Toward a Categorification of Biquandle Brackets

A Khovanov homology-style construction extended to biquandle brackets, and associated Mathematica packages for computations

Vilas Winstein
(Joint work with Adu Vengal)

A thesis presented for the
"Honors Research" graduation distinction
in the College of Arts and Sciences

Mathematics Department
The Ohio State University
Columbus, Ohio
April 20, 2020

Toward a Categorification of Biquandle Brackets

A Khovanov homology-style construction extended to biquandle brackets, and associated Mathematica packages for computations

Adu Vengal **Vilas Winstein**

Abstract

In their paper entitled "Quantum Enhancements and Biquandle Brackets," Nelson, Orrison, and Rivera introduced *biquandle brackets*, which are customized skein invariants for biquandle-colored links. These invariants generalize the Jones polynomial, which is categorified by Khovanov homology. At the end of their paper, Nelson, Orrison, and Rivera asked if the methods of Khovanov homology could be extended to obtain a categorification of biquandle brackets.

We outline herein a Khovanov homology-style construction that is an attempt to obtain such a categorification of biquandle brackets. The resulting knot invariant does generalize Khovanov homology, but the biquandle bracket is not always recoverable, meaning the construction is not a true categorification of biquandle brackets.

However, the construction does lead to a definition that gives a "canonical" biquandle 2-cocycle associated to a biquandle bracket, which, to the authors' knowledge, was not previously known.

Additionally, the authors have created multiple Mathematica packages that can be used for experimental computations with biquandles, biquandle brackets, biquandle 2-cocycles, and the newly-discovered canonical biquandle 2-cocycle associated to a biquandle bracket.

We provide herein an explanation of these Mathematica packages, including example computations and an appendix containing the full source code. The packages may also be downloaded from vilas.us/biquandles.

Contents

Chapter 1

Introduction

A *knot* is an embedding of a circle into three-dimensional space, which can be thought of simply as a curve that has a trajectory that does not intersect itself and that ends where it started so it forms a closed loop. A *link* is an embedding of a collection of circles into three-dimensional space; this means multiple different strands can be knotted together or knotted with themselves. The central question of knot theory asks when two knots (or links) are the same in the sense that one can be continuously deformed into the other without being cut or intersecting itself.

This problem may seem intractible at first, but many simplifications have been made. For example, the use of *knot diagrams* translates the question into one of a more combinatorial nature about pictures containing a finite amount of data. The *Reidemeister moves* are a collection of manipulations that can be performed on such diagrams, and it is a powerful fact that equivalent links have diagrams which are equivalent up to modification by the Reidemeister moves [9].

Knot invariants are another central tool in knot theory. These are mathematical objects that can be assigned to links or link diagrams in a systematic way, such that when the link is continuously deformed in legal ways, the object does not change. If one computes the value of an invariant on two different links (or link diagrams) and finds two distinct values, this gives a proof that the two links are not equivalent.

The *Jones polynomial* is a particular knot invariant taking polynomial values, discovered by Vaughan Jones, that is a central subject of study in knot theory. It can be calculated by computing smoothings of knot diagrams and keeping track of combinatorial data along the way. This process can also be described by a *skein relation*. For a full exposition of the Jones polynomial, see [5].

Khovanov homology is another knot invariant, which is a categorification of the Jones polynomial. The values of this invariant are not polynomials, but rather they are sequences of modules obtained from the cohomology of a certain cochain complex. Khovanov homology categorifies the Jones polynomial because, when a particular quantity (the graded Euler characteristic) is measured from the sequence of modules in the value of the invariant, one recovers the Jones polynomial. For more details and a construction of Khovanov homology, see [6] or [1].

Biquandles are a type of algebraic structure whose axioms parallel the Reidemeister moves of knot theory. Because of this, biquandles are the basis for many invariants of knots and links. In particular, the *biquandle counting invariant* is simply the number of ways to color a link diagram with elements of a biquandle so that relationships between colors at crossings are satisfied. In [8], Sam Nelson et. al. introduced an enhancement of the biquandle counting invariant, called the *biquandle bracket*. This is a type of skein relation depending on biquandle colorings. A *biquandle 2-cocycle* is another type of function on a biquandle that can be used to define a link invariant, arising from cohomology theory.

Biquandle brackets generalize the Jones polynomial in a natural way. In [8], Nelson et. al. asked whether a Khovanov homology-style categorification of the biquandle bracket is possible. Herein, we provide a construction of what seems (to the authors) to be the most natural step from Khovanov homology toward a categorification of biquandle brackets. The invariant we obtain generalizes Khovanov homology, but it is not a true categorification of all biquandle brackets: in some cases, the biquandle bracket cannot be recovered from our invariant.

Nevertheless, the invariant does lead to a way of assigning a biquandle 2-cocycle to any given biquandle

bracket, and the relationship and power of the invariant associated with this new biquandle 2-cocycle may be of interest. To this end, we provide some Mathematica packages that can be used to do experimentations with biquandles, biquandle brackets, and biquandle 2-cocycles, including this new canonical biquandle 2-cocycle associated with a biquandle bracket.

This paper is structured as follows. In chapter 2, we review definitions which we will require for our results, including the definition of biquandles, biquandle brackets, and biquandle 2-cocycles. In chapter 3, we present the construction of our new invariant, which we call "biquandle cohomology," and we provide a proof of its invariance as well as an explanation of the canonical 2-cocycle associated with a biquandle bracket. Section 3.5 also provides some questions for further inquiry. In chapter 4, we present documentation for the Mathematica packages used by the authors to conduct experiments and now made available alongside this paper at vilas.us/biquandles. Finally, in appendix A, we provide the full source code for all of these mathematica packages.

This work has been done as a part of the undergraduate research program "Knots and Graphs" at the Ohio State University, during the summer of 2019. It is a continuation of work done in the same program during the summer of 2018 in [4]. We are grateful to the OSU Honors Program Research Fund and to the NSF-DMS #1547357 RTG grant: Algebraic Topology and Its Applications for financial support. In addition, we are grateful to our advisor, Sergei Chmutov, for his help.

Chapter 2

Preliminaries

To establish our notation and introduce the topics, we provide the following definitions. We follow the notation and conventions in [8].

2.1 Biquandles

Definition 1. A *biquandle* is a set X with two binary operations $\underline{\triangleright}, \overline{\triangleright}$ such that $\forall x, y, z \in X$,

(i) $x \underline{\triangleright} x = x \overline{\triangleright} x$

(ii) The maps $\alpha_y(x) = x \overline{\triangleright} y$, $\beta_y(x) = x \underline{\triangleright} y$, and $S(x, y) = (y \overline{\triangleright} x, x \underline{\triangleright} y)$ are invertible.

(iii) The following exchange laws are satisfied:

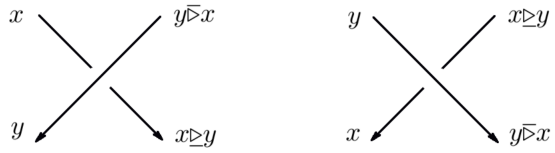
$$(x \underline{\triangleright} y) \underline{\triangleright} (z \underline{\triangleright} y) = (x \underline{\triangleright} z) \underline{\triangleright} (y \overline{\triangleright} z)$$
$$(x \underline{\triangleright} y) \overline{\triangleright} (z \underline{\triangleright} y) = (x \overline{\triangleright} z) \underline{\triangleright} (y \overline{\triangleright} z)$$
$$(x \overline{\triangleright} y) \overline{\triangleright} (z \overline{\triangleright} y) = (x \overline{\triangleright} z) \overline{\triangleright} (y \underline{\triangleright} z).$$

If $x \overline{\triangleright} y = x$ for all $x, y \in X$, then X is called a *quandle*. When there is no danger of confusion, we will write the biquandle $(X, \overline{\triangleright}, \underline{\triangleright})$ simply as X.

Remark 1. If X is a finite biquandle, we can represent all of the information about it in two operation tables. Fix some ordering on the elements of X and label them with the integers 1 through n (where n is the size of X). Then the operation table for $\underline{\triangleright}$ is an $n \times n$ matrix of integers in $\{1, \ldots, n\}$, and the (i, j) entry of this matrix is $i \underline{\triangleright} j$. The operation table for $\overline{\triangleright}$ is defined similarly. For example, the following operation tables represent a biquandle on three elements.

$$\underline{\triangleright}: \begin{bmatrix} 2 & 1 & 2 \\ 1 & 3 & 3 \\ 3 & 2 & 1 \end{bmatrix} \qquad \overline{\triangleright}: \begin{bmatrix} 3 & 3 & 3 \\ 2 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

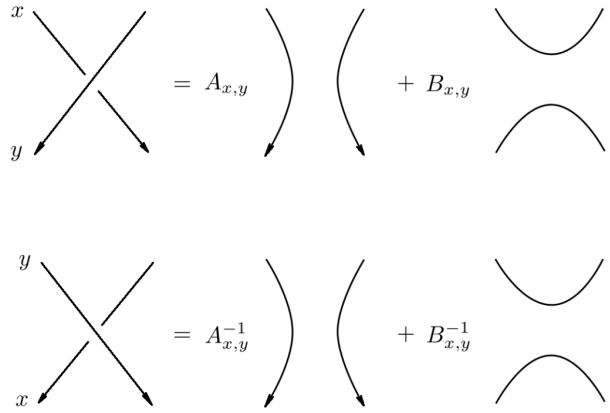
The conditions in the biquandle definition are analogous to the Reidemeister moves in knot theory when we interpret $x \underline{\triangleright} y$ as "x passing under y" and $x \overline{\triangleright} y$ as "x passing over y" in the following way:



Fix a biquandle X. An X-coloring of an oriented knot (or link) diagram L is an assignment of an element of X to each strand in the diagram such that the above relationships hold at each crossing. Then the biquandle axioms are precisely what is required for the X-coloring to be preserved as Reidemeister moves are performed on the diagram. For this reason, the number of X-colorings of a diagram is a link invariant, called the *biquandle counting invariant*, and denoted $\Phi_X^{\mathbb{Z}}(L)$.

2.2 Biquandle Brackets

In [8], an enhancement of the biquandle counting invariant is introduced. For each X-coloring of D, one can perform a smoothing operation similar to the construction in the Kauffman bracket, but this time keeping track of the colorings at each crossing as follows:



Where for each $x, y \in X$, $A_{x,y}$ and $B_{x,y}$ are invertible elements of some commutative ring with unity R. Additionally, the removal of a circle with no crossings should correspond to multiplication by some element $\delta \in R$, and to correct for the additional states generated by kinks (from the first Reidemeister move), a writhe factor should be included, which can simply be an appropriate power of some element $w \in R^\times$. For the bracket to be an invariant of an X-colored link, it should not change when Reidemeister moves are applied and the X-coloring is updated correspondingly. Below are the conditions that must be satisfied by A, B, δ, and w for this to be true. For more details, see [8].

Definition 2. A *biquandle bracket* on a biquandle X with values in commutative ring (with unity) R is a pair of maps $A, B : X \times X \to R^\times$ and two distinguished elements $\delta \in R, w \in R^\times$ which satisfy the following conditions.

