Enumerative Combinatorics in Coq/MathComp: Formal Power Series and the example of Catalan numbers

Florent Hivert

LISN / Université Paris Saclay / CNRS

Math Comp Workshop - December 2022

Outline

https://github.com/hivert/FormalPowerSeries

Flajolet-Sedgwick

Analytic combinatorics

Figure I.2.

The prehistory of Catalan numbers.

1. On September 4, 1751, Euler writes to his friend Goldbach [196]:

Ich bin neulich auf eine Betrachtung gefallen, welche mir nicht wenig merkwürdig vorkam. Dieselbe betrifft, auf wie vielerley Arten ein gegebenes polygonum durch Diagonallinien in triangula zerchnitten werden könne.	I have recently encountered a question, which appears to me rather noteworthy. It concerns the number of ways in which a given [convex] polygon can be decomposed into triangles by diagonal lines.

Euler then describes the problem (for an n–gon, i.e., $(n-2)$ triangles) and concludes:

Setze ich nun die Anzahl dieser verschiedenen Arten $= x$ *[...]. Hieraus habe ich nun den Schluss gemacht, dass generaliter sey* $$x = \frac{2.6.10.14....(4n-10)}{2.3.4.5....(n-1)}$$ *[...] Ueber die Progression der Zahlen* $1, 2, 5, 14, 42, 132$, *etc. habe ich auch diese Eigenschaft angemerket, dass* $1+2a+5a^2+14a^3+42a^4+132a^5+$ *etc.* $= \frac{1-2a-\sqrt{1-4a}}{2aa}$.	Let me now denote by x this number of ways [...]. I have then reached the conclusion that in all generality $$x = \frac{2.6.10.14....(4n-10)}{2.3.4.5....(n-1)}$$ [...] Regarding the progression of the numbers $1, 2, 5, 14, 42, 132$, and so on, I have also observed the following property: $1+2a+5a^2+14a^3+42a^4+132a^5+$ etc. $= \frac{1-2a-\sqrt{1-4a}}{2aa}$.

Thus, as early as 1751, Euler knew the solution as well as the associated **generating function**. From his writing, it is however unclear whether he had found complete proofs.

2. In the course of the 1750s, Euler communicated the problem, together with initial elements of the counting sequence, to Segner, who writes in his publication [146] dated 1758: "The great Euler has benevolently communicated these numbers to me; the way in which he found them, and the law of their progression having remained hidden to me" [*"quos numeros mecum beneuolus communicauit summus Eulerus; modo, quo eos reperit, atque progressionis ordine, celatis"*]. Segner develops a recurrence approach to Catalan numbers. By a root decomposition analogous to ours, on p. 35, he proves (in our notation, for decompositions into n triangles)

$$T_n = \sum_{k=0}^{n-1} T_k T_{n-1-k}, \qquad T_0 = 1, \tag{4}$$

a recurrence by which the Catalan numbers can be computed to any desired order. (Segner's work was to be reviewed in [197], anonymously, but most probably, by Euler.)

3. During the 1830s, Liouville circulated the problem and wrote to Lamé, who answered the next day(!) with a proof [399] based on recurrences similar to (4) of the explicit expression:

$$T_n = \frac{1}{n+1}\binom{2n}{n}. \tag{5}$$

Interestingly enough, Lamé's three-page note [399] appeared in the 1838 issue of the *Journal de mathématiques pures et appliquées* ("Journal de Liouville"), immediately followed by a longer study by Catalan [106], who also observed that the T_n intervene in the number of ways of multiplying n numbers (this book, §I. 5.3, p. 73). Catalan would then return to these problems [107, 108], and the numbers $1, 1, 2, 5, 14, 42, \ldots$ eventually became known as the **Catalan numbers**. In [107], Catalan finally *proves* the validity of Euler's generating function:

$$T(z) := \sum_n T_n z^n = \frac{1-\sqrt{1-4z}}{2z}. \tag{6}$$

Catalan numbers

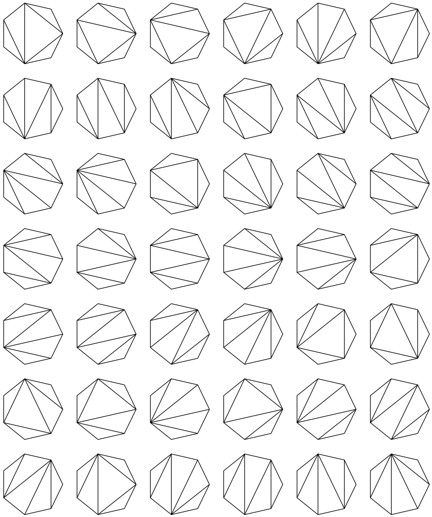
Polygon triangulations

$1, 1, 2, 5, 14, 42, 132 \ldots$

$$T_0 = 1,$$

$$T_n = \sum_{k=0}^{n-1} T_k T_{n-1-k}$$

$$T_n = \frac{1}{n+1}\binom{2n}{2}$$



catalan

Euler, Segner, Liouville, Catalan, ...

