DERIVATION RELATION AND DUALITY FOR MULTIPLE ZETA VALUES

STUDENTS: VIKTOR BALCH BARTH, WILLIAM HORNSLIEN.
ADVISOR: KURUSCH EBRAHIMI-FARD

Abstract. We present a new set of multiple zeta value (MZV) relations together with a simple identity connecting the harmonic product to a set of derivation relations. We also describe a self-made Python Library for MZV computations.

0. Summary

The central topic of this research project are relations among *multiple zeta values*. Our main results are Theorem 13 in Section 5 and Theorem 14 in Section 6. Theorem 13 connects the operator ∂_n to the harmonic (a.k.a. quasi-shuffle) product. This is done using results from Tanaka and Bachmann's rooted tree maps [1, 10]. Theorem 14 involves the dual product, defined by Ebrahimi-Fard et al. in reference [4]. Our theorem states that for any two words, the difference of their harmonic product and their dual product lies always in the kernel of the evaluation map. This gives a set of relations similar to the double shuffle relations of MZVs.

This note documents our work[1] on multiple zeta values (MZVs). It's intended to be self-contained; hopefully readable to the interested non-expert. We have been relying in particular on the paper by Ihara, Kaneko and Zagier [8] and on papers by Tanaka and Bachmann [1, 2, 10], to give a background on the subject.

Date: March 10, 2019.

[1]We have explored the theory of MZVs as part of a StudForsk-project ("Derivation relations and duality for multiple zeta values"), an undergraduate research project programme at the Norwegian University of Science and Technology – NTNU. A huge thanks goes to our supervisor, Kurusch Ebrahimi-Fard, for pointing the way, asking interesting questions, and being excited about our work.

1. Introduction

Multiple zeta values (MZVs) are an active research topic in mathematics. A systematic and in-depth study of the mathematics underlying MZVs only started in the early 1990s with the works of Hoffman [5] and Zagier [11]. Although MZVs can be traced back to Euler's work in the 18th century. Studying the theory of MZVs involves several areas of modern mathematics, including number theory, algebra, combinatorics, arithmetic and algebraic geometry, and Lie group theory. Moreover, MZVs and their siblings, multiple polylogarithms, show deep connections to high-energy physics.

MZVs are nested sums of *depth* $n \in \mathbb{N}$ and *weight* $k = k_1 + k_2 + \cdots + k_n$, where all of k_i are positive integers, and we also require $k_1 \geq 2$.

$$\zeta(k_1, \ldots, k_n) := \sum_{m_1 > \cdots > m_n > 0} \frac{1}{m_1^{k_1} \cdots m_n^{k_n}}. \tag{1}$$

They also have an integral representation, due to Drinfel'd [3] and Zagier [11], as follows.

$$\zeta(k_1, \ldots, k_n) := \int \cdots \int_{1 > t_1 > t_2 > \cdots > t_k > 0} \omega_1(t_1)\omega_2(t_2) \cdots \omega_k(t_k), \tag{2}$$

where $\omega_i(t_i) = dt_i/(1 - t_i)$ if $i \in \{k_1,\ k_1 + k_2,\ \ldots,\ k_1 + k_2 + \cdots + k_n\}$, and $\omega_i(t_i) = dt_i/t_i$ otherwise.

Our work aims at studying the $\mathbb{Q}$-linear relations between MZVs, that is, how non-trivial linear combinations of MZVs with rational coefficients equal zero. The earliest algebraic relations among MZVs can be traced back to the work of Euler, where one can find the simple identity

$$\zeta(2, 1) - \zeta(3) = 0.$$

One observes quickly that one class of such relations is obtained when multiplying two such series. Indeed, it is easy to see that the product of two MZVs in the sum representation (1) evaluates into a linear combination of MZVs due to the product rule for iterated sums. However, one also shows quickly that the product of two MZVs represented by iterated integrals (2) gives rise to another linear combination of MZVs, thanks to integration by parts. The following simple example may elucidate this:

$$\begin{aligned}
\zeta(2)\zeta(2) &= \sum_{n>0} \frac{1}{n^2} \sum_{m>0} \frac{1}{m^2} \\
&= \sum_{m>n>0} \frac{1}{n^2 m^2} + \sum_{n>m>0} \frac{1}{n^2 m^2} + \sum_{n>0} \frac{1}{n^4} \\
&= 2\zeta(2, 2) + \zeta(4) \\
&= \int_0^1 \frac{dt}{t} \int_0^t \frac{ds}{1-s} \int_0^1 \frac{dt}{t} \int_0^t \frac{ds}{1-s} \\
&= 4\int_0^1 \frac{dt}{t} \int_0^t \frac{ds}{s} \int_0^s \frac{du}{1-u} \int_0^u \frac{dv}{1-v} + 2\int_0^1 \frac{dt}{t} \int_0^t \frac{ds}{1-s} \int_0^s \frac{du}{u} \int_0^u \frac{dv}{1-v} \\
&= 4\zeta(3, 1) + 2\zeta(2, 2).
\end{aligned}$$

As a result the relation

$$4\zeta(3, 1) - \zeta(4) = 0$$

is obtained. One of the open problems (roughly speaking, the conjectural algebraic independence of multiple zeta values over the rationals, modulo the set of double-shuffle relation and regularization) is still far from being solved.

A variety of tools have been developed to study these relations, some of which will be introduced in Sections 2 through 4. Our original results are presented in Sections 5 and 6. Our less fruitful attempts are described in 7. In appendix A, we document a self-made python library that facilitates a multitude of MZV-related calculations.

