Proceedings of the Jangjeon Mathematical Society
28 (2025), No. 3, pp. 299 – 317

www.jangjeon.or.kr
http://dx.doi.org/10.17777/pjms2025.28.3.299

ENUMERATION OF WORDS DERIVED FROM UNAMBIGUOUS CONTEXT-FREE GRAMMARS BASED ON POWERS OF GENERATING FUNCTIONS

YURIY SHABLYA, VLADIMIR KRUCHININ, AND DMITRY KRUCHININ

Abstract. Formal languages are widely used in theoretical computer science and its various applications. A generative grammar allows us to form the set of all words of the corresponding formal language by applying possible combinations of production rules. If we fix the length of the generated words, then the set defined by the grammar will be finite. However, further work with such combinatorial sets often requires knowing a formula for calculating the number of elements they contain. In the case of using an unambiguous context-free grammar, we can construct a generating function associated with it. This article considers the problem of obtaining explicit formulas for the coefficients of generating functions associated with unambiguous context-free grammars. The authors propose a method for solving this problem based on the application of the Lagrange inversion formula and compositae of generating functions. In order to test the proposed method, the article also presents several examples of finding explicit formulas for calculating the number of words with a fixed length, which are derived from a given unambiguous context-free grammar.

2020 Mathematics Subject Classification: 05A15; 68Q45.

Keywords and phrases: context-free grammar; combinatorial set; enumeration problem; generating function; Lagrange inversion formula; composita; explicit formula; combinatorial generation.

1. Introduction

A formal language is a set of words obtained using a given finite alphabet. Formal languages are widely used in the field of theoretical computer science and its various applications. The abstract description of discrete structures obtained by applying the theory of formal languages formalizes the processes associated with their syntactic and semantic analysis. In turn, this allows us to achieve a deeper understanding of the studied object and makes it possible to develop effective algorithms for processing the associated information. A well-known example of the use of formal language theory is the field of modern programming languages and their compilers. In this case, formal grammars are used as a tool for describing the syntactic structures of a programming language [1].

A generative grammar allows us to form the set of all words of the corresponding formal language by applying possible combinations of production

The reported study was supported by the Russian Science Foundation (project no. 22-71-10052).

Submission date: March 21, 2025.

rules. Thus, applying a generative grammar that produces some discrete structures, we can create a set of correct samples. Then, such a set of correct samples can be used to check the correctness of the algorithms that process the corresponding discrete structures. In addition, the inverse problem is possible when we have a test sample and we need to check its correctness. For example, in the field of modern programming languages, this is used to check the syntax of a program's source code for its correctness. In this case, the source code of the program is a word, and it is necessary to check the possibility of generating this word by applying the production rules of generative grammar corresponding to the selected programming language.

If we fix the length of the generated words, then the set defined by the grammar will be finite. In this way, a combinatorial set is formed, the elements (combinatorial objects) of which are words of a given length from the corresponding formal language. One of the tasks associated with processing combinatorial sets is the development of combinatorial generation algorithms [2]. Combinatorial generation algorithms can number the elements of a combinatorial set (ranking algorithms) and generate them by a single variant (unranking algorithms or random generation) or by exhaustive generation [3]. For example, the development of combinatorial generation algorithms for a set of words of a formal language defined by some generative grammar can be useful in the following areas of research: the problem of data compression and the problem of modeling complex discrete structures.

In terms of the first problem, the article [4] proposes an approach to compressing data strings, which are words with a fixed length from some formal language, by using a ranking algorithm. The proposed idea of ranking strings is based on calculating the number of words from a formal language that are less than a given word in lexicographic ordering. In particular, for an unambiguous context-free grammar, formulas for calculating the rank with polynomial computational complexity were presented in [4]. The possibilities of generalizing ranking algorithms to ambiguous context-free grammars were investigated in [5], and the paper [6] studied the issues of parallelizing the corresponding computations.

Further development of this problem is proposed in the article [7], where a ranking algorithm is presented together with an unranking algorithm for a set of words with a fixed length from a formal language with polynomial computational complexity. However, this study considers words that describe a sequence of applying production rules of another context-free grammar (words from the left Szilard languages), i.e., an additional representation of the source data is used. The article [8] presents ranking and unranking algorithms that can work with ambiguous context-free grammars by processing parse trees. Also, as an example, the problem of compressing the source code of a program in the C programming language was considered. Further development in this direction will make it possible to create data compression algorithms based on formal grammars [9].

In terms of the problem of modeling complex discrete structures, the articles [10, 11] consider the generation of a random word with a fixed length by applying the production rules of a given context-free grammar in a random order. In this case, to ensure uniform distribution of generated words,

the author proposes preliminary calculation of probabilities for each output rule, i.e., a probabilistic context-free grammar is formed. If we apply an unranking algorithm for random generation, then the uniform distribution of generated words is ensured by using a random number generator. A solution to the problem of non-uniform generation of words using an unranking algorithm is proposed in [12]. For example, such combinatorial generation algorithms allow modeling and compressing datasets of RNA secondary structures [13, 14].

Examples of new ideas in the development of random generation algorithms using context-free grammars include solving the hypergraph generation problem [15] and testing software for vulnerabilities [16]. In addition, combinatorial generation algorithms for context-free languages are used in the field of cryptography, for example, for format-preserving data encryption [8, 17, 18].

Thus, the development of new methods for constructing combinatorial generation algorithms for formal languages is a relevant scientific task, since it has both theoretical and practical significance due to existing applications in data compression and modeling problems. However, the correct operation of combinatorial algorithms often requires a method for calculating the number of elements in the corresponding combinatorial set. For example, to obtain an unranking algorithm using the framework presented in [19], we need to have a function $\texttt{count}(A, n)$ that returns the total number of objects of size n in a combinatorial set A. Similar requirements exist when applying the method of obtaining unranking algorithms based on AND/OR trees [20], where it is necessary to calculate the value of $w(s)$, i.e., the number of variants in the subtree structure of a node s.

In order to use combinatorial generation algorithms for a given formal grammar, it is necessary to be able to calculate the cardinality of the set of generated words. If we work with an unambiguous context-free grammar, then, based on its production rules, we can obtain recurrent formulas for calculating the number of words derived from the start symbol. Such recurrent formulas are not efficient in terms of computational complexity. Therefore, obtaining explicit formulas instead of recurrent ones is an important task of optimizing the corresponding combinatorial algorithms.

