connected by the edge, the statement $\pi(v_i) \neq \pi(v_j)$ is fulfilled. Because of this, symbols of the alphabet V to which the same colour correspond, cannot be coded in neither one state $s \in S$. Thereby we have received a partition on minimum number of subsets in the sense of a condition of the theorem 5. The further reduction of this number will lead to condition violation. We will exclude all isolated nodes from the received graph. Each of them corresponds to the symbol $v \in V$, coded in such states in which any other symbol is not coded. Empty code combinations correspond to all such symbols. We will notice incidentally that the amount of the isolated nodes does not influence on the chromatic number of graph Γ and, further, on the cost of a synthesized code.
E x a m p l e 9. Let we have the following simplified table of $LR(1)$-analyzer operations of some $LR(1)$-language $L(G)$ with the terminal alphabet {a, b, c, d, e}:

S	a	b	c	d	e
s_0	+		+		
s_1		+		+	+
s_2	+			+	
s_3		+	+	+	
s_4	+				
s_5		+			+
s_6	+			+	
s_7		+			+

The graph $\Gamma \subseteq V \times V$ of left-context distinction of alphabet V symbols, corresponding to this table, has the following form:

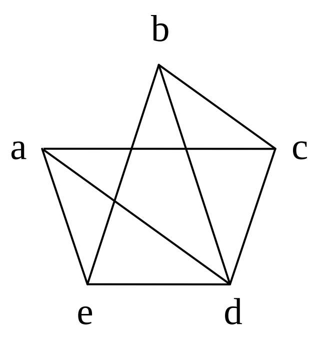

Fig.2. The graph of left-context distinction

The following correct node coloring can correspond to the given graph:
π(a)=1, π(b)=1, π(c)=2, π(d)=3, π(e)=2.
In its turn, the following code K can correspond to the given coloring. This code is decipherable over the language $L(G)$:
K(a)=0, K(b)=0, K(c)=1, K(d)=01, K(e)=1.■
Let us construct Huffman code of the form K': $\{1, \ldots, \chi(\Gamma)\}\rightarrow\{0,1\}^*$, corresponding to the probabilities distribution of p': $\{1, \ldots, \chi(\Gamma)\}\rightarrow[0,1]$.
In the general case some variants of distribution p' are possible owing to feasibility of some nonisomorphic ways of coloring. From all Huffman codes, corresponding to different variants of distribution p' we will choose the one, which has the minimum cost. This K'code univalently defines the required code K_0: $V \rightarrow \{0,1\}^*$:

$$(\forall i \in \{1, ..., \chi(\Gamma)\}) (\forall v \in V_{(i)}) (| V_{(i)} | > 1 \Rightarrow$$
$$\Rightarrow K_0(v) = K'(v) \ \&\ (| V_{(i)} | = 1 \Rightarrow K_0(v) = \Delta), \qquad (31)$$

where Δ is an empty word.

Theorem 6. *Costs of codes K and K' are equal.*

Proof. Probabilities distribution of $\overline{p}: V \rightarrow [0,1]$ and p': $\{1, ..., \chi(\Gamma)\} \rightarrow [0,1]$ are connected by an obvious relation:

$$p'(i) = \sum_{v\in V_{(i)}} \overline{p}(v). \qquad (32)$$

Let us record the expression for code cost K_0 in the form of [1]:

$$C_p(K_0) = \sum_{v\in V} \overline{p}(v)\cdot|K_0(v)| = \sum_{i=1}^{\chi(\Gamma)} \sum_{v\in V_{(i)}} \overline{p}(v)\cdot|K_0(v)|. \qquad (33)$$

By virtue of the fact that according to (31) for all $v \in V_{(i)}$ $K_0(v)=K'(i)$, the last expression is converted to the following form:

$$C_p(K_0) = \sum_{v\in V}^{\chi(\Gamma)} \left(|K'(i)|\cdot \sum_{v\in V} p(v) \right) = \sum_{i=1}^{\chi(\Gamma)} p'(i)\cdot|K'(i)|. \qquad (34)$$

The last by definition is $C_p(K')$, which was to be proved. ■

The distribution $\overline{p}: V \rightarrow [0,1]$, necessary for synthesis of an irredundant code, can be received either empirically, or (in the absence of such possibility) directly from grammar G_p.

In the last case for calculation of probability $\overline{p}(v)$ it is enough to present it in the form of relation of mathematical expectation of number of entrances of the symbol $v \in V$ in the word $w \in L(G)$ to mathematical expectation of this word $|w|_{cp}$ length . The specified values can be received, proceeding from the following base relations, known from [13]. According to [13], mathematical expectation of the length of the word x, derived from nonterminal $a_i \in V_N$ in correlated stochastic grammar G_p for arbitrary number of derivation steps, represents the value i of vector component

$$Y=((E-Q\times C)^{-1}\times W)\times I. \qquad (35)$$

Here E is matrix of dimension $|V_N| \times |V_N|$, such that $l_{ij} = 1$; Q is matrix of dimension $|V_N| \times |R|$, where q_{ij} is the probability of the j rule application from R at generation from nonterminal $a_i \in V$ (for rules of the form $a_0 \xrightarrow{p_{ik}} \beta^i_k$ $q_{ij}=p_{ik}$, for all others $q_{ij}=0$); W is matrix of dimension $|V_N| \times |V|$, where w_{ij} is mathematical expectation of number of entrances of a symbol $v_j \in V$ in the word $x' \in V^*$, derived from a_i; C is matrix of dimension $|R|$ lines and $|V_N|$ of columns, where c_{ij} is the number of non-terminal entrances $a_i \in V_N$ into the right part of i rule from R; I is vector of dimension $|V_N|$, the values of all components of which are 1.
Matrix W is calculated as scalar product $Q \times T$, where T is matrix of dimension $|R| \times |V|$, where t_{ij} is the number of entrances of symbol $v_j \in V$ in the right part of i rule of R.

By this means, $|w|_{cp}$ is the value of the first component of vector Y, and mathematical expectation of number of entrances of symbol $v_i \in V$ in the word $w \in L(G)$ is (1, i) element of matrix W.

That is why

$$p(v_i) = \frac{w_{ij}}{y_1}. \qquad (36)$$

Let us shortly discuss the peculiarities of coding-decoding by means of irredundant codes of the given class. Coding of words $w \in L(G)$ with the help of K_0 differs by nothing from the usual. At the heart of decoding algorithm lies the algorithm of LR(1)-analyzer of the language $L(G)$, in which additionally operations of colour decoding $i \in \{1, ..., \chi(\Gamma)\}$ and subsequent definition according to current condition $s \in S$ of a single admissible symbol of a set of $V_{(i)}$ are included. The specified operations are fulfilled on reaching each state, in which the following symbol of a coded word is analyzed for the first time.
For the synthesis *of automata code* K_0: $S \times V \rightarrow \{0,1\}^*$, irredundant over stochastic LR(1)-language of $L(G_p)$, it is enough for each coding state $s_i \in S'$ to construct Huffman code $K^{H}_{(i)}: V_{(i)} \rightarrow \{0,1\}^*$, corresponding to probabilities distribution p_i: $V_{(i)} \rightarrow [0,1]$. As it is easy to see, the code K_0: $S \times V \rightarrow \{0,1\}^*$ is such that $K_0(s_i, v) = K^{H}_{(i)}(v)$, for all coding s_i and all $v \in V_{(i)}$, is irredundant over $L(G)$.
For the synthesis *of a structural code* $<K_0, D>$, where K_0: $R \rightarrow \{0,1\}^*$, irrespective of the derivation order, assigned by the function D, it is enough to construct Huffman codes $K_{(i)}: R\ \{0,1\}^*$ corresponding to distributions p_i: $R_{(i)} \rightarrow$ [0,1]. The last are set directly in the form of probabilities of generating rules application $a_i \xrightarrow{p_{ij}} \beta^i_j$.

4. Effective Grammatical and Generalized-Prefix Codes Interrelations

Let $M_1^{L(G)}$ be a local model of the language $L(G)$ of depth 1 (in this connection any neighbourhood ε ∈ $M_1^{L(G)}$ is a subset of the alphabet V). There takes place

T h e o r e m 7.

$$\Gamma = \bigcup_{\varepsilon\in M_1^{L(G)}} \varepsilon\times\varepsilon . \qquad (37)$$

P r o o f . The right part of (37) is none other than algebraic representation of the antiprefixability graph in the sense of [11], which is being constructed by way of cliques joining (complete subgraphs ε × ε), corresponding to all possible neighbourhoods of depth 1.
Therefore, for any two symbols v_i and v_j such that the edge $<v_i, v_j>$ belongs to the specified graph, there is at least one left context w in the alphabet V, in which both of them are admissible:

$$\{wv_k\} \cdot V^* \cap L(G) \neq \varnothing, \qquad (38)$$

where $k = i, j$. The latter, in turn, is equivalent to existence of state s of $LR(1)$-analyzer of the language $L(G)$, which corresponds to some unreduced string x in the alphabet $V \cup V_N$ such that $x \overset{*}{\underset{G}{\Rightarrow}} w$. From here $s \in S_{(v_i)}$ and $s \in S_{(v_j)}$ simultaneously, in this connection $<v_i, v_j> \in \Gamma$. Inverse proposition (any edge of graph Γ belongs to antiprefixability graph) is proved similarly: the presence of state $s \in S_{(v_i)} \cap S_{(v_j)}$ is equivalent to existence of unreduced string $x \in (V \cup V_N)^*$, from which at least one string is derived $w \in V^*$ such that $\{wv_i\} \cdot V^* \cap L(G) \neq \varnothing$ and $\{wv_j\} \cdot V^* \cap L(G) \neq \varnothing$. The latter means that there exists neighbourhood ε, including both v_i and v, in this connection $\langle v_i, v_j\rangle \in \varepsilon \times \varepsilon$. By this means graphs of left-context distinction in sense of the present item 2 and antiprefixability in sense [11] coincide, which was to be proved.■
Coincidence of the considered graphs is equivalent to the coincidence of corresponding to them nonredundant *context-free and locally-prefix codes* (with an accuracy to distinctions in definition of nonredundness: in relation to the message $w \in L(G)$, as in [10-12], or in relation to the language, as in the present paragraph).