$$T(z) := \sum_n T_n z^n = 1 + z + 2z^2 + 5z^3 + 14z^4 + 42z^5 + 132z^6 + \dots$$

$$= \frac{1 - \sqrt{1 - 4z}}{2z}$$

Sommaire

Enumerative Combinatorics
A short introduction from an algorithmic point of view

Counting and generating combinatorial objects

Example of finite sets...

n-bits sequences:

0 1

00 01 10 11

000 001 010 011 100 101 110 111

0000 0001 0010 0011 0100 0101 0110 0111
1000 1001 1010 1011 1100 1101 1110 1111

Cardinality (number of elements): `https://oeis.org/A000079`

$$2^n : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096 \dots$$

Permutation of $[1, 2, \ldots, n]$

1

12 21

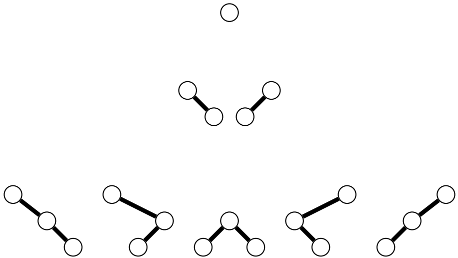
123 132 213 231 312 321

1234 1243 1324 1342 1423 1432 2134 2143 2314 2341 2413 2431
3124 3142 3214 3241 3412 3421 4123 4132 4213 4231 4312 4321

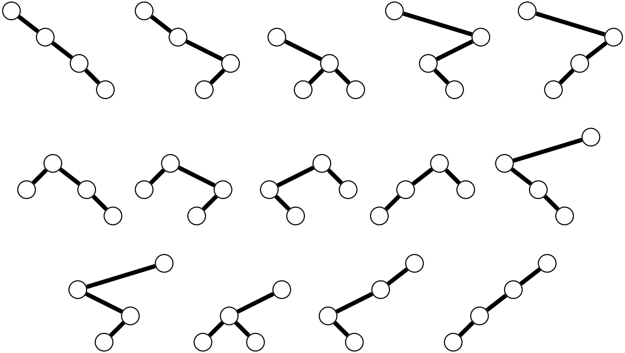
Cardinality (number of elements): `https://oeis.org/A000142`

$n! : 1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800, 39916800 \ldots$

Binary trees with n nodes



Binary trees with n nodes

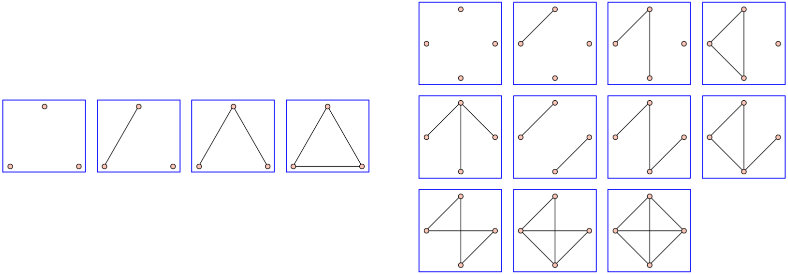


Cardinality (number of elements): `https://oeis.org/A000142`

$\mathrm{Cat}(n)$: $1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012\ldots$

Unlabelled graphs with n vertices

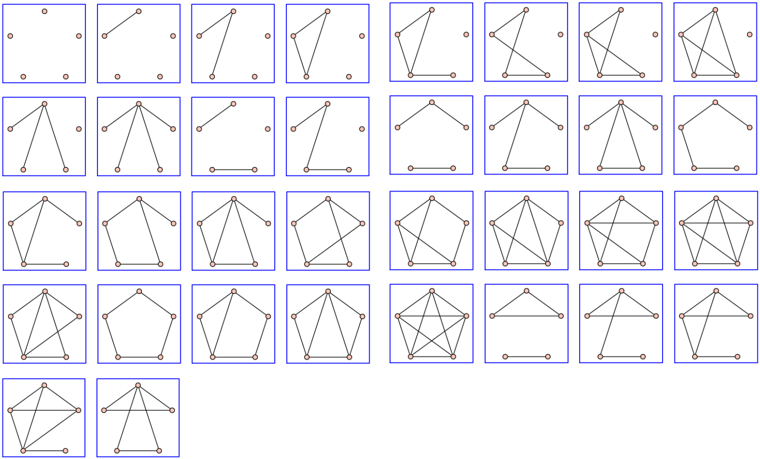
Unlabelled = upto isomorphism



Cardinality (number of elements): `https://oeis.org/A000088`

$\mathrm{Gr}(n) : 1, 2, 4, 11, 34, 156, 1044, 12346, 274668, 12005168, 1018997864\ldots$

