

COMPUTING THE GENERATING FUNCTION OF A
COINVARIANTS MAP

BY

JESSE FROHLICH

A thesis submitted in conformity with
the requirements for the degree of
Doctor of Philosophy
Graduate Department of Mathematics
University of Toronto

© 2023 Jesse Frohlich

ABSTRACT

Computing the generating function of a coinvariants map

Jesse Frohlich

Doctor of Philosophy

Graduate Department of Mathematics

University of Toronto

2023

A well-known source of knot invariants is representations of quasitriangular Hopf algebras (also known as quantum groups). These invariants require exponential time in the number of crossings to compute. Recent work has allowed for polynomial-time computations within the Hopf algebras themselves, using perturbed Gaussian differential operators. This thesis introduces and explores a partial expansion of the tangle invariant Z introduced by Bar-Natan and van der Veen [BNvdVb]. We expand the use of the Hopf algebra $\mathfrak{U}(\mathfrak{sl}_{2+}^0)$ to include its space of coinvariants, providing an extension Z^{tr} of Z from open tangles to links.

We compute a basis for the space of coinvariants of $\mathfrak{U}(\mathfrak{sl}_{2+}^0)$ and a closed-form expression for the exponential generating function of the corresponding trace map. The resulting function is not directly compatible with the previous research, so we also find a method of computing Z^{tr} on a subclass of links and write a program to compute Z^{tr} on two-component links. Contrary to expectations, we find that Z^{tr} is neither stronger nor weaker than the Multivariable Alexander polynomial. This unexplained behaviour warrants further study into Z^{tr} and its relationship with other invariants.

For M. L. E.

ACKNOWLEDGEMENTS

Thank you:

To my advisor, Dror Bar-Natan, whose consistent meetings, guidance, and encouragement brought me to this point, and to my supervisory committee, Joel Kamnitzer and Eckhard Meinrenken, for their feedback and for helping me connect to more of the department and its research.

To Jason Siefken and the other teaching mentors in the math department: you have made me a better educator and communicator.

To the math department administration, especially Jemima Merisca and Sonja Injac, whose kindness made the department a comfortable place to be.

To Assaf Bar-Natan, Vincent Girard, Caleb Jonker, and Adriano Pacifico, whose friendship, bike trips, and cooking sessions provided much joy; to the people I met in the department, especially those who opened my thesis to confirm they were included here: thank you specifically.

To the care-taking staff at the St. George campus, especially Maria and Maria, whose diligent work and kind words have made this campus beautiful.

To the U of T Graduate Christian Fellowship for providing a refreshing breadth of conversation and companionship.

To Rosemary and Alan Johnstone, whose hospitality made me feel immediately welcome in a new city.

To my family, who encouraged me in my academic pursuits from the start: to my mom for embracing my atypical learning styles, for instilling my love of reading, and for teaching me how to think deeply; to my dad for teaching me to cycle, cook, and notice the invisible among us; and to my brother for all the laughter you bring to my life.

To the Perrins: you are my second family, and your support and love for me is precious.

To my wife, Emily Langridge, who helped me with the motivation to finish. I look forward to the next chapters of my life with you.

CONTENTS

1	Executive summary	7
1.1	Algebraic tools for understanding knots	7
1.2	Computational improvements using the universal invariant .	9
1.3	Extending Z to links	9
1.4	Further study	11
2	Tensor products and meta-objects	12
2.1	Tensor product notation	12
2.2	Meta-objects	14
2.3	Algebraic definitions	18
2.4	The meta-algebra of tangle diagrams	23
2.5	The meta-algebra U	35
2.6	Morphisms between meta-objects	38
3	Perturbed Gaußians	41
3.1	Expressing morphisms as generating functions	41
3.2	Computational example	46
4	Constructing the trace	48
4.1	Extending an invariant of open tangles to mixed tangles . .	48
4.2	The space of coinvariants of U	50
4.3	Computational example	55
5	Conclusions	58
5.1	Limitations of Z^{tr}	58
5.2	Comparison with the multivariable Alexander polynomial .	58
5.3	Further work	59
A	Code	60
A.1	Implementation of the perturbed Gaußian framework	60
A.2	Implementation of the trace	71
A.3	Defining Z and Z^{tr}	75

A.4	Implementation of rotation number algorithm	81
B	Table of values	94

EXECUTIVE SUMMARY

1.1 ALGEBRAIC TOOLS FOR UNDERSTANDING KNOTS

Knotted objects

In the field of knot theory, distinguishing between two knots or links has proven to be a difficult task. Computing strong invariants of knotted objects is a popular way to aid with the classification of these objects.

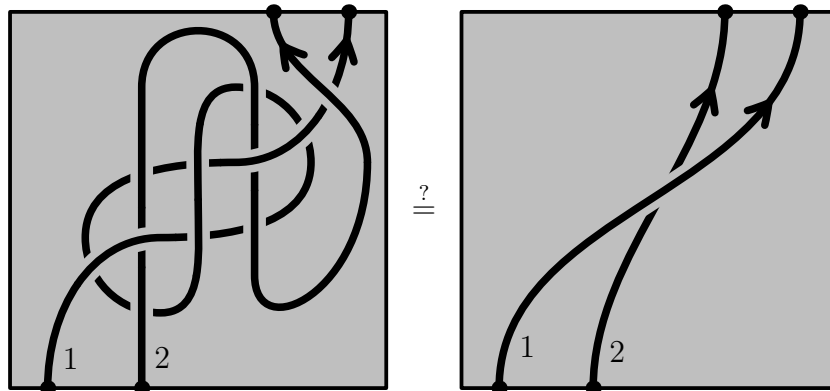


Figure 1.1: Two strings tangled together. Can they be disentangled?

Merely being able to distinguish between two knotted objects does not always provide us with enough information about these topological structures. For instance, one may ask if a link is a satellite (i.e. whether its complement has a non-trivial embedding of a torus), whether a knot is slice (i.e. it is the boundary of a disk in \mathbb{R}^4), or whether it is ribbon (i.e. it is the boundary of a disk in \mathbb{R}^3 with restricted types of singularities). (For more details see, for instance, Lickorish's [Lic].) Many interesting properties of knots can be phrased in terms of certain topological operations, such as strand-doubling (taking a strand and replacing it with two copies of itself, as in figure 1.2) or strand-stitching (joining two open components together to form one longer one).

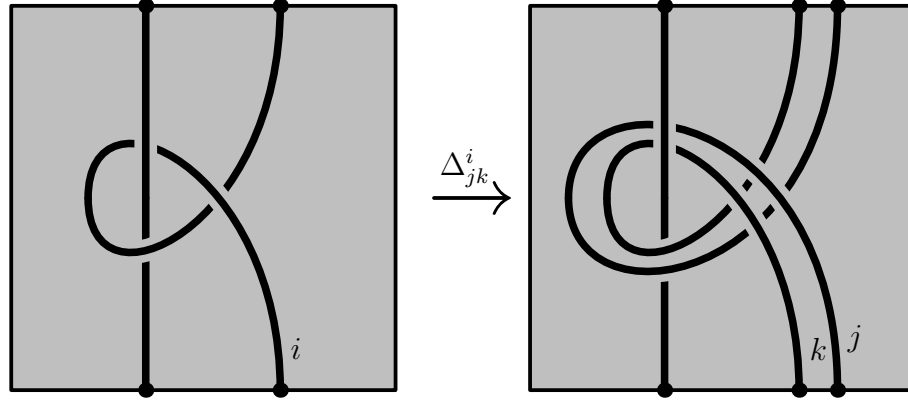


Figure 1.2: An example of strand-doubling.

Open problems such as the Ribbon Slice Conjecture (asking whether there exists a slice knot which is not ribbon, posed by R. H. Fox [Fox]) may be advanced by the development of “topologically aware” invariants—those which preserve topological data in a retrievable way.

Quantum invariants

One such class of topological invariants is derived from quasitriangular Hopf algebras, which are algebraic structures whose operations mimic those of knotted objects. In this approach, one takes a knotted object and decomposes it into a sequence of topological operations (such as stitching strands or doubling strands), then maps each of these operations to a corresponding algebraic operation. The composition of these algebraic operations is the value of the invariant.

More specifically, a Hopf algebra is an algebra A together with several maps between various tensor powers thereof (for instance, a comultiplication map $\Delta: A \rightarrow A \otimes A$). To each crossing is assigned an element of $A \otimes A$. Tensor factors which belong to the same strand are concatenated by multiplying the associated algebra elements. The value of the invariant is an element of a tensor power of A . We go over this in more detail in chapter 2.

While this formulation is elegant, it has a notable drawback: computing the invariant of a tangle with many components requires manipulating large tensor powers of A . One remedy is to instead perform the computation in a representation V of A with a low dimension, though the issue of exponential

growth in complexity remains. This limitation restricts the utility of quantum invariants to smaller knots.

1.2 COMPUTATIONAL IMPROVEMENTS USING THE UNIVERSAL INVARIANT

To avoid the issue of exponential computational complexity, one can instead investigate the set of all values of the universal invariant (using the algebra itself instead of a representation) as a subset of the algebra and its tensor powers. For a particular choice of algebra (namely $\widehat{\mathfrak{U}}(\mathfrak{sl}_{2+}^{\epsilon})$), as investigated by Dror Bar-Natan and Roland van der Veen in [BNvdVb]) the space of values the corresponding invariant Z can take is significantly smaller than the whole space; the rank of the space of values grows only quadratically with the number of crossings in the knotted object. In particular, by looking at the generating functions of the algebra operations, one sees that the value of Z on tangles always takes the form of a (perturbed) Gaussian. Computationally, this means one need only keep track of a quadratic form and a small perturbation. The invariant Z dominates the \mathfrak{sl}_2 -coloured Jones polynomial. Here, we will focus on the case when $\epsilon = 0$, for which Z becomes an efficient computation of the Alexander polynomial Δ on knots. This topic is described in chapter 3.

1.3 EXTENDING Z TO LINKS

The research program outlined by Bar-Natan and van der Veen computes Z only for (open) tangles—that is, collections of open strands whose endpoints are fixed to a boundary circle. (Note that this includes long knots, which are exactly the one-component tangles.) This thesis is focused on extending Z and its computations to tangles with closed components, which includes links. Here we summarize chapter 4.

Computing the extended map

The first task is to determine the space in which the extended invariant, which we will call Z^{tr} , lives. One may observe that in a matrix algebra, one is able to contract two matrices together via matrix multiplication. When one

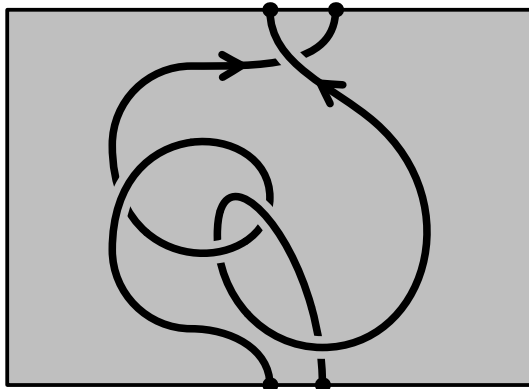


Figure 1.3: An open tangle. All components intersect the boundary.

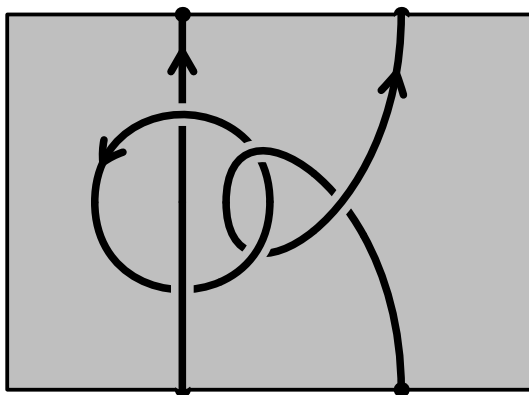


Figure 1.4: A tangle with a closed component.

wishes to contract a matrix along itself, one uses the trace map. Analogously, since stitching two strands in a tangle corresponds to multiplication, closing a strand into a loop should correspond algebraically to a trace map.

In a generic algebra A , the trace map is defined as the projection onto the set of coinvariants: $\text{tr}: A \rightarrow A_A = A/[A, A]$.^{†1} In order to extend Z in this framework, we must first compute the space of coinvariants for the algebra $U = \widehat{\mathfrak{U}}(\mathfrak{sl}_{2+}^0)$, then compute the coinvariants map, and write it as a generating function. (This is accomplished in section 4.2.)

^{†1} Here, $[A, A] = \text{span}\{[x, y] \mid x, y \in A\}$ refers to the vector space of Lie brackets, not the ideal generated thereby. The space A_A does not have an algebra structure in general.

Performing computations

Unfortunately, the resulting trace map does not take the form of a perturbed Gaussian in a way that readily connects to the existing framework. In order to determine whether further study of Z^{tr} is merited, we must find an alternative computation method to get a preliminary sense of the strength of Z^{tr} .

For a subclass of links (which includes all two-component links), we compute an explicit closed form for the trace map, then implement a computer program to compute the value of Z^{tr} on all two-component links with up to 11 crossings. When applied to knots, Z computes the Alexander polynomial. When applied to two-component tangles, one may expect that Z^{tr} would produce the natural generalization to multiple components: the Multivariable Alexander polynomial (MVA). Surprisingly, the MVA and Z^{tr} are incomparable, with each being able to distinguish pairs of links the other cannot. (See section 5.2 for more information.)

1.4 FURTHER STUDY

As Z^{tr} does not generalize Z in the manner expected, several interesting avenues of further research become open. Firstly, the precise relationship between Z^{tr} and the MVA remains unknown. Second is the challenge of finding an efficient method for computing Z^{tr} on links with more than two components. Currently this endeavour is mired in complications with non-elementary functions where quadratic forms normally appear. Third is the question of the existence of other viable trace candidates. In particular, it may be worth exploring whether a universal trace with respect to the perturbed Gaussian framework defines a sufficiently useful invariant. See chapter 5 for more information.

TENSOR PRODUCTS AND META-OBJECTS

2.1 TENSOR PRODUCT NOTATION

In what follows, we will extensively use tensor products, tensor powers, and generalizations thereof. We begin by introducing the notation that will make working with these objects more straightforward, similar to the slot-naming index notation introduced by Penrose [Pen].

Let \mathbb{k} be a field and V a \mathbb{k} -vector space (for the moment assumed to be finite dimensional). When working with a large tensor power $V^{\otimes n}$ of V , it will often be more convenient to label tensor factors with elements of a finite set S (with $|S| = n$) rather than by their position in a linear order.

For example, consider the vector $u \otimes v \otimes w \in V^{\otimes 3}$. Let us choose an index set $S = \{i, j, k\}$. We then may equivalently write this vector by labelling each tensor factor with one of the elements of S , say $u_i v_j w_k$. Since the labels serve to distinguish the separate factors, this vector may equivalently be written as $u_i v_j w_k = v_j u_i w_k = w_k v_j u_i \in V^{\otimes S}$, where $V^{\otimes S}$ denotes the tensor power of V . An additional notation which we will prefer is $V_S = V^{\otimes S}$. We formalize the idea below:

Definition 2.1 (indexed tensor powers). Let V be a vector space and $S = \{s_1, \dots, s_n\}$ be a finite set. We define the indexed tensor power of V to be the collection of formal linear combinations of functions from S to V

$$V_S := V^{\otimes S} := \text{span}\{f: S \rightarrow V\} / \sim \quad (2.1)$$

subject to the standard multilinear relations, namely multi-additivity and the factoring of scalars:

By multi-additivity, we mean that for each $i \in S$ and $f, g \in V_S$ satisfying $f(s) = g(s)$ for each $s \in S \setminus \{i\}$, we have:

$$f + g \sim \left(s \mapsto \begin{cases} f(s) = g(s) & \text{if } s \neq i \\ f(i) + g(i) & \text{if } s = i \end{cases} \right) \quad (2.2)$$

We will write such functions $f: S \rightarrow V$ with $f(s_i) = v_i$ with the following notation:

$$(v_1)_{s_1} (v_2)_{s_2} \cdots (v_n)_{s_n} := f \quad (2.3)$$

The factoring of scalars relation is:

$$(v_1)_{s_1} (v_2)_{s_2} \cdots (\lambda v_i)_{s_i} \cdots (v_n)_{s_n} = \lambda \cdot (v_1)_{s_1} (v_2)_{s_2} \cdots (v_n)_{s_n} \quad (2.4)$$

Similarly, equation (2.2) in the style of equation (2.3) becomes:

$$\begin{aligned} & \left((v_1)_{s_1} (v_2)_{s_2} \cdots x_{s_i} \cdots (v_n)_{s_n} \right) + \left((v_1)_{s_1} (v_2)_{s_2} \cdots y_{s_i} \cdots (v_n)_{s_n} \right) \\ & = (v_1)_{s_1} (v_2)_{s_2} \cdots (x + y)_{s_i} \cdots (v_n)_{s_n} \end{aligned} \quad (2.5)$$

Next, we introduce notation for maps between tensor powers so that we may unambiguously refer to appropriate tensor factors while defining morphisms. We accomplish this task by adding a convenient way of writing the domain and codomain of a map. Let D and C be finite sets, and $T: V_D \rightarrow V_C$. We will denote T alternatively by T_C^D , so that its domain and codomain are easily read off. It is important to note that when T is not symmetric in its arguments, the order of the indices in this notation matters.

Example 2.2. Let $V = \mathbb{R}^3$, and $T_c^{a,b}$ (equivalently, $T: V_{\{a,b\}} \rightarrow V_{\{c\}}$) defined by $T_c^{a,b}(\vec{v}_a \vec{w}_b) = (\vec{v} \times \vec{w})_c$ denote the cross product. Stating that the cross product is antisymmetric may be accomplished without referencing vectors by writing:

$$T_c^{a,b} = -T_c^{b,a} \quad (2.6)$$

Remark 2.3. There are three special cases with this notation:

- Given a (multi)linear functional $\phi: V_S \rightarrow \mathbb{k} \cong V_\emptyset$, we will write ϕ^S instead of ϕ_\emptyset^S . The linear order on S remains in this notation.
- Elements $v \in V_S$ will be interpreted as a map $v: \mathbb{k} = V_\emptyset \rightarrow V_S$ written v_S instead of v_S^\emptyset .

- When only one index is present in a subscript or superscript, and its omission does not introduce an ambiguity in an expression, then it may be omitted to improve readability. For instance, a map $\phi: V_{\{1,2\}} \rightarrow V_{\{3\}}$ may be written as $\phi^{1,2}$ instead of $\phi_3^{1,2}$, with the canonical isomorphism $V \cong V_{\{3\}}$ being suppressed.

When taking the tensor product of two tensor powers, we follow [BNS] and use the notation “ \sqcup ” instead of “ \otimes ”:

$$V_X \sqcup V_Y := V_{X \sqcup Y} \quad (2.7)$$

Additionally, given $\phi_{C_1}^{D_1}$ and $\psi_{C_2}^{D_2}$ such that $D_1 \cap D_2 = \emptyset = C_1 \cap C_2$, we have a product morphism $\phi_{C_1}^{D_1} \psi_{C_2}^{D_2} := \phi \otimes \psi: V_{D_1 \sqcup D_2} \rightarrow V_{C_1 \sqcup C_2}$, which we also write with concatenation.

Finally, we point out that any morphism T_C^D may be extended to one with larger domain and codomain. We introduce the notation $T[S] := T_C^D \text{id}_S^S$ for this concept. When no ambiguity arises, we will also suppress the “[S]” so that T_C^D represents more generally:

$$(T_C^D)(v_D w_S) := (T_C^D v_D) w_S \quad (2.8)$$

for any $v_D \in V_D$ and $w_S \in V_S$.

2.2 META-OBJECTS

While the above notation is helpful when working with vector spaces, we are interested in also using the same notation to describe a tangle. Our formulation of tangles (introduced in section 2.4) is not a tensor product, though it shares many similarities. In particular, the domains and codomains of the maps we have discussed so far have only depended on the index set. With this observation, we replace the notation of tensor powers with that of a so-called meta-object, first introduced by Bar-Natan and Selmani in [BNS]. We introduce this concept by starting with monoids.

We now go through the process of defining a meta-monoid, which is a generalization of a monoid object. Traditionally, the data of a monoid object are the following:

- An object M in a monoidal category $(\mathcal{C}, \otimes, \mathbf{1})$.

- A morphism $m: M \otimes M \rightarrow M$ called the “multiplication” operation.
- A “unit” morphism $\eta: \mathbf{1} \rightarrow M$.^{†1}
- A collection of relations between the operations, written as equalities of morphisms between Cartesian powers of M . For example, associativity may be written:

$$\begin{array}{ccc}
 M \otimes M \otimes M & \xrightarrow{m \otimes \text{id}} & M \otimes M \\
 \text{id} \otimes m \downarrow & & \downarrow m \\
 M \otimes M & \xrightarrow{m} & M
 \end{array} \tag{2.9}$$

Further, the data of these relations is extended to higher powers of M by acting on other components by the identity:

$$\begin{array}{ccc}
 M^{\otimes(n+3)} & \xrightarrow{m \otimes \text{id}^{\otimes(n+1)}} & M^{\otimes(n+2)} \\
 \text{id} \otimes m \otimes \text{id}^{\otimes n} \downarrow & & \downarrow m \otimes \text{id}^{\otimes n} \\
 M^{\otimes(n+2)} & \xrightarrow{m \otimes \text{id}^{\otimes n}} & M^{\otimes(n+1)}
 \end{array} \tag{2.10}$$

Let us alter how we package these data so as to maximize the clarity of the meta-monoid structure:

1. Instead of linear orders of factors $M \otimes \cdots \otimes M$, we will index factors by a finite set X , writing it $M_X := \{f: X \rightarrow M\}$ in the style of equation (2.1).^{†2}
2. The indexed factors will determine how the monoid operations act. For instance, multiplication of factor i and j together, with the result labelled in factor k is to be written $m_k^{ij}: M_{\{i,j\}} \rightarrow M_{\{k\}}$.
3. Instead of implicitly including extensions of operations to higher powers by the identity, we will parametrize the extension by finite sets by $\phi_C^D[X] := \phi_C^D \otimes \text{id}_X^X$. For example, multiplication $m_k^{ij}: M_{\{i,j\}} \rightarrow M_{\{k\}}$ generates a family of maps $m_k^{ij}[X]: M_{\{i,j\} \sqcup X} \rightarrow M_{\{k\} \sqcup X}$, each of which must satisfy the relations of the monoid object such as equation (2.10). (Again, the “[X]” is frequently omitted from writing.)

^{†1} When $\mathcal{C} = \mathbf{Set}$, we usually write the unit as an element $1 = \eta(1) \in M$, where $\mathbf{1} = \{1\}$.

^{†2} Indeed, when $\mathcal{C} = \mathbf{Vect}$, these definitions are identical when the monoidal product is \otimes . In this case, they are called algebras.

This way of packaging the data leads us to the following generalization:

Definition 2.4 (meta-monoid [BNS, §3.3]). A meta-monoid in \mathcal{C} is the following data:

- A family of objects $M_X \in \mathcal{C}$, indexed over finite sets X , with set bijections $\psi: X \xrightarrow{\sim} Y$ inducing isomorphisms $\iota_\psi: M_X \xrightarrow{\sim} M_Y$.
- A family of morphisms $m_k^{ij}[X]: M_{\{i,j\} \sqcup X} \rightarrow M_{\{k\} \sqcup X}$ called “multiplication”.
- A family of “unit” morphisms $\eta_i[X]: M_X \rightarrow M_{\{i\} \sqcup X}$.
- A collection of relations between the morphisms, written as equalities of morphisms between the M_X ’s. In particular,
 - associativity:

$$\begin{array}{ccc}
 M_{\{1,2,3\} \sqcup X} & \xrightarrow{m_1^{1,2}[X \sqcup \{3\}]} & M_{\{1,3\} \sqcup X} \\
 m_2^{2,3}[X \sqcup \{1\}] \downarrow & & \downarrow m_1^{1,3}[X] \\
 M_{\{1,2\} \sqcup X} & \xrightarrow{m_1^{1,2}[X]} & M_{\{1\} \sqcup X}
 \end{array} \quad (2.11)$$

- the identity:

$$\begin{array}{ccc}
 M_{\{1\} \sqcup X} & \xrightarrow{\eta_2[X]} & M_{\{1,2\} \sqcup X} \\
 \searrow \text{id} & & \downarrow m_1^{2,1}[X] \\
 & & M_{\{1\} \sqcup X} \\
 & & \uparrow m_1^{1,2}[X]
 \end{array} \quad (2.12)$$

- index relabelling:

$$\begin{aligned}
 \iota_\psi \circ m_k^{ij}[X] \circ \iota_\psi^{-1} &= m_{\psi(k)}^{\psi(i), \psi(j)}[\psi(X)] \\
 \iota_\psi \circ \eta_i[X] \circ \iota_\psi^{-1} &= \eta_{\psi(i)}[\psi(X)]
 \end{aligned} \quad (2.13)$$

Example 2.5 (monoid objects are meta-monoids). Any monoid object M in a strict, symmetric monoidal category $(\mathcal{C}, \otimes, \{1\})$ has the structure of a meta-monoid $\{M_X\}_X$ via $M_X := M^{\otimes X \dagger 3}$, $m_k^{ij}[X] := m_k^{ij} \otimes \text{id}_X^X$, and $\eta_i[X](v) := 1_i \otimes v$ for any $v \in M^{\otimes X}$.

†3 By $M^{\otimes X}$ we mean $M^{\otimes |X|}$ together with a choice of assigning each factor an index, analogous to definition 2.1. By symmetry, any choice of assignment is equivalent to any other.

Consider the following structure, which satisfies the definition of a meta-monoid, but is not a monoid in the traditional sense:

Example 2.6 (the meta-monoid of square matrices). Let \mathbb{k} be a field and $M_X := \text{Mat}_{X \times X}(\mathbb{k})$ be the set of square matrices whose rows and columns are indexed by the finite set $X = \{x_i\}_i$. Define $m_k^{ij}[X]: M_{X \sqcup \{i,j\}} \rightarrow M_{X \sqcup \{k\}}$ by $m_k^{ij}[X]((a_{rs})_{rs}) := (a_{rs} + \delta_{rk}(a_{is} + a_{js}) + \delta_{sk}(a_{si} + a_{sj}))_{rs}$. That is, the multiplication of two indices corresponds to the summation of their respective rows and columns, the result of which is stored in row and column k :

$$\begin{bmatrix} a_{x_1, x_1} & \cdots & a_{x_1, i} & a_{x_1, j} \\ \vdots & \ddots & \vdots & \vdots \\ a_{i, x_1} & \cdots & a_{ii} & a_{ij} \\ a_{j, x_1} & \cdots & a_{ji} & a_{jj} \end{bmatrix} \xrightarrow{m_k^{ij}} \begin{bmatrix} a_{x_1, x_1} & \cdots & a_{x_1, i} + a_{x_1, j} \\ \vdots & \ddots & \vdots \\ a_{i, x_1} + a_{j, x_1} & \cdots & a_{ii} + a_{ji} + a_{ij} + a_{jj} \end{bmatrix} \tag{2.14}$$

where the last column and row on the right-hand-side is indexed by k . The unit $\eta_i[X]((a_{rs})_{rs})$ extends $(a_{rs})_{rs}$ to include a row and column of 0's, each labelled by the index i .

Example 2.7 (tangles form a meta-algebra). Tangles are the main example of a meta-algebra which is not an algebra in the traditional sense. We go into more detail in section 2.4.

In order to define other meta-objects (such as a meta-coalgebra or a meta-semigroup) we provide the following more general definition:

Definition 2.8 (meta-object). Let \mathcal{C} be a category. A meta-object in \mathcal{C} is four things:

1. A collection of objects A_X , one for each choice of finite set X . (This serves as the analogue to monoidal powers.)
2. For each bijection $\psi: X \xrightarrow{\sim} Y$ of finite sets X and Y , a reindexing isomorphism $\iota_\psi: A_X \xrightarrow{\sim} A_Y$.
3. A collection of operations $\phi_1, \phi_2, \dots, \phi_n$ each with a signature $|\phi_i| \in \mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0}$. For any pair of finite sets (D, C) satisfying $(|D|, |C|) = |\phi|$, we have a morphism:

$$\phi_C^D: A_D \rightarrow A_C \tag{2.15}$$

4. For each operation ϕ_C^D , there is a collection of morphisms $\phi_C^D[\cdot]$ indexed by finite sets such that for each finite set S, T :

- a) $\phi[S]: A_{D \sqcup S} \rightarrow A_{C \sqcup S}$
- b) $\phi[\emptyset] = \phi$
- c) $(\phi[S])[T] = \phi[S \sqcup T]$

When no ambiguity arises, we will omit the portion written in square brackets, so that ϕ will stand for $\phi[X]$, with the set X determined from context.

Finally, we may define the product of two objects A_S and A_T by $A_S A_T = A_{S \sqcup T}$. Given operations $\phi_{C_1}^{D_1}$ and $\psi_{C_2}^{D_2}$ such that $D_1 \cap D_2 = \emptyset = C_1 \cap C_2$, we have a product morphism $\phi_{C_1}^{D_1} \psi_{C_2}^{D_2}: A_{D_1 \sqcup D_2} \rightarrow A_{C_1 \sqcup C_2}$.

Composition of operators $\phi_{C_1}^{D_1}$ and $\psi_{C_2}^{D_2}$ is defined when $C_1 = D_2$ (This is visualized in figure 2.1.):

$$\psi_{C_2}^{D_2} \circ \phi_{C_1}^{D_1}: \mathcal{C}_{D_1} \rightarrow \mathcal{C}_{C_2} \quad (2.16)$$

Remark 2.9. In this text, we will denote left-to-right composition with the symbol “//” (pronounced “then”): $f // g := g \circ f$. Writing function composition in this order assists with readability when there are many functions to apply.

