# A Computer-Algebra-Based Formal Proof of the Irrationality of $\zeta(3)$

Frédéric Chyzak, Assia Mahboubi, Thomas Sibut-Pinote, Enrico Tassi

# A Computer-Algebra-Based Formal Proof of the Irrationality of $\zeta(3)$

Frédéric Chyzak[1], Assia Mahboubi[1], Thomas Sibut-Pinote[2], Enrico Tassi[1]

[1] Inria (France)
[2] ENS de Lyon (France)

**Abstract.** This paper describes the formal verification of an irrationality proof of $\zeta(3)$, the evaluation of the Riemann zeta function, using the Coq proof assistant. This result was first proved by Apéry in 1978, and the proof we have formalized follows the path of his original presentation. The crux of this proof is to establish that some sequences satisfy a common recurrence. We formally prove this result by an a posteriori verification of calculations performed by computer algebra algorithms in a Maple session. The rest of the proof combines arithmetical ingredients and some asymptotic analysis that we conduct by extending the Mathematical Components libraries. The formalization of this proof is complete up to a weak corollary of the Prime Number Theorem.

## 1 Introduction

The irrationality status of the evaluations of the Riemann $\zeta$-function at positive odd integers is a long-standing challenge of number theory. To date, $\zeta(3)$ is the only one known to be irrational, although recent advances obtained by Rivoal [20] and Zudilin [25] showed that one at least of the numbers $\zeta(5), \ldots, \zeta(11)$ must be irrational. The number $\zeta(3)$ is sometimes referred to as the *Apéry constant*, after Roger Apéry who first proved that it is irrational [3]. As reported by van der Poorten [22], Apéry announced this astonishing result by giving a rather obscure lecture that raised more skepticism than enthusiasm among the audience. His exposition indeed involved a number of suspicious assertions, proclaimed without a proof, among which was a mysterious common recurrence for two given sequences (see Lemma 2). After two months of work, however, Cohen, Lenstra, and van der Poorten completed, with the help of Zagier, a verification of Apéry's proof.

**Theorem 1 (Apéry, 1978).** *The constant $\zeta(3)$ is irrational.*

Almost at the same time, symbolic computation was emerging as a scientific area of its own, getting fame with the Risch algorithm [19] for indefinite integration. It gradually provided efficient computer implementations and got attention in experimental mathematics. Beside commutative algebra, differential and recurrence equations remained a central research topic of computer algebra over the years. In particular, the sequences used by Apéry in his proof belong to a class of objects well known to combinatorialists and computer-algebraists. Following seminal work of Zeilberger's [23], algorithms have been designed and implemented in computer-algebra systems, which are able to obtain linear recurrences

for these sequences. For instance the Maple packages gfun and Mgfun (both distributed as part of the Algolib [2] library) implement these algorithms, among other. Basing on this implementation, Salvy wrote a Maple worksheet [21] that follows Apéry's original method but interlaces Maple calculations with human-written parts, illustrating how parts of this proof, including the discovery of Apéry's mysterious recurrence, can be performed by computations.

In the present paper, we describe a formal proof of Theorem 1, based on a Maple session, in the Coq proof assistant. The computer-algebra system is used in a skeptical way [16], to produce conjectures that are a posteriori proved formally. Alternative proofs are known for Theorem 1, as for instance the elegant one proposed by Beukers [5] shortly after Apéry. Our motivation however was to devise a protocol to obtain formal proofs of computer-algebra-generated recurrences in a systematic way. Interestingly, this work challenges the common belief in the computer-algebra community that such an a posteriori checking can be automatized. In addition to the formal verification of these computer-algebra-produced assertions, we have also machine-checked the rest of the proof of irrationality, which involves both elementary number theory and some asymptotic analysis. The latter part of the proof essentially consists in a formal study of the asymptotic behaviors of some sums and of tails of sums. Our formal proof is complete, up to a weak corollary of the repartition of prime numbers that we use as an assumption.

In Section 2, we outline a proof of Theorem 1. Section 3 presents the algorithms which are run in the Maple session we base on. Section 4 describes the formalization of the formal proof we obtain from the data produced by the computer-algebra system. Section 5 provides some concluding remarks and some perspectives for future work.

Our Maple and Coq scripts will be found at `http://specfun.inria.fr/zeta-of-3/`.

## 2 From Apéry's recurrence to the irrationality of $\zeta(3)$

In this section, we outline the path we have followed in our formalization, highlighting the places where we resorted to more elementary variants than Salvy or van der Poorten. In particular, Section 2.3 describes a simple argument we devised to simplify the proof of asymptotic considerations.

### 2.1 Overview

In all what follows, a Cauchy real (number) $x$ is a sequence of rational numbers $(x_n)_{n\in\mathbb{N}}$ for which there exists a function $m_x : \mathbb{Q} \to \mathbb{N}$, such that for any $\epsilon > 0$ and any indices $i$ and $j$, having $i \geq m_x(\epsilon)$ and $j \geq m_x(\epsilon)$ implies $|x_i - x_j| \leq \epsilon$.

**Proposition 1.** *The sequence* $z_n = \sum_{m=1}^{n} \frac{1}{m^3}$ *is a Cauchy real.*

The Cauchy real of Proposition 1 is our definition for $\zeta(3)$. Consider the two sequences $a$ and $b$ of rational numbers defined as:

$$a_n = \sum_{k=0}^{n} \binom{n}{k}^2 \binom{n+k}{k}^2, \qquad b_n = a_n z_n + \sum_{k=1}^{n} \sum_{m=1}^{k} \frac{(-1)^{m+1} \binom{n}{k}^2 \binom{n+k}{k}^2}{2m^3 \binom{n}{m} \binom{n+m}{m}}. \quad (1)$$

Introducing the auxiliary sequences of *real* numbers:

$$\delta_n = a_n\zeta(3) - b_n, \quad \sigma_n = 2\ell_n^3\delta_n, \quad \text{for } \ell_n \text{ the lcm of the integers } 1,\dots,n, \quad (2)$$

the proof goes by showing that the sequence $(\sigma_n)_{n\in\mathbb{N}}$ has positive values and tends to zero. Now if $\zeta(3)$ was a rational number, then for $n$ large enough, every $\sigma_n$ would be a (positive) *integer*, preventing $\sigma$ from tending to zero.

