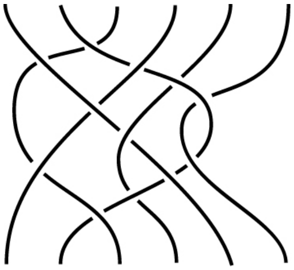


An Introduction to the Brauer and BMW Algebras[1]

(with an emphasis on the canonical basis and cells of BMW_3)

Geordie Williamson

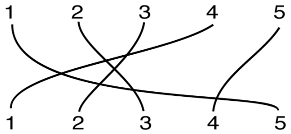
[1] This document was produced as part of a vacation scholarship at the University of New South Wales. Thanks to the University of New South Wales and especially to Jie Du who supervised the project. Thanks also to Steve Ward for proof reading.

Contents

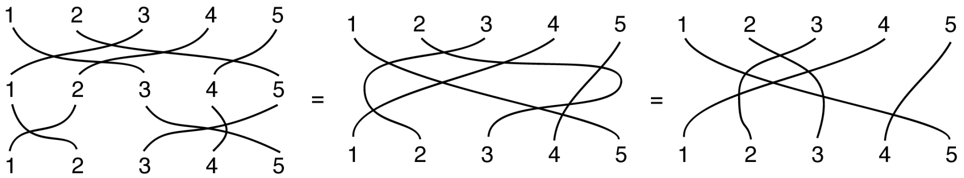
1 The Brauer and BMW Algebras

1.1 The Brauer Monoid

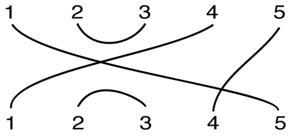
One common way to get an intuitive feel for a permutation $\sigma \in \mathrm{Sym}_n$ is to draw it as a diagram in which two rows of n numbers are linked together with lines from the top to bottom indicating the action of the permutations on the numbered positions. For example, $(154)(23) \in \mathrm{Sym}_5$ would be drawn as:



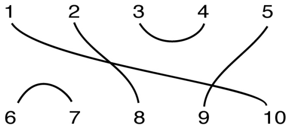
One of the advantages of this representation is that multiplication of permutations becomes a simple concatenation of diagrams. That is, the second permutation is placed on top of the first and the middle set of two rows of n dots are joined. For example the following diagram illustrates the process of multiplying $(12)(35)$ and $(13)(254)$ to get $(154)(23)$:



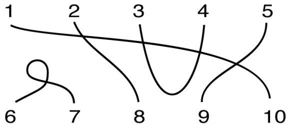
One assumption that is made when drawing these diagrams is that it is not possible to have two dots in the top row linked by a loop. For instance the following would not make sense as a permutation:



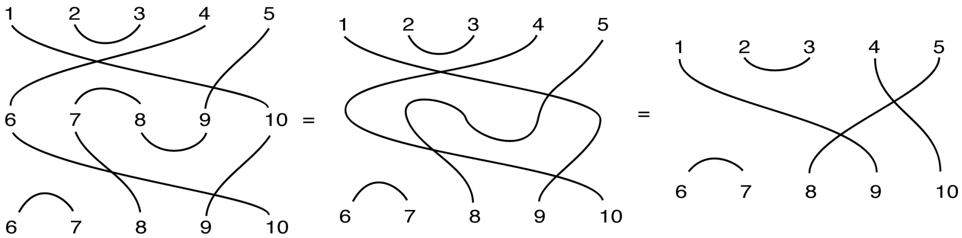
The Brauer monoid is a generalisation of the symmetric group in that it allows pictures such as the one above in which lines can be drawn to link dots on the same line. Being a little more formal we define the Brauer monoid on $2n$ dots, B_n, to be the set of partitions of $[2n] = \{1, 2, \ldots, 2n\}$ into 2-subsets. Given any partition $d = \{[2n] = \lambda_1 \cup \lambda_2 \cup \cdots \cup \lambda_n\}$ we can form a diagram of d by connecting two dots with a line if they belong to the same subset (with the dots labeled as below). If a diagram contains no self-intersection, two lines intersect at most once and at most two lines intersect at any point then the diagram is called a *nice*. For example the partition $d = \{[10] = \{1, 10\} \cup \{2, 8\} \cup \{3, 4\} \cup \{5, 9\} \cup \{6, 7\}\}$ has the following nice diagram:



The following is not a nice diagram of d:

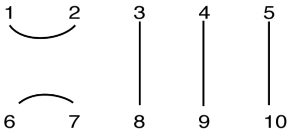


Multiplication is conducted almost identically to the multiplication of permutations except that the first element in the product in B_n is placed on top in the concatenation. For example the following diagram illustrates the multiplication of $\{\{1,10\},\{2,3\},\{4,6\},\{5,9\},\{7,8\}\}$ and $\{\{1,10\},\{2,8\},\{3,4\},\{5,9\},\{6,7\}\}$ to get $\{\{1,9\},\{2,3\},\{4,10\},\{5,8\},\{6,7\}\}$:

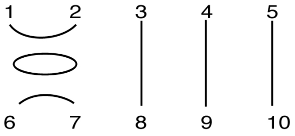


There is one subtlety which is introduced when we define multiplication in the Brauer monoid. It turns out that it is possible to get a number of closed loops when certain members of B_n are multiplied. For example if:

$b =$



then $b^2 =$



So a more accurate description of an element of B_n is as a partition of $[2n]$ into 2-subsets along with an integer $t \geq 0$ specifying a certain number of closed loops. It is often convenient to regard two elements of B_n as equivalent if they differ only in the number of closed loops. We will write $\widetilde{B_n}$ for B_n modulo this equivalence relation.

It turns out that once we have defined multiplication by concatenation the set of Brauer diagrams becomes a monoid (that is there exists an identity element and multiplication is associative). However it will take us a little while to develop the tools that will allow us to prove this.

We first introduce the "joins" notation. If $f \in B_n$ we say that i is joined to j in f and write $i \sim_f j$, if i and j are in the same partition. In terms of pictures i and j are joined if $i = j$ or they are linked with a line. It is immediately obvious that $\sim_f$ is reflexive ($i \sim_f i$ is always true), symmetric ($i \sim_f j$ implies that $j \sim_f i$) and that if $i \sim_f j \sim_f k$ then $i = k$ or $j = k$. From our definition of multiplication of Brauer diagrams it is clear that $i \sim_{fg} j$ if and only if one of the following conditions hold:

(B1) $i, j \leq n$ and $i \sim_f j$.

(B2) $i, j \leq n$ and there exist $k, l \in \{1, 2, \ldots, n\}$ such that $i \sim_f (k+n)$, $k \sim_g l$ and $(l+n) \sim_f j$

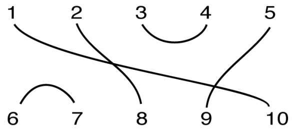
(B3) $i \leq n < j$ and there exists a $k \in \{1, 2, \ldots, n\}$ such that $i \sim_f (k+n)$ and $k \sim_g j$.

(B4) $i, j > n$ and $i \sim_g j$.

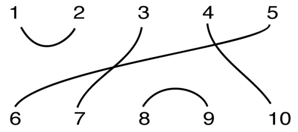
(B5) $i, j > n$ and there exist $k, l \in \{1, 2, \ldots, n\}$ such that $i \sim_g k$, $(k+n) \sim_f (l+n)$ and $l \sim_g j$

To prove that B_n is associative it is then just a matter of enumerating all possible combinations of i and j to show that $i \sim_{(fg)h} j$ implies that $i \sim_{f(gh)} j$. The following lemma saves us a little work when we come to enumerate all possibilities. It concerns the map $\widehat{\ }\colon B_n \to B_n$ which can be seen as flipping the Brauer diagrams so dots on the bottom are taken to the top and vice versa:

$d =$



then $\widehat{d} =$



Note that if we view a picture of a permutation $\sigma \in Sym_n$ as a Brauer diagram then $\widehat{\sigma} = \sigma^{-1}$.

