Quadrangular Polylogarithms

Julian Kaufmann[1], Daniel Yuan[2] Mentor: Prof. Christian Zickert[2]
1 – University of Notre Dame, 2 – University of Maryland

INTRODUCTION

Polylogarithms–functions that extend the notion of the traditional logarithm–have been studied rigorously for their interesting functional relations. Many famous mathematicians such as Abel, Euler, Kummer, and Lobachevsky found functional relations among these polylogarithms. In recent research in the field, a conjecture has been made that all polylogarithm relations of a specific weight stem from one single larger polylogarithm relation. In this project, we developed tools to explicitly compute and create these top-level polylogarithm relations, verified that they indeed hold, and retrieved the known relations from the top-level relation.

PRELIMINARIES

Formally, a polylogarithm of weight n is defined as the power series,

$$Li_n(x) = \sum_{k=1}^{\infty} \frac{x^k}{k^n}$$

Similarly, a multiple polylogarithm of depth d and weight n is defined as the following sum,

$$Li_{n_1,\ldots,n_d}(x_1,\ldots,x_d) = \sum_{k_1<\cdots<k_d} \frac{{x_1}^{k_1}\ldots {x_d}^{k_d}}{{k_1}^{n_1}\ldots {k_d}^{n_d}}$$

Besides these power series definitions, polylogarithms can be equivalently given by iterated integrals of one-forms. As an example:

$$Li_4(x) = \int_\gamma^{\cdot} \frac{dz}{1-z}\,\frac{dz}{z}\,\frac{dz}{z}\,\frac{dz}{z}$$

In fact, any multiple polylogarithm can be written as an iterated integral of the following form [4]:

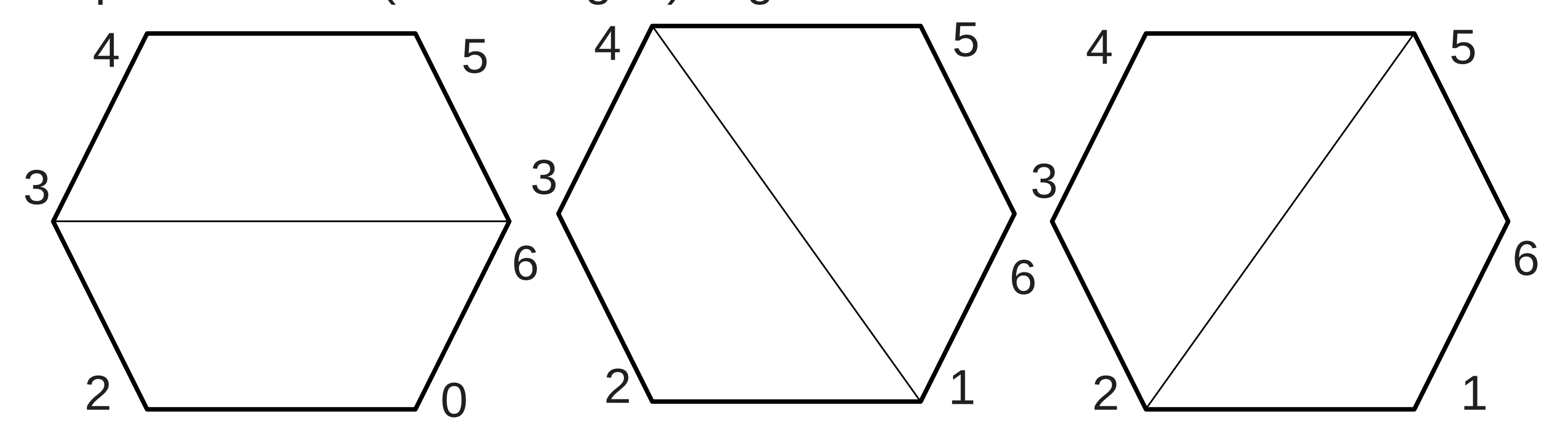
$$\int_\gamma^{\cdot} d\log(f_1)\ldots d\log(f_d)$$

From these expressions, the tensor product of the entries $f_1 \ldots f_d$ can be taken. This is known as the symbol of the multiple polylogarithm. Remarkably, if two polylogarithm expressions are equal, then they have the same symbol and are equivalent up to products of lower weight and constants [3]. These are the relations we are interested in. An example of such a relation, known as the Abel 5-term relation, is shown below:

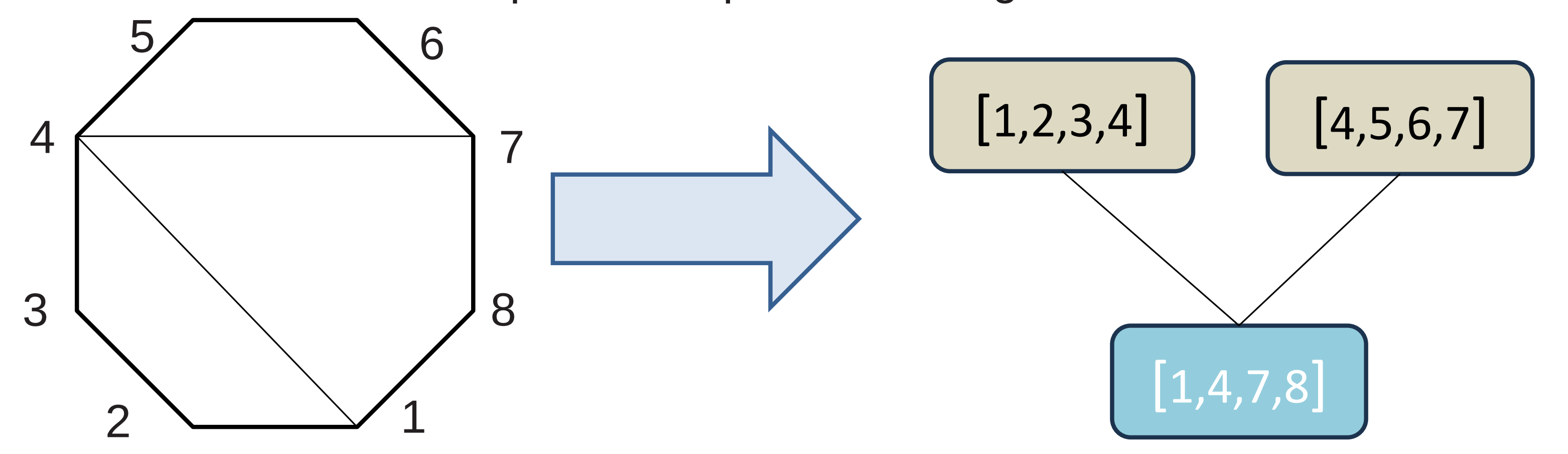
$$Li_2(x) - Li_2(y) + Li_2\left(\frac{y}{x}\right) - Li_2\left(\frac{y(1-x)}{x(1-y)}\right) + Li_2\left(\frac{1-x}{1-y}\right) = -\log(x)\log\left(\frac{1-x}{1-y}\right) + \frac{\pi^2}{6}$$

PROCEDURES

The top-level relation is given in terms of quadrangular polylogarithms. To create the quadrangular polylogarithm of weight n, we examine the $2n+2$-gon (with its vertices labelled 0 through $2n+2$) and subdivide it into quadrilaterals. Each of these subdivisions is called a quadrangulation. An example for $n=2$ (the hexagon) is given below.



Next, we assign a vertex of a tree to each quadrangle in a quadrangulation and label it with the cross-ratio of its four vertices (denoted by a four-tuple encased in brackets). Then, we select the vertex whose corresponding quadrangle includes both 0 and $2n+2$ and denote it as the root. An example of this procedure is given below for $n=3$.



To convert these trees to polylogarithms, we first need the grafting operator. Let $t = t_1,\ldots,t_n$ be a collection of trees. Then, the grafting operator, $B_a^+(t)$ is a map which outputs a single tree obtained by joining the roots of $t_1,\ldots,t_n$ onto the common new root a. Then, we define a map from the tree to mathematical words, the Arborification map, as:

$$Arb\left(B_a^+(t)\right) = \begin{cases} -aArb(t) & \text{if } p(a) = 1 \\ aArb(t) + a \cdot Arb(t) & \text{if } p(a) = 0 \end{cases}$$

where $p(a)$ denotes the parity of the first vertex in each quadrangle. Finally, we can convert the result of this expression into a polylogarithm by having all its letters denote arguments of a multiple polylogarithm. Summing over all the possible quadrangulations then yields the quadrangular polylogarithm of weight n. This process utilizes the Hopf algebra structure of both the trees created by the quadrangulations and the Hopf Algebra of words. In fact, the Arborification map is a unique Hopf Algebra homomorphism between these two [2]. Using these quadrangular polylogarithms, the top-level relation is given by [1]:

$$\sum_{0\le i_0<\cdots<i_{2r+1\le n+2}} (-1)^{i_0+\cdots+i_{2r+1}} QLi_n^{sym}\left(x_{i_0}, x_{i_1},\ldots,x_{i_{2r+1}}\right) = 0$$