(i) For all $x \in X$, $\delta A_{x,x} + B_{x,x} = w$ and $\delta A^{-1}_{x,x} + B^{-1}_{x,x} = w^{-1}$.

(ii) For all $x, y \in X$, $\delta = -A_{x,y}B^{-1}_{x,y} - A^{-1}_{x,y}B_{x,y}$.

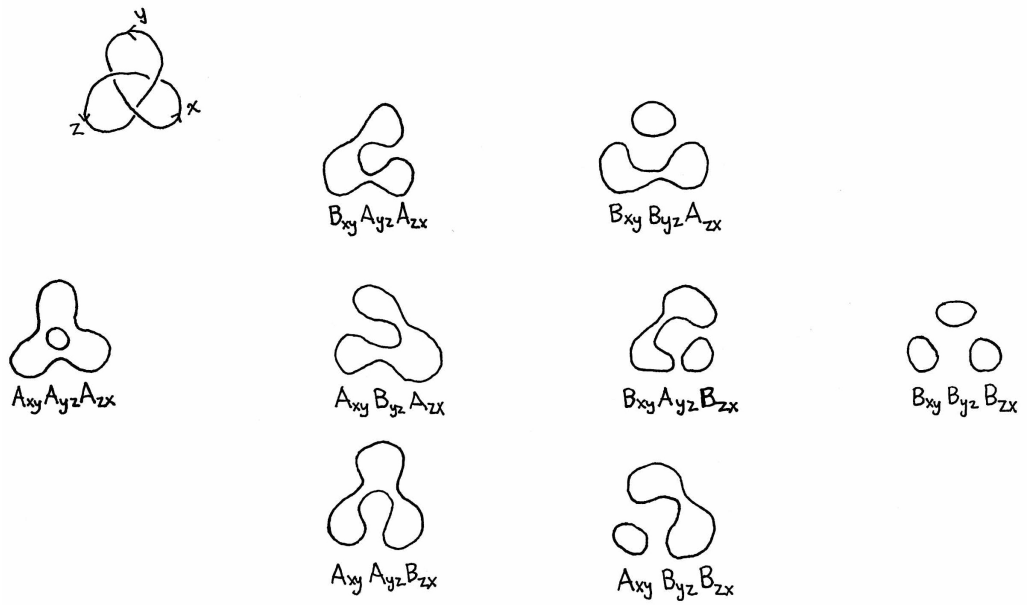
(iii) For all $x, y, z \in X$, all of the following equations hold.

$$
\begin{aligned}
A_{x,y}A_{y,z}A_{x\underline{\triangleright} y,z\overline{\triangleright} y} &= A_{x,z}A_{y\overline{\triangleright} x,z\overline{\triangleright} x}A_{x\underline{\triangleright} z,y\underline{\triangleright} z},\\
A_{x,y}B_{y,z}B_{x\underline{\triangleright} y,z\overline{\triangleright} y} &= B_{x,z}B_{y\overline{\triangleright} x,z\overline{\triangleright} x}A_{x\underline{\triangleright} z,y\underline{\triangleright} z},\\
B_{x,y}A_{y,z}B_{x\underline{\triangleright} y,z\overline{\triangleright} y} &= B_{x,z}A_{y\overline{\triangleright} x,z\overline{\triangleright} x}B_{x\underline{\triangleright} z,y\underline{\triangleright} z},\\
A_{x,y}A_{y,z}B_{x\underline{\triangleright} x,z\overline{\triangleright} y} &= A_{x,z}B_{y\overline{\triangleright} x,z\overline{\triangleright} z}A_{x\underline{\triangleright} z,y\underline{\triangleright} z} + A_{x,z}A_{y\overline{\triangleright} x,z\overline{\triangleright} x}B_{x\underline{\triangleright} z,y\underline{\triangleright} z}\\
&\quad + \delta A_{x,z}B_{y\overline{\triangleright} x,z\overline{\triangleright} x}B_{x\underline{\triangleright} z,y\underline{\triangleright} z} + B_{x,z}B_{y\overline{\triangleright} x,z\overline{\triangleright} x}B_{x\underline{\triangleright} z,y\underline{\triangleright} z},\\
B_{x,z}A_{y\overline{\triangleright} x,z\overline{\triangleright} x}A_{x\underline{\triangleright} z,y\underline{\triangleright} z} &= B_{x,y}A_{y,z}A_{x\underline{\triangleright} y,z\overline{\triangleright} y} + A_{x,y}B_{y,z}A_{x\underline{\triangleright} y,z\overline{\triangleright} y}\\
&\quad + \delta B_{x,y}B_{y,z}A_{x\underline{\triangleright} y,z\overline{\triangleright} y} + B_{x,y}B_{y,z}B_{x\underline{\triangleright} y,z\overline{\triangleright} y}.
\end{aligned}
$$

Note that we denote $A(x, y)$ and $B(x, y)$ by $A_{x,y}$ and $B_{x,y}$. Additionally, since δ and w are determined by the maps A and B, we will generally denote a biquandle bracket simply by the pair $\beta = (A, B)$. Finally, if β is a biquandle bracket on a biquandle X taking values in R, then we say β is an *X-bracket*.

If f is a coloring of an oriented link, the value of the biquandle bracket β on f is denoted $\beta(f)$.

Example 1. Here is a computation of the value of $\beta(f)$ for a coloring f (shown at the top-left corner) of the trefoil knot:



$$\beta(f) = w^{-3}\begin{pmatrix} & +\delta B_{x,y}A_{y,z}A_{z,x} & +\delta^2 B_{x,y}B_{y,z}A_{z,x} & \\ \delta^2 A_{x,y}A_{y,z}A_{z,x} & +\delta A_{x,y}B_{y,z}A_{z,x} & +\delta^2 B_{x,y}A_{y,z}B_{z,x} & +\delta^3 B_{x,y}B_{y,z}B_{z,x} \\ & +\delta A_{x,y}A_{y,z}B_{z,x} & +\delta^2 A_{x,y}B_{y,z}B_{z,x} & \end{pmatrix}.$$

The oriented link invariant corresponding to β, denoted $\Phi_X^\beta(L)$ simply the multiset of all biquandle bracket values, one for each valid X-coloring of the diagram:

$$\Phi_X^\beta(L) = \{\beta(f) : f \text{ is a valid } X\text{-coloring of } L\}.$$

Note that $\Phi_X^\beta(L)$ is an enhancement of the biquandle counting invariant $\Phi_X^{\mathbb{Z}}(L)$ because the counting invariant is simply the cardinality of this multiset.

Remark 2. If X is finite and we fix an ordering $X = \{x_1, \ldots, x_n\}$, we can encapsulate all of the information about a biquandle bracket in a presentation matrix. This is an n by $2n$ matrix M over R with entries $M_{i,j} = A_{x_i,x_j}$ and $M_{i,n+j} = B_{x_i,x_j}$ for $i, j \in \{1, 2, ..., n\}$.

Example 2. Let X be the biquandle given by the following operation table.

$$\underline{\triangleright}: \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix} \qquad \overline{\triangleright}: \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix}$$

This biquandle's operations simply flip the left operand, regardless of the right operand. Thus, in any X-colored link diagram, if one follows a particular strand, the color will alternate at every crossing. Let $R = \mathbb{Z}_2[t]/\left(1 + t + t^3\right)$. Then the following presentation matrix defines an X-bracket.

$$\left[\begin{array}{cc|cc} 1 & 1+t & t & t+t^2 \\ 1+t^2 & 1 & 1 & t \end{array}\right]$$

This biquandle bracket was found in [8].

Example 3. Let X be any biquandle, and let R be any commutative ring. If $A_{x,y} = a$ and $B_{x,y} = b$ for all $x, y \in X$ and some $a, b \in R^\times$, then the X-bracket (A, B) is called a "constant" biquandle bracket (the reader should verify that this does indeed define an X-bracket). In general, the value of a biquandle bracket on links is unchanged when all values of $A_{x,y}$ and $B_{x,y}$ are scaled by a common factor of $R^\times$ (see [8]). So, dividing through by b, the above bracket gives the same invariant as the bracket $A_{x,y} = \frac{a}{b}$, $B_{x,y} = 1$ for all $x, y \in X$. By considering the maps A, B as instead taking values in $R\left[\left(\frac{a}{b}\right)^{\pm 1/2}\right]$, we can make the substitution $q^2 = \frac{a}{b}$ and divide everything through by q to yield the equivalent bracket (when treated over

$R\left[\left(\frac{a}{b}\right)^{\pm 1/2}\right]$), $A_{x,y} = q$, $B_{x,y} = q^{-1}$ for all $x, y \in X$. Now, for any particular X-coloring of a link, the value of this bracket is evidently the Jones polynomial of the link, which is a Laurent polynomial in the variable q^2. Hence the invariant itself still takes values in R rather than $R\left[\left(\frac{a}{b}\right)^{\pm 1/2}\right]$. Therefore, the value of a constant biquandle bracket (with $A_{x,y} = a$ and $B_{x,y} = b$) is a multiset containing the Jones polynomial evaluated at $\frac{a}{b}$, and it contains this value with multiplicity equal to the number of valid X-colorings of the link.

Note in the above construction that we only obtain the *value* of the Jones polynomial at $q \in R$. If we want to retain the full power of the Jones polynomial, we can take $R = \mathbb{Z}[t, t^{-1}]$ and take $q = t$.

2.3 Biquandle 2-Cocycles

Next, we define a biquandle 2-cocycle following the notation of [8].

Definition 3. Let X be a biquandle, and let G be an abelian group (written multiplicatively here). A function $\phi : X \times X \to G$ is a *biquandle 2-cocycle* if, for all $x, y, z \in X$, we have

(i) $\phi(x, x) = 1$,

(ii) $\phi(x, y) \cdot \phi(y, z) \cdot \phi\left(x \underline{\triangleright} y, z \overline{\triangleright} y\right) = \phi(x, z) \cdot \phi\left(y \overline{\triangleright} x, z \overline{\triangleright} x\right) \cdot \phi\left(x \underline{\triangleright} z, y \underline{\triangleright} z\right)$.

A 2-cocycle ϕ can be used to define the *biquandle 2-cocycle invariant*, as seen in [2]. Namely, for each valid X-coloring of a link, compute the value $\prod_\tau \phi\left(x_\tau, y_\tau\right)^{\epsilon(\tau)}$, where τ ranges across all crossings in the colored link, $\epsilon(\tau)$ is the sign (either $+1$ or -1) of τ, and x_τ, y_τ are the biquandle colors of the arcs on the left side of the crossing when it is oriented so that strands point downwards, following a similar convention to the biquandle bracket above. The value of the biquandle 2-cocycle invariant associated to ϕ is then the multiset of all such values, one for each valid X-coloring of the link.

Remark 3. Again if X is finite, we construct a presentation matrix for a cocycle in the same fashion as with the biquandle brackets; fixing the ordering $X = \{x_1, \ldots, x_n\}$, the presentation matrix P for a cocycle is an $n \times n$ matrix over A with entries $P_{i,j} = \phi\left(x_i, x_j\right)$.