### 2.2 Arithmetics, number theory

We extend the usual definition of binomial coefficients $\binom{n}{k}$ for $n,k\in\mathbb{N}$ to $n,k\in\mathbb{Z}$ by enforcing the Pascal triangle recurrence $\binom{n+1}{k+1} = \binom{n}{k+1} + \binom{n}{k}$ for all $n,k\in\mathbb{Z}$. Although this extension is not required by the present proof, it spares us some spurious considerations about subtraction over $\mathbb{N}$. Binomial coefficients being integers, $a_n$ is also an integer for any nonnegative $n\in\mathbb{N}$.

An important property of the sequence $(b_n)_{n\in\mathbb{N}}$ is that for any $n\in\mathbb{N}$, the product $2\ell_n^3 b_n$ is an integer. Therefore if $\zeta(3)$ were a rational number, then $\ell_n\zeta(3)$, and hence $\sigma_n = 2\ell_n^3(a_n\zeta(3) - b_n)$, would be an integer for $n$ larger than the denominator of $\zeta(3)$. We follow the argument described by Salvy in [21], and show that each summand in the double sum defining $b_n$ has a denominator that divides $2\ell_n^3$: after a suitable re-organization in the expression of the summand, which uses standard properties of binomial coefficients, this follows easily from the following slightly less standard property of theirs:

**Lemma 1.** *For any integers $i,j,n$ such that $1\le j\le i\le n$, $j\binom{i}{j}$ divides $\ell_n$.*

Lemma 1 is considered as folklore in number theory. Its proof consists in showing that for any prime $p$, the $p$-valuation of $j\binom{i}{j}$ is smaller than the one of $\ell_n$.

Standard presentations of Apéry's proof make use of the asymptotic bound $\ell_n = e^{n\,(1+o(1))}$, which is a corollary of the distribution of the prime numbers. A bound $3^n$ is however tight enough for our purpose and has been proved by several independent and elementary proofs, for instance by Hanson [14] and Feng [12]. However, we have not yet formalized any proof of this ingredient, which is completely independent from the rest of the irrationality proof. More precisely, our formal proof is parametrized by the following assumption:

**Proposition 2.** *There exists two positive rationals $K$ and $r$, with $r^3 < 33$, such that for any large enough integer $n$, $\ell_n < Kr^n$.*

### 2.3 Consequences of Apéry's recurrence

The Cauchy sequence $(b_n/a_n)_{n\in\mathbb{N}}$ tends to $\zeta(3)$, thus $\delta_n$ tends to zero. In this section, we prove that it does so fast enough to compensate for $\ell_n^3$, while being positive. The starting point is Apéry's recurrence, (3) below:

**Lemma 2.** *For $n\ge 0$, the sequences $(a_n)_{n\in\mathbb{N}}$ and $(b_n)_{n\in\mathbb{N}}$ satisfy the* same *second-order recurrence:*

$$(n+2)^3 y_{n+2} - (17n^2+51n+39)(2n+3)y_{n+1} + (n+1)^3 y_n = 0. \quad (3)$$

Salvy's worksheet [21] demonstrates in particular how to obtain this common recurrence by Maple calculations, performed by the Algolib library [2]. Following van der Poorten [22], we next use Lemma 2 (and initial conditions) to obtain a closed form of the Casoratian $w_n = b_{n+1}a_n - b_n a_{n+1}$. Indeed, we prove $w_n = \frac{6}{(n+1)^3}$ for $n \geq 2$. From this, we prove that $\delta_n$, and hence $\sigma_n$, is positive for any $n \geq 2$. We also use the closed form to estimate the growth of $\delta$ in terms of $a$. The result is that there exists a positive rational number $K$ such that $\delta_n \leq \frac{K}{a_n}$ for large enough $n$. Finally, the zero limit of $\sigma_n$ follows from Proposition 2, the behaviour of $\delta$, and Lemma 3 below, which quantifies that $a$ grows fast enough.

**Lemma 3.** $33^n \in O(a_n)$.

*Proof.* Introduce the sequence $\rho_n = a_{n+1}/a_n$ and observe that $\rho_{51} > 33$. We now show that $\rho$ is increasing. Define rational functions $\alpha$ and $\beta$ so that the conclusion of Lemma 2 for $a_n$ rewrites to $a_{n+2} - \alpha(n)a_{n+1} + \beta(n)a_n = 0$ for $n \geq 0$. Now, for any $n \in \mathbb{N}$, introduce the homography $h_n(x) = \alpha(n) - \frac{\beta(n)}{x}$, so that $\rho_{n+1} = h_n(\rho_n)$. Let $x_n$ be the largest root of $x^2 - \alpha(n)x + \beta(n)$. The result follows by induction on $n$ from the fact that $h([1, x_n]) \subset [1, x_n]$ and from the observation that $\rho_2 \in [1, x_2]$. □

## 3 Algorithms on sequences in computer algebra

Lemma 2 is the bottleneck in Apéry's proof. Both sums $a_n$ and $b_n$ in there are instances of *parametrised summation*: they follow the pattern $F_n = \sum_{k=\alpha(n)}^{\beta(n)} f_{n,k}$ in which the summand $f_{n,k}$, potentially the bounds, and thus the sum, depend on a parameter $n$. This makes it appealing to resort to the algorithmic paradigm of *creative telescoping*, which was developed for this situation in computer algebra.

In order to operate on sequences, computer algebra substitutes implicit representations for explicit representations in terms of named sequences (factorial, binomial, etc). This is the topic of Section 3.1. A typical example of parametrised summation by this approach is provided by the identity $\sum_{k=0}^{n} \binom{n}{k} = 2^n$: from an encoding of the summand $\binom{n}{k}$ by the recurrences

$$\binom{n+1}{k} = \frac{n+1}{n+1-k}\binom{n}{k}, \quad \binom{n}{k+1} = \frac{n-k}{k+1}\binom{n}{k}, \tag{4}$$

deriving the relation, with finite difference with respect to $k$ in right-hand side,

$$\binom{n+1}{k} - 2\binom{n}{k} = \left(\binom{n+1}{k+1} - \binom{n}{k+1}\right) - \left(\binom{n+1}{k} - \binom{n}{k}\right) \tag{5}$$

is sufficient to derive the explicit form $2^n$, as will be explained below.

