AN ABSTRACT OF THE THESIS OF

Leonard Frank Klosinski for the M.A. in Mathematics
(Name) (Degree) (Department)

Date thesis is presented May 2, 1963

Title THE ENUMERATION OF STRINGS OF A GIVEN LENGTH IN AN N-ARY NON-ASSOCIATIVE NON-COMMUTATIVE ALGEBRA

Abstract approved Redacted for privacy
(Major professor)

In his book on abstract algebra, Nathan Jacobson poses and solves the problem of finding the number of ways of inserting parentheses in a string of given length with binary operators. We continue the work of Jacobson and go beyond it in that we no longer consider one binary operator but general n-ary operators chosen from a given set of p n-ary operators. Furthermore, the algebra we consider is neither associative nor commutative. We thus obtain a count of the number of strings of given length where the strings contain any of the given p n-ary operators.

THE ENUMERATION OF STRINGS OF A GIVEN LENGTH IN AN N-ARY NON-ASSOCIATIVE NON-COMMUTATIVE ALGEBRA

by

LEONARD FRANK KLOSINSKI

A THESIS

submitted to

OREGON STATE UNIVERSITY

in partial fulfillment of
the requirements for the
degree of

MASTER OF ARTS

June 1963

APPROVED:

Redacted for privacy

Professor of Mathematics

In Charge of Major

Redacted for privacy

Chairman of Department of Mathematics

Redacted for privacy

Dean of Graduate School

Date thesis is presented May 2, 1963

Typed by Ola Gara

DEDICATION

To those who believed.

TABLE OF CONTENTS

THE ENUMERATION OF STRINGS OF A GIVEN LENGTH IN AN N-ARY NON-ASSOCIATIVE NON-COMMUTATIVE ALGEBRA

CHAPTER I

INTRODUCTION

The digital computer has come of age; it is an indispensable scientific tool; it is a highly sophisticated instrument.

Sophisticated! It is a big word, but it is appropriate to use just such a word to describe the maze of wires, cores, tubes and transistors that make up the computer of today. But what does sophisticated mean?

At one time the word was reserved for descriptions of man. A man was sophisticated if he attended the opera. Some time later, sophistication meant knowing when and with what volume to applaud the diva. But sophistication always called for, and still calls for, greater refinement in manners.

Mathematicians then applied the word to the solution of problems. A solution, they said, is sophisticated if it uses ingenious and powerful methods to achieve its end.

And today, we say a computer is sophisticated. As with man, sophistication requires refinements; and the computer has indeed undergone a considerable number of refinements since the days of mechanical relays. The mathematician can also call the computer sophisticated since it is so designed or programmed that it uses highly ingenious methods to arrive at solutions.

While this high degree of sophistication put Colonel Glenn into orbit, raised the premium on our insurance policies, and helped the businessman and scientist in innumerable ways, it has also carried along new and more difficult problems.

Let us assume that in order to complete a calculation this expression must be evaluated

$$ax^3 + bx^2 + cx + d.$$

Since our computer is sophisticated, our problem will be written in a special language such as FORTRAN, ALGOL, or ALCOM, and fed into the computer. At this point, automatic procedures take over. Let us assume the problem is solved using the following calculations and steps:

x

cx

$$x^2$$

$$bx^2$$

$$x^3$$

$$ax^3$$

$$ax^3 + bx^2$$

$$(ax^3 + bx^2) + cx$$

$$((ax^3 + bx^2) + cx) + d.$$

It is seen that this procedure involves five multiplications and three additions.

Suppose that using a different language or a different means of interpreting the language leads to the following calculations and steps:

$$x$$

$$ax$$

$$ax + b$$

$$x(ax + b)$$

$$x(ax + b) + c$$

$$x(x(ax + b) + c)$$

$$x(x(ax + b) + c) + d.$$

This procedure requires only three multiplications and three additions. Thus it is faster, less expensive, and therefore, more desirable, and, from one point of view, more sophisticated.

We have seen two methods for solving the same problem; others may exist. One or more of the methods are better, i.e., faster, more accurate, than the others. We would like to know the best method, and in order to know this we must know all the methods.

Very briefly, let us look at another problem. In general, because of the finite length of registers in a computer, and the consequent roundoff, it is true that on a computer

$$(ab)/c \neq a(b/c).$$

Which of (ab)/c or a(b/c), if any, is better? Which is more accurate? Which is faster?

The questions that have been posed are important in both computer design and computer language. The answer to these questions is not trivial, and the problem is more than choosing one or two procedures out of a set of known methods; for, in general, we do not know all the methods.

