A Recursive Function for Calculating the Number of Legal Strings of Parentheses and for Calculating Catalan Numbers

Kirke Bent

Parallel Business Software
29 Pine Street
Chatham, NJ 07928, USA

e-mail: parbzsft@bellatlantic.net

Abstract. This paper discusses the number of legal strings of n pairs of parentheses as well as a structure of the set of these strings. As the number of such strings is known to be the Catalan number, a structure of Catalan numbers is thereby developed. A recursive function is developed that counts the set and calculates the Catalan number. The function uses two parameters and is thus a generalization of Catalan numbers.

Key words: Parenthetical strings, recursive functions, stringology, combinatorics, generalized Catalan numbers.

1 Introduction

This paper concerns the problem of calculating the number of legal strings of parentheses that can be constructed from n pairs of parentheses. This number is known to be the Catalan number. There is a large literature of Catalan number interpretations and connections [2, 3, 4, 5, 6, 7]. Stanton and White have a proof of the correspondence between Catalan numbers and legal parenthetical strings[7]. The Catalan number is defined as

$$C_n = \binom{2n}{n} \div (n+1).$$

The ordinary meaning of "legal strings" of parentheses is intended here: 1) The strings are conventionally constructed from left to right. 2) At any point in the string, the number of left parentheses is equal to or greater than the number of right parentheses. 3) all of the $2n$ parentheses are used.

For example, $C_3 = 5$; the legal strings of 3 pairs of parentheses are

((( ))), (( ) ( )), (( ))( ), ( ) (( )), and ( ) ( ) ( ).

The paper offers a way to calculate Catalan numbers with a recursive function and a structure of the strings and the number.

2 Outline

Four areas emerge from consideration of this function:

2.1 A Chart

This is a chart of the construction of the C_n legal parenthetical strings composed of n pairs of parentheses. The number of such strings is an interpretation of Catalan numbers. The chart can be interpreted as a rooted tree. Evaluation of the function counts the leaves of the tree.

2.2 A Function

The function, denoted here by $B_{n,m}$, uses two parameters. The n^{th} Catalan number, C_n, is produced by $B_{n,0}$. The domain of both parameters of $B_{n,m}$ is the non-negative integers. In the recursive descent, m takes on values both higher and lower than n.

2.3 A Generalization

This generalization of Catalan numbers is based on the two parameters. It includes C_n.

2.4 A Structure

This structure of Catalan numbers is suggested by the chart but can be expressed algebraically.

3 Elaboration

3.1 The Chart

The idea behind the chart is simply writing the legal parenthetical expressions according to the definition above.

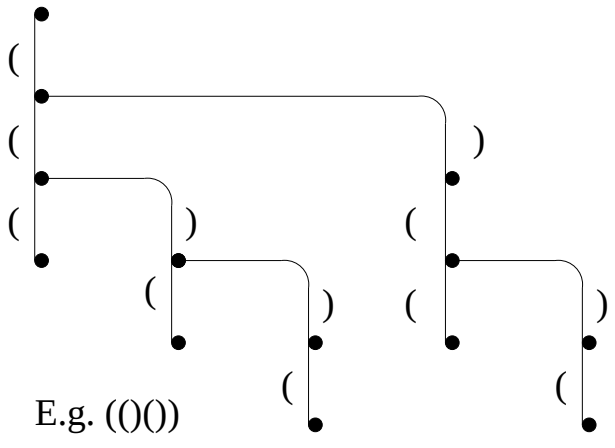

Figure 1: Forming all legal arrangements of 3 pairs of parentheses

Consider this as a rooted tree. Each edge represents adding a parenthesis. If there are two edges descending from a vertex, then there is a choice of adding a left or right parenthesis at that point. By following all paths from the root to a leaf, all legal expressions have been written. Note that final right parentheses are not needed to count leaves.

The steps in drawing the chart are:

1. Start at the top with n pairs of parentheses.

2. Stop if there are no more left parentheses.

3. Draw a vertical line downwards. This represents a left parenthesis and "uses" one. If the number of left parentheses used (before this one was drawn) exceeds the number of right ones used, draw another line from the same starting point but to the right and then curving downwards. This represents a right parenthesis and uses one.

4. Repeat steps 2, 3, and 4 for each end point.

The vertex at the botton of each line drawn represents the parenthetical string as constructed so far.