### 3.1 Recurrences as a data structure for sequences

The implicit representation fruitfully introduced by computer algebra to deal with sequences are systems of linear recurrences. In this spirit, *∂-finite sequences* are algebraic objects that model mathematical sequences and enjoy nice algorithmic properties. Notably, the finiteness property of their definition makes algorithmic most operations under which the class of ∂-finite sequences is stable.

A $\partial$-finite sequence (see [7] for a complete exposition of the subject) is an element of a module over the non-commutative ring $\mathcal{A}$ of skew polynomials in the indeterminates $S_n$ and $S_k$, with coefficients in the rational-function field $\mathbb{Q}(n,k)$, and commutation rule $S_n^i S_k^j c(n,k) = c(n+i,k+j) S_n^i S_k^j$. A skew polynomial $P = \sum_{(i,j)\in I} p_{i,j}(n,k) S_n^i S_k^j \in \mathcal{A}$ acts on a "sequence" $f$ by $(P \cdot f)_{n,k} = \sum_{(i,j)\in I} p_{i,j}(n,k) f_{n+i,k+j}$, where subscripts denote evaluation. For example for $f_{n,k} = \binom{n}{k}$, the recurrences (4) once rewritten as equalities to zero can be represented as $P \cdot f = 0$ for $P = S_n - \frac{n+1}{n+1-k}$ and $P = S_k - \frac{n-k}{k+1}$, respectively.

To any $\partial$-finite sequence $f$, one associates the set of skew polynomials that annihilate it. This set, $\{P \in \mathcal{A} : P \cdot f = 0\}$ is a left ideal of $\mathcal{A}$, named the *annihilating ideal* of $f$, and denoted $\operatorname{ann} f$. A non-commutative extension of the usual Gröbner-basis theory is available, together with algorithmic analogues. In this setting, a good representation of a $\partial$-finite sequence is obtained as a Gröbner basis of $\operatorname{ann} f$ for a suitable ordering on the monomials in $S_n$ and $S_k$. For the example of $f_{n,k} = \binom{n}{k}$, a Gröbner basis consists of both already-mentioned skew polynomials encoding (4). In general, a Gröbner basis provides us with a (vectorial) basis of the quotient module $\mathcal{A}/\operatorname{ann} f$. This basis can be explicitly written in the form $B = \{f_{n+i,k+j}\}_{(i,j)\in\mathcal{U}}$, where the finite set $\mathcal{U}$ of indices is given as the part under the classical stair shape of the Gröbner-basis theory. Given a Gröbner basis GB for $\operatorname{ann} f$, the normal form $\mathrm{NF}(p, \mathrm{GB})$ is unique for any $p \in \mathcal{A}$. Again in the binomial example, the finite set is $\mathcal{U} = \{(0,0)\}$, and normal forms are rational functions.

This is the basis of algorithms for a number of operations under which the $\partial$-finite class is stable, which all process by looking for enough dependencies between normal forms: application of an operator, addition, product. The case of summing a sequence $(f_{n,k})$ into a parametrised sum $F_n = \sum_{k=0}^{n} f_{n,k}$ is more involved: it performs according to the *method of creative telescoping* [24], in two stages. First, an *algorithmic* step determines pairs $(P, Q)$ satisfying

$$P \cdot f = (S_k - 1) Q \cdot f \tag{6}$$

with $P \in \mathcal{A}'$ and $Q \in \mathcal{A}$, where $\mathcal{A}'$ is the subalgebra $\mathbb{Q}(n)\langle S_n\rangle$ of $\mathcal{A}$. To continue with our example $f_{n,k} = \binom{n}{k}$, Eq. (5) can be recast into this framework by choosing $P = S_n - 2$ and $Q = S_n - 1$. Second, a *systematic* but not fully algorithmic step follows: summing (6) for $k$ between 0 and $n + \deg_{S_n} P$ yields

$$(P \cdot F)_n = (Q \cdot f)_{k=n+\deg_{S_n} P+1} - (Q \cdot f)_{k=0}. \tag{7}$$

Continuing with our binomial example, summing (5) (or its equivalent form (6)) for $k$ from 0 to $n+1$ (and taking special values into account) yields $\sum_{k=0}^{n+1} \binom{n+1}{k} - 2\sum_{k=0}^{n} \binom{n}{k} = 0$, a special form of (7) with right-hand side canceling to zero. The formula (7) in fact assumes several hypotheses that hold not so often in practice; this will be formalized by Eq. (8) below.

### 3.2 Apéry's sequences are $\partial$-finite constructions

The sequences $a$ and $b$ in (1) are $\partial$-finite: they have been announced to be solutions of (3). But more precisely, they can be viewed as constructed from

| step | explicit form | GB | operation | input(s) |
|---|---|---|---|---|
| 1 | $c_{n,k} = \binom{n}{k}^2 \binom{n+k}{k}^2$ | $C$ | direct | |
| 2 | $a_n = \sum_{k=1}^{n} c_{n,k}$ | $A$ | creative telescoping | $C$ |
| 3 | $d_{n,m} = \frac{(-1)^{m+1}}{2m^3\binom{n}{m}\binom{n+m}{m}}$ | $D$ | direct | |
| 4 | $s_{n,k} = \sum_{m=1}^{k} d_{n,m}$ | $S$ | creative telescoping | $D$ |
| 5 | $z_n = \sum_{m=1}^{n} \frac{1}{m^3}$ | $Z$ | direct | |
| 6 | $u_{n,k} = z_n + s_{n,k}$ | $U$ | addition | $Z$ and $S$ |
| 7 | $v_{n,k} = c_{n,k} u_{n,k}$ | $V$ | product | $C$ and $U$ |
| 8 | $b_n = \sum_{k=1}^{n} v_{n,k}$ | $B$ | creative telescoping | $V$ |

Table 1: Construction of $a_n$ and $b_n$: At each step, the Gröbner basis named in column GB, which annihilates the sequence given in explicit form, is obtained by the corresponding operation *on ideals*, with input(s) given on the last column.

"atomic" sequences by operations under which the class of $\partial$-finite sequences is stable. This is summarised in Table 1.