Remark 2.10. To make expressions easier to read, we introduce the domain extension implicitly in the following context: given morphisms $\phi_{C_1}^{D_1}$ and $\psi_{C_2}^{D_2}$ such that $C_2 \cap (C_1 \setminus D_2) = \emptyset = D_1 \cap (D_2 \setminus C_1)$, we define:

$$\phi_{C_1}^{D_1} // \psi_{C_2}^{D_2} := \phi_{C_1}^{D_1}[D_2 \setminus C_1] // \psi_{C_2}^{D_2}[C_1 \setminus D_2] \quad (2.17)$$

Figure 2.2 visualizes this extension.

The two extreme cases of this definition are:

- When $C_1 \cap D_2 = \emptyset$, equation (2.17) becomes $\phi_{C_1}^{D_1} \psi_{C_2}^{D_2}$.
- When $C_1 = D_2$, equation (2.17) becomes the composition $\phi_{C_1}^{D_1} // \psi_{C_2}^{D_2}$ exactly.

2.3 ALGEBRAIC DEFINITIONS

We now introduce the algebraic structures which will be used to define the tangle invariant. These definitions follow those given by Majid in [Maj], presented in a way that their corresponding meta-structure are readily visible.

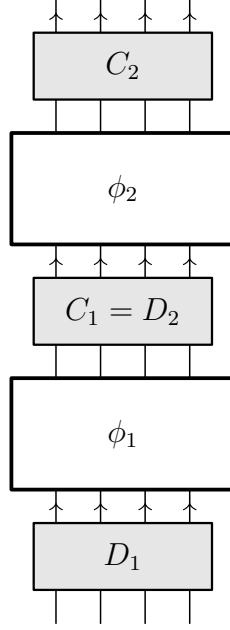


Figure 2.1: We may visualize a composition of morphisms with a graphical calculus. This graphic represents $(\phi_1)_{C_1}^{D_1} \parallel (\phi_2)_{C_2}^{D_2}$ when $C_1 = D_2$. Each arrow represents one factor. Indices are written in grey boxes.

Definition 2.11 (meta-algebra). A meta-algebra (or meta-monoid^{†4}) is a meta-object $\{A_X\}_X$ in \mathcal{C} with operations an associative multiplication $m_k^{i,j} : A_{\{i,j\}} \rightarrow A_{\{k\}}$ (satisfying equation (2.18)) and a unit $\eta_i : A_{\emptyset} \rightarrow A_{\{i\}}$ satisfying equation (2.19).

Remark 2.12. When $\mathcal{C} = (\mathbb{k}\text{-Vect}, \otimes)$ and $A_X = V_X = V^{\otimes X}$ for some vector space V , definition 2.11 becomes the more familiar definition of an algebra. Then A_{\emptyset} is the field \mathbb{k} . It is more common to think of the unit as an element $\mathbf{1} \in V$. The unit map is then defined by linearly extending the assignment $\eta_i(\mathbf{1}) = \mathbf{1}_i$.

$$\begin{array}{ccc}
 A_{\{1,2,3\}} & \xrightarrow{m_1^{1,2}} & A_{\{1,3\}} \\
 m_2^{2,3} \downarrow & & \downarrow m_1^{1,3} \\
 A_{\{1,2\}} & \xrightarrow{m_1^{1,2}} & A_{\{1\}}
 \end{array} \quad (2.18)$$

$$\begin{array}{ccc}
 A_{\{1\}} & \xrightarrow{\eta_2} & A_{\{1,2\}} \\
 \searrow \text{id} & & \downarrow m_1^{2,1} \\
 & & A_{\{1\}}
 \end{array} \quad \begin{array}{c} \Downarrow \\ m_1^{1,2} \end{array} \quad (2.19)$$

^{†4} This is a repeat of definition 2.4. The only difference between an algebra object and a monoid object is the presence of a linear structure.

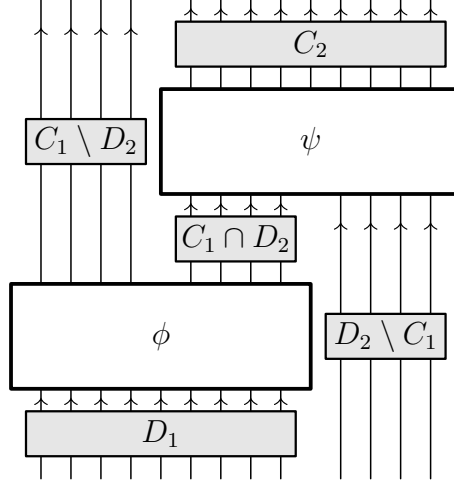


Figure 2.2: Visual mnemonic for extending morphisms. This graphic represents equation (2.17).

Remark 2.13. Associativity allows us to denote repeated multiplication by using extra indices. For instance: $m_\ell^{i,j,k} := m_r^{i,j} \parallel m_\ell^{r,k} = m_s^{j,k} \parallel m_\ell^{i,s}$.

There is also the dual notion of a coalgebra, which arises by reversing the arrows in equations (2.18) and (2.19):

Definition 2.14 (meta-coalgebra). A meta-coalgebra (or meta-comonoid) is a meta-object $\{C_X\}_X$ with operations a comultiplication $\Delta_{jk}^i: C_{\{i\}} \rightarrow C_{\{j,k\}}$ which is coassociative (equation (2.20)) and a counit, which is a map $\epsilon^i: A_i \rightarrow A_\emptyset$ satisfying equation (2.21).

$$\begin{array}{ccc}
 C_{\{1,2,3\}} & \xleftarrow{\Delta_{1,2}^1} & C_{\{2,3\}} \\
 \Delta_{2,3}^2 \uparrow & & \uparrow \Delta_{1,3}^1 \\
 C_{\{1,2\}} & \xleftarrow{\Delta_{1,2}^1} & C_{\{1\}}
 \end{array} \quad (2.20)$$

$$\begin{array}{ccc}
 C_{\{1\}} & \xleftarrow{\epsilon^2} & C_{\{1,2\}} \\
 \swarrow \text{id} & & \uparrow \Delta_{1,2}^1 \\
 & & C_{\{1\}}
 \end{array} \quad (2.21)$$

Remark 2.15. Coassociativity allows us to denote repeated comultiplication by using extra indices. For instance: $\Delta_{j,k,\ell}^i := \Delta_{j,r}^i \parallel \Delta_{k,\ell}^r = \Delta_{s,\ell}^i \parallel \Delta_{j,j}^s$.

If a meta-object $\{B_X\}_x$ has the structure of both an algebra and a coalgebra, we introduce a definition for when the structures are compatible with each other:

Definition 2.16 (meta-bialgebra). A meta-bialgebra (or meta-bimonoid) is a meta-algebra (B, m, η) and a meta-coalgebra (B, Δ, ϵ) , such that Δ and ϵ are meta-algebra morphisms. ^{†5}

$$\begin{array}{ccc}
 B_{\{1,2\}} & \xrightarrow{m_1^{1,2}} & B_{\{1\}} \\
 \downarrow \Delta_{1,3}^1 // \Delta_{2,4}^2 & & \downarrow \Delta_{1,2}^1 \\
 B_{\{1,2,3,4\}} & \xrightarrow{m_1^{1,2} // m_2^{3,4}} & B_{\{1,2\}}
 \end{array} \quad (2.22)$$

$$\begin{array}{ccc}
 B_{\{1,2\}} & \xrightarrow{m_1^{1,2}} & B_{\{1\}} \\
 \swarrow \epsilon^1 // \epsilon^2 & & \searrow \epsilon^1 \\
 & & B_\emptyset
 \end{array} \quad (2.24)$$

$$\begin{array}{ccc}
 & & B_{\{1\}} \\
 \eta_1 \nearrow & & \downarrow \Delta_{1,2}^1 \\
 B_\emptyset & & B_{\{1,2\}} \\
 \eta_1 // \eta_2 \searrow & &
 \end{array} \quad (2.23)$$

$$\begin{array}{ccc}
 B_\emptyset & \xrightarrow{\eta_1} & B_{\{1\}} \\
 \searrow \text{id} & & \downarrow \epsilon^1 \\
 & & B_\emptyset
 \end{array} \quad (2.25)$$

Remark 2.17. The conditions for Δ being an algebra morphism are presented in equations (2.22) and (2.23), while those for ϵ are in equations (2.24) and (2.25). Observing invariance under arrow reversal, it may not come as a surprise that equations (2.22) and (2.24) also are the conditions for m being a coalgebra morphism, and equations (2.23) and (2.25) tell us that η is as well.

Next, we introduce a notion of invertibility which extends a (meta-)bialgebra to a (meta-)Hopf algebra:

Definition 2.18 (meta-Hopf algebra). A meta-Hopf algebra (or meta-Hopf monoid) is a meta-bialgebra H together with a map $S: H \rightarrow H$ called the antipode, which satisfies $\Delta_{1,2}^1 // S_1^1 // m_1^{1,2} = \epsilon^1 // \eta_1 = \Delta_{1,2}^1 // S_2^2 // m_1^{1,2}$. As a commutative diagram, this looks like equation (2.26)

$$\begin{array}{ccccc}
 H_{\{1\}} & \xrightarrow{\epsilon^1} & H_\emptyset & \xrightarrow{\eta_1} & H_{\{1\}} \\
 \searrow \Delta_{1,2}^1 & & & & \nearrow m_1^{1,2} \\
 & & H_{\{1,2\}} & \xrightleftharpoons[S_1^1]{S_2^2} & H_{\{1,2\}}
 \end{array} \quad (2.26)$$

In order to do knot theory, we need an algebraic way to represent a crossing of two strands. This is accomplished by the \mathcal{R} -matrix:

^{†5} B_X inherits a meta-(co)algebra structure from B , given by $(B_X)_Y := B_{XY}$ and component-wise operations. The bialgebra structure on B_\emptyset is given by $m = \eta = \Delta = \epsilon = \text{id}$.

Definition 2.19 (quasitriangular meta-Hopf algebra). A quasitriangular meta-Hopf algebra (or quasitriangular meta-Hopf monoid) is a meta-Hopf algebra H , together with an invertible element $\mathcal{R}_{i,j} \in H_{i,j}$, called the \mathcal{R} -matrix, which satisfies the following properties: (we will denote the inverse by $\overline{\mathcal{R}}$)

$$\mathcal{R}_{13} \parallel \Delta_{12}^1 = \mathcal{R}_{13} \mathcal{R}_{24} \parallel m_3^{34} \quad (2.27)$$

$$\mathcal{R}_{13} \parallel \Delta_{23}^3 = \mathcal{R}_{13} \mathcal{R}_{42} \parallel m_1^{14} \quad (2.28)$$

$$\Delta_{21}^1 = \Delta_{12}^1 \mathcal{R}_{a_1, a_2} \overline{\mathcal{R}}_{p_1, p_2} \parallel m_1^{a_1, 1, p_1} \parallel m_2^{a_2, 2, p_2} \quad (2.29)$$

Definition 2.20 (Drinfeld element). In a quasitriangular meta-Hopf algebra H , the Drinfeld element, $u \in H$ is:

$$u := \mathcal{R}_{21} \parallel S_1^1 \parallel m^{12} \quad (2.30)$$

Definition 2.21 (monodromy). Each quasitriangular meta-Hopf algebra has a monodromy $Q_{12} := \mathcal{R}_{12} \mathcal{R}_{34} \parallel m_1^{14} \parallel m_2^{23}$. Its inverse will be denoted $\overline{Q}_{12} = \overline{\mathcal{R}}_{12} \overline{\mathcal{R}}_{34} \parallel m_1^{14} \parallel m_2^{23}$.

Definition 2.22 (ribbon meta-Hopf algebra). A quasitriangular meta-Hopf algebra H is called ribbon if it has an element $\nu \in Z(H)$ such that:

$$\nu_1 \nu_2 \parallel m^{12} = u_1 u_2 \parallel S_2^2 \parallel m^{12} \quad (2.31)$$

$$\nu_1 \parallel \Delta_{12}^1 = \nu_1 \nu_2 \parallel \overline{Q}_{34} \parallel m_1^{13} \parallel m_2^{24} \quad (2.32)$$

$$\nu \parallel S = \nu \quad (2.33)$$

$$\nu \parallel \epsilon = \eta \parallel \epsilon = 1 \quad (2.34)$$

Definition 2.23 (spinner). A spinner^{†6} in a ribbon meta-Hopf algebra H is an invertible element $C \in H$ (with inverse \overline{C}) such that for all $x \in H$:

$$C_1 \nu_2 C_3 \parallel S_2^2 \parallel m^{123} = \nu \quad (2.35)$$

$$C_1 \parallel \Delta_{12}^1 = C_1 C_2 \quad (2.36)$$

$$C \parallel S = \overline{C} \quad (2.37)$$

$$C_1 x_2 \overline{C}_3 \parallel m^{123} = x \parallel S \parallel S \quad (2.38)$$

$$C \parallel \epsilon = \eta \parallel \epsilon = 1 \quad (2.39)$$

†6 These are more commonly referred to as distinguished grouplike elements. The term we use is inspired by the tangle in figure 2.17.

Lemma 2.24 (spinners and ribbon Hopf algebras). *If a (meta-)Hopf algebra has either a ribbon element ν or a spinner C , then it must have the other as well, given by the formula: $C_1\nu_2 \parallel m^{12} = \mathbf{u}$.*

Proof. See Majid’s work in [Maj] or Etingof and Schiffmann in [ES] for more details on this standard result. Note that the proof does not rely on the additive structure of the Hopf algebra, which allows us to extend this result to the realm of meta-Hopf algebras. \square

2.4 THE META-ALGEBRA OF TANGLE DIAGRAMS

The particular structures introduced were chosen for their ability to represent the topological properties of knotted objects. We will now introduce the notion of a tangle and demonstrate its meta-algebraic structure.

Upright tangles

For our purposes, a tangle will be visualised as follows: take a stiff (topologically) circular metal frame forming a Jordan curve (i.e. with a defined inside and outside), then attach a collection of strings to the wire, ensuring that the strings always remain inside the wire, and that each string is tied to the metal frame in two unique locations (that is, no two strings share an endpoint).

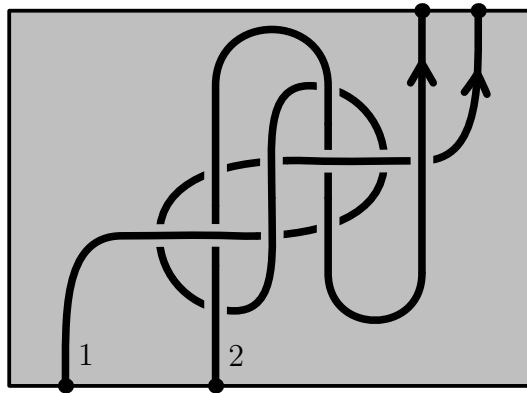


Figure 2.3: Example of a tangle with strands labelled 1 and 2.

Definition 2.25 (open tangle). An open tangle is an embedding of line segments (called components or strands) into a thickened topological disk

$D \times [-1, 1]$ (or a disjoint union of such disks) such that the endpoints of the line segments are fixed along $\partial D \times \{0\}$. Each strand is labelled with elements of a set X . Two open tangles are considered equivalent if there exists an isotopy of the embeddings which fixes the endpoints of the strands. The set of all tangles with strands indexed by X will be denoted \mathcal{T}_X . (The term “open” refers to the absence of closed loops.)

The objects which are more natural for us to study are tangles with a framing, which one may think of as open tangles with the strings replaced with thin ribbons.

Definition 2.26 (framed tangle). A framed tangle is an open tangle together with a choice of section of the normal bundle for each component, with endpoints of the section fixed pointing to the right of the tangent vector. This choice is taken up to endpoint-fixing homotopy. Unless otherwise mentioned, it will be assumed that all tangles are framed.

Observe that a generic projection of a tangle to its central core $D \times \{0\}$ results in the strands forming a graph, with each crossing of two strands in the tangle producing a vertex in the graph. By assigning to each vertex the sign of the corresponding crossing (either “positive” or “negative”), we end up with a combinatorial object which is equivalent to the original tangle.

Definition 2.27 (open tangle diagram). An open tangle diagram is a projection of a tangle onto its central core such that all the line segments are immersions which intersect both the boundary disk and the other strands transversally, together with an assignment of a sign to each strand intersection. Small open neighbourhoods of these intersections are called crossings, while the complement of the crossings is a collection of embedded line segments called arcs.

Two open tangle diagrams are considered equivalent if they differ by a finite sequence of Reidemeister moves, as outlined in figures 2.4 to 2.6

The rotation numbers of arcs will play a role in this thesis, so we will capture these data in the following way (as described in [BNvdVa]):

Definition 2.28 (upright open tangle diagram). An upright tangle diagram is a tangle diagram with the further requirement that the endpoints of each arc must have a vertical tangent vector, and each crossing must involve only curves with tangent vectors that point (diagonally) upwards. Here, each

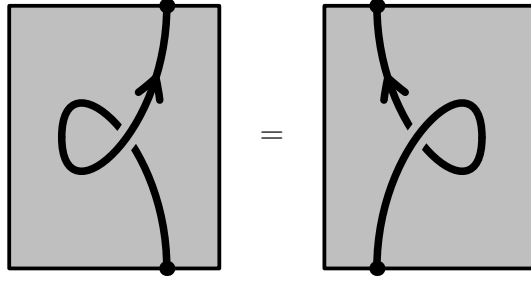


Figure 2.4: (Framed) Reidemeister move $R1'$

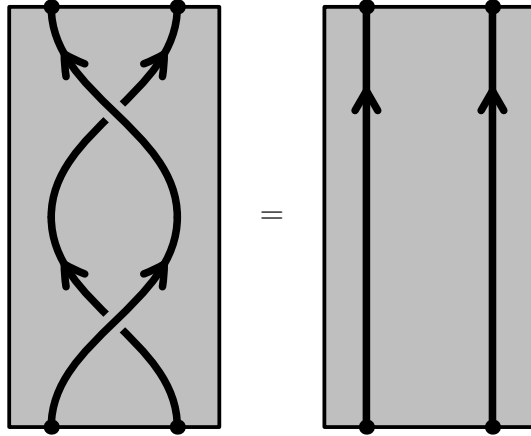


Figure 2.5: Reidemeister move $R2$

arc has well-defined integer rotation number. Two tangles are considered equivalent if they agree under the “rotational Reidemeister moves”, which are figures 2.4 to 2.8. Given a finite set X , the set of equivalence classes of X -indexed upright tangle diagrams will be denoted $\mathcal{T}_X^{\text{up}}$.

Remark 2.29. The concept of upright tangles was first introduced by Louis Kauffman in [Kau] under the name rotational virtual knot theory. In the formulation here, we insist that all strands end pointing upwards instead of merely requiring that endpoint vectors are vertical, so we will use the term “upright” to remind the reader of this difference.

Fortunately, ambient isotopy allows us to rotate any classical tangle into an upward-pointing form. Additionally, there is only one way to do this. We reproduce the proof of this fact by Bar-Natan and van der Veen in [BNvdVb] below:

Lemma 2.30 (tangles inject into upright tangles). *To each open tangle diagram D there exists an upright open tangle diagram D' obtained from*

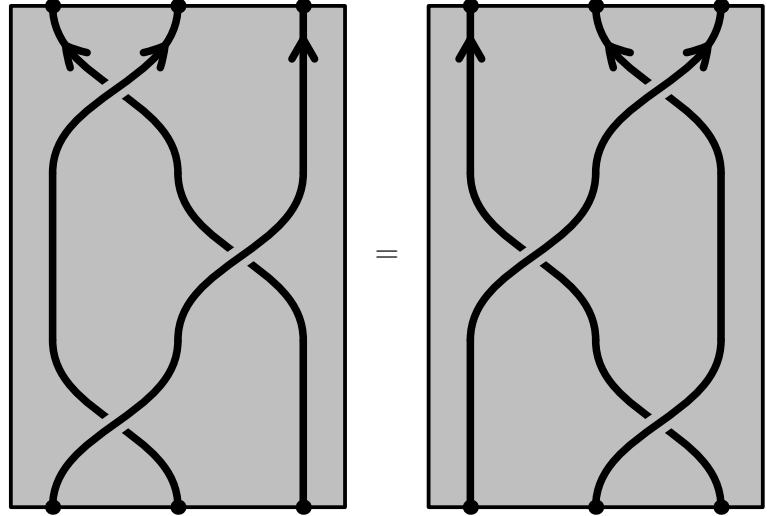


Figure 2.6: Reidemeister move $R3$

D by a planar isotopy. Further, if D'' is another such upright open tangle diagram obtained from D , then D' and D'' differ by a finite sequence of rotational Reidemeister moves and a change of rotation number at the endpoints.

Proof. Each arc and crossing in the diagram D may be rotated so that its endpoints are pointing upwards, giving rise to a diagram D' . Two (nonupright) tangle diagrams are equivalent when they differ by a finite sequence of Reidemeister moves. Each of these Reidemeister may also be rotated to an equivalence of upright tangles, each of which is given as a rotational Reidemeister move (figures 2.4 to 2.7). The last possibility is the rotation of an entire crossing, which is covered by figure 2.8. \square

The meta-algebra structure of upright tangle diagrams

We now formally connect tangle diagrams with meta-algebras.

Theorem 2.31 (tangles form a ribbon meta-Hopf algebra). *The collection $\{\mathcal{T}_X^{up}\}_X$ forms a ribbon meta-Hopf algebra (in the category **Set**) with the following operations:*

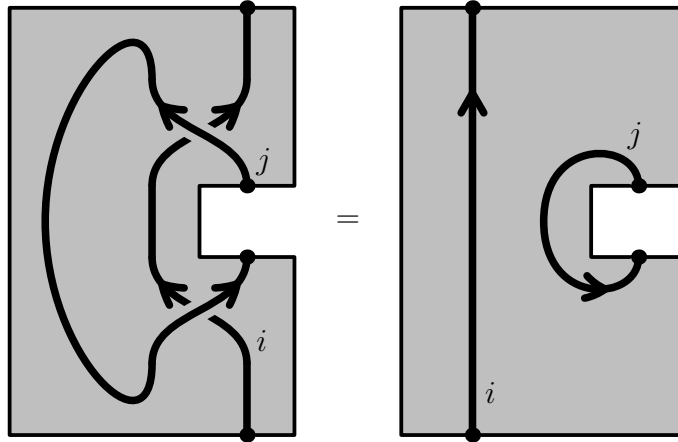


Figure 2.7: The (cyclic) rotational Reidemeister move $R2_{\text{rot}}$

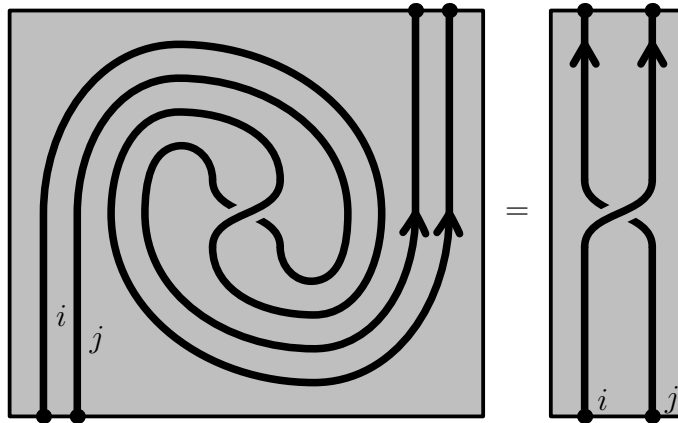


Figure 2.8: The whirling move

- multiplication $m_k^{ij}[X]$ takes a tangle with strands $X \sqcup \{i, j\}$ and glues the end of strand i to strand j , labelling the resulting strand k (figure 2.9).^{†7}
- the unit $\eta_i[X]$ takes a tangle diagram with strands X and introduces a new strand i which does not touch any of the other strands (figure 2.10).^{†8}

^{†7} Strictly speaking, this operation is only defined when the end of strand i is adjacent to strand j . See remark 2.32 for more details.

^{†8} In general, there are multiple choices for the placement of such a new strand. See remark 2.33 for more details.

- the comultiplication $\Delta_{jk}^i[X]$ takes a tangle with strands $X \sqcup \{i\}$ and doubles strand i , separating the two strands along the framing of strand i , calling the right strand j and the left one k (figure 2.12).^{†9}
- the counit $\epsilon^i[X]$ takes a tangle with strands indexed by $X \sqcup \{i\}$ and returns the tangle with strand labelled by i deleted (figure 2.11).
- The antipode $S_j^i[X]$ takes a tangle with strands labelled by $X \sqcup \{i\}$ and reverses the direction of strand i , then adds a counter-clockwise cap to the new beginning, and a clockwise cup to the end. This new strand is called j . When applied to a single vertical strand, the resulting tangle looks like the letter “S” (figure 2.13).
- the \mathcal{R} -matrix \mathcal{R}_{ij} is given by the two-strand tangle with a single positive crossing of strand i over strand j . The inverse \mathcal{R} -matrix $\overline{\mathcal{R}}_{ij}$ is the two-strand tangle with a negative crossing of strand i over strand j (figure 2.14).
- The spinner $C_i[X]$ takes a tangle in $\mathcal{T}_X^{\text{up}}$ and adds a new strand with rotation number 1 which has no interactions with any other strands. This new strand looks like the letter “C” (figure 2.17).

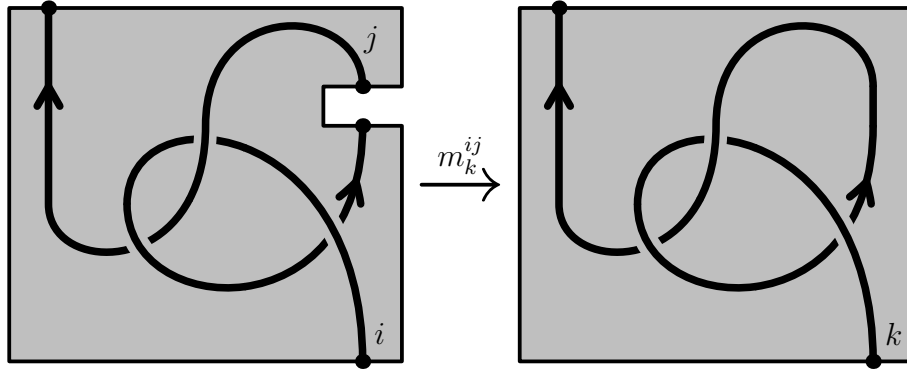


Figure 2.9: Multiplication m_k^{ij} stitches two strands in a tangle together.

Proof. We go through a straightforward verification of the various axioms:

1. We show $\mathcal{T}_X^{\text{up}}$ is a meta-algebra:

^{†9} While this convention appears unfortunate, we follow the notation laid out in [BNvdVb] so that the antipode and spinner have a more memorable representation, namely looking like the letters they are represented by (see figures 2.13 and 2.17 for the resemblance).

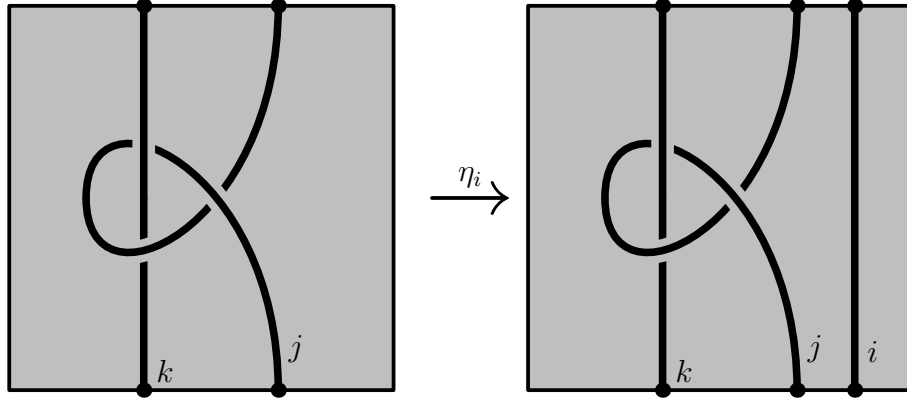


Figure 2.10: The unit η_i introduces a new strand in a tangle.

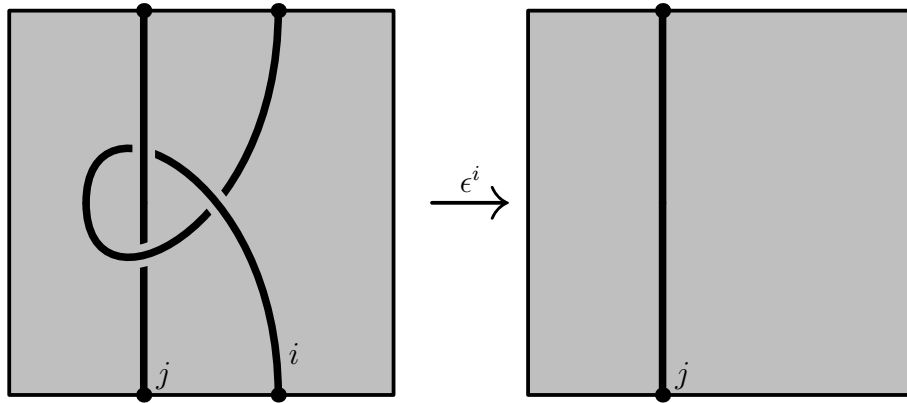


Figure 2.11: The counit ϵ^i deletes a strand in a tangle.

- a) To establish associativity (equation (2.18)), notice that when stitching a sequence of strands together in a fixed order, the order in which the individual stitches are made does not affect the resulting strand; the resulting strand has the same endpoints and follows the same path.
- b) To see that the unit axiom (equation (2.19)) holds, observe that adding a non-interacting strand to a diagram, then stitching it to an existing strand (without loss of generality, we may assume that the new strand is placed adjacent to the existing one) does not change any of the combinatorial data, resulting in identical diagrams.

