Catalan Numbers

Manuel Eberl

April 14, 2026

Abstract

In this work, we define the Catalan numbers C_n and prove several equivalent definitions (including some closed-form formulae). We also show one of their applications (counting the number of binary trees of size n), prove the asymptotic growth approximation $C_n \sim \frac{4^n}{\sqrt{\pi} n^{1.5}}$, and provide reasonably efficient executable code to compute them.

The derivation of the closed-form formulae uses algebraic manipulations of the ordinary generating function of the Catalan numbers, and the asymptotic approximation is then done using generalised binomial coefficients and the Gamma function. Thanks to these highly non-elementary mathematical tools, the proofs are very short and simple.

Contents

1 Catalan numbers

theory *Catalan-Auxiliary-Integral*
imports *HOL−Analysis.Analysis HOL−Real-Asymp.Real-Asymp*
begin

1.1 Auxiliary integral

First, we will prove the integral

$$\int_0^4 \sqrt{\frac{4-x}{x}}\,\mathrm{d}x = 2\pi$$

which occurs in the proof for the integral formula for the Catalan numbers.

context
begin

We prove the integral by explicitly constructing the indefinite integral.

lemma *catalan-aux-integral*:
  ((λx::real. sqrt ((4 − x) / x)) has-integral 2 ∗ pi) {0..4}
⟨*proof*⟩

end

end

theory *Catalan-Numbers*
imports
  Complex-Main
  Catalan-Auxiliary-Integral
  HOL−Analysis.Analysis
  HOL−Computational-Algebra.Formal-Power-Series
  HOL−Library.Landau-Symbols
  Landau-Symbols.Landau-More
begin

1.2 Other auxiliary lemmas

lemma *mult-eq-imp-eq-div*:
  assumes *a ∗ b = c (a :: 'a :: semidom-divide) ≠ 0*
  shows *b = c div a*
  ⟨*proof*⟩

lemma *Gamma-minus-one-half-real*:
  Gamma (−(1/2) :: real) = − 2 ∗ sqrt pi
  ⟨*proof*⟩

lemma *gbinomial-asymptotic'*:
assumes $z \notin \mathbb{N}$
shows $(\lambda n.\ z\ gchoose\ (n + k)) \sim[at\text{-}top]$
$(\lambda n.\ (-1)\hat{}(n{+}k)\ /\ (Gamma\ (-z) * of\text{-}nat\ n\ powr\ (z + 1)) :: real)$
⟨*proof*⟩

1.3 Definition

We define Catalan numbers by their well-known recursive definition. We shall later derive a few more equivalent definitions from this one.

fun *catalan* :: *nat* ⇒ *nat* **where**
catalan 0 = *1*
| *catalan* (*Suc n*) = ($\sum$ *i*≤*n*. *catalan i* ∗ *catalan* (*n* − *i*))
⟨*proof*⟩

The easiest proof of the more profound properties of the Catalan numbers (such as their closed-form equation and their asymptotic growth) uses their ordinary generating function (OGF). This proof is almost mechanical in the sense that it does not require 'guessing' the closed form; one can read it directly from the generating function.

We therefore define the OGF of the Catalan numbers ($\sum_{n=0}^{\infty} C_n z^n$ in standard mathematical notation):

definition *fps-catalan* = *Abs-fps* (*of-nat* ∘ *catalan*)

lemma *fps-catalan-nth* [*simp*]: *fps-nth fps-catalan n* = *of-nat* (*catalan n*)
⟨*proof*⟩

Given their recursive definition, it is easy to see that the OGF of the Catalan numbers satisfies the following recursive equation:

lemma *fps-catalan-recurrence*:
fps-catalan = *1* + *fps-X* ∗ *fps-catalan*^*2*
⟨*proof*⟩

We can now easily solve this equation for *fps-catalan*: if we denote the unknown OGF as $F(z)$, we get $F(z) = \frac{1}{2}(1 - \sqrt{1-4z})$.