Both systems $C$ and $D$ are first-order systems obtained directly as easy consequences of (4); they consist respectively of expressions for $c_{n+1,k}$ and $c_{n,k+1}$ in terms of $c_{n,k}$ and of expressions for $d_{n+1,k}$ and $d_{n,k+1}$ in terms of $d_{n,k}$. The case of $Z$ is almost the same: it is not a parametrised summation but an indefinite summation. A (univariate, second-order) recurrence is easily obtained for it, without referring to any creative telescoping.

For each of $C$, $D$, and $V$, which undergo a summation operation, we obtain creative-telescoping pairs $(P, Q)$: one for $C$ for a set $\mathcal{U} = \{(0,0)\}$; one for $V$ for a set $\mathcal{U} = \{(0,0),(1,0),(0,1)\}$; four for $D$ for the same set. In all cases, we have had our computer-algebra program (informally) ensure that the corresponding $P$ cancels the sum. For $C$ and $V$, the single $P$ thus obtained is (trivially) a Gröbner basis of the annihilating ideal of $A$ or $B$, respectively. But for $D$, the four $P$ have to be recombined, leading to three operators.

It should be observed that Gröbner bases is the only data used by computer-algebra algorithms in the program above, including when simplifying the right-hand side in (7) and in its generalization to come, Eq. (8) below. For instance, computer algebra computes the system $V$ from the systems $C$ and $U$ alone, without resorting to any other knowledge of particular values of $c$ and $u$ in Table 1, and so does our formal proof to verify that the pointwise product $xy$ is annihilated by $V$ whenever $C$ and $U$ respectively annihilate sequences $x$ and $y$. In other words, although a $\partial$-finite sequence is fully determined by a system of recurrences and sufficiently many initial conditions (that is, values of $u_{i,j}$ for small values of $i$ and $j$), we do not maintain those values along our proofs.

Our formal proof as well models each sequence by a system of recurrences obtained solely from the operators of the Gröbner basis. We hence bet that the computer-algebra implementation of the algorithmic operations of addition, product, and summation, as well as the parts of our Maple script relying on less algorithmic operations do not take decisions based on private knowledge they

could have on their input, viewed as a specific solution to the recurrence system used to encode it. Would the implementation do so without letting us know, then our a posteriori verification would have required guessing an appropriate description of this additional knowledge, like operators for specializations of the sequences. Fortunately, the Mgfun package we used has the wanted property.

### 3.3 Provisos and sound creative telescoping

Observe the denominators in (4): they prevent the rules to be used, respectively when $k = n + 1$ and $k = -1$. For example, one can "almost prove" Pascal's triangle rule by

$$\binom{n+1}{k+1} - \binom{n}{k+1} - \binom{n}{k} = \left(\frac{n+1}{n-k}\frac{n-k}{k+1} - \frac{n-k}{k+1} - 1\right)\binom{n}{k} = 0 \times \binom{n}{k} = 0,$$

*but this requires* $k \neq -1$ *and* $k \neq n$. Therefore, this does not prove Pascal's rule for all $n$ and $k$. The phenomenon is general: computer algebra is unable to take denominators into account. This incomplete modelling of sequences by algebraic objects may cast doubt on these computer-algebra proofs, in particular when it comes to the output of creative-telescoping algorithms.

By contrast, in our formal proofs, we augmented the recurrences with provisos that restrict their applicability. In this setting, we validate a candidate identity like the Pascal triangle rule by a normalization modulo the elements of a Gröbner basis plus a verification that this normalization only involves legal instances of the recurrences. In the case of creative telescoping, Eq. (6) takes the form:

$$(n,k) \notin \Delta \Rightarrow (P \cdot f_{\_,k})_n = (Q \cdot f)_{n,k+1} - (Q \cdot f)_{n,k}, \tag{8}$$

where $\Delta \subset \mathbb{Z}^2$ guards the relation and where $f_{\_,j}$ denotes the univariate sequence obtained by specializing the second argument of $f$ to $j$. Thus our formal analogue of Eq. (7) takes this restriction into account and has the shape

$$\begin{aligned}(P \cdot F)_n = \Big((Q \cdot f)_{n,n+\beta+1} - (Q \cdot f)_{n,\alpha}\Big) + \sum_{i=1}^{r}\sum_{j=1}^{i} p_i(n)\, f_{n+i,n+\beta+j} \\ + \sum_{\alpha \le k \le n+\beta \ \wedge \ (n,k)\in\Delta} (P \cdot f_{\_,k})_n - (Q \cdot f)_{n,k+1} + (Q \cdot f)_{n,k},\end{aligned} \tag{9}$$

for $F$ the sequence with general term $F_n = \sum_{k=\alpha}^{n+\beta} f_{n,k}$. The proof of identity (9) is a straightforward reordering of the terms of the left-hand side, $(P \cdot F)_n = \sum_{i=0}^{r} p(n)\, F_{n+i}$, after unfolding the definition of $F$ and applying relation (8) everywhere allowed in the interval $\alpha \le k \le n + \beta$. The first part of the right-hand side is the usual difference of border terms, already present in Eq. (7). The middle part is a collection of terms that arise from the fact that the upper bound of the sum defining $F_n$ depends linearly on $n$ and that we do not assume any nullity of the summand outside the summation domain. The last part, which we will call the singular part, witnesses the possible partial domain of validity of relation (8). The operator $P$ is a valid recurrence for the sequence $F$ if the right-hand side of Eq. (9) normalizes to zero, at least outside of an algebraic locus that will guard the recurrence.

## 4 Formal proof of the common recurrence

This section describes the computer-algebra-aided formal proof of Lemma 2, based on a Maple session implementing the program described in Table 1.

### 4.1 Generated operators, hand-written provisos, and formal proofs

For each step in Table 1, we make use of the data computed by the Maple session in a systematic way. Figure 1 illustrates this pattern on the example of step 7. As mentioned in Section 3.3, we annotate each operator produced by the computer-algebra program with provisos (see below) and turn it this way into a conditional recurrence predicate on sequences. To each sequence in the program corresponds a file defining the corresponding conditional recurrences, for instance `annotated_recs_c`, `annotated_recs_u`, and `annotated_recs_v` for $c$, $u$, and $v$, respectively. More precisely these files contain *all* the operators obtained by the Maple script for a given sequence, not only the Gröbner basis. We use rounded boxes to depict the files that store the *definitions* of these predicates. These are generated by the Maple script which pretty-prints its output in Coq syntax, with the exception of the definition of provisos. Throughout this section, a maple leaf tags the files that are generated by our Maple script. Yet automating these annotations is currently out of reach.