These conventions are somewhat arbitrary, as conventions must be, but they result in a picture that is regular and easy to understand. The chart was helpful in defining the function and discovering the structure.

3.2 The Function

$$B_{n,m} = \begin{cases} B_{n-1,m+1} + B_{n,m-1} & if \quad (n > 0) \wedge (m > 0) \\ B_{n-1,m+1} & if \quad (n > 0) \wedge (m = 0) \\ 1 & if \quad (n = 0) \end{cases}$$

Each part of the chart corresponds to a case of the function. Figure 2 relates the parts of the chart to the cases of the function.

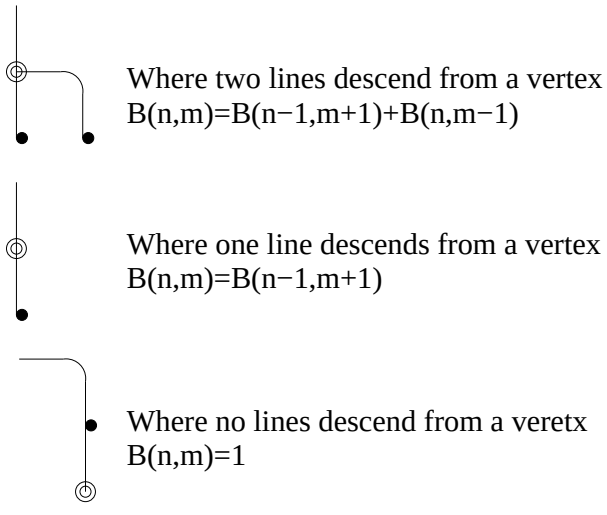

Figure 2: Relationship between the chart and cases of the function

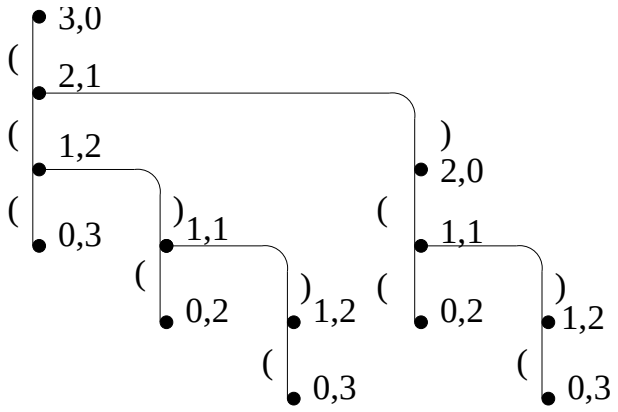

Figure 3: Parameters of $B_{3,0}$ at each vertex

The parameters of the function $B_{n,m}$ take on different values at different points in the recursive descent. Figure 3 shows the parameters at each stage for $B_{3,0}$.

Parameter n represents the number of left parentheses that can be used from that point onward. Parameter m represents the number of additional right parentheses needed to balance the number of left parentheses already used. Considered constructively, m represents the number of right parentheses that may be written at that point. When a left parenthesis is written, n is reduced and m is increased. When a right parenthesis is written, m is reduced.

Of course, once the function is defined, it is freed of any necessary tie to parentheses.

If we say it is possible for any recursive function to be simple, then this function is simple and perhaps more fundamental than the closed form. The closed form is simpler to write. However, while the notation for "$2n$ choose n" is simple, it implies more complex ideas. The closed form has multiplication and division operations. While the comparisons in the recursive functions are obvious and explicitly shown, there are also comparisons implied in any evaluation of the closed form.

Assuming that it is not possible to do algebra with the recursive function, it seems less useful than the closed form. However, it is possible to do substitutions. For example, $B_{4,1}$ can be restated as $B_{3,2} + B_{4,0}$, and vice versa. Substitution could be used to define the function differently, but the way the function was defined above seems simple and it fits well with the parentheses chart.

$B_{n,0}$ is far less efficient computationally than the closed form. This will be developed in the Appendix.

3.3 The Generalization

This function is a generalization of Catalan numbers. The standard Catalan number $C_n = B_{n,0}$. Table 1 also includes some of the others:

	M=0	1	2	3	4	5
N=0	1	1	1	1	1	1
1	1	2	3	4	5	6
2	2	5	9	14	20	27
3	5	14	28	48	75	110
4	14	42	90	165	275	429
5	42	132	297	572	1001	1638
6	132	429	1001	2002	3640	6188
7	429	1430	3432	7072	13260	23256
8	1430	4862	11934	25194	48450	87210

Table 1: Generalized Catalan Numbers $B_{n,m}$ for $n \in [0,8]$, $m \in [0,5]$.