The interrelation between context-free and generalized-prefix codes is also obvious enough and it is defined by the following theorem in which the statement $M_l^{L(G)}$ means local model of the language $L(G)$of the depth l, and $X^l_{(i)}$ is a set of blocks of the length l, admissible in a state $s_i \in S$.

T h e o r e m 8.

$$M_l^{L(G)} = \bigcup_{s\in S} \{X^l_{(i)}\}. \qquad (39)$$

P r o o f . According to definition of local model of language of the depth l [10-12], $M_1^{L(G)}$ is a set of all neighbourhoods of words $w \in V^*$ of the depth l. In its turn, the neighbourhood $\varepsilon_l(w)$ is defined by the expression

$$\varepsilon_l(w) = \{z \mid |z| = l \ \& \ \{wz\}\cdot V^* \cap L(G) \neq \varnothing \vee |z| < l \ \& \ wz \in L(G)\} \qquad (40)$$

Taking into account that the length of a coded word always is multiple to l (if it is not so, it is appended from the right with a string of the form Δ^r, where r is a missing number of auxiliary symbols Δ), it is possible to consider that the case $| z | < l$ in the right part (40) is excluded. Then, it can be put in correspondence to each neighbourhood $\varepsilon_l(w)$ the unreduced left context – a string $x \in (V \cup V_N)^*$, such that $x \overset{*}{\Rightarrow} w$. In its turn, x univalently corresponds to some state $s_i \in S$. That is why $\varepsilon_i(w)$ is a set of blocks of the length l, admissible in the state s_i, that is $X^l_{(i)}$. By this means, some set $X^l_{(i)}$ corresponds to each set $\varepsilon_i(w)$. And by contrast, if there is $X^l_{(i)}$, then there exists at least one string $w \in V^*$, which is being derived from at least one unreduced string x $V \cap V_N$, corresponding to the state s_i. But then $X^l_{(i)}$ is the neighbourhood of the word w of the depth l, or some set $\varepsilon_i(w)$ univalently corresponds to each set $X^l_{(i)}$. From here follows the correctness of (39).■

Synthesis of an alphabetic code, irredundant in sense of $M_l^{L(G)}$, is reduced to construction of finite system of antiprefixability equations in the sense of [11], corresponding to this model, and finding of its incompressible solution. From conceptual point of view this code supposes symbols with identical code combinations which are not distinguished by left reduced context $x \in (V \cup V_N)^*$, but are distinguished by x and the limited right context in the alphabet $\{0,1\}$.

As to comparison of *irredundant automata* and *locally-prefix* codes, the basic distinction between them is included already in their definitions. If the locally-prefix code is based on the fixed assignment of code combinations to symbols and consequently it is constructed, proceeding from joining of cliques $\varepsilon \times \varepsilon$, so the automata coding thanks to a possibility of symbols codes modification from state to state allows to correlate a unique clique to the recurrent symbol of the message and corresponding to it partial alphabetic code $K^{\mathrm{H}}_{(i)}\colon V \rightarrow \{0,1\}^*$, adequate to the current state of $LR(1)$-analyzer, i.e. finally to the left context of a symbol. As generally $| \varepsilon | \leq | V |$, the cost of automata codes, as a rule, is essentially less the cost of alphabetic and, according to the theorem 7, locally-prefix codes.

In case of automata languages, the reduced left context for each of finite states of automata has the length 1 and represents non-terminal, corresponding to this state. Thus the partial code $K^{\mathrm{H}}_{(i)}\colon V \rightarrow \{0,1\}^*$, is constructed for the partial alphabet of symbols, admissible in a state s_i, i.e. sets of the marked arcs of the diagram of this automat, starting from node s_i.

In conclusion of this item 4, we will bring reasoning of generalized character.

For the classical theory of coding is characteristic the replacement of the source, generating the messages, by a source, generating symbols, probabilities of which occurrence in messages are mutually independent. At this assumption, the source entropy is the value

$$H(V) = -\sum_{v \in V} p(v) \cdot \log_2 p(v), \tag{41}$$

which is the lower border of classical codes cost. Grammatical interpretation, analyzed in the present work, developing ideas of the pioneer work of A.A.Markov [10], is oriented on coding of messages from sets $W \subseteq V^*$, the source entropy of which is the value

$$H(W) = -\sum_{w \in W} p(w) \cdot \log_2 p(w). \tag{42}$$

In common case $H(W) \leq H(V)$, that is the theoretic prerequisite of coding efficiency raising for similar sources. It is necessary to notice that the specified prerequisite is taken as a principle of some known heuristic methods of compression used in practice (see, for example, [8]). However, grammatical interpretation allows to realize this prerequisite on the basis of usage of the unified logic means for description of sets of coded messages – formal grammars – and by that, to ensure the necessary level of theoretical generality. From the practical point of view the language $L(G) \supseteq W$ is essentially more exact approximation of a set of coded messages, rather than V^*, that, in turn, allows to approach considerably the cost of synthesized effective codes to value $H(W)$. Conceptually it is explained by the circumstance that the considered methods of codes synthesis ensure the elimination not only statistical, but also logic redundancy of coding.

5. Block Decoding

The considered grammatical codes, as well as classical Huffman codes, are characterized by rather low decoding rate, implemented by means of bit-by-bit scanning of binary string. A traditional mode of increasing the decoding rate is transition to block scanning of the decoded string. As an example of this method implementation are so-called *VB*-codes [14] in which code combinations of equal length are compared to blocks of symbols of primary alphabet, having various lengths, but close probabilities of occurrence on a coding device input. However, *VB*-codes, as well as traditional Huffman codes, are synthesized proceeding from a possibility of coding of any words in the primary alphabet. In this connection their cost, being a little smaller in comparison with the cost of Huffman codes (thanks to the partial account of mutual correlation of symbols in coded words), nevertheless concedes to the cost of grammatical codes synthesized on exact combinatorial-probabilistic descriptions of sets of coded words (stochastic grammars). Therefore, rather perspective attempt of combination of virtues of grammatical codes and *VB*-codes – small cost and high rate of decoding – is represented.
The main idea lying at the heart of considered below approach to its solution, consists in construction and use of the automata implementing block decoding and corresponding to symbol-by-symbol coding automata, synthesized according to mentioned above stochastic grammars. The correspondence is understood in the sense that outputs of bit-by-bit and block decoding coincide. Let us consider basic elements of block decoding for a case *of automata codes, decipherable with a finite delay over LR*(1)-languages.
Let we have function of operations f: $S \times V \rightarrow A$ and function of transitions g: $S \times (V_N \cup V) \rightarrow S$ of LR(1)-analyzer of the language $L(G)$, and automata code also K: $S \times V \rightarrow \{0,1\}^*$. Here A is a set of possible operations (*transition, accept, error* and *convolution* (a_i, m_{ij})) for all $a_i \in V_N$, $m_{ij} = |\beta_i|$, where $a_i \rightarrow \beta_{ij} \in R$, that is the number of symbols in the right part of a generating rule $a_i \rightarrow \beta_{ij}$.
Let at the beginning construct LR(1)-analyzer of the language $L' = \{A_K(w) \mid w \in L(G)\} \subseteq \{0,1\}^*$, and then on its basis – the automat, implementing decoding of words $A_K(w) \in L'$ by binary blocks of the length m.
Let us define the function of operations f': $S' \times (V \cup \{0,1\}) \rightarrow A$ and the function of g: $S' \times (V \cup V_N \cup \{0,1\}) \rightarrow S'$ of the specified analyzer. With this aim for all $b_1 b_2 ... b_r = K(s, v)$ we will define

$$\begin{array}{ll} f'(s,b_1) = \textit{transition},\ g'(s, b_1) = s^{(1)}, & \\ \ldots & \\ f'(s^{(k-1)},b_k) = \textit{transition},\ g'(s^{(k-1)}, b_k) = s^{(k)}, & (43) \\ \ldots & \\ f'(s^{(r-1)},b_r) = \textit{convolution}\ (v, r),\ g'(s^{(r-1)}, b_r) = s, & \end{array}$$

where $s^{(k)}$ (k=1, ..., r-1) are additional states, a set of which corresponds to a set of inner nodes of the prefix code K_s: $V \rightarrow \{0,1\}^*$ such that $K_s(v) = K(s, v)$. For all
$v \in V$ and $a \in V_N$ we will define

$$\begin{array}{ll} f'(s, v) = f(s, v), & \\ g'(s, v) = g(s, v), & (44) \\ g'(s, a) = f(s, a). & \end{array}$$

Difference of algorithm of the language *L'* analyzer from similar algorithm for case *L(G)* is that at fulfilling of *convolution* (*v*, *r*) operation after removal from an analysis stack *r* the upper symbols of the alphabet {0,1} a symbol *v* is recorded in the decoded string, the reading head is established on a position in which the symbol *v* is situated, after that the analysis proceeds in the usual manner. Thus, in essence, the string is analyzed

$$K(s_{i_1}, v_{i_1}) \cdot v_{i_1} \ldots K(s_{i_n}, v_{i_n}) \cdot v_{i_n}, \tag{45}$$

where $s_{i_j} \in S$, $v_{i_j} \in V$ (j= 1, …, *n*), and also $i_1 = 0$.