2. Word algebras, duality and products

We consider the polynomial algebra in two non-commutative variables x, y, which will be denoted by $\mathfrak{H} := \mathbb{Q}\langle x, y\rangle$. The subalgebra $\mathfrak{H}^1 := \mathbb{Q} + \mathfrak{H}y$ is linearly spanned by words $z_{k_1} \cdots z_{k_n}$ in the blocks $z_k := x^{k-1}y$, for $k \in \mathbb{N}$ (throughout the text, $\mathbb{N} = \{1, 2, 3, \ldots\}$). The subalgebra $\mathfrak{H}^0 = \mathbb{Q} + x\mathfrak{H}y$ consists of so-called *admissible words* spanned by the subset of words $z_{k_1} \cdots z_{k_n}$ with $k_i \in \mathbb{N}$, and in particular $k_1 \geq 2$. Notice that these are the exact same requirements as we had for the k_i in equation (1). In fact, we define a $\mathbb{Q}$-linear map $Z\colon \mathfrak{H}^0 \to \mathbb{R}$ sending admissible words $z_{k_1} \cdots z_{k_n} = x^{k_1-1}y \cdots x^{k_n-1}y \in \mathfrak{H}^0$ to MZVs by

$$Z(x^{k_1-1}y \cdots x^{k_n-1}y) := \zeta(k_1, \ldots, k_n).$$

Thanks to the coexistence of sum and integral representations of MZVs, the $\mathbb{Q}$-algebra spanned by those values contains many linear relations, called *double shuffle relations*. The latter are most naturally described in terms of the algebraic framework of formal word algebras equipped with shuffle and harmonic products. The harmonic product is sometimes called quasi-shuffle or stuffle product in the literature.

2.1. Shuffle and harmonic products.

The *harmonic product* $m_*\colon \mathfrak{H}^1 \otimes \mathfrak{H}^1 \to \mathfrak{H}^1$ is iteratively defined:

(1) $\mathbf{1} * w := w * \mathbf{1} := w$

(2) $z_n u * z_m v := z_n(u * z_m v) + z_m(z_n u * v) + z_{n+m}(u * v)$

for words $u, v, w \in \mathfrak{H}^1$ and $n, m \in \mathbb{N}$. The empty word is denoted $\mathbf{1} \in \mathfrak{H}^1$. Further the *shuffle product* $m_{\sqcup\!\sqcup}\colon \mathfrak{H} \otimes \mathfrak{H} \to \mathfrak{H}$ is given by

(1) $\mathbf{1} \sqcup\!\sqcup w := w \sqcup\!\sqcup \mathbf{1} := w$

(2) $au \sqcup\!\sqcup bv := a(u \sqcup\!\sqcup bv) + b(au \sqcup\!\sqcup v)$

for words $u, v, w \in \mathfrak{H}$ and letters $a, b \in \{x, y\}$. Notice that the shuffle product preserves $\mathfrak{H}^1$, i.e. $m_{\sqcup\!\sqcup}\colon \mathfrak{H}^1 \otimes \mathfrak{H}^1 \to \mathfrak{H}^1$. The difference between both products is seen best via an example, comparing the harmonic product $z_2 * z_2 = 2z_2z_2 + z_4$ with the shuffle product $xy \sqcup\!\sqcup xy = 2xyxy + 4xxyy$. Later on in this paper the notation $\mathcal{H}_w$ will be used for an operator on words in $\mathfrak{H}^1$, defining harmonic shuffle by w, that is, $\mathcal{H}_w(v) := v * w$. For example

$$\mathcal{H}_{z_2}(z_3) = z_3 * z_2.$$

Theorem 1 (Theorem 4 and 5 in [12])**.** *The map $Z\colon \mathfrak{H}^0 \to \mathbb{R}$ is an algebra morphism with respect to both algebras $(\mathfrak{H}^0, m_*)$ and $(\mathfrak{H}^0, m_{\sqcup\!\sqcup})$. Moreover, the following so-called* extended double shuffle relations *hold*

$$Z(u * v - u \sqcup\!\sqcup v) = 0 \qquad \text{and} \qquad Z(z_1 * w - z_1 \sqcup\!\sqcup w) = 0, \tag{3}$$

for any words $u, v, w \in \mathfrak{H}^0$.

In fact, conjecturally all linear relations among MZVs are assumed to follow from the relations in equation (3). See reference [8] for more details.

Example 2. *Let $u = v = z_2 = xy$. Then*

$$Z(u * v - u \sqcup v) = Z(z_2 * z_2 - xy \sqcup xy) = Z(2z_2z_2 + z_4 - 2z_2z_2 + 4z_3z_1) = Z(z_4 - 4z_3z_1) = 0.$$

This gives us the MZV relation $\zeta(4) = 4\zeta(3,1)$.

Example 3. *Let $w = z_2$. Then*

$$Z(z_1 * w - z_1 \sqcup w) = Z(z_1 * z_2 - y \sqcup xy) = Z(z_1z_2 + z_2z_1 + z_3 - z_1z_2 - 2z_2z_1) = Z(z_3 - z_2z_1) = 0.$$

This gives us the MZV relation $\zeta(3) = \zeta(2,1)$. Notice that the non-admissible terms cancel out.

2.2. **Duality.** The anti-automorphism τ of $\mathfrak{H}$, defined by

$$\tau(x) := y$$
$$\tau(y) := x.$$

It is an involution and preserves $\mathfrak{H}^0$. Recall that τ being an anti-automorphism means that $\tau(vw) = \tau(w)\tau(v)$ for all words w, v, and that τ being an involution means that it is its own inverse, i.e., $\tau \circ \tau = id$. We have the following well-known duality result:

Theorem 4 (Corollary 6.2 in [5])**.** *For any word $w \in \mathfrak{H}^0$ we have $Z(w) = Z(\tau(w))$.*

Examples 5. *By considering $w = z_3 = xxy$ and applying Theorem 4, we get*

$$Z(z_3) = Z(xxy) = Z(\tau(xxy)) = Z(xyy) = Z(z_2z_1).$$

This gives us the MZV relation $\zeta(3) = \zeta(2,1)$.
Similarly,

$$\zeta(4) = Z(z_4) = Z(xxxy) = Z(\tau(xxxy)) = Z(xyyy) = Z(z_2z_1z_1) = \zeta(2,1,1).$$

2.3. **Dual product.** The product $m_\square : x\mathfrak{H} \otimes x\mathfrak{H} \to x\mathfrak{H}$ is defined in terms of τ and the harmonic shuffle m_* on $\mathfrak{H}^1$.

$$m_\square := \tau \circ m_* \circ (\tau \otimes \tau).$$

Note that $m_\square : \mathfrak{H}^0 \otimes \mathfrak{H}^0 \to \mathfrak{H}^0$. Zudilin remarked in [13] that $m_\square$ does not coincide with the usual shuffle product for MZVs. This can be seen by noticing that the weight is preserved under both duality as well as the harmonic shuffle. Depth on the other hand is transformed by duality to the difference of weight and depth. As the harmonic shuffle leads to a decrease in depth the product $m_\square$ leads to an increase in depth, in contrast to the shuffle product, which preserves both weight and depth. For example

$$m_\square(xy \otimes xy) = 2xyxy + xyyy.$$

2.4. **Other maps.** For future reference in Section 5, we now define the following maps on $\mathfrak{H}$:

$$\varphi, \quad R_u, \quad L_u, \quad \chi,$$

and their inverses.