2. We show $\mathcal{T}_X^{\text{up}}$ is a meta-coalgebra:

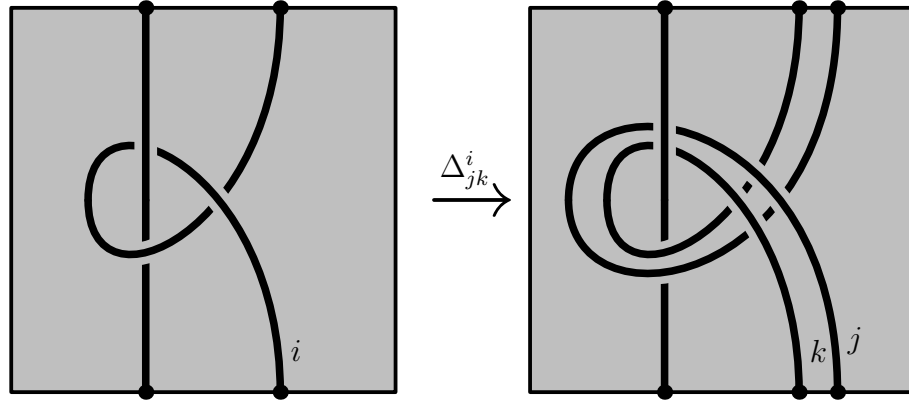


Figure 2.12: The comultiplication Δ_{jk}^i doubles a strand in a tangle along its framing. Notice the right-to-left strand labels.

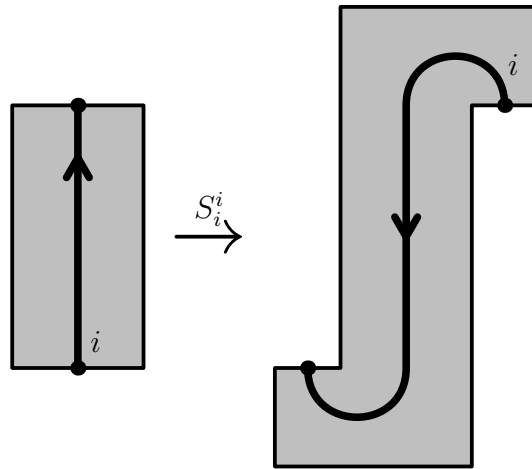


Figure 2.13: The antipode S_i^i reverses a strand, rotating the endpoints to maintain an upright tangle.

- a) Establishing coassociativity (equation (2.20)) amounts to the same argument that cutting a strip of paper into three parallel strips does not depend on the order of cutting.
 - b) The counit identity (equation (2.21)) states deleting a strand is the same operation as first doubling it, then deleting both resulting strands.
3. The meta-bialgebra axioms we verify next:
- a) Equation (2.22) states that if two strands are stitched together, then the resulting strand is doubled, this could have equivalently

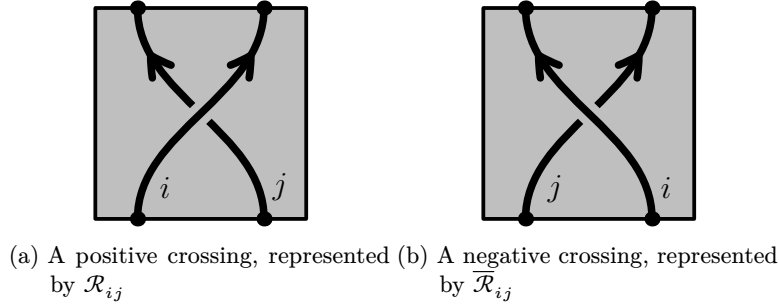


Figure 2.14: The \mathcal{R} -matrix and its inverse represent a tangle with a single crossing.

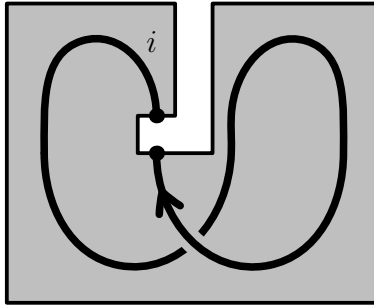


Figure 2.15: The Drinfeld element u_i in the meta-Hopf algebra of tangles.

- been achieved by doubling each of the original strands, then performing a stitching on both resulting pairs of strands.
- b) Equation (2.24) simply states that stitching two strands together, then removing the resulting strand could have equally been achieved by removing both of the original strands without stitching them first.
 - c) Equation (2.25) states that introducing a strand, then immediately removing it is the identity operation.
 - d) Equation (2.23) says that doubling a newly-introduced (and therefore free of crossings) strand is the same operation as introducing two strands separately. (For those worried that this equation depends on the location of the separately introduced strands, this is one place that the use of virtual tangles will be used, which does not heed the relative locations of disjoint strands.)
4. Next, we show $\mathcal{J}_X^{\text{up}}$ is a meta-Hopf algebra. Equation (2.26) states that when a strand is doubled, then one of the two strands is reversed, multiplying the two strands together results in a strand which can be

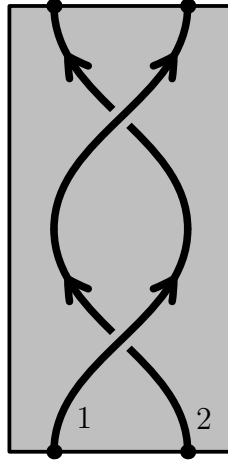
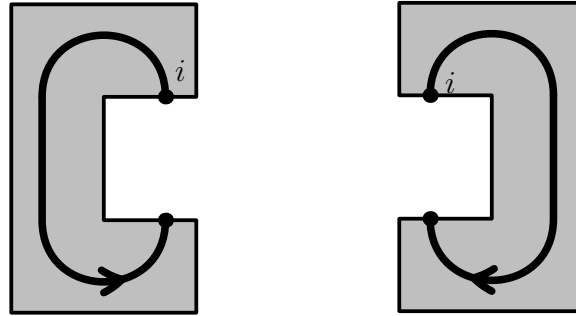


Figure 2.16: The monodromy in the meta-Hopf algebra of tangles.



(a) The spinner C_i has rotation number 1. (b) The inverse spinner \overline{C}_i has rotation number -1 .

Figure 2.17: The spinners represent strands with a unit rotation number.

rearranged to not interact with any of the other strands. This can be readily seen, as this newly-created strand looks like a snake weaving through the tangle diagram. One can remove the snake by applying a series of Reidemeister 2 moves, resulting in a strand disjoint from the rest of the diagram. This is the same as deleting the original strand, then introducing a new disjoint one.

5. The quasitriangular axioms are equalities of pairs of three-strand tangles:

- a) Equations (2.27) and (2.28) tell us that doubling a strand involved in a single crossing can also be built by adjoining two crossings together. This is visualized in figures 2.19 and 2.20.

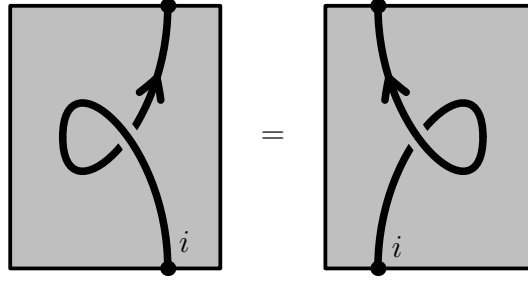


Figure 2.18: A ribbon element ν_i in the meta-Hopf algebra of tangles. One can use lemma 2.24 to verify this is compatible with the spinner.

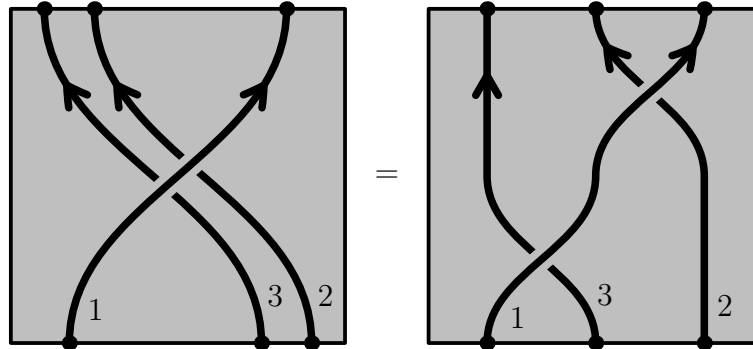


Figure 2.19: Example of a tangle satisfying equation (2.28)

b) Equation (2.29) tells us that we can swap the order of a doubled strand by adding crossings to either end (reminiscent of a Reidemeister 2 move).

6. Next, we establish that $\mathcal{T}_X^{\text{up}}$ is ribbon. Using lemma 2.24, it is enough to verify the spinner axioms (equations (2.35) to (2.39)). All these axioms have corresponding pictures one can draw, keeping in mind the orientations in the definitions of the relevant operations.

Finally, we observe that the Reidemeister moves do not introduce any new relations. Reidemeister 2 follows from the invertibility of the \mathcal{R} -matrix. Next, it is readily seen that the quasitriangular relations governing the \mathcal{R} -matrix force it to solve the Yang-Baxter equation, which is one equivalent to the Reidemeister 3 in this case. \square

Remark 2.32. One may object that strand-stitching m_k^{ij} is not defined when the endpoint of strand i cannot be connected to the starting point of strand j using only an upward-pointing arc, and the two points are adjacent in the

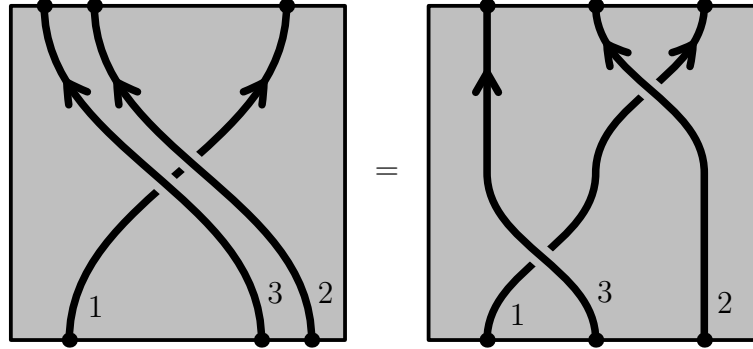


Figure 2.20: Example of a tangle satisfying equation (2.27)

cyclic ordering induced by the boundary of the disk. This issue is resolved in multiple ways:

1. We may extend the collection of tangles we work with to include virtual tangles. This generalization of tangles deals exactly with the issue that multiplication need not produce a planar tangle diagram. In fact, virtual tangles can be thought of as merely non-planar tangle diagrams.
2. We may commit to only apply multiplication when doing so would result in a valid (classical) tangle. This is the approach we will take when performing computations on tangles, though we emphasize that the true context we are working in is that of virtual tangles.

Remark 2.33. One may object that inserting a strand η_i is not well-defined. This issue is resolved in multiple ways:

1. We may extend the collection of tangles we work with to include virtual tangles, which encode only the combinatorial information of a tangle, so the placement of a new strand is not relevant.
2. We may commit to place the new strand in one location; the relations above will suffice to define tangles. Moving strands' intersection with the boundary disk does not play a role in deciding the equivalence of two classical links.

2.5 THE META-ALGEBRA U

Here we define the ribbon Hopf algebra $U \cong \widehat{\mathfrak{U}}(\mathfrak{sl}_{2+}^0)$, and point out some of its properties.

Definition 2.34 (The ribbon Hopf algebra U). Define the Lie algebra

$$\mathfrak{g} := \text{span}_{\mathbb{Q}} \left\{ \mathbf{y}, \mathbf{b}, \mathbf{a}, \mathbf{x} \mid [\mathbf{a}, \mathbf{x}] = \mathbf{x}, [\mathbf{a}, \mathbf{y}] = -\mathbf{y}, [\mathbf{x}, \mathbf{y}] = \mathbf{b}, [\mathbf{b}, \cdot] = 0 \right\} \quad (2.40)$$

We put a grading on \mathfrak{g} by $\deg(y) = \deg(b) = 1$ and $\deg(a) = \deg(x) = 0$. Then the algebra U is defined to be the graded completion of the universal enveloping algebra $\widehat{\mathfrak{U}}(\mathfrak{g})$. With $\mathbf{B} := e^{-\mathbf{b}}$, the bialgebra structure of U is:

$$\begin{aligned} \Delta_{i,j}(\mathbf{y}) &= \frac{\mathbf{b}_i + \mathbf{b}_j}{1 - \mathbf{B}_i \mathbf{B}_j} \left(\mathbf{B}_j \frac{1 - \mathbf{B}_i}{\mathbf{b}_i} \mathbf{y}_i + \frac{1 - \mathbf{B}_j}{\mathbf{b}_j} \mathbf{y}_j \right) \\ \Delta_{i,j}(\mathbf{b}) &= \mathbf{b}_i + \mathbf{b}_j \\ \Delta_{i,j}(\mathbf{a}) &= \mathbf{a}_i + \mathbf{a}_j \\ \Delta_{i,j}(\mathbf{x}) &= \mathbf{x}_i + \mathbf{x}_j \end{aligned} \quad (2.41)$$

For any $\mathbf{z} \in \{\mathbf{y}, \mathbf{b}, \mathbf{a}, \mathbf{x}\}$, we have $\epsilon(\mathbf{z}) = 0$ (extended multiplicatively by equation (2.24)).

Next, we define the Hopf algebra structure by defining the antipode, which is defined as $S(\mathbf{z}) := -\mathbf{z}$ for each $\mathbf{z} \in \{\mathbf{y}, \mathbf{b}, \mathbf{a}, \mathbf{x}\}$, extended antimultiplicatively.

Next, we introduce the ribbon structure of U with an \mathcal{R} -matrix and the spinner C :

$$\mathcal{R}_{i,j} := \exp(\mathbf{b}_i \mathbf{a}_j) \exp\left(\frac{1 - \mathbf{B}_i}{\mathbf{b}_i} \mathbf{y}_i \mathbf{x}_j\right) \quad (2.42)$$

$$C := \sqrt{\mathbf{B}} \quad (2.43)$$

$$\nu := \overline{\mathcal{R}}_{31} \overline{C}_2 \parallel m^{123} = \overline{\mathcal{R}}_{13} C_2 \parallel m^{123} \quad (2.44)$$

Remark 2.35. The algebra U is isomorphic to the algebra $\widehat{\mathfrak{U}}(\mathfrak{sl}_{2+}^0)$ defined in [BNvdVa, §4] when $\epsilon = 0$ under the following change-of-variables:

$$\begin{aligned} \mathbb{D} &\xrightarrow{\sim} U \\ \mathbf{y} &\mapsto \frac{1 - \mathbf{B}}{\mathbf{b}} \mathbf{y} \\ \mathbf{b} &\mapsto \mathbf{b} \\ \mathbf{a} &\mapsto \mathbf{a} \\ \mathbf{x} &\mapsto \mathbf{x} \end{aligned} \tag{2.45}$$

This change-of-variables was implemented to facilitate the pullback of the \mathcal{R} -matrix structure of QU , while maintaining cocommutativity.

Lemma 2.36 (Commutation relations in U). *Given $f \in \mathbb{Q}[\mathbf{a}]$, we have the following relations in U :*

$$f(\mathbf{a})\mathbf{y}^r = \mathbf{y}^r f(\mathbf{a} - r) \quad \mathbf{x}^r f(\mathbf{a}) = f(\mathbf{a} - r)\mathbf{x}^r \tag{2.46}$$

These relations also hold when $f(a)$ is replaced with $f(a, b) \in \mathbb{Q}[\mathbf{a}][[\mathbf{b}]]$.

Proof. We begin by commuting a single “ \mathbf{a} ” past a power of \mathbf{y} :

$$\mathbf{a}\mathbf{y}^r = \mathbf{y}(\mathbf{a} - 1)\mathbf{y}^{r-1} = \mathbf{y}^2(\mathbf{a} - 2)\mathbf{y}^{r-2} = \dots = \mathbf{y}^r(\mathbf{a} - r) \tag{2.47}$$

Repeating the above process for multiple copies of a yields:

$$\mathbf{a}^k \mathbf{y}^r = \mathbf{a}^{k-1}(\mathbf{a}\mathbf{y}^r) = \mathbf{a}^{k-1}\mathbf{y}^r(\mathbf{a} - r) = \mathbf{a}^{k-2}\mathbf{y}^r(\mathbf{a} - r)^2 = \dots = \mathbf{y}^r(\mathbf{a} - r)^k \tag{2.48}$$

Equation (2.48) establishes a linear relation, so the result extends to any polynomial in \mathbf{a} , and thereby to any element of $\mathbb{Q}[\mathbf{a}][[\mathbf{b}]]$.

A similar argument exists to show $\mathbf{x}^r f(\mathbf{a}) = f(\mathbf{a} - r)\mathbf{x}^r$. \square

Next, we observe that since $[\mathbf{x}, \mathbf{y}] = \mathbf{b}$ is central, the Weyl canonical commutation relation holds:

Lemma 2.37 (Weyl canonical commutation relation). *In the ring $U[[\xi, \eta]]$, we have the following relation:*

$$\mathbf{e}^{\xi\mathbf{x}}\mathbf{e}^{\eta\mathbf{y}} = \mathbf{e}^{\xi\eta\mathbf{b}}\mathbf{e}^{\eta\mathbf{y}}\mathbf{e}^{\xi\mathbf{x}} \tag{2.49}$$

Proof. See, Weyl’s [Wey] or Hall’s [Hal] for details on this result. \square

We have further commutation relations with exponentials:

Lemma 2.38 (Exponential commutation relations in U). *In the ring $U[[\alpha, \eta, \xi]]$, let $\mathcal{A} := e^\alpha$. Then we have:*

$$e^{\alpha \mathbf{a}} e^{\eta \mathbf{y}} = e^{\frac{\eta}{\mathcal{A}} \mathbf{y}} e^{\alpha \mathbf{a}} \quad (2.50)$$

$$e^{\xi \mathbf{x}} e^{\alpha \mathbf{a}} = e^{\alpha \mathbf{a}} e^{\frac{\xi}{\mathcal{A}} \mathbf{x}} \quad (2.51)$$

Proof. Using equation (2.46), we notice

$$e^{\alpha \mathbf{a}} e^{\eta \mathbf{y}} = e^{\alpha \mathbf{a}} \sum_n \frac{(\eta \mathbf{y})^n}{n!} = \sum_n \frac{(\eta \mathbf{y})^n}{n!} e^{\alpha(\mathbf{a}-n)} = e^{\frac{\eta}{\mathcal{A}} \mathbf{y}} e^{\alpha \mathbf{a}} \quad (2.52)$$

similarly,

$$e^{\xi \mathbf{x}} e^{\alpha \mathbf{a}} = \sum_n \frac{(\xi \mathbf{x})^n}{n!} e^{\alpha \mathbf{a}} = \sum_n e^{\alpha(\mathbf{a}-n)} \frac{(\xi \mathbf{x})^n}{n!} = e^{\alpha \mathbf{a}} e^{\frac{\xi}{\mathcal{A}} \mathbf{x}} \quad (2.53)$$

□

Lemma 2.39 (the algebra U is ribbon). *The algebra U has a ribbon structure given by the above \mathcal{R} -matrix and spinner C .*

Proof. This follows from [BNvdVb, theorem 29]. For completeness, we produce a more direct proof:

The Hopf algebra structure of U is straightforward, and is left to the reader to verify. We will focus our attention on verifying quasitriangularity and the ribbon structure.

Let us verify equation (2.28) first. The left-hand side is:

$$\mathcal{R}_{12} // \Delta_{23}^2 = \exp(\mathbf{b}_1(\mathbf{a}_2 + \mathbf{a}_3)) \exp\left(\frac{1 - \mathbf{B}_1}{\mathbf{b}_1} \mathbf{y}_1(\mathbf{x}_2 + \mathbf{x}_3)\right) \quad (2.54)$$

Equality with the right-hand side follows by commutativity of \mathbf{b}_1 and \mathbf{y}_1 :

$$\begin{aligned} & \mathcal{R}_{13} \mathcal{R}_{42} // m_1^{14} \\ &= \exp(\mathbf{b}_1 \mathbf{a}_3) \exp\left(\frac{1 - \mathbf{B}_1}{\mathbf{b}_1} \mathbf{y}_1 \mathbf{x}_3\right) \exp(\mathbf{b}_1 \mathbf{a}_2) \exp\left(\frac{1 - \mathbf{B}_1}{\mathbf{b}_1} \mathbf{y}_1 \mathbf{x}_2\right) \quad (2.55) \\ &= \exp(\mathbf{b}_1(\mathbf{a}_2 + \mathbf{a}_3)) \exp\left(\frac{1 - \mathbf{B}_1}{\mathbf{b}_1} \mathbf{y}_1(\mathbf{x}_2 + \mathbf{x}_3)\right) \end{aligned}$$

Next we verify equation (2.27), whose left-hand side is:

$$\mathcal{R}_{13} // \Delta_{12}^1 = \exp((\mathbf{b}_1 + \mathbf{b}_2)\mathbf{a}_3) \exp\left(\left(\mathbf{B}_2 \frac{1 - \mathbf{B}_1}{\mathbf{b}_1} y_1 + \frac{1 - \mathbf{B}_2}{\mathbf{b}_2} y_2\right) \mathbf{x}_3\right) \quad (2.56)$$

On the right-hand side, we have

$$\begin{aligned} & \mathcal{R}_{13} \mathcal{R}_{24} // m_3^{34} \\ &= \exp(\mathbf{b}_1 \mathbf{a}_3) \exp\left(\frac{1 - \mathbf{B}_1}{\mathbf{b}_1} y_1 \mathbf{x}_3\right) \exp(\mathbf{b}_2 \mathbf{a}_3) \exp\left(\frac{1 - \mathbf{B}_2}{\mathbf{b}_2} y_2 \mathbf{x}_3\right) \quad (2.57) \\ &= \exp((\mathbf{b}_1 + \mathbf{b}_2)\mathbf{a}_3) \exp\left(\frac{1 - \mathbf{B}_1}{\mathbf{b}_1} \mathbf{B}_2 y_1 \mathbf{x}_3\right) \exp\left(\frac{1 - \mathbf{B}_2}{\mathbf{b}_2} y_2 \mathbf{x}_3\right) \end{aligned}$$

We use lemma 2.38 to write the expression in a canonical order. Finally, the right two exponentials may be combined since each variable commutes with the others, either by belonging to separate tensor factors, or in the case of \mathbf{b} , being central. The verifications of equation (2.29) and equations (2.35) to (2.39) follow with similar computations. \square

2.6 MORPHISMS BETWEEN META-OBJECTS

We can define a tangle invariant by considering a morphism between the meta-structure of tangles and that of an algebraic object. We now define a morphism between meta-objects:

Definition 2.40 (morphism of meta-objects). Let $\{A_X\}_X$ and $\{B_X\}_X$ be compatible meta-objects (i.e. ones with the same operations and relations between the operations). A morphism Φ between these meta-objects is a map $\Phi: \{A_X\}_X \rightarrow \{B_X\}_X$ sending $A_X \mapsto B_X$ such that each operation ϕ_Y^X in A intertwines with that in B :

$$\Phi(\phi_Y^X) = \phi_Y^X \quad (2.58)$$

Upright tangle invariants from a ribbon meta-Hopf algebra

Given a ribbon-Hopf algebra, there is an associated universal invariant, called the Reshetikhin–Turaev invariant (introduced in [RT]). What follows is an outline of the definition of the universal invariant Z for U :

1. Given a open tangle, disconnect each crossing from its neighbours, as well as each arc with a nonzero rotation number.
2. Replace each crossing with an \mathcal{R} -matrix $\mathcal{R}_{ij} \in U_{\{i,j\}}$, and each rotation of an arc with a spinner $C_i \in U_{\{i\}}$.
3. For each disconnection, there is a corresponding stitching operation required to bring the tangle back to its original state. Replace each stitching operation with a multiplication operation in U .

Figure 2.21 provides an example of this process. There, the meta-morphism Z sends the right-hand tangle to $\mathcal{R}_{3,7}\mathcal{R}_{6,2}\mathcal{R}_{1,5}\overline{C}_4$. By equation (2.58), the image of the left-hand tangle (which is the knot 3_1) under Z is $\mathcal{R}_{3,7}\mathcal{R}_{6,2}\mathcal{R}_{1,5}\overline{C}_4 // m_i^{1,2,\dots,7}$. As an algebra element, this is

$$\begin{aligned} \exp(\mathbf{b}_3 \mathbf{a}_7) \exp\left(\frac{1 - \mathbf{B}_3}{\mathbf{b}_3} \mathbf{y}_3 \mathbf{x}_7\right) \exp(\mathbf{b}_6 \mathbf{a}_2) \exp\left(\frac{1 - \mathbf{B}_6}{\mathbf{b}_6} \mathbf{y}_6 \mathbf{x}_2\right) & \quad (2.59) \\ \cdot \exp(\mathbf{b}_1 \mathbf{a}_5) \exp\left(\frac{1 - \mathbf{B}_1}{\mathbf{b}_1} \mathbf{y}_1 \mathbf{x}_5\right) \mathbf{B}_4^{-1/2} // m_i^{1,2,\dots,7} & \end{aligned}$$

In order to meaningfully compare such expressions, we must first write them in a canonical form. The form we use here is to order monomials so that they are of the form $\mathbf{y}^{n_1} \mathbf{b}^{n_2} \mathbf{a}^{n_3} \mathbf{x}^{n_4}$. This may always be done through the use of the relations with the Lie bracket. Doing so efficiently is a difficult task, and will be addressed in the following chapter.

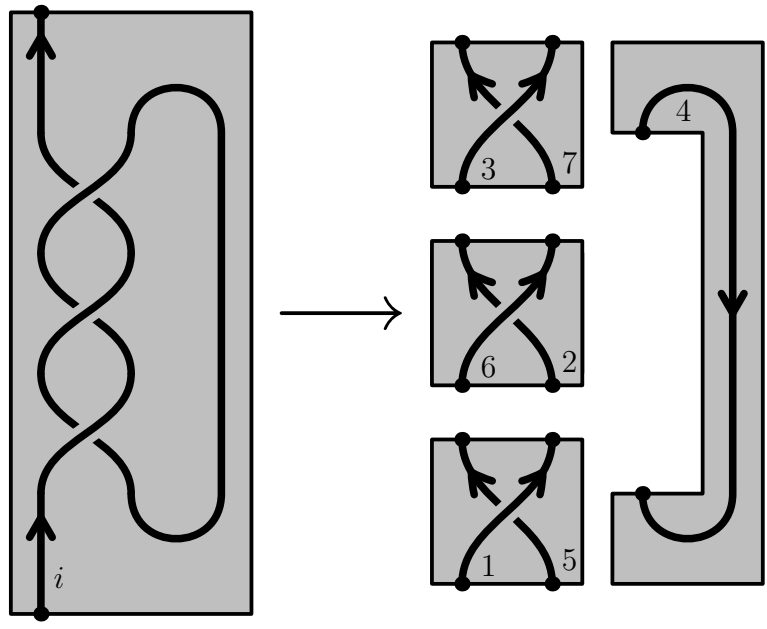


Figure 2.21: Breaking up a tangle into its constituent components.

PERTURBED GAUSSIANS

We now summarize the work of Bar-Natan and van der Veen in [BNvdVb], which uses perturbed Gaussians to compute Z quickly.

3.1 EXPRESSING MORPHISMS AS GENERATING FUNCTIONS

In order to be able to compute Z efficiently, we need an effective way to reduce expressions in U to a closed form. In [BNvdVb], Bar-Natan and van der Veen achieve this by compactly encoding operations. We outline their formulation in what follows.

To describe \mathbb{k} -linear maps between tensor powers of the algebra U , we define some categories \mathcal{U} , \mathcal{H} , and \mathcal{C} , each with objects given by finite sets. For the first two categories, their homsets are given by:

$$\mathrm{Hom}_{\mathcal{U}}(J, K) := \mathrm{Hom}_{\mathbb{k}}(U_J, U_K) \quad (3.1)$$

$$\mathrm{Hom}_{\mathcal{H}}(J, K) := \mathrm{Hom}_{\mathbb{k}}(\mathbb{Q}[z_J], \mathbb{Q}[z_K]) \quad (3.2)$$

where z_X is a shorthand for all the variables y_i , b_i , a_i , and x_i for all $i \in X$. Equations (3.1) and (3.2) explicitly denote vector space maps, not just algebra or ring homomorphisms.

By the PBW theorem, we know that U is isomorphic as a vector space to the polynomial ring $\mathbb{Q}[y, b, a, x]$ by choosing an ordering of the generators (following [BNvdVb], we use $(\mathbf{y}, \mathbf{b}, \mathbf{a}, \mathbf{x})$):

$$\begin{aligned} \mathbb{O}: \mathbb{Q}[y, b, a, x] &\xrightarrow{\sim} U \\ y^{n_1} b^{n_2} a^{n_3} x^{n_4} &\mapsto \mathbf{y}^{n_1} \mathbf{b}^{n_2} \mathbf{a}^{n_3} \mathbf{x}^{n_4} \end{aligned} \quad (3.3)$$

This map extends to a linear isomorphism of the homsets of \mathcal{H} and \mathcal{U} , inducing an isomorphism of categories $\mathbb{O}: \mathcal{H} \xrightarrow{\sim} \mathcal{U}$. By this ordering map, we can work with linear maps between polynomial spaces instead of with linear maps between the non-commuting polynomials of U .

Next we seek to better understand linear maps between polynomial spaces. Bar-Natan and van der Veen use generating functions, whose definition we reproduce below:

For A and B finite sets, consider the set $\text{Hom}(\mathbb{Q}[z_A], \mathbb{Q}[z_B])$ of linear maps between multivariate polynomial rings. Such a map is determined by its values on the monomials $z_A^{\mathbf{n}}$ for each multi-index $\mathbf{n} \in \mathbb{N}^A$.