Unlabelled graphs with 5 vertices:



More «real life examples»

- XML document with n balises
- n character program in C
- possible execution pathes in a code

Combinatorial Generation

Question

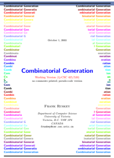
Find efficient algorithms

- count, find the list, iterate
- fair random sampling

Application:

- **search of a solution** using brute force or randomization
- analysis of algorithms, **complexity computation**
- **tests**, fuzzing
- biology, chemistry, statistical physics

Standard references



- Frank Ruskey, *Combinatorial Generation* `doi:10.1.1.93.5967`, 2003, unpublished
- A. Nijenhuis and H.S. Wilf, *Combinatorial algorithms*, 2nd ed., Academic Press, 1978

  `http://www.math.upenn.edu/~wilf/website/CombinatorialAlgorithms.pdf`
- P. Flajolet and R. Sedgewick, *Analytic Combinatorics*, Cambridge University Press, 2009. Electronic version

  `http://algo.inria.fr/flajolet/Publications/AnaCombi/book.pdf`
- The On-Line Encyclopedia of Integer Sequences `http://oeis.org`
- A list of combinatorial software:

  `https://www.mat.univie.ac.at/~slc/divers/software.html`

Generic algorithms

Question

How to avoid *ad hoc* solution for each and every type of combinatorial objects ?

- **basic components** $\Longrightarrow$ singleton, union, cartesian product, set and multiset, cycle...
- **combine** the basic components $\Longrightarrow$ combinatorial class, description grammar

Today: Only counting !

Generic algorithms

Question

How to avoid *ad hoc* solution for each and every type of combinatorial objects ?

- **basic components** $\Longrightarrow$ singleton, union, cartesian product, set and multiset, cycle...
- **combine** the basic components $\Longrightarrow$ combinatorial class, description grammar

Today: Only counting !

Generic algorithms

Question

How to avoid *ad hoc* solution for each and every type of combinatorial objects ?

- **basic components** $\Longrightarrow$ singleton, union, cartesian product, set and multiset, cycle...
- **combine** the basic components $\Longrightarrow$ combinatorial class, description grammar

Today: Only counting !

Generic algorithms

Question

How to avoid *ad hoc* solution for each and every type of combinatorial objects ?

- **basic components** $\Longrightarrow$ singleton, union, cartesian product, set and multiset, cycle...
- **combine** the basic components $\Longrightarrow$ combinatorial class, description grammar

Today: Only counting !

Sommaire

Notion of combinatorial class

Definition (Combinatorial class)

A combinatorial class, is a finite or denumerable set $\mathcal{C}$ whose elements e have a size (also called degree) noted $|e|$, satisfying the following conditions:

- *the size of an element is a non-negative integer*
- *the number of elements of any given size is finite*

$$\mathrm{card}\{\in \mathcal{C} \mid |e| = n\} < \infty$$

Example:

- Binary trees where size is the number of nodes

Generating series

Definition

The (ordinary) generating series of a sequence $(a_n)_n$ is the formal power series

$$A(z) := \sum_{n=0}^{\infty} a_n z^n \,.$$

The generating series of a combinatorial class $\mathcal{A}$ is the generating series of the numbers $a_n := \mathrm{card}(\mathcal{A}_n)$. Equivalently,

$$A(z) = \sum_{\alpha \in \mathcal{A}} z^{|\alpha|} \,.$$

The graded disjoint union

If $\mathcal{C} = \mathcal{A} \sqcup \mathcal{B}$, the elements of $\mathcal{A}$ and $\mathcal{B}$ keep their size in the graded disjoint union:

$$\mathcal{C}_n := \mathcal{A}_n \sqcup \mathcal{B}_n$$

Therefore

$$\operatorname{card} \mathcal{C}_n = \operatorname{card} \mathcal{A}_n + \operatorname{card} \mathcal{B}_n$$