Example 4. Let X be the biquandle described in Example 1 above. Let A be the free abelian group on two symbols, a and b. Then the following presentation matrix defines a biquandle 2-cocycle $\phi : X \times X \to A$.

$$\begin{bmatrix} 1 & a \\ b & 1 \end{bmatrix}$$

The invariant corresponding to ϕ is trivial on all knots (single-component links). In fact, more is true: for any X-colored knot diagram, at any crossing τ, we have $x_\tau = y_\tau$ (so that $\phi\left(x_\tau, y_\tau\right) = 1$). Additionally, for a two-component link, the invariant corresponding to ϕ is the multiset $\left\{1, 1, (ab)^\ell, (ab)^\ell\right\}$, where ℓ is the linking number of the two components of the link. For a proof of these facts, see example 3 in [4].

Chapter 3

Biquandle Homology

We now present the construction of a Khovanov homology-style invariant of links which extends Khovanov's original construction in [6].

3.1 Group-Graded Modules and Complexes

Khovanov's original construction involves taking the homology of chain complexes of $\mathbb{Z}$-graded modules. Calculating the graded Euler characteristic of these homologies yields the Jones polynomial in a variable q, where q is the generator of $\mathbb{Z}$, and the additive structure of $\mathbb{Z}$ is written multiplicatively (i.e. $\mathbb{Z} = \{\dots, q^{-2}, q^{-1}, 1, q, q^2, \dots\}$). With the aim of obtaining the biquandle bracket as the graded Euler characteristic of a homology invariant, we need to extend our grading from $\mathbb{Z}$ to an arbitrary abelian group.

Definition 4. If H is an abelian group and S is a commutative ring, an *H-graded S-module* is an S-module M which can be decomposed as a direct sum

$$M = \bigoplus_{h \in H} M_h$$

of S-modules. If $a \in M_h$, we say that a has *degree* h, and write $\deg(a) = h$.

Note that direct sums and tensor products of H-graded S-modules are still H-graded S-modules. If we have

$$M = \bigoplus_{h \in H} M_h \qquad \text{and} \qquad N = \bigoplus_{h \in H} N_h,$$

then the direct sum can be written as

$$M \oplus N = \bigoplus_{h \in H} M_h \oplus N_h,$$

so an H-grading can be given by $(M \oplus N)_h = M_h \oplus N_h$. Also, the tensor product can be written as

$$M \otimes N = \bigoplus_{h \in H} \left(\bigoplus_{gf = h} M_g \otimes N_f \right),$$

and an H-grading can be given by

$$(M \otimes N)_h = \bigoplus_{gf = h} M_g \otimes N_f.$$

Definition 5. If $M = \bigoplus_{h \in H} M_h$ is an H-graded S-module, the *graded dimension* of M is

$$\operatorname{rdim}(M) = \sum_{h \in H} h \cdot \operatorname{rank}(M_h).$$

This is a (possibly infinite) formal sum of group elements with coefficients in $\mathbb{Z}$.

Definition 6. If M is an H-graded S-module, we can *shift the grading* of M by an element $g \in H$ and obtain a new H-graded S-module $M\{g\}$. The underlying module structure is the same, but if $a \in M$ has degree h, then the same element $a \in M\{g\}$ has degree hg.

Definition 7. A *cochain complex* of H-graded S-modules is a sequence $C = \left(C^i\right)_{i\in\mathbb{Z}}$ of H-graded S-modules along with differentials $d^i : C^i \to C^{i+1}$ such that $d^{i+1} \circ d^i = 0$ for all $i \in \mathbb{Z}$. We say that the differentials are *degree-preserving* if $\deg(d^i(a)) = h$ whenever $\deg(a) = h$.

Definition 8. The *cohomology sequence* of a chain complex (C, d) of H-graded S-modules is $\mathcal{H}(C) = (\mathcal{H}^i)_{i\in\mathbb{Z}}$, where

$$\mathcal{H}^i = \operatorname{im}(d^{i-1})/\ker(d^i).$$

Note that each $\mathcal{H}^i$ is again an H-graded S-module.

Definition 9. If $C = (C^i)_{i\in\mathbb{Z}}$ is a sequence of H-graded S-modules, the *graded Euler characteristic* of C is

$$\chi(C) = \sum_{i\in\mathbb{Z}} (-1)^i \operatorname{rdim}(C^i).$$

Note that if the differentials of a chain complex C are degree-preserving, then the Euler characteristic of the cohomology sequence is the same as the Euler characteristic of the original chain complex: $\chi(C) = \chi(\mathcal{H}(C))$.

Definition 10. If $C = (C^i)_{i\in\mathbb{Z}}$ is a cochain complex of H-graded S-modules, we can *shift the index* of C by an integer j and obtain a new cochain complex $C[j]$. Again, the underlying complex structure is the same aside from this shift, we simply have $C[j]^i = C^{i-j}$, and the differential maps are shifted accordingly.

Note that we will also use the notation $C\{g\}$ to denote a cochain complex derived from C by shifting the internal grading of each H-graded S-module in C by the element $g \in H$. So $\{\cdot\}$ always corresponds to a shift of the H-grading, and $[\cdot]$ always corresponds to a shift of the index of the complex.

3.2 The Ring S and the Algebra M

Throughout the rest of the paper, let X be a fixed biquandle with a distinguished element x_0, and let R be a fixed commutative ring. Also let $\beta = (A, B)$ be a fixed X-bracket taking values in R, which will be the basis for our construction.

For each $x, y \in X$, let

$$q_{x,y} = -\frac{B_{x,y}}{A_{x,y}},$$

and let $q = q_{x_0,x_0}$. Let G be the group generated by the elements $q_{x,y}^{-1}q$ of $R^\times$:

$$G = \left\langle \frac{q}{q_{x,y}} : x, y \in X \right\rangle \leq R^\times.$$

Finally, let S be the $R^\times$ graded group algebra $\mathbb{Z}[G]$, with the $R^\times$-grading given by $\deg(g) = g$ for all $g \in G$.

Now let M be the $R^\times$-graded S-module $S[t]/(t^2)$ with the additional grading given by $\deg(1) = q$ and $\deg(t) = q^{-1}$. This means that, for example, the element $gt \in M$ has degree gq^{-1}, while the element $g \in M$ has degree gq. M is a Frobenius algebra with the following multiplication and comultiplication operations:

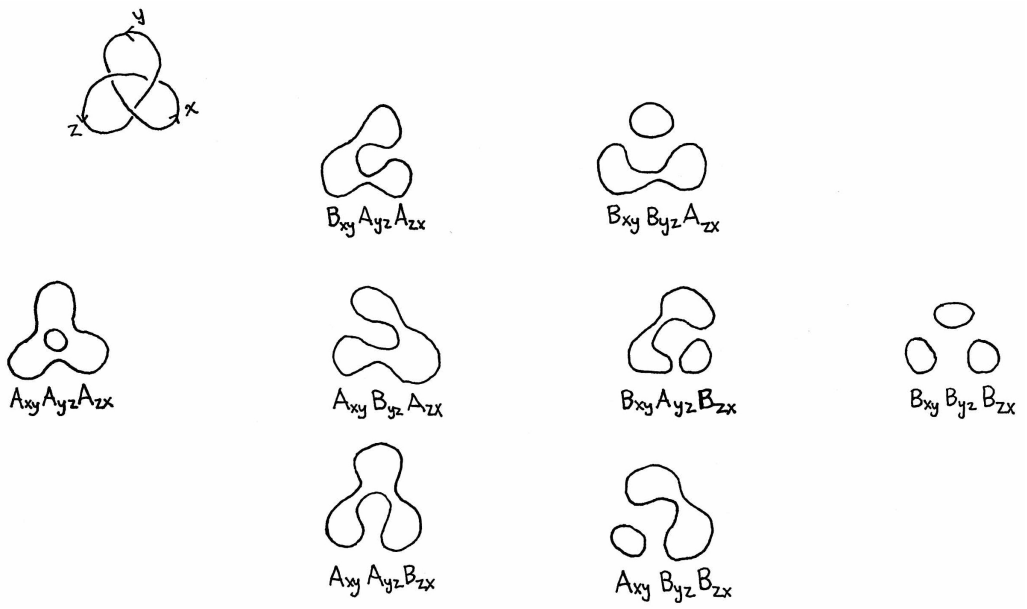
$$\begin{aligned} &m : M \otimes M \to M \\ &m : 1 \otimes 1 \mapsto 1, \qquad 1 \otimes t \mapsto t, \\ &\quad t \otimes 1 \mapsto t, \qquad t \otimes t \mapsto 0. \end{aligned}$$

$$\begin{aligned} &\Delta : M \to M \otimes M \\ &\Delta : 1 \mapsto 1 \otimes t + t \otimes 1, \qquad t \mapsto t \otimes t. \end{aligned}$$

Both of these operations are "degree-lowering," in the sense that the degree of the image of an element is q^{-1} times the degree of the element in the domain.

3.3 Construction of the Cohomology Invariant

We are now ready to present the construction of the link invariant. Suppose f is an X-coloring of an oriented link L. Perform smoothings as in the construction of the biquandle bracket link invariant (recall the image from example 1, which has been replicated below):



There will be 2^n smoothings in total, where n is the number of crossings in the link diagram. We arrange these smoothings as the vertices of an n-dimensional cube, ordered from left to right by the number of B-type splittings in the smoothing.

Now replace each circle in each smoothing with a copy of M and tensor adjacent copies together. Shift the $R^\times$-grading of each resultant S-module by the coefficient extracted by the smoothing, as follows:

$$\overset{M}{\{-B_{x,y}A_{y,z}A_{z,x}\}} \qquad \overset{M\otimes M}{\{B_{x,y}B_{y,z}A_{z,x}\}}$$

$$\overset{M\otimes M}{\{A_{x,y}A_{y,z}A_{z,x}\}} \qquad \overset{M}{\{-A_{x,y}B_{y,z}A_{z,x}\}} \qquad \overset{M\otimes M}{\{B_{x,y}A_{y,z}B_{z,x}\}} \qquad \overset{M\otimes M\otimes M}{\{-B_{x,y}B_{y,z}B_{z,x}\}}$$

$$\overset{M}{\{-A_{x,y}A_{y,z}B_{z,x}\}} \qquad \overset{M\otimes M}{\{A_{x,y}B_{y,z}B_{z,x}\}}$$

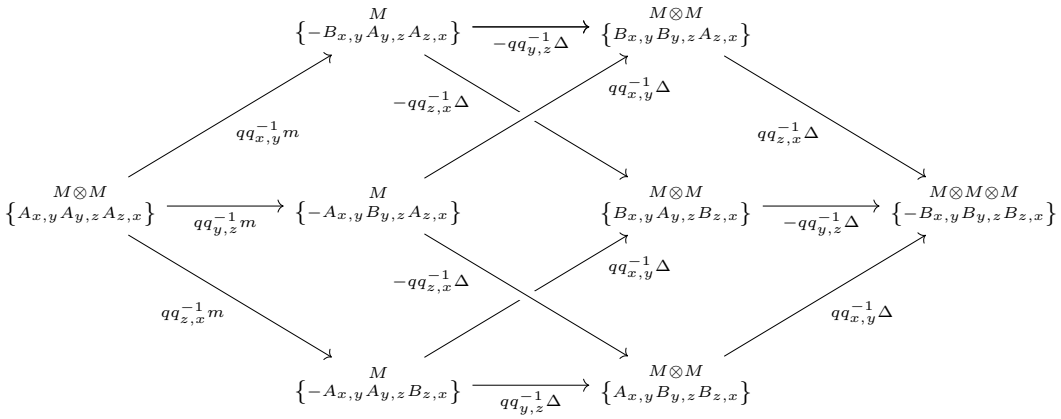
Next, we add maps in between the modules to form a cube. The maps are derived from the Frobenius algebra structure on M, which mimics the data of a topological quantum field theory. Thus, if we'd like to form a map $M \otimes M \to M$, we'll use the multiplication map m, and if we'd like to form a map $M \to M \otimes M$, we'll use the comultiplication map Δ.