What is the best procedure for forming the sum of

$$a + b + c + d + e?$$

Is it

$$(((a + b) + c) + d) + e$$

or

$$((a + (b + c)) + d) + e$$

or

$$a + (b + (c + (d + e)))$$

or perhaps some other method? This last problem gives a hint to a more basic question, and that is: how many distinct methods of summing five elements exist?

With a little time and patience, it is seen that there are fourteen methods. But suppose there are twelve elements or twenty or seventy-five; how many distinct methods are there? Or if, in the place of binary operators, we are concerned with ternary operators, or some other type of operator, what then?

While this paper does not solve the problem of finding a best method, in general, it is a start on the problem of evaluation of a string with arbitrary but fixed operators. Before the optimalization, comes the counting problem. And it is precisely this problem that we solve in the following pages.

CHAPTER II

THE COUNTING PROBLEM

The introduction has given us a glance at one of the problems the automatist must face. Now, however, we leave the realm of the computer and turn to the abstract mathematical problem. Applications, though never completely out of sight, are put aside; the idiosyncrasies of the real world are forgotten, and we step through the looking glass into pure mathematics. We do this in order that our results should not seem narrow or restrictive, and because we believe the theoretical aspect of the subject carries its own appeal.

With this in mind, our problem becomes that of finding the number of strings of length k in an n-ary, non-associative, non-commutative algebra.

As with any subject, a question of terminology quickly arises. We give some answer to that question by defining a string as an algebraic expression containing elements and operators composed in a meaningful manner. By length, we mean the total number of elements and operators occurring in the string. Thus

$$a + b$$

has length three, and

$$a + b \cdot c/d$$

has length seven.

The last of the new terms introduced is form. A form is an equivalence class of strings. Two strings are said to have the same form if, when written in Lukasiewicz notation, an element occurs in the second string whenever an element occurs in the first string and an operator occurs in the second string whenever an operator occurs in the first string.

To further answer the question of terminology, we introduce the notation

$$L_k(p_n)$$

to mean the number of strings of length k with p n-ary operators. We further define the symbol

$$Z(r_n, s)$$

to be the set of strings of length r + s with r spaces for n-ary operators and s spaces for elements. Note that the symbols r and r_n will be used interchangeably; the latter symbol will be used primarily to emphasize the type of operator under consideration.

With these definitions at our disposal, let us now consider strings with one ternary operator, that is, strings in $Z(r_3, s)$. For simplicity, the ternary operation on a, b and c will be written as

abc.

Further

(abc)de

will mean the operation is first performed on a, b, and c, and then on (abc), d, and e. Similar meanings will apply if there are more than five elements.

The first question that arises asks what values r and s may have, or what is similar, what values can k take on? It is immediately obvious that k cannot take on all values. For example, if k were 2, we would be looking for a string in $Z(r_3, s)$ such that

$$r + s = 2.$$

The various possibilities for r and s are

r	0	1	2
s	2	1	0.

None of these combinations gives a string which has meaning.

Since we are concerned with strings of positive length, the smallest value of k is 1. A string of length 1 would contain one element and no operators, and would be of the form

a.

It has been shown above that there exist strings in $Z(1_3, 3)$ and $Z(2_3, 5)$. Thus we have seen that k can equal 1, 4, 7. And in general, the length of a string with ternary operators is of the form 3m - 2. Furthermore, the string is in $Z[(m - 1)_3, 2m - 1]$.

The proof of the previous statements follows from these arguments. Since the ternary operators operate one at a time, let o_i denote the i-th operator in the operating sequence. Note that after a ternary operator is applied to three elements, it reduces the number of the elements by two.

Suppose there are 2m elements in a string. Then after o_1 operates on the string with 2m elements, there remain 2m - 2 elements. After o_2 is applied, the length of the string is again reduced by 2, and 2m - 4 elements remain. Thus after o_{m-1} operates on the string, there remain 2m - 2(m - 1) or 2 elements. Another ternary

operator cannot be applied since only two elements remain. And if fewer than m - 1 operators were applied, more than two elements would remain, and additional ternary operators would be required to give meaning to the string. Thus by starting with 2m elements, we are left with a string without meaning.

Assume that there are 2m - 1 elements in the string. Since each successive application of operators reduces the number of elements by two, it is clear that after m - 1 operators are applied, there remains but a single element which is the result of the calculation. If fewer than m - 1 operators were applied, more than one element would remain, thereby leaving an expression without meaning.