Generating functions are a widely used tool in the field of enumerative combinatorics [21, 22]. Research in the field of generating function theory is constantly updated with new ideas that affect their relationship with special numbers [23] and polynomials [24, 25], or consider their various generalizations [26]. The coefficients of a given generating function associated with a combinatorial set show the number of elements in the set. In the case of unambiguous context-free grammars, it is possible to get a generating function for a sequence of numbers that are the number of derived words with a fixed length. Therefore, the task of obtaining formulas for the coefficients of generating functions associated with context-free grammars is important and relevant. For example, in [27] the authors consider the same task with the usage of generating functions and show that this problem has the complexity class NC. However, the authors do not present any specific

rules for obtaining formulas for the coefficients of generating functions associated with context-free grammars. At the same time, there are examples of solving special cases where context-free grammars and generating functions are used simultaneously (for instance, those related to special numbers and polynomials [28, 29, 30]).

Thus, the main aim of this study is to develop a method for obtaining explicit formulas for the coefficients of generating functions associated with unambiguous context-free grammars. In Section 2, we described the main concepts and methods used in this study. Then, we present own method for obtaining coefficients of generating functions associated with unambiguous context-free grammars. In order to test the proposed method, in Section 3, we show several examples of finding explicit formulas for calculating the number of words with a fixed length, which are derived from a given unambiguous context-free grammar. A discussion of the obtained results is shown in Section 4.

2. Materials and Methods

2.1. **Description of the used methods.** A formal grammar is a tuple $G = (T, N, S, P)$, where:

- T is the set of terminal symbols of the grammar;
- N is the set of non-terminal symbols of the grammar, $T \cap N = \varnothing$;
- S is the start symbol of the grammar, $S \in N$;
- P is the set of production rules of the grammar in the form $\alpha \to \beta$, where $\alpha \in (N \cup T)^* N (N \cup T)^*$ is a sequence of grammar symbols with at least one non-terminal symbol, $\beta \in (N \cup T)^*$ is a sequence of any grammar symbols.

By sequentially applying the production rules, starting from the initial symbol of the grammar $S \in N$, a sequence of terminal symbols (word) $\omega \in T^*$ is derived. The derivation of a word $\omega \in T^*$ is denoted by $S \Rightarrow^* \omega$. The language of the grammar $G = (T, N, S, P)$ is the set of all words derived from the start symbol S of the grammar, i.e.,

$$L(G) = \{\omega \in T^* \,|S \Rightarrow^* \omega\}.$$

This article discusses formal grammars that belong to the type of context-free grammars [32, 31]. Such grammars contain only production rules of the form $\alpha \to \beta$, where $\alpha \in N$ and $\beta \in (N \cup T)^*$. In addition, we consider the limitation on the length of words derived from a given formal grammar G, i.e., we have a subset $L_n(G) \subseteq L(G)$, where each word $\omega \in L_n(G)$ has a fixed length $|\omega| = n$:

$$L_n(G) = \{\omega \in T^* \,|S \Rightarrow^* \omega, \;\; |\omega| = n\}.$$

Therefore, for a fixed n, the formal grammar $G = (T, N, S, P)$ defines a finite set $L_n(G)$ containing $|L_n(G)|$ distinct words with length n. Based on the production rules P, we can easily derive the recurrent formula for calculating the number of words with a fixed length n, which can be derived from the formal grammar G. To obtain an explicit formula from a recurrent one, it is necessary to apply appropriate techniques for solving recurrence relations. However, this task is difficult and has no universal solution.

If we consider different values of n, we obtain the following sequence of values for the number of words derived from the formal grammar G:

$$|L_0(G)|, \quad |L_1(G)|, \quad |L_2(G)|, \quad \ldots$$

We can also represent this sequence as a generating function as follows:

$$|L_0(G)|+|L_1(G)|\,x+|L_2(G)|\,x^2+\ldots=\sum_{n\geq 0}|L_n(G)|\,x^n=\sum_{n\geq 0}s(n)x^n=S(x).$$

Thus, the coefficients $s(n)$ of the obtained generating function $S(x)$ show the number of words ω with a fixed length n that are derived from the start symbol S of the grammar $G=(T,N,S,P)$.

According to the Chomsky–Schutzenberger enumeration theorem [33], it is possible to get a functional equation related to a generating function for a sequence of numbers that are the number of fixed-length words derived from a given unambiguous context-free grammar. To do this, it is necessary to represent all production rules for S in the form $S\to\beta_1|\beta_2|\ldots|\beta_m$, where $S\in N$ and $\beta_i\in(N\cup T)^*$, instead of $S\to\beta_1, S\to\beta_2,\ldots,S\to\beta_m$ and perform the following actions:

- replace the sign '$\to$' with the sign '=';
- replace the sign '|' with the sign '+' (addition);
- replace the concatenation of terminal and non-terminal symbols with the sign '$*$' (multiplication);
- replace start symbol S with a generating function $S(x)$, including all its appearances in each β_i;
- replace all other non-terminal symbols in β_i with their corresponding generating functions;
- replace each ε terminal symbol in β_i with 1;
- replace all other terminal symbols in β_i with the variable x;
- repeat the above actions for other non-terminal symbols and their corresponding production rules.

Thus, based on the production rules of a given unambiguous context-free grammar G, we can obtain a functional equation for the generating function $S(x)$.

Therefore, if we find an explicit formula for the coefficients $s(n)$ of the generating function $S(x)$, then we get an efficient way to calculate the cardinality function of the set of generated words with a fixed length n. The solution of the obtained functional equation allows us to find a closed-form for the generating function $S(x)$. One of the actively applied techniques in solving functional equations associated with generating functions and their coefficients is the use of the Lagrange inversion formula [34]. According to this formula, if we have a functional equation

$$F(x)=xG(F(x)), \tag{1}$$

where

$$F(x)=\sum_{n>0}f(n)x^n, \quad G(x)=\sum_{n\geq 0}g(n)x^n,$$

then, for $0\leq k\leq n$,

$$n[x^n]F(x)^k=k[x^{n-k}]G(x)^n.$$

In addition, if we use the coefficients of the power of the generating function

$$F(x)^k = \sum_{n>0} f(n,k)x^n, \quad G(x)^k = \sum_{n\geq 0} g(n,k)x^n,$$

then we get

$$f(n,k) = \frac{k}{n} g(n-k,n).$$

Thus, if a given functional equation can be transformed to the form (1), then we can calculate the coefficients of the generating function $F(x)$ and its powers through the corresponding coefficients of the generating function $G(x)$ or vice versa.

