Zeta Types and Tannakian Symbols as a Method for Representing Mathematical Knowledge

Andreas Holmstrom[1,2] ✉ Torstein Vik[2]

(1) Stockholm University, Stockholm, Sweden
(2) Fagerlia vgs, Ålesund, Norway
andreas.holmstrom@gmail.com
torsteinv64@gmail.com

Abstract. We present two closely related notions called (1) a zeta type and (2) a Tannakian symbol. These are data types for representing information about number-theoretic objects, and we argue that a database built out of zeta types and Tannakian symbols could lead to interesting discoveries, similar to what has been achieved for example by the OEIS, the LMFDB, and other existing databases of mathematical objects. We give several examples illustrating what database records would look like, and we describe a tiny prototype database which has already been used to find and automatically prove new theorems about multiplicative functions.

Keywords: Number theory · Multiplicative functions · Tannakian symbols · Zeta types · Zeta functions · L-functions · Automated conjecture-making · Automated theorem proving

1 Introduction

The aim of this paper is to introduce two new data types, to argue for their usefulness with regards to computer representations of objects studied in modern number theory, and finally to propose a design for a database containing such objects.

We choose to focus in this paper on a specific type of mathematical object called a multiplicative function. Multiplicative functions are ubiquitous in number theory - in addition to classical examples like the Euler function or the Liouville function, there is a multiplicative function associated to any object to which one can assign a *zeta function with Euler product.* The objects falling into this category include motives, Galois representations, automorphic representations, schemes, and certain classes of finitely generated groups. We emphasize however that the scope of this project is much bigger, and we refer to the CICM 2016 work-in progress paper [9] for more details, where the first author made an attempt at formulating these ideas in greater generality, focusing among other things on the *lambda-ring structures* which exist on classes of number-theoretic objects.

The algebraic theory underlying our computational ideas are also described in a report by Ane Espeseth and the second author [6] and in more depth in the paper [5] which should appear on arXiv soon. The main point to take away from these papers should be that the design choices that we propose here are guided by specific domain knowledge in number theory and related research fields, and this domain knowledge is crucial if we hope to achieve a database that is useful in practice for conjecture-making and automated reasoning about number-theoretic objects.

1.1 Notation

In this paper, a natural number is the same thing as a strictly positive integer, and we write $\mathbb{N}$ for the set of natural numbers. We also use the standard notation $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{C}$ for the set of integers, the set of rationals, the set of reals, and the set of complex numbers respectively. We write $\mathbb{P}$ for the set of prime numbers, and we use the notation $gcd(a, b)$ for the greatest common divisor of two natural numbers a and b.

2 Zeta Types

We begin by a few definitions, together with several examples of classical multiplicative functions. A general reference for the theory of multiplicative functions is the book of McCarthy [13].

Definition 1. *An* arithmetical function *is a function from $\mathbb{N}$ to $\mathbb{C}$. An arithmetical function f is called a* multiplicative function *if it satisfies the two conditions*

(i) $f(1) = 1$
(ii) $f(mn) = f(m)f(n)$ *for all m, n which are coprime.*

A multiplicative function is called completely multiplicative *if the second condition holds for* all *m, n in $\mathbb{N}$.*

Definition 2. *A* prime power *is an integer of the form p^e, where p is a prime number and e is a natural number.*

The first few prime powers are 2, 3, 4, 5, 7, 8, 9, 11, 13, 16, 17, 19, ...

That a function f is multiplicative implies that all function values can be computed if we know the function values for prime power arguments.

Example 1. The Euler totient function is defined by

$$\varphi(n) = \text{the number of integers } x \text{ such that } 1 \leq x \leq n \text{ and } gcd(x, n) = 1$$

From this explicit definition, we can easily compute values of $\varphi(n)$ for small n. These function values are stored in the Online Encyclopedia of Integer Sequences (OEIS). To be more specific, the first 69 values are stored, and they are:

1, 1, 2, 2, 4, 2, 6, 4, 6, 4, 10, 4, 1, 2, 6, 8, 8, 16, 6, 18, 8, 12, 10, 22, 8, 20, 12, 18, 12, 28, 8, 30, 16, 20, 16, 24, 12, 36, 18, 24, 16, 40, 12, 42, 20, 24, 22, 46, 16, 42, 20, 32, 24, 52, 18, 40, 24, 36, 28, 58, 16, 60, 30, 36, 32, 48, 20, 66, 32, 44

The fact that the Euler function is multiplicative means that many of these values are redundant; in fact, it would be a good idea to store only the numbers $\varphi(p^e)$ for primes p and exponents e. If we set up a table with these numbers (for small values of p and e, we get Table 1.