Thus we see that there must be 2m - 1 elements and m - 1 operators when we consider strings with ternary operators. And this by definition says the string is in $Z[(m-1)_3, 2m-1]$. Since the length of the string is k or r + s, we have that

$$k = (m-1) + (2m-1) = 3m-2.$$

In order to find $L_k(l_n)$, it will again be necessary to know what values k, r, and s may assume. It is again clear that not all values for k are permissable, but which are?

An n-ary operator requires n elements on which to operate. When it is applied to the n elements, it yields one element. Thus, if a string has s elements, $s > n$, the application of one n-ary operator to the string will reduce the number of elements by $n - 1$.

In a string, any number of operators can occur provided there is a sufficient number of elements on which to operate. Let us consider the string in which $m - 1$ ($m = 1, 2, \ldots, j$) n-ary operators occur. We wish to find the number of elements required. We write

$$s = am + b.$$

Since each n-ary operator reduces the $am + b$ elements by $n - 1$, after $m - 1$ operators are applied, it must be true that

$$am + b - (m - 1)(n - 1) = 1.$$

Equating like terms, we have that

$$a = n - 1$$

$$b = -(n - 2).$$

Thus we have that

$$s = (n - 1)m - (n - 2)$$

$$r = m - 1$$

and

$$k = [(n - 1)m - (n - 2)] + (m - 1)$$

$$= nm - (n - 1).$$

Knowing the length of strings is the beginning of the solution of the problem. But now, the counting problem begins in earnest. We once more consider strings in $Z(r_3, s)$. The first strings that occur are in $Z(0_3, 1)$. The only form of such a string is

a.

If we have t elements, each of the elements could be written in place of a. Thus our first result is

$$L_1(1_3) = t.$$

The next strings to consider are in $Z(1_3, 3)$. Once again only one form is possible and that is

abc.

Since our algebra is non-commutative, any of the t elements could replace a, any of them could replace b, and any could replace c. Therefore

$$L_4(1_3) = t^3.$$

We point out that in the general case in which there occur s spaces for elements, any of the given t elements, because of non-commutativity, can replace any of the elements in the s spaces. Therefore, each form of a string will have t^s meanings.

Strings in $Z(0_3, 1)$ and $Z(1_3, 3)$ had but one form. The number of forms of strings increases as r + s becomes larger. As examples, strings in $Z(2_3, 5)$ have three possible forms:

ab(cde)

a(bcd)e

(abc)de.

And strings in $Z(3_3, 7)$ have these twelve forms

ab(cd(efg))

ab(c(def)g)

ab((cde)fg)

a(bc(def))g

a(b(cde)f)g

a((bcd)ef)g

a(bcd)(efg)

(ab(cde))fg

(a(bcd)e)fg

((abc)de)fg

(abc)d(efg)

(abc)(def)g.

We introduce the symbol f_i to mean the number of forms with i elements. It has been shown that

$$f_1 = 1$$

$$f_3 = 1$$

$$f_5 = 3$$

$$f_7 = 12.$$

And once f_{2k-1} is found, we immediately have $L_{3K-2}(l_3)$. We proceed to search for f_{2k-1}.

Let us look at the placement of parentheses in a string with 2k - 1 elements

$$a_1a_2a_3\cdots a_{2k-2}a_{2k-1},$$

and assume we know the values for f_i, $i < 2k - 1$.

Since it is desired to find f_{2k-1}, there would not be any meaning if we placed the parentheses thus:

$$(a_1a_2a_3\cdots a_{2k-2}a_{2k-1}).$$

Before deciding how to place parentheses, consider how a ternary operator works. It operates on three elements and reduces them to one element. When a certain number of operators is applied to the string with 2k - 1 elements, the string is reduced to one element. Note that the step of obtaining that final element is the application of a ternary operator to three elements. Thus the 2k - 1 elements must first be reduced to three elements, and this is what we wish to consider. In what various ways can the 2k - 1 elements be grouped into three groups, each group

containing an odd number of elements? The first group can have one element, three elements, five and so on up to $2k - 3$ elements. The second and third groups can also contain one, three and so on up to $2k - 3$ elements. The only requirement is that the sum of the elements in the three groups be $2k - 1$. The following table gives us some of the possible groupings:

Group 1	Group 2	Group 3
1	1	$2k - 3$
1	3	$2k - 5$
1	5	$2k - 7$
3	3	$2k - 7$
5	7	$2k - 13$.