Note that we do not actually use the square root as defined on real or complex numbers. Any $(1+cz)^\alpha$ can be expressed using the formal power series whose coefficients are the generalised binomial coefficients, and thus we can do all of these transformations in a purely algebraic way: $\sqrt{1-4z} = (1+z)^{\frac{1}{2}} \circ (-4z)$ (where $\circ$ denotes composition of formal power series) and $(1+z)^\alpha$ has the well-known expansion $\sum_{n=0}^{\infty} \binom{\alpha}{n} z^n$.

lemma *fps-catalan-fps-binomial*:
fps-catalan = (*1/2* ∗ (*1* − (*fps-binomial* (*1/2*) *oo* (−*4*∗*fps-X*)))) / *fps-X*
⟨*proof*⟩

1.4 Closed-form formulae and more recurrences

We can now read a closed-form formula for the Catalan numbers directly from the generating function $\frac{1}{2z}(1 - (1+z)^{\frac{1}{2}} \circ (-4z))$.

theorem *catalan-closed-form-gbinomial*:
real (catalan n) = 2 ∗ (− 4) ^ n ∗ (1/2 gchoose Suc n)
⟨*proof*⟩

This closed-form formula can easily be rewritten to the form $C_n = \frac{1}{n+1}\binom{2n}{n}$, which contains only 'normal' binomial coefficients and not the generalised ones:

lemma *catalan-closed-form-aux*:
catalan n ∗ Suc n = (2∗n) choose n
⟨*proof*⟩

theorem *of-nat-catalan-closed-form*:
of-nat (catalan n) = (of-nat ((2∗n) choose n) / of-nat (Suc n) :: 'a :: field-char-0)
⟨*proof*⟩

theorem *catalan-closed-form*:
catalan n = ((2∗n) choose n) div Suc n
⟨*proof*⟩

The following is another nice closed-form formula for the Catalan numbers, which directly follows from the previous one:

corollary *catalan-closed-form'*:
catalan n = ((2∗n) choose n) − ((2∗n) choose (Suc n))
⟨*proof*⟩

We can now easily show that the Catalan numbers also satisfy another, simpler recurrence, namely $C_{n+1} = \frac{2(2n+1)}{n+2}C_n$. We will later use this to prove code equations to compute the Catalan numbers more efficiently.

lemma *catalan-Suc-aux*:
(n + 2) ∗ catalan (Suc n) = 2 ∗ (2 ∗ n + 1) ∗ catalan n
⟨*proof*⟩

theorem *of-nat-catalan-Suc'*:
of-nat (catalan (Suc n)) =
(of-nat (2∗(2∗n+1)) / of-nat (n+2) ∗ of-nat (catalan n) :: 'a :: field-char-0)
⟨*proof*⟩

theorem *catalan-Suc'*:
catalan (Suc n) = (catalan n ∗ (2∗(2∗n+1))) div (n+2)
⟨*proof*⟩

1.5 Integral formula

The recursive formula we have just proven allows us to derive an integral formula for the Catalan numbers. The proof was adapted from a textbook proof by Steven Roman. [1]

context
begin

private definition *I* :: *nat* ⇒ *real* **where**
  I n = *integral* {*0..4*} (λ*x*. *x powr* (*of-nat n* − *1/2*) ∗ *sqrt* (*4* − *x*))

private lemma *has-integral-I0*: ((λ*x*. *x powr* (−(*1/2*)) ∗ *sqrt* (*4* − *x*)) *has-integral* *2*∗*pi*) {*0..4*}
⟨*proof*⟩ **lemma** *integrable-I*:
  (λ*x*. *x powr* (*of-nat n* − *1/2*) ∗ *sqrt* (*4* − *x*)) *integrable-on* {*0..4*}
⟨*proof*⟩ **lemma** *I-Suc*: *I* (*Suc n*) = *real* (*2* ∗ (*2*∗*n* + *1*)) / *real* (*n* + *2*) ∗ *I n*
⟨*proof*⟩ **lemma** *catalan-eq-I*: *real* (*catalan n*) = *I n* / (*2* ∗ *pi*)
⟨*proof*⟩