Table 1. Bell table of the Euler φ function

$\varphi(p^e)$	$e=1$	$e=2$	$e=3$	$e=4$	$e=5$	$e=6$	$e=7$
$p=2$	1	2	4	8	16	32	64
$p=3$	2	6	18	54	162	486	1458
$p=5$	4	20	100	500	2500	12500	62500
$p=7$	6	42	294	2058	14406	100842	705894
$p=11$	10	110	1210	13310	146410	1610510	17715610
$p=13$	12	156	2028	26364	342732	4455516	57921708

This table has 6 rows and 7 columns, and a small combinatorics argument shows that it gives us access to precisely $(7+1)^6 = 262144$ different values of the Euler function. Clearly this is better than having only 69 values, and we have used approximately the same amount of storage space. But an even bigger advantage is that with the new representation, it is easy to spot patterns in the function values. Looking at the list from OEIS, there is no obvious pattern, beyond perhaps the fact that most of the function values are even. In Table 1 on the other hand, it is easy to see that each row is a geometric sequence, with a very simple formula both for the initial term and for the successive quotient!

This example leads us to the definition of our first new data type.

Definition 3. *We define a* zeta type *to be a two-dimensional array of numbers, indexed in one direction by the prime numbers, and in the other direction by the natural numbers. More formally, for any commutative ring R, we define an R-valued zeta type to be a partially defined function from $\mathbb{P} \times \mathbb{N}$ to R.*

If we want to specify how many rows and columns we store in the computer, we may speak of a zeta type *truncated after the prime $p = 13$ and the exponent $e = 7$*, or use some other similar description.

In the present article, our focus will be on examples arising from multiplicative functions, but we emphasize that the concept of a zeta type also has many other applications.

Definition 4. *Let f be a multiplicative function. The* Bell table *of f is the array of function values $f(p^e)$ for prime numbers p and positive integer exponents e.*

Example 2. The Möbius μ function is stored in the OEIS with the following sequence of function values:

1, -1, -1, 0, -1, 1, -1, 0, 0, 1, -1, 0, -1, 1, 1, 0, -1, 0, -1, 0, 1, 1, -1, 0, 0, 1, 0, 0, -1, -1, -1, 0, 1, 1, 1, 0, -1, 1, 1, 0, -1, -1, -1, 0, 0, 1, -1, 0, 0, 0, 1, 0, -1, 0, 1, 0, 1, 1, -1, 0, -1, 1, 0, 0, 1, -1, -1, 0, 1, -1, -1, 0, -1, 1, 0, 0, 1, -1

It is very hard to spot any patterns here. But look at Table 2, where the same function is represented by its Bell table.

Table 2. Bell table of the Möbius μ function

$\mu(p^e)$	$e = 1$	$e = 2$	$e = 3$	$e = 4$	$e = 5$	$e = 6$	$e = 7$
$p = 2$	-1	0	0	0	0	0	0
$p = 3$	-1	0	0	0	0	0	0
$p = 5$	-1	0	0	0	0	0	0
$p = 7$	-1	0	0	0	0	0	0
$p = 11$	-1	0	0	0	0	0	0
$p = 13$	-1	0	0	0	0	0	0

We hope that these two examples show the advantages of the zeta type representation over the more traditional "list of function values" representation.

One may ask how many rows and columns should be included in a finite display of a zeta type. This will vary, depending on the situation and the intended applications. As we will see in the next section, the rows of a zeta type typically satisfy some simple recursion formula, and the number of columns displayed should in such cases be large enough to detect what the recursion formula is.

3 Tannakian Symbols

In the previous section we compared the representation of a multiplicative function as a list of function values with its representation as a zeta type. The aim of this section is to construct yet another representation, which is (for many purposes) even better than the zeta type representation.

The construction does not work for all multiplicative functions, only for those which satisfy a property we call rationality. The good news is that the vast majority of all multiplicative functions appearing in the number theory literature satisfy this property.

Definition 5. *Let f be a multiplicative function, and let p be a prime number. The* Bell series of f at p *(see McCarthy [13], p. 60) is the formal power series*

$$f_p(t) = 1 + f(p)\ t + f(p^2)\ t^2 + f(p^3)\ t^3 + \dots$$

Here t is a formal variable, and the series is an element of the formal power series ring $\mathbb{C}[[t]]$.