It is important to notice that there is a symmetry among the groups, and that the groups can be permuted. Thus the above table, with the columns interchanged, will be repeated in a complete listing.

Let us look at two cases. The first case will have the grouping $(1, 1, 2k - 3)$:

$$a_1 a_2 (a_3 \cdots a_{2k-1}).$$

There are f_1 possible forms for the elements in the first group, f_1 forms for the elements in the second group, and

f_{2k-3} forms for the elements in group three. Since the forms the elements in any group take are independent of the forms the elements take in any other group, the total number of forms for

$$a_1a_2(a_3\cdots a_{2k-1})$$

is

$$f_1f_1f_{2k-3}.$$

Consider the grouping (3, 3, 2k - 7):

$$(a_1a_2a_3)(a_4a_5a_6)(a_7\cdots a_{2k-1}).$$

In the first group there are f_3 possible forms, in the second f_3, and in the third f_{2k-7}. By the same reasoning as above, the total number of forms is

$$f_3f_3f_{2k-7}.$$

Since the two groupings are exclusive, that is, the grouping (3, 3, 2k - 7) cannot occur at the same time as the grouping (1, 1, 2k - 3), we have that the total number of forms for the two groupings is

$$f_1f_1f_{2k-3} + f_3f_3f_{2k-7}.$$

We continue this process for all possible groupings and we have that

$$f_{2k-1} = f_1f_1f_{2k-3} + f_1f_3f_{2k-5} + f_1f_5f_{2k-7}$$
$$+ \cdots + f_1f_{2k-5}f_3 + f_1f_{2k-3}f_1 + f_3f_1f_{2k-5}$$

$$+ f_3 f_3 f_{2k-7} + \cdots + f_3 f_{2k-5} f_1$$

$$+ \cdots + f_{2k-3} f_1 f_1 .$$

At this point, we have f_{2k-1} written as a recursion formula. An explicit formula is desired, but first let us consider the search for a recursion formula in the case of n-ary operators.

As with strings in $Z(0_3, 1)$ and $Z(1_3, 3)$, strings in $Z(0_n, 1)$ and in $Z(1_n, n)$ have but one form. The forms for such strings are respectively

$$a$$

and

$$a_1 a_2 \cdots a_n .$$

From previous investigations, it is known that there occur no strings with less than n elements but more than one element. Then using the f_i notation, we have that

$$f_1 = 1$$

$$f_n = 1.$$

The arguments that follow in finding the recursion formula for $f_{(n-1)k-(n-2)}$ are similar to those used to find f_{2k-1} in the case of ternary operators.

Assuming we know the values for f_i, $i < (n - 1)\, k - (n - 2)$, where i has the form $(n - 1)m - (n - 2)$, consider

the placing of parentheses in a string with $(n - 1)k - (n - 2)$ elements:

$$a_1 a_2 a_3 \cdots a_{(n-1)k-(n-2)}.$$

In the step immediately prior to the application of the last n-ary operator, the $(n - 1)k - (n - 2)$ elements are grouped into n groups. It is clear that the number of elements in each group has the form $(n - 1)m - (n - 2)$. Each group can have 1, n, $2n - 1$ or x elements where x is of the proper form provided that the sum of the elements in the n groups is $(n - 1)k - (n - 2)$. As an example, we list three possible groupings in the following table.

Group 1	Group 2	Group 3	Group 4	...	Group n-1	Group n
1	1	1	1	...	1	$(n-1)(k-1)-(n-2)$
n	1	1	1	...	1	$(n-1)(k-2)-(n-2)$
n	$2n-1$	1	1	...	1	$(n-1)(k-4)-(n-2)$

Each grouping can be written as an n-tuple. We define two n-tuples $(x_1, x_2, \cdots, x_n)$, $(y_1, y_2, \cdots, y_n)$ to be different if the y_i's are not merely a permutation of the x_i's. In order to have a complete listing of all possible groupings, we first consider all different

n-tuples. We then extend the set of different n-tuples to include the permutations of each n-tuple, throwing out those permutations which leave the n-tuple unchanged. This extended set of n-tuples then gives a complete listing of all groupings.

We proceed to investigate two cases: the first with the grouping $[1,1,1,1, \ldots, 1, (n-1)(k-1)-(n-2)]$ and the second with the grouping $[n,1, (n-1)(k-2)-(n-2), 1, \ldots, 1]$.