Note

The generating series of a disjoint union is the sum of generating series:

$$C(z) = A(z) + B(z)\,.$$

The graded disjoint union

If $\mathcal{C} = \mathcal{A} \sqcup \mathcal{B}$, the elements of $\mathcal{A}$ and $\mathcal{B}$ keep their size in the graded disjoint union:

$$\mathcal{C}_n := \mathcal{A}_n \sqcup \mathcal{B}_n$$

Therefore

$$\operatorname{card} \mathcal{C}_n = \operatorname{card} \mathcal{A}_n + \operatorname{card} \mathcal{B}_n$$

Note

The generating series of a disjoint union is the sum of generating series:

$$C(z) = A(z) + B(z)\,.$$

The graded Cartesian product

Idea: sizes (cost, number of memory locations, ...) are added:

$$|(a,b)|_{\mathcal{A}\times\mathcal{B}} := |a|_{\mathcal{A}} + |b|_{\mathcal{B}}$$

If $\mathcal{C} = \mathcal{A} \times \mathcal{B}$ then $\mathcal{C}_n = \bigsqcup_{i+j=n} \mathcal{A}_i \times \mathcal{B}_j$

Cardinality:

$$\operatorname{card}\mathcal{C}_n = \sum_{i+j=n} \operatorname{card}\mathcal{A}_i \cdot \operatorname{card}\mathcal{B}_j$$

$$[z^n](A(z)\cdot B(z)) = \sum_{i+j=n} [z^i]A(z)\cdot[X^j]B(z)$$

Note

The generating series of a cartesian product is the product of the generating series:

$$C(z) = A(z)\cdot B(z)\,.$$

The graded Cartesian product

Idea: sizes (cost, number of memory locations, ...) are added:

$$|(a,b)|_{\mathcal{A}\times\mathcal{B}} := |a|_{\mathcal{A}} + |b|_{\mathcal{B}}$$

If $\mathcal{C} = \mathcal{A}\times\mathcal{B}$ then $\mathcal{C}_n = \bigsqcup_{i+j=n} \mathcal{A}_i \times \mathcal{B}_j$

Cardinality:

$$\operatorname{card}\mathcal{C}_n = \sum_{i+j=n} \operatorname{card}\mathcal{A}_i \cdot \operatorname{card}\mathcal{B}_j$$

$$[z^n](A(z)\cdot B(z)) = \sum_{i+j=n} [z^i]A(z)\cdot[X^j]B(z)$$

Note

The generating series of a cartesian product is the product of the generating series:

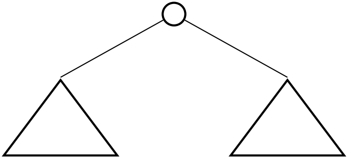
$$C(z) = A(z)\cdot B(z)\,.$$

A dictionary : Comb. Classes / Gen. Fun. (unlabeled case)

Neutral ($\lvert\epsilon\rvert = 0$)	$\mathcal{E} = \{\epsilon\}$	$E(z) = 1$
Atom ($\lvert \bullet \rvert = 1$)	$\mathcal{Z} = \{\bullet\}$	$Z(z) = z$
Disjoint Union	$\mathcal{A} = \mathcal{B} \sqcup \mathcal{C}$	$A(z) = B(z) + C(z)$
Cartesian product	$\mathcal{A} = \mathcal{B} \times \mathcal{C}$	$A(z) = B(z) \cdot C(z)$
Sequence	$\mathcal{A} = \mathrm{Seq}(\mathcal{B})$	$A(z) = \dfrac{1}{1 - B(z)}$
Powerset	$\mathcal{A} = \mathrm{PSet}(\mathcal{B})$	$A(z) = \exp\left(\sum_{k=1}^{\infty} \dfrac{(-1)^{k-1}}{k} B(z^k)\right)$
Multiset	$\mathcal{A} = \mathrm{MSet}(\mathcal{B})$	$A(z) = \exp\left(\sum_{k=1}^{\infty} \dfrac{1}{k} B(z^k)\right)$
Cycle	$\mathcal{A} = \mathrm{Cycle}(\mathcal{B})$	$A(z) = \sum_{k=1}^{\infty} \dfrac{\phi(k)}{k} \log \dfrac{1}{1 - B(z^k)}$

Example : Binary trees

Size = number of internal nodes



$$\mathrm{BinTree} = \{\epsilon\} \sqcup \mathrm{BinTree} \times \mathcal{Z} \times \mathrm{BinTree}$$

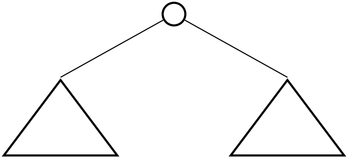
$$T(z) = 1 + T(z) \cdot z \cdot T(z) = 1 + z \cdot T(z)^2$$

Solution by radicals (quadratic equation):

$$T(z) = \frac{1 - \sqrt{1 - 4z}}{2z}$$

Example : Binary trees

Size = number of internal nodes



$$\mathrm{BinTree} = \{\epsilon\} \sqcup \mathrm{BinTree} \times \mathcal{Z} \times \mathrm{BinTree}$$

$$T(z) = 1 + T(z) \cdot z \cdot T(z) = 1 + z \cdot T(z)^2$$

Solution by radicals (quadratic equation):

$$T(z) = \frac{1 - \sqrt{1 - 4z}}{2z}$$

Summary: Why generating series ?