In our formal proof, each step in Table 1 consists in proving that some conditional recurrences on a composed sequence can be proved from some conditional recurrences known for the arguments of the operation. We use square boxes to depict the files that store these *formal proofs*. The statement of the theorems proved in these files are composed from the predicates defined in the round boxes: a dashed line points to (predicates used to state) conclusions and a labelled solid line points to (predicates used to state) hypotheses.

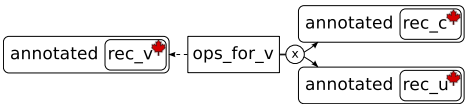


Fig. 1: Proving that $V$ is $C \times U$

### 4.2 Definitions of conditional recurrence predicates

All files defining the conditional recurrence predicates obtained from the operators annihilating sequences of the program share the same structure. An excerpt of the generated part of the file `annotated_recs_c` is displayed on Listing 1.1. The constants `Sn`, `Sk`, and `CT_premise` are recurrences predicates, defined in terms of a bound variable `c`. Constants `Sn` and `Sk` are elements of the Gröbner basis. The definition of these recurrences is named to reflect the term it rewrites, e.g., the left-hand sides in (4): these names are the result of pretty-printing the (skew) monomial that encodes these left-hand sides, the prefix `S` standing for

"shift". For example `Sn` is the name of a recurrence defining $c_{n+1,k}$, while `SnSk` would be for $c_{n+1,k+1}$. Rewriting a given term with such an equation makes the term decrease for the order associated with the Gröbner basis. Another part of the file defines the recurrences obtained from a creative-telescoping pair $(P, Q)$ generated for the purpose of the summation defining the sequence $a$.

```
(* Coefficients of every recurrence, P, and Q. *)
Definition Sn00 n k := (n + 1 + k)^2 / (-n - 1 + k)^2.
Definition Sk00 n k := (-n + k)^2 * (n + 1 + k)^2 / (k + 1)^4.
Definition P0 n := (n + 1)^3.
...
(* Conditional recurrences. *)
Definition Sn c := ∀ n k, precond.Sn n k → c (n + 1) k = Sn00 n k * c n k
Definition Sk c := ∀ n k, precond.Sk n k → c n (k + 1) = Sk00 n k * c n k

(* Operators P and Q. *)
Definition P c n := P0 n * c n + P1 n * c (n + 1) + P2 n * c (n + 2).
...
(* Statement P = Δ_k Q. *)
Definition CT_premise c := ∀ n k, precond.CT_premise n k →
  P (c ␣ k) n = Q c n (k + 1) - Q c n k.
```

Listing 1.1: Generated part of annotated_rec_c

Observe that these generated definitions feature named provisos that are in fact placeholders. In the preamble of the file, displayed on Listing 1.2, we provide by a manual annotation a concrete definition for the proviso of each recurrence defined in the generated part. Observe however that part of these definitions can be inferred from the coefficients of the recurrences. For example the $k \neq n+1$ condition in `precond.Sn`, the proviso of recurrence `Sn`, is due to the denominator $(-n-1+k)^2$ of the coefficient `(Sn00n k)`.

```
Module precond.
Definition Sn n k := (k ≠ n + 1) ∧ (n ≠ -1).
Definition Sk n k := (k + 1 ≠ 0) ∧ (n ≠ 0).
Definition CT_premise n k := (n ≥ 0) ∧ (k ≥ 0) ∧ (k < n).
End precond.
```

Listing 1.2: Hand-written provisos in annotated_rec_c

In the last part of the file, see Listing 1.3, a record collects the elements of the Gröbner basis $C$. Maple indeed often produces a larger set of annihilators for a given sequence, for instance `CT_premise` in Listing 1.1 is related to a creative telescoping pair but not to the Gröbner basis. Also, the Gröbner basis can be obtained by refining a first set of annihilators, which happens at step 4 of Table 1.

```
(* Choice of recurrences forming a Groebner basis. *)
Record Annihilators c := { Sn : Sn c; Sk : Sk c }.
```

Listing 1.3: Selection of a Gröbner basis

### 4.3 Formal proofs of a conditional recurrence

We take as a running example the file `ops_for_a`, which models step 2 in Table 1. This file proves theorems about an arbitrary sequence $c$ satisfying the recurrences in the Gröbner basis displayed on Listing 1.3, and about the sequence $a$ by definite summation over $c$.

```
Require Import annotated_recs_c.
Variables (c : int → int → rat) (ann_c : Annihilators c).

Theorem P_eq_Delta_k_Q : CT_premise c. Proof. ... Qed.

Let a n := \sum_(0 ≤ k < n + 1) c n k.
```

The formal proof of lemma `P_eq_Delta_k_Q` is an instance of Eq. 8. Using this property, we prove that the sequence $a$ verifies a conditional recurrence associated to the operator $P$. As suggested in Section 3.1, this proof consists in applying the lemma `punk.sound_telescoping`, which formalizes a sound creative telescoping and in normalizing to zero the resulting right-hand side of Eq. 9. Listing 1.4 displays the first lines of the corresponding proof script, which select and name the three components of the right-hand side of Eq. 9, with self-explanatory names. The resulting proof context is displayed on Listing 1.5.

```
Theorem recApery_a n (nge2 : n ≥ 2) : P a n = 0.
Proof.
rewrite (punk.sound_telescoping P_eq_Delta_k_Q).
set boundary_part := (X in X + _ + _).
set singular_part := (X in _ + X + _).
set overhead_part := (X in _ + _ + X).
```

Listing 1.4: Begining of a proof of sound creative telescoping

```
boundary_part := Q c n (n + 1) - Q c n 0
singular_part := \sum_(0 ≤ i < n + 1 | precond.CT_premise n i)
         P (c ␣ i) n - (Q c n (i + 1) - Q c n i)
overhead_part := \sum_(0 ≤ i < degree P)
          \sum_(0 ≤ j < i) P_i n * c (n + i) (n + j + 1)
============================
 boundary_part + singular_part + overhead_part = 0
```

Listing 1.5: Corresponding goal

In Listing 1.5, `(c ␣ i)` denotes the expression `(fun x => c x i)`, $\texttt{P}_i$ denotes the $i$-th coefficient of the polynomial `P`, and `degree P` the degree of `P` (two in this specific case). Note that we have access to `degree P` because in addition to the definition displayed on Listing 1.1, we also have at our disposal a list representation `[:: P_0; P_1]` of the same operator.