At the same time, a methodology based on compositae of generating functions can be used to find explicit formulas for the coefficients of a generating function [35]. The composita $F^{\Delta}(n,k)$ of a generating function $F(x)$, where $F(0) \neq 0$, is a coefficients function of its k-th power, i.e., $F^{\Delta}(n,k) = [x^n]F(x)^k$. According to this methodology, a given generating function must be decomposed into simpler ones, for which explicit formulas for the coefficients of their powers can be easily found. In this case, we can apply operations on generating functions such as addition, multiplication, composition, reciprocation and compositional inversion. Moreover, compositae have a generalization to the case of bivariate and multivariate generating functions [36, 37].

2.2. **Description of the proposed method.** Using a combination of the above methods, it is possible to obtain explicit formulas for the coefficients of generating functions associated with unambiguous context-free grammars. In particular, based on the Chomsky–Schutzenberger enumeration theorem, we can obtain a functional equation for the generating function $S(x)$ associated with a given unambiguous context-free grammar G. If there is a closed-form solution to the obtained functional equation, then we can find an explicit formula for the coefficients of $S(x)$ by decomposing it into simpler generating functions and applying the rules for calculating compositae. Otherwise, we can transform the obtained functional equation to the form of the functional equation used in the Lagrange inversion formula. Then we get a formula for calculating the coefficients of $S(x)$ through the coefficients of another generating function given by closed-form expression. Applying the methodology of compositae for this generating function, we get an explicit formula for the coefficients of $S(x)$.

Next, we describe in steps the proposed method for obtaining coefficients of generating functions associated with unambiguous context-free grammars.

Step 1. Based on the production rules of a given unambiguous context-free grammar $G = (T, N, S, P)$, construct functional equations for the generating functions associated with each non-terminal symbol. The resulting system contains functional equations, the number of which is equal to the number of non-terminal symbols in the grammar G. In addition, the obtained equations can be transformed into the form of multivariable polynomial equations, where the variables are x and all participating generating functions.

Step 2. Transform the system of functional equations and get an equation that includes only one generating function $S(x)$.

Step 3. If $s(0) \neq 0$ (it is possible to derive an empty word $S \Rightarrow^* \varepsilon$), then make the substitution $S(x) = \frac{R(x)}{x}$ in the functional equation. Otherwise make the substitution $S(x) = R(x)$ in the functional equation. Thus, the following generating function is obtained:

$$R(x) = \sum_{n>0} r(n)x^n.$$

This requirement is caused by the limitations of the mathematical techniques used further (the Lagrange inversion formula requires $f(0) = 0$ for $F(x)$ in (1)).

Step 4. Obtain an explicit formula for the coefficients $r(n)$ of the generating function $R(x)$. The following two ways are possible:

Step 4.1. Solve the functional equation with respect to $R(x)$ and obtain a closed-form for the generating function $R(x)$. Based on the closed-form solution, find an explicit formula for the coefficients of $R(x)$ (for example, by applying the methodology of compositae of generating functions).

Step 4.2. Transform the functional equation into one of the following forms and find an explicit formula for the coefficients of $R(x)$:

- Case 1: if we get $H(R(x)) = x$, where $R(x) = F(x)$ and

$$F(x) = \sum_{n>0} f(n)x^n, \quad H(x) = \sum_{n>0} h(n)x^n, \quad H(x)^k = \sum_{n>0} H^{\Delta}(n,k)x^n,$$

then

$$f(n) = \frac{1}{n} \sum_{i=0}^{n-1} \binom{2n-1}{n+i} \binom{n-1+i}{i} (-1)^i \frac{H^{\Delta}(n-1+i,i)}{h(1)^{n+i}} \tag{2}$$

or

$$f(n) = \frac{1}{n} g(n-1,n), \tag{3}$$

where

$$g(n,k) = [x^n]G(x)^k = [x^n] \left(\frac{x}{H(x)} \right)^k.$$

Therefore,

$$r(n) = [x^n]R(x) = [x^n]F(x) = f(n); \tag{4}$$

- Case 2: if we get $H(x, R(x)) = x$, where $R(x) = F(x,x)$ and

$$F(x,y) = \sum_{n\geq 0} \sum_{m>0} f(n,m)x^n y^m,$$

$$H(x,y) = \sum_{n\geq 0} \sum_{m>0} h(n,m)x^n y^m, \quad H(x,y)^k = \sum_{n\geq 0} \sum_{m>0} H^{\Delta}(n,m,k)x^n y^m,$$

then

$$f(n,m) = \frac{1}{m} \sum_{i=0}^{n+m} \binom{n+2m-1}{m+i} \binom{m-1+i}{i} (-1)^i \frac{H^{\Delta}(n,m-1+i,i)}{h(0,1)^{m+i}} \tag{5}$$

or

$$f(n,m) = \frac{1}{m} g(n, m-1, m), \tag{6}$$

where

$$g(n,m,k) = [x^n y^m] G(x,y)^k = [x^n] \left(\frac{y}{H(x,y)} \right)^k.$$

Therefore,

$$r(n) = [x^n] R(x) = [x^n] F(x,x) = \sum_{i=0}^{n-1} f(i, n-i). \tag{7}$$

Step 5. Obtain an explicit formula for the coefficients $s(n)$ of the generating function $S(x)$:

- if we have $s(0) = 0$, then $S(x) = R(x)$ and

$$s(n) = r(n); \tag{8}$$

- if we have $s(0) \neq 0$, then $S(x) = \frac{R(x)}{x}$ and

$$s(n) = r(n+1). \tag{9}$$

Note that the application of this method requires choosing which way will be used to obtain an explicit formula in Step 4 (Step 4.1 or Step 4.2). This choice may be based on the complexity of transformations that need to be performed for a given functional equation. To do this, it is necessary to compare the closed-form solution for the generating function $R(x)$ with the closed-form expression for the generating function $H(x)$ or with the closed-form expression for the generating function $G(x) = \frac{x}{H(x)}$. Then it is necessary to select a simpler generating function from them.