Definition 3.1 (Exponential generating function). Given a linear map $\Phi: \mathbb{Q}[z_A] \rightarrow \mathbb{Q}[z_B]$ between polynomial spaces, its exponential generating function is:

$$\mathcal{G}(\Phi) := \sum_{\mathbf{n} \in \mathbb{N}^A} \frac{\Phi(z_A^{\mathbf{n}})}{\mathbf{n}!} \zeta_A^{\mathbf{n}} \in \mathbb{Q}[z_B][[\zeta_A]] \tag{3.4}$$

We also extend the domain of Φ from $\mathbb{Q}[z_A]$ to $\mathbb{Q}[z_A][[\zeta_A]]$ via extension of scalars $\Phi \mapsto \Phi \otimes \text{id}_{\mathbb{Q}[[\zeta_A]]}$. This extension allows us to write the exponential generating function in a cleaner way:

$$\mathcal{G}(\Phi) = \Phi \left(\sum_{\mathbf{n} \in \mathbb{N}^A} \frac{(z_A \zeta_A)^{\mathbf{n}}}{\mathbf{n}!} \right) = \Phi(\mathcal{G}(\text{id}_{\mathbb{Q}[z_A]})) \tag{3.5}$$

We may now define the homsets for the category \mathcal{C} of generating functions. Given finite sets J and K we set:

$$\text{Hom}_{\mathcal{C}}(J, K) := \mathbb{Q}[z_K][[\zeta_J]] \tag{3.6}$$

Composition of generating functions $\mathcal{T} \in \text{Hom}_{\mathcal{C}}(J, K)$ and $\mathcal{S} \in \text{Hom}_{\mathcal{C}}(K, L)$ is defined as:

$$\mathcal{T} // \mathcal{S} := \left(\mathcal{T} \Big|_{z_K \mapsto \partial_{\zeta_K}} \mathcal{S} \right) \Big|_{\zeta_K=0} \tag{3.7}$$

The map \mathcal{G} intertwines the compositions on \mathcal{C} and \mathcal{H} . On \mathcal{C} , the pullback of composition through \mathcal{G}^{-1} takes the following form (quoted from [BNvdVb, Lemma 3]):

Lemma 3.2 (Composition of generating functions). *Suppose J, K, L are finite sets and $\phi \in \text{Hom}(\mathbb{Q}[z_J], \mathbb{Q}[z_K])$ and $\psi \in \text{Hom}(\mathbb{Q}[z_K], \mathbb{Q}[z_L])$. We have*

$$\mathcal{G}(\phi // \psi) = \left(\mathcal{G}(\phi) \Big|_{z_K \mapsto \partial_{\zeta_K}} \mathcal{G}(\psi) \right) \Big|_{\zeta_K=0} \tag{3.8}$$

The maps $\mathbb{O}: \mathcal{H} \xrightarrow{\sim} \mathcal{U}$ and $\mathcal{G}: \mathcal{H} \xrightarrow{\sim} \mathcal{C}$ as introduced in equations (3.3) and (3.4) induce linear isomorphisms between the corresponding homsets, which extend to monoidal isomorphisms.

We use this formulation because of the existence of a computationally amenable subcategory of \mathcal{C} which contains the image of this invariant. Instead of considering all power series in \mathcal{C} , we may instead work with a subset which is much easier to compute with.

Since the composition as in equation (3.7) will occur several times, we will use the notion of contraction used by Bar-Natan and van der Veen (taken from [BNvdVb, Definition 4]):

Definition 3.3 (Contraction). Let $f \in \mathbb{k}[[r, s]]$ be a power series. The contraction of $f = \sum_{k,l} c_{k,l} r^k s^l$ along the pair (r, s) is:

$$\langle f \rangle_{(r,s)} := \sum_k c_{k,k} k! = \sum_{k,l} c_{k,l} \partial_s^k s^l \Big|_{s=0} \tag{3.9}$$

Further, this notation is to be extended to allow for multiple consecutive contractions for $f \in \mathbb{k}[[r_i, s_i]]_{i \leq n}$:

$$\langle f \rangle_{((r_i)_i, (s_i)_i)} := \left\langle \left\langle \langle f \rangle_{(r_1, s_1)} \right\rangle_{(r_2, s_2)} \cdots \right\rangle_{(r_n, s_n)} \tag{3.10}$$

Using this notation, we write the extended composition notation introduced in equation (2.17) as:

$$\mathcal{G}(\phi_{C_1}^{D_1} // \psi_{C_2}^{D_2}) = \left\langle \mathcal{G}(\phi_{C_1}^{D_1}) \mathcal{G}(\psi_{C_2}^{D_2}) \right\rangle_{(z_{C_1 \cap D_2}, \zeta_{C_1 \cap D_2})} \tag{3.11}$$

where z_X and ζ_X are shorthand for y_i, b_i, a_i, x_i and $\eta_i, \beta_i, \alpha_i, \xi_i$ respectively.

It is important to note that contraction does not always define a convergent expression. We will focus our attention on cases when convergence is well-defined, and especially those where the computation is accessible.

The theorem we will rely heavily on in this thesis is the following, taken from [BNvdVb, Theorem 6]:

Theorem 3.4 (Contraction theorem). *For any $n \in \mathbb{N}$, consider the ring $R_n = \mathbb{Q}[r_j, g_j][[s_j, W_{ij}, f_j \mid 1 \leq i, j \leq n]]$. Then*

$$\langle e^{gs+rf+rWs} \rangle_{r,s} = \det(\tilde{W}) e^{g\tilde{W}f} \tag{3.12}$$

where $\tilde{W} = (1 - W)^{-1}$.

The main takeaway of this theorem is this: morphisms whose generating functions are Gaussians have a clean formula for composition. Furthermore, this formula is computationally reasonable, growing only polynomially in complexity with n . This is contrasted with the conventional approach of choosing a representation V of U . When one considers morphisms between large tensor powers $V^{\otimes n}$, the computational complexity is exponential in n .

Expressing ribbon Hopf algebra operations as perturbed Gaussians

Using this vector space isomorphism, [BNvdVb] expresses all ribbon Hopf algebra operations as power series in a closed form, namely as perturbed Gaussians.

Theorem 3.5 (The ribbon meta-Hopf structure of U is Gaussian). *Each of the ribbon meta-Hopf algebra operations for U as defined in section 2.5 all have the form of a perturbed Gaussian. That is, when the generators (y, b, a, x) are assigned weights of $(1, 0, 2, 1)$ respectively, and their dual variables $(\eta, \beta, \alpha, \xi)$ are assigned complementary weights so that $\text{wt } z + \text{wt } \zeta = 2$, we*

have the following expressions which are either Gaussian or are generating functions of central elements (with $B = e^{-b}$ and $A = e^a$):

$$\mathcal{G}(m_k^{ij}) = \exp\left((\alpha_i + \alpha_j)a_k + (\beta_i + \beta_j + \xi_i\eta_j)b_k + \left(\frac{\xi_i}{A_j} + \xi_j\right)x_k + \left(\frac{\eta_j}{A_i} + \eta_i\right)y_k\right) \quad (3.13)$$

$$\mathcal{G}(\eta_i) = 1 \quad (3.14)$$

$$\mathcal{G}(\Delta_{jk}^i) = \exp\left(\eta_i \frac{b_j + b_k}{1 - B_j B_k} \left(B_k \frac{1 - B_j}{b_j} y_j + \frac{1 - B_k}{b_k} y_k\right) + \beta_i(b_j + b_k) + \alpha_i(a_j + a_k) + \xi_i(x_j + x_k)\right) \quad (3.15)$$

$$\mathcal{G}(\epsilon^i) = 1 \quad (3.16)$$

$$\mathcal{G}(S_i^i) = \exp(-\eta_i A_i y_i - \beta_i b_i + \eta_i A_i \xi_i b_i - a_i \alpha_i - A_i \xi_i x_i) \quad (3.17)$$

$$\mathcal{G}(\mathcal{R}_{ij}) = \exp\left(a_j b_i + \frac{1 - B_i}{b_i} y_i x_j\right) \quad (3.18)$$

$$\mathcal{G}(C_i) = \sqrt{B_i} \quad (3.19)$$

$$\mathcal{G}(\nu_i) = \sqrt{B_i} \exp\left(a_i b_i + \frac{1 - B_i}{b_i} x_i y_i\right) \quad (3.20)$$

Proof. This follows from [BNvdVb, §6]. For completeness, we include a direct proof below:

To prove equation (3.13), we use equations (2.49) to (2.51), which allows us to commute exponentials past each other to bring expressions into yba -order. Below we omit the index k for readability:

$$\begin{aligned} \mathcal{G}(m^{ij}) &= (\mathbb{O}^{-1} \circ m^{ij} \circ \mathbb{O})(e^{\eta_i y_i + \beta_i b_i + \alpha_i a_i + \xi_i x_i} e^{\eta_j y_j + \beta_j b_j + \alpha_j a_j + \xi_j x_j}) \\ &= \mathbb{O}^{-1}(e^{\eta_i y} e^{\beta_i b} e^{\alpha_i a} e^{\xi_i x} e^{\eta_j y} e^{\beta_j b} e^{\alpha_j a} e^{\xi_j x}) \\ &\stackrel{(2.49)}{=} \mathbb{O}^{-1}(e^{\eta_i y} e^{\beta_i b} e^{\alpha_i a} (e^{\xi_i \eta_j b} e^{\eta_j y} e^{\xi_i x}) e^{\beta_j b} e^{\alpha_j a} e^{\xi_j x}) \\ &\stackrel{(2.50)}{=} \mathbb{O}^{-1}\left(e^{(\beta_j + \beta_i + \xi_i \eta_j) b} e^{\eta_i y} e^{\frac{\eta_j}{A_i} y} e^{\alpha_i a} e^{\xi_i x} e^{\alpha_j a} e^{\xi_j x}\right) \\ &\stackrel{(2.51)}{=} \mathbb{O}^{-1}\left(e^{\left(\eta_i + \frac{\eta_j}{A_i}\right) y} e^{(\beta_j + \beta_i + \xi_i \eta_j) b} e^{(\alpha_i + \alpha_j) a} e^{\left(\frac{\xi_i}{A_j} + \xi_j\right) x}\right) \\ &= e^{\left(\eta_i + \frac{\eta_j}{A_i}\right) y} e^{(\beta_j + \beta_i + \xi_i \eta_j) b} e^{(\alpha_i + \alpha_j) a} e^{\left(\frac{\xi_i}{A_j} + \xi_j\right) x} \end{aligned} \quad (3.21)$$

Since this expression is now written in the **y**ba-order, we conclude that the corresponding generating function is this same expression, but written with commuting variables.

The other computation we must verify is the antipode, which follows similarly:

$$\begin{aligned}
\mathcal{G}(S) &= (\mathbb{O}^{-1} \circ S \circ \mathbb{O})(\mathbf{e}^{\eta y + \beta b + \alpha a + \xi x}) \\
&= \mathbb{O}^{-1}(\mathbf{e}^{-\xi x} \mathbf{e}^{-\alpha a} \mathbf{e}^{-\beta b} \mathbf{e}^{-\eta y}) \\
&\stackrel{(2.50)}{=} \mathbb{O}^{-1}(\mathbf{e}^{-\xi x} \mathbf{e}^{-\mathcal{A}\eta y} \mathbf{e}^{-\beta b} \mathbf{e}^{-\alpha a}) \\
&\stackrel{(2.49)}{=} \mathbb{O}^{-1}(\mathbf{e}^{\xi \mathcal{A}\eta b} \mathbf{e}^{-\mathcal{A}\eta y} \mathbf{e}^{-\xi x} \mathbf{e}^{-\alpha a} \mathbf{e}^{-\beta b}) \\
&\stackrel{(2.51)}{=} \mathbb{O}^{-1}(\mathbf{e}^{-\mathcal{A}\eta y} \mathbf{e}^{(\eta \mathcal{A}\xi - \beta) b} \mathbf{e}^{-\alpha a} \mathbf{e}^{-\xi \mathcal{A}x}) \\
&= \mathbf{e}^{-\mathcal{A}\eta y + (\eta \mathcal{A}\xi - \beta) b - \alpha a - \xi \mathcal{A}x}
\end{aligned} \tag{3.22}$$

Equation (3.15) follows similarly, though the centrality of **b** makes for an easier computation than the rest. Finally, equations (3.14), (3.16) and (3.18) to (3.20) follow immediately. \square

Following the convention laid out earlier, we will write a morphism with domain D , codomain C , and generating function $f(\zeta_D, z_C)$ as $f(\zeta_D, z_C)_{C^D}$ (omitting any set if it is empty).

3.2 COMPUTATIONAL EXAMPLE

Continuing the computation in equation (2.59), we now have the tools to bring the expression into a reduced form:

$$\left(B_4^{1/2} \mathbf{e}^{a_5 b_1 + a_7 b_3 + a_2 b_6 - \frac{(B_1-1)x_5 y_1}{b_1} - \frac{(B_3-1)x_7 y_3}{b_3} - \frac{(B_6-1)x_2 y_6}{b_6}} \right)_{\{1, \dots, 7\}} \parallel m_1^{\{1, \dots, 7\}} \tag{3.23}$$

Computing this contraction yields:

$$\left(\exp \left(3a_1 b_1 + \frac{x_1 y_1 - B_1^3 x_1 y_1}{b_1} \right) \frac{1}{-B_1^{3/2} + B_1^{5/2} + \sqrt{B_1}} \right)_{\{1\}} \tag{3.24}$$

To bring this into a more recognizable form, we will also correct for the framing by adding twists (that is, ribbon elements ν):

$$(3.24) \cdot \nu_2 \nu_3 \nu_4 // m_1^{1,2,3,4} = \left(\frac{1}{B_1 - 1 + B_1^{-1}} \right)_{\{1\}} \quad (3.25)$$

Thus we finish with the observation that Z applied to the (positive) trefoil is equal to the reciprocal of the Alexander polynomial Δ : $Z(K_{3,1}) = \Delta(K_{3,1})^{-1}$. In fact, this holds in general:

Proposition 3.6 ([BNvdVb]). *Let K be a (framed) upright knot diagram with self-linking number 0. Then:*

$$Z(K) = \Delta(K)^{-1} \quad (3.26)$$

CONSTRUCTING THE TRACE

4.1 EXTENDING AN INVARIANT OF OPEN TANGLES TO MIXED TANGLES

Thus far, the algebraic framework we have defined allows us to describe invariants of tangles with no closed components. We now provide an extension to include closed components.

Definition 4.1 (traced meta-algebra). A traced meta-algebra (or traced meta-monoid) is two things:

1. A collection of meta-algebras: for each finite set L , we assign one meta-algebra $\{A_{S,L}\}_S$.^{†1} The multiplication maps $m_k^{i,j}[L]$ then take the form:

$$m_k^{i,j}[L][S]: A_{\{i,j\} \sqcup S, L} \rightarrow A_{\{k\} \sqcup S, L} \quad (4.1)$$

for i, j, k disjoint from both S and L .

2. For each pair of disjoint finite sets S, L and index $i \notin S \cup L$, a trace $\text{tr}^i: A_{\{i\} \sqcup S, L} \rightarrow A_{S, \{i\} \sqcup L}$ which governs the compatibility of the families of meta-algebras in the following way:

$$m_k^{i,j} // \text{tr}^k = m_k^{j,i} // \text{tr}^k \quad (4.2)$$

Equation (4.2) is called the cyclic property of the trace.

Furthermore, tr^i commutes with operations which do not involve the same strands. That is, for any operation ϕ_C^D and $i \notin D \cup C$:

$$\phi_C^D // \text{tr}^i = \text{tr}^i // \phi_C^D \quad (4.3)$$

^{†1} These sets index the “strands” S and the “loops” L .

The first example we give is that of mixed tangles.

Definition 4.2 (mixed upright tangles). Let $\overline{\mathcal{T}}_{L,S}^{\text{up}}$ be the set of upright tangles with open strands indexed by S and closed strands indexed by L . The operations $\phi[L][S]$ are defined analogously to the $\phi[S]$ given in theorem 2.31. (Here ϕ varies over $m, \eta, \Delta, \epsilon, S, \mathcal{R}$, and C .)

Lemma 4.3 (tangles are a traced algebra). *The collection of all $\overline{\mathcal{T}}_{L,S}^{\text{up}}$ is a traced ribbon meta-Hopf algebra, with trace map given by closing a strand into a loop.*

Proof. When $L = \emptyset$, the situation is exactly the case of theorem 2.31, so $\overline{\mathcal{T}}_{\emptyset,S}^{\text{up}} = \mathcal{T}_S^{\text{up}}$ is a meta-Hopf algebra. Furthermore, since the Reidemeister moves are local operations, the presence of closed components does not affect our ability to verify the identities on the Hopf-algebra operations.

The last point to verify is that closing a strand into a loop is a cyclic operation. Given two strands, we must verify that stitching one end together, then tracing the other yields the same diagram as stitching the other ends together, then taking the trace. However, by definition of trace, these two actions yield identical diagrams: the two strands are replaced by the same closed loop. \square

Remark 4.4. As in remark 2.32, the use of virtual tangles allows us to ensure this result holds regardless of the diagram in question. In the computations to be done in this paper, we will ensure that the two endpoints being joined are situated so that joining them results in a planar diagram.

Lemma 4.5 (coinvariants are a trace map). *Let A be an algebra, and denote by $A_A := A/[A, A]$ its set of coinvariants.^{†2} Define $A_{S,L} := A_S \otimes (A_A)_L$. Then A defines a traced meta-algebra with trace map given by $\text{tr}^i[S][L]: A_{S \sqcup \{i\}, L} \rightarrow A_{S, L \sqcup \{i\}}$.*

Proof. Observe that for any choice of L , extending morphisms by the identity yield an isomorphism of traced meta-Hopf algebras:

$$\begin{aligned} \phi_L: \{A_S\} &\xrightarrow{\sim} \{A_{S,L}\} \\ A_S &\mapsto A_{S,L} \\ f_D^C &\mapsto f_D^C \otimes \text{id}_{(A_A)_L} \end{aligned} \tag{4.4}$$

^{†2} Here, $[A, A] = \text{span}\{[x, y] \mid x, y \in A\}$ refers to the vector space of Lie brackets, not the ideal generated thereby. The space A_A does not have an algebra structure in general.

Next, we must show that $m_k^{ij} \parallel \text{tr}^k = m_k^{ji} \parallel \text{tr}^k$. This amounts to showing that, given $u, v \in A$, that $\overline{uv} = \overline{vu} \in A_A$. However, by the construction of the coinvariants, $\overline{uv} - \overline{vu} = \overline{uv - vu} = \overline{0} \in A$.

Finally, we must show that tr intertwines the S -indexed collection of algebras. When $i \notin D \cup C$, an operation ϕ_C^D and tr^i manipulate disjoint sets of tensor factors, we notice that both composition orders are equivalent to a tensor product:

$$\phi_C^D \parallel \text{tr}^i = \phi_C^D \text{tr}^i = \text{tr}^i \parallel \phi_C^D \quad (4.5)$$

We conclude that vector-space algebras have a traced algebraic structure. \square

4.2 THE SPACE OF COINVARIANTS OF U

We start with a result that simplifies working with coinvariants:

Lemma 4.6 (Coinvariant simplification). *Let \mathfrak{h} be a Lie algebra. Then $\mathfrak{U}(\mathfrak{h})_{\mathfrak{U}(\mathfrak{h})} = \mathfrak{U}(\mathfrak{h})_{\mathfrak{h}}$.*

Proof. First, observe that for any $u, v, f \in \mathfrak{U}(\mathfrak{h})$, $\text{ad}_{uv}(f) = \text{ad}_u(vf) + \text{ad}_v(fu)$. Proceeding inductively, for any monomial $\mu \in \mathfrak{U}(\mathfrak{h})$, $\text{ad}_\mu(f)$ is a linear combination of elements of $[\mathfrak{h}, \mathfrak{U}(\mathfrak{h})]$. By linearity of ad , we conclude $[\mathfrak{U}(\mathfrak{h}), \mathfrak{U}(\mathfrak{h})] = [\mathfrak{h}, \mathfrak{U}(\mathfrak{h})]$. \square

Theorem 4.7. *The space of coinvariants of U, U_U , has basis $\{\mathbf{y}^n \mathbf{a}^k \mathbf{x}^n\}_{n,k \geq 0}$.*

Proof. By lemma 4.6, we need only compute $[\mathfrak{g}, U]$ to determine U_U . Using lemma 2.36, we compute the adjoint actions of \mathbf{y} , \mathbf{a} , and \mathbf{x} . (Recall \mathbf{b} is central.)

$$\text{ad}_{\mathbf{a}} f(\mathbf{x}) = \mathbf{x}f'(\mathbf{x}) \quad \text{ad}_{\mathbf{a}} f(\mathbf{y}) = -\mathbf{y}f'(\mathbf{y}) \quad (4.6)$$

$$\text{ad}_{\mathbf{x}} f(\mathbf{y}) = \mathbf{b}f'(\mathbf{y}) \quad \text{ad}_{\mathbf{x}} f(\mathbf{a}) = -\nabla[f](\mathbf{a})\mathbf{x} \quad (4.7)$$

$$\text{ad}_{\mathbf{y}} f(\mathbf{x}) = -\mathbf{b}f'(\mathbf{x}) \quad \text{ad}_{\mathbf{y}} f(\mathbf{a}) = \mathbf{y}\nabla[f](\mathbf{a}) \quad (4.8)$$

(Here ∇ is the backwards finite difference operator $\nabla[f](x) := f(x) - f(x-1)$.) Observe for any n, m, k , and polynomials f and g :

$$\text{ad}_{\mathbf{a}}(\mathbf{y}^m g(\mathbf{b}, \mathbf{a}) \mathbf{x}^n) = (n - m) \mathbf{y}^m g(\mathbf{b}, \mathbf{a}) \mathbf{x}^n \quad (4.9)$$

$$\text{ad}_{\mathbf{x}}(\mathbf{y}^{n+1} \mathbf{b}^{m-1} f(\mathbf{a}) \mathbf{x}^k) = (n + 1) \mathbf{y}^n \mathbf{b}^m f(\mathbf{a}) \mathbf{x}^k - \mathbf{y}^{n+1} \mathbf{b}^{m-1} \nabla[f](\mathbf{a}) \mathbf{x}^{k+1} \quad (4.10)$$

$$\text{ad}_{\mathbf{y}}(\mathbf{y}^n \mathbf{b}^{m-1} f(\mathbf{a}) \mathbf{x}^{k+1}) = -(k + 1) \mathbf{y}^n \mathbf{b}^m f(\mathbf{a}) \mathbf{x}^k + \mathbf{y}^{n+1} \mathbf{b}^{m-1} \nabla[f](\mathbf{a}) \mathbf{x}^{k+1} \quad (4.11)$$

By equation (4.9), any monomial whose powers of \mathbf{y} and \mathbf{x} differ vanish in $U_{\mathfrak{g}}$. As a consequence, in equations (4.10) and (4.11), the only nontrivial case is when $n = k$, resulting in the same relation. By induction on n , we conclude that:

$$\mathbf{y}^n \mathbf{b}^m f(\mathbf{a}) \mathbf{x}^k \sim \delta_{nk} \frac{n!}{(n+m)!} \mathbf{y}^{n+m} \nabla^m[f](\mathbf{a}) \mathbf{x}^{n+m} \quad (4.12)$$

where \sim refers to equivalence in the set of coinvariants. When f is a monomial in equation (4.12), we see directly that $U_{\mathfrak{g}}$ is spanned by $\{\mathbf{y}^n \mathbf{a}^k \mathbf{x}^n\}_{n,k \geq 0}$.

Finally, all that remains to show is linear independence. We do so by defining the trace map:

$$\begin{aligned} \text{tr}: U &\rightarrow U_U \\ \mathbf{y}^n \mathbf{b}^m f(\mathbf{a}) \mathbf{x}^k &\mapsto \delta_{nk} \frac{n!}{(n+m)!} \mathbf{y}^{n+m} \nabla^m[f](\mathbf{a}) \mathbf{x}^{n+m} \end{aligned} \quad (4.13)$$

We will now show that the trace sends each relation defined in equations (4.9) to (4.11) to 0. For the relations coming from $\text{ad}_{\mathbf{a}}$, notice that:

$$\text{tr}(\text{ad}_{\mathbf{a}}(\mathbf{y}^n \mathbf{b}^m f(\mathbf{a}) \mathbf{x}^k)) = \text{tr}\left((n - k) \delta_{nk} \frac{n!}{(n+m)!} \mathbf{y}^{n+m} \nabla^m[f](\mathbf{a}) \mathbf{x}^{n+m}\right) = 0 \quad (4.14)$$

For the relations from $\text{ad}_{\mathbf{y}}$, observe:

$$\begin{aligned}
 & \text{tr}(\text{ad}_{\mathbf{y}}(\mathbf{y}^n \mathbf{b}^{m-1} f(\mathbf{a}) \mathbf{x}^{k+1})) \\
 &= \text{tr}(-(k+1) \mathbf{y}^n \mathbf{b}^m f(\mathbf{a}) \mathbf{x}^k + \mathbf{y}^{n+1} \mathbf{b}^{m-1} \nabla[f](\mathbf{a}) \mathbf{x}^{k+1}) \\
 &= -(k+1) \delta_{nk} \frac{n!}{(n+m)!} \mathbf{y}^{n+m} \nabla^m[f](\mathbf{a}) \mathbf{x}^{n+m} \\
 &\quad + \delta_{n+1, k+1} \frac{(n+1)!}{(n+1+m-1)!} \mathbf{y}^{n+m} \nabla^{m-1}[\nabla f](\mathbf{a}) \mathbf{x}^{n+m} \\
 &= 0
 \end{aligned} \tag{4.15}$$

The relations from $\text{ad}_{\mathbf{x}}$ follow almost identically as those of $\text{ad}_{\mathbf{y}}$. As \mathbf{b} is central, $\text{ad}_{\mathbf{b}}$ generates no more relations, so linear independence is established. \square

A generating function for the coinvariants

In order to define a generating function, we need to choose a basis for the space of coinvariants. We define an isomorphism from the space of coinvariants to a polynomial space, tweaking the basis defined in theorem 4.7 by scalar multiples. Since it plays the role of the ordering map, we also name it \mathbb{O} .

$$\begin{aligned}
 \mathbb{O}: \mathbb{Q}[a, z] &\xrightarrow{\sim} U_U \\
 a^n z^k &\mapsto \frac{1}{k!} \mathbf{y}^k \mathbf{a}^n \mathbf{x}^k \\
 k! \nabla^m[f](a) z^{k+m} &\leftarrow \mathbf{y}^k \mathbf{b}^m f(\mathbf{a}) \mathbf{x}^k
 \end{aligned} \tag{4.16}$$

This defines a commutative square upon whose bottom edge ($\tau = \mathbb{O} // \text{tr} // \mathbb{O}^{-1}$) we compute the generating function:

$$\begin{array}{ccc}
 U & \xrightarrow{\text{tr}} & U_U \\
 \mathbb{O} \uparrow & & \uparrow \mathbb{O} \\
 \mathbb{Q}[y, b, a, x] & \xrightarrow{\tau} & \mathbb{Q}[a, z]
 \end{array} \tag{4.17}$$

We begin with a result on finite differences:

Lemma 4.8 (finite differences of exponentials). *The finite difference operator acts in the following way on exponentials:*

$$\nabla^n[\mathbf{e}^{\alpha a}](a) = (1 - \mathbf{e}^{-\alpha})^n \mathbf{e}^{\alpha a} \tag{4.18}$$

Proof. Using the fact that $\nabla^n[f](x) = \sum_{k=0}^n \binom{n}{k} (-1)^k f(x-k)$, we see that $\nabla^n[e^{\alpha a}](a) = \sum_{k=0}^n \binom{n}{k} (-1)^k e^{\alpha a - \alpha k} = (1 - e^{-\alpha})^n e^{\alpha a}$. \square

We now are ready to compute the generating function for the trace:

Theorem 4.9 (Generating function for the trace of U).

$$\mathcal{G}(\text{tr}) = \exp\left(\alpha a + (\eta\xi + \beta(1 - e^{-\alpha}))z\right) \quad (4.19)$$

Proof. Using lemma 4.8 and the extension of scalars of tr to $\mathbb{Q}[[\eta, \beta, \alpha, \xi]]$, we see

$$\begin{aligned} \mathcal{G}(\mathbb{O} // \text{tr} // \mathbb{O}^{-1}) &= (e^{\eta y} e^{\beta b} e^{\alpha a} e^{\xi x}) // \text{tr} // \mathbb{O}^{-1} \\ &= \mathbb{O}^{-1} \sum_{i,j,k} \text{tr} \left(\frac{(\eta y)^i}{i!} \frac{(\beta b)^j}{j!} e^{\alpha a} \frac{(\xi x)^k}{k!} \right) \\ &= \sum_{i,j} \frac{\eta^i \beta^j \xi^i}{i! j!} (1 - e^{-\alpha})^j e^{\alpha a} z^{i+j} = e^{\alpha a + (\eta\xi + \beta(1 - e^{-\alpha}))z} \quad \square \end{aligned} \quad (4.20)$$

Evaluation of the trace on a generic element

Here we will outline a computation involving the trace by using Bar-Natan and van der Veen's Contraction Theorem.

A typical value for a tangle invariant that arises is of the form:

$$P e^{c + c_a a_i + c_b b_i + \sigma a_i b_i + c_x (b_i) x_i + c_y (b_i) y_i + \lambda (b_i) x_i y_i} \quad (4.21)$$

Here, c , c_a , c_b , and σ denote constants with respect to the variables y_i , b_i , a_i , and x_i (collectively referred to as “ v_i ”s), while c_x , c_y , and λ are potentially b_i -dependent, and P is a (rational) function in (the square root of) B_i .

Theorem 4.10 (The trace of a Gaussian). *With symbols as defined above, let $f(y_i, b_i, a_i, x_i) = P(B_i) e^{c + c_a a_i + c_b b_i + \sigma a_i b_i + c_x (b_i) x_i + c_y (b_i) y_i + \lambda (b_i) x_i y_i}$. Then when $\sigma = 0$,*

$$\langle f(y_i, b_i, a_i, x_i) \text{tr}^i \rangle_{v_i} = \frac{P(e^{-\mu})}{1 - \lambda(\mu) \bar{z}_i} e^{c + c_a \bar{a}_i + c_b \mu + \frac{c_y(\mu) c_x(\mu) \bar{z}_i}{1 - \lambda(\mu) \bar{z}_i}} \quad (4.22)$$

where $\mu := (1 - e^{-c_a}) \bar{z}_i$.