A string in the first grouping will appear as

$$(a_1)(a_2)(a_3)(a_4)\cdots(a_{n-1})(a_n a_{n+1}\cdots a_{(n-1)k-(n-2)}).$$

(The parentheses are placed around the single elements merely to emphasize that the single element is a group.) The elements in group 1 have f_1 forms. Likewise the elements in each of the next n - 2 groups have f_1 forms. The elements in the n-th group have $f_{(n-1)(k-1)-(n-2)}$ forms. Since the form that the elements take in any group is independent of the form elements in another group take, the total number of forms for the above grouping is

$$f_1 f_1 f_1 f_1 \cdots f_1 f_{(n-1)(k-1)-(n-2)}.$$

The second grouping yields a string of this appearance

$$(a_1 a_2 \cdots a_n)(a_{n+1})(a_{n+2} \cdots a_{(n+1)k-(2n+5)})$$
$$(a_{(n+1)k-(2n+4)}) \cdots (a_{(n-1)k-(n-2)}).$$

With this grouping, group 1 has f_n forms, group 2 has f_1 forms, group 3 has $f_{(n-1)(k-2)-(n-2)}$ forms, and groups 4 to n each have f_1 forms. The total number of forms for the second grouping then is

$$f_n f_1 f_{(n-1)(k-2)-(n-2)} f_1 \cdots f_1.$$

The two groupings are, however, exclusive, and thus the total number of forms for the two groupings is the sum of the number of forms for each grouping.

When all the possible groupings are considered, each grouping is exclusive of the other groupings. Then we have that $f_{(n-1)\ k-(n-2)}$ is equal to the sum of the number of forms for each grouping:

$$f_{(n-1)k-(n-2)} = \sum f_{i_1} f_{i_2} f_{i_3} \cdots f_{i_n}$$

where $i_1, i_2, i_3, \ldots, i_n$ is an n-tuple in the extended set; furthermore, the summation is over every n-tuple in the extended set.

Returning to the case of ternary operators, we next let

$$f = f(x)$$

and

$$f = f_1x + f_3x^3 + f_5x^5 + \ldots + f_{2k-1}x^{2k-1} + \ldots .$$

Cubing both sides gives

$$f^3 = f_1f_1f_1x^3 + (f_1f_1f_3 + f_1f_3f_1 + f_3f_1f_1)x^5$$
$$+ (f_1f_1f_5 + f_1f_3f_3 + f_1f_5f_1 + f_3f_1f_3$$
$$+ f_3f_3f_1 + f_5f_1f_1)x^7 + \ldots .$$

We then apply the recursion formula to the above to obtain

$$f^3 = f_3x^3 + f_5x^5 + f_7x^7 + \ldots$$

and it follows that

$$f^3 = f - f_1x.$$

Since $f_1 = 1$

$$f^3 = f - x.$$

Then by Lagrange's formula for the reversion of a series, we have that

$$f = \sum \frac{x^i}{i!} \left[\frac{d^{i-1}}{df^{i-1}} (1 - f^2)^{-i} \right]_{f=0}$$

which, when written explicitly, is

$$f = x + \frac{x^3}{3!} 2\cdot 3 + \frac{x^5}{5!} \frac{5\cdot 6(4!)}{2!} + \ldots$$
$$+ \frac{x^{2k-1}}{(2k-1)!} \frac{(2k-1)\ldots(3k-3)(2m-2)!}{(k-1)!} + \ldots .$$

Equating coefficients, we obtain

$$f_{2k-1} = \frac{(3k-3)!}{(2k-1)!(k-1)!}.$$

We now have the number of forms of strings with $2k - 1$ elements. Since each element can be replaced by any of the given t elements, our final result is

$$L_{3k-2}(l_3) = \frac{(3k - 3)!}{(2k - 1)!(k - 1)!} t^{2k-1}.$$

Only the general case, the case with n-ary operators, remains. As previously, the method for finding $f_{(n-1)k-(n-2)}$ follows that for finding f_{2k-1}.