2.4.1. *The φ automorphism.* The map φ is an involutory ($\varphi \circ \varphi = id$) automorphism on $\mathfrak{H}$, defined by

$$\varphi(x) := x + y$$
$$\varphi(y) := -y.$$

Here are some examples of φ applied to various words.

$$\varphi(xy) = -(x+y)y = -xy - yy$$
$$\varphi(xyy) = (x+y)yy = xyy + yyy$$
$$\varphi(yyy) = -yyy$$

$$\varphi(xyy + yyy) = xyy.$$

Notice that φ preserves $\mathfrak{H}^1$.

2.4.2. *The concatenation maps L_u and R_u.* The map $R_u : \mathfrak{H} \to \mathfrak{H}u$ appends $u \in \{x, y\}$ to the (right) end of a word, and R_u^{-1} is its inverse, i.e., removing a u from the end. Similarly, $L_u : \mathfrak{H} \to u\mathfrak{H}$ adds $u \in \{x, y\}$ to the left of a word, and L_u^{-1} is its inverse. Here are some examples

$$\begin{aligned} R_y(xy) &= xyy \\ R_y(xy + yy) &= xyy + yyy \\ R_x^{-1}(xyx) &= xy \\ L_y(xyy) &= yxyy \\ L_y^{-1}(yxy + yy) &= xy + y. \end{aligned}$$

2.4.3. *The map χ.* We let $\chi : \mathfrak{H} \to \mathfrak{H}$ be defined by the composition

$$\chi := R_y \circ \tau \circ \varphi.$$

Note that since $\varphi : \mathfrak{H}y \to \mathfrak{H}y$, $\tau : \mathfrak{H}y \to x\mathfrak{H}$, and $R_y : x\mathfrak{H} \to x\mathfrak{H}y$, the map χ sends $\mathfrak{H}^1$ to $\mathfrak{H}^0$. Note also that $\chi : \mathfrak{H} \to \mathfrak{H}^1$, since the last operation is R_y. Its inverse

$$\chi^{-1} = \varphi \circ \tau \circ R_y^{-1}$$

is defined for all inputs from $\mathfrak{H}y$, and $\chi^{-1} : \mathfrak{H}y \to \mathfrak{H}$. Let's see some examples.

$$\begin{aligned} \chi(xy) &= R_y\tau\varphi(xy) = R_y\tau(-xy - yy) = R_y(-xy - xx) = -xyy - xxy \\ \chi(xyy) &= R_y\tau\varphi(xyy) = R_y\tau(xyyy + yyy) = R_y(xxy + xxx) = xxyy + xxxy. \end{aligned}$$

2.5. **The Kawashima Relations.** In reference [9] a set of relations were proven to always be in the kernel of the evaluation map Z. Define the binary operation $\circledast$ for words in $\mathfrak{H}^1$ as

$$z_s v \circledast z_t w = z_{s+t}(v * w).$$

Then the Kawashima relations are summarised in following

Theorem 6 (Corollary 5.4 in [9])**.** *For all v,w $\in \mathfrak{H}y$ and $m \geq 1$*

$$\sum_{\substack{i+j=m \\ i,j\geq 1}} Z(\varphi(v) \circledast y^i)Z(\varphi(w) \circledast y^j) = Z(\varphi(v * w) \circledast y^m). \tag{4}$$

When $m = 1$ we have the following

$$L_x\varphi(v * w) \in \ker Z. \tag{5}$$

When $m = 2$:

$$(L_x\varphi(v)) * (L_x\varphi(w)) - (\varphi(v * w) \circledast z_1 z_1) \in \ker Z. \tag{6}$$

It is conjectured that the Kawashima relations with $m = 1$ and $m = 2$ give all linear MZV relations.

3. Derivation relations

We consider the so-called *derivation relations*, which give rise to linear relations among MZVs. A *derivation* d is an operator that obeys Leibniz' rule: $d(wu) = d(w)u + wd(u)$. Observe that as a consequence, derivations are completely determined by where they send the letters x and y. For $n \in \mathbb{N}$ we introduce the derivation $\partial_n \colon \mathfrak{H}^0 \to \mathfrak{H}^0$ defined by

$$\partial_n(x) := x(x+y)^{n-1}y \qquad \text{and} \qquad \partial_n(y) := -x(x+y)^{n-1}y.$$

Then we have the next result.

Theorem 7 (Theorem 3 in [8])**.** *For any word $w \in \mathfrak{H}^0$ and any $n > 0$ we have $Z(\partial_n(w)) = 0$.*

The following identity is due to Hoffman and Ohno.

Theorem 8 (Theorem 4.3 in [7])**.** *For any $w \in \mathfrak{H}^0$*

$$\partial_1(w) = w \sqcup z_1 - w * z_1. \tag{7}$$

Compare this to the following result, presented in reference [4], relating the product $m_\square$ to the derivation $\partial_2 \colon \mathfrak{H}^0 \to \mathfrak{H}^0$, defined by $\partial_2(x) := x(x+y)y$ and $\partial_2(y) := -x(x+y)y$.

Theorem 9 (Theorem 4.4 in [4])**.** *Let $w \in \mathfrak{H}^0$. Then we have*

$$\partial_2(w) = w \square z_2 - w * z_2. \tag{8}$$

The proof in [4] goes by induction and will not be rewritten here. Instead, we illustrate the theorem by calculating a few examples.