Lemma 1.1. *The map* $\widehat{\ }\colon B_n \to B_n$ *defined by* $i \sim_{\widehat{f}} j$ *if* $(2n-i) \sim_f (2n-j)$ *is a bijection satisfying* $\widehat{\widehat{f}} = f$. *Moverover,* $\widehat{fg} = \widehat{g}\widehat{f}$.

Proof. Let $f \in B_n$ and assume that $i \sim_f j$. Then $(2n-i) \sim_{\widehat{f}} (2n-j)$ so $i \sim_{\widehat{\widehat{f}}} j$. Hence $\widehat{\widehat{f}} = f$ and so $\widehat{\ }$ is a bijection.

Now assume that $i \sim_{\widehat{fg}} j$. Then $(2n-1) \sim_{fg} (2n-j)$. If $(2n-i), (2n-j) \leq n$ there are two possibilities. Either $(2n-i) \sim_f (2n-j)$ or there exists $k, l \in \{1, 2, \ldots, n\}$ such that $(2n-i) \sim_f (k+n)$, $k \sim_g l$ and $(l+n) \sim_f (2n-j)$. If $(2n-i) \sim_f (2n-j)$ then $i \sim_{\widehat{f}} j$ with $i, j > n$ and so $i \sim_{\widehat{g}\widehat{f}} j$. On the other hand if there exists $k, l \in \{1, 2, \ldots, n\}$ such that $(2n-i) \sim_f (k+n)$, $k \sim_g l$ and $(l+n) \sim_f (2n-j)$ then $i \sim_{\widehat{f}} k'$, $(k'+n) \sim_{\widehat{g}(l'+n)}$ and $l' \sim_{\widehat{f}} j$ and so $i \sim_{\widehat{g}\widehat{f}}$.

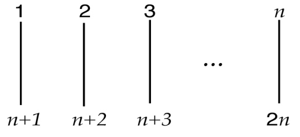
If $(2n-i) \leq n < (2n-j)$ then there exists a $k \in \{1, 2, \ldots, n\}$ such that $(2n-i) \sim_f (k+n)$ and $k \sim_g (2n-j)$. Setting $k' = n-k$ we see that $i \sim_{\widehat{f}} k'$ and $(k'+n) \sim_{\widehat{g}} j$. That is $i \sim_{\widehat{g}\widehat{f}} j$.

The last case, $(2n-i), (2n-j) > n$, is similar to the case $(2n-i), (2n-j) \leq n$. □

We are now ready to prove the theorem:

Theorem 1.2. *The set of Brauer diagrams on* $2n$ *dots,* B_n, *with multiplication defined by concatenation, is a monoid.*

Proof. We first show that B_n has an identity. Consider the picture in which each top dot is joined to the dot immediately below it:



In the joins notation this is the element $k \sim_{id} (k+n)$ for all $j \in \{1, 2, \ldots, n\}$. Let $f \in B_n$ be arbitrary and assume that $j \sim_f k$. If $j < n$ then $i \sim_{f \cdot id} j$ by B1. If $i \leq n < j$ then we can write $i \sim_f (j-n)+n$ and $(j-n) \sim_{id} j$ which implies that $i \sim_{f \cdot id} j$ by B3. Lastly, if $i, j > n$ then we can write $i' + n = i$ and $j' + n = j$. Then we have $j' \sim_{id} (j'+n)$, $i' \sim_{id} (i'+n)$ and $(i'+n) \sim_f (j'+n$ so $i \sim_{f \cdot id} j$ by B5. So id is a right identity. A very similar argument shows that id is a left identity.