Proof. Let us compute the trace of equation (4.21). For clarity, we will put bars over the coinvariants variables a_i and z_i , as they do not play a role in the contraction.

$$\begin{aligned} & \langle P(B_i) \mathbf{e}^{c+c_a a_i+c_b b_i+c_x(b_i)x_i+c_y(b_i)y_i+\lambda(b_i)x_i y_i} \mathbf{tr}^i \rangle_{v_i} \\ &= \langle P(B_i) \mathbf{e}^{c+c_b b_i+c_x(b_i)x_i+c_y(b_i)y_i+\lambda(b_i)x_i y_i+\eta_i \xi_i \bar{z}_i+\beta_i(1-\mathbf{e}^{-\alpha_i})\bar{z}_i} \mathbf{e}^{c_a a_i+\alpha_i \bar{a}_i} \rangle_{v_i} \\ &= \mathbf{e}^{c_a \bar{a}_i} \langle P(B_i) \mathbf{e}^{c+c_x(b_i)x_i+c_y(b_i)y_i+\lambda(b_i)x_i y_i+\eta_i \xi_i \bar{z}_i} \mathbf{e}^{c_b b_i+\beta_i(1-\mathbf{e}^{-c_a})\bar{z}_i} \rangle_{b_i, x_i, y_i} \end{aligned}$$

In what follows, we let $\mu := (1 - \mathbf{e}^{-c_a})\bar{z}_i$:

$$\begin{aligned} &= \mathbf{e}^{c+c_a \bar{a}_i+c_b \mu} P(\mathbf{e}^{-\mu}) \langle \mathbf{e}^{c_y(\mu)y_i} \mathbf{e}^{(c_x(\mu)+\lambda(\mu)y_i)x_i+\xi_i \eta_i \bar{z}_i} \rangle_{x_i, y_i} \\ &= \mathbf{e}^{c+c_a \bar{a}_i+c_b \mu} P(\mathbf{e}^{-\mu}) \langle \mathbf{e}^{c_y(\mu)y_i+c_x(\mu)\bar{z}_i \eta_i+\lambda(\mu)\bar{z}_i \eta_i y_i} \rangle_{y_i} \\ &= \frac{P(\mathbf{e}^{-\mu})}{1-\lambda(\mu)\bar{z}_i} \mathbf{e}^{c+c_a \bar{a}_i+c_b \mu+\frac{c_y(\mu)c_x(\mu)\bar{z}_i}{1-\lambda(\mu)\bar{z}_i}} \end{aligned} \tag{4.23}$$

□

Remark 4.11. Theorem 4.10 requires that the coefficient σ of the $a_i b_i$ -term in the exponential is 0. This restriction allows the computed value to be a composition of elementary functions. We demonstrate this by considering the case where $c = c_b = c_x = c_y = \lambda = 0$. Expanding equation (4.22) yields the following expression, which we name S :

$$\langle \mathbf{e}^{\beta_i(1-\mathbf{e}^{-c_a-\sigma b_i})\bar{z}_i} \rangle_{b_i} = S \tag{4.24}$$

The quantity S satisfies the relation $S = \mathbf{e}^{-c_a-\sigma S}$, whose solution is $S = \frac{W(\mathbf{e}^{-c_a \sigma})}{\sigma}$ where W denotes the Lambert W function. This expression now lies far outside of the Gaussian framework that Z is computed in.

Proof that S satisfies $S = \mathbf{e}^{-c_a-\sigma S}$. Denote by γ_n the b_i -coefficients of the expression $1 - \mathbf{e}^{-c_a-\sigma b_i}$, so that $\beta_i(1 - \mathbf{e}^{-c_a-\sigma b_i}) = \sum_{n=0}^{\infty} \beta_i \gamma_n b_i^n$. Visualizing the γ_n -term as a node of a directed tree with one incoming edge (corresponding to β_i) and n outgoing edges (corresponding to each of the b_i factors), we get the relation

$$S = \gamma_1 S + \gamma_2 S^2 + \gamma_3 S^3 + \dots \tag{4.25}$$

With the observation that $\gamma_n = \frac{\mathbf{e}^{-c_a}(-\mu)^n}{n!}$, we conclude that $S = \mathbf{e}^{-c_a-\mu S}$.

□

The $a_i b_i$ -coefficient associated to an open tangle is exactly the self-linking number of component i . By adding correcting writhe terms, one can ensure this coefficient is 0.

When applying equation (4.22) to a tangle with multiple closed components, the $a_i b_i$ -coefficient no longer has the interpretation of a self-linking number, so the writhe-correction technique fails. The author did not find a workable closed form for the trace in this case.

We point out that the outcome of this computation is not guaranteed to be a Gaussian. This puts a limitation on the applicability of this formula to links with more than two components, explored in section 5.1.

4.3 COMPUTATIONAL EXAMPLE

Using the formula given in equation (4.22), let us work through an example. Consider the Hopf link (figure 4.1). In order to compute Z^{tr} , we first open one of the components, ensuring the rotation number at the endpoints is 0 (figure 4.2a). We do this because the total trace of many links we consider is a trivial 1. The use of a partial trace provides a stronger invariant. We then break open all the closed components (here, there is just one), so that an open tangle remains, as in figure 4.2b, which we call H .

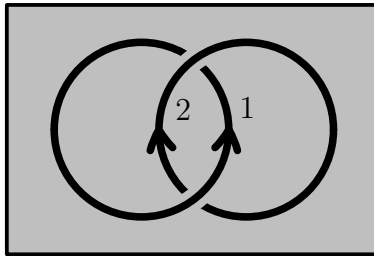
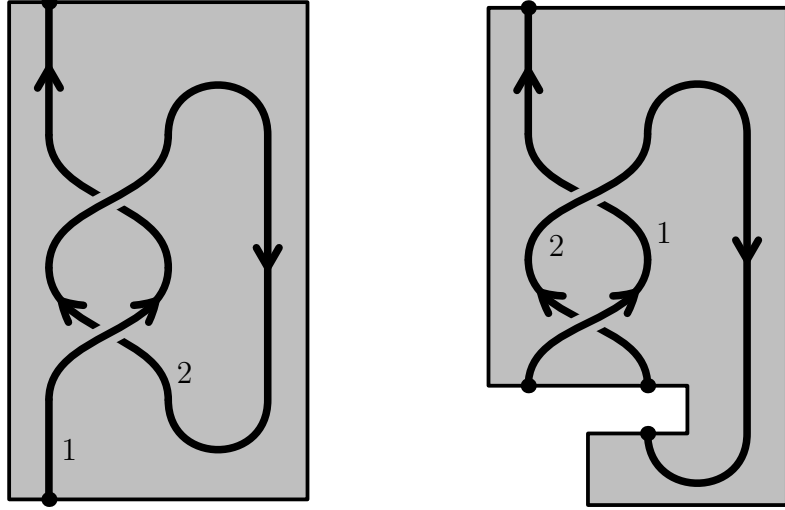


Figure 4.1: The (positive) Hopf link.

Computing Z on the open tangle H yields:

$$Z(H) = e^{a_2 b_1 + a_1 b_2 + \frac{(B_1 - 1)y_1((B_2 - 1)x_1 - x_2)}{b_1} - \frac{B_1(B_2 - 1)x_1 y_2}{b_2}} \sqrt{B_2} \quad (4.26)$$

Applying tr^2 then provides us with the invariant corresponding to figure 4.2a: (Here, we use w_i to represent a_i to be consistent with the computer-



(a) Opening the Hopf link to a tangle with one open component. (b) A tangle H representing the Hopf link with both tangles open.

Figure 4.2: Opening the strands of the closed Hopf link.

generated table of values in appendix B, which allows for easier distinguishing between variables corresponding to U and those to U_U .)

$$\mathrm{tr}^2(Z(H)) = e^{a_1(z_2 - B_1 z_2) + b_1 w_2 + \frac{e^{-z_2}(x_1 y_1 e^{z_2} - x_1 y_1 e^{B_1 z_2})}{b_1}} e^{\frac{1}{2} B_1 z_2 - \frac{z_2}{2}} \quad (4.27)$$

We repeat this procedure with the other component, opening and straightening component 2, then opening component 1 without straightening to get an open tangle digram H' , then computing $\mathrm{tr}^1(Z(H'))$. The symmetry of the Hopf link results in the same computation with indices 1 and 2 exchanged:

$$\mathrm{tr}^1(Z(H')) = e^{a_2(z_1 - B_2 z_1) + b_2 w_1 + \frac{e^{-z_1}(x_2 y_2 e^{z_1} - x_2 y_2 e^{B_2 z_1})}{b_2}} e^{\frac{1}{2} B_2 z_1 - \frac{z_1}{2}} \quad (4.28)$$

Finally, we keep track of which indices correspond to open strands and closed strands. We will extend the notation from the previous section to differentiate between open and closed indices. We write a morphism with domain $D = D_o \sqcup D_c$, codomain $C = C_o \sqcup C_c$ (here D_o denotes domain indices which are open, while D_c those which are closed, with the same convention for C) and generating function $f(\zeta_D, z_C)$ as $f(\zeta_D, z_C)_{(C_o, C_c)}^{(D_o, D_c)}$.

We package these data into a multiset:

$$Z(L_{\text{Hopf}}) = \left(\left(\mathbf{e}^{a_1(z_2 - B_1 z_2) + b_1 w_2 + \frac{e^{-z_2}(x_1 y_1 e^{z_2 - x_1 y_1 e^{B_1 z_2}})}{b_1}} \mathbf{e}^{\frac{1}{2} B_1 z_2 - \frac{z_2}{2}} \right)_{\{1\}, \{2\}}, \right. \\ \left. \left(\mathbf{e}^{a_2(z_1 - B_2 z_1) + b_2 w_1 + \frac{e^{-z_1}(x_2 y_2 e^{z_1 - x_2 y_2 e^{B_2 z_1}})}{b_2}} \mathbf{e}^{\frac{1}{2} B_2 z_1 - \frac{z_1}{2}} \right)_{\{2\}, \{1\}} \right) \quad (4.29)$$

CONCLUSIONS

5.1 LIMITATIONS OF Z^{tr}

For certain links, computing Z^{tr} results in expressions involving the Lambert W -function. The presence of this non-elementary function complicates attempts to keep the invariant valued in either the space of perturbed Gaussians, or a manageable extension thereof.

As a result of this limitation, we restrict our attention to a class of links that does yields only elementary expressions, namely two-component links.

5.2 COMPARISON WITH THE MULTIVARIABLE ALEXANDER POLYNOMIAL

Given that the long knot (i.e. one-component) case of this invariant encodes the Alexander Polynomial, it was suspected that the invariant on long links (i.e. multiple components, one of which is long) formed by adding the trace would encode the MVA. However, there are links which the MVA separates which this invariant does not.

Using the Thistlethwaite table of prime links [BNMea], on all prime two-component links with at most 11 crossings (a collection of size 914), the trace map attains 878 distinct values, while the MVA attains only 778. However, the two invariants are incomparable in terms of their strength.

The links L_{5a1} and L_{10a95} are not distinguished by their partial traces, with both returning a value of:

$$\left(\left(\frac{B_1}{B_1^2 t_2 - 2B_1 t_2 + B_1 + t_2} \right)_{(\{1\}, \{2\})}, \left(\frac{B_2}{B_2^2 t_1 - 2B_2 t_1 + B_2 + t_1} \right)_{(\{2\}, \{1\})} \right) \quad (5.1)$$

However, the values of these links under the MVA are $\frac{(B_1-1)(B_2-1)}{\sqrt{B_1}\sqrt{B_2}}$ and $-\frac{(B_1-1)(B_2-1)(B_1+B_2-1)(B_2B_1-B_1-B_2)}{B_1^{3/2}B_2^{3/2}}$ respectively.

In the other direction, there are also pairs of links in the same fibre of the MVA which this traced invariant can distinguish. In particular L_{5a1} and L_{7n2} both have the same value under the MVA, namely $\frac{(B_1-1)(B_2-1)}{\sqrt{B_1}\sqrt{B_2}}$. The trace yields the values $\left(\left(\frac{B_1}{B_1^2t_2-2B_1t_2+B_1+t_2} \right)_{(\{1\},\{2\})}, \left(\frac{B_2}{B_2^2t_1-2B_2t_1+B_2+t_1} \right)_{(\{2\},\{1\})} \right)$ and $\left(\left(\frac{B_1}{B_1^2t_2-2B_1t_2+B_1+t_2} \right)_{(\{1\},\{2\})}, \left(\frac{B_2}{B_2^2t_1-2B_2t_1+B_2^2-B_2+t_1+1} \right)_{(\{2\},\{1\})} \right)$ respectively.

This example also serves to highlight that the information provided by leaving one strand open is not enough to recover the value of a different strand being left open.

5.3 FURTHER WORK

While all other Hopf algebra operations in U are expressed by [BNvdVb] as perturbed Gaussians, the form in equation (4.19) does not conform to the same structure. Further work is needed to either implement this operation into the established framework, or to suitably extend the framework (perhaps with the use of Lambert W -functions).

Another potential approach is to search for a trace within the framework of perturbed Gaussians themselves. The coinvariants map defined here are universal with respect to the algebra U (meaning that any other cyclic map factors through this one). Instead, one can probe the existence of a cyclic map tr_{PG} in the space of perturbed Gaussians which is universal— that is, all other perturbed Gaussians which are cyclic factor through tr_{PG} .



CODE

A.1 IMPLEMENTATION OF THE PERTURBED GAUSSIAN FRAMEWORK

This is a Mathematica™ implementation by Bar-Natan and van der Veen in [BNvdVb], modified by the author. The full source code is available at <https://github.com/phro/GD0>.

Use case

For example, to compute the value of Z on the Whitehead link, one need only run the following:

```
1 In[0]:= Z@Knot[5, Alternating, 1]
```

```
2 Out[0]= GD0["do" -> {}, "dc" -> {}, "co" -> {1, 5}, "cc" -> {}, "PG"
  ↪ -> PG["L" -> 0, "Q" -> -((-1 + B[1])*(-1 + B[5])*(B[5]*x[5] +
  ↪ B[1]*((-1 + B[5])*x[1] - B[5]*x[5]))*(b[5]*y[1] - b[1]*y[5])]/
  ↪ (b[1]*b[5]*B[1]*(B[1]*(-1 + B[5]) - B[5])*B[5]), "P" ->
  ↪ Sqrt[B[5]]/(B[1] + B[5] - B[1]*B[5])]
```

Written in a more readable format, this becomes:

$$\left(\frac{\sqrt{B_5}}{-B_5 B_1 + B_1 + B_5} e^{-\frac{(B_1-1)(B_5-1)(b_5 y_1 - b_1 y_5)(B_5 x_5 + B_1((B_5-1)x_1 - B_5 x_5))}{b_1 b_5 B_1 (B_1(B_5-1) - B_5) B_5}} \right)_{\substack{\{\}, \{\} \\ \{1,5\}, \{\} \\ \text{(A.1)}}$$

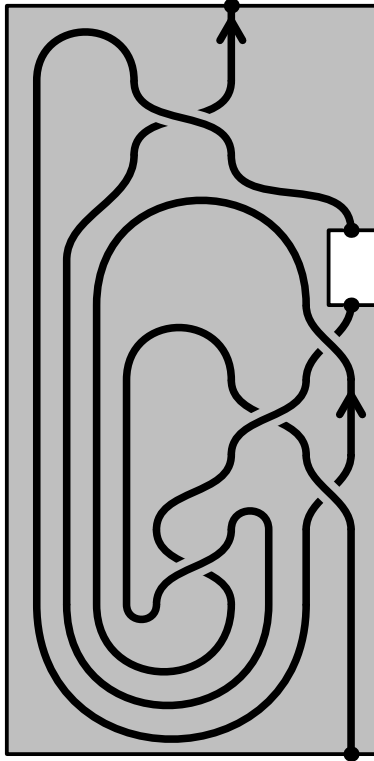


Figure A.1: The Whitehead link with one of its components ready to have the trace applied. Since the rotation number of the two components differ, the value of Z is not symmetric.

Implementation

We begin by setting some variables, as well as a method for modifying associations.

```

3  $\gamma = 1$ ;  $\hbar = 1$ ;  $\$k = 0$ ;
4 setValue[value_, obj_, coord_] := Module[
5     {b = Association @@ obj},
6     b[coord] = value; Head[obj] @@ Normal@b
7 ]

```

We introduce notation $\text{PG}[L, Q, P]$ to be interpreted as the Perturbed Gaussian Pe^{L+Q} . The function `fromE` serves as a compatibility layer between a Bar-Natan and van der Veen's implementation and this one.

```

8 toPG[L_, Q_, P_] := PG["L" -> L, "Q" -> Q, "P" -> P]
9 fromE[e_][DoubleStruckCapitalE] := toPG@@e/.
10     Subscript[(v:y|b|t|a|x|B|T|η|β|τ|α|ξ|A), i_] -> v[i]

```

We define the Kronecker- δ function next.

```

11 δ[i_, j_] := If[SameQ[i, j], 1, 0]

```

Next we introduce helper functions for the reading and manipulating of PG-objects:

```

12 getL[pg_PG] := Lookup[Association@@pg, "L", 0]
13 getQ[pg_PG] := Lookup[Association@@pg, "Q", 0]
14 getP[pg_PG] := Lookup[Association@@pg, "P", 1]
15
16 setL[L_][pg_PG] := setValue[L, pg, "L"];
17 setQ[Q_][pg_PG] := setValue[Q, pg, "Q"];
18 setP[P_][pg_PG] := setValue[P, pg, "P"];
19
20 applyToL[f_][pg_PG] := pg//setL[pg//getL//f]
21 applyToQ[f_][pg_PG] := pg//setQ[pg//getQ//f]
22 applyToP[f_][pg_PG] := pg//setP[pg//getP//f]

```

Next is a function CF, which bring objects into canonical form allows us to compare for equality effectively. This is defined by Bar-Natan and van der Veen.

```

23 CCF[e_] := ExpandDenominator@ExpandNumerator@Together[
24     Expand[e] //. E^x_ E^y_ :=> E^(x + y) /. E^x_ :=> E^CCF[x]
25 ];
26 CF[sd_SeriesData] := MapAt[CF, sd, 3];
27 CF[e_] := Module[
28     {vs = Union[
29         Cases[e, (y|b|t|a|x|η|β|τ|α|ξ)[_], ∞],
30         {y, b, t, a, x, η, β, τ, α, ξ}
31     ]},
32     Total[CoefficientRules[Expand[e], vs] /.
33         (ps_ -> c_) := CCF[c] (Times @@ (vs^ps))
34     ]

```

```

35 ];
36 CF[e_PG] := e//applyToL[CF]//applyToQ[CF]//applyToP[CF]

```

We define the notion of equality for PG-objects, as well as what it means to multiply them.

```

37 Congruent[x_, y_, z_] := And[Congruent[x, y], Congruent[y, z]]
38 PG /: Congruent[pg1_PG, pg2_PG] := And[
39     CF[getL@pg1 == getL@pg2],
40     CF[getQ@pg1 == getQ@pg2],
41     CF[Normal[getP@pg1-getP@pg2] == 0]
42 ]
43
44 PG /: pg1_PG * pg2_PG := toPG[
45     getL@pg1 + getL@pg2,
46     getQ@pg1 + getQ@pg2,
47     getP@pg1 * getP@pg2
48 ]
49
50 setEpsilonDegree[k_Integer][pg_PG] := setP[Series[Normal@getP@pg, {ε,
    ↪ 0, k}]] [pg]

```

The variables y , b , t , a , and x are paired with their dual variables η , β , τ , α , and ζ . This applies as well when they have subscripts.

```

51 dds12vars = {y, b, t, a, x, z};
52 dds12varsDual = {η, β, τ, α, ξ, ζ};
53
54 Evaluate[Dual/@dds12vars] = dds12varsDual;
55 Evaluate[Dual/@dds12varsDual] = dds12vars;
56 Dual@z = ζ;
57 Dual@ξ = z;
58 Dual[u_[i_]] := Dual[u][i]

```

Since various exponentials of the lowercase variables frequently appear, we introduce capital variable names to handle various exponentiated forms.

```

59 U2l = {
60     B[i_]^p_. := E^(-p ħ γ b[i]), B^p_. := E^(-p ħ γ b),

```

```

61      W[i_]^p_ := E^(w[i])      , W^p_ := E^(p w),
62      T[i_]^p_ := E^(-p ħ t[i]) , T^p_ := E^(-p ħ t),
63      A[i_]^p_ := E^(p γ α[i])  , A^p_ := E^(-p γ α)
64 };
65 l2U = {
66      E^(c_ b[i_] + d_) := B[i]^(-c/(ħ γ))E^d,
67      E^(c_ b + d_)   := B^(-c/(ħ γ))E^d,
68      E^(c_ t[i_] + d_) := T[i]^(-c/ħ)E^d,
69      E^(c_ t + d_)     := T^(-c/ħ)E^d,
70      E^(c_ α[i_] + d_) := A[i]^(c/γ)E^d,
71      E^(c_ α + d_)     := A^(c/γ)E^d,
72      E^(c_ w[i_] + d_) := W[i]^cE^d,
73      E^(c_ w + d_)     := W^cE^d,
74      E^expr_          := E^Expand@expr
75 };

```

Below the notion of differentiation is defined for expressions which involve both upper- and lower-case variables.

```

76 DD[f_, b] := D[f, b] - ħ γ B D[f, B];
77 DD[f_, b[i_]] := D[f, b[i]] - ħ γ B[i] D[f, B[i]];
78
79 DD[f_, t] := D[f, t] - ħ T D[f, T];
80 DD[f_, t[i_]] := D[f, t[i]] - ħ T[i] D[f, T[i]];
81
82 DD[f_, α] := D[f, α] + γ A D[f, A];
83 DD[f_, α[i_]] := D[f, α[i]] + γ A[i] D[f, A[i]];
84
85 DD[f_, v_] := D[f, v];
86 DD[f_, {v_, 0}] := f;
87 DD[f_, {}] := f;
88 DD[f_, {v_, n_Integer}] := DD[DD[f, v], {v, n-1}];
89 DD[f_, {l_List, ls___}] := DD[DD[f, l], {ls}];

```

What follows now is the implementation of contraction as introduced in definition 3.3. We begin with the introduction of contractions of (finite) polynomials.

```

90 collect[sd_SeriesData, ζ_] := MapAt[collect[#, ζ] &, sd, 3];
91 collect[expr_, ζ_] := Collect[expr, ζ];

```



```

92
93 Zip[{}][P_] := P;
94 Zip[ζs_List][Ps_List] := Zip[ζs]/@Ps;
95 Zip[{ζ_, ζs__}][P_] := (collect[P // Zip[{ζs}], ζ] /.
96     f_. ζ^d_. :> DD[f, {Dual[ζ], d}]) /.
97     Dual[ζ] -> 0 /.
98     ((Dual[ζ] /. {b->B, t->T, α -> A}) -> 1)

```

We define contraction along the variables x and y (here packaged into the matrix Q).

```

99 QZip[ζs_List][pg_PG] := Module[{Q, P, ζ, z, zs, c, ys, ηs, qt, zrule,
    ↪ ζrule},
100     zs = Dual/@ζs;
101     Q = pg//getQ;
102     P = pg//getP;
103     c = CF[Q/.Alternatives@@Union[ζs, zs]->0];
104     ys = CF/@Table[D[Q, ζ] /. Alternatives@@zs->0, {ζ, ζs}];
105     ηs = CF/@Table[D[Q, z] /. Alternatives@@ζs->0, {z, zs}];
106     qt = CF/#&/@(Inverse@Table[
107         δ[z, Dual[ζ]] - D[Q, z, ζ],
108         {ζ, ζs}, {z, zs}
109     ]);
110     zrule = Thread[zs -> CF/(qt . (zs + ys))];
111     ζrule = Thread[ζs -> ζs + ηs . qt];
112     CF@setQ[c + ηs.qt.ys]@setP[Det[qt] Zip[ζs][P /. Union[zrule,
    ↪ ζrule]]]@pg
113 ]

```

We define contraction along the variables a and b (here packaged into the matrix L).

```

114 LZip[ζs_List][pg_PG] := Module[
115     {
116         L, Q, P, ζ, z, zs, Zs, c, ys, ηs, lt,
117         zrule, Zrule, ζrule, Q1, EEQ, EQ, U
118     },
119     zs = Dual/@ζs;
120     {L, Q, P} = Through[{getL, getQ, getP}@pg];
121     Zs = zs /. {b -> B, t -> T, α -> A};

```

```

122     c = CF[L/.Alternatives@@Union[ζs, zs]->0/.Alternatives@@Zs ->
      ↪ 1];
123     ys = CF/@Table[D[L,ζ]/.Alternatives@@zs->0,{ζ,ζs}];
124     ηs = CF/@Table[D[L,z]/.Alternatives@@ζs->0,{z,zs}];
125     lt = CF/@#&/@Inverse@Table[
126         δ[z, Dual[ζ]] - D[L,z,ζ],
127         {ζ,ζs},{z,zs}
128     ];
129     zrule = Thread[zs -> CF/@(lt . (zs + ys))];
130     Zrule = Join[zrule, zrule /.
131         r_Rule -> ( (U = r[[1]] /. {b -> B, t -> T, α -> A})
      ↪ ->
      (U /. U2l /. r //. l2U))
132     ];
133     \[Zeta]rule = Thread[\[Zeta]s -> \[Zeta]s + \[Eta]s . lt];
134     Q1 = Q /. Union[Zrule, ζrule];
135     EEQ[ps___] :=
136         EEQ[ps] = (
137             CF[E^-Q1 DD[E^Q1,Thread[{zs,{ps}}]] /.
138                 {Alternatives@@zs -> 0, Alternatives
139                 ↪ @@Zs -> 1}
140             );
141     CF@toPG[
142         c + ηs.lt.ys,
143         Q1 /. {Alternatives@@zs -> 0, Alternatives@@Zs -> 1},
144         Det[lt] (Zip[ζs]((EQ@@zs) (P /. Union[Zrule,ζrule]))
      ↪ /.
145             Derivative[ps___][EQ][___] -> EEQ[ps] /. _EQ
      ↪ ->1
146         )
147     ]
148
149 ]

```

The function `Pair` combines the above zipping functions into the final contraction map.

```

150 Pair[{}][L_PG,R_PG] := L R;
151 Pair[is_List][L_PG,R_PG] := Module[{n},
152     Times[
153         L /. ((v: b|B|t|T|a|x|y)[#] -> v[n@#]&/@is),
154         R /. ((v: β|τ|α|A|ξ|η)[#] -> v[n@#]&/@is)

```

```

155     ] // LZip[Join@@Table[Through[{\beta, \tau, a}[n@i]],{i, is}]] //
156     QZip[Join@@Table[Through[{\xi, y}[n@i]],{i, is}]]
157 ]

```

Our next task is to provide domain and codomain information for the PG-objects. These will be packaged inside a GDO, (Gaußian Differential Operator). The four lists' names refer to whether it is a domain or a codomain, and whether the index corresponds to an open strand or a closed one.

```

158 toGDO[do_List,dc_List,co_List,cc_List,L,Q,P_] := GDO[
159     "do" -> do,
160     "dc" -> dc,
161     "co" -> co,
162     "cc" -> cc,
163     "PG" -> toPG[L, Q, P]
164 ]
165
166 toGDO[do_List,dc_List,co_List,cc_List,pg_PG] := GDO[
167     "do" -> do,
168     "dc" -> dc,
169     "co" -> co,
170     "cc" -> cc,
171     "PG" -> pg
172 ]

```

Next are functions for accessing and modifying sub-parts of GDO-objects. The last argument of Lookup is the default value if nothing is specified. This means that a morphism with empty domain or codomain may be specified as such by omitting that portion of the definition.

```

173 getDO[gdo_GDO] := Lookup[Association@@gdo, "do", {}]
174 getDC[gdo_GDO] := Lookup[Association@@gdo, "dc", {}]
175 getCO[gdo_GDO] := Lookup[Association@@gdo, "co", {}]
176 getCC[gdo_GDO] := Lookup[Association@@gdo, "cc", {}]
177
178 getPG[gdo_GDO] := Lookup[Association@@gdo, "PG", PG[]]
179
180 getL[gdo_GDO] := gdo//getPG//getL
181 getQ[gdo_GDO] := gdo//getPG//getQ
182 getP[gdo_GDO] := gdo//getPG//getP

```

```

183
184 setPG[pg_PG][gdo_GD0] := setValue[pg, gdo, "PG"]
185
186 setL[L_][gdo_GD0] := setValue[setL[L][gdo//getPG], gdo, "PG"]
187 setQ[Q_][gdo_GD0] := setValue[setQ[Q][gdo//getPG], gdo, "PG"]
188 setP[P_][gdo_GD0] := setValue[setP[P][gdo//getPG], gdo, "PG"]
189
190 setD0[do_][gdo_GD0] := setValue[do, gdo, "do"]
191 setDC[dc_][gdo_GD0] := setValue[dc, gdo, "dc"]
192 setC0[co_][gdo_GD0] := setValue[co, gdo, "co"]
193 setCC[cc_][gdo_GD0] := setValue[cc, gdo, "cc"]
194
195 applyToD0[f_][gdo_GD0] := gdo//setD0[gdo//getD0//f]
196 applyToDC[f_][gdo_GD0] := gdo//setDC[gdo//getDC//f]
197 applyToC0[f_][gdo_GD0] := gdo//setC0[gdo//getC0//f]
198 applyToCC[f_][gdo_GD0] := gdo//setCC[gdo//getCC//f]
199
200 applyToPG[f_][gdo_GD0] := gdo//setPG[gdo//getPG//f]
201
202 applyToL[f_][gdo_GD0] := gdo//setL[gdo//getL//f]
203 applyToQ[f_][gdo_GD0] := gdo//setQ[gdo//getQ//f]
204 applyToP[f_][gdo_GD0] := gdo//setP[gdo//getP//f]