Examples 10. *First we calculate $\partial_2(w)$ and then $w\square z_2 - w * z_2$, for the word $w = xy = z_2$.*

$$\begin{aligned}\partial_2(xy) &= xxyy + xyyy - xxxy - xxyy \\ &= xyyy - xxxy \\ &= z_2z_1z_1 - z_4\end{aligned}$$

$$\begin{aligned}z_2\square z_2 - z_2 * z_2 &= \tau(\tau(z_2) * \tau(z_2)) - z_2 * z_2 \\ &= \tau(2z_2z_2 + z_4) - 2z_2z_2 + z_4 \\ &= z_2z_1z_1 - z_4.\end{aligned}$$

As another example, we do the same calculations, now with the word $w = xxy = z_3$.

$$\begin{aligned}\partial_2(xxy) &= xxyxy + xyyxy + xxxyy + xxyyy - xxxxy - xxxyy \\ &= xxyxy + xyyxy + xxyyy - xxxxy \\ &= z_2z_1z_2 + z_3z_1z_1 + z_3z_2 - z_5\end{aligned}$$

$$\begin{aligned}z_3\square z_2 - z_3 * z_2 &= \tau(z_2z_1 * z_2) - z_3 * z_2 \\ &= \tau(2z_2z_2z_1 + z_4z_1 + z_2z_3 + z_2z_1z_2) - (z_2z_3 + z_3z_2 + z_5) \\ &= z_2z_3 + z_2z_1z_2 + z_3z_1z_1 + 2z_3z_2 - z_2z_3 - z_3z_2 - z_5 \\ &= z_2z_1z_2 + z_3z_1z_1 + z_3z_2 - z_5.\end{aligned}$$

Another useful derivation is $D_n : \mathfrak{H}^0 \to \mathfrak{H}^0$, defined by

$$D_n(x) := 0 \qquad \text{and} \qquad D_n(y) := x^n y.$$

For any derivation, a bar denotes its conjugation by the anti-automorphism τ. For instance, $\overline{D_n} := \tau \circ D_n \circ \tau$. To illustrate its workings we compare: $D_1(xxy) = xxxy$, with

$$\overline{D_1}(xxy) = \tau \circ D_1 \circ \tau(xxy) = \tau \circ D_1(xyy) = \tau(xxyy + xyxy) = xxyy + xyxy.$$

4. Rooted Tree Maps

In reference [10], Tanaka introduced the notion rooted tree map, which provides a new way of looking at MZVs. Recall that a tree is a connected graph with no loops. A tree is called a rooted tree if it has a particular node called the root where any edge is oriented away from it. The trees we consider are non-planar, that is, without a plane structure. For instance, we do not distinguish between the rooted tree  and the rooted tree  . The product of rooted trees is called a rooted forest. The rooted tree maps are maps from $\mathfrak{H}$ to $\mathfrak{H}$, but to understand how to utilize them, some background information is needed.

We denote the empty forest (the one containing zero nodes) by $\mathbb{I}$. All trees are drawn with the root as the top node. For instance, the following tree,  , has one root and three more nodes directly connected to it.

We can use rooted forests to generate a free commutative algebra H over $\mathbb{Q}$.

$$H = \sum_{f:\text{rooted forests}} \mathbb{Q} \cdot f$$

The bilinear product on H is the disjoint union of forests. An example would be:

$$(2\bullet + \mathbb{I}) \cdot (3\bullet) = 6\,\bullet\bullet + 3\bullet.$$

4.1. **B_+, the grafting operator.** The operator B_+ takes a forest of rooted trees and connects all the roots to a new node, which then becomes the root of a new rooted tree. We define $B_+(\mathbb{I}) := \bullet$. Some simple examples for the grafting operator are:

$$B_+(\bullet) = \text{  },$$
$$B_+(\bullet\bullet) = \text{  },$$
$$B_+(\bullet\bullet\bullet) = \text{  }.$$

4.2. **Coproduct.** A coproduct $\Delta : H \to H \otimes H$ is defined on forest of rooted trees. It is multiplicative, such that for any two $f, g \in H, \Delta(fg) = \Delta(f)\Delta(g)$. Since all forests in H are products of trees, a definition for the coproduct of a rooted tree is needed. Let t be a rooted tree, then there is a $f \in H$ such that $t = B_+(f)$

$$\Delta(t) = \Delta \circ B_+(f) = t \otimes \mathbb{I} + (id \otimes B_+) \circ \Delta(f).$$

By letting $\Delta(\mathbb{I}) = \mathbb{I} \otimes \mathbb{I}$, the coproduct of all rooted forests can be computed recursively. Here are some simple examples.

$$\begin{aligned}
\Delta(\bullet) &= \Delta \circ B_+(\mathbb{I}) \\
&= \bullet \otimes \mathbb{I} + (id \otimes B_+) \circ \Delta(\mathbb{I}) \\
&= \bullet \otimes \mathbb{I} + (id \otimes B_+) \circ (\mathbb{I} \otimes \mathbb{I}) \\
&= \bullet \otimes \mathbb{I} + \mathbb{I} \otimes \bullet \\
\Delta(\bullet\bullet) &= \Delta(\bullet)\Delta(\bullet) \\
&= (\bullet \otimes \mathbb{I} + \mathbb{I} \otimes \bullet) \cdot (\bullet \otimes \mathbb{I} + \mathbb{I} \otimes \bullet)
\end{aligned}$$

$$= \bullet\bullet \otimes \mathbb{I} + 2\bullet \otimes \bullet + \mathbb{I} \otimes \bullet\bullet .$$

4.3. **Rooted Tree Maps.** It turns out all forests in H can be used as maps from $\mathfrak{H}$ to $\mathfrak{H}$, and the following theorem from [10] explains how it works. M is a multiplication map sending $\mathfrak{H} \otimes \mathfrak{H}$ to $\mathfrak{H}$ given by $M(v \otimes w) = vw$.

Definition 11 (Theorem 1.1 in [10])**.** *For any rooted forest f (different from $\mathbb{I}$), we can define the $\mathbb{Q}$-linear map from $\mathfrak{H}$ to $\mathfrak{H}$, which is also denoted by f, by*

(i) *If $f = \bullet$, then $f(x) := xy$ and $f(y) := -xy$,*

(i') *$B_+(f)(u) := R_y R_{x+2y} R_y^{-1} f(u)$ for $u \in \{x, y\}$,*

(i") *If $f = gh$ with $g, h \neq \mathbb{I}$, then $f(u) := g(h(u))$ for $u \in \{x, y\}$,*

(ii) *For $w \in \mathfrak{H}$ and $u \in \{x, y\}$, $f(wu) := M(\Delta(f)(w \otimes u))$.*

This definition is labeled a theorem in reference [10], where it is proven that the notion of rooted tree map is in fact well-defined and $\mathbb{Q}$-linear. We omit the proof here, and take Definition 11 to be a proper definition. Instead, we will include some examples.