Associativity is a little more tricky to show. Assume $i \sim_{(fg)h} j$. Now assume that $i, j \leq n$. There are two possibilities:

(1) If $i \sim_{fg} j$ then either $i \sim_f j$ or there exists $k, l \in \{1, 2, \ldots, n\}$ such that $i \sim_f (k+n)$, $k \sim_g l$ and $(l+n) \sim_f j$. If $i \sim_f j$ then $i \sim_{f(gh)} j$ by B1. On the other hand if there exists $k, l \in \{1, 2, \ldots, n\}$ such that $i \sim_f (k+n)$, $k \sim_g l$ and $(l+n) \sim_f j$ then $k \sim_{gh} l$ since $k, l \leq n$ by B1. Furthermore since $i \sim_f (k+n)$, $k \sim_{gh} l$ and $(l+n) \sim_f j$ implies that $i \sim_{f(gh)} j$ by B2.

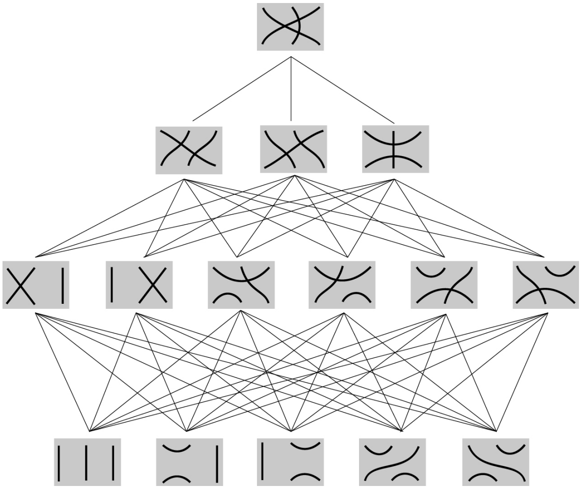
(2) If there exists $k, l \in \{1, 2, \ldots, n\}$ with $i \sim_{fg} (k+n)$, $k \sim_g l$ and $(l+n) \sim_f j$ then the two conditions $i \sim_{fg} (k+n)$ and $j \sim_{fg} (l+n)$ imply that there exists $p, q \in \{1, 2, \ldots, n\}$ such that $i \sim_f (p+n)$, $p \sim_g (k+n)$, $j \sim_f (q+n)$ and $q \sim_g (l+n)$ by B3. Now $p \sim_g (k+n)$, $k \sim_h l$ and $(l+n) \sim_g q$ implies that $p \sim_{gh} q$. Furthermore, $i \sim_f (p+n)$, $p \sim_{gh} q$ and $(q+n) \sim_f j$ implies that $i \sim_{f(gh)} j$.

Now assume that $i \leq n < j$. Then B3 tells us that there exists a $p \in \{1, 2, \ldots, n\}$ such that $i \sim_{fg} (p+n)$ and $p \sim_h j$. We can again apply B3 to $i \sim_{fg} (p+n)$ to conclude that there exists a $q \in \{1, 2, \ldots, n\}$ such that $i \sim_f (q+n)$ and $q \sim_g (p+n)$. But $q \sim_g (p+n)$ and $p \sim_h j$ implies that $q \sim_{gh} j$ by B3. Lastly $i \sim_f (q+n)$ and $q \sim_{gh} j$ implies that $i \sim_{f(gh)} j$.

We can now use Lemma 1.1 to show that $i \sim_{(fg)h} j$ implies $i \sim_{f(gh)} j$ if $i, j > n$. If $i, j > n$ then there exist $i', j' \in \{1, 2, \ldots, n\}$ such that $2n - i' = i$ and $2n - j' = j$ so that $i \sim_{(fg)h}$ implies that $i' \sim_{\widehat{(fg)h}} j'$ so $i' \sim_{\widehat{h}(\widehat{g}\widehat{f})} j'$. We can now apply the result above to conclude that $i' \sim_{(\widehat{h}\widehat{g})\widehat{f}} j'$ and so $i' \sim_{\widehat{(f(gh)}} j'$ hence $i \sim_{f(gh)} j$.