Let

$$f = f(x)$$

and

$$(1) \qquad f = f_1 x + f_n x^n + f_{2n-1} x^{2n-1} + \cdots + f_{(n-1)k-(n-2)} x^{(n-1)k-(n-2)} + \cdots .$$

Raising both sides to the n-th power gives

$$\begin{aligned}(2) \quad f^n = {} & f_1 f_1 f_1 \cdots f_1 x^n \\ & + (f_1 f_1 f_1 \cdots f_1 f_n + f_1 f_1 \cdots f_1 f_n f_1 + \cdots \\ & + f_n f_1 f_1 \cdots f_1) x^{2n-1} + (f_1 f_1 \cdots f_1 f_{2n-1} \\ & + f_1 f_1 \cdots f_1 f_{2n-1} f_1 + \cdots + f_{2n-1} f_1 f_1 \cdots f_1 \\ & + f_1 f_1 \cdots f_1 f_n f_n + f_1 f_1 \cdots f_1 f_n f_1 f_n + \cdots \\ & + f_n f_n f_1 \cdots f_1) x^{3n-2} + \cdots .\end{aligned}$$

Before using the recursion formula to simplify the above expression, let us look at the coefficient of $x^{(n-1)k-(n-2)}$ in (2). It is

$$\sum f_{i_1} f_{i_2} f_{i_3} \cdots f_{i_n}$$

or perhaps there are more or fewer terms? Let us assume the term

$$f_{j_1} f_{j_2} f_{j_3} \cdots f_{j_n}$$

is missing, where $j_1 + j_2 + j_3 + \ldots + j_n = (n-1)k - (n-2)$.

Since f_{j_i} is the coefficient of x^{j_i} in (1), the product

$$x^{j_1} x^{j_2} x^{j_3} \cdots x^{j_n} = x^{(n-1)k-(n-2)}$$

and therefore

$$f_{j_1} f_{j_2} f_{j_3} \cdots f_{j_n}$$

does occur in (2) as the coefficient of $x^{(n-1)k-(n-2)}$.

Similarly, there can be no terms not in $\sum f_{i_1} f_{i_2} f_{i_3} \cdots f_{i_n}$ as coefficients of $x^{(n-1)k-(n-2)}$ in (2).

Then applying the recursion formula to (2), we obtain

$$f^n = f_n x^n + f_{2n-1} x^{2n-1} + \ldots$$

$$+ f_{(n-1)k-(n-2)} x^{(n-1)k-(n-2)} + \ldots$$

and further

$$f^n = f - x.$$

We again use the Lagrange formula for the reversion of a series, and obtain

$$f = \sum \frac{x^i}{i!} \left[\frac{d^{i-1}}{df^{i-1}} (1 - f^{n-1})^{-i} \right]_{f=0}$$

Expanding, we have (3)

$$f = x + \frac{x^n}{n!} (n-1)!n + \frac{x^{2n-1}}{(2n-1)!} \frac{(2n-2)!(2n-1)2n}{2!} + \ldots$$

$$+ \frac{x^{(n-1)k-(n-2)}}{[(n-1)k-(n-2)]!} \left[\frac{[k(n-1)-(n-1)]!}{(k-1)!} \right.$$

$$\left. [k(n-1)-(n-2)]\ldots[k(n-1)-(n+k)]] + \ldots \right.$$

Equating the coefficients of $x^{(n-1)k-(n-2)}$ in (1) and (3), we arrive with

$$f_{(n-1)k-(n-2)} = \frac{(nk-n)!}{[(n-1)k-(n-2)]!(k-1)!}.$$

It follows immediately that

$$L_{nk-(n-1)}(1_n) = \frac{(nk-n)!}{[(n-1)k-(n-2)]!(k-1)!} \cdot t^{(n-1)k-(n-2)}.$$

The last case we consider is that in which we are given not one n-ary operator, but a set of p n-ary operators. It is evident that each of the k - 1 spaces for operators can be filled by any of the given p operators. Thus, the final result

$$L_{nk-(n-1)}(p_n) = \frac{(nk-n)!}{[(n-1)k-(n-2)]!(k-1)!}\, t^{(n-1)k-(n-2)}\, p^{k-1}.$$

We have completed the problem which we set out to solve. But before leaving it, let us point out some extensions. In this investigation, our strings contained only n-ary operators. It would be possible to consider strings with n_1-ary , n_2-ary , ... , n_m-ary operators. Then, of course the non-associativity and non-commutativity conditions can be dropped.

The problem grows in complexity. But the challenge it presents is a most inviting one.

BIBLIOGRAPHY

1. Copson, E. T. An introduction to the theory of functions of a complex variable. Oxford, Clarendon Press, 1960. 448 p.

2. Hartman, Philip H. The Lukasiewicz parenthesis-free notation for algebraic expressions, and its application in a Boolean algebraic compiler. Corvallis, Oregon, Oregon State University, Department of Mathematics, July 10, 1962. 36 p.

3. Jacobson, Nathan. Lectures in abstract algebra. Vol. 1. Toronto, D. Van Nostrand, 1951. 217 p.