$$\begin{aligned}
\bullet(xxy) &= M(\Delta(\bullet)(xx \otimes y)) \\
&= M(\bullet \otimes \mathbb{I} + \mathbb{I} \otimes \bullet)(xx \otimes y) \\
&= M(\bullet(xx) \otimes y) + M(xx \otimes \bullet(y)) \\
&= \bullet(xx)y + xx\bullet(y) \\
&= M(\bullet \otimes \mathbb{I} + \mathbb{I} \otimes \bullet)(x \otimes x))y - xxxy \\
&= \bullet(x)xy + x\bullet(x)y - xxxy \\
&= xyxy + xxyy - xxxy
\end{aligned}$$

$$\begin{aligned}
\begin{smallmatrix}\bullet\\\bullet\end{smallmatrix}(xy) &= M(\Delta(\begin{smallmatrix}\bullet\\\bullet\end{smallmatrix})(x \otimes y)) \\
&= M(\begin{smallmatrix}\bullet\\\bullet\end{smallmatrix} \otimes \mathbb{I} + \bullet \otimes \bullet + \mathbb{I} \otimes \begin{smallmatrix}\bullet\\\bullet\end{smallmatrix})(x \otimes y) \\
&= \begin{smallmatrix}\bullet\\\bullet\end{smallmatrix}(x)y + \bullet(x)\bullet(y) + x\begin{smallmatrix}\bullet\\\bullet\end{smallmatrix}(y) \\
&= (R_y R_{x+2y} R_y^{-1} \bullet(x))y - xyxy + x(R_y R_{x+2y} R_y^{-1} \bullet(y)) \\
&= (R_y R_{x+2y} R_y^{-1}(xy))y - xyxy + x(R_y R_{x+2y} R_y^{-1}(-xy)) \\
&= xxyy + 2xyyy - xyxy - xxxy - 2xxyy \\
&= 2xyyy - xyxy - xxxy - xxyy.
\end{aligned}$$

We will also need the following result from [10].

Theorem 12 (Corollary 4.5 and remark 4.6 in [10])**.** *For any rooted forest map $f \neq \mathbb{I}$, there is an element $w \in \mathfrak{H}y$ such that*

$$f\chi = \chi \mathcal{H}_w.$$

Such w is determined by

$$w = \mathcal{H}_w(\mathbf{1}) = \chi^{-1} f \chi(\mathbf{1}) = \chi^{-1} f(y).$$

Theorem 12 remarkably relates the rooted tree maps to the harmonic product, through the map χ. In Section 5 we build upon this result to find a new set of relations between the derivations ∂_n and the harmonic product.

5. Relating ∂_n to the harmonic product

The following is our first original result. It makes use of the maps defined in subsection 2.4. We give two proofs: first a calculation-heavy one, then a shorter proof using rooted tree maps.

Theorem 13. *For all $n \in \mathbb{N}$,*

$$\partial_n = \chi \circ \mathcal{H}_{z_n} \circ \chi^{-1}.$$

Proof. It's sufficient to prove that $\chi^{-1}\partial_n\chi = \mathcal{H}_{z_n}$. Let w be a word in $x\mathfrak{H}y$, which can be written as

$$w = z_{k_1} z_{k_2} \cdots z_{k_r}$$

Let's first calculate the harmonic shuffle product.

$$\begin{aligned}(9)\quad \mathcal{H}_{z_n}(w) &= w * z_n = z_n z_{k_1} z_{k_2} \cdots z_{k_r} + z_{k_1+n} z_{k_2} \cdots z_{k_r} + z_{k_1}(z_{k_2} \cdots z_{k_r} * z_n) \\ &= z_n z_{k_1} z_{k_2} \cdots z_{k_r} + z_{k_1+n} z_{k_2} \cdots z_{k_r} + z_{k_1} z_n z_{k_2} \cdots z_{k_r} + z_{k_1} z_{k_2+n} \cdots z_{k_r} + z_{k_1} z_{k_2}(z_{k_3} \cdots z_{k_r} * z_n).\end{aligned}$$

Observe that the resulting sum has $2r + 1$ terms. Each z_{k_i} in w gives 2 terms, one of the form $z_n z_{k_i}$ and another one z_{k_i+n}. In addition, there is one term of the form $z_{k_1} z_{k_2} \cdots z_{k_r} z_n$. The rest of the proof amounts to calculating $\chi^{-1}\partial_n\chi(w)$, and observing that it is in fact equal to $\mathcal{H}_{z_n}(w)$.
First, calculate $\chi(w)$.

$$\begin{aligned}\chi(w) = R_y\tau\varphi(w) &= (-1)^r R_y\tau((x+y)^{k_1-1}y \cdots (x+y)^{k_r-1}y) \\ &= (-1)^r[x(x+y)^{k_r-1} \cdots x(x+y)^{k_1-1}y].\end{aligned}$$

Recall that $\partial_n(x) = -\partial_n(y) = x(x+y)^{n-1}y$. Since $\partial_n(x+y) = 0$, the only terms of $\chi(w)$ contributing to the result of $\partial_n\chi(w)$ will be the standalone x's and the single last y. Next, calculate $\partial_n\chi(w)$:

$$\begin{aligned}\partial_n\chi(w) = (-1)^r\Big[\quad & x(x+y)^{n-1}y(x+y)^{k_r-1} \cdots x(x+y)^{k_1-1}y \\ + \quad & x(x+y)^{k_r-1}x(x+y)^{n-1}y(x+y)^{k_{r-1}-1} \cdots x(x+y)^{k_1-1}y \\ \vdots \quad & \\ + \quad & x(x+y)^{k_r-1} \cdots x(x+y)^{n-1}yx(x+y)^{k_1-1}y \\ - \quad & x(x+y)^{k_r-1} \cdots x(x+y)^{k_1-1}x(x+y)^{n-1}y \quad \Big].\end{aligned}$$