Let us get down now to construction of the required decoding automat. We will define mappings λ_b: *S'* → *S'* and ψ_b: *S'* → ($V \cup \{\Delta\}$) so that at input of bit $b \in \{0,1\}$ in a state $s \in S'$ automat goes over to a state $\lambda_b(s)$ and gives $\psi_b(s)$-symbol of the alphabet *v* or an empty word Δ. We will express λ_b and ψ_b through *f '*, *g'* and mappings φ_v: *S'* → *S'* , $v \in V$, corresponding to the functions *f* and *g* of the language *L(G)* analyzer. Recurrent definition of φ_v, which correctness directly follows from algorithm of *LR*(1)-analysis, has the following form:

$$\varphi_v(s) = \begin{cases} g(s,v), & \text{if } f(s,v) \in \{transition, accept\}, \\ \varphi_v(g(s,a)), & \text{if } f(s,v) = convolution(a,m) \text{ and } g(s,a) \neq s, \\ s, & \text{if } f(s,v) = convolution(a,m) \text{ and } g(s,a) = s. \end{cases} \tag{46}$$

From the conceptual point of view $\varphi_v(s)$ is a state in which the analyzer goes over from a state *s* because of handling of the next symbol *v* and in which the analysis of the symbol following *v* is necessary. Thus, the third variant in the given definition excludes the possibility of cycling at construction of $\varphi_v(s)$.

As it is easy to see,

$$\lambda_b(s) = \begin{cases} f'(s,b), & \text{if } g'(s,b) = transition, \\ \varphi_v(s), & \text{if } g'(s,b) = convolution(v,r), \end{cases}$$

$$\psi_b(s) = \begin{cases} \Delta, & \text{if } g'(s,b) = transition, \\ v, & \text{if } g'(s,b) = convolution(v,r), \end{cases} \tag{47}$$

Proceeding from these interrelations, the required mapping of δ: $S' \times \{0,1\}^m \to V^*$, defining the automat decoding binary sequences from a set *L'* by blocks of the length *m*, can be recorded as follows:

$$\delta(s, b_1, ..., b_m) = \psi_{b_1}(s)\psi_{b_2}(\lambda_{b_1}(s))...\psi_{b_m}(\lambda_{b_{m-1}} \cdot ... \cdot \lambda_{b_1}(s)), \tag{48}$$

where · is the symbol of functions superposition. By this means, the decoding automat represents *LR*(1)-analyzer of the language *L'*, during which work in the state $s \in S'$ are read out recurrent *m* bits of decoded binary sequence. A line $\delta(s, b_1, ..., b_m) \in V^*$ is being recorded on their place, the reading head moves to its beginning and the ordinary *LR*(1)-analysis proceeds up to reaching the next symbol from the alphabet {0,1}. The specified steps are fulfilled until the final state will be reached. Apparently, the constructed automat is defined by relations

$f'(s, 0) = f'(s, 1) = decoding\ (m)$,
$g'(s, 0) = g'(s, 1) = s$,
$f'(s, v) = f\ (s, v)$, (49)

$g'(s, v) = g\ (s, v)$,
$g'(s, a) = g\ (s, a)$,

where as well as earlier, f and g are functions of operations and transitions of $LR(1)$-analyzer of the language $L(G)$, $v \in V$, $a \in V_N$, and *decoding* (m) is the additional action including described above operations over the next m bits.
For reduction of the memory size, used during decoding process, it is appropriate to take into consideration only those states $s \in S$ and, accordingly, to store only those values $\delta(s, b_1, ..., b_m)$, which are really passed by the automat. For this purpose, we will recurrently define the infinite sequence of sets
$S_{(i)}$ ($i = 0,1...$):

$$S_{(0)} = \{s_{(0)}\}, \qquad (50)$$
$$S_{(i+1)} = S_{(i)} \cup F^m_{(i)},$$

where $F^m_{(i)}$ is a set of states, in which the automat can go over from states $s \in S_{(i)}$ as a result of decoding of the recurrent bit block of the length m. $F^m_{(i)}$ is defined with the help of the obvious recurrent state:

$$F^m_{(i)} = \bigcup_{s \in S_{(i)}} \bigcup_{b_1..b_m \in \{0,1\}^m} \{\psi_{b_m} \cdot ... \cdot \psi_{b_1}(s)\}. \qquad (51)$$

Thus a set S' of states, which are really gone over by the automat during decoding process, is defined as follows:

$$S' = \bigcup_{i=0}^{\infty} S_{(i)}. \qquad (52)$$

The criteria of checking the end of S' construction process is obvious enough: if value j is such that $S_{(j-1)} \subset S_{(i)}$ and $S_{(j)} = S_{(j+1)}$, so $S_{(j)} = S'$.
Let us pass to problems of block coding for a case *of automata codes, which are decipherable with a finite delay over regular languages*.
All presented reasoning was referred to automata codes, decipherable over $LR(1)$-languages. By virtue of the fact that any regular language is $LR(1)$- language, all described results concern regular languages too. At the same time, the specificity of the latter supposes more simple, rather than general, statements and modes of the subsequent practical application of the specified results. Let some regular language is presented by a finite automat f: $S \times V \to S$ that is designated as $L(f)$. The automat which is coding words of the language $L(f)$ by means of automat code K: $S \times V$ $\{0,1\}^*$, decipherable with a finite delay over this language, can be defined by means of function Φ: $S \times V \to S \times \{0,1\}^*$, where $\Phi(s, v) = \langle f(s, v), K(s, v)\rangle$. We will construct the automat decoding codes of words of the language $L(f)$ by blocks of the length m. For this purpose, it is enough, starting with Φ, to define the function Φ_m^{-1}: $S \times \{0, 1\}^m \to S' \times V^*$. The decoding automat, which corresponds to this function, at receiving of the current binary block $b_1 \dots b_m$ in a state $s_i \in S'$ goes over to a new state $s_j \in S'$ and produces some word $w \in V^*$. Thus $\Phi_m^{-1}(s_i, b_1 \dots b_m) = \langle s_j, w \rangle$. We will discover the correlation between s_i, $b_1 \dots b_m$ from one side and $\langle s_j, w \rangle$ – from the other.
Elements of a set of states S' we will designate through $s_i^{(x)}$, where $s_i \in S$, $x \in \{0,1\}^*$. Thus, by definition,
$s_i^{(\Delta)} = s_i$, (53)
$s_i^{(i)} = f(s_i, v)$,
at $K(s_i, v) = x$.

States from the left parts of these equalities are, consequently, elements of a set S and in this connection they are below called as the core. All remaining states $s_i^{(x)} \in S'\text{-}S$, corresponding to inadmissible prefixes of symbols codes, admissible in a state s_i, are called additional.
It is obvious that the function of transitions of the automat, which is decoding by blocks of length 1, that is bit by bit, is defined as follows:

$$f_{(1)}\,(s_i^{(x)}, b) = s_i^{(xb)}. \qquad (54)$$

On the assumption of this, the function of transitions of the automat, which decodes by blocks of the length $m > 1$, is defined by recurrent relation:

$$f_{(m)}\,(s_i^{(x)}, b_1\ldots b_{m-1}\, b_m) = f_{(1)}\,(f_{(m-1)}\,(s_i^{(x)}, b_1\ldots b_{m-1}), b_m). \qquad (55)$$

The function of operations of the automat, which is decoding bit by bit, is defined by the following expression:

$$g_{(1)}(s_i^{(xb)}, b) = \begin{cases} \Delta, & \text{if} \quad s_i^{(xb)} \in S'-S, \\ v, & \text{if} \quad s_i^{(xb)} \in S \;\; \& \;\; xb = K(s_i, v). \end{cases} \qquad (56)$$

In other words, at going over to a state, which is additional, the automat will produce an empty word, and at going over to a state, which is core – a symbol v such that the sequence of bits which have arrived after the automat going over to the previous core state s_i, is a code of the specified symbol in this state.
The function of the automat operations, which is decoding by blocks of the length $m > 1$, is defined by recurrent relation

$$g_{(m)}\,(s_i^{(x)}, b_1\ldots b_{m-1}\, b_m) = g_{(1)}\,(s_i^{(x)}, b_1) \cdot g_{(m-1)}\,(f_{(1)}\,(s_i^{(x)}, b_1), b_2,\ldots b_m). \qquad (57)$$