The multiplication and comultiplication maps should be applied to the tensor factors involved in a cobordism between the two images that the modules correspond to. For example, the upper rightmost map in the following diagram is applied to the tensor factors corresponding with the cobordism taking the circle

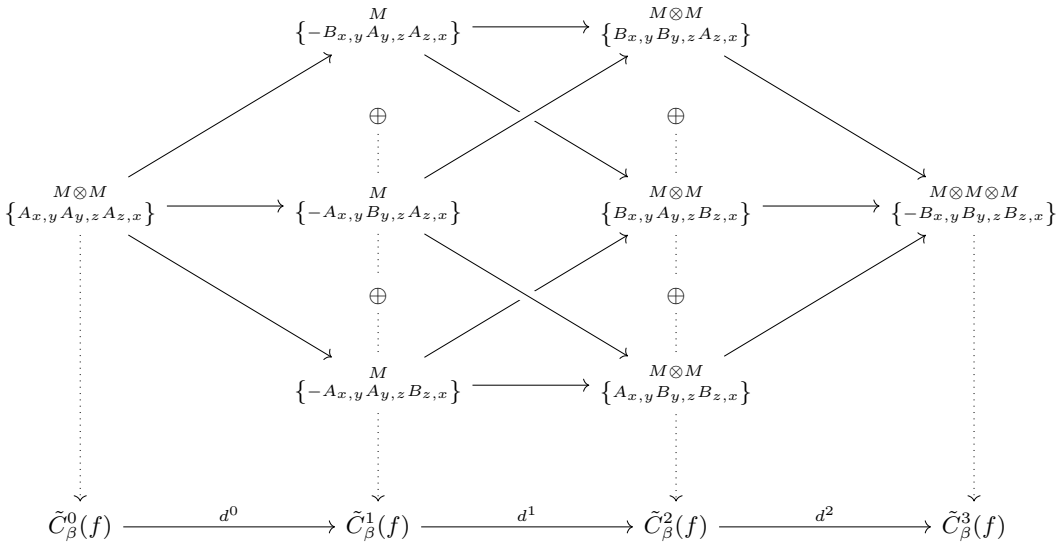
at the bottom of the top-right smoothing to the pair of circles at the bottom of the right-most smoothing of the link diagram.

Each map must be augmented by multiplication by a particular element of G so that the end result will be a cochain complex with degree-preserving differentials. The element of G to be multiplied is simply the quotient of the shift applied to the target space over the shift applied to the domain.

This yields a cube with commutative faces. But we will be summing the modules along the columns, and we'd like a cochain complex, so we need the faces of the cube to be *anti*-commutative. This is why some of the maps in the following image have minus signs. For a detailed explanation of the position of the minus signs, see Section 3.2 of [1].



Now take the direct sum of the modules and maps along the columns to obtain a sequence $\tilde{C}_\beta(f)$ of $R^\times$-graded H-modules with maps between them. This in fact constitutes a cochain complex because the faces of the cube are anti-commutative. Each element in each module will have two images in each target space under the double-differential map, corresponding to the two different ways to get around the square between the domain and target. And these images will cancel out, leading to $d^{i+1}\circ d^i = 0$.



Now we must apply a shift to account for the writhe of the original link. Let n_+ be the number of positive crossings in L and let n_- be the number of negative crossings. Recall that $w = \delta A_{x_0,x_0} + B_{x_0,x_0} \in R^\times$. So define $C_\beta(f)$ to be the following shifted cochain complex:

$$C_\beta(f) = \tilde{C}_\beta(f)[n_-]\{(-1)^{n_-}w^{-n_+}w^{n_-}\}.$$

Finally, take the cohomology of $C_\beta(f)$ to obtain a sequence $\mathcal{H}_\beta(f) = \mathcal{H}(C_\beta(f))$ of $R^\times$-graded S-modules. This is an invariant of X-colored oriented links f. If we want to start with an uncolored oriented link we must consider all X-colorings simultaneously as follows:

$$\mathrm{Bh}_\beta(L) = \{\mathcal{H}_\beta(f) : f \text{ is a valid } X\text{-coloring of } L\}.$$

The quantity $\mathrm{Bh}_\beta(L)$ is an invariant of oriented links L. This will be proved in the next section.

Example 5. When we take β to be any constant biquandle bracket as in example 3, we have $q_{x,y} = q$ for all $x, y \in X$. Suppose we also started with $R = \mathbb{Z}[t, t^{-1}]$ and $q = t$ so that the biquandle bracket value is the same as the Jones polynomial. Thus, as can be seen by following Khovanov's construction [6] or in Bar-Natan's paper [1], $\mathcal{H}_\beta(f)$ is isomorphic to the Khovanov homology invariant of L, denoted $\mathrm{Kh}(L)$.

Now, if instead of $\mathbb{Z}[t, t^{-1}]$ we had started with some other ring R, the biquandle bracket value is the Jones polynomial *evaluated* at q. If q satisfies some nontrivial algebraic relations in R, then we will not recover the full $\mathrm{Kh}(L)$. Instead, we will have to quotient out each module in the $\mathrm{Kh}(L)$ by the relations satisfied by q. Since $\mathrm{Kh}(L)$ is an invariant of links L, the object we obtain will still be invariant of links.

Also, since β is a constant biquandle bracket, $\mathcal{H}_\beta(f)$ will not depend on the coloring f of L, only the link itself. So $\mathrm{Bh}_\beta(L)$ is a multiset containing a quotient of $\mathrm{Kh}(L)$ with multiplicity equal to the number of X-colorings of L. This shows that $\mathrm{Bh}_\beta(L)$ is an invariant of links when β is a constant biquandle bracket.

Remark 4. $\mathrm{Bh}_\beta(L)$ is *not* a true categorification of the biquandle bracket invariant $\Phi_X^\beta(L)$ in the sense that $\mathrm{Kh}(L)$ is a categorification of the Jones polynomial. This is because the Euler characteristic of $\mathrm{Bh}_\beta(L)$ is actually $\mathrm{rdim}(S) \cdot \Phi_X^\beta(L)$. The biquandle bracket invariant $\Phi_X^\beta(L)$ can be considered as the "Euler characteristic *over* S," but when actually computing the Euler characteristic of the graded homology sequence, the grading of S itself cannot be ignored. Note that $\mathrm{rdim}(S)$ is a formal sum of the elements of G.

For some particular biquandle brackets, $\mathrm{rdim}(S)$ is an invertible element of R, and in these cases one can recover $\Phi_X^\beta(L)$ from $\mathrm{Bh}_\beta(L)$. For example, in the situation in the first part of example 5 above, G is the trivial group since $q = q_{x,y}$ for all $x, y \in X$. Thus $\mathrm{rdim}(S) = 1$. This is unsurprising since in this situation we exactly recover Khovanov homology as a true categorification of the Jones polynomial.

Examples of biquandle brackets which lead to $\mathrm{rdim}(S)$ being a non-invertible element of R may be found in [4]. In these situations, one can still ask how much information about the biquandle bracket is retained in the Euler characteristic of $\mathrm{Bh}_\beta(L)$. These questions are strongly related to questions about the canonical biquandle 2-cocycle associated with a biquandle bracket, which will be discussed in the next section.

3.4 Proof of Invariance and the Canonical 2-Cocycle

To prove that $\mathrm{Bh}_\beta(L)$ is an invariant of oriented links L it is sufficient to prove that $\mathcal{H}_\beta(f)$ is an invariant of X-colored oriented links f. For this, we will actually prove that $\mathcal{H}_\beta(f)$ is isomorphic to a shift of a quotient of $\mathrm{Kh}(L)$, the original Khovanov homology invariant of L. The shift itself turns out to be an invariant of X-colored links, and it is obtained from a particular biquandle 2-cocycle. In this way, it becomes possible to canonically assign a biquandle 2-cocycle to an biquandle bracket. First we need a lemma.

Lemma 1. *Let G be an abelian group and let M be a $\mathbb{Z}[G]$-module graded by some group $H \geq G$ such that $\deg(gm) = g\deg(m)$ for all $m \in M$ and $g \in G$. Then $M \cong gM$ as H-graded $\mathbb{Z}[G]$-modules for all $g \in G$.*

Proof. Suppose $M = \bigoplus_{h\in H} M_h$. Then the "multiplication-by-g" map is an isomorphism $M_h \to M_{gh}$, since the "multiplication-by-g^{-1}" map is the inverse. Then

$$gM = \bigoplus_{h\in H} gM_h = \bigoplus_{h\in H} M_{gh} \cong \bigoplus_{h'\in H} M_{h'} = M,$$

since $gH = H$ because H is a group. ∎

Let $\mathrm{Kh}_\beta(L)$ denote the quotient of Khovanov homology described in example 5 above. Both $\mathcal{H}_\beta(f)$ and $\mathrm{Kh}_\beta(L)$ are sequences of homologies of cochain complexes. The claim is now that $\mathcal{H}_\beta(f) \cong \mathrm{Kh}_\beta(L)\{Z_\beta(f)\}$, where $Z_\beta(f)$ is an invariant of X-colored links f. So to see that $\mathcal{H}_\beta(f)$ and $\mathrm{Kh}_\beta(L)\{Z_\beta(f)\}$ are isomorphic,

it suffices to construct an isomorphism of the cochain complexes they are derived from. And, since both cochain complexes are direct sums of cube diagrams, it suffices to construct an isomorphism of cubes, by which we mean a collection of isomorphisms from each vertex in one cube to the corresponding vertex of the other cube such that the squares thus adjoined to each edge commute. We henceforth construct the isomorphism of the cube used to construct $\mathcal{H}_\beta(f)$ with the cube used to construct $\mathrm{Kh}_\beta(L)$ (before the final writhe-correcting shift, which will be the same for both complexes).