The proof of the goal of Listing 1.5, proceeds in three steps. The first step is to inspect the terms in `singular_part` and to chase ill-formed denominators, like $n - n$. These can arise from the specialisations, like $k = n$, induced when unrolling the definition of (the negation of) `precond.CT_premise`. In our formalization, a division by zero is represented by a conventional value: we check that these terms vanish by natural compensations, independently of the convention,

and we keep only the terms in `singular_part` that represent genuine rational numbers. The second step consists in using the annihilator `ann_c` of the summand to reduce the resulting expression under the stairs of the Gröbner basis. In fact, this latter expression features several collections of terms, that will be reduced to as many independent copies of the stairs. In the present example, we observe two such collections: *(i)* terms that are around the lower bound $(n, 0)$ of the sum, of the form $c_{n,0}, \ldots c_{n,s}$; *(ii)* terms that are around the upper bound $(n, n)$ of the summation, of the form $c_{n,n}, \ldots, c_{n,n+s}$ for a constant $s$. The border terms induce two such collections but there might be more, depending in particular on the shape of the `precond.CT_premise` proviso. For example, the sum $\sum_{k=0}^{n}(-1)^k\binom{n}{k}\binom{3k}{n} = (-3)^n$ leads to a proviso involving $n = 3k + 1$ and similar terms: an additional category of terms around $(n, n/3)$ drifts away from both $(n, 0)$ and $(n, n)$ when $n$ grows.

```
============================
P₂ n * c (n + 2) n + P₁ n * c (n + 1) n +
P₀ n * c n n + Q₀₀ n n * c n n + P₁ n * c (n + 1) (n + 1) +
P₂ n * c (n + 2) (n + 1) + P₂ n * c (n + 2) (n + 2) = 0
```

Listing 1.6: Terms around the upper bound

The collection of terms around the upper bound in our running example is displayed on Listing 1.6. The script of Listing 1.7 reduces this collection under the stairs of `ann_c`, producing the expression displayed on Listing 1.8. The premise of each rule in this basis being an integer linear arithmetic expression, we check its satisfiability using our front-end `intlia` to the `lia` proof command [4], which automates the formal proof of first-order formulae of linear arithmetics.

```
rewrite (ann_c.Sk (n + 2) (n + 1)); last by intlia.
rewrite (ann_c.Sk (n + 2) n); last by intlia.
rewrite (ann_c.Sk (n + 1) n); last by intlia.
rewrite (ann_c.Sn (n + 1) n); last by intlia.
rewrite (ann_c.Sn n n); last by intlia.
set cnn := c n n.
Fail set no_more_c := c _ _.
```

Listing 1.7: Reduction modulo the Gröbner basis of $c$

```
============================
P₂ n * Sn₀₀ (n + 1) n * Sn₀₀ n n * cnn + P₁ n * Sn₀₀ n n * cnn +
P₀ n * cnn + Q₀₀ n n * cnn + P₁ n * Sk₀₀ (n + 1) n * Sn₀₀ n n * cnn +
P₂ n * Sk₀₀ (n + 2) n * Sn₀₀ (n + 1) n * Sn₀₀ n n * cnn +
P₂ n * Sk₀₀ (n + 2) (n + 1) * Sk₀₀ (n + 2) n *
  Sn₀₀ (n + 1) n * Sn₀₀ n n * cnn = 0
```

Listing 1.8: Rational function with folded coefficients

The third and last step consists in checking that the rational-function coefficient of every remaining evaluation of $c$ is zero. For this purpose, we start by unfolding the definitions of the coefficients `P₂`, `Sn₀`. Previous steps kept them carefully folded as these values play no role in the previous normalizations but can lead to pretty large expressions if expanded, significantly slowing down any other proof command. The resulting rational function is proved to be zero by a

combination of the `field` [11] and `lia` [4] proof commands. The former reduces the rational equation into a polynomial one between the cross product of two rational functions. This equation is then solved by the `ring` proof command [18]. The algebraic manipulations performed by `field` produce a set of non-nullity conditions for the denominators. These are solved by the `lia` proof command. To this end, our Maple script generates rational fractions with factored denominators, that happen to feature only linear factors in these examples.

### 4.4 Composing closures and reducing the order of $B$

Figure 2 describes the global dependencies of the files proving all the steps in Table 1. In order to complete the formal proof of Lemma 2, we verify formally in

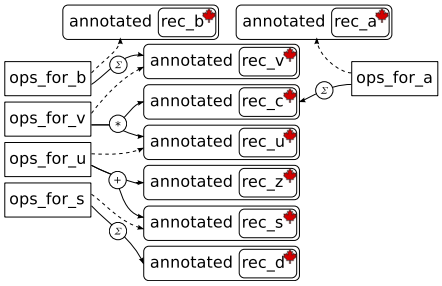


Fig. 2: Formal proofs of Table 1

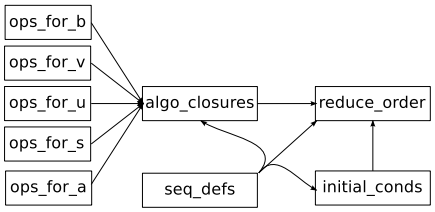


Fig. 3: Formal proof of Lemma 2

file `algo_closures` that each sequence involved in the construction of $a_n$ and $b_n$ is a solution of the corresponding Gröbner system of annotated recurrence, starting from $c_n$, $d_n$, and $z_n$ and applying the lemmas proved in the `ops_for_*` files all the way to the the final conclusions of `ops_for_a` and `ops_for_b`. This proves that $a_n$ is a solution of the recurrence (3) but provides only a recurrence of order four for $b_n$. In file `reduce_order`, we prove that $b$ as well satisfies the recurrence (3) using four evaluations $b_0, b_1, b_2, b_3$ that we compute in file `initial_conds`.