By analogy to the automats considered above, the states which are not gone over in the decoding process, can be excluded from a set S' that generally allows to reduce the volume of the table $\Phi_{m\text{-}1}$.
Let us pass to problems of block decoding for a case *of structural codes, decipherable with a finite delay over context-free languages*. We will remind that to each structural code K: $R \rightarrow \{0,1\}^*$ it is possible to put in correspondence a set of rules R' in which every non-terminal has exactly two alternatives: $\alpha \rightarrow \beta_0$ and $\alpha \rightarrow \beta_1$.
In its turn the mapping $\Theta_{(1)}$: $V_N' \times \{0,1\} \rightarrow (V \cup V_N')$, where V_N' is a set of non-terminals, taking place in the rules entering in R', is put into correspondence to a set R'. Thus, $\Theta(a, b) = \beta_b$, if $a \rightarrow \beta_b \in R'$.
The logic of algorithm of the bit-by-bit decoding, using the mapping $\Theta_{(1)}$, is defined by following recurrent relations:

$$K^{-1}(bx, a\alpha z) = K^{-1}(x, a\, \Theta_{(1)}(b,\alpha)\, z), \qquad (58)$$
$$K^{-1}(\Delta, a) = a,$$

in which $b \in \{0,1\}$, $a \in V^*$, $\alpha \in V_N'$, $z \in (V \cup V_N')^*$. Here bx is an unprocessed part of decoded binary sequence, b is its first bit, a is the result of decoding of the processed part of the specified sequence, α is the first at the left nonterminal of the sentential form generated in the course of decoding, z is a part of this SF, following α. After all decoded sequence is settled, SF does not contain non-terminals.
The result of decoding of initial binary sequence x is $K^{-1}(x, a_0)$, where a_0 is an axiom of grammar G' with set of rules R'.

Let us construct the mapping $\Theta_{(m)}$, ensuring decoding by blocks of length $m>1$. We will find an expression for $\Theta_{(m)}(\alpha, b_1\ldots b_m)$, where $\alpha \in V_N'$, $b_i \in \{0,1\}$ (i=1,…, m). For this purpose we will record a sequence of direct derivations $x_0 \Rightarrow x_1$, $x_1 \Rightarrow x_2, \ldots, x_{m-1} \Rightarrow x_m$, where

$$
\begin{aligned}
&x_0 = \alpha,\\
&x_1 = \Theta_{(1)}(\alpha, b_1) = a_1\, \alpha'\, z_1,\\
&x_2 = a_1\, \Theta_{(1)}(\alpha', b_2)\, z_1 = a_2\, \alpha''\, z_2,\\
&\ldots\\
&x_i = a_{i-1}\, \Theta_{(1)}(\alpha^{(i-1)}, b_i)\, z_{i-1} = a_i\, \alpha^{(i)}\, z_i,\\
&\ldots\\
&x_m = a_{m-1}\, \Theta_{(1)}(\alpha^{(m-1)}, b_{m-1})\, z_{m-1},
\end{aligned}
\qquad (59)
$$

It is obvious from here that $\Theta_{(m)}(\alpha, b_1\ldots b_m) = x_m$. Using these expressions, generally it is possible to receive 2^m values of the function $\Theta_{(m)}(\alpha, b_1\ldots b_m)$ for all binary blocks of length m. We will note, however, that the case $x_m \in V^*$ is possible at $i < m$. In this connection, formally following the specified expressions, on all blocks of a form $b_1\ldots b_i\, b_{i+1}\ldots b_m$ such that $x_m \in V^*$, the function $\Theta_{(m)}$ is not defined. For elimination of this inconvenience we will redefine this function in the form
$\Theta_{(m)} : V_N \times \{0,1\} \rightarrow (V \cup V_N) \times \{1,\ldots, m\}$. Thus for all blocks $b_1\ldots b_i\, b_{i+1}\ldots b_m$ such that $x_m \in V^*$, we suppose $\Theta_{(m)}(\alpha, b_1\ldots b_i\, b_{i+1}\ldots b_m) = \langle x_i, i\rangle$, and for blocks $b_1\ldots b_m$ $\Theta_{(m)}(\alpha, b_1\ldots b_m) = \langle x_m, m\rangle$. At such definition of $\Theta_{(m)}$ the decoding is performed, strictly speaking, by variable length blocks so on each step the value of the index of the first bit of the current block increases by the value of the second component of the tuple, which is the value $\Theta_{(m)}$, corresponding to the previous block.
Through $\Theta_{(m)}(\alpha, x)\,[i]$, we will designate i component of the tuple which is the value of specified function (at values of arguments α and x) (i =1, 2), and through $\delta(w, j)$ is a substring of a string w, including numerals w beginning with j till $|w|$ at their indexing from left to right. Thus, the logic of decoding algorithm with usage of the mapping $\Theta_{(m)}$ can be recorded by means of a relation

$$K^{-1}(xy, a\alpha z) = K^{-1}(\delta(xy, \Theta_{(m)}(\alpha, x)\,[2])+1, a\Theta_{(m)}(\alpha, x)\,[1]\, z). \qquad (60)$$

Here $x \in \{0,1\}^*$ is the next decoded block of length m, $y \in \{0,1\}^*$ is an unprocessed part of decoded binary sequence. Remaining designations have the same sense, as earlier. As well as in case of automat codes, the information content, used in the course of decoding, is great enough and is estimated by the value $|V_N'| \cdot 2^m \cdot \bar{d}$, where $\bar{d}$ is the average memory size occupied by one value $\Theta_{(m)}(\alpha, x)$. However, as well as for the automat codes, the specified volume can be essentially reduced at the expense of elimination of the values corresponding to unattainable non-terminals, not used in the course of derivation. Non-terminal α' is directly achievable from the nonterminal α during coding-decoding process by means of the mapping $\Theta_{(m)}$ if there is a binary block x of length m such that $\Theta_{(m)}(\alpha, x) = \langle z_1\, \alpha'\, z_2, i\rangle$, where $z_1, z_2 \in (V \cup V_N)^*$. This fact is designated through $\alpha \rightarrow \alpha'$. The transitive closure of binary relation $\rightarrow$, designated through $\overset{+}{\rightarrow}$, is the binary relation, defining a set of pairs $\langle \alpha, \alpha' \rangle$ such that α' is achievable from α during coding-decoding. Thus all non-terminals α' which are not achievable from an axiom α_0, and corresponding to them values $\Theta_{(m)}(\alpha', x)$ can be excluded from reviewing.
Let's illustrate the described techniques of map Θ construction.
E x a m p l e 10. Let we have structural code K: $R \rightarrow \{0,1\}^*$, set by the following equalities (A - an axiom):

$$\begin{aligned}&K(A \to aB) = 0,\\ &K(A \to aBc) = 10,\\ &K(A \to \alpha) = 11,\\ &K(B \to d) = 0,\\ &K(B \to bB) = 1.\end{aligned} \qquad (61)$$

After transformation of a number of rules *R* to bialternative form we will receive the mapping Θ, presented in the form of the table 1a (*A'* is an auxiliary nonterminal). The mapping $\Theta_{(2)}$, ensuring decoding by blocks of length 2 and constructed on a basis of $\Theta_{(1)}$, is given in the format of the table 1b.

Table 1.

$\Theta_{(1)}$	0	1
A	$A \to aB$	$A \to A'$
A'	$A' \to aBc$	$A \to \alpha$
B	$B \to d$	$B \to bB$

a)

$\Theta_{(2)}$	00	01	10	11
A	$\langle ad, 2\rangle$	$\langle abB, 2\rangle$	$\langle aBc, 2\rangle$	$\langle d, 2\rangle$
A'	$\langle adc, 2\rangle$	$\langle abBc,$	$\langle d, 1\rangle$	$\langle d, 1\rangle$
B	$\langle d, 1\rangle$	$\langle d, 1\rangle$	$\langle bd, 2\rangle$	$\langle bbB, 2\rangle$

b)

Let we have the word *abbd*, generated from an axiom *A* by means of a chain of direct derivations $A \to aB \to abB \to abbB \to abbd$. From here the result of coding the word *abbd* with the help of structural code *K* is 0101. The course of bit-by-bit decoding is shown in the table 2a, each line of which corresponds to the current step, and the right column contains the result of fulfillment of this and the previous steps. The course of block decoding is shown in the table 2b, similar to 2a. ■

Table 2.

α	*b*	$\Theta_{(m)}(\alpha, x)$	
A	0	*aB*	*a*
B	1	*bB*	*ab*
B	1	*bB*	*abb*
B	0	α	*abbd*

a)

α	*x*	$\Theta_{(1)}(\alpha, b)$	
A	01	$\langle abB, 2\rangle$	*ab*
B	10	$\langle bd, 2\rangle$	*abbd*

b)

The methods of block decoding, considered in the present monography, are easily spread to alphabetic codes too, including the traditional codes of variable length synthesized for sources on the exit of which there can appear any words in the alphabet *V*. It is possible to present any such source in the form of a finite automation with the only state *s* and transitions function *f* such that $f(s, v) = s$ for any $v \in V$, then to construct a decoding automat according to this automat and a variable length code, just as it is described above.