Example 3. Looking at the Bell table of the Euler φ function above, we see that the Bell series of φ at the prime number 5 is equal to

$$1 + 4t + 20t^2 + 100t^3 + 500t^4 + 2500t^5 + \dots$$

Definition 6. *A power series is called* rational *if it can be expressed as a quotient $v(t)/u(t)$, where both $u(t)$ and $v(t)$ are polynomials.*

Definition 7. *We say that a multiplicative function is* rational *if for every prime number p, the Bell series $f_p(t)$ is a rational power series.*

Equivalently (see Stanley [17, Theorem 4.1.1]), a multiplicative function is rational if each row in its Bell table is a linearly recursive sequence (where we allow any complex coefficients in the linear recursion relation).

Given a rational multiplicative function f and a prime number p, we want to explain what we mean by the Tannakian symbol of f at the prime p. First of all, recall that the *reciprocal* of a non-zero complex number z is by definition the number $\frac{1}{z}$. Let $u(t)$ and $v(t)$ be two polynomials with no complex roots in common, such that

$$f_p(t) = \frac{v(t)}{u(t)}$$

Such polynomials exist because f is rational, and they are uniquely determined up to a constant factor. Let $\alpha_1, \alpha_2, \dots, \alpha_m$ be the list of all reciprocals of the complex roots of $u(t)$, listed with multiplicity. Similarly, let $\beta_1, \dots, \beta_n$ be the reciprocals of the complex roots of $v(t)$, listed with multiplicity. This definition means precisely that the Bell series of f at the prime p can be expressed by the formula

$$f_p(t) = \frac{\prod(1 - \beta_j t)}{\prod(1 - \alpha_i t)} \tag{1}$$

Definition 8. *Let f be a rational multiplicative function and let p be a prime. We define the Tannakian symbol of f at p to be the formal symbol*

$$\frac{\{\alpha_1, \alpha_2, \dots, \alpha_m\}}{\{\beta_1, \dots, \beta_n\}}$$

where the numbers α_i and β_j are taken from the right hand side of Equation 1.

The elements α_i are referred to as the upstairs elements, and the elements β_j are referred to as the downstairs elements. The object $\{\alpha_1, \alpha_2, \dots, \alpha_m\}$ (and also the object $\{\beta_1, \dots, \beta_n\}$) is a multiset of complex numbers (so repeated elements are allowed, and the order of the elements is irrelevant). For the reader who wants a precise mathematical explanation of what type of object a Tannakian symbol is, we can formulate this as follows.

Definition 9. *A Tannakian symbol is an ordered pair of finite disjoint multisets of complex numbers, or (equivalently) a function with finite support from $\mathbb{C}$ to $\mathbb{Z}$.*

To illustrate all this with an example, let us compute the Tannakian symbol of the Möbius function. Every row in the Bell table is the same, so in this example the Tannakian symbol is independent of the chosen prime. The Bell series is

$$1 - 1 \cdot t + 0 \cdot t^2 + 0 \cdot t^3 + 0 \cdot t^4 + \ldots = 1 - t = \frac{1-t}{1}$$

We read off the coefficients of t in the numerator (ignoring the minus sign), and the only such coefficient is 1. In the denominator there are no coefficients at all, so we get an empty set. The Tannakian symbol of the Möbius function therefore equals

$$\frac{\emptyset}{\{1\}}$$

As a second example, we compute the Tannakian symbol of the Euler function at the prime $p = 5$, using the method of adding and subtracting identical terms in order to split the sum into two geometric series.

$$1 + 4t + 20t^2 + 100t^3 + 500t^4 + 2500t^5 + \ldots =$$
$$= (1 + 5t + 25t^2 + 125t^3 + \ldots) - (t + 5t^2 + 25t^3 + \ldots) =$$
$$= \frac{1}{1-5t} - \frac{t}{1-5t} = \frac{1-t}{1-5t}$$

Now we can read off the Tannakian symbol; it is

$$\frac{\{5\}}{\{1\}}$$

Automation of this procedure (which we have implemented in Sage [16]) relies on the Berlekamp-Massey algorithm (see [3]), which checks a sequence for any possible linear recursion relation and, if it finds such a relation, produces a rational expression for the generating function.