The final part amounts to calculate $\chi^{-1}\partial_n\chi(w) = \varphi\tau R_y^{-1}\partial_n\chi(w)$:

$$\begin{aligned}\chi^{-1}\partial_n\chi(w) = (-1)^r\,\varphi\Big(\quad & (x+y)^{k_1-1}y \cdots (x+y)^{k_r-1}x(x+y)^{n-1}y \\ + \quad & (x+y)^{k_1-1}y \cdots (x+y)^{k_{r-1}-1}x(x+y)^{n-1}y(x+y)^{k_r-1}y \\ \vdots \quad & \\ + \quad & (x+y)^{k_1-1}x(x+y)^{n-1}y \cdots (x+y)^{k_r-1}y \\ - \quad & (x+y)^{n-1}y(x+y)^{k_1-1}y \cdots (x+y)^{k_r-1}y \quad \Big) \\ = (-1)^{2r}\Big[\quad & x^{k_1-1}y \cdots x^{k_r-1}(x+y)x^{n-1}y \\ + \quad & x^{k_1-1}y \cdots x^{k_{r-1}-1}(x+y)x^{n-1}yx^{k_r-1}y \\ \vdots \quad & \\ + \quad & x^{k_1-1}(x+y)x^{n-1}yx^{k_2-1}y \cdots x^{k_r-1}y\end{aligned}$$

$$
\begin{aligned}
&+ \quad x^{n-1}yx^{k_1-1}yx^{k_2-1}y\cdots x^{k_r-1}y \qquad \Big] \\
=&\Big[\quad z_{k_1}\cdots x^{k_r-1}(x+y)z_n \\
&+ \quad z_{k_1}\cdots x^{k_{r-1}}(x+y)z_nz_{k_r} \\
&\vdots \\
&+ \quad x^{k_1-1}(x+y)z_nz_{k_2}\cdots z_{k_r} \\
&+ \quad z_nz_{k_1}z_{k_2}\cdots z_{k_r} \qquad \Big]
\end{aligned}
$$

Notice that the term $x^{k_i-1}(x+y)z_n$, appearing multiple times in the sum above, equals $z_{k_i}z_n + z_{k_i+n}$. Using this fact, the expression can be rewritten in the following way.

$$
\begin{aligned}
& \quad z_{k_1}\cdots z_{k_r}z_n + z_{k_1}\cdots z_{k_r+n} \\
&+ \quad z_{k_1}\cdots z_{k_{r-1}}z_nz_{k_r} + z_{k_1}\cdots z_{k_{r-1}+n}z_{k_r} \\
&\vdots \\
&+ \quad z_{k_1}z_nz_{k_2}\cdots z_{k_r} + z_{k_1+n}z_{k_2}\cdots z_{k_r} \\
&+ \quad z_nz_{k_1}z_{k_2}\cdots z_{k_r}
\end{aligned}
$$

Comparing this with equation 9, one can see that the expressions are equal, which concludes the proof. □

Theorem 13 is also provable by use of rooted tree maps, and that was in fact how the identity was discovered in the first place.

Proof. In [1] it is proven that ∂_n can be expressed as a $\mathbb{Q}$-linear combination of trees in H. Let

$$\partial_n = q_1f_1 + q_2f_2 + \cdots + q_rf_r \quad q_i \in \mathbb{Q}, f_i \in H.$$

By using Theorem 12, this can be written as

$$\partial_n = q_1\chi\mathcal{H}_{w_1}\chi^{-1} + q_2\chi\mathcal{H}_{w_2}\chi^{-1} + \cdots + q_r\chi\mathcal{H}_{w_r}\chi^{-1}, \quad q_i \in \mathbb{Q}, w_i \in \mathfrak{H}y.$$

By factorizing out the χ and χ^{-1} in each term

$$\partial_n = \chi(q_1\mathcal{H}_{w_1} + q_2\mathcal{H}_{w_2} + \cdots + q_r\mathcal{H}_{w_r})\chi^{-1}, \quad w_i \in \mathfrak{H}y.$$

And by using the harmonic shuffle product is distributive, it can be simplified to

$$\partial_n = \chi\mathcal{H}_{q_1w_1+q_2w_2+\cdots+q_rw_r}\chi^{-1}, \quad w_i \in \mathfrak{H}y.$$

Let $w = q_1w_1 + q_2w_2 + \cdots + q_rw_r$. We calculate w as in Theorem 12

$$
\begin{aligned}
w = \chi^{-1}\partial_n(y) &= \chi^{-1}(-x(x+y)^{n-1}y) \\
&= \varphi\tau R_y^{-1}(-x(x+y)^{n-1}y) \\
&= \varphi\tau(-x(x+y)^{n-1}) \\
&= \varphi(-(x+y)^{n-1}y) = x^{n-1}y.
\end{aligned}
$$

Which concludes the proof. □

6. The dual double shuffle relations

In [4] the authors discovered the connection between the dual product, the harmonic product, and ∂_2. It turns out that the set of relations in Theorem 9 is just a special case of another set of relations which always maps to the kernel of the evaluation map, Z. Recall from subsection 2.3 that the dual product of two words is defined as $w\square v := \tau(\tau(w) * \tau(v))$. The following is our second original result.

Theorem 14. *For all words w, v in $\mathfrak{H}^0$, we have*

$$w\square v - w * v \in \ker Z, \tag{10}$$

as well as

$$w\square v - w \sqcup\!\sqcup v \in \ker Z. \tag{11}$$

Proof. By using the facts that $Z(w*v) = Z(w \sqcup\!\sqcup v)$ and that $Z(w) = Z(\tau(w))$, we calculate the following.

$$\begin{aligned} Z(w\square v) &= Z(\tau(\tau(w) * \tau(v))) = Z(\tau(w) * \tau(v)) \\ &= Z(\tau(w) \sqcup\!\sqcup \tau(v)) = Z(\tau(w \sqcup\!\sqcup v)) \\ &= Z(w \sqcup\!\sqcup v) = Z(w * v) \end{aligned}$$

□

6.1. **MZV Python library.** We have developed a python library to simplify calculations. It can be found at `https://github.com/whorn/Rooted-Tree-Maps`.

It contains an implementation of the non-commutative polynomial ring $\mathbb{Q}\langle x, y\rangle = \mathfrak{H}$, of the rooted tree maps described by Tanaka in [10], of multiple different derivation operators (e.g. ∂_n, D_n and $\overline{D_n}$), of the shuffle product ($\sqcup\!\sqcup$), the harmonic product ($*$) and the dual product ($\square$).

The program permits the user doing calculations that otherwise would be unfeasible to do by hand.

See appendix A for more technical information and examples showing how to use the library.