6. Optimization of Syntax Analysis of Coded Messages

The general requirement to methods of syntax analysis (parsing) of the messages, coded by means of structural codes, is their generality, that is the possibility of application to any unambiguous context-free grammars.

It means that known methods of the deterministic analysis (both sequential [4], and parallel [15, 16]), developed for certain classes of grammars (*LR*(*k*), *LL*(*k*), precedence etc.), from positions of the specified restriction are nonapplicable. At the same time, the methods of the global analysis of Unger type of [17] and its modifications of [18] are applicable. Thereupon we will pay attention that further the term "syntax analysis optimization" is used by us not casually: optimization is wider concept, relating to any variety of context-free grammars and, generally speaking, not assuming the determined analysis.

On the other side, implementation of the deterministic analysis in variety of cases demands rather considerable storage consumption on operating tables and spending of time for searching in these tables. In this connection, this analysis can be less effective on the messages which length does not exceed certain critical value, than the optimized universal analysis (fig. 3). Apparently, the purpose of optimization of the universal analysis procedure is "shift" of critical value to the right on an axis l ($l_1 < l_2$) to the greatest possible limits.

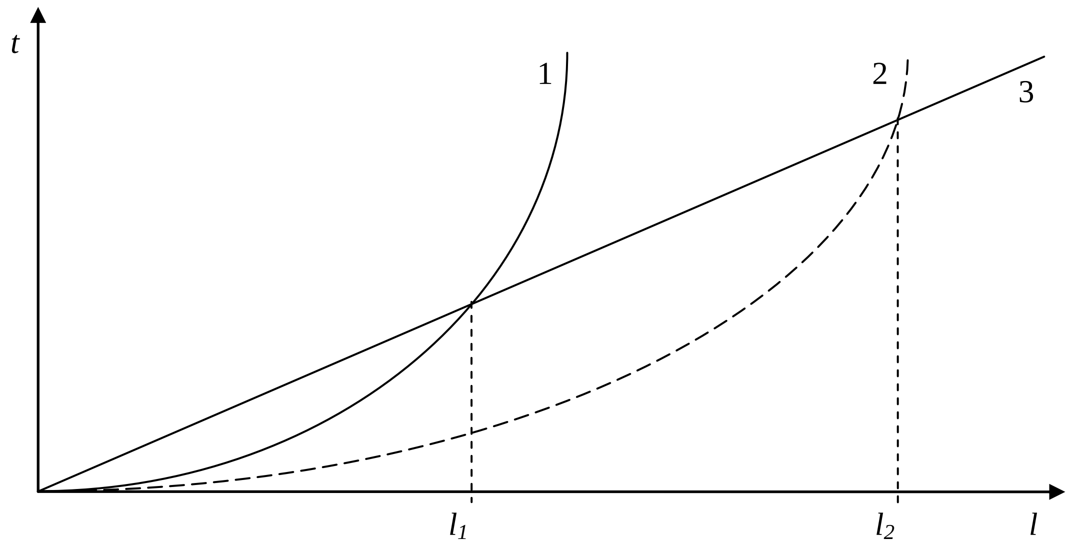

Fig. 3. Qualitative dependences of the message analysis time on its length:
1 – universal non-deterministic method; 2 – universal non-deterministic method with elements of optimization; 3 – deterministic method.

In this connection, the analysis optimization is treated as minimization of redundant search of generating rules.

Let us assume as a basis the classical procedure of top-down analysis which logic is defined by relations (62)-(64):

$$PARS(vx, w) = \begin{cases} PARS(x, w'), & \text{at } w = vw', \\ \varnothing, & \text{in other case;} \end{cases} \qquad (62)$$

$$PARS(\alpha x, w) = \bigcup_{\alpha \to \beta \in R} PARS(\beta x, w), \qquad (63)$$

$$PARS(\Delta, \Delta) = \{\Delta\}, \qquad (64)$$

where $v \in V$, $\alpha \in V_N$, $x \in (V \cup V_N)^*$, R is a set of generating rules of context-free grammar G, w is a message, $w \in V^*$. If $PARS(x, w) = \{\Delta\}$, then $x \overset{*}{\Rightarrow} w$, and if $PARS(x, w) = \varnothing$, then w is not derived from $x \in (V \cup V_N)^*$. In algorithmic implementation if there is no necessity to receive all deduction trees, the analysis is completed immediately after application of relation (64).

Computation complexity of top-down analysis procedure is defined finally by enumeration of alternatives of non-terminals according to expression (63). In this connection the problem of optimization of the given procedure is reduced to looking for ways of reduction of number of alternatives by cutting off those of them, for which obviously $PARS(\beta x, w) = \varnothing$. The most obvious method of this problem solution is the construction and use of sets of initial segments of words derived from alternatives and having the specified length *k*. Thus (63) is being transformed to the following form:

$$PARS(\alpha x, w) = \bigcup_{\substack{\alpha\to\beta\in R \\ S_k(w)\in FIRST_k(\beta)}} PARS(\beta x, w), \tag{65}$$

where $S_k(w)$ is *k*-symbol prefix of the word *w*, a $FIRST_k(\beta) = \{w \mid x \overset{*}{\Rightarrow} w\ w' \ \&\ w \in V^k \ \&\ w' \in V^* \}$ is a set of *k*-symbol prefixes of words, deduced from β. If for any $\alpha \in V_N$ and any β_i and β_j such that $\{\alpha\to\beta_i , \alpha\to\beta_j\} \subseteq R$,

$$FIRST_k(\beta_i) \cap FIRST_k(\beta_j) = \varnothing \tag{66}$$

(that corresponds to partition of $FIRST_k(\alpha)$ into disjoint subsets), then the grammar $G = <V , V_N , \alpha_0, R>$ is $LL(k)$-grammar. At that for any $\alpha \in V_N$ and $w \in V^*$
$|\{\beta \mid \alpha\to\beta \in R \ \&\ S_k(w) \in FIRST_k(\beta)\} \mid \leq 1$,
and the analysis is deterministic.

In global Unger analyzer with accelerating tests [17] more refined techniques of reduction of the alternatives enumeration, based on use of ranges of lengths of words derived from non-terminals (and, hence, any strings in the alphabet $V \cup V_N$), and sets of symbols, which complete the specified words (generally – *k*-symbol suffixes of these words) is applied. Let us consider the given techniques in more details. By analogy with $FIRST_k(x)$, where $x \in (V \cup V_N)^*$, we define $LAST_k(x) = \{w \mid x \overset{*}{\Rightarrow} w'\ w \ \&\ w' \in V^* \ \&\ w \in V^k\}$. The logic of construction $LAST_k(x)$ is as follows:

$$LAST(x\alpha) = \bigcup_{\alpha\to\beta\in R} LAST(x\beta), \tag{67}$$

$$LAST(x\,w) = \begin{cases} \{-(S_k(-w)))\} & \text{at } |w| \geq k, \\ LAST_{k-|w|}(x)\cdot\{w\} & \text{at } |w| \leq k, \end{cases} \tag{68}$$

where $\alpha \in V_N$, $w \in V^*$, $-(S_k(-w))$ is the result of inverting of *k*-symbol prefix of a word, received as a result of inverting of a word *w* (that is in the end *k*-symbol suffix).

Let us discover also expressions for the minimum and maximum lengths of the words derived from strings $x \in V \cup V_N$. Lengths are designated as $\lambda_{\min}(x)$ and $\lambda_{\max}(x)$ accordingly. We will define functions *l* and *l '* from two arguments everyone. The first argument is a string in the alphabet $V \cup V_N$, the minimum (maximum) length of which is defined, and the second argument is a set of non-terminals, used at generation of this string from initial. Upon that, the following relations take place

$$\lambda_{\min}(x) = l(x, \varnothing), \tag{69}$$

$$\lambda_{\max}(x) = l'(x, \varnothing), \tag{70}$$

$$l(\alpha x, a) = \begin{cases} \min\limits_{\alpha\to\beta\in R}\{l(\beta, a\cup\{\alpha\})\} + l(x,a) & \text{at } \alpha \notin a, \\ \infty & \text{at } |w| \le k, \end{cases} \tag{71}$$

$$l(wx, a) = |\,w\,| + l(x, a), \tag{72}$$

$$l'(\alpha x, a) = \begin{cases} \max\limits_{\alpha\to\beta\in R}\{l'(\beta, a\cup\{\alpha\})\} + l'(x,a) & \text{at } \alpha \notin a, \\ \infty & \text{at } |w| \le k, \end{cases} \tag{73}$$

$$l'(wx, a) = |\,w\,| + l'(x, a), \tag{74}$$

where in all expressions $x \in (V \cup V_N)^*$, $a \subseteq V_N$,$\alpha \in V_N$, $w \in V^*$. Correctness of relations $l(\alpha x) = \infty$ and $l'(\alpha x, a) = \infty$ at $\alpha \in a$ follows from that, in this case there takes place the generation of the form $\alpha \overset{*}{\Rightarrow} y\,\alpha\, y'$, where $yy' \in (V \cup V_N)^+$, and, in such a way, the length of words in the alphabet V, derived from α, is not limited from the top. We exclude the case $\alpha \overset{*}{\Rightarrow} \alpha$, supposing that context-free grammar is noncyclic.
With account of (67)-(74) the equality (65) is recorded in the following form:

$$PARS(\alpha\, x, w) = \bigcup_{\beta\in B_\alpha(w)} PARS(\beta\, x, w), \tag{75}$$

where

$$B_\alpha(w) = \{\beta \mid \alpha\to\beta \in R \;\&\; S_k(w) \in FIRST_k(\beta)\;\& \tag{76}$$
$$(\exists b \in LAST_n(\beta))\,(\exists\, a \in V^*)\; w = aba' \;\&\; \lambda_{\min}(\beta) \le |\,ab\,| \le \lambda_{\max}(\beta)\}.$$