Thus, it becomes possible to obtain explicit formulas for calculating values of $s(n)$, i.e., calculating the number of words with a fixed length n, which are derived from a given unambiguous context-free grammar G. In contrast to existing studies that address the problem of enumerating a specific formal language, the proposed method is a general method and can be applied to any unambiguous context-free grammar.

The above formula (3) was obtained by using the Lagrange inversion formula for a functional equation of the following form:

$$F(x) = xG(F(x)).$$

The above formula (6) was obtained by using the Lagrange inversion formula for a functional equation of the following form:

$$F(x,y) = yG(x, F(x,y)).$$

The above formulas (2) and (5) were obtained by applying the methodology of compositae of generating functions to the generating functions $H(x)$ or $H(x,y)$ associated with these functional equations. Next we present the details of their proof.

Theorem 2.1. *For the functional equation*

$$H(F(x)) = x,$$

where

$$F(x)=\sum_{n>0} f(n)x^n, \quad F(x)^k=\sum_{n>0} F^{\Delta}(n,k)x^n,$$
$$H(x)=\sum_{n>0} h(n)x^n, \quad H(x)^k=\sum_{n>0} H^{\Delta}(n,k)x^n,$$

the following is true:

$$F^{\Delta}(n,k)=\frac{k}{n}\sum_{i=0}^{n-k}\binom{2n-k}{n+i}\binom{n-1+i}{i}(-1)^i\frac{H^{\Delta}(n-k+i,i)}{h(1)^{n+i}}.$$

Proof. Let us consider the following functional equation:

$$F(x)=xG(F(x)),$$

where

$$G(x)=\sum_{n\geq 0} g(n)x^n, \quad G(x)^k=\sum_{n\geq 0} g(n,k)x^n.$$

According to the Lagrange inversion formula, we have

$$F^{\Delta}(n,k)=\frac{k}{n}g(n-k,n). \tag{10}$$

For the case when

$$H(x)=\frac{x}{G(x)},$$

we transform the original functional equation to the following form:

$$H(F(x))=x.$$

Applying the result of Theorem 5 from [36] for the case

$$H_x(x)G(x)=1,$$

where

$$H_x(x)=\frac{H(x)}{x}=\sum_{n\geq 0} h_x(n)x^n, \quad H_x(x)^k=\sum_{n\geq 0} h_x(n,k)x^n,$$

we have

$$g(n,k)=\sum_{i=0}^{n}\binom{n+k}{k+i}\binom{k-1+i}{i}(-1)^i\frac{h_x0(n,i)}{h_x(0)^{k+i}}. \tag{11}$$

Applying

$$H_x(x)=\frac{H(x)}{x}=\sum_{n>0} h(n)x^{n-1}=\sum_{n>0} h(n+1,k)x^n=\sum_{n\geq 0} h_x(n)x^n,$$

$$\left(\frac{H(x)}{x}\right)^k=\sum_{n>0} H^{\Delta}(n,k)x^{n-k}=\sum_{n>0} H^{\Delta}(n+k,k)x^n=\sum_{n\geq 0} h_x(n,k)x^n$$

and (11) for (10), we get the desired result.

In addition, we have

$$f(n)=F^{\Delta}(n,1)=\frac{1}{n}\sum_{i=0}^{n-1}\binom{2n-1}{n+i}\binom{n-1+i}{i}(-1)^i\frac{H^{\Delta}(n-1+i,i)}{h(1)^{n+i}}.$$

□

Theorem 2.2. *For the functional equation*

$$H(x, F(x, y)) = y,$$

where

$$F(x,y) = \sum_{n\geq 0}\sum_{m>0} f(n,m)x^n y^m, \quad F(x,y)^k = \sum_{n\geq 0}\sum_{m>0} F^{\Delta}(n,m,k)x^n y^m,$$

$$H(x,y) = \sum_{n\geq 0}\sum_{m>0} h(n,m)x^n y^m, \quad H(x,y)^k = \sum_{n\geq 0}\sum_{m>0} H^{\Delta}(n,m,k)x^n y^m,$$

the following is true:

$$F^{\Delta}(n,m,k) = \frac{k}{m}\sum_{i=0}^{n+m}\binom{n+2m-k}{m+i}\binom{m-1+i}{i}(-1)^i \frac{H^{\Delta}(n,m-k+i,i)}{h(0,1)^{m+i}}.$$

Proof. The proof is similar to the proof of Theorem 6 in [36], where the following case of a functional equation was considered:

$$H(F(x,y),y) = x.$$

In our case, it is necessary to consider replacing variables x and y with each other.

□

In addition, we have

$$f(n,m) = F^{\Delta}(n,m,1) =$$

$$= \frac{1}{m}\sum_{i=0}^{n+m}\binom{n+2m-1}{m+i}\binom{m-1+i}{i}(-1)^i \frac{H^{\Delta}(n,m-1+i,i)}{h(0,1)^{m+i}}.$$

Note that for $y = x$, we have

$$H(x, F(x,x)) = x,$$

i.e., we get Case 2 of the proposed method.

3. Application of the proposed method

Next, in order to test the proposed method, we show several examples of finding explicit formulas for calculating the number of words with a fixed length, which are derived from a given unambiguous context-free grammar.

Example 3.1. *Let us consider the following unambiguous context-free grammar:*

$$G = (T, N, S, P), \quad T = \{(,), \varepsilon\}, \quad N = \{S\}, \quad P = \{S \to (S)S \,|\, \varepsilon\}.$$

Table 1 presents examples for several first values of $s(n)$ and the corresponding set of all generated words.