3.4 The Structure

The structure can be expressed as:

$$C_n = B_{n-3,3} + 2C_{n-1}$$

or as

$$C_n = B_{n-3,3} + 2B_{n-1,0}$$

The chart for C_n can be characterized as having a left lobe and two equal right lobes. The right lobes are equal both in structure and value. They are also each equal to C_{n-1} in structure and value. Figures 4, 5, and 6 show the structure.

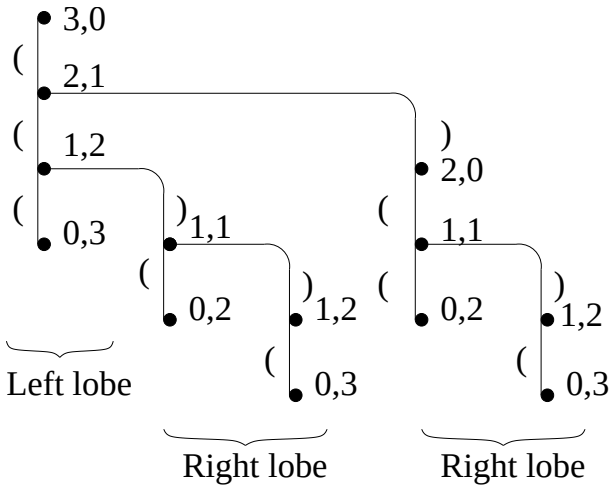

Figure 4: Structure of C_n

In Figure 5, the numeric parameters are replaced by symbolic parameters in terms of n and m. The chart "grows" from the bottom as n increases. The three lobes will always have the values $B_{n-3,3}$, $B_{n-2,1}$, and $B_{n-2,1}$. These can be put in correspondence to the ways legal strings of parentheses may start: (((, (( ), ( ) (. This is a basis of a partition of any set of legal strings.

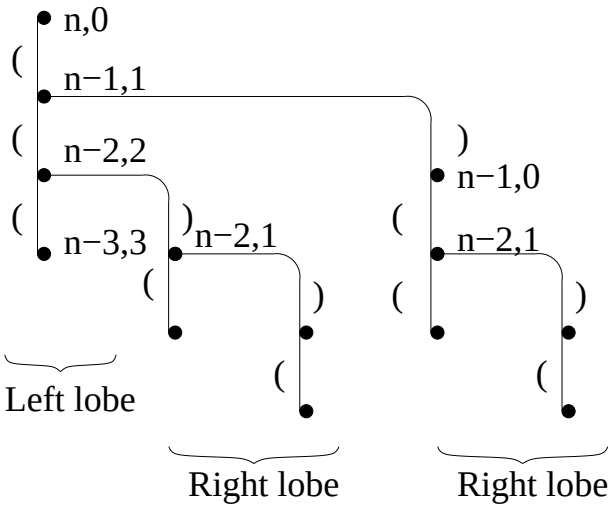

Figure 5: Structure of C_n contd.

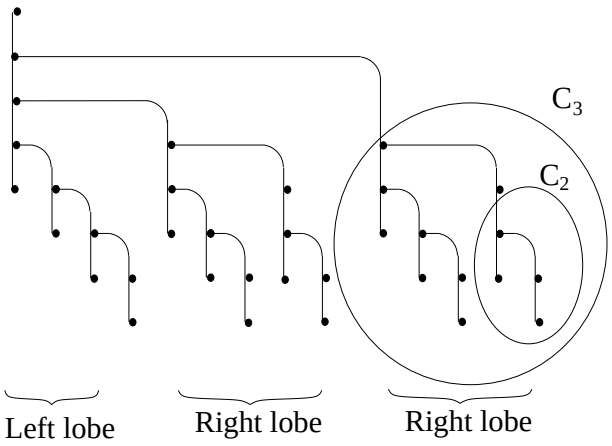

Figure 6: Structure of C_4 or $B_{4,0}$

Figure 6 emphasizes the nested repetitions of structure. Note that C_3 (or C_{n-1}) is found twice and C_2 (or C_{n-2}) is found four times.