As it follows from (75)-(76), for direct application are chosen only those of alternatives, for which there exists at least one string $b \in LAST_n(\beta)$ (n – some integer number only, not more $\lambda_{\max}(\beta)$), being a substring w, and also the length of the preceding it prefix a of a string w is in limits of $[\lambda_{\min}(\beta) - n, \lambda_{\max}(\beta) - n]$ (at that $\lambda_{\min}(\beta) \le |\,ab\,| \le \lambda_{\max}(\beta)$).
It is necessary to notice that ranges of the strings lengths derived from non-terminals, are defined a priori and in the course of analysis, remain constant. At the same time, the real length of the remained not parsed part of a string allows to correct dynamically an upper bound of the specified range and by that even more essentially to reduce redundant search. It is obvious that if the string w from (75) is derived from $\alpha\, x$, then the length of prefixes of strings, derived from α, is maximum in that case, when the length of suffixes *of* strings, derived from x, is minimum. In this connection, it is enough to redefine (76) as follows:

$$PARS(\alpha x, w) = \bigcup_{\beta\in \overline{B}_\alpha(x,w)} PARS(\beta x, w), \tag{77}$$

where

$$\overline{B}_\alpha(x,w) = \{\beta \mid \alpha \to \beta \in R \;\&\; S_k(w) \in FIRST_k(\beta)\;\& \tag{78}$$
$$(\exists b \in LAST_n(\beta))\,(\exists\, a \in V^*)\; w = aba' \;\&\; \lambda_{\min}(\beta) \le |\,ab\,| \le \lambda_{\max}(x)\}.$$

At this

$$\lambda_{\min}(a_1\alpha_{i_1}a_2...a_m\alpha_{i_m}a_{m+1}) = \sum_{i=1}^{m+1}|\,a_i\,| + \sum_{j=1}^{m}\lambda_{\min}(\alpha_{i_j}), \tag{79}$$

where $\lambda_{\min}(\alpha_{i_j})$ ($j = 1, \ldots, m$) is defined singlefold relating to a set R.

Rather effective means of reduction of alternatives search $\alpha \to \beta$ is checking of analyzed substring as for symbols entering it, achievable at derivation from these alternatives [18]. We will remind that the symbol *v* is achievable from nonterminal α, if there exist $x, y \in (V \cup V_N)^*$ such that $\alpha \overset{*}{\Rightarrow} x v y$ [4]. Let us designate a set of symbols of alphabet *V*, achievable from the string $z \in (V \cup V_N)^*$, through $Д(z)$:

$$Д(z) = \{v \mid v \in V \,\&\, (\exists\, x \in (V \cup V_N)^*)\,(\exists\, y \in (V \cup V_N)^*)\ z \overset{*}{\Rightarrow} x v y\}$$

The relations which are used for definition of $Д(z)$, are obvious enough (as well as earlier $v \in V$, $\alpha \in V_N$, $x \in (V \cup V_N)^*$):

$$Д(vx) = \{v\} \cup Д(x), \tag{80}$$

$$Д(\alpha x) = Д(x) \cup (\bigcup_{\alpha \to \beta \in R} Д(\beta)), \tag{81}$$

Having designated through $V(a)$ a set of symbols of the alphabet *V*, entering the string *a*, we will receive the following integral expression for "filtration" of alternatives, generalizing (78):

$$\overline{B}_\alpha(x, w) = \{\beta \mid \alpha \to \beta \in R \,\&\, S_k(w) \in FIRST_k(\beta) \,\&\, (\exists b \in LAST_n(\beta)) \tag{82}$$
$$(\exists\, a \in V^*)\ w = ba' \,\&\, V(ab) \subseteq Д(\beta) \,\&\, \lambda_{\min}(\beta) \le |\,ab\,| \le |\,w\,| - \lambda_{\min}(x)\}.$$

From the point of view of practical implementation, the efficiency of alternatives filtration, according to (82), to a large extent depends on an application order of "elementary filters", entering into the common filter. The most appropriate is the following order:
1) Filtration on entrance of $S_k(w) \in FIRST_k(\beta)$.
2) For each of the remained alternatives β: in the substring *w*, including symbols from $\lambda_{\min}(\beta)$ to $|\,w\,| - \lambda_{\min}(x)\}$, the definition of the position *j* number of the first to the left symbol $v \in V - Д(\beta)$ (it is clear that for all left substrings *a* of the string *w*, having the length larger than *j*, the condition $V(a) \subseteq Д(\beta)$ is not fulfilled).
3) Redefinition of maximum length of the string deduced from β, till $j - 1$.
4) A cycle on *i* - number of a string *w* position from *j* -1 to *k*: until the last *n* symbols of left substring *w* of the length *i* do not enter in $LAST_n(\beta)$.
For an account of non-uniform distribution of alternatives probability at outcome of the strings, arriving at an input of the syntactic analyzer, it is appropriate to implement dynamic rearrangement of alternatives on the method of "stack of books" [14] which subject matter is more known to developers of operating systems from discipline of substitution *LRU*. Thus, the used alternative is put into the list beginning that leads to shift of all remained alternatives on one position to the right. It is clear that most often used alternatives, as a rule, will be closer to the beginning of the list, and besides there is no necessity of accumulation of the statistical information here.

7. Parallel Syntax Analysis of Coded Messages

Let us begin with a parallelizing of top-down analysis with reference to the techniques considered in item 6. The most simple way in this direction is execution of function *PARS* and all its modifications in the form of parallel procedures. However, such parallelizing, on the one hand, demands extremely high expenditures of hardware resource, and with another it has, obviously, the upper border of efficiency resulting from consecutive "from left to right" processing of the analyzed word. In this sense, the method of the two-sided analysis at which the analysis is conducted from both ends of a word towards each other [16] is more perspective. Such approach under condition of "limiting" parallelizing of each of counter processes can ensure the further reduction of time of the analysis in comparison with one-sided parsing. The idea of the piecewise-parallel analysis [16, 19], which subject matter consists in partition of a parsing word into substrings (for example, programs into operators) and in parallel analysis of these substrings, is even more fruitful.

All told concerned top-down syntax analysis. As to parallelizing of bottom-up analysis, here there are certain premises for use of possibilities of computing structures with mass parallelism.

The techniques of parallel top-down analysis considered below is based on construction of the specialized computing structure, corresponding to a set of rules of the concrete context-free grammar. The specified structure represents a aggregate of active elements and physical connections ensuring direct information interchanging between elements. Simplicity of the last and the general principle ofoperating do such structures similar to systolic. In this connection, they are called hereafter quasi-systolic structures (QSS). We will notice that construction methods of systolic recognizers of regular languages are known and are studied well enough [20, 21]. The method of QSS construction considered below is the development of these methods with reference to wider class of context-free languages.

Let we have some context-free grammar of the grammar $G = <V, V_{\mathrm{N}}, \alpha_0, R>$. We will put in correspondence to it the oriented graph $\varphi(G)$ which techniques of construction is described, in particular, in [4]. This techniques is illustrated by the following example.

E x a m p l e 11. Let we have a context-free grammar with a set of rules $\{A \to BC, A \to Ba, B \to bc, C \to BBc, C \to c\}$, the axiom A, the terminal alphabet $\{a, b, c\}$ and nonterminal alphabet $\{A, B, C)$. The graph $\varphi(G)$, represented in fig. 4, corresponds to this grammar. ■

In the graph $\varphi(G)$ the number of nodes is $| V | + | V_{\mathrm{N}} | + | R |$, and to each rule there corresponds the so-called *-node containing a special auxiliary numeral "*" ("concatenator"). The amount of the edges entering the node $\alpha \in V_{\mathrm{N}}$, is equal to an amount of alternatives of nonterminal α, and each edge is marked by the alternative number.