So we have shown that $i \sim_{(fg)h} j$ implies that $i \sim_{f(gh)} j$ for $i, j \leq n$, $i \leq n < j$ and $i, j > n$. The last case $j \leq n < i$ follows by reflexivity from $i \leq n < j$. Hence multiplication of Brauer diagrams is associative and B_n is a monoid. □

There is one very useful function when dealing with Brauer diagrams. Define $\ell(d)$ (the *length* of d) to be the number of pairs of 2-subsets $\{i, j\}, \{k, l\}$ of $d \in \widetilde{B_n}$ such that $i < k < l < j$. In a nice diagram of d this is just the number of intersections of lines in the diagram. Equipped with the function $\ell : \widetilde{B_n} \to \mathbb{N}$ we are able to define a partial order on $\widetilde{B_n}$. We say that $d \leq e$ if $\ell(d) < \ell(e)$ or $d = e$. It turns out that this partial order is a very useful one. For example $\widetilde{B_3}$ has the following Hasse diagram:



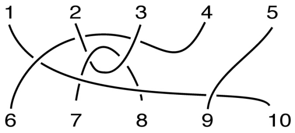
Note that each element of each row has the same length and that each element is covered by every element of the row above.

From the above diagram it is clear that $\widetilde{B_3}$ has 15 elements. In fact there is a very simple general formula for the number of elements of $\widetilde{B_n}$. First imagine that we place $2n$ labeled balls in a line. Our task is to count the number of ways that we can divide these balls into groups of two. We first take the first ball from the front of the line and group it with a randomly chosen ball: there are $2n-1$ ways to do this. We then take the next ball from the front of the line and group it with another randomly chosen ball: since three balls are gone there are $2n-3$ ways to do this. Continuing in this manner we see that the number of ways to divide the balls into groups of 2 is $(2n-1)(2n-3)(2n-5)\dots 5.3.1$. This "odd factorial" is very useful and is given its own notation (defined if n is even): $n!! = (n-1)(n-3)\dots 5.3.1$. From the above observations it is clear that $\widetilde{B_n}$ has $(2n)!!$ elements.

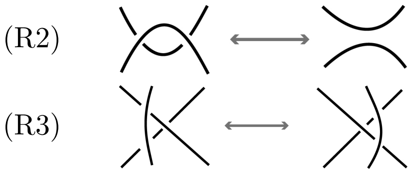
1.2 Tangles and the BMW Monoid

In the previous section we introduced the Brauer monoid as a generalisation of the Symmetric group. In this section we introduce Tangles and BMW monoid which can be seen as a further generalisation of the Brauer monoid. The basic idea is that, in diagrams representing elements of the Brauer monoid, we saw each line joining two dots as a two dimensional thing; it neither passed under or over each line it crossed. In Tangles and the BMW monoid we imagine the lines as little pieces of string which join rows of dots in three-dimensional space. Thus whether they pass under or over each other is important.

Because over and under crossings are important there is no easy way to represent elements as partitions of sets etc. So we define a n-tangle to be a picture in the plane in which 2 rows of n dots are joined by lines such that each line connects two dots and every dot is joined to precisely one line. For example the following is a 5-tangle:



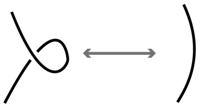
Since an element of the tangle monoid is defined to be a picture we need a simple way to tell if two tangles are “essentially the same”. By “essentially the same” we mean that if we constructed the two tangles physically could we make one identical to the other by moving the bits of string about without cutting or retying. It turns out that the following two identities (known famously as the Reidemeister moves of types II and III) very nearly reflect our physical notion of “essentially the same”:



So if it is possible to transform one tangle into the other using only the above Reidemeister moves (or any rotation of them) in a local portion of the plane then we regard them as equivalent. We then define $\mathcal{T}_n$ to be the set of all possible tangles on $2n$ dots modulo the equivalence relation generated by R2 and R3.

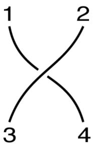
The move from the set of all pictures to equivalence classes of pictures occasionally provides some difficulties when we want to talk about pictures on a very basic level. Thus it is helpful to define simple ways to move between pictures and and elements of the equivalence class. We write t_p for a representative picture of $t \in \mathcal{T}_n$. If no sequence of Reidemeister moves exist that will reduce the number of crossings of t_p we say that t_p is a *nice* picture of t. And, given a picture p we write $\underline{p}$ for the equivalence class of which p is a member. We also write $p \sim q$ if p and q are equivalent under Reidemeister moves. Thus if $t_p \sim p$ then $t = \underline{p}$.