Note (The symbolic method)

- describe combinatorial objects by grammars using basic elementary constructions
- translate these grammar to systems of functional equations on generating series
- «solve» these systems by algebraic manipulation allows to extract the coefficients

Going further: asymptotic analysis thanks to complex analysis

Sommaire

What are power series ?

K : a ring.

Definition

The ring $K[[X]]$ of formal power series is the set of sequences $(a_n)_n$, written as $\sum_n a_n X^n$ together with the natural sum, and product.

More structures: derivation, integration, substitution...

Problem: series are infinite objects

- Impossible to store one in a finite data structure
- Equality cannot be decided (MathComp requirement)

What are power series ?

K : a ring.

Definition

The ring $K[[X]]$ of formal power series is the set of sequences $(a_n)_n$, written as $\sum_n a_n X^n$ together with the natural sum, and product.

More structures: derivation, integration, substitution...

Problem: series are infinite objects

- Impossible to store one in a finite data structure
- Equality cannot be decided (MathComp requirement)

Two possible implementations

Note

- truncated formal power series : For a fixed n, we only know coefficients upto x^n
- infinite formal power series : we store all the coefficient but we need infinite objects and classical axioms

Remark: this is also a problem in Computer Algebra Systems

Power series in Sagemath

```
sage: F.<z> = LazyPowerSeriesRing(QQ)   # coeff. computed on demand
sage: T = 1/(1-z); T
Uninitialized lazy power series
sage: T[3]
1
sage: T
1 + z + z^2 + z^3 + O(x^4)
sage: T[5]
1
sage: T
1 + z + z^2 + z^3 + z^4 + z^5 + O(x^6)

sage: FF.<x> = PowerSeriesRing(QQ)      # truncated (to X^20 by default)
sage: T = 1/(1-x)
sage: T
1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 +
 x^10 + x^11 + x^12 + x^13 + x^14 + x^15 + x^16 + x^17 +
 x^18 + x^19 + O(x^20)
sage: T[40]
IndexError: coefficient not known
```

Truncated formal power series

Definition

The ring $K[[X]]_n$ of n th-Truncated power series is the quotient ring $K[X]/\langle x^n \rangle$.

Major rewrite of the code from Cyril Cohen and Boris Djalal to adapt it to any ring (e.g. $\mathbb{Z}$) – not just a field.

Truncation = cutting a list (instead of taking the remainder in the Euclidian division).

Truncated formal power series

Definition

The ring $K[[X]]_n$ of n th-Truncated power series is the quotient ring $K[X]/\langle x^n \rangle$.

Major rewrite of the code from Cyril Cohen and Boris Djalal to adapt it to any ring (e.g. $\mathbb{Z}$) – not just a field.

Truncation = cutting a list (instead of taking the remainder in the Euclidian division).

Truncated formal power series

Definition:

```
Variable R : ringType.
Variable n : nat.
Record truncfps := MkTfps { tfps : {poly R}; _ : size tfps <= n.+1 }.
```

Extracting coefficients:

```
Implicit Types (p q r s : {poly R}) (f g : {tfps R n}).

Lemma coef_tfps f i : f`_i = if i <= n then f`_i else 0.
Lemma tfpsP f g : (forall i, i <= n -> f`_i = g`_i) <-> (f = g).
```

Truncated formal power series (2)

Polynomial truncation

```
Fact trXn_subproof p : size (Poly (take n.+1 p)) <= n.+1.
Definition trXn p := MkTfps (trXn_subproof p).
```

gives the ring structure:

```
Lemma trXn_mul p q :
  trXn n (tfps (trXn n p) * tfps (trXn n q)) = trXn n (p * q).

Fact one_tfps_proof : size (1 : {poly R}) <= n.+1.
Definition one_tfps : {tfps R n} := Tfps_of one_tfps_proof.

Definition mul_tfps f g := trXn n (tfps f * tfps g).
```

Truncated formal power series (2)