The left lobe is different. It starts out smaller than either right lobe and then becomes larger, perhaps approaching the sum of the two right lobes as n gets large. The value of the left lobe is $B_{n-3,3}$. Here's a table of the first few values:

n	3	4	5	6	7	8
$\mathbf{B_{n-3,3}}$	1	4	14	48	165	572

Table 2: Values of the left lobe $B_{n-3,3}$ for $n \in [3, 8]$.

These values were recognized by the On-Line Encyclopedia of Integer Sequences as Sequence A002057, named the Fourth Convolution of Catalan Numbers [5]. This sequence is not pursued here.

4 Further Work

1. What are applications or interpretations of the generalized Catalan numbers?

2. There is doubtless something inherent in the problem that is reflected in the structure, but it is not obvious what. The structure looks natural in terms of the chart, but the chart is just one picture of one interpretation. Why not two lobes? Four? Why any?

3. What is the precise behavior of the size of the left lobe?

4. Is $B_{n-3,3}$ the Fourth Convolution of Catalan numbers?

5 Conclusion

Consideration of the set of legal strings of n pairs of parentheses exposes a structure of this set and of Catalan numbers. The rules for construction of legal strings of parentheses can be recast from a general statement of principles to particular statements of all the cases. This restatement can be expressed as a chart showing all of the cases.

Examination of the chart shows the structure of the sets of strings. Given that the count of legal strings is known to be the Catalan number, the chart exposes a simple and easily understood structure of Catalan numbers. Interpreting the chart as a graph, a recursive function $B_{n,m}$ counts the leaves of the graph (a tree) and therefore calculates the Catalan number.

Taken together, the chart and the function provide a useful tool for gaining an intuitive understanding of an important combinatorial number. Developing the function would be a good problem for students studying recursive functions.

The function $B_{n,m}$ is interesting in its own right. First, it is remarkably simple, using only addition, subtraction, and comparison. It should probably should be considered more fundamental than the closed form which additionally uses multiplication, division, and factorials. Second, the function $B_{n,m}$ has two parameters and is thus a generalization of Catalan numbers.

6 Appendix. Computational Complexity and Efficiency.

The closed form for calculating C_n is clearly more efficient than the recursive $B_{n,0}$. However, examining complexity and efficiency can further illuminate the structure of parenthetical strings and Catalan numbers. The complexity of the closed form is linear in n while that of $B_{n,0}$ is exponential.

This section will only treat $B_{n,0}$ to facilitate comparison with the closed form. The term "C_n" is used here to denote the number, not the method of calculating it.

6.1 Comparison with the closed form.

Even without a precise expression for the complexity of $B_{n,m}$, it is possible to reason about complexity and do some measurements of it. The reasoning goes like this: 1) The complexity of $B_{n,0}$ is greater than the number C_n. 2) C_n is greater than the complexity of the closed form. 3) Therefore the complexity of $B_{n,0}$ is greater than that of the closed form. (It is much greater.)

The unit counted for the closed form is the number of multiplications. After canceling common factors in the numerator and the denominator, the closed form can be expressed as $(2n)(2n-1)...(2n-(n+2))$, calling for $n-2$ multiplications, here called $f(n)$.

The unit counted for $B_{n,0}$ is the number of executions of the function. In many architectures these two measures would not be commensurate. However, the sizes of the complexity numbers dominate any difference. Using C_n as a complexity number, the expression $(2n)(2n-1)...(2n-(n+2))$ expands to a degree $n-1$ polynomial in n, here called $g(n)$.

It can be seen that $f(n)$ is little-oh of $g(n)$ since $\lim_{n\to\infty} f(n)/g(n) = 0$. In other words, f(n) grows more slowly than g(n). In this case it grows much more slowly [8].

The fact that the complexity of $B_{n,0}$ is greater than the number C_n is clear from the chart. The chart has C_n leaves, each contributing 1 to the number of executions. In addition there are many intermediate nodes above the leaves, so that the sum of all executions is greater than C_n. All this demonstrates that the computational complexity of the closed form is little-oh of the complexity of $B_{n,0}$.

A numeric measurement of $B_{n,0}$ is shown in Table 3. (The algorithm based on $B_{n,m}$ can be instrumented to count executions by the appropriate placement of "+1" in the cases of the function.)

n	3	4	5	6	7	8
n − 2	1	2	3	4	5	6
$\mathbf{B_{n,0}}$	13	36	106	328	1034	3485

Table 3: Complexity of the closed form vs. $B_{n,0}$.