## 5 Conclusion

### 5.1 Formally proving the consequences of Apéry's recurrence

The present paper focuses on the computer-algebra-aided part of our formalization. Another significant part of it addresses the asymptotic study of some sequences of rational numbers. Formalizing this part of the proof requires introducing a type with more inhabitants than just rational numbers, for it deals with the properties of limits of sequences, notably the constant $\zeta(3)$ itself: we hence need to include at least computable real numbers. The construction of Cauchy reals proposed by Cohen in his PhD [9] turned out to perfectly suit our needs, although we had to enrich the existing library with a few basic properties. Indeed, we use the type of Cauchy real numbers mostly to state our theorems, like Theorem 1, but the proofs only involve reasoning with *rational* values of the

sequences. We also benefited from the proof commands provided by Cohen [9] to postpone the definition of "large enough" bounds or moduli of convergence.

The proof of Lemma 3 however involves non-rational real numbers that are not defined as converging sequences but as roots of polynomials. We resort in this case to Cohen's construction of real algebraic numbers [8] and benefit constructively from the decidability of comparison relations. Navigating between the different natures of numbers this irrationality proof features, integer, rational numbers, real algebraic numbers, and Cauchy reals was made possible by the genericity of the core Mathematical Component libraries [1].

The proof of the same Lemma 3 involves two computational steps, namely the observations that $\rho_{51} > 33$ and that $\rho_2 \in [1, x_2]$. These numbers are rather large: for instance the normalized fraction $\rho_{51}$ features integers with more than 70 digits. The issue here is to obtain a formal proof of these facts that both fits with the rest of the proof of Lemma 3 and does not take ages to compute. We used the CoqEAL [10] library and the framework it provides for writing parametric code which can be executed either symbolically, in abstract parts of a proof, or with appropriate data structures for efficient computations. The programs that implement the computation of $\rho_n$ and $x_n$ are very short and boil down to evaluating a (rather small) rational fraction defined recursively. At this place and when computing evaluations of the sequence (see Section 4.4), computations use (interpreted) binary integer arithmetics and are almost instantaneous.

### 5.2 Asymptotic behavior of $\mathrm{lcm}(1, \ldots, n)$

We plan to complete our irrationality proof by providing a formal proof of Proposition 2 following the proof of Hanson [14]. This involves considerations of elementary number theory as well as asymptotic-analysis studies that are rather similar to the ones we dealt with in the present work. The Prime Number Theorem has been machine-checked by Harrison [15] in HOL-Light. We considered importing this result into Coq by using the automated translation tool developed by Keller and Werner [17]. Unfortunately, according to Keller, the import of such an involved proof is currently beyond reach because of its size.

### 5.3 Formal proofs on objects of size typical of computer algebra

The size of the mathematical objects in our formalization makes the interactive writing of proof scripts quite unusual and challenging. For example, the recurrence $P \cdot y = 0$ of order four satisfied by the sequence $b$ spans over 8,000 lines when pretty-printed. Both proofs of the fact $P \cdot b = 0$, named `recApery_b` in our formalization, and of its premise $P \cdot v = (S_k - 1)Q \cdot v$, named `P_eq_Delta_k_Q`, end by normalizing rational functions to 0. Figure 4 reports on the size of the polynomials that occur in these proofs, together with the amount of time needed by Coq's `field` proof command to check that the rational functions are zero.

These objects can be easily manipulated by a computer-algebra system: Maple normalizes $P \cdot b$ to zero in less than 2 seconds. Coq, in its latest stable version (8.4pl3), takes roughly 4 minutes, but of course it produces and checks a formal proof of such normalization. To achieve this reasonable timing, we contributed to version 8.4pl3 a patch to substantially improve the performances of

| lemma | lines | terms | # of digits | | time |
|---|---|---|---|---|---|
| | | | avg | max | (seconds) |
| `P_eq_Delta_k_Q` | 1006 | 1714 | 6 | 13 | 247 |
| `recApery_b` | 7811 | 18602 | 2 | 13 | 179 |

Fig. 4: Statistics about the polynomials in `ops_for_b` and their normalization

rational-function normalization: in Coq 8.4pl2 the very same recurrence requires a bit less than one hour to be normalized to zero.

*Navigating expressions of size typical of computer algebra* Under the circumstances described above, where the goal to be proved cannot even be displayed, the need for uncommon proof commands arises. The pattern matching facilities offered by the `set` command [13] can be used to *probe* the goal for a subterm of the shape `u (n + _) (k + _)`, or to identify all the occurrences of a closed terms.

Unfortunately not all uncommon needs are easily satisfiable with standard proof commands. A typical one is to invoke a computer-algebra system to rearrange an expression in order decide how to proceed with the proof. This was performed by hand, massaging text files to accommodate the small syntactic differences between the syntax of Coq and Maple. This task has been automated in the past in the Coq-Maple bridge developed by Delahaye and Mayero [11], for the purpose of enhancing proof automation in Coq. We foresee to generalise this work by allowing arbitrary Maple commands to let one explore algebraic expressions, and not necessarily prove the correctness of the results.

*Proof-language implementation* The size of the mathematical expressions poses additional efficiency challenges when combined with the ability of the Coq logic to identify terms up to computation. For example a pattern like `(3 * _)` is matched by searching for the head symbol `*` verbatim, but its non-wildcard argument, `3` here, is compared taking computation into account. Given the size of the expressions, the fast filtering performed on the head symbol is not sufficient to discard enough false positives. Comparing two terms up to computation often triggers a full normalization before failing and this is an expensive operation.

The definitions of integer and rational numbers that are part of the Mathematical Components library are designed for the purpose of a comprehensive library of theorems, and not for fast computations: this makes the aforementioned pattern-matching operation often intractable. The commonly accepted idea of having two representations of the same concept, one good for reasoning and one good for computing, linked by a morphism does not work for expressions of this size, as switching from one representation to the other one relies again on pattern matching to push the morphism through the huge expression. For instance, we had to craft boilerplate code, which carefully controls computations and use locked casts for large constant numbers, in order to make the `field` and `lia` tactic work on these data-structures.

One possible work-around would be to have available, in the proof language, two distinct pattern-matching facilities: one that eases reasoning steps by identi-

fying terms up to computation, and another one that performs a dumb syntactic comparison and is efficient even when applied to large, homogeneous expressions.