Polynomial truncation

```
Fact trXn_subproof p : size (Poly (take n.+1 p)) <= n.+1.
Definition trXn p := MkTfps (trXn_subproof p).
```

gives the ring structure:

```
Lemma trXn_mul p q :
  trXn n (tfps (trXn n p) * tfps (trXn n q)) = trXn n (p * q).

Fact one_tfps_proof : size (1 : {poly R}) <= n.+1.
Definition one_tfps : {tfps R n} := Tfps_of one_tfps_proof.

Definition mul_tfps f g := trXn n (tfps f * tfps g).
```

More structure on formal power series

Example: if R is a unitary ring then $R[[X]]_n$ is too.

```
Variable R : unitRingType.
Definition unit_tfps : pred {tfps R n} := fun f => f`_0 \in GRing.unit.
```

Definition by fixed point:

```
Fixpoint inv_tfps_rec f bound m :=
  if bound is b.+1 then
    if (m <= b)%N then inv_tfps_rec f b m
    else -f`_0%N^-1 * (\sum_(i < m) f`_i.+1 * (inv_tfps_rec f b (m - i.+1)%N))
  else f`_0%N^-1.
Definition inv_tfps f : {tfps R n} :=
  if unit_tfps f then [tfps i <= n => inv_tfps_rec f i i] else f.
```

```
Lemma mul_tfpsVr : {in unit_tfps, right_inverse 1 inv_tfps *%R}.
Lemma mul_tfpsrV : {in unit_tfps, left_inverse 1 inv_tfps  *%R}.
```

More structure on formal power series

Example: formal derivative

```
Definition deriv_tfps f := [tfps j <= n.-1 => f`_j.+1 *+ j.+1].

deriv_tfps
     : {tfps R n} -> {tfps R n.-1}
```

Classical formulas

```
Fact derivD_tfps f g : (f + g)^`() = f^`()%tfps + g^`()%tfps.
Theorem derivM_tfps n (f g : {tfps R n}) :
  (f * g)^`() = f^`()%tfps * (trXnt n.-1 g) + (trXnt n.-1 f) * g^`()%tfps.
```

All the formula for analysis:

- primitive
- exponential, logarithm
- composition $F(G(X))$ (needs $G_0 = 0$ no convergence).

More structure on formal power series

Example: formal derivative

```
Definition deriv_tfps f := [tfps j <= n.-1 => f`_j.+1 *+ j.+1].

deriv_tfps
     : {tfps R n} -> {tfps R n.-1}
```

Classical formulas

```
Fact derivD_tfps f g : (f + g)^`() = f^`()%tfps + g^`()%tfps.
Theorem derivM_tfps n (f g : {tfps R n}) :
  (f * g)^`() = f^`()%tfps * (trXnt n.-1 g) + (trXnt n.-1 f) * g^`()%tfps.
```

All the formula for analysis:

- primitive
- exponential, logarithm
- composition $F(G(X))$ (needs $G_0 = 0$ no convergence).

Infinite power series

Using the classical axiom from Mathcomp's analysis.

Problem

One can construct them from scratch, but one has to redo all the work of polynomials.

Question

Is there a way to get infinite power series from truncated one ?

Limit of $K[[X]]_n$ as n tends to infinity ? In what sense ?

Infinite power series

Using the classical axiom from Mathcomp's analysis.

Problem

One can construct them from scratch, but one has to redo all the work of polynomials.

Question

Is there a way to get infinite power series from truncated one ?

Limit of $K[[X]]_n$ as n tends to infinity ? In what sense ?

Infinite power series

Using the classical axiom from Mathcomp's analysis.

Problem

One can construct them from scratch, but one has to redo all the work of polynomials.

Question

Is there a way to get infinite power series from truncated one ?

Limit of $K[[X]]_n$ as n tends to infinity ? In what sense ?

Categorical Limit

Suitable notion: (Categorical) Inverse Limit.

Warning (Inverse limit $\neq$ topological limit)

Algebraic notion of limit coming from category theory $\neq$ notion of convergence of the series $F(X)$ for a given value of X.

In particular:

- There is no topology involved;
- We don't care about the radius of convergence of a series: The series $\sum_n n!z^n$ is a perfectly valid power series, though its convergence radius is 0.
- The moto here is: all coefficients must be computed using only finitely many algebraic operation.