6.2. **Dimension calculation.** Let L_k denote the number of linearly independent relations of weight k obtained by the linear part of the Kawashima relations (5). Denote D_k as the number of linearly independent MZV relations of weight k obtained by the dual double shuffle relations. $L_k \cup D_k$ is the number of linearly independent MZV reations obtained by the linear part of the Kawashima relations and the dual double shuffle relations of weight k. C_k is the conjectured amount of $\mathbb{Q}$-linearly independent MZV relations of weight k. In the table below, the values for $L_k, D_k, L_k \cup D_k$ and C_k for low values of k are calculated using our MZV python library.

k	2	3	4	5	6	7	8	9	10	11	12	13
L_k	0	1	2	5	10	23	46	98	200	413	838	1713
D_k	0	0	1	2	7	16	39	89	194	420	881	1836
$L_k \cup D_k$	0	1	2	5	11	25	53	112	232	475	971	1968
C_k	0	1	3	6	14	29	60	123	249	503	1012	2032

It turns out that the dual double shuffle relation is smaller in size than the linear part of Kawashima relations for words of weight ≤ 10, but larger for words of weight > 10. It is neither a proper subset nor a proper superset of the linear part of the Kawashima relations. We have not yet understood exactly where this set of relations fits into the bigger picture of MZV relations.

7. Our less fruitful endeavours

For comprehensiveness, this section is included, documenting some of our less successful attempts to generalize Theorems 8 and 9. They state respectively that $\partial_1(w) = w \sqcup z_1 - w * z_1$ and $\partial_2(w) = w\square z_2 - w * z_2$. Our aim was to discover well-defined products, $m_{\diamond_n} : \mathfrak{H}^0 \otimes \mathfrak{H}^0 \to \mathfrak{H}^0$, satisfying $\partial_n(w) = w \diamond_n z_n - w * z_n$ for all words w. In particular, we put some effort into looking for a product $\diamond_3$, satisfying $\partial_3(w) = w \diamond_3 z_3 - w * z_3$.

Proposition 15. *Notice that Theorem 9 can be rewritten as*

$$\partial_2(w) = w\square z_2 - w * z_2 = \tau[R^*_{z_2}, \tau](w), \tag{12}$$

where $[R^*_{z_2}, \tau] := R^*_{z_2} \circ \tau - \tau \circ R^*_{z_2}$, *and*

$$\partial_2 = \overline{D_2} - D_2 - [\overline{D_1}, D_1] = \tau[D_2 - D_1\tau D_1\tau, \tau]. \tag{13}$$

We can express ∂_3 *in terms of* $\overline{D_n}$ *and* $D_n, n = 1, 2, 3$*:*

$$\partial_3 = \overline{D_3} - D_3 - \frac{3}{4}[\overline{D_1}, D_2] - \frac{3}{4}[\overline{D_2}, D_1] + \frac{1}{4}[[\overline{D_1}, D_1], D_1] + \frac{1}{4}[\overline{D_1}[\overline{D_1}, D_1]]. \tag{14}$$

This expression can be further simplified similarly to

$$\partial_3 = \tau[D_3 - \frac{3}{4}(D_1\tau D_2\tau - D_2\tau D_1\tau) + \frac{1}{4}(2D_1\tau D_1\tau D_1 - D_1D_1\tau D_1\tau + D_1\tau D_1D_1\tau), \tau]. \tag{15}$$

Equations (12) and (13) are quite similar, but note that $R^*_{z_2} \neq D_2 - D_1\tau D_1\tau$. Deriving (12) from (13) or conversely is a non-trivial problem. As a consequence, deriving a product, $m_{\diamond_3}$, from the way more complicated expression (15) seems unfeasible.

Another approach was to define $m_{\diamond_3}$ in terms of the equality we wanted it to satisfy, i.e., $w \diamond_3 z_3 = \partial_3(w) + w * z_3$. As it turns out, naively extending the input space from $\mathfrak{H}^0 \otimes z_3$ to $\mathfrak{H}^0 \otimes \mathfrak{H}^0$ like this $w \diamond_3 v := \partial_3(w) + w * v$, does not go very well. Defined in this way, $\diamond_3$ is non-commutative and non-associative. In general it does not even return a homogeneous (i.e., all terms having the same weight) sum of words.
Realizing that the condition $w \diamond_3 z_3 = \partial_3(w) + w * z_3$ is too weak to uniquely define a product, and seeing how the naive approach is unfruitful, we moved on to other problems.

8. Outlook

The results from Sections 5 and 6 may be interesting to further explore. In Section 5, one may ask how conjugation by χ connects the seemingly disparate operations $\mathcal{H}_{z_n}$ and ∂_n? The niceness of the expression suggest the possibility that something deeper might be going on.
Regarding Section 6, one may wonder how big is the set of dual double shuffle relations, and how does it fit in with the already known sets of relations? Related to this, one may ask which relations are "missing", that is, what could we add to the set of dual double shuffle relations to end up with all MZV relations?

References

[1] H. Bachmann, T. Tanaka, *Rooted tree maps and the derivation relation for multiple zeta values*, International Journal of Number Theory, **14** No. 10 (2018), 2657–2662. arXiv:1712.01601 [math.NT]

[2] H. Bachmann, T. Tanaka, *Rooted Tree Maps and the Kawashima relations for multiple zeta values*, arXiv:1801.05381 [math.NT].

[3] V. G. Drinfel'd, *On quasitriangular quasi-Hopf algebras and on a group that is closely connected with* $\mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q})$, Algebra i Analiz **2** (1990), 149–181 (in Russian). (English translation in Leningrad Math. J., **2**(4) (1991), 829–860).

[4] K. Ebrahimi-Fard, D. Manchon, J. Singer, *Duality and (q-)multiple zeta values*, Advances in Mathematics, **298** (2016), 254–285.

[5] M. E. Hoffman, *The Algebra of Multiple Harmonic Series*, Journal of Algebra, **194** (1997), 477–495.

[6] M. E. Hoffman, *Quasi-shuffle products*, Journal of Algebraic Combinatorics, **11** (2000), 49–68.

[7] M. E. Hoffman, Y. Ohno, *Relations of multiple zeta values and their algebraic expression*, Journal of Algebra, **262** (2003), 332–347.

[8] K. Ihara, M. Kaneko, D. Zagier, *Derivation and double shuffle relations for multiple zeta values*, Compositio Mathematica, **142** (2006), 307–338.