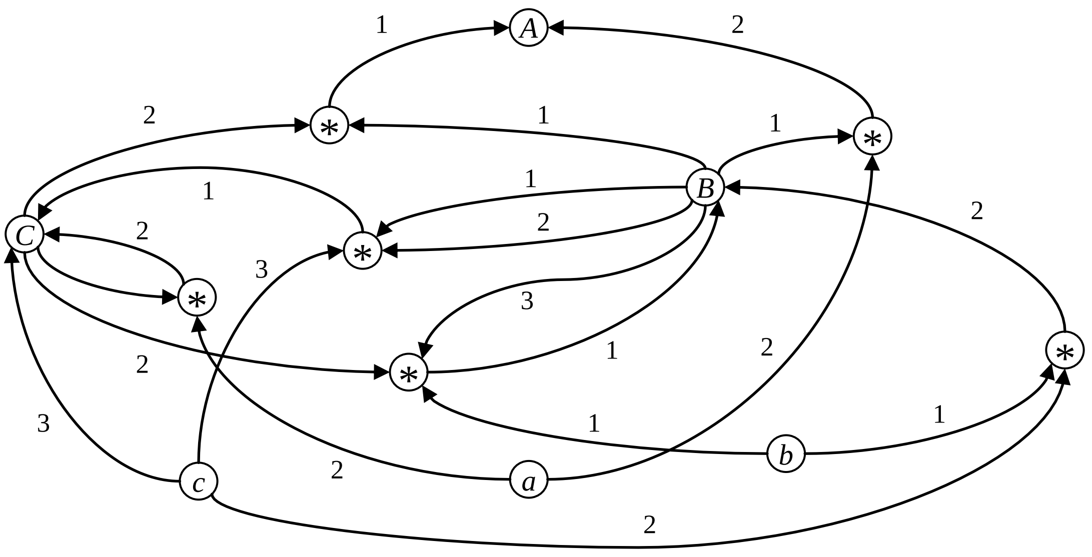

Fig. 4. Graph $\varphi(G)$, corresponding to grammar G from an example 11

The amount of the edges which arise from the node $a \in V \cup V_N$, is equal to the number of entrances of a symbol a into right parts of rules from a set R. The edge connecting the node top $a \in V \cup V_N$, with node top *, corresponding to a rule, into which right part enters symbol a, is marked with a number of position of the right part occupied with this symbol. Junction of the node * with the node $\alpha \in V_N$ by a

edge i means that the string of symbols of the alphabet $V \cup V_N$, "collected" by node *, is i alternative of non-terminal α. Naturally, no one edge enters the nodes $v \in V$, and any edge does not arise from the node of axiom of G. Generally the edge i can connect to the node $\alpha \in V$ not only the node *, but also the node $\alpha \in V \cup V_N$. It means that $\alpha \rightarrow a \in R$ and a is i alternative of nonterminal α.
Let us regularize the graph $\varphi(G)$, i.e. we will lead it to form when no more than two edges enter each node and arise from each node. Transformation of regularity assumes performance of following operations:
1 – regularity of α-nodes on inputs;
2 – regularity of *-nodes on inputs;
3 – regularity of all nodes on exits.
Let we have a tree – a subgraph of the graph $\varphi(G)$ which root is α-node, edges – all edges of the graph $\varphi(G)$ entering into it and leaves – nodes $\varphi(G)$ from which these edges arise from. If number n of leaves of this tree is more than two, we include a new α-node in $\varphi(G)$, which we connect to a root by the edge marked with number 2 and with leaves from which the edges arise from, marked with numbers 1..., $n - 1$ accordingly. We remove all edges connecting a root of a tree with leaves. Further on the described transformation it is applied to subtree, the root of which is the introduced node and so until we will receive from an initial tree such in which the branching factor is equal to two. The regularity of α-nodes of the graph $\varphi(G)$ on inputs consists in performance of described operations in relation to all α – to nodes, the number of edges entering each of which is more than two. If this number is less than three, the corresponding subgraph of the graph is left without modifications.
Regularization of *-nodes on inputs is similar to described with that difference that *-nodes, but not α- are included in the graph. The nuance, however, consists that the edges marked with numbers of positions enter *-node, occupied with symbols $a \in V \cup V_N$, in the right part of the rule corresponding to this *-node. In connection with *-node "multiplication" the specified numbers should be corrected. The logic of such correcting is obvious: the edge, connecting two *-node, is always marked with number 2, and the edge connecting α-node with *-node is marked with number 1. Exception is the tree which two leaves are α-node. In that case, the edge with a smaller number is marked with one, and with bigger number – with two.
At last, the regularization of nodes on exits differs from previous by that a root of a tree being transformed is the node $a \in V \cup V_N$., and leaves are the nodes connected to it by edges arising from it. It is essential that indexing of these edges is not continuous, as in two previous cases. So, in the graph $\varphi(G)$ in fig. 4 from the node c, arise two edges marked with number 3, and from the node B arise five edges, marked with numbers 1, 1, 1, 2 and 3. After the regularization of *-nodes on inputs the arising edges are marked only by numbers 1 and 2, however the edges marked with the same number, generally more than one. In this connection, the described above logic demands some adaptation to new conditions. The specified adaptation comes to that after construction of binary prefix tree with a root $a \in V \cup V_N$ and leaves from a set $V_N \cup \{*\}$ the edges entering in leaves, are marked with the same numbers, as edges of a graph entering into the same nodes before the regularization exits. The latter after marking of new edges are removed. In interior non-root nodes of the constructed binary prefix tree the special symbol $\text{v} \notin V \cup V_N \cup \{*\}$ ("multiplicator") is put. In this connection, they are called hereafter as v-nodes. The edges entering into v-nodes, are not marked.
Naturally, the regularization exits is fulfilled only for nodes from which more than two edges arise.
The graph received from the graph $\varphi(G)$ as a result of described operations fulfillment, will be designated further on through $\bar{\varphi}(G)$. For more natural representation, from positions of the further use, we will mark edges $\bar{\varphi}(G)$ with binary figures 0 and 1 instead of decimal 1 and 2 accordingly. Thus the regularization of the graph of context-free grammar becomes the generalization of McNaughton – Yamada method [20], regularizing the graphs, corresponding to automat grammars. We will notice that in the graph $\bar{\varphi}(G)$ the symbols, which are taking place in α-nodes, are of no importance. In this connection, further on we use a unique symbol α. Analogously, only those binary

figures 0 and 1 which mark the edges entering into *-nodes are of importance. In this connection all remaining edges are not marked.

The regularized graph $\overline{\varphi}(G)$, corresponding to grammar G with a set of rules $\{A \to cbb, A \to abBB, B \to Ba, B \to b\}$, is represented in fig. 5.

While interpreting nodes as separate elements, and edges as connections between them, it is possible to consider the graph $\overline{\varphi}(G)$ as computing structure. However $\overline{\varphi}(G)$ yet is not computing structure as no algorithm of operation is connected with it, i.e. the information circulating on connections, and the transformations implemented by elements are finally not defined.

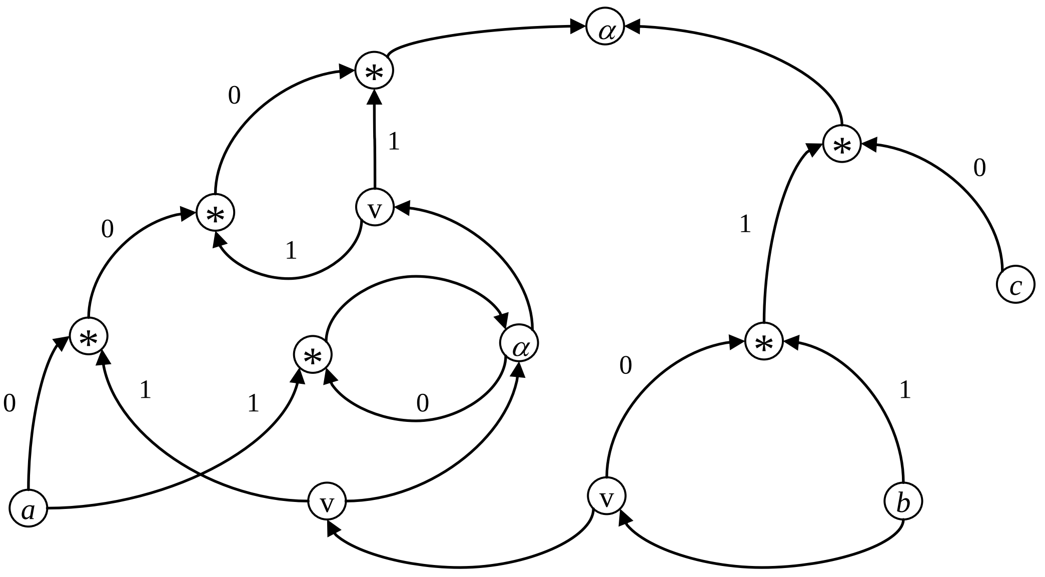

Fig. 5. Regularized graph $\overline{\varphi}(G)$

Let it be established that a substring of a parsed word w, including (i - j) symbols w, is derived from nonterminal α', a substring w, including its (j + 1 - k) symbols, is derived from nonterminal α'', a set of rules of grammar includes a rule $\alpha \to \alpha' \alpha''$. From here it is obvious that a substring w, including its (i - k) symbols, is derived from nonterminal α. The asserted is possible to record formally in the form of the following deduction rule:

$$\frac{\alpha'}{i:j}, \frac{\alpha''}{j+1:k}, \alpha \to \alpha'\alpha'' \in R \mid - \frac{\alpha}{i:k}. \tag{83}$$

This rule forms the basis of the required operation algorithm. On the one hand, it fixes the information transmitted on connections. This information is non other than couples of numbers, defining co-ordinates of parsed substrings. And on the other hand, it prescribes the basic transformation, implementing the bottom-up analysis.