Inverse (projective) systems

- I: ordered directed set – only need nonnegative integers.
- In an algebraic category (e.g. ring with ring morphisms)

Definition

An inverse system is given by

- *Objects: a sequence* $(A_i)_{i \in I}$
- *Bonding morphisms: for each* $i \leq j$ *a morphism* $\phi_{i,j} : A_j \mapsto A_i$ *(beware the inverse direction).*

such that for all $i \in I$, *the morphism* $\phi_{i,i}$ *is the identity and*

- *Compatibility: for all* $i \leq j \leq k$, *one has* $\phi_{i,j} \circ \phi_{j,k} = \phi_{i,k}$

$$\begin{tikzcd} A_i & A_j \arrow[l, "\phi_{ij}"'] & A_k \arrow[l, "\phi_{jk}"'] \arrow[ll, bend right, "\phi_{ik}"'] \end{tikzcd}$$

Inverse (projective) systems

Definition

An inverse limit of an inverse system, is given by

- *Object:* A
- *Projection morphisms: for each* i *a morphism* $\mu_i : A \mapsto A_i$.

such that

- *Compatibility: for all* $i \leq j$, *one has* $\phi_{i,j} \circ \mu_j = \phi_i$

$$\begin{tikzcd}
A_i & A_j \arrow[l, "\phi_{ij}"'] & A_k \arrow[l, "\phi_{jk}"'] \arrow[ll, bend right, "\phi_{ik}"'] \\
& A \arrow[ul, "\mu_i"] \arrow[u, "\mu_j"'] \arrow[ur, "\mu_k"'] &
\end{tikzcd}$$

Inverse limits universal property

Theorem

Inverse limit *do exists and are unique* *(upto isomorphism) in most algebraic categories.*

Universal property:

$$
\begin{tikzcd}
A_i & & A_j \arrow[ll, "\phi_{ij}"'] \\
& A \arrow[ul, "\mu_i"] \arrow[ur, "\mu_j"'] & \\
& Y \arrow[u, "U"'] \arrow[uul, "\nu_i"] \arrow[uur, "\nu_j"'] &
\end{tikzcd}
$$

Proof of the abstract nonsense

Dealing with concrete categories (Objects are Sets).

Definition

A cone of an inverse system is a sequence $c = (c_i)_{i \in I}$ where $x_i \in A_i$ such that

$$\phi_{i,j}(c_j) = c_i \; .$$

Defining

- A : the set of cones
- $\mu_i(c) := c_i$.

constructs indeed an inverse limit !

Inverse systems and limits in Mathcomp

```
Variable Obj : I -> Type.
Variable bonding : (forall i j, i <= j -> Obj j -> Obj i).
Record invsys : Type := InvSys {
      invsys_inh : I;
      invsys_id  : forall i (Hii : i <= i), (bonding Hii) =1 id;
      invsys_comp : forall i j k  (Hij : i <= j) (Hjk : j <= k),
          (bonding Hij) \o (bonding Hjk) =1 (bonding (le_trans Hij Hjk));
  }.
(* The Mixin formalizes the universal property *)
Record mixin_of (TLim : Type) := Mixin {
  invlim_proj : forall i, TLim -> Obj i;
  invlim_ind  : forall (T : Type) (f : forall i, T -> Obj i),
      (cone Sys f) -> T -> TLim;
  _ : cone Sys invlim_proj;
  _ : forall T (f : forall i, T -> Obj i) (Hcone : cone Sys f),
      forall i, invlim_proj i \o invlim_ind Hcone =1 f i;
  _ : forall T (f : forall i, T -> Obj i) (Hcone : cone Sys f),
      forall (ind : T -> TLim),
        (forall i, invlim_proj i \o ind =1 f i) ->
        ind =1 invlim_ind Hcone
  }.
```

Back to power series

```
Definition fps_bond := fun (i j : nat) of (i <= j)%O => @trXnt R j i.

Record fpseries := FPSeries { seriesfun : nat -> R }.
Definition fpsproj n (f : {fps R}) : {tfps R n} := [tfps i <= n => f``_i].
Lemma fpsprojP : cone fps_invsys fpsproj.

Notation "''pi_' i" := (fpsproj i).
Lemma fpsprojE x y : (forall i : nat, 'pi_i x = 'pi_i y) -> x = y.

Canonical fps_invlimType := Eval hnf in InvLimType {fps R} fps_invlimMixin.
```

Now algebraic structures are for free:

```
Canonical fps_zmodType :=
  Eval hnf in ZmodType {fps R} [zmodMixin of {fps R} by <-].
Canonical fps_ringType :=
  Eval hnf in RingType {fps R} [ringMixin of {fps R} by <-].
[...]
```

Back to power series

One can reuse everything from truncated power series !