If we are in a situation where we know a formula for function values at prime powers, we don't have to rely on numerical procedures such as Berlekamp-Massey, but we can do an exact symbolic calculation to get a formula for the Tannakian symbol at any prime. For the Euler function, it is well-known that $\varphi(p^e) = p^e - p^{e-1}$, and using this we get

$$f_p(t) = 1 + \sum_{e=1}^{\infty} (p^e - p^{e-1}) t^e = \sum_{e=0}^{\infty} p^e t^e - t \cdot \sum_{e=0}^{\infty} p^e t^e =$$
$$= (1-t) \cdot \sum_{e=0}^{\infty} p^e t^e = \frac{1-t}{1-pt}$$

which shows that the Tannakian symbol of the Euler φ function at any prime p is

$$\frac{\{p\}}{\{1\}}$$

Similar calculations for other functions reveal that all functions appearing in classical/elementary number theory have very simple expressions for their symbols.

Example 4. The Liouville function has Tannakian symbol $\frac{\{-1\}}{\emptyset}$ at all primes.

Example 5. The τ function (which gives the number of positive divisors of a number n) has Tannakian symbol $\frac{\{1,1\}}{\emptyset}$ at all primes.

Example 6. The σ function (which gives the sum of all positive divisors of a number) has Tannakian symbol $\frac{\{p,1\}}{\emptyset}$ at all primes. More generally, the k'th divisor function σ_k, which gives the sum of k'th powers of all positive divisors, has Tannakian symbol $\frac{\{p^k,1\}}{\emptyset}$

Representing a multiplicative function by its Tannakian symbol has two major advantages over the previous two representations. First of all, whenever we can find a closed expression for the symbol as a function of p, we have a *finite* representation which captures *all* function values of the function. Secondly, there is a calculus of Tannakian symbols which means we can easily compute the symbol of a Dirichlet convolution or a product of two multiplicative functions with known symbols. In fact, almost all unary and binary operations on multiplicative functions which appear in the literature have an explicit counterpart in the language of Tannakian symbols; this is the subject of our article in progress with Ane Espeseth [5].

In addition, properties of many mathematical objects can be read directly off their Tannakian symbols. In the case of multiplicative functions, we have for instance:

1. Completely multiplicative (as mentioned in Definition 1) corresponds to the Tannakian symbol having no elements downstairs, and at most one element upstairs, the typical case being $\frac{\{a\}}{\varnothing}$ for some a.
2. Similarly, "specially multiplicative" corresponds to the Tannakian symbol having no elements downstairs and at most *two* elements upstairs, for example being of the form $\frac{\{a,b\}}{\varnothing}$ for some a and b.
3. Finally, a multiplicative function being a "totient" is equivalent to its Tannakian symbol having at most one element upstairs and at most one element downstairs.

4 A Tiny Prototype Database

We want to suggest that a database of zeta types and Tannakian symbols could be a useful tool for researchers in number theory, and if designed in a good way, such a database may even be useful in certain AI-inspired mathematical endeavours, such as automated conjecturing and automated theorem discovery/proving.

We have implemented a very small prototype database of multiplicative functions to illustrate what we have in mind. The database contains approximately

20 classical multiplicative functions, and in addition 5 infinite families of multiplicative function (like the family σ_k of divisor functions, indexed by a positive integer k).