Based on the production rules $S \to (S)S \,|\, \varepsilon$, we obtain the following recurrent formula for calculating the number of words with a fixed length n, which are derived from the grammar:

$$s(n) = \begin{cases} 0 & n \text{ is odd}; \\ 1 & n = 0; \\ \sum\limits_{i=0}^{n-2} s(i)\cdot s(n-2-i) & \text{otherwise}. \end{cases} \tag{12}$$

TABLE 1. Several first values of $s(n)$ and the corresponding set of all generated words.

n	$s(n)$	***Generated words***
0	1	ε
1	0	$-$
2	1	( )
3	0	$-$
4	2	( ) ( ), (( ))
5	0	$-$
6	5	( ) ( ) ( ), ( ) (( )), (( )) ( ), (( ) ( )), ((( )))

However, using this recurrence formula is inefficient because it has exponential complexity. Next, to obtain an explicit formula, we apply the method proposed in the article.

Step 1: *Based on the production rules* $S \to (S)S \,|\, \varepsilon$, *we obtain the following functional equation:*

$$S(x) = x^2 S(x)^2 + 1. \tag{13}$$

Step 2: *The obtained functional equation includes only one generating function* $S(x)$. *Therefore, no additional transformations are required.*

Step 3: *Since* $s(0) \neq 0$, *we make the substitution* $S(x) = \frac{R(x)}{x}$ *in the functional equation* (13) *and get*

$$\frac{R(x)}{x} = x^2 \left(\frac{R(x)}{x}\right)^2 + 1$$

or

$$xR(x)^2 - R(x) + x = 0. \tag{14}$$

Step 4.1: *The closed-form solution for the functional equation* (14) *with respect to* $R(x)$ *is*

$$R(x) = \frac{1 - \sqrt{1 - 4x^2}}{2x} = x + x^3 + 2x^5 + 5x^7 + 14x^9 + \ldots = \sum_{n>0} r(n)x^n. \tag{15}$$

Next, we try to find the coefficients $r(n)$ *of the generating function* $R(x)$ *based on its closed-form expression* (15).

For example, the generating function $R(x)$ *can be represent as the following composition of generating functions:*

$$R(x) = xC(A(x)), \tag{16}$$

where

$$C(x) = \sum_{n\geq 0} C_n x^n = \frac{1 - \sqrt{1 - 4x}}{2x}, \quad C_n = \frac{1}{n+1}\binom{2n}{n},$$

$$A(x) = \sum_{n\geq 0} a(n)x^n = x^2, \quad a(n) = \delta(n, 2).$$

In these generating functions, C_n *is the* n*-th Catalan number and* $\delta(i, j)$ *is the Kronecker delta function that equals* 1 *when* $i = j$ *and* 0 *otherwise.*

The composita of the generating function $A(x)$ is

$$A^{\Delta}(n,k) = [x^n]A(x)^k = [x^n](x^2)^k = [x^n]x^{2k} = \delta(n,2k).$$

According to the methodology of compositae of generating functions, the coefficients of $R(x)$ represented as the composition (16) can be calculated by

$$r(n) = \sum_{k=1}^{n-1} A^{\Delta}(n-1,k)C_k = \sum_{k=1}^{n-1} \delta(n-1,2k)C_k = \begin{cases} C_{(n-1)/2} & n \text{ is odd;} \\ 0 & \text{otherwise.} \end{cases}$$

Step 5: *Hence, applying (9), we get*

$$s(n) = r(n+1) = \begin{cases} 0 & n \text{ is odd;} \\ C_{n/2} & \text{otherwise.} \end{cases}$$

Next, we consider an alternative way described in the proposed method.

Step 4.2: *We transform the functional equation (14) into the following form:*

$$\frac{R(x)}{1+R(x)^2} = x.$$

Therefore, we obtain $H(R(x)) = x$ (Case 1), where

$$H(x) = \sum_{n>0} h(n)x^n = \frac{x}{1+x^2}.$$

To simplify the calculations, we use the following generating function:

$$G(x) = \frac{x}{H(x)} = 1 + x^2.$$

Using the binomial theorem, we derive the following expression for its k-th power:

$$G(x)^k = \sum_{n\geq 0} g(n,k)x^n = (1+x^2)^k = \sum_{j\geq 0} \binom{k}{j} x^{2j}.$$

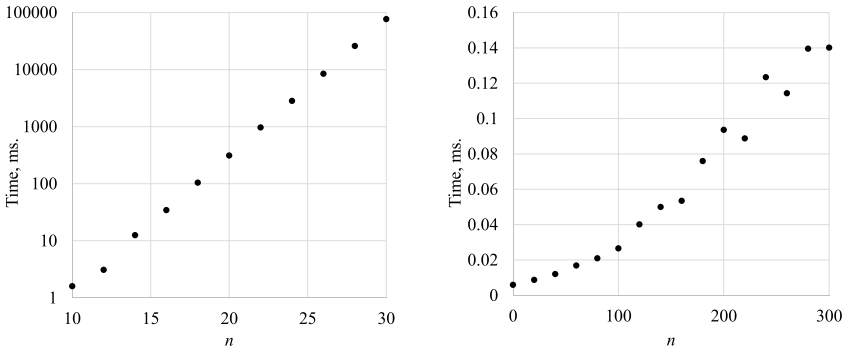
This can also be rewritten as follows:

$$g(n,k) = \begin{cases} 0 & n \text{ is odd;} \\ \binom{k}{\frac{n}{2}} & \text{otherwise.} \end{cases} \tag{17}$$

Step 5: *Combining (17) with (3), (4) and (9), we obtain the following formula for calculating values of $s(n)$:*

$$s(n) = r(n+1) = \frac{1}{n+1}g(n,n+1) = \begin{cases} 0 & n \text{ is odd;} \\ \dfrac{1}{n+1}\binom{n+1}{\frac{n}{2}} & \text{otherwise.} \end{cases} \tag{18}$$

To experimentally confirm the efficiency of the obtained explicit formulas over the recurrent one, the evaluation time of calculating the values of $s(n)$ was measured. We implemented these formulas in the computer algebra system Maxima on a laptop (Intel i7-9750H, 2.6 GHz, Windows 10, 64 bit). Figure 1 presents the results of this computational experiment. Here we can clearly see the exponential increase in computation time for the recurrent formula and the polynomial increase in computation time for the explicit formula.



(**a**) (**b**)

FIGURE 1. Average time for calculating values of $s(n)$: (**a**) based on (12); (**b**) based on (18).