[9] G. Kawashima, *A class of relations among multiple zeta values*, Journal of Number Theory, **129** (4) (2009), 755–788.

[10] T. Tanaka, *Rooted Tree Maps*, arXiv:1712.01029 [math.NT].

[11] D. Zagier, *Values of zeta functions and their applications*, in First European Congress of Mathematics, Vol. II, Progress in Mathematics, **120** (1994), 497–512.

[12] W. Zudilin, *Algebraic relations for multiple zeta values*, Russian Mathematical Surveys, **58**:1 (2003), 1–29.

[13] W. Zudilin, *Multiple q-zeta brackets*, Mathematics, **3**(1) (2015), 119–130.

A. The rooted tree maps Python Library

The RTM library was originally written to evaluate rooted tree maps, but was later expanded to include derivations and various products.

This appendix will explain how to do various calculations with the program and include some examples.

A.1. The `Word` class.

The `Word` class represents the words in the polynomial algebra $\mathfrak{H}$.

```
import classes as cl

A = cl.Word([1],["xy"])
B = cl.Word([1,2],["xxy","y"]
print(A)
>>> "xy"
print(B)
>>> "xxy + 2y"
```

Code 1. Initialization of a `Word` object

Various algebraic operations are also possible to do

```
import classes as cl

A = cl.Word([1],["xy"])
B = cl.Word([1,2],["xxy","y"]
print(A+B) #addition
>>> "xy + xxy + 2y"
print(B-A) #subtraction
>>> "xxy + 2y - xy"
print(A*A) #multiplication
>>> "xxyxxy + 2xxyy + 2yxxy + 4yy"
print(B**4) #exponentiation
>>> "xyxyxyxy"
print(2*B) #scalar multiplication
>>> "2xxy + 4y"
```

Code 2. Algebra `Word` object

One can also calculate the various maps that have been used in this report: $\tau, \varphi, \chi, \chi^{-1}, R_u$ and R_u^{-1}.

```
import classes as cl

A = cl.Word([1],["xyy"])

print(A.tau())
>>> "xxy"
print(A.phi())
>>> "xyy + yyy"

B = cl.Word([1],["xy"])
print(B.chi())
>>> "-xyy -xxy"
print(B.chi_inv())
>>> "-y"

```

```
#BEWARE! the R_u functions act on the object itself
A.R("y",2)
print(A)
>>> "2xyyy"
A.R_inv("y")
print(A)
>>> "2xyy"
```

CODE 3. Examples of $\tau, \varphi, \chi, \chi^{-1}, R_y$ and R_y^{-1}

`Word` objects can also be printed in different notations. Recall that a word $w \in \mathfrak{H}^1$ can be written as $w = z_{k_1} z_{k_2} \cdots z_{k_r}$.

```
import classes as cl

A = cl.Word([1,2],["xy","y"])

print(A)
>>> "xyy + 2y"
print(A.toBracketNotation())
>>> "(2,1) + 2(1)"
print(A.toZeta())
>>> "z(2,1) + 2z(1)"
```

CODE 4. Various ways of printing `Word` objects

It is also possible to calculate the shuffle, harmonic shuffle and dual shuffle product of two `Word` objects.

```
import classes as cl

A = cl.Word([1],["xy"])
B = cl.Word([1],["xxy"])

print(A.harmonic_shuffle(B))
>>> "xyxxy + xxyxy + xxxxy"
print(A.shuffle(A))
>>> "2xyxy + 4xxyy"
print(A.square_shuffle(A)) #dual shuffle
>>> "xyyxy + xyxyy + xyyyy"
```

CODE 5. An example of the three different products

The last useful feature for the words are derivations. The derivations ∂_n, D_n and $\overline{D_n}$ have been implemented. We let ∂_n be `delta` and D_n be `DELTA`.

```
import classes as cl

A = cl.Word([1],["xy"])

print(A.delta(2))
>>> "xyyy - xxxy"
print(A.DELTA(1))
>>> "xxy"
print(A.DELTA_BAR(1))
>>> "xyy"
```

CODE 6. An example of various derivations applied on a `Word` object

A.2. **The `Forest` class.**

The `Forest` class represents the rooted tree maps. The way we have chosen to represent trees and forests is with a string of 1's, 2's and 0's. 0 acts as B_+ operator on the forest. We'll start off with some examples to ease the explanation of the notation.

The empty forest, $\mathbb{I}$, is represented by the empty string `""`. $B_+(\mathbb{I}) = \bullet$ is represented by the string `"0"`.The forest $\vdots = B_+(\bullet)$ is represented by the string `"00"`. $\bullet\bullet = B_+(\mathbb{I}) \cdot B_+(\mathbb{I})$ is represented by "0102". $\curlyvee = B_+(\bullet\bullet)$ is represented by `"01020"`

`"0"` acts as a B_+-operator on the *current* forest, while `"1"` and `"2"` works as opening and ending parenthesis. This also means that representations of forest are not unique, as the forest themselves are non-planar graphs.

```
import classes as cl

A = cl.forest_from_string("0")
print(A)
>>> "0"
```

Code 7. Initialization of a `Forest` object

It is also possible to merge to forests by multiplying them together.

```
import classes as cl

A = cl.forest_from_string("0")
B = cl.forest_from_string("00")
C = A*B
print(C)
>>> "01002"
```

Code 8. Multiplication of two `Forest` objects

Calculation of the coproduct of any forest is also possible. We will use the coproduct of $\Delta(\curlyvee)$ and

```
import classes as cl

A = cl.forest_from_string("01020")
A_cp = A.coproduct()
print(A_cp)
>>> "(01020)X(I) + (0102)X(0) + 2(0)X(00) + (I)X(01020)"
```

Code 9. Example of coproduct calculation

The forest class can also evaluate objects from the `Word` class. This can be used to evaluate the rooted tree maps. Here is an example.

```
import classes as cl

A = cl.forest_from_string("01020")
w = cl.Word([1],["xy"])
v = A(w)
print(v)
>>> "-3xyxyy + 1xxxyy + 2xyyyy + -4xxyyy + -1xyyxy + -2xyxxy + 1xxxxy"
```

Code 10. Evaluation of a rooted tree map

Department of Mathematical Sciences, Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway.

E-mail address: viktorba@stud.ntnu.no

E-mail address: w.hornslien@gmail.com

E-mail address: kurusch.ebrahimi-fard@ntnu.no