Starting with the cube in the construction of $\mathcal{H}_\beta(f)$, shift all of the $R^\times$-gradings by the grading shift applied to the left-most vertex (in the example above, this would be $A_{x,y}A_{y,z}A_{z,x}$). This yields a cube where the left-most vertex is a tensor power of M with no shift applied, and each other vertex will be a tensor power of M with a shift that can be written as a product of the elements $q_{x,y} \in R^\times$. The maps in the cube are unchanged by this shift, since the coefficient on a map is the quotient of the shifts on the target and domain spaces.

For each vertex $M^{\otimes k}\{\prod_{i=1}^{j} q_{x_i,y_i}\}$ of the shifted cube, the "multiplication-by-$(\prod_{i=1}^{j} q^{-1}q_{x_i,y_i})$" map is an isomorphism $M^{\otimes k}\{\prod q_{x,y}\} \to M^{\otimes k}\{q^j\}$ by lemma 1. And these maps assemble into an isomorphism of cubes because each edge in the cube is a map of the form $q_{z,w}^{-1} q\,\Delta$ or $q_{z,w}^{-1} q\,m$, and the following two squares commute:

$$\begin{array}{ccc} M^{\otimes k}\{\prod_{i=1}^{j} q_{x_i,y_i}\} & \xrightarrow{q_{z,w}^{-1} q\,\Delta} & M^{\otimes (k+1)}\{q_{z,w}\prod_{i=1}^{j} q_{x_i,y_i}\} \\ \Big\downarrow{\scriptstyle \text{multiplication-by-}\left(\prod_{i=1}^{j} q^{-1}q_{x_i,y_i}\right)} & & \Big\downarrow{\scriptstyle \text{multiplication-by-}\left(q^{-1}q_{z,w}\prod_{i=1}^{j} q^{-1}q_{x_i,y_i}\right)} \\ M^{\otimes k}\{q^j\} & \xrightarrow{\Delta} & M^{\otimes (k+1)}\{q^{j+1}\} \end{array}$$

$$\begin{array}{ccc} M^{\otimes k}\{\prod_{i=1}^{j} q_{x_i,y_i}\} & \xrightarrow{q_{z,w}^{-1} q\,m} & M^{\otimes (k-1)}\{q_{z,w}\prod_{i=1}^{j} q_{x_i,y_i}\} \\ \Big\downarrow{\scriptstyle \text{multiplication-by-}\left(\prod_{i=1}^{j} q^{-1}q_{x_i,y_i}\right)} & & \Big\downarrow{\scriptstyle \text{multiplication-by-}\left(q^{-1}q_{z,w}\prod_{i=1}^{j} q^{-1}q_{x_i,y_i}\right)} \\ M^{\otimes k}\{q^j\} & \xrightarrow{m} & M^{\otimes (k-1)}\{q^{j+1}\} \end{array}$$

Thus we have an isomorphism from the shifted cube to a cube with only powers of q as shifts and with no coefficients on the maps between vertices. This is almost exactly the cube in the construction of $\mathrm{Kh}(L)$. However, as noted before, this q may satisfy some algebraic relation in R and so it is really a replica of the quotient $\mathrm{Kh}_\beta(L)$. Additionally, the cube for $\mathrm{Kh}(L)$ contains modules defined over a ring that has trivial grading, whereas the cube we have now contains modules defined over S, which may have nontrivial $R^\times$-grading (see remark 4 above).

Thus, what we truly obtain is an isomorphism between $\mathcal{H}_\beta(f)$ and $\mathrm{Kh}_\beta(L)$ shifted by the grading shift applied to the left-most vertex in the original cube (again, this would be $A_{x,y}A_{y,z}A_{z,x}$ in the above example) *and* shifted by $\mathrm{rdim}(S)$. The shift applied to the left-most vertex in the original cube turns out to be

$$\left(\prod_{\tau^+} A_{x,y}\right)\left(\prod_{\tau^-}(-B_{x,y}^{-1})\right),$$

where the first product is taken over all positive crossings τ^+ in the link and the second product is taken over all negative crossings τ^-, and (x, y) is the X-coloring at the crossing in question. Multiplying by the writhe correction shift and $\mathrm{rdim}(S)$, we find that

$$Z_\beta(f) = \left(\prod_{\tau^+} A_{x,y}A_{x_0,x_0}^{-1}\right)\left(\prod_{\tau^-} B_{x,y}^{-1}B_{x_0,x_0}\right)\cdot \mathrm{rdim}(S)$$

is the correct shift to obtain $\mathcal{H}_\beta(f) \cong \mathrm{Kh}_\beta(L)\{Z_\beta(f)\}$.

Since all of the factors in the two products defining $Z_\beta(f)$ are invertible elements of R, and since $\mathrm{rdim}(S)$ is a formal sum of all elements in $G \leq R^\times$, we can view this shift as taking values in the abelian quotient

group $R^\times/G$. Now it becomes clear that $Z_\beta(f)$ is in fact the value of a biquandle 2-cocycle invariant on f; the associated 2-cocycle is $\phi_\beta : X \times X \to R^\times/G$, defined by $\phi_\beta(x,y) = A_{x,y}A^{-1}_{x_0,x_0} \cdot G$. It is clear that on positive crossings τ^+, the value of $Z_\beta(f)$ is derived from ϕ_β. For negative crossings, we observe that

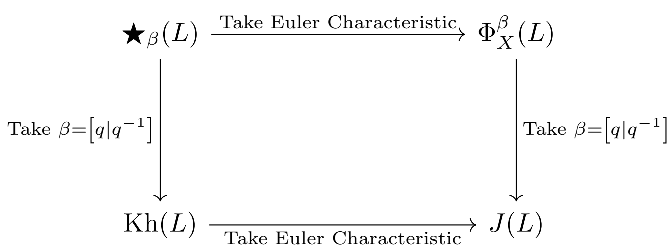
$$A_{x,y}A^{-1}_{x_0,x_0}B^{-1}_{x,y}B_{x_0,x_0} = q^{-1}_{x,y}q \in G$$

so that $B^{-1}_{x,y}B_{x_0,x_0}$ is the inverse of $A_{x,y}A^{-1}_{x_0,x_0}$, mod G. The biquandle 2-cocycle ϕ_β is called the *canonical 2-cocycle* associated with a biquandle bracket β.

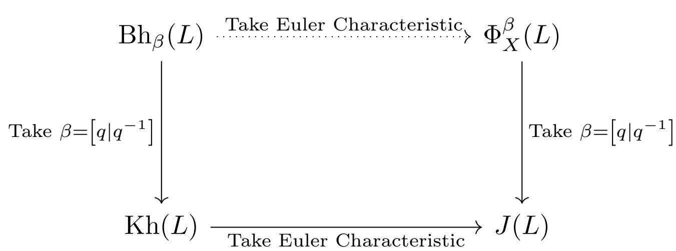
Thus $Z_\beta(f)$ is an invariant of X-colored links f. Since $\mathrm{Kh}_\beta(L)$ is an invariant of oriented links L, this shows that $\mathcal{H}_\beta(f)$ is an invariant of X-colored links f. Therefore, the biquandle homology multiset, $\mathrm{Bh}_\beta(L)$, is in fact an invariant of oriented uncolored links L.

3.5 Conclusions and Further Questions

The biquandle bracket generalizes the Jones polynomial $J(L)$, which Khovanov homology categorifies. Our goal was to generalize Khovanov homology to a categorification of biquandle brackets and obtain the invariant ★ in the following diagram:



However, the top arrow in this diagram fails to hold in general with our invariant $\mathrm{Bh}_\beta(L)$:



This opens up a few questions.

First, which biquandle brackets *are* truly categorified by this construction? This question is strongly related to the properties of the group G derived from the biquandle bracket β. Some progress towards understanding this group is outlined in [4].

Second, how does the invariant $\mathrm{Bh}_\beta(L)$ compare in power to the biquandle bracket invariant $\Phi_X^\beta(L)$? Specifically, we can ask how the two separate pieces, $\mathrm{Kh}_\beta(L)$ (the quotient of Khovanov homology) and $Z_\beta(L)$ (the invariant obtained from the canonical 2-cocycle associated to β) compare in power to $\Phi_X^\beta(L)$.

As far as $\mathrm{Kh}_\beta(L)$ goes, it is simply weaker than the previously-known $\mathrm{Kh}(L)$. And the invariant $Z_\beta(L)$ is actually easier to compute than $\mathrm{Bh}_\beta(L)$. So the most important comparison to make is between $Z_\beta(L)$ and $\Phi_X^\beta(L)$. Computer testing using mathematica packages (to be discussed in chapter 4) has exhibited examples of knots which are distinguished by $\Phi_X^\beta(L)$ but *not* by $Z_\beta(L)$. So far no knots have been found to be distinguished by $Z_\beta(L)$ but not by $\Phi_X^\beta(L)$, and so we conjecture that $Z_\beta(L)$ is weaker than $\Phi_X^\beta(L)$, although this has not been proven.

Finally, does the invariant ★ exist? The construction we have outlined seems (to the authors) to be the most natural step away from Khovanov homology toward a categorification of biquandle brackets, but technical limitations prevent the construction from truly categorifying all biquandle brackets. Is it possible, with more advanced techniques, to create an invariant that simultaneously categorifies every biquandle bracket and generalizes Khovanov homology?

Chapter 4

Mathematica Packages

In this section, we present documentation for the Mathematica packages developed by the authors to be used for doing computations with biquandles, biquandle brackets, and biquandle 2-cocycles, including the canonical biquandle 2-cocycle associated with a biquandle bracket. These packages have been used to find examples of biquandles and biquandle brackets in [4], and can be used to perform more experiments with these mathematical objects.

The full source code for all Mathematica packages discussed here can be found in appendix A, and the packages can also be downloaded from vilas.us/biquandles.

4.1 Biquandles

This package contains functions used for computations on finite biquandles. Following is a list of visible functions from the `Biquandles.wl` package, along with a description of each one.

`Biquandle` is the data type for a biquandle. It contains an integer, which represents the number of elements in the biquandle, and it contains two matrices, which represent the two operations on the biquandle, as in remark 1. Note that Mathematica does not do any sort of type-checking, so one can put in any matrices and will obtain a "valid" biquandle object, although it may not actually represent a biquandle.

`BiquandleQ` is a function that checks the biquandle axioms. This is how we ensure that the `Biquandle` object we are working with does actually represent a biquandle. This function, when passed a `Biquandle` object, will return either `True`, if the object's matrices satisfy all of the biquandle axioms, or `False` otherwise. To check each of the three biquandle axioms separately, one can use the `BiquandleAxiom1Q`, `BiquandleAxiom2Q`, and `BiquandleAxiom3Q` functions.

`UnTri` represents the $\underline{\triangleright}$ operation on biquandles. When passed a `Biquandle` object and two integers, it will perform the $\underline{\triangleright}$ operation on the biquandle elements represented by the two integers and return the integer representing the result. Similarly, `OvTri` represents the $\overline{\triangleright}$ operation.

