How Many Parse Trees Are Generated by the Ambiguous <E>? Catalan Numbers

We've seen the following BNF for arithmetic expressions is ambiguous:
<E> → <id> | <int> | <float> | <E> + <E> | <E> − <E> | <E> * <E> | <E> / <E> | "(" <E> ")"
What makes this grammar ambiguous is the recursive part for the operators:
<E> → <E> + <E> | <E> − <E> | <E> * <E> | <E> / <E>
Given any expression, we can choose any of its operators not appearing within parentheses to be the top-level operator, and split the expression to the left and right subexpressions. This latitude of choice propagates recursively to each of the two subexpressions. Consider for example: A + B * C − D / E + F * G. We can choose any of the six operators to be the top-level splitting operator, and each choice yields two subexpressions. Within each subexpression in turn, we have (in general) choices for the top-level operator, and so on the process goes down recursively. So we suspect the number of parse trees should multiply exponentially, and this turns out to be the case.

Note that the kinds of arguments and operators are irrelevant to the number of parse trees. A + B * C − D / E + F * G has the same number of parse trees as A + A + A + A + A + A + A. So the total number of parse trees is determined solely by the number of arguments. Let T(n) be the number of distinct parse trees that can be constructed by the above grammar for the parenthesis-free arithmetic expressions with n arguments and $n-1$ operators. T(n) is equal to the total number of binary trees with n leaves. It is also equal to the total number of ways to fully factor a sequence of n symbols. For example, A+A+A+A has five parse trees as there are five ways to fully factor AAAA:
(AA)(AA)
((AA)A)A
(A(AA))A
A((AA)A)
A(A(AA))
Let's write down a recurrence relation for T(n) based on this observation. For one argument A, there vacuously is just one way to factor it, A itself; also there is just one parse tree. So T(1) = 1. For $n \geq 2$, consider the sequence of n A's, AA ⋯ AA. Let the top-level factorization split it between the i-th and the (i+1)-th A:
(A ⋯ A)(A ⋯ A)
where the 1st factor has i A's and the 2nd has $n-i$ A's. Then the total number of factorizations with this particular top-level factorization is T(i)×T($n-i$). Since the top-level split position can be between any two successive A's, we obtain: $T(n) = \Sigma_{1 \leq i \leq n-1}\ T(i)T(n-i)$. So the recurrence relation is

$$T(1) = 1$$
$$T(n) = \Sigma_{1 \leq i \leq n-1}\ T(i)T(n-i), \quad n \geq 2$$

The solution is $\mathbf{T(n) = C(2n-2, n-1)/n = (2n-2)!/(n!(n-1)!)}$, where C($m$, j) is the standard binomial coefficient, i.e., the total number of ways to choose j objects out of m objects. It is known that $T(n) = \Omega(2^n)$, that is, the asymptotic growth rate of T(n) is at least exponential.

Here are some of its values:

n	T(n)
1	1
2	1
3	2
4	5
5	14
6	42
7	132
8	429
9	1,430
10	4,862
20	1,767,263,190
30	1,002,242,216,651,368

It's interesting that the exploding number of parse trees is reduced to a unique one by requiring simple rules of precedence and associativity.

T(*n*) is known as the Catalan numbers. It is the solution of a number of seemingly different but actually equivalent counting problems.