The entire database is contained in a single `json` file, which is displayed below. The reader will recognize the Tannakian symbols for the Möbius function, the Euler φ function, and the other examples from the previous section.


```
{
    "format": "Edinburgh 1",
    "functions": [
        {
            "name":       "zero",
            "symbol":     "∅/∅",
            "latex":      "\\epsilon"
        }, {
            "name":       "one",
            "symbol":     "{1}/∅",
            "latex":      "1",
            "latex_eval": "1"
        }, {
            "name":       "mu",
            "symbol":     "∅/{1}",
            "latex":      "\\mu"
        }, {
            "name":       "id",
            "symbol":     "{p}/∅",
            "latex":      "Id",
            "latex_eval": "{-}"
        }, {
            "name":       "id_2",
            "symbol":     "{p^2}/∅",
            "latex":      "Id_2",
            "latex_eval": "{\\left(-\\right)}^2"
        }, {
            "name":       "id_3",
            "symbol":     "{p^3}/∅",
            "latex":      "Id_3",
            "latex_eval": "{\\left(-\\right)}^3"
        }, {
            "name":       "tau",
            "symbol":     "{1, 1}/∅",
            "latex":      "\\tau"
        }, {
            "name":       "tau_3",
            "symbol":     "{1, 1, 1}/∅",
            "latex":      "\\tau_3"
        }, {
            "name":       "tau_4",
            "symbol":     "{1, 1, 1, 1}/∅",
            "latex":      "\\tau_4"
        }, {
            "name":       "sigma",
            "symbol":     "{1, p}/∅",
            "latex":      "\\sigma"
        }, {
            "name":       "euler_phi",
            "symbol":     "{p}/{1}",
            "latex":      "\\varphi"
        }, {
            "name":       "liouville",
            "symbol":     "{-1}/∅",
            "latex":      "\\lambda"
        }, {
            "name":       "gamma",
            "symbol":     "{1}/{2}",
            "latex":      "\\gamma"
        }, {
            "name":       "dedekind_psi",
            "symbol":     "{p}/{-1}",
            "latex":      "\\psi"
        }, {
            "name":       "theta",
            "symbol":     "{1}/{-1}",
            "latex":      "\\theta"
        }, {
            "name":       "core",
            "symbol":     "{1}/{1-p}",
            "latex":      "\\text{Core}_1"
        }, {
            "name":       "beta",
            "symbol":     "{p,-1}/∅",
            "latex":      "\\beta"
        }, {
            "name":       "char_fn_squares",
            "symbol":     "{1, -1}/∅",
            "latex":      "\\epsilon_2"
        }, {
            "name":       "char_fn_squarefree",
            "symbol":     "∅/{-1}",
            "latex":      "\\xi"
        }
    ],
    "function-families": [
        {
            "indexed_by": ["k"],
            "name":       "id_$k$",
            "symbol":     "{p^$k$}/∅",
            "latex":      "Id_{$k$}",
            "latex_eval": "{\\left(-\\right)}^{$k$}"
        }, {
            "indexed_by": ["k"],
            "name":       "sigma_$k$",
            "symbol":     "{p^$k$, 1}/∅",
            "latex":      "\\sigma_{$k$}"
        }, {
            "indexed_by": ["k"],
            "name":       "jordan_$k$",
            "symbol":     "{p^$k$}/{1}",
            "latex":      "\\jordan_{$k$}"
        }, {
            "indexed_by": ["k"],
            "name":       "psi_$k$",
            "symbol":     "{p^$k$}/{-1}",
            "latex":      "\\psi_{$k$}"
        }, {
            "indexed_by": ["k"],
            "name":       "tau_$k$",
            "symbol":     "|$k$ * [1]|",
            "latex":      "\\tau_{$k$}"
        }

    ]
}
```


This file and related code are available at the second author's GitHub repository [18]. For specification of the format, we also refer to this repository, more specifically to "Edinburgh 1.md" in the Classical-Multiplicative-Functions folder.

5 An Application to Automated Theorem Generation

5.1 Identities between Multiplicative Functions

There are many theorems in the number theory literature about how values of different multiplicative functions are related to each other. One of the simplest examples is a well-known statement about the Euler φ function.

The point of this example and the next is to illustrate what typical identities between multiplicative functions look like, and also to illustrate how Tannakian symbols work in actual proofs. For more details on exactly how and why these techniques work, we have to refer to our other papers with Espeseth ([5] and [6]).

Example 7. Let's look again at a list of function values $\varphi(n)$ for small values of n.

n	1	2	3	4	5	6	7	8	9	10	11	12	...	20
$\varphi(n)$	1	1	2	2	4	2	6	4	6	4	10	4	...	8

We said earlier that it is difficult to spot patterns in these values, but this doesn't mean no patterns exist. In fact, one particular pattern can be found by choosing a number n (let us choose the number 6), and looking at all the function values of *divisors* of 6. The divisors are 1, 2, 3 and 6, and the corresponding function values (circled below) are 1, 1, 2, and 2. Now take the sum of the function values. We get the number 6!

n	**1**	**2**	**3**	4	5	**6**	7	8	9	10	11	12	...	20
$\varphi(n)$	1	1	2	2	4	2	6	4	6	4	10	4	...	8

Doing the same again, but with the number 20, we get the divisors 1, 2, 4, 5, 10 and 20, and the corresponding function values are 1, 1, 2, 4, 4 and 8.

n	**1**	**2**	3	**4**	**5**	6	7	8	9	**10**	11	12	...	**20**
$\varphi(n)$	1	1	2	2	4	2	6	4	6	4	10	4	...	8

Their sum happens to be precisely 20!

We are led to believe that there is a general law here, which says that the sum of the values of the Euler function over the divisors of a given integer is precisely that given integer. This statement can be rewritten as an identity:

$$\sum_{d|n} \varphi(d) = n \tag{2}$$

There are several ways of proving this identity, but we want to give a proof sketch that illustrates the use of Tannakian symbols. We first note that φ is a multiplicative function whose Tannakian symbol is $\frac{\{p\}}{\{1\}}$, and that the identity function is also multiplicative, with Tannakian symbol $\frac{\{p\}}{\emptyset}$. The right hand side is of course the value of the identity function applied to n. The left hand side is the Dirichlet convolution of φ with the constant function given by $f(n) = 1$, and this constant function has Tannakian symbol $\frac{\{1\}}{\emptyset}$. The identity (2) is therefore equivalent to the statement

$$\frac{\{p\}}{\{1\}} \oplus \frac{\{1\}}{\emptyset} = \frac{\{p\}}{\emptyset} \tag{3}$$

which is obviously true by the general rules for taking Dirichlet convolution of two Tannakian symbols. Hence the original identity is proved.