`UnTriInverse` represents the inverse operation $\underline{\triangleright}^{-1}$. The notation $x \,\underline{\triangleright}^{-1}\, y$ really means $\beta_y^{-1}(x)$, where $\beta_y(x) = x \,\underline{\triangleright}\, y$ as in axiom (ii) of definition 1 (this inverse function is guaranteed to exist by that biquandle axiom). Similarly, `OvTriInverse` represents $\overline{\triangleright}^{-1}$.

`BiquandleElements` returns a list of the elements in a particular `Biquandle` object. At the time of writing, this will always be a list of integers of the form $\{1, 2, \ldots, n\}$, where n is the size of the biquandle. However, if in the future there are more different ways to represent a biquandle in the system, this function will be overriden to be used to obtain an iterable object containing all elements of a given biquandle.

`BiquandleColoringQ` is a function used to determine whether a given assignment of biquandle elements to strands in a link diagram is a valid biquandle coloring. This requires some explanation of how knots and links are represented in Mathematica using the `KnotTheory` package.

For the purposes of these packages, all knots and links should be represented by planar diagrams, which are just lists of symbols that can be read by a computer. In a planar diagram of a link, a the strands of a link are labeled by integers, increasing as the knot is traversed. The crossings in the link are then represented by symbols that include data about which strands are involved in the crossing and in which order. See [7] for more information about planar diagrams.

The function `BiquandleColoringQ` takes as input a planar diagram of a link, a `Biquandle` object, and a list of elements of the biquandle. The interpretation of this is that the strand labeled with the integer i will be colored by the biquandle element in position i of the given list. The function will return `True` if the given coloring is valid (i.e. the crossing relations are satisfied at each crossing) and `False` otherwise.

`AllBiquandleColorings` is a function that takes as input a planar diagram of a link and a `Biquandle` object, and returns a list of all valid colorings of the link by the biquandle. These colorings are represented by lists of colors as in the input for `BiquandleColoringQ`.

`BiquandleCountingInvariant` takes a planar diagram of a link L and a `Biquandle` object representing a biquandle X, and returns the biquandle counting invariant $\Phi_X^{\mathbb{Z}}(L)$. This function simply just takes the output of `AllBiquandleColorings` and computes the cardinality.

4.2 Biquandle Brackets

This package contains functions used for computations with biquandle brackets over finite biquandles, taking values in a certain class of commutative rings. Following is a list of visible functions from the `BiquandleBrackets.wl` package, along with a description of each one.

`BiquandleBracket` is the data type for a biquandle bracket. It contains a `Biquandle` object, which the biquandle bracket is defined over. It also contains data that represents the ring R where the biquandle bracket takes values. The rings that can be represented in the framework of this Mathematica package are all quotients of the rings of polynomials over $\mathbb{Q}$ or any finite field of prime order. Additionally, $\mathbb{Z}/n\mathbb{Z}$ can be used, for any integer n.

These rings are represented in the `BiquandleBracket` object by a list, called `Ideal`, of polynomials which serve as generators for an ideal I in a ring of polynomials over $\mathbb{Q}$. For example, if `Ideal = { x^2 - 2 }`, then the ring represented is $\mathbb{Q}[x]/(x^2-2) \cong \mathbb{Q}[\sqrt{2}]$. To obtain a quotient of a ring of polynomials over a finite field $\mathbb{F}_p$ of prime order, one simply includes the prime integer p in the list `Ideal`. For example, if we have `Ideal = { 7, 2 x^4 + 3 y, z^8 - 2 }`, then the ring represented is $\mathbb{F}_7[x,y,z]/(2x^4+3y, z^8-2)$. Finally, to represent $\mathbb{Z}/n\mathbb{Z}$, the `Ideal` list must contain only the integer n. For some reason, Mathematica has trouble with quotients of polynomial rings over rings that are not fields, and the authors have not been able to represent rings like $(\mathbb{Z}/4\mathbb{Z})[x,y]/(x^2-y^3)$.

The `BiquandleBracket` data type also contains two matrices represenging A and B as in remark 2, and the elements δ and w detailed in definition 2. There is also a constructor for the `BiquandleBracket` data type that doesn't require the user to input δ and w, since these two elements can be determined from the matrices A and B.

`BiquandleBracketQ` is used to check that a given `BiquandleBracket` object does in fact represent a valid biquandle bracket. It checks that the elements in the A and B matrices are invertible elements of the ring specified by `Ideal`, and it also checks the three biquandle bracket axioms, which are accessible separately as `BiquandleBracketAxiom1Q`, `BiquandleBracketAxiom2Q`, and `BiquandleBracketAxiom3Q`. The function `BiquandleBracketQ` returns `True` if all of these checks pass, and `False` otherwise.

`GetBiquandle` takes a `BiquandleBracket` object and returns the underlying `Biquandle` object that the represented biquandle bracket is defined over.

`BiquandleBracketColoringValue` takes as input a `BiquandleBracket` object representing a biquandle bracket β, a planar diagram representing a link, and a list representing a coloring f of the link. It returns the value $\beta(f)$ of the biquandle bracket invariant on the coloring.

`BiquandleBracketValue` takes as input a `BiquandleBracket` object representing a biquandle bracket β defined over a biquandle X, and a planar diagram representing a link L. It returns the multiset $\Phi_X^\beta(L)$.

4.3 Biquandle 2-Cocycles

This package contains functions used for computations with biquandle 2-cocycles over finite biquandles taking values in certain abelian groups that are written multiplicatively. Following is a list of visible functions from the `Biquandle2Cocycles.wl` package, along with a description of each one.

`Biquandle2Cocycle` is the data type for a biquandle 2-cocycle. It contains a `Biquandle` object and a matrix representing the 2-cocycle as in remark 3.

`Biquandle2CocycleQ` checks that the given `Biquandle2Cocycle` object actually represents a biquandle 2-cocycle. It does this by checking the two axioms, which are accessible as `Biquandle2CocycleAxiom1Q` and `Biquandle2CocycleAxiom2Q`. The function `Biquandle2CocycleQ` returns `True` if the axioms hold and `False` otherwise.

`Biquandle2CocycleInvariantValue` takes as input a planar diagram representing a link L, as well as a `Biquandle2Cocycle` object representing a biquandle 2-cocycle ϕ. The function returns the value of the biquandle 2-cocycle invariant associated with ϕ on the link L.

4.4 The Canonical Biquandle 2-Cocycle

This package contains functions used for computations with the canonical biquandle 2-cocycle associated with a biquandle bracket. This is a list of visible functions from the `BiquandleBracketCanonical2Cocycle.wl` package, along with a description of each one.

`BiquandleBracketCanonical2Cocycle` takes as input a `BiquandleBracket` object representing a biquandle bracket β, and returns the canonical biquandle 2-cocycle ϕ_β associated with β. Note that this biquandle 2-cocycle will *not* necessarily pass the checks in the `Biquandle2CocycleQ` function, since ϕ_β takes values in a quotient group and only returns representatives for the classes in the group. The `Biquandle2CocycleQ` function only checks for raw equality, and doesn't have information about the quotient group structure of the target of ϕ_β. Nevertheless, as long as the `BiquandleBracket` represents a valid biquandle bracket, this function will return a valid biquandle 2-cocycle, although checking the equality of values of the invariant associated with ϕ_β will rely on a helper function discussed below.

`BiquandleBracketKernel` takes as input a `BiquandleBracket` object representing a biquandle bracket β, and returns a list of generators of the group G associated with β (as discussed in section 3.2).

`MultisetQuotientEqualityQ` is a function that takes as input two multisets of elements in a quotient group, as well as the list of generators for the kernel of the projection associated with the quotient group. This function returns `True` if the two multisets are the same, when their elements are considered modulo the given kernel, and `False` otherwise.

Appendix A

Source Code

Following is the full source code for all Mathematica packages used and described in this paper. The files may be downloaded directly from vilas.us/biquandles, and that site will contain the most up-to-date versions.