Example: derivative of a product:

```
Theorem derivM_tfps n (f g : {tfps R n}) :
  (f * g)^`() = f^`()%tfps * (trXnt n.-1 g) + (trXnt n.-1 f) * g^`()%tfps.

Theorem derivM_fps (f g : {fps R}) :
  (f * g)^`()%fps = f^`()%fps * g + f * g^`()%fps.
Proof.
apply invlimE => i.
rewrite !(proj_simpl, proj_deriv_fps) derivM_tfps /=.
by rewrite -!fps_bondE !ilprojE.
Qed.
```

Sommaire

Back to Catalan numbers

Theorem

Suppose that $(C_i)_{i\in\mathbb{N}}$ *verifies*

$$C_0 = 1 \quad et \quad C_n = \sum_{k=0}^{n-1} C_k C_{n-1-k} \, ,$$

then

$$T_n = \frac{1}{n+1}\binom{2n}{2}$$

```
forall C : nat -> nat,
       C 0 = 1 ->
       (forall n : nat, C n.+1 = \sum_(i < n.+1) C i * C (n - i)) ->
       forall i : nat, C i = 'C(i.*2, i) %/ i.+1
```

Proofs in mathcomp

$$2 \times 3 + 1 = 7 \text{ proofs !}$$

- Truncated or infinite power series

```
Definition FC : {fps Rat} := \fps (C i)%:R .X^i.
Proposition FC_algebraic_eq : FC = 1 + ''X * FC ^+ 2.

Definition FC : {tfps Rat n} := [tfps i => (C i)%:R].
Proposition FC_algebraic_eq : FC = 1 + \X * FC ^+ 2.
```

3 methods to extract the coefficients:

- generalized Newton binomial formula
- Lagrange inversion formula
- holonomic computation

+ 1 bijective proof

Generalized Newton binomial formula

$$F(X) = \frac{1-\sqrt{1-4X}}{2X}$$

Theorem

For any α (non necessarily integer):

$$(1+X)^\alpha = \sum_n \frac{\alpha(\alpha-1)\dots(\alpha-m+1)}{m!}X^n$$

```
Theorem coef_expr1cX c a m :
  ((1 + c *: ''X) ^^ a)%fps``_m =
  c ^+ m * \prod_(i < m) (a - i%:R) / m`!%:R :> R.
```

General methods ?

Problem

Newton's only works for equation which are solvable by radicals.

More general solutions:

- Lagrange inversion
- Holonomic computation

Lagrange formula

Fact: $\{$series starting with $X\}$ is a group for $\circ$ (neutral X).

Proposition (Lagrange Inversion)

$F(X) = X + \dots$ has a unique inverse F^* (Lagrange inverse):

$$F(F^*(X)) = F^*(F(X)) = X\,.$$

Writing $F = \frac{X}{G}$ fixed point version $F^*(X) = X \cdot G(F^*(X))$.

Its coefficient are given by $[X^{i+1}](F^*) = \dfrac{[X^i](G^{i+1})}{i+1}$.

For catalan: $G(X) = (1+X)^2$

```
Proposition FC_fixpoint_eq : FC - 1 = lagrfix ((1 + ''X) ^+ 2).
```

Holonomic computation

- F is rational if $F = \frac{N}{P}$ with N, P polynomials.
- F is algebraic if there exists polynomials $(P_0, \ldots, P_d)$ s.t.

$$P_0 + P_1 F + P_2 F^2 + \ldots P_d F^d = 0 \,.$$

- F is holonomic if there exists polynomials $(P_c, P_0, \ldots, P_d)$ s.t.

$$P_C + P_0 F + P_1 F' + P_2 F'' + \ldots P_d F^{(d)} = 0 \,.$$

Theorem

rational $\rightarrow$ *algebraic* $\rightarrow$ *holonomic.*

Catalan from holonomic computation

$$F = 1 + X \cdot F^2$$

gives

$$(1 - X^2)F + (1 - X^4)XF' = 1\,.$$

```
Proposition FC_differential_eq n :      (* truncated series version *)
   (1 - \X *+ 2) * (FC n.+1) + (1 - \X *+ 4) * tmulX (FC n.+1)^`() = 1.

Proposition FC_differential_eq :        (* infinite series version *)
  (1 - ''X *+ 2) * FC + (1 - ''X *+ 4) * ''X * FC^`() = 1.
```

Extracting the coefficients gives the simplest possible recurrence:

$$(n+2)C_{n+1} = (4n+2)C_n\,.$$

Sommaire

Conclusion

- Once we have the formalization of power series, using them is quite simple (300 lines for the three proofs)
- I strongly regret I didn't had them for Littlewood-Richardson !
- Automation (eg: `ring` tactic) should make it even shorter;
- Much shorter that the bijective proofs (500 lines);
- Using truncated power series makes the proof only slightly longer (to deal with the degree bound), but the statement more complicated (degree bound, 2 different multiplications by X, keeping or adding one to the degree).
- Work in progress on limits (need help with Canonical / Mixin / HierarchyBuilder).
- Everything I presented here could be entirely automatized thanks to computer algebra (see e.g. `combstruct` Maple©).