```

The canonical form function (CF) and the contraction mapping (Pair) we extend to include GDO-objects. Furthermore, on the level of GDO-objects we can compose morphisms and keep track of the corresponding domains and codomains, using the left-to-right composition operator “//”.

```

205 CF[e_GD0] := e//
206     applyToD0[Union]//
207     applyToDC[Union]//
208     applyToC0[Union]//
209     applyToCC[Union]//
210     applyToPG[CF]
211
212 Pair[is_List][gdo1_GD0, gdo2_GD0] := GDO[
213     "do" -> Union[gdo1//getD0, Complement[gdo2//getD0, is]],
214     "dc" -> Union[gdo1//getDC, gdo2//getDC],
215     "co" -> Union[gdo2//getC0, Complement[gdo1//getC0, is]],
216     "cc" -> Union[gdo1//getCC, gdo2//getCC],
217     "PG" -> Pair[is][gdo1//getPG, gdo2//getPG]

```

```

218 ]
219
220 gdo1_GDO // gdo2_GDO :=
  ↪ Pair[Intersection[gdo1//getC0,gdo2//getD0]][gdo1,gdo2];

```

We also define notions of equality and multiplication (by concatenation) for GDO's.

```

221 GDO /: Congruent[gdo1_GDO, gdo2_GDO] := And[
222     Sort@*getD0/@Equal[gdo1, gdo2],
223     Sort@*getDC/@Equal[gdo1, gdo2],
224     Sort@*getC0/@Equal[gdo1, gdo2],
225     Sort@*getCC/@Equal[gdo1, gdo2],
226     Congruent[gdo1//getPG, gdo2//getPG]
227 ]
228
229 GDO /: gdo1_GDO gdo2_GDO := GDO[
230     "do" -> Union[gdo1//getD0, gdo2//getD0],
231     "dc" -> Union[gdo1//getDC, gdo2//getDC],
232     "co" -> Union[gdo1//getC0, gdo2//getC0],
233     "cc" -> Union[gdo1//getCC, gdo2//getCC],
234     "PG" -> (gdo1//getPG)*(gdo2//getPG)
235 ]

```

For the sake of compatibility with Bar-Natan and van der Veen's program, we introduce several conversion functions between the two notations.

```

236 setEpsilonDegree[k_Integer][gdo_GDO] :=
237     setP[Series[Normal@getP@gdo, { $\epsilon$ , 0, k}]] [gdo]
238
239 fromE[Subscript[\[DoubleStruckCapitalE], {do_List, dc_List}->{co_List,
  ↪ cc_List}][
240     L_, Q_, P_
241 ]] := toGDO[do, dc, co, cc, fromE[\[DoubleStruckCapitalE][L, Q, P]]]
242
243 fromE[Subscript[\[DoubleStruckCapitalE], dom_List->cod_List][
244     L_, Q_, P_
245 ]] := GDO["do" -> dom, "co" -> cod,
246     "PG" -> fromE[\[DoubleStruckCapitalE][L, Q, P]]]
247 ]

```

It is at this point that we implement the morphisms of the algebra U . Each operation is prepended with a “c” to emphasize that this is a classical algebra, not a quantum deformation. These match the quantities given in theorem 3.5.

```

248 fromLog[l_] := CF@Module[
249     {L, l0 = Limit[l, ε->0]},
250     L = l0 /. (η|y|ξ|x)[_]->0;
251     PG[
252         "L" -> L,
253         "Q" -> l0 - L
254     ]/.l2U
255 ]
256
257 cΛ = (η[i] + E^(-γ α[i] - ε β[i]) η[j]/(1+γ ε η[j]ξ[i])) y[k] +
258     (β[i] + β[j] + Log[1 + γ ε η[j]ξ[i]]/ε ) b[k] +
259     (α[i] + α[j] + Log[1 + γ ε η[j]ξ[i]]/γ ) a[k] +
260     (ξ[j] + E^(-γ α[j] - ε β[j]) ξ[i]/(1+γ ε η[j]ξ[i])) x[k];
261
262 cm[i_, j_, k_] = GDO["do" -> {i,j}, "co" -> {k}, "PG" -> fromLog[cΛ]];
263
264 cη[i_] = GDO["co" -> {i}];
265 cσ[i_, j_] = GDO["do"->{i}, "co"->{j},
266     "PG" -> fromLog[β[i] b[j] + α[i] a[j] + η[i] y[j] + ξ[i] x[j]]
267 ];
268 ce[i_] = GDO["do" -> {i}];
269 cΔ[i_, j_, k_] = GDO["do"->{i}, "co"->{j, k},
270     "PG" -> fromLog[
271         β[i](b[j] + b[k]) +
272         α[i](a[j] + a[k]) +
273         η[i]
274             ((b[j]+b[k])/(1-B[j]B[k]))
275             (
276                 B[k>((1-B[j])/b[j])y[j]+
277                 ((1-B[k])/b[k])y[k]
278             ) +
279             ξ[i](x[j] + x[k])
280     ]
281 ];
282
283 sY[i_, j_, k_, l_, m_] = GDO["do"->{i}, "co"->{j, k, l, m},
284     "PG" -> fromLog[β[i]b[k] + α[i]a[l] + η[i]y[j] + ξ[i]x[m]]
285 ];

```

```

286
287 sS[i_] = GDO["do"->{i}, "co"->{i},
288     "PG"->fromLog[-(β[i] b[i] + α[i] a[i] + η[i] y[i] + ξ[i]
      ⇨ x[i])]
289 ];
290
291 cS[i_] = sS[i] // sY[i, 1, 2, 3, 4] // cm[4,3, i] // cm[i, 2, i] //
      ⇨ cm[i, 1, i];
292
293 cR[i_, j_] = GDO[
294     "co" -> {i,j},
295     "PG" -> toPG[ħ a[j] b[i], (B[i]-1)/(-b[i]) x[j] y[i], 1]
296 ]
297
298 cRi[i_, j_] = GDO[
299     "co" -> {i,j},
300     "PG" -> toPG[-ħ a[j] b[i], (B[i]-1)/(B[i] b[i]) x[j] y[i], 1]
301 ]
302
303 CC[i_] := GDO["co"->{i}, "PG"->PG["P"->B[i]^(1/2)]]
304 CCi[i_] := GDO["co"->{i}, "PG"->PG["P"->B[i]^(-1/2)]]
305
306 cKink[i_] = Module[{k}, cR[i,k] CCi[k] // cm[i, k, i]]
307 cKinki[i_] = Module[{k}, cRi[i,k] CC[k] // cm[i, k, i]]
308
309 cKinkn[0][i_] = cη[i]
310 cKinkn[1][i_] = cKink[i]
311 cKinkn[-1][i_] = cKinki[i]
312 cKinkn[n_Integer][i_] :=
      ⇨ Module[{j}, cKinkn[n-1][i] cKink[j] // cm[i, j, i]]; n > 1
313 cKinkn[n_Integer][i_] :=
      ⇨ Module[{j}, cKinkn[n+1][i] cKinki[j] // cm[i, j, i]]; n < -1
314
315 uR[i_, j_] = Module[{k}, cR[i,j] cKinki[k] // cm[i, k, i]]
316 uRi[i_, j_] = Module[{k}, cRi[i,j] cKink[k] // cm[i, k, i]]

```

A.2 IMPLEMENTATION OF THE TRACE

Now we implement the trace. We introduce several functions which extract the various coefficients of a GDO, so that we may apply equation (4.22).

Coefficients are extracted based on whether they belong to the matrix L or the matrix Q.

```

317 getConstLCoeff::usage = "getConstLCoeff[i][gdo] returns the terms in the
    ↪ L-portion of a GDO expression which are not a function of y[i],
    ↪ b[i], a[i], nor x[i]."
318 getConstLCoeff[i_][gdo_] :=
319     (SeriesCoefficient[#, {b[i],0,0}]&) @*
320     (Coefficient[#, y[i], 0]&) @*
321     (Coefficient[#, a[i], 0]&) @*
322     (Coefficient[#, x[i], 0]&) @*
323     ReplaceAll[U2l] @*
324     getL@
325     gdo
326
327 getConstQCoeff::usage = "getConstQCoeff[i][gdo] returns the terms in the
    ↪ Q-portion of a GDO expression which are not a function of y[i],
    ↪ b[i], a[i], nor x[i]."
328 getConstQCoeff[i_][gdo_][bb_] :=
329     ReplaceAll[{b[i]->bb}] @*
330     (Coefficient[#, y[i], 0]&) @*
331     (Coefficient[#, a[i], 0]&) @*
332     (Coefficient[#, x[i], 0]&) @*
333     ReplaceAll[U2l] @*
334     getQ@
335     gdo
336
337 getyCoeff::usage = "getyCoeff[i][gdo][b[i]] returns the linear
    ↪ coefficient of y[i] as a function of b[i]."
338 getyCoeff[i_][gdo_][bb_] :=
339     ReplaceAll[{b[i]->bb}] @*
340     ReplaceAll[U2l] @*
341     (Coefficient[#, x[i],0]&) @*
342     (Coefficient[#, y[i],1]&) @*
343     getQ@
344     gdo
345
346 getbCoeff::usage = "getbCoeff[i][gdo] returns the linear coefficient of
    ↪ b[i]."
347 getbCoeff[i_][gdo_] :=
348     (SeriesCoefficient[#, {b[i],0,1}]&) @*
349     (Coefficient[#, a[i],0]&) @*
350     (Coefficient[#, x[i],0]&) @*

```



```

351     (Coefficient[#, y[i],0]&) @*
352     ReplaceAll[U2l] @*
353     getL@
354     gdo
355
356 getPCoef::usage = "getPCoef[i][gdo] returns the perturbation P of a
    ↪ GDO as a function of b[i]."
357 getPCoef[i_][gdo_][bb_] :=
358     ReplaceAll[{b[i]->bb}] @*
359     (Coefficient[#, a[i],0]&) @*
360     (Coefficient[#, x[i],0]&) @*
361     (Coefficient[#, y[i],0]&) @*
362     ReplaceAll[U2l] @*
363     getP@
364     gdo
365
366 getaCoef::usage = "getaCoef[i][gdo] returns the linear coefficient of
    ↪ a[i]."
367 getaCoef[i_][gdo_] :=
368     (SeriesCoefficient[#, {b[i],0,0}]&) @*
369     (Coefficient[#, a[i],1]&) @*
370     ReplaceAll[U2l] @*
371     getL@
372     gdo
373
374 getxCoef::usage = "getxCoef[i][gdo][b[i]] returns the linear
    ↪ coefficient of x[i] as a function of b[i]."
375 getxCoef[i_][gdo_][bb_] :=
376     ReplaceAll[{b[i]->bb}] @*
377     ReplaceAll[U2l] @*
378     (Coefficient[#, y[i],0]&) @*
379     (Coefficient[#, x[i],1]&) @*
380     getQ@
381     gdo
382
383 getabCoef::usage = "getabCoef[i][gdo] returns the linear coefficient
    ↪ of a[i]b[i]."
384 getabCoef[i_][gdo_] :=
385     (SeriesCoefficient[#, {b[i],0,1}]&) @*
386     (Coefficient[#, a[i],1]&) @*
387     ReplaceAll[U2l] @*
388     getL@
389     gdo

```

```

390
391 getxyCoef::usage = "getxyCoef[i][gdo][b[i]] returns the linear
    ↪ coefficient of x[i]y[i] as a function of b[i]."
392 getxyCoef[i_][gdo_][bb_] :=
393     ReplaceAll[{b[i]->bb}] @*
394     ReplaceAll[U2l] @*
395     (Coefficient[#,x[i],1]&) @*
396     (Coefficient[#,y[i],1]&) @*
397     getQ@
398     gdo

```

In order to run more efficiently, limits are first computed by direct evaluation, unless such an operation is ill-defined. In such a case, the corresponding series is computed and evaluated at the limit point.

```

399 safeEval[f_][x_] := Module[{fx, x0},
400     If[(fx=Quiet[f[x]]) === Indeterminate,
401         Series[f[x0],{x0,x,0}]/Normal,
402         fx
403     ]
404 ]
405
406 closeComponent[i_][gdo_GDO]:=gdo//
407     setCO[Complement[gdo//getCO,{i}]]//
408     setCC[Union[gdo//getCC,{i}]]

```

Now we come to the implementation of the trace map. The current implementation requires that the coefficient of $a_i b_i$ be zero. (See section 4.2 for how this restriction limits computability.)

```

409 tr::usage = "tr[i] computes the trace of a GDO element on component i.
    ↪ Current implementation assumes the Subscript[a, i] Subscript[b, i]
    ↪ term vanishes and $k=0."
410 tr::nonzeroSigma = "tr[`1`]: Component `1` has writhe: `2`, expected:
    ↪ 0."
411 tr[i_][gdo_GDO] := Module[
412     {
413         cL = getConstLCoef[i][gdo],
414         cQ = getConstQCoef[i][gdo],
415         βP = getPCoef[i][gdo],

```

```

416         ηη = getyCoef[i][gdo],
417         ββ = getbCoef[i][gdo],
418         αα = getaCoef[i][gdo],
419         ξξ = getxCoef[i][gdo],
420         λ = getxyCoef[i][gdo],
421         ta
422     },
423     ta = (1-Exp[-αα]) z[i];
424     expL = cL + αα w[i] + ββ ta;
425     expQ = safeEval[cQ[#] + z[i]ηη[#]ξξ[#]/(1-z[i] λ[#])&][ta];
426     expP = safeEval[βP[#]/(1-z[i] λ[#])&][ta];
427     CF[(gdo//closeComponent[i]//
    ↪ setL[expL]//setQ[expQ]//setP[expP])//.l2U]
428 ] //; Module[
429     {σ = getabCoef[i][gdo]},
430     If[σ == 0,
431         True,
432         Message[tr::nonzeroSigma, i, ToString[σ]]; False
433     ]
434 ]

```

Here we introduce some formatting to display the output more aesthetically.

```

435 Format[gdo_GD0] := Subsuperscript[\[DoubleStruckCapitalE],
436     Row[{gdo//getC0, ",", gdo//getCC}],
437     Row[{gdo//getD0, ",", gdo//getDC}]
438 ][gdo//getL, gdo//getQ, gdo//getP];
439 Format[pg_PG] := \[DoubleStruckCapitalE][pg//getL, pg//getQ,
    ↪ pg//getP];
440
441 SubscriptFormat[v_] := (Format[v[i_]] := Subscript[v, i]);
442
443 SubscriptFormat/@{y,b,t,a,x,z,w,η,β,α,ξ,A,B,T,W};

```

A.3 DEFINING Z AND Z^{tr}

Now we are in a position to implement the Z invariant to tangles with a closed component. We begin by defining an object representing an isolated strand with arbitrary integer rotation number, CCn:

```

444 CCn[i_][n_Integer]:=Module[{j},
445   If[n==0,
446     GDO["co"->{i}],
447     If[n>0,
448       If[n==1,
449         CC[i],
450         CC[j]//CCn[i][n-1]//cm[i,j,i]
451       ],
452       If[n==-1,
453         CCi[i],
454         CCi[j]//CCn[i][n+1]//cm[i,j,i]
455       ]
456     ]
457   ]
458 ]

```

Since multiplication is associative, we may implement a generalized multiplication which can take any number of arguments. It is also named `cm`, with a first argument given as an ordered list of indices to be concatenated.

```

459 cm[{}, j_] := cη[j]
460 cm[{i_}, j_] := cσ[i,j]
461 cm[{i_, j_}, k_] := cm[i,j,k]
462 cm[ii_List, k_] := Module[
463   {
464     i = First[ii],
465     is = Rest[ii],
466     j ,
467     js ,
468     l
469   },
470   j = First[is];
471   js = Rest[is];
472   cm[i,j,l] // cm[Prepend[js, l], k]
473 ]

```

The function `toGDO` serves as the invariant for the generators of the tangles. We define its value on crossings and on concatenations of elements.

```

474 toGDO[Xp[i_,j_]] := cR[i,j]
475 toGDO[Xm[i_,j_]] := cRi[i,j]
476 toGDO[xs_Strand] := cm[List@@xs, First[xs]]
477 toGDO[xs_Loop]   := Module[{x = First[xs]}, cm[List@@xs, x]//tr[x]]
478
479 getIndices[RVT[cs_List, _List, _List]] := Sort@Catenate@(List@@@cs)
480
481 TerminalQ[cs_List][i_] := MemberQ[Last/@cs,i];
482 next[cs_List][i_] := If[TerminalQ[cs][i],
483     Nothing,
484     Extract[cs, ((#/.{c_,j_}->{c,j+1}& )@FirstPosition[i]@cs)]
485 ]
486
487 InitialQ[cs_List][i_] := MemberQ[First/@cs,i];
488 prev[cs_List][i_] := If[InitialQ[cs][i],
489     Nothing,
490     Extract[cs, ((#/.{c_,j_}->{c,j-1}& )@FirstPosition[i]@cs)]
491 ]

```

To minimize the size of computations, whenever adjacent indices are present in the partial computation, they are to be concatenated before more crossings are introduced.

```

492 MultiplyAdjacentIndices[{cs_List, calc_GDO}] := Module[
493     { is=getCO[calc]
494     , i
495     , i2
496     },
497     i = SelectFirst[is, MemberQ[is, next[cs][#]]&];
498     If[Head[i]===Missing,
499         {cs, calc},
500         i2 = next[cs][i];
501         {DeleteCases[cs, i2, 2], calc//cm[i, i2, i]}
502     ]
503 ]
504
505 MultiplyAllAdjacentIndices[{cs_List, calc_GDO}] :=
506     FixedPoint[MultiplyAdjacentIndices, {cs, calc}]
507
508 generateGDOFromXing[x: _Xp|_Xm, rs_Association] := Module[
509     {p, i, j, in, jn},
510     {i, j} = List@@x;

```

```

511     {in,jn} = Lookup[rs,{i,j},0];
512     toGDO[x]*CCn[p[i]][in]*CCn[p[j]][jn]
        ↪ //cm[p[i],i,i]//cm[p[j],j,j]
513 ]
514
515 addRotsToXingFreeStrands[rvt_RVT] := GDO[] * Times @@ (
516     CCn[#][Lookup[rvt[[3]], #, 0]] & /@
517     First /@ Select[rvt[[1]], Length[#] == 1 &]
518 )

```

Next we implement the framed link invariant Z^{Framed} .

```

519 ZFramedStep[[_List,{}],_Association,calc_GDO]:={{},{},<|>,calc};
520 ZFramedStep[{cs_List,xs_List,rs_Association,calc_GDO}]:=Module[
521     { x=First[xs], xss=Rest[xs]
522     , csOut, calcOut
523     , new
524     },
525     new=calc*generateGDOFromXing[x,rs];
526     {csOut,calcOut} = MultiplyAllAdjacentIndices[{cs,new}];
527     {csOut,xss,rs,calcOut}
528 ]
529
530 ZFramed[rvt_RVT] := Last@FixedPoint[ZFramedStep, {Sequence @@ rvt,
531     addRotsToXingFreeStrands[rvt]}]
532 ZFramed[L_] := ZFramed[toRVT@L]

```

Finally, when we wish to consider the unframed invariant, we apply the function Unwrite , defined below.

```

533 Z[rvt_RVT] := Unwrite@Last@FixedPoint[ZFramedStep, {Sequence @@ rvt,
        ↪ GDO[]}]
534 Z[L_] := Z[toRVT@L]
535
536 combineBySecond[l_List] := mergeWith[Total,#]& /@ GatherBy[l, First];
537 combineBySecond[lis___] := combineBySecond[Join[lis]]
538
539 mergeWith[f_, l_] := {l[[1, 1]], f@(#[[2]] & /@ l)}
540
541 Reindex[RVT[cs_, xs_, rs_]] := Module[
542     {

```

```

543   sf,
544   cs2, xs2, rs2,
545   repl, repl2
546 },
547 sf = Flatten[List@@#&/@cs];
548 repl = (Thread[sf -> Range[Length[sf]]]);
549 repl2 = repl /. {(a_ -> b_) -> ({a, i_} -> {b, i})};
550 cs2 = cs /. repl;
551 xs2 = xs /. repl;
552 rs2 = rs /. repl2;
553 RVT[cs2, xs2, rs2]
554 ]
555
556 UnwriteComp[i_][gdo_GDO] := Module[
557     {n = gdo//getL//SeriesCoefficient[#, {a[i]b[i], 0, 1}]&, j},
558     gdo//(cKinkn[-n][j])//cm[i, j, i]
559 ]
560
561 Unwrite[gdo_GDO] := (Composition@@(UnwriteComp/@(gdo//getCO)))@gdo
562
563 toRVT[L_RVT] := L

```

The partial trace is what we use to close a subset of the strands in a tangle. It takes the trace of all but one component, then returns the collection of all such ways of leaving one component open (as described in section 5.2).

```

564 ptr[L_RVT] := Module[
565     {
566         ZL = Z[L],
567         cod
568     },
569     cod = getCO@ZL;
570     Table[(Composition@@Table[tr[j],
571         ↪ {j, Complement[cod, {i}]}])[ZL], {i, cod}]
571 ]
572 ptr[L_] := ptr[toRVT[L]]

```

In order to be able to compare GDO's properly, we require a way to canonically represent them. This is achieved by reindexing the strands of the link and selecting one who's resulting invariant comes first in an (arbitrarily-

selected) order, in this case the built-in ordering of expressions as defined by Mathematica™.

```

573 getGDOIndices[gdo_GDO]:=Sort@Catenate@Through[{getD0, getDC, getC0,
    ↪ getCC}@gdo]
574
575 isolateVarIndices[i_ -> j_] :=
    ↪ (v:y|b|t|a|x|η|β|α|ξ|A|B|T|w|z|W)[i]->v[j];
576
577 ReindexBy[f_][gdo_GDO] := Module[
578     {
579     replacementRules,
580     varIndexFunc,
581     repFunc,
582     indices = getGDOIndices[gdo]
583     },
584     replacementRules = Thread[indices->(f/@indices)];
585     repFunc = ReplaceAll[replacementRules];
586     varIndexFunc =
    ↪ ReplaceAll[Thread[isolateVarIndices[replacementRules]]];
587     gdo//applyToPG[varIndexFunc]//
588         applyToC0[repFunc]//
589         applyToDo[repFunc]//
590         applyToDC[repFunc]//
591         applyToCC[repFunc]
592 ]
593
594 fromAssoc[ass_] := Association[ass][#] &
595
596 ReindexToInteger[gdos_List] := Module[
597     {is = getGDOIndices@gdos[[1]], f},
598     f = fromAssoc@Thread[is -> Range[Length[is]]];
599     ReindexBy[f]/@gdos
600 ]
601
602 getReindications[gdos_List] := Module[
603     {
604     gdosInt = ReindexToInteger[gdos],
605     is,
606     fs,
607     ls
608     },
609     is = getGDOIndices[gdosInt[[1]]];

```



```

610     fs = (fromAssoc@*Association@*Thread)/@(is -> # & /@
        ↪ Permutations[is]);
611     ls = CF@ReindexBy[#]/@gdoInt&/@fs;
612     Sort[Sort/@ls]
613 ]
614
615 getCanonicalIndex[gdo_] := First@getReindications@gdo
616
617 deleteIndex[i_][expr_] := SeriesCoefficient[expr/.U2l, Sequence @@
        ↪ ({#[i], 0, 0} & /@ {
618     y, b, t, a, x, z, w
619 })]/.l2U

```

Here we introduce functions to further verify the co-algebra structure of a traced ribbon meta-Hopf algebra. In particular, the counit is responsible for deleting a strand. This has further applications in determining whether the invariants of individual components are contained in those of more complex links.

```

620 deleteIndexPG[i_][pg_PG] := pg//
621     applyToL[deleteIndex[i]]//
622     applyToQ[deleteIndex[i]]//
623     applyToP[deleteIndex[i]]
624
625 deleteLoop[i_][gdo_] := gdo//
626     applyToCC[Complement[#, {i}]&]//
627     applyToPG[deleteIndexPG[i]]

```

A.4 IMPLEMENTATION OF ROTATION NUMBER ALGORITHM

Description of algorithm for knots

Bar-Natan and van der Veen develop an algorithm to convert a classical long knot into an upright tangle. It involves passing a line segment, called the front, over the knot, requiring that everything behind the front is in upright form. For example, consider the link in figure A.2. By pulling the crossings along the arc which touches the front, we can bring the knot to

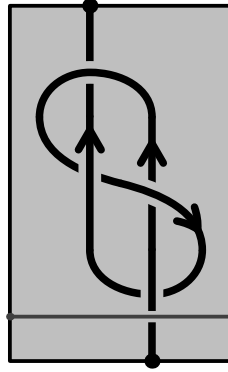


Figure A.2: A knot which is not in upright form. The front is written in dark grey.

upright form. The crossings are absorbed into the front in the order the knot's strand interacts with them.

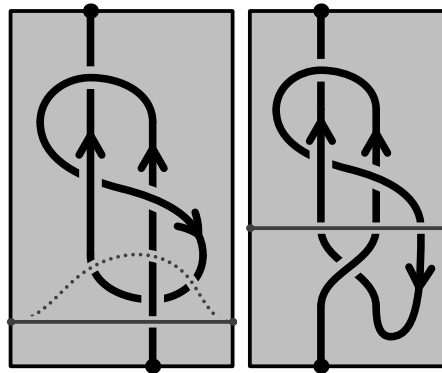


Figure A.3: By advancing the front over a crossing, we bring a crossing into upright form. A dashed line indicates where the front is advancing to.

Extending the algorithm to multiple components

Here we generalize the algorithm to convert a classical tangle with one open component to an upright tangle diagram. This generalization allows us to consider tangle diagrams with multiple components.

Uniqueness of the resulting tangle follows from the following lemma:

Lemma A.1 ([BNvdVb], Lemma 43). *For each classical tangle with one open component, there exists a unique upright tangle whose unbounded arcs have rotation numbers 0.*

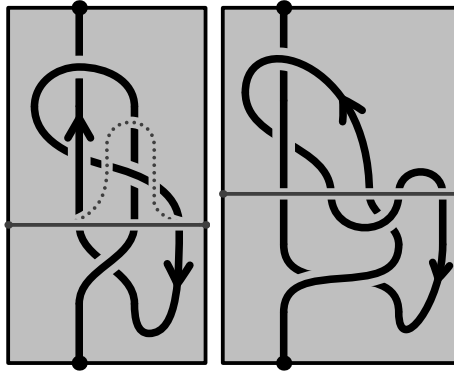


Figure A.4: By advancing the front over a crossing, we bring a crossing into upright form. A dashed front indicates where the front is advancing to.

This is a Haskell implementation^{†1} of the algorithm `toRVT`,^{†2} which takes a classical tangle and produces an upright tangle by computing a compatible choice of rotation numbers for each arc. This follows largely the same logic as above, except the leftmost strand is always prioritized for absorption, regardless of which component it belongs to.

Use case

For example, to query the **SX**-form of a link (i.e. its skeleton-crossing form), one writes:

```
1 >>> link 4 True 1
```

```
2 SX [Loop [1,2,3,4], Loop [5,6,7,8]]
3   [Xm 1 6, Xm 3 8, Xm 5 2, Xm 7 4]
```

To convert from the **SX** form to an upright tangle form (here written **RVT**), we must first replace one of the closed **Loops** with an open **Strand** (accomplished by `openFirstStrand`):

```
4 >>> toRVT . openFirstStrand $ link 4 True 1
```

^{†1} The full source code is available at <https://github.com/phro/KnotTheory>.

^{†2} Here, the acronym RVT stands for “Rotational Virtual Tangle”, which is another term for “upright tangle”.

```

5 RVT [Strand [1,2,3,4], Loop [5,6,7,8]]
6   [Xm 1 6, Xm 3 8, Xm 5 2, Xm 7 4]
7   [(5, -1), (6, 1), (8, 1)]

```

Reading off the final line, we see that arc 5 has rotation number -1 , arcs 6 and 8 have rotation number 1, and the rest of the arcs have rotation number 0.