Let's put now into correspondence the type of an active element to each type of the node of regularized graph $\overline{\varphi}(G)$. We have, thus, elements of following types:

- *t-elements* corresponding to symbols of the terminal alphabet;
- *α-elements* corresponding to non-terminals (except for an axiom α_0);
- **-elements* corresponding to generating rules;
- *v-elements*;
- α_0-*element* (unique in structure).

For the organization of structure operation we will connect decipherer to it on which input symbols of a parsed word arrive sequentially, and the exits which number is equal $| V | + 1$, are connected to inputs of t-elements. After inflow from the left of i symbol of a parsed word (let this symbol be v) from its v exit on an input of a corresponding i-element the value i arrives. We will consider the logic of structure elements operation.

Each *t-element* has one input connected to the corresponding exit of the decipherer, and two exits, one of which can be blocked (in the event if the terminal symbol has unique entrance into right parts of rules of a set R as, for example, the symbol c in fig. 5). On receipt of the number i on the input, the t-element will transmit simultaneously to both exits the pair $<i, i>$.

Each *α-element* has two inputs, one of which can be blocked (if corresponding nonterminal has a unique alternative), and two exits, one of which also can be blocked (if this nonterminal has a unique entrance into right parts of rules). On receipt of a pair $<i, i>$ on one of inputs, *α-element* transmits this pair simultaneously to both exits. If both inputs activate, the transmission of both pairs happens sequentially.

Each **-element* has exactly two inputs and only one exit. Besides, *-elements unlike all others possess memory in which two arrays are accumulated: M_0 and M_1 – pairs of numbers which have arrived on "zero" and "one" inputs accordingly. On receipt of a pair $<i, j>$ on a zero input it is brought in array M_0, after that in M_1 array a pair $<j + 1, k>$ is searched, and if it takes place, the pair $<i, k>$ arrives on input of *-element. The operations fulfilled after inflow of a pair $<i, j>$ on the only one input are similar: $<i, j>$ it is brought in array M_1, and the pair $<k, i - 1>$ is searched in array M_0. In case of success, the pair $<k, j>$ arrives on the exit.

Each *v-element* has a single input and exactly two exits. On receipt of a pair $<i, j>$ on an input, it is transmitted simultaneously to both exits.

As for *α_0-element* it differs from α-element by memory presence on one number, the additional operating input, connected with $| V | + 1$ exits of the decipherer, and a single exit for transmission of resulting signal (1/0 – success/failure). $| V | + 1$-st exit of the decipherer corresponds to the right terminator of a parsed word and serves for transmission of position number of this terminator on an operating input of α-element. It is clear that at transmission of value i the length of a parsed word equals to $i – 1$. That is why the operation of α_0-element at inflow on any of informational inputs of the pair $<1, k>$ is composed of comparison of k with value k', available in memory, and, if $k > k'$, in replacement k' for k. If on an operating input the number i arrives, and in memory there is the number $i – 1$, the α_0-element will produce a resulting signal "success"; otherwise – "failure". In an initial condition, 0 is stored in memory of α_0-element.

As is obvious, structure elements are extremely simple and similar among themselves. This allows unifying t-, α- and v-elements.

It is clear, why the structure is named quasi-systolic: each step of its operation consists in information "pushing through" on connections, and with simultaneous adding on upwards subtrees of analysis of an arriving word, which in the form of arrays M_0 and M_1 are stored in memory of *-elements. Thus, subtrees are constructed in a parallel way and substantially independently from each other. Synchronization of parallel processes and "assembling" of subtrees is fulfilled by *-elements.

Speed and cost of quasi-systolic structures (QSS) fatefully depend on *-elements: volumes of their memory and access time to arrays M_0 and M_1. Various approaches to the organization of the specified arrays are possible.

For reduction of information volume in arrays M_0 and M_1 it is possible to complicate a little bit the algorithm of α-element, having included in it the checking of performance of a condition $\lambda_{min} \le j - i + 1 \le \lambda_{max}$, where λ_{min} and λ_{max} are minimum and maximum lengths of strings in the alphabet V, derived from nonterminal, corresponding to this element. Values λ_{min} and λ_{max} are defined from relations (69)-(74). Upon that, transmission of the pair $<i, j>$ on exits of α-element is made only at performance of the specified condition. An obvious consequence of it is absence of such pairs in *-element, the "nearest" to α-element, which "absorbed" them. We will pay attention also that the recurrent entering symbol and the number of its position can arrive on the decipherer immediately after the release of the latter, after handling of a previous symbol. The same principle of synchronization can be implemented

for all remaining elements. Hardware implementation of syntactic analysis of coded messages on the basis of QSS ensures formation of codes of messages synchronously with construction of their derivation trees.
More detailed analysis of quasi-systolic structures (QSS) is beyond the scope of problems of this monograph.

Conclusion

The grammatical approach to coding, considered in the presented work, is universal enough theoretical tool for the analysis of efficiency of possible methods of compression of the big databases with the known structure of elements. Practical application of the considered algorithms during more than twenty years has shown that the most expedient area of use of the approach is archiving of databases within the framework of which, in some cases, it was possible to reach compression of accumulated arrays 2-3 times more essential, than at use of known methods (Huffman, Lempel-Ziv, etc.). Thus, the decrease in rate of coding in comparison with analogues was compensated by high enough rate of block decoding and by hardware implementation of codecs.
From positions of development of the described approach, the following directions represent the greatest interest:
1) Coding-decoding on the basis of "stack of books" techniques, excluding the necessity of accumulation and constant actualization of statistical information (distributions of probabilities of generating rules application in structural codes and occurrence of symbols in coding states in automata codes).
2) Generalization of approach on grammars, generating two-dimensional and three-dimensional images.
3) Usage of structural codes as a basis of implementation of associative access to elements of databases, which is considered in the basics in [6].
Possible versions of grammatical codings for solution of these problems will make a subject of subsequent publications.

Bibliography

1. Markov A.A. An Introduction to the Theory of Coding. – Moscow: Nauka, 1982 (in Russian).
2. Huffman D. Methods of Codes with Minimum Redundancy Construction. – Cybernetics collection. Vol. 3. – Moscow: Foreign Literature, 1961 (Russian Translation).
3. Fu K. Structural Methods in Automatical Recognition. – Moscow: Mir, 1977 (Russian Translation).
4. Aho A., Ullman J. Theory of Syntax Analysis, Interpreting and Compilation. Vol. 1, 2. – Moscow: Mir, 1978 (Russian Translation).
5. Sheremet I.A. Effective Coding of the Formalized Messages. - Cybernetics and the System Analysis, 1992, No. 3 (in Russian).
6. Sheremet I.A. Intellectual Software Medias for Automated Information Processing System (AIPS). – Moscow: Nauka, 1994 (in Russian).
7. Zhiltsova L.P. About Algorithmic Complexity of Problems of Optimum Alphabetic Coding for Context-free Languages. – Discrete Mathematics, 1989, No. 2 (in Russian).
8. Cormack G.V. Data Compression on a Database System. – Comminucations of the ACM. – Vol. 28 (1986), No. 12.
9. Sachkov B.H. Combinatorial methods of discrete mathematics. – Moscow: Nauka, 1977 (in Russian).
10. Markov A.A. About Dependence of Efficiency of Alphabetic Coding on the Language of Messages. – Reports of Academy of Sciences of the Union of Soviet Socialist Republics (AS USSR). – Vol. 258 (1981), No. 2 (in Russian).
11. Markov A.A. System of Equations of Antiprefixability in Words. – Discrete Mathematics, 1990, No 2 (in Russian).
12. Markov A.A., Smirnova T.G. Algorithmic foundation of generalized –prefix coding. – Reports of AS USSR. – Vol. 274 (1984), No. 4 (in Russian).
13. Wetherel C.S. Probabilistic Languages: A Review and Some Open Questions. – ACM Computing Surveys. – Vol. 12 (1980), No. 4.
14. Krichevskiy R.E. Compression and Retrieving of Information. – Moscow: Radio and Communication, 1989 (in Russian).
15. Trakhtenherts E.A. An Introduction to the Theory of Analysis and Parallelizing of Computer Programs in Translation Process. – Moscow: Nauka, 1981 (in Russian).
16. Trakhtenherts E.A. Software Parallel Processes. – Moscow: Nauka, 1987 (in Russian).

17. Unger S.H. A Global Parser for Context-Free Phrase Structure Grammars. – Communications ACM. – Vol. 11 (1968), No. 4.
18. Bratchikov I.L. Syntax of Programming Languages. – Moscow: Nauka, 1975 (in Russian).
19. Kupriyanov B.V. Parallel Syntactic Analysis Based on *LR*(*k*)-analysis. – Cybernetics Problems. Issue 3. – Moscow: 1978 (in Russian).
20. Ullman J. Computation Aspects VLSI (very-large-scale integration circuit). – Moscow: Radio and Communication, 1990 (Russian Translation).
21. Forster J. Silicon Compiler and Crystal for Clarification // Electronics VLSI. Designing of Microstructures. – Moscow: Mir, 1989 (Russian Translation).