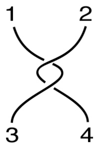
The astute reader may have noticed that the relations allowed under the Reidemeister moves doe not entirely cover our physical conception of “essentially the same”. We would physically expect to be able to untangle a self-intersecting loop:

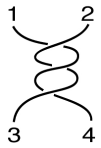


However it is not possible to achieve this with the two Reidemeister moves given above. (This move is the missing Reidemeister of type I). It turns out that the extra structure given by not allowing ourselves to untangle self-intersecting loops is important.

Since the set of tangles $\mathcal{T}_n$ is a generalisation of the Brauer monoid, we want to define multiplication by concatenation. However, when we try to do this we run into problems. Recall that, disregarding closed loops, the Brauer monoid was finite with $(2n)!!$ elements. However, $\mathcal{T}_n$ is infinite even when closed loops are ignored. It is very easy to get an infinite submonoid of $\mathcal{T}_n$. In the case $n = 2$ consider the following powers of x in $\mathcal{T}_n$:

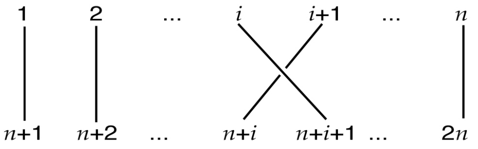
if $x =$ 

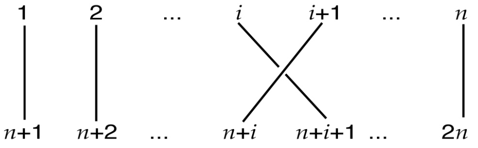
then $x^2 =$ 

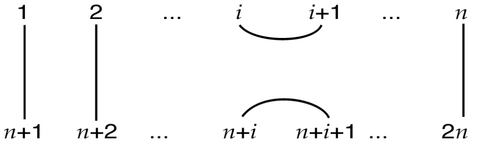
and $x^3 =$ 

Clearly $x^i \neq x^j$ if $i \neq j$ and so $\mathcal{T}_n$ has an infinite number of elements. However these elements don't present too much of a problem because then all occur as powers of one element. That is, even though the submonoid generated by x is infinite it is *finitely generated.* However the full monoid is not even finitely-generated.

To get around this problem of finite presentation we only consider a submonoid of $\mathcal{T}_n$. We call an element $d \in \mathcal{T}_n$ reachable if it can be expressed as a product of finitely many s_i, s_i^{-1} and e_i's where the s_i's, s_i^{-1}'s and e_i's are:

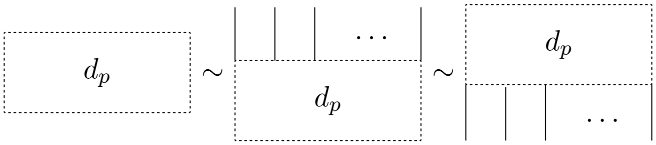
$s_i =$  for $1 \leq i \leq n-1$

$s_i^{-1} =$  for $1 \leq i \leq n-1$

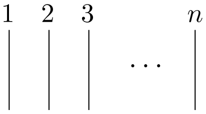
$e_i =$  for $1 \leq i \leq n-1$

Lemma 1.3. *The set of reachable elements of T_n, with multiplication defined by concatenation form a finitely generated monoid, the Birman-Murakami-Wenzl or BMW$_n$ monoid.*

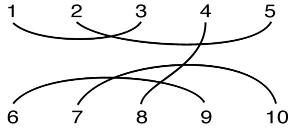
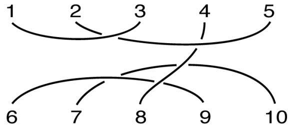
Proof. BMW$_n$ is finitely generated by definition. To show associativity it is enough to show that any three elements chosen from $\{s_i\} \cup \{s_i^{-1}\} \cup \{e_i\}$ satisfy associativity. This is straightforward and will not be shown. Lastly, to see that BMW$_n$ has an identity let $d \in$ BMW$_n$ be arbitrary, and let d_p be a picture or d. Then we can apply a continuous transformation to d_p to see that:



And hence $id_{\text{BMW}}\ d = d\ id_{\text{BMW}} = d$ with:

$id_{\text{BMW}} =$  □

There are two very important functions between the Brauer and BMW monoids. The first function ϕ "flattens" an element of BMW_n by throwing away over and under crossings. Define $\phi : \text{BMW}_n \to B_n$ by $\{i, j\} \in \phi(t)$ if i is joined to j in $t \in \text{BMW}_n$. The second function, called the trace of $d \in B_n$, provides a means of "lifting" an element $d \in B_n$ to BMW_n. To form the trace of an element $d \in B_n$ we draw the edges from left to right across the top and then left to right across the bottom lifting the pen briefly when we cross a line that has already been drawn. For example if:

$d =$  then $T_d =$ 

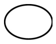
1.3 The Monoid Algebras of B_n and BMW_n

An *algebra* is an R-module M together with some rule that allows multiplication of elements. A simple example of an algebra is the complex numbers $\mathbb{C}$ considered as a vector space over $\mathbb{R}$ with multiplication defined by $(a + ib)(c + id) = ac - bd + i(ad + bc)$. It is often inconvenient (and impossible if our module has infinite rank) to specify multiplication in terms of two arbitrary elements of the module as we have just done. Instead it is convenient to specify *generators* and *relations*. For example, we could say that the complex numbers are an $\mathbb{R}$-algebra on generators 1 and i with defining relation $i^2 = -1$. Given the product $(a + ib)(c + id)$ we expand to get $ac + iad + ibc + i^2bd$ and then apply the relation to get $ac - bd + i(ad + bc)$.

As the number of generators grows we need to specify more and more relations in order to keep the rank of the module finite. One convenient situation occurs when we use a group, monoid or semi-group as a basis for a free R-module. Because we already know how elements multiply in the group, monoid or semi-group there is no need to specify any relations at all; we just specify the group, monoid or semi-group together with the ring R. To multiply two elements we expand the product and apply the group, monoid or semi-group multiplication. In the case of a monoid this construction is known as the *monoid algebra* and is denoted RM where R is the ring and M is the monoid. For example, consider the symmetric group Sym_2 which has two elements: 1 and σ where $\sigma^2 = 1$. Since Sym_2 is a group it is a monoid and so we can form the monoid algebra $\mathbb{C}Sym_2$. We have:

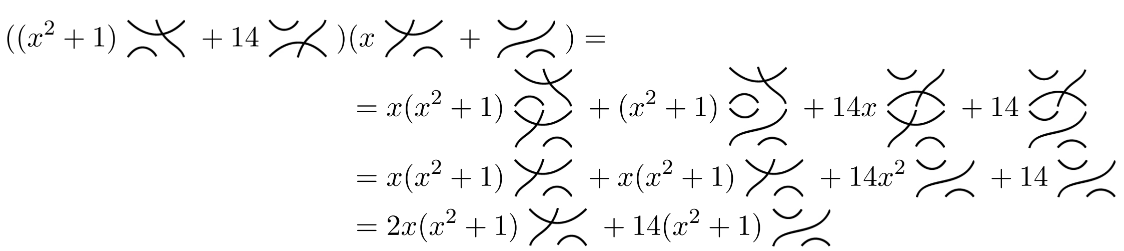
$$\begin{aligned}(a + b\sigma)(c + d\sigma) &= ac + ad\sigma + bc\sigma + bd\sigma^2 \\ &= (ac + bd) + (ad + bc)\sigma \qquad\qquad (\text{Since } \sigma^2 = 1)\end{aligned}$$