Implementation

We begin with a series of imports of common functions, relating to list manipulations and type-wrangling. The exact details are not too important.

```

8 {-# LANGUAGE DeriveFunctor #-}
9 module KnotTheory.PD where
10 import Data.Maybe (listToMaybe, catMaybes, mapMaybe, fromMaybe,
   ↪ fromJust)
11 import Data.List (find, groupBy, sortOn, partition, intersect, union,
   ↪ (\))
12 import Data.Tuple (swap)
13 import Data.Function (on)
14 import Control.Monad ((>=>))
15 import Control.Arrow ((>>>))

```

Next, we introduce the crossing type, which can be either positive **Xp** or negative **Xm** (using the mnemonic “plus” and “minus”):

```

16 type Index = Int
17 data Xing i = Xp i i | Xm i i -- | Xv i i
18 deriving (Eq, Show, Functor)

```

We define several functions which extract basic data from a crossing.

```

19 sign :: (Integral b) => Xing Index -> b
20 sign (Xp _ _) = 1
21 sign (Xm _ _) = -1
22
23 isPositive :: Xing i -> Bool

```

```

24 isPositive (Xp _ _) = True
25 isPositive (Xm _ _) = False
26
27 isNegative :: Xing i -> Bool
28 isNegative (Xp _ _) = False
29 isNegative (Xm _ _) = True
30
31 overStrand :: Xing i -> i
32 overStrand (Xp i _) = i
33 overStrand (Xm i _) = i
34
35 underStrand :: Xing i -> i
36 underStrand (Xp _ i) = i
37 underStrand (Xm _ i) = i

```

Next, we introduce the notion of a planar diagram, whose data is comprised of a collection of **Strands** and **Loops** (indexed by some type **i**, typically an integer). The **Skeleton** of a planar diagram is defined to be the collection of **Components**, each of which is either an open **Strand** or a closed **Loop**.

```

38 type Strand i = [i]
39 type Loop i = [i]
40 data Component i = Strand (Strand i) | Loop (Loop i)
41 deriving (Eq, Show, Functor)
42 type Skeleton i = [Component i]

```

Next, we introduce the notion of a **KnotObject**, which has its components labelled by the same type **i**. We further define a function **toRVT** which converts a generic **KnotObject** into an upright tangle. (In this codebase, the term Rotational Virtual Tangle is frequently used for the notion of an upright tangle.) We call an object a planar diagram (or **PD**) if it has a notion of **Skeleton** and a collection of crossings.

```

43 class KnotObject k where
44   toSX :: (Ord i) => k i -> SX i
45   toRVT :: (Ord i) => k i -> RVT i
46   toRVT = toRVT . toSX
47
48 class PD k where

```

```

49 skeleton :: k i -> Skeleton i
50 xings    :: k i -> [Xing i]

```

The **SX** form of a diagram just contains the **Skeleton** and the **Xings** (crossings), while the **RVT** form also assigns each arc an integral rotation number.

```

51 data SX i = SX (Skeleton i) [Xing i] deriving (Show, Eq, Functor)
52 data RVT i = RVT (Skeleton i) [Xing i] [(i,Int)] deriving (Show, Eq,
  ↪ Functor)

```

Given any labelling of the arcs in a diagram, we can re-label the arcs using consecutive whole numbers. This is accomplished with `reindex`:

```

53 reindex :: (PD k, Functor k, Eq i) => k i -> k Int
54 reindex k = fmap (fromJust . flip lookup table) k
55   where
56     table = zip (skeletonIndices s) [1..]
57     s = skeleton k

```

Most importantly, we now declare that a diagram expressed in **SX** form (that is, without any rotation data) may be assigned rotation numbers to each of its arcs in a meaningful way. The bulk of the work is done by `getRotNums`, which is defined farther below. We handle the case where the entire tangle is a single crossingless strand separately.

```

58 instance KnotObject SX where
59   toSX = id
60   toRVT k@(SX cs xs) = RVT cs xs rs where
61     rs = filter ((/=0) . snd) . mergeBy sum $ getRotNums k f1
62     i1 = head . toList $ s
63     Just s = find isStrand cs
64     f1 = case next i1 (toList s) of
65         Just _ -> [(Out,i1)]
66         Nothing -> []
67
68 instance KnotObject RVT where
69   toRVT = id
70   toSX (RVT s xs _) = SX s xs

```

```

71
72 instance PD SX where
73   skeleton (SX s _) = s
74   xings (SX _ xs)   = xs
75
76 instance PD RVT where
77   skeleton (RVT s _ _) = s
78   xings (RVT _ xs _)   = xs

```

Next, we include a series of functions which answer basic questions about planar diagrams. Note in `rotnum`, if a rotation number is not present in the table of values, it is assumed to be 0.

```

79 rotnums :: RVT i -> [(i,Int)]
80 rotnums (RVT _ _ rs) = rs
81
82 rotnum :: (Eq i) => RVT i -> i -> Int
83 rotnum k i = fromMaybe 0 . lookup i . rotnums $ k
84
85 isStrand :: Component i -> Bool
86 isStrand (Strand _) = True
87 isStrand _          = False
88
89 isLoop :: Component i -> Bool
90 isLoop (Loop _) = True
91 isLoop _       = False
92
93 toList :: Component i -> [i]
94 toList (Strand is) = is
95 toList (Loop is)   = is
96
97 skeletonIndices :: Skeleton i -> [i]
98 skeletonIndices = concatMap toList
99
100 involves :: (Eq i) => Xing i -> i -> Bool
101 x `involves` k = k `elem` [underStrand x, overStrand x]
102
103 otherArc :: (Eq i) => Xing i -> i -> Maybe i
104 otherArc x i
105 | i == o    = Just u
106 | i == u    = Just o
107 | otherwise = Nothing

```

```

108  where o = overStrand x
109         u = underStrand x
110
111  next :: (Eq i) => i -> Strand i -> Maybe i
112  next e = listToMaybe . drop 1 . dropWhile (/= e)
113
114  prev :: (Eq i) => i -> Strand i -> Maybe i
115  prev e = next e . reverse
116
117  nextCyc :: (Eq i) => i -> Loop i -> Maybe i
118  nextCyc e xs = next e . take (length xs + 1). cycle $ xs
119
120  prevCyc :: (Eq i) => i -> Loop i -> Maybe i
121  prevCyc e xs = prev e . take (length xs + 1). cycle $ xs
122
123  isHeadOf :: (Eq i) => i -> [i] -> Bool
124  x `isHeadOf` ys = x == head ys
125
126  isLastOf :: (Eq i) => i -> [i] -> Bool
127  x `isLastOf` ys = x == last ys
128
129  nextComponentIndex :: (Eq i) => i -> Component i -> Maybe i
130  nextComponentIndex i (Strand is) = next i is
131  nextComponentIndex i (Loop is) = nextCyc i is
132
133  prevComponentIndex :: (Eq i) => i -> Component i -> Maybe i
134  prevComponentIndex i (Strand is) = prev i is
135  prevComponentIndex i (Loop is) = prevCyc i is
136
137  isHeadOfComponent :: (Eq i) => i -> Component i -> Bool
138  isHeadOfComponent _ (Loop _) = False
139  isHeadOfComponent i (Strand is) = i `isHeadOf` is
140
141  isLastOfComponent :: (Eq i) => i -> Component i -> Bool
142  isLastOfComponent _ (Loop _) = False
143  isLastOfComponent i (Strand is) = i `isLastOf` is
144
145  isTerminalOfComponent :: (Eq i) => Component i -> i -> Bool
146  isTerminalOfComponent c i = i `isHeadOfComponent` c || i
    ↪ `isLastOfComponent` c
147
148  isTerminalIndex :: (Eq i) => Skeleton i -> i -> Bool
149  isTerminalIndex cs i = any (`isTerminalOfComponent` i) cs

```



```

150
151 nextSkeletonIndex :: (Eq i) => Skeleton i -> i -> Maybe i
152 nextSkeletonIndex s i = listToMaybe . mapMaybe (nextComponentIndex i)
    ↪ $ s
153
154 prevSkeletonIndex :: (Eq i) => Skeleton i -> i -> Maybe i
155 prevSkeletonIndex s i = listToMaybe . mapMaybe (prevComponentIndex i)
    ↪ $ s

```

In order to obtain all the crossing indices, we must take every combination of the under- and over-strands and their following indices:

```

156 getXingIndices :: (Eq i) => Skeleton i -> Xing i -> [i]
157 getXingIndices s x = catMaybes
158     [ f a | f <- [id, (>=> nextSkeletonIndex s)], a <- [o, u] ]
159     where o = return (overStrand x)
160           u = return (underStrand x)
161
162 Ⓟ :: (Eq a) => a -> a -> Int
163 Ⓟ x y
164 | x == y     = 1
165 | otherwise  = 0
166
167 mergeBy :: (Ord i) => ([a] -> b) -> [(i,a)] -> [(i,b)]
168 mergeBy f = map (wrapIndex f) . groupBy ((==) `on` fst) . sortOn fst
169 where
170   wrapIndex :: ([a] -> b) -> [(i,a)] -> (i,b)
171   wrapIndex g xs@(x:_) = (fst x, g . map snd $ xs)

```

Here we come to the main function, `getRotNums`, for which we have the following requirements (not expressed in the code):

1. The diagram k is a $(1, n)$ -tangle (a tangle with only one open component)
2. The underlying graph of k is a planar.
3. The diagram k is a connected.

Only in this case will the function `toRVT` output a planar $(1, n)$ -upright tangle which corresponds to a classical (i.e. planar) diagram.

This function involves taking a simple open curve (a Jordan curve passing through infinity) called the **Front**, and passing it over arcs in the diagram.

This curve is characterized by the arcs it passes through, together with their orientations. Each intersection of the **Front** with the diagram provides a different **View**, either **In** or **Out** of the **Front** when following the orientation of the intersecting arc.

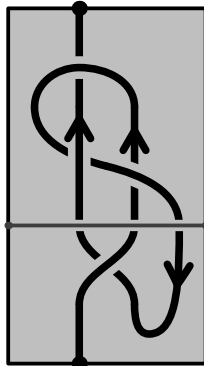


Figure A.5: A tangle with a front passing over it. The portion of the tangle below the front has all crossings in upright form. All arcs fully below the front have an integer rotation number.

```
172 type Front i = [View i]
173 type View i = (Dir, i)
```

We obtain the rotation numbers by successively passing the front across new crossings (achieved by `advanceFront`), keeping track of the rotation numbers of arcs which have already passed by the front. Once the front has passed across every crossing, all the rotation numbers have been computed.

Next, we define `converge`, which iterates a function until a fixed point is achieved.

```
174 converge :: (Eq a) => (a -> a) -> a -> a
175 converge f x
176     | x == x'   = x
177     | otherwise = converge f x'
178     where x' = f x
```

The function `convergeT` wraps `converge` in monadic transformations. In our context, the monad will be used to keep track of rotation numbers of the arcs.

```

179 convergeT :: (Monad m, Eq (m a)) => (a -> m a) -> a -> m a
180 convergeT f = return >>> converge (>= f)

```

The implementation of `getRotNums` takes a front and advances it along a diagram until no more changes occur.

```

181 getRotNums :: (Eq i) => SX i -> Front i -> [(i,Int)]
182 getRotNums k = convergeT (advanceFront k) >>> fst

```

When advancing the **Front**, we start by absorbing arcs that intersect with the front twice until the leftmost **View** no longer connects directly back to the **Front**. At this point, we can absorb a crossing into the front.

```

183 advanceFront :: (Eq i) => SX i -> Front i -> [(i,Int)], Front i
184 advanceFront k = convergeT (absorbArc k) >=> absorbXing k

```

We next check for the case where the leftmost arc connects back to the **Front**. If it is pointing **Out** (and therefore connects back **In** further to the right), we adjust the rotation number of the arc by -1 . Otherwise, we leave both the **Front** and the rotation numbers unchanged.

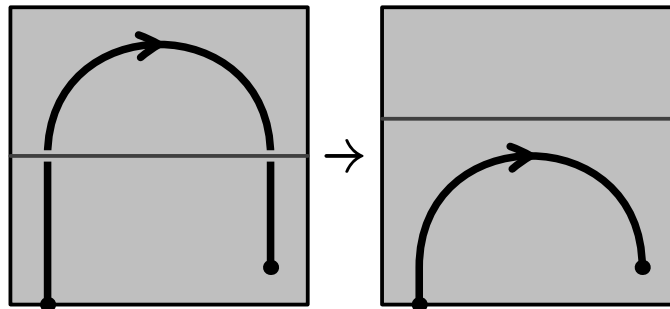


Figure A.6: Example of absorbing an arc which intersects the front multiple times. If the horizontal tangent vector points to the right, as in this picture, then the rotation number of the arc is decreased by 1. Otherwise, no change in the rotation number is recorded.

```

185 absorbArc :: (Eq i) => SX i -> Front i -> [(i,Int)], Front i
186 absorbArc k [] = return []
187 absorbArc k f@(f1:fs) = case fs1 of

```

```

188     ( In,i):_ -> (return (i,-1), fss)
189     (Out,i):_ -> return fss           -- No new rotation numbers
190     []      -> return f
191     where (fs1,fss) = partition (((==) `on` snd) f1) fs

```

Our goal is to repeat this operation until we get a fixed point, which is encoded in `absorbArcs`:

```

192 absorbArcs :: (Eq i) => SX i -> Front i -> [(i,Int)],Front i
193 absorbArcs k = convergeT (absorbArc k)

```

Absorb a crossing involves expanding one's view at an arc from looking at a crossing to all the views one gets when looking in every direction at the crossing (namely, to the left, along the arc, and to the right). The function `absorbXing` performs this task on the leftmost **View** on the **Front**. The transverse strand receives a positive rotation number if it moves from left to right. The arc receiving the rotation depends on how the crossing is oriented.

```

194 absorbXing :: (Eq i) => SX i -> Front i -> [(i,Int)],Front i
195 absorbXing _ [] = return []
196 absorbXing k (f:fs) = (rs,newFront++fs) where
197     newFront = catMaybes [l, a, r]
198     l = lookLeft k f
199     a = lookAlong k f
200     r = lookRight k f
201     rs = case (l,f,r) of
202         (Just (In,i), (Out,_),_) -> [(i,1)]
203         (_, (In ,_),Just (Out, j)) -> [(j,1)]
204         _ -> []
205
206 data Dir = In | Out
207 deriving (Eq, Show)

```

The following functions take a **View**, returning the **View** one has when looking in the corresponding direction. Since it is possible for the resulting gaze to be merely the boundary, it is possible for these functions to return **Nothing**.

```

208 lookAlong :: (Eq i, PD k) => k i -> View i -> Maybe (View i)
209 lookAlong k (d, i) = case d of
210     Out -> sequence (Out, nextSkeletonIndex s i)
211     In  -> sequence (In , prevSkeletonIndex s i)
212     where s = skeleton k
213
214 lookSide :: (Eq i, PD k) => Bool -> k i -> View i -> Maybe (View i)
215 lookSide isLeft k di@(Out,i) = do
216     x <- findNextXing k di
217     j <- otherArc x i
218     if isLeft == ((underStrand x == i) == isPositive x)
219     then return (In, j)
220     else sequence (Out, nextSkeletonIndex (skeleton k) j)
221 lookSide isLeft k (In,i) =
222     sequence (Out, prevSkeletonIndex (skeleton k) i) >>=
223     lookSide (not isLeft) k
224
225 lookLeft  :: (Eq i, PD k) => k i -> View i -> Maybe (View i)
226 lookLeft = lookSide True
227
228 lookRight :: (Eq i, PD k) => k i -> View i -> Maybe (View i)
229 lookRight = lookSide False
230
231 findNextXing :: (Eq i, PD k) => k i -> View i -> Maybe (Xing i)
232 findNextXing k (Out,i) = find (`involves` i) $ xings k
233 findNextXing k (In ,i) = do
234     i' <- prevSkeletonIndex (skeleton k) i
235     find (`involves` i') $ xings k

```

TABLE OF VALUES

Here we include the table of values for the partial trace map. The full table up to 11 crossings is available at <https://github.com/phro/GD0/blob/main/misc/ptrs-table.m>, but here we include only the values for links with up to 8 crossings.

Table B.1: Values of the partial trace invariant on links up to 8 crossings.

L_{2a1}	$((\exp((a_1(z_2 - B_1 z_2))(B_1)^{-1} - b_1 w_2 + (\exp(-(z_2)(B_1)^{-1})(x_1 y_1 \exp((z_2)(B_1)^{-1}) - x_1 y_1 \exp(z_2)))(b_1)^{-1} \exp((z_2)(2B_1)^{-1} - (z_2)(2)^{-1}))_{(\{1\}, \{2\})}, (\exp((a_2(z_1 - B_2 z_1))(B_2)^{-1} - b_2 w_1 + (\exp(-(z_1)(B_2)^{-1})(x_2 y_2 \exp((z_1)(B_2)^{-1}) - x_2 y_2 \exp(z_1)))(b_2)^{-1} \exp((z_1)(2B_2)^{-1} - (z_1)(2)^{-1}))_{(\{2\}, \{1\})})$
L_{4a1}	$((\exp((a_1(2z_2 - 2B_1^2 z_2))(B_1^2)^{-1} - 2b_1 w_2 + (\exp(-(2z_2)(B_1^2)^{-1})(x_1 y_1 \exp((2z_2)(B_1^2)^{-1}) - x_1 y_1 \exp(2z_2)))(b_1)^{-1} (B_1 \exp((3z_2)(2B_1^2)^{-1}) + \exp((3z_2)(2B_1^2)^{-1})) (B_1 \exp((3z_2)(2)^{-1}) + \exp((z_2)(B_1^2)^{-1} + (z_2)(2)^{-1}))^{-1})_{(\{1\}, \{2\})}, (\exp((a_2(2z_1 - 2B_2^2 z_1))(B_2^2)^{-1} - 2b_2 w_1 + (\exp(-(2z_1)(B_2^2)^{-1})(x_2 y_2 \exp((2z_1)(B_2^2)^{-1}) - x_2 y_2 \exp(2z_1)))(b_2)^{-1} (B_2 \exp((3z_1)(2B_2^2)^{-1}) + \exp((3z_1)(2B_2^2)^{-1})) (B_2 \exp((3z_1)(2)^{-1}) + \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}))^{-1})_{(\{2\}, \{1\})})$
L_{5a1}	$(((B_1)(B_1^2 z_2 - 2B_1 z_2 + B_1 + z_2)^{-1})_{(\{1\}, \{2\})}, ((B_2)(B_2^2 z_1 - 2B_2 z_1 + B_2 + z_1)^{-1})_{(\{2\}, \{1\})})$
L_{6a1}	$((\exp((a_1(2z_2 - 2B_1^2 z_2))(B_1^2)^{-1} - 2b_1 w_2 + (\exp(-(2z_2)(B_1^2)^{-1})(x_1 y_1 \exp((2z_2)(B_1^2)^{-1}) - x_1 y_1 \exp(2z_2)))(b_1)^{-1} (B_1(-\exp((3z_2)(2B_1^2)^{-1})) - \exp((3z_2)(2B_1^2)^{-1})) (-2B_1 \exp((3z_2)(2)^{-1}) + B_1 \exp((z_2)(B_1^2)^{-1} + (z_2)(2)^{-1} - 2 \exp((z_2)(B_1^2)^{-1} + (z_2)(2)^{-1} + \exp((3z_2)(2)^{-1}))^{-1})_{(\{1\}, \{2\})}, (\exp((a_2(2z_1 - 2B_2^2 z_1))(B_2^2)^{-1} - 2b_2 w_1 + (\exp(-(2z_1)(B_2^2)^{-1})(x_2 y_2 \exp((2z_1)(B_2^2)^{-1}) - x_2 y_2 \exp(2z_1)))(b_2)^{-1} (B_2(-\exp((3z_1)(2B_2^2)^{-1})) - \exp((3z_1)(2B_2^2)^{-1})) (-2B_2 \exp((3z_1)(2)^{-1}) + B_2 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1} - 2 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1} + \exp((3z_1)(2)^{-1}))^{-1})_{(\{2\}, \{1\})})$
Continued on next page	

Table B.1– continued from previous page

Link	Partial trace Z^{tr} of link
L_{6a2}	$ \begin{aligned} & ((\exp((a_1(3z_2 - 3B_1^3 z_2))(B_1^3)^{-1} - 3b_1 w_2 + (\exp(-(3z_2)(B_1^3)^{-1})(x_1 y_1 \exp((3z_2)(B_1^3)^{-1}) - \\ & x_1 y_1 \exp(3z_2)))(b_1)^{-1})(B_1^2 \exp((5z_2)(2B_1^3)^{-1}) + B_1 \exp((5z_2)(2B_1^3)^{-1}) + \\ & \exp((5z_2)(2B_1^3)^{-1}))(B_1^2 \exp((z_2)(B_1^3)^{-1} + (3z_2)(2)^{-1}) + B_1 \exp((5z_2)(2)^{-1}) - B_1 \exp((z_2)(B_1^3)^{-1} + (3z_2)(2)^{-1}) + \\ & B_1 \exp((2z_2)(B_1^3)^{-1} + (z_2)(2)^{-1}) + \exp((z_2)(B_1^3)^{-1} + (3z_2)(2)^{-1}))^{-1}_{(\{1\},\{2\})}, (\exp((a_2(3z_1 - 3B_2^3 z_1))(B_2^3)^{-1} - \\ & 3b_2 w_1 + (\exp(-(3z_1)(B_2^3)^{-1})(x_2 y_2 \exp((3z_1)(B_2^3)^{-1}) - x_2 y_2 \exp(3z_1)))(b_2)^{-1})(B_2^2 \exp((5z_1)(2B_2^3)^{-1}) + \\ & B_2 \exp((5z_1)(2B_2^3)^{-1}) + \exp((5z_1)(2B_2^3)^{-1}))(B_2^2 \exp((z_1)(B_2^3)^{-1} + (3z_1)(2)^{-1}) + B_2 \exp((5z_1)(2)^{-1}) - \\ & B_2 \exp((z_1)(B_2^3)^{-1} + (3z_1)(2)^{-1}) + B_2 \exp((2z_1)(B_2^3)^{-1} + (z_1)(2)^{-1}) + \exp((z_1)(B_2^3)^{-1} + (3z_1)(2)^{-1}))^{-1}_{(\{2\},\{1\})}) \end{aligned} $
L_{6a3}	$ \begin{aligned} & ((\exp((a_1(3z_2 - 3B_1^3 z_2))(B_1^3)^{-1} - 3b_1 w_2 + (\exp(-(3z_2)(B_1^3)^{-1})(x_1 y_1 \exp((3z_2)(B_1^3)^{-1}) - \\ & x_1 y_1 \exp(3z_2)))(b_1)^{-1})(B_1^2 \exp((5z_2)(2B_1^3)^{-1}) + B_1 \exp((5z_2)(2B_1^3)^{-1}) + \\ & \exp((5z_2)(2B_1^3)^{-1}))(B_1^2 \exp((2z_2)(B_1^3)^{-1} + (z_2)(2)^{-1}) + B_1 \exp((z_2)(B_1^3)^{-1} + \\ & (3z_2)(2)^{-1}) + \exp((5z_2)(2)^{-1}))^{-1}_{(\{1\},\{2\})}, (\exp((a_2(3z_1 - 3B_2^3 z_1))(B_2^3)^{-1} - 3b_2 w_1 + \\ & (\exp(-(3z_1)(B_2^3)^{-1})(x_2 y_2 \exp((3z_1)(B_2^3)^{-1}) - x_2 y_2 \exp(3z_1)))(b_2)^{-1})(B_2^2 \exp((5z_1)(2B_2^3)^{-1}) + \\ & B_2 \exp((5z_1)(2B_2^3)^{-1}) + \exp((5z_1)(2B_2^3)^{-1}))(B_2^2 \exp((2z_1)(B_2^3)^{-1} + (z_1)(2)^{-1}) + B_2 \exp((z_1)(B_2^3)^{-1} + \\ & (3z_1)(2)^{-1}) + \exp((5z_1)(2)^{-1}))^{-1}_{(\{2\},\{1\})}) \end{aligned} $
L_{7a1}	$ ((B_1)(B_1^2 z_2 - 2B_1 z_2 + B_1 + z_2)^{-1})_{(\{1\},\{2\})}, ((B_2^2)(B_2^4 z_1 - 3B_2^3 z_1 + 4B_2^2 z_1 - 3B_2 z_1 + B_2^2 + z_1)^{-1})_{(\{2\},\{1\})}) $
Continued on next page	

Table B.1– continued from previous page

Link	Partial trace Z^{tr} of link
L_{7a2}	$\begin{aligned} & ((\exp((a_1(2z_2 - 2B_1^2 z_2))(B_1^2)^{-1} - 2b_1 w_2 + (\exp(-(2z_2)(B_1^2)^{-1})(x_1 y_1 \exp((2z_2)(B_1^2)^{-1}) - \\ & x_1 y_1 \exp(2z_2))) (b_1)^{-1})(B_1 \exp((5z_2)(2B_1^2)^{-1} + (z_2)(2)^{-1}) + \exp((5z_2)(2B_1^2)^{-1} + (z_2)(2)^{-1}))(B_1 \exp(3z_2) - \\ & 2B_1 \exp((z_2)(B_1^2)^{-1} + 2z_2) + 2B_1 \exp((2z_2)(B_1^2)^{-1} + z_2) + \exp((3z_2)(B_1^2)^{-1} + 2 \exp((z_2)(B_1^2)^{-1} + \\ & 2z_2) - 2 \exp((2z_2)(B_1^2)^{-1} + z_2))^{-1})_{\{\{1\}, \{2\}\}}, (\exp((a_2(2z_1 - 2B_2^2 z_1))(B_2^2)^{-1} - 2b_2 w_1 + \\ & (\exp(-(2z_1)(B_2^2)^{-1})(x_2 y_2 \exp((2z_1)(B_2^2)^{-1}) - x_2 y_2 \exp(2z_1))) (b_2)^{-1})(B_2^2 \exp((3z_1)(2B_2^2)^{-1}) + \\ & B_2 \exp((3z_1)(2B_2^2)^{-1}))(B_2^3 \exp((3z_1)(2)^{-1}) - 2B_2^2 \exp((3z_1)(2)^{-1}) + 2B_2^2 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}) + \\ & 2B_2 \exp((3z_1)(2)^{-1}) - 2B_2 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}) + \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}))^{-1})_{\{\{2\}, \{1\}\}} \end{aligned}$
L_{7a3}	$\begin{aligned} & ((-(B_1)(2B_1^2 z_2 - 4B_1 z_2 - B_1 + 2z_2)^{-1})_{\{\{1\}, \{2\}\}}, (-(B_2^2)(B_2^4 z_1 - 2B_2^3 z_1 + 2B_2^2 z_1 - 2B_2 z_1 - B_2^3 + B_2^2 - B_2 + \\ & z_1)^{-1})_{\{\{2\}, \{1\}\}}) \end{aligned}$
L_{7a4}	$\begin{aligned} & ((-(B_1)(2B_1^2 z_2 - 4B_1 z_2 - B_1 + 2z_2)^{-1})_{\{\{1\}, \{2\}\}}, (-(B_2)(2B_2^2 z_1 - 4B_2 z_1 - B_2 + 2z_1)^{-1})_{\{\{2\}, \{1\}\}}) \end{aligned}$
L_{7a5}	$\begin{aligned} & ((\exp((a_1(z_2 - B_1 z_2))(B_1)^{-1} - b_1 w_2 + (\exp(-(z_2)(B_1)^{-1})(x_1 y_1 \exp((z_2)(B_1)^{-1}) - x_1 y_1 \exp(z_2))) (b_1)^{-1} - \\ & (B_1 \exp((3z_2)(2B_1)^{-1} + (z_2)(2)^{-1}))(B_1^2(-\exp(2z_2)) + B_1^2 \exp((z_2)(B_1)^{-1} + z_2) + B_1 \exp(2z_2) + B_1 \exp((2z_2)(B_1)^{-1}) - \\ & 3B_1 \exp((z_2)(B_1)^{-1} + z_2) - \exp((2z_2)(B_1)^{-1}) + \exp((z_2)(B_1)^{-1} + z_2))^{-1})_{\{\{1\}, \{2\}\}}, (\exp((a_2(z_1 - B_2 z_1))(B_2)^{-1} - \\ & b_2 w_1 + (\exp(-(z_1)(B_2)^{-1})(x_2 y_2 \exp((z_1)(B_2)^{-1}) - x_2 y_2 \exp(z_1))) (b_2)^{-1} - (B_2 \exp((3z_1)(2B_2)^{-1} + \\ & (z_1)(2)^{-1}))(B_2^2(-\exp(2z_1)) + B_2^2 \exp((z_1)(B_2)^{-1} + z_1) + B_2 \exp(2z_1) + B_2 \exp((2z_1)(B_2)^{-1}) - 3B_2 \exp((z_1)(B_2)^{-1} + \\ & z_1) - \exp((2z_1)(B_2)^{-1}) + \exp((z_1)(B_2)^{-1} + z_1))^{-1})_{\{\{2\}, \{1\}\}}) \end{aligned}$
Continued on next page	

Table B.1– continued from previous page

Link	Partial trace Z^{tr} of link
L_{7a6}	$((\exp(a_1(z_2 - B_1 z_2) + b_1 w_2 + (\exp(-z_2)(x_1 y_1 \exp(z_2) - x_1 y_1 \exp(B_1 z_2))))(b_1)^{-1})(B_1 \exp((B_1 z_2)(2)^{-1} + (3z_2)(2)^{-1}))(B_1^2(-\exp(2B_1 z_2)) + B_1^2 \exp(B_1 z_2 + z_2) + B_1 \exp(2z_2) + B_1 \exp(2B_1 z_2) - B_1 \exp(B_1 z_2 + z_2) + \exp(B_1 z_2 + z_2) - \exp(2z_2))^{-1})_{(\{1\}, \{2\})}, (\exp(a_2(z_1 - B_2 z_1) + b_2 w_1 + (\exp(-z_1)(x_2 y_2 \exp(z_1) - x_2 y_2 \exp(B_2 z_1))))(b_2)^{-1})(B_2 \exp((B_2 z_1)(2)^{-1} + (3z_1)(2)^{-1}))(B_2^2(-\exp(2B_2 z_1)) + B_2^2 \exp(B_2 z_1 + z_1) + B_2 \exp(2z_1) + B_2 \exp(2B_2 z_1) - B_2 \exp(B_2 z_1 + z_1) + \exp(B_2 z_1 + z_1) - \exp(2z_1))^{-1})_{(\{2\}, \{1\})})$
L_{7n1}	$((\exp((a_1(2z_2 - 2B_1^2 z_2))(B_1^2)^{-1} - 2b_1 w_2 + (\exp(-(2z_2)(B_1^2)^{-1})(x_1 y_1 \exp((2z_2)(B_1^2)^{-1}) - x_1 y_1 \exp(2z_2))))(b_1)^{-1})(B_1 \exp((5z_2)(2B_1^2)^{-1} + (z_2)(2)^{-1}) + \exp((5z_2)(2B_1^2)^{-1} + (z_2)(2)^{-1}))(B_1 \exp((3z_2)(B_1^2)^{-1}) + \exp(3z_2))^{-1})_{(\{1\}, \{2\})}, (\exp((a_2(2z_1 - 2B_2^2 z_1))(B_2^2)^{-1} - 2b_2 w_1 + (\exp(-(2z_1)(B_2^2)^{-1})(x_2 y_2 \exp((2z_1)(B_2^2)^{-1}) - x_2 y_2 \exp(2z_1))))(b_2)^{-1})(B_2^2 \exp((3z_1)(2B_2^2)^{-1}) + B_2 \exp((3z_1)(2B_2^2)^{-1}))(B_2^3 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}) + \exp((3z_1)(2)^{-1}))^{-1})_{(\{2\}, \{1\})})$
L_{7n2}	$((\exp((B_1)(B_1^2 z_2 - 2B_1 z_2 + B_1 + z_2)^{-1})_{(\{1\}, \{2\})}, (\exp((B_2)(B_2^2 z_1 - 2B_2 z_1 + B_2^2 - B_2 + z_1 + 1)^{-1})_{(\{2\}, \{1\})})$
L_{8a1}	$((\exp((B_1)(B_1^2 z_2 - 2B_1 z_2 + B_1 + z_2)^{-1})_{(\{1\}, \{2\})}, (\exp(-(B_2^2)(B_2^4 z_1 - 5B_2^3 z_1 + 8B_2^2 z_1 - 5B_2 z_1 - B_2^2 + z_1)^{-1})_{(\{2\}, \{1\})})$
L_{8a2}	$((\exp((1)_{(\{1\}, \{2\})}, (\exp((B_2^2)(B_2^4 z_1 - 4B_2^3 z_1 + 6B_2^2 z_1 - 4B_2 z_1 - B_2^3 + 3B_2^2 - B_2 + z_1)^{-1})_{(\{2\}, \{1\})})$