Example 3.2. *Let us consider the following unambiguous context-free grammar:*

$$G = (T, N, S, P), \quad T = \{(,), [,], \varepsilon\}, \quad N = \{S, M\},$$
$$P = \{S \to (S)S \,|\, [M]S \,|\, \varepsilon,\ M \to (M \,|\, (S)M \,|\, [M]M \,|\, \varepsilon\}.$$

Table 2 presents examples for several first values of $s(n)$ and the corresponding set of all generated words.

TABLE 2. Several first values of $s(n)$ and the corresponding set of all generated words.

n	$s(n)$	***Generated words***
0	1	ε
1	0	–
2	2	( ), [ ]
3	1	[( ]
4	9	( ) ( ), [ ] ( ), ( ) [ ], [ ] [ ], (( )), ([ ]), [( )], [[ ]], [(( ]

Step 1: *Based on the production rules $S \to (S)S \,|\, [M]S \,|\, \varepsilon$, we obtain the following functional equation:*

$$S(x) = x^2 S(x)^2 + x^2 M(x) S(x) + 1.$$

Based on the production rules $M \to (M \,|\, (S)M \,|\, [M]M \,|\, \varepsilon$, we obtain the following functional equation:

$$M(x) = xM(x) + x^2 S(x) M(x) + x^2 M(x)^2 + 1.$$

Thus, we have the system of two functional equations with two generating functions

$$\begin{cases} S(x) = x^2 S(x)^2 + x^2 M(x) S(x) + 1; \\ M(x) = xM(x) + x^2 S(x) M(x) + x^2 M(x)^2 + 1. \end{cases}$$

Step 2: *Next, we transform these functional equations into a single equation that includes only one generating function $S(x)$:*

$$x^3S(x)^3-(2x^2+x)S(x)^2+(x+1)S(x)-1=0. \tag{19}$$

Step 3: *Since $s(0)\neq 0$, we make the substitution $S(x)=\frac{R(x)}{x}$ in the functional equation* (19) *and get*

$$x^3\left(\frac{R(x)}{x}\right)^3-(2x^2+x)\left(\frac{R(x)}{x}\right)^2+(x+1)\left(\frac{R(x)}{x}\right)-1=0$$

or

$$xR(x)^3-(2x+1)R(x)^2+(x+1)R(x)-x=0 \tag{20}$$

Step 4.1: *The closed-form solution for the functional equation* (20) *with respect to $R(x)$ is*

$$\left(\frac{\sqrt{-3}-1}{2}\right)\left(\frac{\sqrt{23x^4-6x^3+5x^2+2x-1}}{2\sqrt{27}x^2}+\frac{25x^3-3x^2+3x+2}{54x^3}\right)^{\frac{1}{3}}-$$

$$-\left(\frac{\sqrt{-3}+1}{2}\right)\left(\frac{\sqrt{23x^4-6x^3+5x^2+2x-1}}{2\sqrt{27}x^2}+\frac{25x^3-3x^2+3x+2}{54x^3}\right)^{-\frac{1}{3}}\times$$

$$\times\left(\frac{x^2+x+1}{9x^2}\right)+\frac{2x+1}{3x}=x+x^3+2x^5+5x^7+14x^9+\ldots=\sum_{n>0}r(n)x^n.$$

This closed-form solution was obtained using the computer algebra system Maxima. Due to the extreme complexity of the resulting expression, it is very difficult to obtain an explicit formula for the coefficients $r(n)$ of the generating function $R(x)$. Next, we consider an alternative way described in the proposed method.

Step 4.2: *We transform the functional equation* (20) *into the following form:*

$$\frac{R(x)^2-R(x)}{R(x)^3-2R(x)^2+R(x)-1}=x.$$

Therefore, we obtain $H(R(x))=x$ (Case 1), where

$$H(x)=\sum_{n>0}h(n)x^n=\frac{x^2-x}{x^3-2x^2+x-1}.$$

To simplify the calculations, we use the following generating function:

$$G(x)=\frac{x}{H(x)}=\frac{x^3-2x^2+x-1}{x-1}=x(x-1)+\frac{1}{1-x}.$$

Using the binomial theorem, we derive the following expression for its k-th power:

$$G(x)^k=\sum_{n\geq 0}g(n,k)x^n=\left(x(x-1)+\frac{1}{1-x}\right)^k=(A(x)+B(x))^k=$$

$$=\sum_{j=0}^{k}\binom{k}{j}A(x)^jB(x)^{k-j},$$

where

$$A(x) = x(x-1), \quad A^{\Delta}(n,k) = [x^n]A(x)^k = [x^n]\left(x^2 - x\right)^k = (-1)^n \binom{k}{n-k},$$

$$B(x) = \frac{1}{1-x}, \quad b(n,k) = [x^n]B(x)^k = [x^n]\left(\frac{1}{1-x}\right)^k = \binom{n+k-1}{k-1}.$$

According to the methodology of compositae of generating functions, the coefficients $g(n,k)$ for $G(x) = A(x) + B(x)$ can be calculated by

$$g(n,k) = \sum_{j=0}^{k} \binom{k}{j} \sum_{i=j}^{n} A^{\Delta}(i,j)\, b(n-i, k-j). \tag{21}$$

Step 5: *Combining* (21) *with* (3), (4) *and* (9), *we obtain the following formula for calculating values of $s(n)$:*

$$s(n) = r(n+1) = f(n+1) = \frac{1}{n+1} g(n, n+1) =$$

$$= \frac{1}{n+1} \sum_{j=0}^{n} \binom{n+1}{j} \sum_{i=j}^{n} (-1)^i \binom{j}{i-j} \binom{2n-i-j}{n-j}.$$

Example 3.3. *Let us consider the following unambiguous context-free grammar:*

$$G = (T, N, S, P), \quad T = \{+, \times, (,), a\}, \quad N = \{S, M, E\},$$

$$P = \{S \to S + M \,|\, M, M \to M \times E \,|\, E, E \to a \,|\, (S)\}.$$

Step 1: *Based on the production rules $S \to S + M \,|\, M$, we obtain the following functional equation:*

$$S(x) = xS(x)M(x) + M(x). \tag{22}$$

Based on the production rules $M \to M \times E \,|\, T$, we obtain the following functional equation:

$$M(x) = xM(x)E(x) + E(x). \tag{23}$$

Based on the production rules $E \to a \,|\, (S)$, we obtain the following functional equation:

$$E(x) = x + x^2 S(x). \tag{24}$$

Thus, we have the system of three functional equations with three generating functions

$$\begin{cases} S(x) = xS(x)M(x) + M(x); \\ M(x) = xM(x)E(x) + E(x); \\ E(x) = x + x^2 S(x). \end{cases}$$