RESULTS

We developed tools to compute a quadrangular polylogarithm of arbitrary weight by using recursion to compute all the quadrangulations of a polygon. Then we encoded these data in trees which were evaluated using the Arborification map, giving us the quadrangular polylogarithm for any weight. The quadrangular polylogarithm for $n=3$ is given below:

$$\begin{gathered}
-Li_{1,1,1}([1,2,3,8],[3,4,5,8],[5,6,7,8]) + Li_{1,1,1}([1,2,3,8],[3,4,7,8],[4,5,6,7]) \\
-Li_{1,1,1}([1,2,3,8],[3,6,7,8],[3,4,5,6]) + Li_{1,1,1}([1,2,5,8],[2,3,4,5],[5,6,7,8]) \\
+Li_{1,2}([1,2,5,8],[2,3,4,5]\cdot[5,6,7,8]) + Li_{1,1,1}([1,2,5,8],[5,6,7,8],[2,3,4,5]) \\
-Li_{1,2}([1,2,7,8],[2,3,4,5,6,7]) - Li_{1,1,1}([1,2,7,8],[2,3,4,7],[4,5,6,7]) \\
+Li_{1,1,1}([1,2,7,8],[2,3,6,7],[3,4,5,6]) - Li_{1,1,1}([1,2,7,8],[2,5,6,7],[2,3,4,5]) \\
-Li_{1,1,1}([1,4,5,8],[1,2,3,4],[5,6,7,8]) - Li_{1,2}([1,4,5,8],[1,2,3,4]\cdot[5,6,7,8]) \\
-Li_{1,1,1}([1,4,5,8],[5,6,7,8],[1,2,3,4]) + Li_{1,1,1}([1,4,7,8],[1,2,3,4],[4,5,6,7]) \\
+Li_{1,2}([1,4,7,8],[1,2,3,4]\cdot[4,5,6,7]) + Li_{1,1,1}([1,4,7,8],[4,5,6,7],[1,2,3,4]) \\
-Li_{1,1,1}([1,6,7,8],[1,2,3,6],[3,4,5,6]) + Li_{1,1,1}([1,6,7,8],[1,2,5,6],[2,3,4,5]) \\
-Li_{1,1,1}([1,6,7,8],[1,4,5,6],[1,2,3,4])
\end{gathered}$$

Summing over the symmetrized versions of the quadrangular polylogarithms with various indices and checking the results with our own symbol maps we found the top-level expression to have several sign errors as its symbol (modulo product) was not zero. So, we used a pattern of sign correcting factors to develop the true top-level expressions. The expression for $n=2$ is given below:

$$\begin{gathered}
Li_3([1,2,3,5]) + Li_{1,2}([1,2,3,6],[3,4,5,6]) + Li_{2,1}([1,2,3,6],[3,4,5,6]) \\
-Li_3([1,2,3,6]) - Li_3([1,2,4,5]) + Li_3([1,2,4,6]) - Li_{1,2}([1,2,5,6],[2,3,4,5]) \\
-Li_{2,1}([1,2,5,6],[2,3,4,5]) + Li_3([1,3,4,5]) - Li_3([1,3,4,6]) + Li_3([1,3,5,6]) \\
+Li_{1,2}([1,4,5,6],[1,2,3,4]) + Li_{2,1}([1,4,5,6],[1,2,3,4]) - Li_3([1,4,5,6]) \\
-Li_3([2,3,4,5]) + Li_3([2,3,4,6]) - Li_3([2,3,5,6]) + Li_3([2,4,5,6]) = \mathbf{0}
\end{gathered}$$

REFREENCES:

[1] – Andrei Matveiakin and Daniil Rudenko. Cluster polylogarithms i: Quadrangular polylogarithms. August 2022.

[2] – Daniil Rudenko. On the goncharov depth conjecture and a formula for volumes of orthoschemes. July 2022.

[3] - Don Zagier. Polylogarithms, Dedekind zeta functions and the algebraic K-theory of elds. In Arithmetic algebraic geometry (Texel, 1989), volume 89 of Progr. Math., pages 391430. Birkhäuser Boston, Boston, MA, 1991.

[4] – Zachary Greenberg, Dani Kaufman, Haoran Li, and Christian K. Zickert. Hopf algebras of multiple polylogarithms, and holomorphic 1-forms. arXiv:2211.08337, 2022.

Acknowledgement: NSF REU grant DMS-2149913