6.2 Complexity of different implementations of $B_{n,m}$.

6.2.1 "Bottom-up" implementation of a recursive function.

Due to the highly repetitive structure of $B_{n,m}$, results toward the bottom of the chart are recalculated many times over. To justify this, consider that the tree gets much wider than it is high. For example, at $n = 8$ the number of leaves is $C_n = 1430$. The longest path from the root to a leaf is $2n-1$. This shows that many of the computations are towards the bottom.

Blass and Gurevich use the term "bottom-up" to describe the use of precalculated results to avoid many recalculations [1]. As an example, the following fragment of pseudo-code expresses the $B_{n,m}$ as an algorithm. It avoids recalculation of $B_{n,m}$ for $m, n \in [0, 3]$.

The values of T are from Table 1. Note that the cases are not disjoint. The order of execution resolves ambiguity.

```
var T = new Array ([1,1,1,1], [1,2,3,4], [2,5,9,14], [5,14,28,48]);
function B(n,m) {
    if ((n<4)&&(m<4))     return (T[n][m]);
    if ((n>0)&&(m>0))     return (B(n-1, m+1) + B(n, m-1));
    if ((n>0)&&(m==0))    return (B(n-1, m+1));
    if (n==0)             return (1);
}
```

Table 4 shows measured complexity for this version.

n	3	4	5	6	7	8
top-down	13	36	106	328	1054	3485
bottom-up	1	2	5	13	52	212

Table 4: Complexity of top-down vs. bottom up evaluation of $B_{n,0}$.

6.2.2 Parallel Processing.

The structure of $B_{n,m}$ presents both obstacles and opportunities for parallelization. The word "executions" will be used here the way "processes" and "threads" are often used.

Dividing the work.

It is easy to divide the function into parts to run on separate processors. Consider placing a horizontal line on a drawing of the chart such as Figure 4. Horizontal lines can be drawn at various levels. The point at which the new line intersects a vertical line marks a place where a separate process can consist of all the executions below the intersection. The level of the horizontal line would determine the number of parts. This method would be suitable for a multi-processor with few processors.

Another approach uses the fact that the second case calls for two child evaluations of the function. One of these could be sent to another processor. This would lead to many requests for processors at large n.

Latency.

Latency is another important factor in parallelization. "Latency" is used here to mean the time to initiate and terminate an execution, including passing parameters and returning results. Since the amount of processing in the function is small, latency would be very important if the function were distributed over many processors.

A Single Instruction Multiple Data (SIMD) machine with many processors and low latency would be good here. It would also take advantage of the fact that each execution of the algorithm would use the same small program. However, in general the structure of the function would limit its use on machines with large numbers of processors unless latency was very small.

Inter-process communication.

Since there would be no peer-to-peer communication among executions, an execution would never be interrupted and suspended in the middle of processing. Network contention and overhead would both benefit from this characteristic of $B_{n,m}$. Of course, there is much passing of parameters and results. This contributes to latency, as developed above, and would be a significant use of resources.

References

[1] A. Blass and Y. Gurevich, Algorithms vs. Machines, *Bulletin of the European Association for Theoretical Computer Sciences*, Number 77, pp.96-118, June 2002.

[2] R. Graham, D. Knuth, and O. Patashnik, *Concrete Mathematics: A foundation for Computer Science, 1 ed.*, Reading, Mass.: Addison-Wesley, 1988.

[3] P. Hilton and J. Pedersen, Catalan Numbers, Their Generalization, and Their Uses, *The Mathematical Intellegencer*, Volume 13, Number 2, pp.64-75, 1991.

[4] P. Hilton, D. Holton, and J. Pedersen, *Mathematical Vistas: From a Room with Many Windows*, New York, Springer-Verlag, 2002.

[5] N. Sloane, *On-Line Encyclopedia of Integer Sequences*, published electronically at `http://www.research.att.com/~njas/sequences/Seis.html`, 2002.

[6] R. Stanley, *Enumerative Combinatorics Volume 2*, New York, Cambridge University Press, 1999.

[7] D. Stanton and D. White, *Constructive Combinatorics*, New York, Springer-Verlag, 1986.

[8] G. Thomas and R. Finney, *Calculus and Analytic Geometry, 8 ed.*, Reading, Mass., Addison-Wesley, 1992, Reprinted with corrections, April 1993.