Step 2: *Next, we transform these functional equations into a single equation that includes only one generating function $S(x)$. To do this, we express $M(x)$ from* (23) *in terms of $E(x)$ and substitute* (24) *into the resulting expression:*

$$M(x) = \frac{E(x)}{1 - xE(x)} = \frac{x + x^2 S(x)}{1 - x^2 - x^3 S(x)}. \tag{25}$$

Applying (25) *to* (22)*, after simplification we obtain the following functional equation that includes only one generating function* $S(x)$*:*

$$2x^3S(x)^2 + (3x^2 - 1)S(x) + x = 0. \tag{26}$$

Step 3: *Since* $s(0) = 0$ *(it is not possible to derive an empty word), we make the substitution* $S(x) = R(x)$ *in the functional equation* (26) *and get*

$$2x^3R(x)^2 + (3x^2 - 1)R(x) + x = 0. \tag{27}$$

Step 4.1: *The closed-form solution for the functional equation* (27) *with respect to* $R(x)$ *is*

$$R(x) = \frac{1 - 3x^2 - \sqrt{x^4 - 6x^2 + 1}}{4x^3} = x + 3x^3 + 11x^5 + 45x^7 + \ldots = \sum_{n\geq 0} r(n)x^n.$$

Suppose that we can not represent the generating function $R(x)$ *as a composition of simpler ones. Then, we consider an alternative way described in the proposed method.*

Step 4.2: *We transform the functional equation* (27) *into the following form:*

$$(1 - 3x^2)R(x) - 2x^3R(x)^2 = x.$$

Therefore, we obtain $H(x, R(x)) = x$ *(Case 2), where*

$$H(x, y) = \sum_{n>0}\sum_{m>0} h(n, m)x^n y^m = (1 - 3x^2)y - 2x^3y^2.$$

Using the binomial theorem, we derive the following expression for the composita of the generating function $H(x, y)$*:*

$$H(x, y)^k = \sum_{n>0}\sum_{m>0} H^{\Delta}(n, m, k)x^n y^m = y^k((1 - 3x^2) - 2x^3y)^k =$$

$$= y^k \sum_{n\geq 0} \binom{k}{n}(1 - 3x^2)^n(-2x^3y)^{k-n} =$$

$$= y^k \sum_{n\geq 0} \binom{k}{n} \sum_{m\geq 0} \binom{n}{m}(-3x^2)^m(-2x^3y)^{k-n} =$$

$$= \sum_{n\geq 0}\sum_{m\geq 0} \binom{k}{n}\binom{n}{m}(-3)^m(-2)^{k-n}x^{2m+3(k-n)}y^{2k-n}.$$

This can also be rewritten as follows:

$$H^{\Delta}(n, m, k) =$$

$$= \begin{cases} 0 & (n - m + k) \text{ is odd;} \\ 1 & n = m = k = 0; \\ \binom{k}{m-k}\binom{2k-m}{\frac{-n+m+k}{2}}(-3)^{\frac{n-m+k}{2}}\left(\frac{2}{3}\right)^{m-k} & \text{otherwise.} \end{cases}$$

Step 5: *Combining* $H^{\Delta}(n, m, k)$ *with* (5)*,* (7) *and* (8)*, we can calculate the values of* $s(n)$*.*

4. Discussion

Context-free grammars are one of the possible mathematical tools that allow generating combinatorial objects in the form of words. In addition, when working with sets of combinatorial objects, it becomes necessary to solve problems of enumerative combinatorics. Thus, it is necessary to obtain a formula that can calculate the number of elements in a given combinatorial set. The presented study is devoted to solving this problem by finding explicit formulas for calculating the cardinality function of a combinatorial set.

Generating functions are a widely used tool in the field of enumerative combinatorics. To solve this problem, it is necessary to construct a generating function associated with the formal grammar and find an explicit formula for calculating its coefficients. The following were selected as the main methods used in this study: the Lagrange inversion formula and compositae of generating functions.

Thus, the main contribution of this article is the proposed method for obtaining explicit formulas for calculating the number of words with a fixed length, which are derived from a given unambiguous context-free grammar. A unique feature of the proposed method is the combination of methods based on the Lagrange inversion formula and compositae of generating functions, and their application in the field of context-free grammars. The formulas obtained by using the proposed method can find their application in solving the problem of developing combinatorial generation algorithms. For example, to solve practical problems related to data compression (by encoding information through the use of ranking and unranking algorithms) or modeling complex discrete structures (by constructing discrete structures through the use of random generation algorithms). In this case, the proposed method can be applied to calculate the number of discrete structures in a given combinatorial set.

References

[1] Chavan, P.V.; Jadhav, A. *Automata Theory and Formal Languages*; Academic Press: Cambridge, USA, 2023.

[2] Kreher, D.L.; Stinson, D.R. *Combinatorial Algorithms: Generation, Enumeration, and Search*; CRC Press: Boca Raton, USA, 1999.

[3] Ruskey, F. Combinatorial Generation; 2003. Available online: https://page.math.tu-berlin.de/ felsner/SemWS17-18/Ruskey-Comb-Gen.pdf (accessed on 1 March 2025).

[4] Goldberg, A.V.; Sipser, M. Compression and ranking. *SIAM J. Comput.* **1991**, *20*, 524–536.

[5] Huynh, D.T. The complexity of ranking simple languages. *Mathematical Systems Theory* **1990**, *23*, 1–19.

[6] Lange, K.-J.; Rossmanith, P.; Rytter, W. Parallel recognition and ranking of context-free languages. In Proceedings of the 17th Symposium on Mathematical Foundations of Computer Science, Prague, Czechoslovakia, 24–28 August 1992.

[7] Makinen, E. Ranking and unranking left Szilard languages. *Int. J. Comput. Math.* **1998**, *68*, 29–38.