Example 8. Let us look at a more complicated example. Recall that the τ function counts the number of positive divisors of a positive integer n. For example, the number 10 has four positive divisors (1, 2, 5 and 10), so $\tau(10) = 4$. The table of function values looks like this:

n	1	2	3	4	5	6	7	8	9	10	11	12	..	16	..	25
$\tau(n)$	1	2	2	3	2	4	2	4	3	4	2	6	..	5	..	3

Now take any integer - for example the number 4. Square the number - in our case we get 16. Now let's play a game with the divisors of 16 (which are 1, 2, 4, 8 and 16). Consider the alternating sum

$$\tau(1) \cdot \tau(16) - \tau(2) \cdot \tau(8) + \tau(4) \cdot \tau(4) - \tau(8) \cdot \tau(2) + \tau(16) \cdot \tau(1)$$

If we plug in the values of the τ function here, we see that the sum is equal to $\tau(4)$.

Let's try again. We pick the number 5, and compute

$$\tau(1) \cdot \tau(25) - \tau(5) \cdot \tau(5) + \tau(25) \cdot \tau(1)$$

Now this sum is equal to $\tau(5)$! The pattern we see here can be generalized to any n, providing we take care in the placement of plus and minus signs in the sum. If we denote by $\Omega(n)$ the number of prime factors of n counted with multiplicity (so that $\Omega(4) = \Omega(6) = 2$ and $\Omega(8) = 3$), then the general identity is

$$\sum_{d|n^2} (-1)^{\Omega(d)} \tau(d) \tau(\frac{n^2}{d}) = \tau(n) \tag{4}$$

Again, let us provide a proof sketch. The τ function is multiplicative, and its Tannakian symbol is $\frac{\{1,1\}}{\emptyset}$. The left hand side is what Redmond and Sivaramakrishnan [15] call the *norm operator* applied to τ, and in our new language, this is an example of an *Adams operation* in a lambda-ring structure on multiplicative functions, and for such Adams operations, we have explicit formulas (in the world of Tannakian symbols). This particular Adams operation acts by squaring all elements of the symbol, so the left hand side of the identity is $\frac{\{1^2,1^2\}}{\emptyset}$ while the right hand side is $\frac{\{1,1\}}{\emptyset}$. Now the identity follows immediately from the elementary fact that $1 \cdot 1 = 1$.

5.2 Automatic Generation of New Identities with Proofs

What we can do using our miniature database is to search for more identities like this. This procedure relies on the algebraic theory of Tannakian symbols, which is illustrated by the examples above, and covered in detail in [5]. The methodology is somewhat similar to Zeilberger's classical work [19] on identities between special functions. Whereas Zeilberger exploits the fact that special functions have finite representations via the theory of holonomic power series, we exploit the fact that many multiplicative functions have finite representations via the theory of Tannakian symbols.

The methodology we follow is:

1. Represent each function in the database by its Tannakian symbol.
2. Generate a list of Tannakian symbol expressions by applying various unary and binary operations to combinations of symbols from the database.
3. Partition the list of expressions using the equivalence relation given by equality after simplification.
4. Any two expressions in the same equivalence class gives an identity between multiplicative functions. This gives a list of new theorems.
5. Discard identities that are trivial, for example of the form $X + Y = Y + X$.

Using this procedure, we find thousands of new identities, many of which seem interesting enough to be publishable in a decent number theory journal. We present a small screenshot of a simplified version of our Sage interface, to give a feeling for what the code and the output looks like.

In [73]:
```
funcs = [phi, id, id_2, tau, sigma, liouville, mu]

sums = [l + r for l, r in itertools.combinations_with_replacement(funcs, 2)]

for lhs in sums:
    for rhs in funcs:
        if lhs == rhs:
            print html(lhs & rhs)
```

$$\sum_{d_0|n} \varphi(d_0) \cdot \tau\left(\frac{n}{d_0}\right) = \sigma(n)$$

$$\sum_{d_0|n} d_0 \cdot \mu\left(\frac{n}{d_0}\right) = \varphi(n)$$

$$\sum_{d_0|n} \sigma(d_0) \cdot \mu\left(\frac{n}{d_0}\right) = n$$

The reason that these identities (which in many cases were not known before) are theorems and not conjectures is that we have a closed formula for the relevant Tannakian symbols as functions of the prime p. This means that when two expressions evaluate to the same thing (as Tannakian symbols), then all function values of the left and the right hand side of the identity must also be equal.