theorem *catalan-integral-form*:
  ((λ*x*. *x powr* (*real n* − *1* / *2*) ∗ *sqrt* (*4* − *x*) / (*2*∗*pi*))
    has-integral real (*catalan n*)) {*0..4*}
⟨*proof*⟩

end

1.6 Asymptotics

Using the closed form $C_n = 2 \cdot (-4)^n \binom{\frac{1}{2}}{n+1}$ and the fact that $\binom{\alpha}{n} \sim \frac{(-1)^n}{\Gamma(-\alpha) n^{\alpha+1}}$ for any $\alpha \notin \mathbb{N}$, wwe can now easily analyse the asymptotic behaviour of the Catalan numbers:

theorem *catalan-asymptotics*:
  catalan ∼[*at-top*] (λ*n*. *4* ^ *n* / (*sqrt pi* ∗ *n powr* (*3/2*)))
⟨*proof*⟩

1.7 Relation to binary trees

It is well-known that the Catalan number C_n is the number of rooted binary trees with n internal nodes (where internal nodes are those with two children and external nodes are those with no children).

We will briefly show this here to show that the above asymptotic formula also describes the number of binary trees of a given size.

qualified datatype *tree* = *Leaf* | *Node tree tree*

qualified primrec *count-nodes* :: *tree* ⇒ *nat* **where**
  count-nodes Leaf = *0*

| *count-nodes* (*Node l r*) = *1* + *count-nodes l* + *count-nodes r*

qualified definition *trees-of-size* :: *nat* ⇒ *tree set* **where**
trees-of-size n = {*t*. *count-nodes t* = *n*}

lemma *count-nodes-eq-0-iff* [*simp*]: *count-nodes t* = *0* ⟷ *t* = *Leaf*
⟨*proof*⟩

lemma *trees-of-size-0* [*simp*]: *trees-of-size 0* = {*Leaf*}
⟨*proof*⟩

lemma *trees-of-size-Suc*:
trees-of-size (*Suc n*) = (λ(*l*,*r*). *Node l r*) ' (⋃*k*≤*n*. *trees-of-size k* × *trees-of-size* (*n* − *k*))
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *finite-trees-of-size* [*simp*,*intro*]: *finite* (*trees-of-size n*)
⟨*proof*⟩

lemma *trees-of-size-nonempty*: *trees-of-size n* ≠ {}
⟨*proof*⟩

lemma *trees-of-size-disjoint*:
assumes *m* ≠ *n*
shows *trees-of-size m* ∩ *trees-of-size n* = {}
⟨*proof*⟩

theorem *card-trees-of-size*: *card* (*trees-of-size n*) = *catalan n*
⟨*proof*⟩

1.8 Efficient computation

We shall now prove code equations that allow more efficient computation of Catalan numbers. In order to do this, we define a tail-recursive function that uses the recurrence *catalan* (*Suc n*) = *catalan n* ∗ (*2* ∗ (*2* ∗ *n* + *1*)) *div* (*n* + *2*):

qualified function *catalan-aux* **where** [*simp del*]:
catalan-aux n k acc =
(*if k* ≥ *n then acc else catalan-aux n* (*Suc k*) ((*acc* ∗ (*2*∗(*2*∗*k*+*1*))) *div* (*k*+*2*)))
⟨*proof*⟩
termination ⟨*proof*⟩ **lemma** *catalan-aux-simps*:
k ≥ *n* ⟹ *catalan-aux n k acc* = *acc*
k < *n* ⟹ *catalan-aux n k acc* = *catalan-aux n* (*Suc k*) ((*acc* ∗ (*2*∗(*2*∗*k*+*1*))) *div* (*k*+*2*))
⟨*proof*⟩ **lemma** *catalan-aux-correct*:
assumes *k* ≤ *n*
shows *catalan-aux n k* (*catalan k*) = *catalan n*

⟨*proof*⟩

lemma *catalan-code* [*code*]: *catalan n* = *catalan-aux n 0 1*
⟨*proof*⟩
end

References

[1] S. Roman. *An Introduction to Catalan Numbers*. Birkhäuser Basel, 2015.