[8] Luchaup, D.; Shrimpton, T.; Ristenpart, T.; Jha, S. Formatted encryption beyond regular languages. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, USA, 3–7 November 2014.

[9] Naganuma, H.; Hendrian, D.; Yoshinaka, R.; Shinohara, A.; Kobayashi, N. Grammar compression with probabilistic context-free grammar. In Proceedings of the Data Compression Conference, Snowbird, USA, 24–27 March 2020.

[10] Hickey, T.; Cohen, J. Uniform random generation of strings in a context-free language. *SIAM J. Comput.* **1983**, *12*, 645–655.
[11] Mairson, H.G. Generating words in a context-free language uniformly at random. *Inform. Process. Lett.* **1994**, *49*, 95–99.
[12] Weinberg, F.; Nebel, M.E. Non uniform generation of combinatorial objects; 2010. Available online: https://d-nb.info/1027389740/34 (accessed on 1 March 2025).
[13] Nebel, M.E.; Scheid, A.; Weinberg, F. Random generation of RNA secondary structures according to native distributions. *Algorithms for Molecular Biology* **2011**, *6*, 24.
[14] Onokpasa, E.; Wild, S.; Wong, P.W.H. RNA secondary structures: from ab initio prediction to better compression, and back. In Proceedings of the Data Compression Conference, Snowbird, USA, 21–24 March 2023.
[15] Vastarini, F.; Plump, D. Random graph generation in context-free graph languages. In Proceedings of the 13th International Workshop on Developments in Computational Models, Rome, Italy, 2 July 2023.
[16] Amaya, J.A.Z. Shaping test inputs in grammar-based fuzzing. In Proceedings of the 33rd International Symposium on Software Testing and Analysis, Vienna, Austria, 16–20 September 2024.
[17] Miracle, S.; Yilek, S. Targeted invertible pseudorandom functions and deterministic format-transforming encryption. In Proceedings of the Cryptographer's Track at the RSA Conference, San Francisco, USA, 24–27 April 2023.
[18] Bellare, M.; Ristenpart, T.; Rogaway, P.; Stegers, T. Format-preserving encryption. In Proceedings of the 16th Workshop on Selected Areas in Cryptography, Calgary, Canada, 13–14 August 2009.
[19] Martinez, C.; Molinero, X. A generic approach for the unranking of labeled combinatorial classes. *Random Structures Algorithms* **2001**, *19*, 472–497.
[20] Shablya, Y.; Kruchinin, D.; Kruchinin, V. Method for developing combinatorial generation algorithms based on AND/OR trees and its application. *Mathematics* **2020**, *8*, 962.
[21] Srivastava, H.M.; Manocha, H.L. *A Treatise on Generating Functions*; Ellis Horwood Limited: Chichester, UK, 1984.
[22] Stanley, R.P. *Enumerative Combinatorics*, 2nd ed.; Cambridge University Press: New York, USA, 2012.
[23] Kim, D.S.; Kim, T.; Kwon, J. Study on degenerate Stirling numbers. *European Journal of Pure and Applied Mathematics* **2024**, *17*, 1–10.
[24] Colombo, F.; Krausshar, R.S.; Sabadini, I.; Simsek, Y. On the generating functions and special functions associated with superoscillations. *Discrete Appl. Math.* **2023**, *340*, 215–227.
[25] Costabile, F.A.; Gualtieri, M.I.; Napoli, A. Towards the centenary of Sheffer polynomial sequences: old and recent results. *Mathematics* **2022**, *10*, 4435.
[26] Akhtamova, S.S.; Cuchta, T.; Lyapin, A.P. An approach to multidimensional discrete generating series. *Mathematics* **2024**, *12*, 143.
[27] Bertoni, A.; Goldwurm, M.; Sabadini, N. Computing the counting function of context-free languages. In Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science, Passau, FRG, 19–21 February 1987.
[28] Zhou, R.R.; Yeh, J.; Ren, F. Context-free grammars for several triangular arrays. *Axioms* **2022**, *11*, 297.
[29] Chen, W.Y.C.; Fu, A.M. A context-free grammar for the e-positivity of the trivariate second-order Eulerian polynomials. *Discrete Mathematics* **2022**, *345*, 112661.
[30] Yang, H.R. A context-free grammar associated with Fibonacci and Lucas sequences. *J. Math.* **2023**, *2023*, 6497710.
[31] Meduna, A. *Formal Languages and Computation: Models and Their Applications*; Auerbach Publications: New York, USA, 2014.
[32] Hopcroft, J.E.; Motwani, R.; Ullman, J.D. *Introduction to Automata Theory, Languages, and Computation*; Addison-Wesley: Boston, USA, 2001.
[33] Panholzer, A. Grobner bases and the defining polynomial of a context-free grammar generating function. *J. Autom. Lang. Comb.* **2005**, *10*, 79–97.

[34] Stanley, R.P. *Enumerative Combinatorics*, Volume 2; Cambridge University Press: New York, USA, 2001; pp. 36–44.
[35] Kruchinin, D.V.; Kruchinin, V.V. A method for obtaining generating functions for central coefficients of triangles. *J. Integer Seq.* **2012**, *15*, 12.9.3.
[36] Kruchinin, D.; Kruchinin, V.; Shablya, Y. Method for obtaining coefficients of powers of bivariate generating functions. *Mathematics* **2021**, *9*, 428.
[37] Kruchinin, D.; Kruchinin, V.; Shablya, Y. Method for obtaining coefficients of powers of multivariate generating functions. *Mathematics* **2023**, *11*, 2859.

Laboratory of Algorithms and Technologies for Discrete Structures Research, Tomsk State University of Control Systems and Radioelectronics, Tomsk, Russia
Email address: shablya-yv@mail.ru

Laboratory of Algorithms and Technologies for Discrete Structures Research, Tomsk State University of Control Systems and Radioelectronics, Tomsk, Russia
Email address: kru@2i.tusur.ru

Laboratory of Algorithms and Technologies for Discrete Structures Research, Tomsk State University of Control Systems and Radioelectronics, Tomsk, Russia; Sirius University of Science and Technology, Krasnodar region, Russia
Email address: kruchinindm@gmail.com