### 5.4 Theoretical and practical limitations of data structures

*Algebraic rewriting with provisos* An obstruction to turning our approach into a complete protocol is that we do not know how to determine the provisos other than by trial and error. This is connected to the fact that recurrences that have well understood rewriting properties when forming a Gröbner basis (a priori) lose all this nice structure when decorated with provisos. We do not understand what the new critical pairs have to be and how to ensure we give ourselves complete information with our selection of rules-with-provisos that simply lift a Gröbner basis. To the best of our knowledge, there is no literature on the topic yet.

We have been lucky with the few cases in the present formalization, in that we could just guide our reduction by the usual strategy of division by a Gröbner basis: all implied integer premises could be solved, without us understanding why they would.

*Polynomial data structures* Manipulating polynomials in computer algebra is made natural by the ease to keep them in collected form, whether automatically or by easy-to-call procedures. On the other hand, our treatment of recurrences amounts mostly to collecting polynomials and skew polynomials, through the encoding suggested in Section 3.1. A (skew) polynomial data structure, with computational product and division, and the corresponding theory, would help keeping goals tidy and more manageable, especially during the rewriting steps. In particular, the current strategy of reserving the call to `field` to the end is certainly a cause of blow-up of the goals and inefficiency of the computations.

In addition, making it generic with respect to the nature of skew-polynomial commutation would provide similar functionality for both recurrence and differential equations, paving the way to the formalization of integral identities by a known differential counterpart to creative telescoping.

### 5.5 Building on top of state-of-the-art libraries

It was a help to have at our disposal the broad scope of the Mathematical Components library. It was a cause for trouble, too. This research started as an interdisciplinary activity: half of the authors were newcomers to formal proofs. As a matter of fact understanding the sophisticated design of the library turned out to be a very arduous task for them.

We can identify in the combined use of implicit coercions, notations, and structure inference the major source of complexity. Their formalization activity was additionally hindered by the fact that various features of the Coq system do not interact well with the design patterns employed in the Mathematical Components library. The most notable example being search, that is almost ineffective in the context of the algebraic hierarchy.

We believe such problems could be alleviated by: introductory material specific to the Mathematical Components library and written with newcomers in mind; a "teaching/debug" mode in which type inference explicates steps at the desired level of detail; finally a proper searching facility.

## References


[1] Mathematical Components Libraries. http://www.msr-inria.fr/projects/mathematical-components, 2013. Version 1.4. For Coq 8.4pl3.

[2] Algolib. http://algo.inria.fr/libraries/, 2013. Version 17.0. For Maple 17.

[3] R. Apéry. Irrationalité de $\zeta(2)$ et $\zeta(3)$. *Astérisque*, 61, 1979. Société Mathématique de France.

[4] F. Besson. Fast reflexive arithmetic tactics the linear case and beyond. In *TYPES'06*, LNCS, pages 48–62, Berlin, Heidelberg, 2007. Springer-Verlag.

[5] F. Beukers. A note on the irrationality of $\zeta(2)$ and $\zeta(3)$. *Bull. London Math. Soc.*, 11(3):268–272, 1979.

[6] F. Chyzak, A. Mahboubi, and T. Sibut-Pinote. Do creative-telescoping algorithms provide complete proofs? a formal study of Apéry's theorem. Submitted, 2014.

[7] F. Chyzak and B. Salvy. Non-commutative elimination in Ore algebras proves multivariate identities. *J. Symbolic Comput.*, 26(2):187–227, 1998.

[8] C. Cohen. Construction of real algebraic numbers in Coq. In *ITP*, volume 7406 of *LNCS*. Springer, Aug. 2012.

[9] C. Cohen. *Formalized algebraic numbers: construction and first-order theory.* PhD thesis, École polytechnique, Nov 2012.

[10] C. Cohen, M. Dénès, and A. Mörtberg. Refinements for free! In *CPP*, volume 8307 of *LNCS*, pages 147–162. Springer International Publishing, 2013.

[11] D. Delahaye and M. Mayero. Dealing with algebraic expressions over a field in Coq using Maple. *J. Symb. Comput.*, 39(5):569–592, May 2005.

[12] B.-y. Feng. A simple elementary proof for the inequality $d_n < 3^n$. *Acta Math. Appl. Sin. Engl. Ser.*, 21(3):455–458, 2005.

[13] G. Gonthier, A. Mahboubi, and E. Tassi. *A small scale reflection extension for the Coq system*, 2013. RR-6455. Version 12.

[14] D. Hanson. On the product of the primes. *Canad. Math. Bull.*, 15:33–37, 1972.

[15] J. Harrison. Formalizing an analytic proof of the Prime Number Theorem. *Journal of Automated Reasoning*, 43:243–261, 2009.

[16] J. Harrison and L. Théry. A skeptic's approach to combining HOL and Maple. *J. Automat. Reason.*, 21(3):279–294, 1998.

[17] C. Keller and B. Werner. Importing HOL Light into Coq. In *ITP*, volume 6172 of *LNCS*, pages 307–322. Springer, 2010.

[18] A. Mahboubi and B. Gregoire. Proving equalities in a commutative ring done right in Coq. In *TPHOLs 2005*, volume 3603 of *LNCS*, pages 98–113, Oxford, United Kingdom, Aug. 2005. Springer.

[19] R. H. Risch. The solution of the problem of integration in finite terms. *Bull. Amer. Math. Soc.*, 76:605–608, 1970.

[20] T. Rivoal. *Propriétés diophantiennes de la fonction zêta de Riemann aux entiers impairs.* PhD thesis, Université de Caen, 2001.

[21] B. Salvy. An Algolib-aided version of Apéry's proof of the irrationality of $\zeta(3)$. http://algo.inria.fr/libraries/autocomb/Apery2-html/apery.html, 2003.

[22] A. van der Poorten. A proof that Euler missed: Apéry's proof of the irrationality of $\zeta(3)$. *Math. Intelligencer*, 1(4):195–203, 1979. An informal report.

[23] D. Zeilberger. A holonomic systems approach to special functions identities. *J. Comput. Appl. Math.*, 32(3):321–368, 1990.

[24] D. Zeilberger. The method of creative telescoping. *J. Symbolic Comput.*, 11(3):195–204, 1991.

[25] V. V. Zudilin. One of the numbers $\zeta(5)$, $\zeta(7)$, $\zeta(9)$, $\zeta(11)$ is irrational. *Uspekhi Mat. Nauk*, 56(4(340)):149–150, 2001.