A.1 Biquandles.wl

```
(* Author: Vilas Winstein *)

BeginPackage["Biquandles`"]
Needs["KnotTheory`"]

Biquandle::usage="The symbol type for a biquandle.";
BiquandleAxiom1Q::usage="Returns true if the given Biquandle symbol satisfies biquandle axiom 1, false
    otherwise.";
BiquandleAxiom2Q::usage="Returns true if the given Biquandle symbol satisfies biquandle axiom 2, false
    otherwise.";
BiquandleAxiom3Q::usage="Returns true if the given Biquandle symbol satisfies biquandle axiom 3, false
    otherwise.";
BiquandleQ::usage="Returns true if the given Biquandle symbol satisfies the biquandle axioms, false otherwise
    . Use the Verbose option to see which axiom fails.";
UnTri::usage="Apply the 'Underlined Triangle' operation of a given biquandle to two elements.";
OvTri::usage="Apply the 'Overlined Triangle' operation of a given biquandle to two elements.";
UnTriInverse::usage="Apply the inverse of the 'Underlined Triangle' operation of a given biquandle to two
    elements.";
UnTriInverse::usage="Apply the inverse of the 'Overlined Triangle' operation of a given biquandle to two
    elements.";
BiquandleElements::usage="Get a list of the elements of a particular biquandle.";
BiquandleColoringQ::usage="Returns true if the given coloring (with the given biquandle) is valid for the
    given knot Planar Diagram.";
AllBiquandleColorings::usage="Returns a list of all valid colorings of a given link with a given biquandle.";
BiquandleCountingInvariant::usage="Returns the biquandle counting invariant of the given link with the given
    biquandle.";

Begin["Private`"]

(* How to declare a biquandle *)
Biquandle[n_Integer,untri_List,ovtri_List];

(* Checks that the shape of operation matrices are correct *)
BiquandleMatrixShapeQ[Biquandle[n_,u_,o_]] := Dimensions[u]=={n,n}&&Dimensions[o]=={n,n};

(* Biquandle Axiom 1 *)
BiquandleAxiom1Q[Biquandle[n_,u_,o_]] := AllTrue[Range[n],Function[x, u[[x,x]]==o[[x,x]]]];

(* Biquandle Axiom 2 *)
BiquandleAxiom2Q[Biquandle[n_,u_,o_]] :=
```

```
        Catch[
                If[Not[AllTrue[Range[n],Function[y, Sort[u[[All,y]]]==Range[n] && Sort[o[[All,y]]]==Range[n
                    ]]]],
                        Throw[False]
                ];
                If[Not[Sort[Thread[List[Flatten[Transpose[o]],Flatten[u]]]]==Tuples[Range[n],2]],
                        Throw[False]
                ];
                Throw[True];
        ];

(* Helper function which tests the exchange laws for a particular triple of biquandle elements *)
ExchangeLawsQ[u_,o_,List[x_,y_,z_]] :=
        Catch[
                Module[{law1, law2, law3},
                        law1 = u[[u[[x,y]],u[[z,y]]]]==u[[u[[x,z]],o[[y,z]]]];
                        law2 = o[[u[[x,y]],u[[z,y]]]]==u[[o[[x,z]],o[[y,z]]]];
                        law3 = o[[o[[x,y]],o[[z,y]]]]==o[[o[[x,z]],u[[y,z]]]];
                        Throw[law1&&law2&&law3];
                ];
        ];

(* Biquandle Axiom 3 *)
BiquandleAxiom3Q[Biquandle[n_,u_,o_]] :=
        Catch[
                Do[
                        If[Not[ExchangeLawsQ[u,o,T]], Throw[False]],
                        {T, Tuples[Range[n],3]}
                ];
                Throw[True];
        ];

(* A list of all biquandle axioms *)
BiquandleAxioms = {BiquandleAxiom1Q, BiquandleAxiom2Q, BiquandleAxiom3Q};

(* Function that checks whether a "biquandle" satisfies the biquandle axioms *)
BiquandleQ[X_Biquandle, OptionsPattern[]]:=
        Catch[
                If[Not[BiquandleMatrixShapeQ[X]],
                        If[OptionValue[Verbose], Print["Dimensions of operation matrices are incorrect."]];
                        Throw[False];
                        ];
                For[i=1,i<=3,i++,
                        If[Not[BiquandleAxioms[[i]][X]],
                                If[OptionValue[Verbose], Print["Biquandle fails Axiom " <> ToString[i]]];
                                Throw[False];
                        ];
                ];
                Throw[True];
        ];

(* By default, BiquandleQ doesn't print anything *)
Options[BiquandleQ] = {Verbose -> False};

(* Given a biquandle, perform the operations *)
UnTri[Biquandle[n_,u_,o_],x_Integer,y_Integer]:=u[[x,y]];
OvTri[Biquandle[n_,u_,o_],x_Integer,y_Integer]:=o[[x,y]];

(* Perform the inverse operations *)
UnTriInverse[Biquandle[n_,u_,o_],x_Integer,y_Integer]:=Position[u[[All,y]],x][[1,1]];
OvTriInverse[Biquandle[n_,u_,o_],x_Integer,y_Integer]:=Position[o[[All,y]],x][[1,1]];

(* Get the elements of a biquandle *)
BiquandleElements[Biquandle[n_,u_,o_]]:=Range[n];

(* See if a particular coloring is valid at a given crossing *)
ColoringSatisfiedAtCrossing[X[i_,j_,k_,l_],B_Biquandle,coloring_List]:=
```

```
        If[j-l==1||l-j>1,
                UnTri[B,coloring[[i]],coloring[[j]]]==coloring[[k]]&&OvTri[B,coloring[[j]],coloring[[i]]]==
                     coloring[[l]],
                OvTri[B,coloring[[j]],coloring[[k]]]==coloring[[l]]&&UnTri[B,coloring[[k]],coloring[[j]]]==
                     coloring[[i]]];

ColoringSatisfiedAtCrossing[Xp[i_,j_,k_,l_],B_Biquandle,coloring_List]:=
                UnTri[B,coloring[[i]],coloring[[j]]]==coloring[[k]]&&OvTri[B,coloring[[j]],coloring[[i]]]==
                     coloring[[l]];

ColoringSatisfiedAtCrossing[Xm[i_,j_,k_,l_],B_Biquandle,coloring_List]:=
                OvTri[B,coloring[[j]],coloring[[k]]]==coloring[[l]]&&UnTri[B,coloring[[k]],coloring[[j]]]==
                     coloring[[i]];

BiquandleColoringQ[link_PD,B_Biquandle,coloring_List]:=
        AllTrue[link,ColoringSatisfiedAtCrossing[#,B,coloring]&];

NumberOfEdges[link_PD]:=
        Max[Flatten[Map[List @@ # &, List @@ link]]];

AllBiquandleColorings[link_PD,B_Biquandle]:=
        Select[Tuples[BiquandleElements[B],NumberOfEdges[link]], BiquandleColoringQ[link,B,#]&];

BiquandleCountingInvariant[link_PD,B_Biquandle]:=Length[AllBiquandleColorings[link,B]];

End[]
EndPackage[]
```

A.2 BiquandleBrackets.wl

```
(* Author: Vilas Winstein *)

BeginPackage["BiquandleBrackets`"]
Needs["KnotTheory`"]
Needs["Biquandles`"]

BiquandleBracket::usage="The symbol type for a biquandle bracket.";
BiquandleBracketAxiom1Q::usage="Returns true if the given BiquandleBracket symbol satisfies biquandle bracket
      axiom 1, false otherwise.";
BiquandleBracketAxiom2Q::usage="Returns true if the given BiquandleBracket symbol satisfies biquandle bracket
      axiom 2, false otherwise.";
BiquandleBracketAxiom3Q::usage="Returns true if the given BiquandleBracket symbol satisfies biquandle bracket
      axiom 3, false otherwise.";
BiquandleBracketQ::usage="Returns true if the given BiquandleBracket symbol satisfies the biquandle bracket
     axioms, and in addition takes values in a factor of Z[x,y,z,...], false otherwise. Use the Verbose
     option to see which axiom fails.";
BiquandleBracketColoringValue::usage="Returns the value of the given biquandle bracket on the given link with
      the given biquandle coloring.";
BiquandleBracketValue::usage="Returns the value (a multiset) of the given biquandle bracket on the given link
     .";
GetBiquandle::usage="Get the biquandle used by a particular biquandle bracket.";

Begin["Private`"]

(* How to declare a biquandle bracket *)
BiquandleBracket[X_Biquandle, Ideal_List, A_List, B_List, \[Delta], w];

(* Another biquandle bracket constructor that doesn't require you to specify \[Delta] or w *)
BiquandleBracket[X_Biquandle, Ideal_List, A_List, B_List] :=
        BiquandleBracket[X, Ideal, A, B, \[Delta], PolynomialMod[\[Delta] A[[1,1]]+B[[1,1]], Ideal]] /.\[
             Delta] -> PolynomialMod[-A[[1,1]]/B[[1,1]]-B[[1,1]]/A[[1,1]], Ideal];

(* Function for checking whether two things are equal mod an ideal *)
```

```
PolynomialModEquality[p_, q_, Ideal_] := PolynomialMod[p - q, Ideal] === 0;

(* Biquandle bracket Axiom 1 *)
BiquandleBracketAxiom1Q[BiquandleBracket[X_Biquandle, Ideal_List, A_List, B_List, \[Delta]_, w_]] :=
        Catch[
                Do[
                        If[Not[PolynomialModEquality[\[Delta] A[[x,x]] + B[[x,x]], w, Ideal]], Throw[False]];
                        If[Not[PolynomialModEquality[\[Delta]/A[[x,x]] + 1/B[[x,x]], 1/w, Ideal]], Throw[
                             False]],
                        {x, BiquandleElements[X]}
                ];
                Throw[True];
        ];

(* Helper method for Axiom 2 *)
BiquandleBracketAxiom2Condition[Ideal_List, A_List, B_List, \[Delta]_, List[x_, y_]] := PolynomialModEquality
     [\[Delta], -A[[x,y]]/B[[x,y]]-B[[x,y]]/A[[x,y]], Ideal];

(* Biquandle bracket Axiom 2 *)
BiquandleBracketAxiom2Q[BiquandleBracket[X_Biquandle, Ideal_List, A_List, B_List, \[Delta]_, w_]] :=
        Catch[
                Do[
                        If[Not[BiquandleBracketAxiom2Condition[Ideal, A, B, \[Delta], T]], Throw[False]],
                        {T, Tuples[BiquandleElements[X], 2]}
                ];
                Throw[True];
        ];

(* Helper function for the first group of conditions in axiom 3---requires additional input of order type *)
BiquandleBracketAxiom3ConditionGroup1[X_Biquandle, Ideal_List, List[L1_List, L2_List, L3_List, R1_List,
     R2_List, R3_List], List[x_, y_, z_]] :=
        PolynomialModEquality[L1[[x,y]] L2[[y,z]] L3[[UnTri[X,x,y], OvTri[X,z,y]]], R1[[x,z]] R2[[OvTri[X,y,x
             ], OvTri[X,z,x]]] R3[[UnTri[X,x,z], UnTri[X,y,z]]], Ideal];

(* Helper function for the fourth condition in axiom 3 *)
BiquandleBracketAxiom3Condition4[X_Biquandle, Ideal_List, A_List, B_List, \[Delta]_, List[x_, y_, z_]] :=
        PolynomialModEquality[A[[x,y]] A[[y,z]] B[[UnTri[X,x,y], OvTri[X,z,y]]], A[[x,z]] B[[OvTri[X,y,x],
             OvTri[X,z,x]]] A[[UnTri[X,x,z], UnTri[X,y,z]]] + A[[x,z]] A[[OvTri[X,y,x], OvTri[X,z,x]]] B[[
             UnTri[X,x,z], UnTri[X,y,z]]] + \[Delta] A[[x,z]] B[[OvTri[X,y,x], OvTri[X,z,x]]] B[[UnTri[X,x,z
             ], UnTri[X,y,z]]] + B[[x,z]] B[[OvTri[X,y,x], OvTri[X,z,x]]] B[[UnTri[X,x,z], UnTri[X,y,z]]],
             Ideal];

(* Helper function for the fifth condition in axiom 3 *)
BiquandleBracketAxiom3Condition5[X_Biquandle, Ideal_List, A_List, B_List, \[Delta]_, List[x_, y_, z_]] :=
        PolynomialModEquality[B[[x,y]] A[[y,z]] A[[UnTri[X,x,y], OvTri[X,z,y]]] + A[[x,y]] B[[y,z]] A[[UnTri[
             X,x,y], OvTri[X,z,y]]] + \[Delta] B[[x,y]] B[[y,z]] A[[UnTri[X,x,y], OvTri[X,z,y]]] + B[[x,y]] B
             [[y,z]] B[[UnTri[X,x,y], OvTri[X,z,y]]], B[[x,z]] A[[OvTri[X,y,x], OvTri[X,z,x]]] A[[UnTri[X,x,z
             ], UnTri[X,y,z]]], Ideal];

(* Biquandle bracket Axiom 3 *)
BiquandleBracketAxiom3Q[BiquandleBracket[X_Biquandle, Ideal_List, A_List, B_List, \[Delta]_, w_]] :=
        Catch[
                Module[{condGroup1Types},
                        condGroup1Types = { {A, A, A, A, A, A}, {A, B, B, B, B, A}, {B, A, B, B, A, B} };
                        Do[
                                For[i=1,i<=3,i++,
                                        If[Not[BiquandleBracketAxiom3ConditionGroup1[X, Ideal, A, B,
                                             condGroup1Types[i], T]], Throw[False]]
                                ];
                                If[Not[BiquandleBracketAxiom3Condition4[X, Ideal, A, B, \[Delta], T]], Throw[
                                     False]];
                                If[Not[BiquandleBracketAxiom3Condition5[X, Ideal, A, B, \[Delta], T]], Throw[
                                     False]],
                                {T, Tuples[BiquandleElements[X],3]}
                        ];
                        Throw[True];
                ];
```

```
        ];

(* A list of all biquandle bracket axioms *)
BiquandleBracketAxioms = {BiquandleBracketAxiom1Q, BiquandleBracketAxiom2Q, BiquandleBracketAxiom3Q};

(* Check whether the "biquandle bracket" satisfies the biquandle bracket axioms and takes values in a factor
     of Z[x,y,z,...]  *)
BiquandleBracketQ[AB_BiquandleBracket, OptionsPattern[]] :=
        Catch[
                For[i=1,i<=3,i++,
                        If[Not[BiquandleBracketAxioms[[i]][AB]],
                                If[OptionValue[Verbose], Print["Biquandle bracket fails Axiom " <> ToString[i
                                     ]]];
                                Throw[False];
                        ];
                ];
                Throw[True];
        ];

(* Default verbosity for BiquandleBracketQ *)
Options[BiquandleBracketQ] = {Verbose -> False};

(* Self explanatory *)
GetBiquandle[BiquandleBracket[X_Biquandle,_,_,_,_,_]] := X;

(* Section involving computing values of biquandle brackets below *)
SetAttributes[BBPP, Orderless];

(* Represents the value of a biquandle bracket at a particular smoothing *)
Smoothing[link_PD,state_List, crossingColors_List, A_List,B_List,\[Delta]_,Ideal_List] :=
        Module[{t1,t2,t3,t4,t5},
                t1 = Thread[{List @@ link, state, crossingColors}];
                t2 = t1 /. {
                                {X[i_,j_,k_,l_],0,{x_,y_}}:>If[j-l==1||l-j>1,A[[x,y]],1/B[[x,y]]] BBPP[i,j]
                                     BBPP[k,l],
                                {X[i_,j_,k_,l_],1,{x_,y_}}:>If[j-l==1||l-j>1,B[[x,y]],1/A[[x,y]]] BBPP[i,l]
                                     BBPP[j,k],
                                {Xp[i_,j_,k_,l_],0,{x_,y_}}:> A[[x,y]]BBPP[i,j]BBPP[k,l],
                                {Xp[i_,j_,k_,l_],1,{x_,y_}}:> B[[x,y]]BBPP[i,l]BBPP[j,k],
                                {Xm[i_,j_,k_,l_],0,{x_,y_}}:>(1/B[[x,y]])BBPP[i,j]BBPP[k,l],
                                {Xm[i_,j_,k_,l_],1,{x_,y_}}:>(1/A[[x,y]])BBPP[i,l]BBPP[j,k]
                        };
                t3 = Times @@ t2;
                t4 = t3 //. {BBPP[i_,j_]BBPP[j_,k_]:>BBPP[i,k]};
                t5 = t4/.BBPP[_,_]^2:>\[Delta];
                Return[PolynomialMod[t5,Ideal]];
        ];

(* Returns a list of pairs of colors for each crossing of a link *)
CrossingColors[link_PD,coloring_List] :=
        List@@(link/.{
                                X[i_,j_,k_,l_]:>If[j-l==1||l-j>1,{coloring[[i]],coloring[[j]]},{coloring[[k
                                     ]],coloring[[j]]}],
                                Xp[i_,j_,k_,l_]:>{coloring[[i]],coloring[[j]]},
                                Xm[i_,j_,k_,l_]:>{coloring[[k]],coloring[[j]]}
                                });

(* Return the number of positive and negative crossings respectively *)
BBnp[link_PD] := Count[link,X[i_,j_,k_,l_]/;j-l==1||l-j>1]+Count[link,x_Xp];
BBnm[link_PD] := Count[link,X[i_,j_,k_,l_]/;l-j==1||j-l>1]+Count[link,x_Xm];

(* Get the value of a biquandle bracket on a particular colored link *)
BiquandleBracketColoringValue[BiquandleBracket[_,Ideal_List,A_List,B_List,\[Delta]_,w_], link_PD,
     coloring_List] :=
        PolynomialMod[Total[Map[Smoothing[link,#,CrossingColors[link,coloring],A,B,\[Delta],Ideal]&,Tuples
             [{0,1},Length[link]]]]*w^(BBnm[link]-BBnp[link]),Ideal];
```

```
(* Get the (multiset) value of a biquandle bracket on a particular link *)
BiquandleBracketValue[AB_BiquandleBracket, link_PD] :=
        Map[BiquandleBracketColoringValue[AB, link, #]&, AllBiquandleColorings[link, GetBiquandle[AB]]];

End[]
EndPackage[]
```