5.3 A Brief Introduction to Other Operations

With our formalism of Tannakian symbols and zeta types, we can analyze all operations on multiplicative functions that we have been able to find in the number theory literature. There is no space to develop this theory in any depth here, but we can give a brief survey of what the operations are, with references to definitions.

There are four fundamental binary operations on the set of all multiplicative functions. These are:

1. Dirichlet convolution.
2. Tensor product.
3. Unitary convolution.
4. Ordinary product.

The last operation is just the ordinary and elementary operation given by multiplying complex-valued functions. The first and third are defined in the Encyclopedia of Mathematics, in the entry on Dirichlet convolution [4], while the second is defined for general multiplicative functions in [5], and for Tannakian symbols (and hence for all rational multiplicative functions) in [6].

There are four fundamental unary operations, namely:

1. The norm operator of Redmond and Sivaramakrishnan, and various generalizations.
2. Precomposition with a power function.
3. The k'th convolute operator.
4. A certain nameless operation related to base change of varieties over finite fields.

To convey the flavour of what these operations are, let us include here just the definitions of Dirichlet convolution and the norm operator, both of which appeared in the proof sketches above.

Definition 10. *Let f and g be two multiplicative functions. The* Dirichlet convolution *of f and g is the function h defined by the formula*

$$h(n) = \sum_{d|n} f(d)g(n/d)$$

where the sum on the right hand side is taken over all positive divisors d of n.

Definition 11. *The norm of a multiplicative function f is the function g defined by*

$$g(n) = \sum_{d|n^2} (-1)^{\Omega(d)} f(n^2/d) f(d)$$

where $\Omega(d)$ is the number of prime factors of d counted with multiplicity. Here the sum is taken over all positive divisors of n^2.

6 Conclusion

We have argued that zeta types and Tannakian symbols are useful tools for a better understanding of multiplicative functions, and we have implemented a small program for automated theorem discovery, which generates identities between such functions. Many of the identities seem to be new and interesting results.

For future work, we propose setting up a much larger database of zeta types and Tannakian symbols, which would make it possible to experiment with automated conjecturing and other computational procedures in even more interesting settings. As a motivating example, we display the zeta type associated to an L-function of one elliptic curve.

	$e = 1$	$e = 2$	$e = 3$	$e = 4$	$e = 5$
$p = 2$	1	-1	-3	-1	5
$p = 3$	0	-3	0	9	0
$p = 5$	-3	4	3	-29	72
$p = 7$	-1	-6	13	29	-120
$p = 11$	-1	1	-1	1	-1

Understanding L-functions like this one is among the greatest challenges of modern number theory. Recent years have seen several major advances in faster algorithms for computing L-functions (i.e. computing the numbers appearing in zeta types like the one above), and the vast amount of data generated this way offers many intriguing challenges to researchers willing to combine a deep mathematical understanding of the objects involved with large-scale computations of various kinds. We refer to the recent work of Brown and Schnetz [1], of Costa and Tschinkel [2], of Harvey [7], [8] and of Kedlaya [10] for more details on current research in this field.

When it comes to database content, there would be significant overlap between our database and the already established L-functions and Modular Forms Database (LMFDB) [12], and some limited overlap with the Online Encyclopedia of Integer Sequences (OEIS) [14] but our database would represent the objects in a different way, and it would also contain many objects not directly related to what is currently in the LMFDB or the OEIS.

We have preliminary work in place on `json` specifications for various types of mathematical objects represented by zeta types and Tannakian symbols. However, we believe that the next step in our process should not be to set up a huge database and fill it with lots and lots of data, but to choose a specific research project and set up a medium-sized database to support this project. Gaining experience this way will lead to better design choices if we at a later stage choose to go ahead with a more ambitious large-scale project.