Continued on next page

Table B.1– continued from previous page

Link	Partial trace Z^{tr} of link
L_{8a3}	$ \begin{aligned} & ((\exp((a_1(2z_2 - 2B_1^2z_2))(B_1^2)^{-1} - 2b_1w_2 + (\exp(-(2z_2)(B_1^2)^{-1})(x_1y_1 \exp((2z_2)(B_1^2)^{-1}) - \\ & x_1y_1 \exp(2z_2))) (b_1)^{-1})(B_1(-\exp((5z_2)(2B_1^2)^{-1} + (z_2)(2)^{-1})) - \exp((5z_2)(2B_1^2)^{-1} + (z_2)(2)^{-1}))(B_1 \exp(3z_2) - \\ & 4B_1 \exp((z_2)(B_1^2)^{-1} + 2z_2) + 2B_1 \exp((2z_2)(B_1^2)^{-1} + z_2) + \exp((3z_2)(B_1^2)^{-1} + 2 \exp((z_2)(B_1^2)^{-1} + \\ & 2z_2) - 4 \exp((2z_2)(B_1^2)^{-1} + z_2))^{-1})_{(\{1\}, \{2\})}, (\exp((a_2(2z_1 - 2B_2^2z_1))(B_2^2)^{-1} - 2b_2w_1 + \\ & (\exp(-(2z_1)(B_2^2)^{-1})(x_2y_2 \exp((2z_1)(B_2^2)^{-1}) - x_2y_2 \exp(2z_1))) (b_2)^{-1})(B_2^2(-\exp((3z_1)(2B_2^2)^{-1})) - \\ & B_2 \exp((3z_1)(2B_2^2)^{-1}))(B_2^3 \exp((3z_1)(2)^{-1}) - 4B_2^2 \exp((3z_1)(2)^{-1}) + 2B_2^2 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}) + \\ & 2B_2 \exp((3z_1)(2)^{-1}) - 4B_2 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}) + \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}))^{-1})_{(\{2\}, \{1\})} \end{aligned} $
L_{8a4}	$ ((1)_{(\{1\}, \{2\})}, (- (B_2^2)(B_2^4z_1 - 4B_2^3z_1 + 6B_2^2z_1 - 4B_2z_1 - B_2^3 + B_2^2 - B_2 + z_1)^{-1})_{(\{2\}, \{1\})}) $
L_{8a5}	$ \begin{aligned} & ((\exp((a_1(2z_2 - 2B_1^2z_2))(B_1^2)^{-1} - 2b_1w_2 + (\exp(-(2z_2)(B_1^2)^{-1})(x_1y_1 \exp((2z_2)(B_1^2)^{-1}) - \\ & x_1y_1 \exp(2z_2))) (b_1)^{-1})(B_1(-\exp((5z_2)(2B_1^2)^{-1} + (z_2)(2)^{-1})) - \exp((5z_2)(2B_1^2)^{-1} + (z_2)(2)^{-1}))(-2B_1 \exp(3z_2) + \\ & B_1 \exp((3z_2)(B_1^2)^{-1} + 2B_1 \exp((z_2)(B_1^2)^{-1} + 2z_2) - 2B_1 \exp((2z_2)(B_1^2)^{-1} + z_2) - 2 \exp((3z_2)(B_1^2)^{-1} - \\ & 2 \exp((z_2)(B_1^2)^{-1} + 2z_2) + 2 \exp((2z_2)(B_1^2)^{-1} + z_2) + \exp(3z_2))^{-1})_{(\{1\}, \{2\})}, (\exp((a_2(2z_1 - 2B_2^2z_1))(B_2^2)^{-1} - \\ & 2b_2w_1 + (\exp(-(2z_1)(B_2^2)^{-1})(x_2y_2 \exp((2z_1)(B_2^2)^{-1}) - x_2y_2 \exp(2z_1))) (b_2)^{-1})(B_2^2(-\exp((3z_1)(2B_2^2)^{-1})) - \\ & B_2 \exp((3z_1)(2B_2^2)^{-1}))(-2B_2^3 \exp((3z_1)(2)^{-1}) + B_2^3 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}) + 2B_2^2 \exp((3z_1)(2)^{-1} - \\ & 2B_2^2 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}) - 2B_2 \exp((3z_1)(2)^{-1}) + 2B_2 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}) - 2 \exp((z_1)(B_2^2)^{-1} + \\ & (z_1)(2)^{-1}) + \exp((3z_1)(2)^{-1}))^{-1})_{(\{2\}, \{1\})} \end{aligned} $
Continued on next page	

Table B.1– continued from previous page

Link	Partial trace Z^{tr} of link
L_{8a6}	$ \begin{aligned} & ((\exp((a_1(2z_2 - 2B_1^2 z_2))(B_1^2)^{-1} - 2b_1 w_2 + (\exp(-(2z_2)(B_1^2)^{-1})(x_1 y_1 \exp((2z_2)(B_1^2)^{-1}) - \\ & x_1 y_1 \exp(2z_2))) (b_1)^{-1})(B_1(-\exp((3z_2)(2B_1^2)^{-1})) - \exp((3z_2)(2B_1^2)^{-1}))(-3B_1 \exp((3z_2)(2)^{-1}) + \\ & 2B_1 \exp((z_2)(B_1^2)^{-1} + (z_2)(2)^{-1}) - 3 \exp((z_2)(B_1^2)^{-1} + (z_2)(2)^{-1}) + 2 \exp((3z_2)(2)^{-1}))^{-1})_{\{\{1\},\{2\}\}}, (\exp((a_2(2z_1 - \\ & 2B_2^2 z_1))(B_2^2)^{-1} - 2b_2 w_1 + (\exp(-(2z_1)(B_2^2)^{-1})(x_2 y_2 \exp((2z_1)(B_2^2)^{-1}) - \\ & x_2 y_2 \exp(2z_1))) (b_2)^{-1})(B_2(-\exp((3z_1)(2B_2^2)^{-1})) - \exp((3z_1)(2B_2^2)^{-1}))(-3B_2 \exp((3z_1)(2)^{-1}) + \\ & 2B_2 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}) - 3 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}) + 2 \exp((3z_1)(2)^{-1}))^{-1})_{\{\{2\},\{1\}\}} \end{aligned} $
L_{8a7}	$ \begin{aligned} & ((\exp((a_1(2z_2 - 2B_1^2 z_2))(B_1^2)^{-1} - 2b_1 w_2 + (\exp(-(2z_2)(B_1^2)^{-1})(x_1 y_1 \exp((2z_2)(B_1^2)^{-1}) - \\ & x_1 y_1 \exp(2z_2))) (b_1)^{-1})(B_1 \exp((5z_2)(2B_1^2)^{-1} + (z_2)(2)^{-1}) + \exp((5z_2)(2B_1^2)^{-1} + (z_2)(2)^{-1}))(B_1 \exp(3z_2) - \\ & 4B_1 \exp((z_2)(B_1^2)^{-1} + 2z_2) + 4B_1 \exp((2z_2)(B_1^2)^{-1} + z_2) + \exp((3z_2)(B_1^2)^{-1} + 4 \exp((z_2)(B_1^2)^{-1} + \\ & 2z_2) - 4 \exp((2z_2)(B_1^2)^{-1} + z_2))^{-1})_{\{\{1\},\{2\}\}}, (\exp((a_2(2z_1 - 2B_2^2 z_1))(B_2^2)^{-1} - 2b_2 w_1 + \\ & (\exp(-(2z_1)(B_2^2)^{-1})(x_2 y_2 \exp((2z_1)(B_2^2)^{-1}) - x_2 y_2 \exp(2z_1))) (b_2)^{-1})(B_2^2 \exp((3z_1)(2B_2^2)^{-1}) + \\ & B_2 \exp((3z_1)(2B_2^2)^{-1}))(B_2^3 \exp((3z_1)(2)^{-1}) - 4B_2^2 \exp((3z_1)(2)^{-1}) + 4B_2^2 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}) + \\ & 4B_2 \exp((3z_1)(2)^{-1}) - 4B_2 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}) + \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}))^{-1})_{\{\{2\},\{1\}\}} \end{aligned} $
Continued on next page	

Table B.1– continued from previous page

Link	Partial trace Z^{tr} of link
L_{8a8}	$ \begin{aligned} & ((\exp((a_1(z_2 - B_1 z_2))(B_1)^{-1} - b_1 w_2 + (\exp(-(z_2)(B_1)^{-1})(x_1 y_1 \exp((z_2)(B_1)^{-1}) - x_1 y_1 \exp(z_2)))(b_1)^{-1} - \\ & (B_1 \exp((3z_2)(2B_1)^{-1} + (z_2)(2)^{-1}))(B_1^2 \exp(2z_2) + B_1^2 \exp((2z_2)(B_1)^{-1}) - 2B_1^2 \exp((z_2)(B_1)^{-1} + z_2) - \\ & 2B_1 \exp(2z_2) - 2B_1 \exp((2z_2)(B_1)^{-1}) + 3B_1 \exp((z_2)(B_1)^{-1} + z_2) + \exp((2z_2)(B_1)^{-1}) - 2 \exp((z_2)(B_1)^{-1} + \\ & z_2) + \exp(2z_2))^{-1})_{(\{1\}, \{2\})}, (\exp((a_2(z_1 - B_2 z_1))(B_2)^{-1} - b_2 w_1 + (\exp(-(z_1)(B_2)^{-1})(x_2 y_2 \exp((z_1)(B_2)^{-1}) - \\ & x_2 y_2 \exp(z_1)))(b_2)^{-1} - (B_2 \exp((3z_1)(2B_2)^{-1} + (z_1)(2)^{-1}))(B_2^2 \exp(2z_1) + B_2^2 \exp((2z_1)(B_2)^{-1}) - \\ & 2B_2^2 \exp((z_1)(B_2)^{-1} + z_1) - 2B_2 \exp(2z_1) - 2B_2 \exp((2z_1)(B_2)^{-1}) + 3B_2 \exp((z_1)(B_2)^{-1} + z_1) + \exp((2z_1)(B_2)^{-1}) - \\ & 2 \exp((z_1)(B_2)^{-1} + z_1) + \exp(2z_1))^{-1})_{(\{2\}, \{1\})} \end{aligned} $
L_{8a9}	$ \begin{aligned} & ((\exp((a_1(z_2 - B_1 z_2))(B_1)^{-1} - b_1 w_2 + (\exp(-(z_2)(B_1)^{-1})(x_1 y_1 \exp((z_2)(B_1)^{-1}) - \\ & x_1 y_1 \exp(z_2)))(b_1)^{-1} (B_1 \exp((3z_2)(2B_1)^{-1} + (z_2)(2)^{-1}))(B_1^2 \exp(2z_2) + B_1^2 \exp((2z_2)(B_1)^{-1}) - \\ & 2B_1^2 \exp((z_2)(B_1)^{-1} + z_2) - 2B_1 \exp(2z_2) - 2B_1 \exp((2z_2)(B_1)^{-1}) + 5B_1 \exp((z_2)(B_1)^{-1} + z_2) + \\ & \exp((2z_2)(B_1)^{-1}) - 2 \exp((z_2)(B_1)^{-1} + z_2) + \exp(2z_2))^{-1})_{(\{1\}, \{2\})}, (\exp((a_2(z_1 - B_2 z_1))(B_2)^{-1} - b_2 w_1 + \\ & (\exp(-(z_1)(B_2)^{-1})(x_2 y_2 \exp((z_1)(B_2)^{-1}) - x_2 y_2 \exp(z_1)))(b_2)^{-1} (B_2 \exp((3z_1)(2B_2)^{-1} + (z_1)(2)^{-1}))(B_2^2 \exp(2z_1) + \\ & B_2^2 \exp((2z_1)(B_2)^{-1}) - 2B_2^2 \exp((z_1)(B_2)^{-1} + z_1) - 2B_2 \exp(2z_1) - 2B_2 \exp((2z_1)(B_2)^{-1}) + 5B_2 \exp((z_1)(B_2)^{-1} + \\ & z_1) + \exp((2z_1)(B_2)^{-1}) - 2 \exp((z_1)(B_2)^{-1} + z_1) + \exp(2z_1))^{-1})_{(\{2\}, \{1\})} \end{aligned} $

Continued on next page

Table B.1– continued from previous page

Link	Partial trace Z^{tr} of link
L_{8a10}	$ \begin{aligned} & ((\exp((a_1(3z_2 - 3B_1^3 z_2))(B_1^3)^{-1} - 3b_1 w_2 + (\exp(-(3z_2)(B_1^3)^{-1})(x_1 y_1 \exp((3z_2)(B_1^3)^{-1}) - \\ & x_1 y_1 \exp(3z_2)))(b_1)^{-1})(B_1^2 \exp((5z_2)(2B_1^3)^{-1}) + B_1 \exp((5z_2)(2B_1^3)^{-1}) + \\ & \exp((5z_2)(2B_1^3)^{-1}))(B_1^2(-\exp((5z_2)(2)^{-1})) + 2B_1^2 \exp((z_2)(B_1^3)^{-1} + (3z_2)(2)^{-1}) + 2B_1 \exp((5z_2)(2)^{-1}) - \\ & 3B_1 \exp((z_2)(B_1^3)^{-1} + (3z_2)(2)^{-1}) + 2B_1 \exp((2z_2)(B_1^3)^{-1} + (z_2)(2)^{-1}) + 2 \exp((z_2)(B_1^3)^{-1} + \\ & (3z_2)(2)^{-1}) - \exp((2z_2)(B_1^3)^{-1} + (z_2)(2)^{-1}))^{-1})_{(\{1\}, \{2\})}, (\exp((a_2(3z_1 - 3B_2^3 z_1))(B_2^3)^{-1} - 3b_2 w_1 + \\ & (\exp(-(3z_1)(B_2^3)^{-1})(x_2 y_2 \exp((3z_1)(B_2^3)^{-1}) - x_2 y_2 \exp(3z_1)))(b_2)^{-1})(B_2^2 \exp((5z_1)(2B_2^3)^{-1}) + \\ & B_2 \exp((5z_1)(2B_2^3)^{-1}) + \exp((5z_1)(2B_2^3)^{-1}))(B_2^2(-\exp((5z_1)(2)^{-1})) + 2B_2^2 \exp((z_1)(B_2^3)^{-1} + (3z_1)(2)^{-1}) + \\ & 2B_2 \exp((5z_1)(2)^{-1}) - 3B_2 \exp((z_1)(B_2^3)^{-1} + (3z_1)(2)^{-1}) + 2B_2 \exp((2z_1)(B_2^3)^{-1} + (z_1)(2)^{-1}) + 2 \exp((z_1)(B_2^3)^{-1} + \\ & (3z_1)(2)^{-1}) - \exp((2z_1)(B_2^3)^{-1} + (z_1)(2)^{-1}))^{-1})_{(\{2\}, \{1\})} \end{aligned} $
L_{8a11}	$ \begin{aligned} & ((\exp((a_1(3z_2 - 3B_1^3 z_2))(B_1^3)^{-1} - 3b_1 w_2 + (\exp(-(3z_2)(B_1^3)^{-1})(x_1 y_1 \exp((3z_2)(B_1^3)^{-1}) - \\ & x_1 y_1 \exp(3z_2)))(b_1)^{-1})(B_1^2 \exp((5z_2)(2B_1^3)^{-1}) + B_1 \exp((5z_2)(2B_1^3)^{-1}) + \\ & \exp((5z_2)(2B_1^3)^{-1}))(B_1^2(-\exp((z_2)(B_1^3)^{-1} + (3z_2)(2)^{-1})) + 2B_1^2 \exp((2z_2)(B_1^3)^{-1} + (z_2)(2)^{-1}) - \\ & B_1 \exp((5z_2)(2)^{-1}) + 3B_1 \exp((z_2)(B_1^3)^{-1} + (3z_2)(2)^{-1}) - B_1 \exp((2z_2)(B_1^3)^{-1} + (z_2)(2)^{-1}) - \\ & \exp((z_2)(B_1^3)^{-1} + (3z_2)(2)^{-1}) + 2 \exp((5z_2)(2)^{-1}))^{-1})_{(\{1\}, \{2\})}, (\exp((a_2(3z_1 - 3B_2^3 z_1))(B_2^3)^{-1} - \\ & 3b_2 w_1 + (\exp(-(3z_1)(B_2^3)^{-1})(x_2 y_2 \exp((3z_1)(B_2^3)^{-1}) - x_2 y_2 \exp(3z_1)))(b_2)^{-1})(B_2^2 \exp((5z_1)(2B_2^3)^{-1}) + \\ & B_2 \exp((5z_1)(2B_2^3)^{-1}) + \exp((5z_1)(2B_2^3)^{-1}))(B_2^2(-\exp((z_1)(B_2^3)^{-1} + (3z_1)(2)^{-1})) + 2B_2^2 \exp((2z_1)(B_2^3)^{-1} + \\ & (z_1)(2)^{-1}) - B_2 \exp((5z_1)(2)^{-1}) + 3B_2 \exp((z_1)(B_2^3)^{-1} + (3z_1)(2)^{-1}) - B_2 \exp((2z_1)(B_2^3)^{-1} + (z_1)(2)^{-1}) - \\ & \exp((z_1)(B_2^3)^{-1} + (3z_1)(2)^{-1}) + 2 \exp((5z_1)(2)^{-1}))^{-1})_{(\{2\}, \{1\})} \end{aligned} $
Continued on next page	

Table B.1– continued from previous page

Link	Partial trace Z^{tr} of link
L_{8a12}	$ \begin{aligned} & ((\exp((a_1(4z_2 - 4B_1^4 z_2))(B_1^4)^{-1} - 4b_1 w_2 + (\exp(-(4z_2)(B_1^4)^{-1})(x_1 y_1 \exp((4z_2)(B_1^4)^{-1}) - \\ & x_1 y_1 \exp(4z_2))) (b_1)^{-1} (B_1^3 \exp((7z_2)(2B_1^4)^{-1}) + B_1^2 \exp((7z_2)(2B_1^4)^{-1}) + B_1 \exp((7z_2)(2B_1^4)^{-1}) + \\ & \exp((7z_2)(2B_1^4)^{-1})) (B_1^3 \exp((2z_2)(B_1^4)^{-1} + (3z_2)(2)^{-1}) + B_1^2 \exp((z_2)(B_1^4)^{-1} + (5z_2)(2)^{-1}) - \\ & B_1^2 \exp((2z_2)(B_1^4)^{-1} + (3z_2)(2)^{-1}) + B_1^2 \exp((3z_2)(B_1^4)^{-1} + (z_2)(2)^{-1}) + B_1 \exp((7z_2)(2)^{-1}) - \\ & B_1 \exp((z_2)(B_1^4)^{-1} + (5z_2)(2)^{-1}) + B_1 \exp((2z_2)(B_1^4)^{-1} + (3z_2)(2)^{-1}) + \exp((z_2)(B_1^4)^{-1} + \\ & (5z_2)(2)^{-1}))^{-1}_{(\{1\}, \{2\})}, (\exp((a_2(4z_1 - 4B_2^4 z_1))(B_2^4)^{-1} - 4b_2 w_1 + (\exp(-(4z_1)(B_2^4)^{-1})(x_2 y_2 \exp((4z_1)(B_2^4)^{-1}) - \\ & x_2 y_2 \exp(4z_1))) (b_2)^{-1} (B_2^3 \exp((7z_1)(2B_2^4)^{-1}) + B_2^2 \exp((7z_1)(2B_2^4)^{-1}) + B_2 \exp((7z_1)(2B_2^4)^{-1}) + \\ & \exp((7z_1)(2B_2^4)^{-1})) (B_2^3 \exp((2z_1)(B_2^4)^{-1} + (3z_1)(2)^{-1}) + B_2^2 \exp((z_1)(B_2^4)^{-1} + (5z_1)(2)^{-1}) - \\ & B_2^2 \exp((2z_1)(B_2^4)^{-1} + (3z_1)(2)^{-1}) + B_2^2 \exp((3z_1)(B_2^4)^{-1} + (z_1)(2)^{-1}) + B_2 \exp((7z_1)(2)^{-1}) - B_2 \exp((z_1)(B_2^4)^{-1} + \\ & (5z_1)(2)^{-1}) + B_2 \exp((2z_1)(B_2^4)^{-1} + (3z_1)(2)^{-1}) + \exp((z_1)(B_2^4)^{-1} + (5z_1)(2)^{-1}))^{-1}_{(\{2\}, \{1\})} \end{aligned} $
Continued on next page	

Table B.1– continued from previous page

Link	Partial trace Z^{tr} of link
L_{8a13}	$ \begin{aligned} & ((\exp((a_1(4z_2 - 4B_1^4 z_2))(B_1^4)^{-1} - 4b_1 w_2 + (\exp(-(4z_2)(B_1^4)^{-1})(x_1 y_1 \exp((4z_2)(B_1^4)^{-1}) - \\ & x_1 y_1 \exp(4z_2))) (b_1)^{-1})(B_1^3 \exp((7z_2)(2B_1^4)^{-1}) + B_1^2 \exp((7z_2)(2B_1^4)^{-1}) + B_1 \exp((7z_2)(2B_1^4)^{-1}) + \\ & \exp((7z_2)(2B_1^4)^{-1}))(B_1^3 \exp((z_2)(B_1^4)^{-1} + (5z_2)(2)^{-1}) + B_1^2 \exp((7z_2)(2)^{-1}) - 2B_1^2 \exp((z_2)(B_1^4)^{-1} + (5z_2)(2)^{-1}) + \\ & 2B_1^2 \exp((2z_2)(B_1^4)^{-1} + (3z_2)(2)^{-1}) + 2B_1 \exp((z_2)(B_1^4)^{-1} + (5z_2)(2)^{-1}) - 2B_1 \exp((2z_2)(B_1^4)^{-1} + (3z_2)(2)^{-1}) + \\ & B_1 \exp((3z_2)(B_1^4)^{-1} + (z_2)(2)^{-1}) + \exp((2z_2)(B_1^4)^{-1} + (3z_2)(2)^{-1}))^{-1}_{(\{1\}, \{2\})}, (\exp((a_2(4z_1 - 4B_2^4 z_1))(B_2^4)^{-1} - \\ & 4b_2 w_1 + (\exp(-(4z_1)(B_2^4)^{-1})(x_2 y_2 \exp((4z_1)(B_2^4)^{-1}) - x_2 y_2 \exp(4z_1))) (b_2)^{-1})(B_2^3 \exp((7z_1)(2B_2^4)^{-1}) + \\ & B_2^2 \exp((7z_1)(2B_2^4)^{-1}) + B_2 \exp((7z_1)(2B_2^4)^{-1}) + \exp((7z_1)(2B_2^4)^{-1}))(B_2^3 \exp((z_1)(B_2^4)^{-1} + (5z_1)(2)^{-1}) + \\ & B_2^2 \exp((7z_1)(2)^{-1}) - 2B_2^2 \exp((z_1)(B_2^4)^{-1} + (5z_1)(2)^{-1}) + 2B_2^2 \exp((2z_1)(B_2^4)^{-1} + (3z_1)(2)^{-1}) + \\ & 2B_2 \exp((z_1)(B_2^4)^{-1} + (5z_1)(2)^{-1}) - 2B_2 \exp((2z_1)(B_2^4)^{-1} + (3z_1)(2)^{-1}) + B_2 \exp((3z_1)(B_2^4)^{-1} + (z_1)(2)^{-1}) + \\ & \exp((2z_1)(B_2^4)^{-1} + (3z_1)(2)^{-1}))^{-1}_{(\{2\}, \{1\})}) \end{aligned} $
L_{8a14}	$ \begin{aligned} & ((\exp((a_1(4z_2 - 4B_1^4 z_2))(B_1^4)^{-1} - 4b_1 w_2 + (\exp(-(4z_2)(B_1^4)^{-1})(x_1 y_1 \exp((4z_2)(B_1^4)^{-1}) - \\ & x_1 y_1 \exp(4z_2))) (b_1)^{-1})(B_1^3 \exp((7z_2)(2B_1^4)^{-1}) + B_1^2 \exp((7z_2)(2B_1^4)^{-1}) + B_1 \exp((7z_2)(2B_1^4)^{-1}) + \\ & \exp((7z_2)(2B_1^4)^{-1}))(B_1^3 \exp((3z_2)(B_1^4)^{-1} + (z_2)(2)^{-1}) + B_1^2 \exp((2z_2)(B_1^4)^{-1} + (3z_2)(2)^{-1}) + \\ & B_1 \exp((z_2)(B_1^4)^{-1} + (5z_2)(2)^{-1}) + \exp((7z_2)(2)^{-1}))^{-1}_{(\{1\}, \{2\})}, (\exp((a_2(4z_1 - 4B_2^4 z_1))(B_2^4)^{-1} - \\ & 4b_2 w_1 + (\exp(-(4z_1)(B_2^4)^{-1})(x_2 y_2 \exp((4z_1)(B_2^4)^{-1}) - x_2 y_2 \exp(4z_1))) (b_2)^{-1})(B_2^3 \exp((7z_1)(2B_2^4)^{-1}) + \\ & B_2^2 \exp((7z_1)(2B_2^4)^{-1}) + B_2 \exp((7z_1)(2B_2^4)^{-1}) + \exp((7z_1)(2B_2^4)^{-1}))(B_2^3 \exp((3z_1)(B_2^4)^{-1} + (z_1)(2)^{-1}) + \\ & B_2^2 \exp((2z_1)(B_2^4)^{-1} + (3z_1)(2)^{-1}) + B_2 \exp((z_1)(B_2^4)^{-1} + (5z_1)(2)^{-1}) + \exp((7z_1)(2)^{-1}))^{-1}_{(\{2\}, \{1\})}) \end{aligned} $
Continued on next page	

Table B.1– continued from previous page

Link	Partial trace Z^{tr} of link
L_{8n1}	$ \begin{aligned} & ((\exp((a_1(2z_2 - 2B_1^2z_2))(B_1^2)^{-1} - 2b_1w_2 + (\exp(-(2z_2)(B_1^2)^{-1})(x_1y_1 \exp((2z_2)(B_1^2)^{-1}) - \\ & x_1y_1 \exp(2z_2))) (b_1)^{-1})(B_1(-\exp((5z_2)(2B_1^2)^{-1} + (z_2)(2)^{-1})) - \exp((5z_2)(2B_1^2)^{-1} + \\ & (z_2)(2)^{-1}))(B_1 \exp((3z_2)(B_1^2)^{-1}) - 2B_1 \exp((2z_2)(B_1^2)^{-1} + z_2) - 2 \exp((z_2)(B_1^2)^{-1} + 2z_2) + \\ & \exp(3z_2))^{-1})_{(\{1\},\{2\})}, (\exp((a_2(2z_1 - 2B_2^2z_1))(B_2^2)^{-1} - 2b_2w_1 + (\exp(-(2z_1)(B_2^2)^{-1})(x_2y_2 \exp((2z_1)(B_2^2)^{-1}) - \\ & x_2y_2 \exp(2z_1))) (b_2)^{-1})(B_2^2(-\exp((3z_1)(2B_2^2)^{-1})) - B_2 \exp((3z_1)(2B_2^2)^{-1}))(B_2^3 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}) - \\ & 2B_2^2 \exp((z_1)(B_2^2)^{-1} + (z_1)(2)^{-1}) - 2B_2 \exp((3z_1)(2)^{-1}) + \exp((3z_1)(2)^{-1}))^{-1})_{(\{2\},\{1\})} \end{aligned} $
L_{8n2}	$ ((-(B_1)(B_1^2z_2 - 2B_1z_2 - B_1 + z_2)^{-1})_{(\{1\},\{2\})}, (-(B_2)(B_2^2z_1 - 2B_2z_1 + B_2^2 - 3B_2 + z_1 + 1)^{-1})_{(\{2\},\{1\})}) $

BIBLIOGRAPHY

- [BNMea] Dror Bar-Natan, Scott Morrison, and et al., The Knot Atlas.
- [BNS] Dror Bar-Natan and Sam Selmani, Meta-monoids, meta-bicrossed products, and the Alexander polynomial, no. 10, 1350058, Publisher: World Scientific Publishing Co.
- [BNvdVa] Dror Bar-Natan and Roland van der Veen, A perturbed-Alexander invariant, no. arXiv:2206.12298.
- [BNvdVb] ———, Perturbed Gaußian generating functions for universal knot invariants, no. arXiv:2109.02057.
- [ES] Pavel I. Etingof and Olivier Schiffmann, Lectures on quantum groups, International Press.
- [Fox] R. H. Fox, Some problems in knot theory.
- [Hal] Brian C. Hall, Quantum theory for mathematicians, Graduate Texts in Mathematics, vol. 267, Springer.
- [Kau] Louis H. Kauffman, Rotational virtual knots and quantum link invariants.
- [Lic] W. B. Raymond Lickorish, An introduction to knot theory, Springer Science & Business Media.
- [Maj] Shahn Majid, A quantum groups primer, London Mathematical Society Lecture Note Series, Cambridge University Press.
- [Pen] Roger Penrose, Spinors and space-time. volume 1, two-spinor calculus and relativistic fields, Cambridge monographs on mathematical physics, University Press.
- [RT] N. Reshetikhin and V. G. Turaev, Invariants of 3-manifolds via link polynomials and quantum groups, no. 1, 547–597.
- [Wey] H. Weyl, Quantenmechanik und gruppentheorie, no. 1, 1–46.

COLOPHON

This thesis was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić.

The style was inspired by Robert Bringhurst's seminal book on typography [The Elements of Typographic Style](#).

Figures were created with Asymptote: The Vector Graphics Language (<https://asymptote.sourceforge.io/>).

This manuscript and its amendments are hosted on GitHub at <https://github.com/phro/thesis>.

Version 1.0.0 (electronic)