A.3 Biquandle2Cocycles.wl

```
(* Author: Vilas Winstein *)

BeginPackage["Biquandle2Cocycles`"]
Needs["KnotTheory`"]
Needs["Biquandles`"]

Biquandle2Cocycle::usage="The symbol type for a biquandle 2-cocycle.";
Biquandle2CocycleAxiom1::usage="Returns true if the given Biquandle2Cocycle symbol satisfies biquandle 2-
    cocycle axiom 1, false otherwise.";
Biquandle2CocycleAxiom2::usage="Returns true if the given Biquandle2Cocycle symbol satisfies biquandle 2-
    cocycle axiom 2, false otherwise.";
Biquandle2CocycleQ::usage="Returns true if the given Biquandle2Cocycle symbol satisfies all biquandle 2-
    cocycle axioms, false otherwise.";
Biquandle2CocycleInvariantValue::usage="Returns the value (a multiset) of the given biquandle 2-cocycle
    invariant on the given link.";

Begin["Private`"]

(* How to declare a biquandle 2-cocycle *)
Biquandle2Cocycle[B_Biquandle, \[Phi]_List];

Biquandle2CocycleAxiom1[Biquandle2Cocycle[B_Biquandle, \[Phi]_List]] := AllTrue[Table[\[Phi][[i,i]], {i, 1,
    Length[BiquandleElements[B]]}], #==1&];

Biquandle2CocycleAxiom2Helper[B_Biquandle, \[Phi]_List, List[x_,y_,z_]] := \[Phi][[x,y]] \[Phi][[y,z]] \[Phi
    ][[OvTri[B,x,y], UnTri[B,z,y]]] == \[Phi][[x,z]] \[Phi][[OvTri[B,y,x], OvTri[B,z,x]]] \[Phi][[UnTri[B,x,
    z], UnTri[B,y,z]]];

Biquandle2CocycleAxiom2[Biquandle2Cocycle[B_Biquandle, \[Phi]_List]] := AllTrue[Tuples[BiquandleElements[B
    ],3], Biquandle2CocycleAxiom2Helper[B,\[Phi],#]&];

Biquandle2CocycleQ[\[Phi]_Biquandle2Cocycle] := Biquandle2CocycleAxiom1[\[Phi]] && Biquandle2CocycleAxiom2[\[
    Phi]];

(* The value of the invariant on a particular colored crossing *)
Biquandle2CocycleInvariantCrossingValue[X[i_,j_,k_,l_], coloring_List, \[Phi]_List] := If[j-l==1||l-j>1, \[
    Phi][[coloring[[i]],coloring[[j]]]], 1/\[Phi][[coloring[[k]],coloring[[j]]]]];
Biquandle2CocycleInvariantCrossingValue[Xp[i_,j_,k_,l_], coloring_List, \[Phi]_List] := \[Phi][[coloring[[i
    ]],coloring[[j]]]];
Biquandle2CocycleInvariantCrossingValue[Xm[i_,j_,k_,l_], coloring_List, \[Phi]_List] := 1/\[Phi][[coloring[[k
    ]],coloring[[j]]]];

(* The value of the invariant on a particular colored link *)
Biquandle2CocycleInvariantColoringValue[L_PD, coloring_List, \[Phi]_List] := Times @@ (
    Biquandle2CocycleInvariantCrossingValue[#,coloring,\[Phi]]& /@ L);

Biquandle2CocycleInvariantValue[L_PD, Biquandle2Cocycle[B_Biquandle, \[Phi]_List]] :=
    Biquandle2CocycleInvariantColoringValue[L, #, \[Phi]]& /@ AllBiquandleColorings[L, B];

End[]
EndPackage[]
```

A.4 BiquandleBracketCanonical2Cocycle.wl

```
(* Author: Vilas Winstein *)

BeginPackage["BiquandleBracketCanonical2Cocycle`"]
Needs["KnotTheory`"]
Needs["Biquandles`"]
Needs["BiquandleBrackets`"]
Needs["Biquandle2Cocycles`"]

BiquandleBracketCanonical2Cocycle::usage="Return the canonical 2-cocycle for a given biquandle bracket.";
BiquandleBracketKernel::usage="Gives a set of generators for G = < q q[x,y]^(-1) : x,y \[Element] X >.";
MultisetQuotientEqualityQ::usage="Check for equality of two multisets with entries in a multiplicative
    quotient group.";

Begin["Private`"]

BiquandleBracketCanonical2Cocycle[BiquandleBracket[X_Biquandle, Ideal_List, A_List, B_List, \[Delta]_, w_]]
    := Biquandle2Cocycle[X, A[[1,1]]^(-1) A];

q[A_List, B_List, x_Integer, y_Integer] := -B[[x,y]] / A[[x,y]];

BiquandleBracketKernel[BiquandleBracket[X_Biquandle, Ideal_List, A_List, B_List, \[Delta]_, w_]] := Table[
    PolynomialMod[q[A,B,1,1]/q[A,B,x,y], Ideal], {x, BiquandleElements[X]}, {y, BiquandleElements[X]}] //
    Flatten // DeleteDuplicates;

MultiplicativeMod[kernel_List] := #->1& /@ kernel;

MultisetQuotientEqualityQ[mset1_List, mset2_List, kernel_List] := Length[mset1] == Length[mset2] && AllTrue[
    mset1, Count[Factor[#^(-1) mset1] //. MultiplicativeMod[kernel], 1] == Count[Factor[#^(-1) mset2] //.
    MultiplicativeMod[kernel], 1] &];

End[]
EndPackage[]
```

Bibliography

[1] Dror Bar-Natan. On khovanov's categorification of the jones polynomial. *Algebraic & Geometric Topology*, 2(1):337–370, 2002.

[2] Jose Ceniceros, Mohamed Elhamdadi, Matthew Green, and Sam Nelson. Augmented biracks and their homology. *International Journal of Mathematics*, 25(09):1450087, 2014.

[3] Mohamed Elhamdadi and Sam Nelson. *Quandles*, volume 74. American Mathematical Soc., 2015.

[4] Will Hoffer, Adu Vengal, and Vilas Winstein. The structure of biquandle brackets. *arXiv preprint arXiv:1907.11487*, 2019. To appear in the Journal of Knot Theory and its Ramifications.

[5] Vaughan FR Jones. The jones polynomial. *Discrete Math*, 294:275–277, 2005.

[6] Mikhail Khovanov. A categorification of the jones polynomial. *Duke Mathematical Journal*, 101(3):359–426, 2000.

[7] The Knot Atlas. Planar diagrams. http://katlas.org/wiki/Planar_Diagrams, 21 February 2013. Accessed: 12 April 2020.

[8] Sam Nelson, Michael E Orrison, and Veronica Rivera. Quantum enhancements and biquandle brackets. *Journal of Knot Theory and Its Ramifications*, 26(05):1750034, 2017.

[9] Kurt Reidemeister. Knoten und gruppen. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, volume 5, pages 7–23. Springer, 1927.