The projects we have in mind as such a possible first step include:

- Find Tannakian symbol representations for all (or almost all) of the multiplicative functions in the OEIS (currently there is a total of 1459 such sequences) and search for new identities between them.
- Build a database of zeta types associated to L-functions of hypergeometric motives and check various standard conjectures on these objects.
- Try out the recent Sage package of Larson and Van Cleemput [11] which implements Dalmatian heuristics for automated conjecture-making. This is a domain-independent package for producing conjectures about inequalities between combinations of real-valued invariants, and could easily be set up to work on a database of zeta types or Tannakian symbols.
- Set up a database of Tannakian symbol representations for Dirichlet characters, and do computations aiming to elucidate the lambda-ring structure of the lambda-ring generated by all Dirichlet characters.
- Set up a database of Tannakian symbols associated to representations of finite groups, and use this database in a project on the lambda-ring structure of representation rings.

Acknowledgements. Aspects of this work has been presented in various talks, including at CICM 2016, at AITP 2017 (Conference on Artificial Intelligence and Theorem Proving), at the Representation Theory 2016 Conference in Uppsala, and at seminars in Stockholm and in Leicester. We would like to acknowledge the encouragement and many helpful comments received on these occasions, and in particular we thank Michael Kohlhase, Volodymyr Mazorchuk, Frank Neumann, Florian Rabe, Andreas Strömbergsson, and Josef Urban.

(Slides from some of these talks are available on the first author's webpage; `andreasholmstrom.org`. Together with the other documents mentioned, they may serve as a complement to this paper.)

Finally, we also want to thank Magnus Hellebust Haaland and Olav Hellebust Haaland for implementing the Berlekamp-Massey algorithm for us in Sage; many of the automated procedures for computing with Tannakian symbols rely on this implementation.

References

1. Brown, F., Schnetz, O.: Modular forms in quantum field theory. Communications in Number Theory and Physics 7(2), 293–325 (2013)
2. Costa, E., Tschinkel, Y.: Variation of Neron-Severi ranks of reductions of K3 surfaces. Experimental Mathematics 23, pp. 475–481 (2014)
3. Encyclopedia of Mathematics: *Berlekamp-Massey algorithm.* `https://www.encyclopediaofmath.org/index.php/Berlekamp-Massey_algorithm`
4. Encyclopedia of Mathematics: *Dirichlet convolution.* `https://www.encyclopediaofmath.org/index.php/Dirichlet_convolution`
5. Espeseth, A., Holmstrom, A., Vik, T.: New perspectives on multiplicative functions: Lambda-rings and Tannakian symbols. Preprint in preparation (draft version available upon request, but see also [6]).

6. Espeseth, A., Vik, T.: Motivic Symbols and Classical Multiplicative Functions. Research report submitted to the Norwegian Contest for Young Scientists 2016. Available at `http://andreasholmstrom.org/wp-content/2016/05/motivic-symbols-classical-final-version.pdf`
7. Harvey, D.: Counting points on hyperelliptic curves in average polynomial time. Ann. of Math. (2) 179(2), 783–803 (2014)
8. Harvey, D.: Computing zeta functions of arithmetic schemes. Proc. Lond. Math. Soc. 111, no. 6, 1379–1401 (2015)
9. Holmstrom, A.: Towards automated conjecture-making in higher arithmetic geometry. CEUR Workshop Proceedings, Volume 1785, pp. 204–218 (2016)
10. Kedlaya, K.S.: Computing Zeta Functions via p-adic Cohomology. In: Buell (ed.); Algorithmic Number Theory. LNCS, vol. 3076, pp. 1–17. Springer Berlin Heidelberg (2004)
11. Larson, C. E., Van Cleemput, N.: Automated Conjecturing I: Fajtlowicz's Dalmatian Heuristic Revisited. Artificial Intelligence 231, 17–38 (2016)
12. The LMFDB Collaboration: The L-functions and Modular Forms Database. `http://www.lmfdb.org`
13. McCarthy, P. J.: *Introduction to Arithmetical Functions.* Springer-Verlag, 1986.
14. Online Encyclopedia of Integer Sequences, `http://oeis.org/`
15. Redmond, D., Sivaramakrishnan, R.: Some properties of specially multiplicative functions. Journal of Number Theory, Vol. 13 (2), pp. 210-227 (1981)
16. The Sage Developers: *SageMath, the Sage Mathematics Software System (Version 7.6)*, 2017. `http://www.sagemath.org`
17. Stanley, R.: *Enumerative Combinatorics: Volume 1 (2nd edition)*. Cambridge University Press, 2014.
18. Vik, T., Classical multiplicative functions. GitHub repository (2017). `https://github.com/torstein-vik/zeta-types-smc/blob/master/Classical-Multiplicative-Functions/`
19. Zeilberger, D.: A holonomic systems approach to special functions identities. J. Comput. Appl. Math. 32(3), 321–368 (1990)