The purpose of forming the algebra of a group or monoid is to further understand the structure of the group or monoid. So often it is worthwhile to choose the ring over which the algebra is defined carefully as it is often possible to translate certain structural features of the monoid into features of the ring over which the algebra is defined. For example, in the case of B_n, closed loops presented a problem and caused the monoid to be infinite. We might hope to find a ring and some relation which removes the difficultly introduced by closed loops. Note that if we use a polynomial ring $R[x]$ together with the relation:

 $= x$

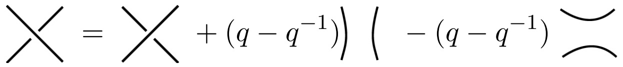
Then we can use the exponent of x to count the number of closed loops of an element $d \in B_n$. For example if $d \in B_n$ has four closed loops and d' is the same as d except with the closed loops removed then $d = x^4 d'$

It is conventional to choose $R = \mathbb{Z}$ and so the ring over which we define the Brauer algebra is $\mathbb{Z}[x]$. As an example we expand a product in $\mathbb{Z}[x]B_3$:

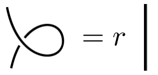

$((x^2+1)$ $+ 14$ $)(x$ $+$ $) =$
$= x(x^2+1)$ $+ (x^2+1)$ $+ 14x$ $+ 14$
$= x(x^2+1)$ $+ x(x^2+1)$ $+ 14x^2$ $+ 14$
$= 2x(x^2+1)$ $+ 14(x^2+1)$

We have already seen that $|\widetilde{B_n}| = (2n)!!$ and it follows that $\mathbb{Z}[x]B_n$ has rank $(2n)!!$.

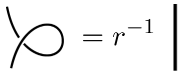
The monoid algebra construction for BMW$_n$ is more complicated. We have already seen that BMW$_n$ is infinite and so we cannot hope that the monoid algebra of BMW$_n$ will have finite rank unless we add some extra relations. The goal is to add enough relations so that the module will have finite rank but not add so many that we loose all of the structure of the BMW$_n$ monoid. The following relations probably seem quite arbitrary initially:



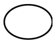
(Untangling)



(Self-Intersection I)

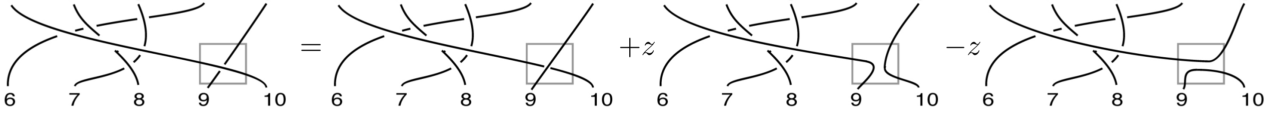


(Self-Intersection II)

 $= x$

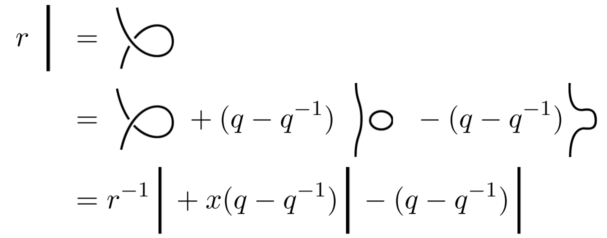
(Closed Loop Removal)

What we mean by these relations is that we can alter any tangle in a local portion of the picture by these identities. For example:



where $z = q - q^{-1}$.

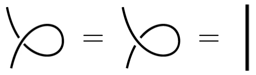
We can also use these identities to get a relation between x, r and q:



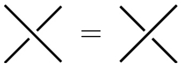
$$\therefore \qquad r - r^{-1} = (q - q^{-1})(x - 1)$$

So it makes sense to define the ring over which we want to build the monoid algebra of BMW_n to be $\mathcal{A} = \mathbb{Z}[x, q, q^{-1}, r, r^{-1}]/((r - r^{-1}) + (q - q^{-1})(1 - x))$. It is unfortunate that in the literature there is no distinction in the notation for the BMW_n algebra $\mathcal{A}\mathrm{BMW}_n$ and the BMW_n monoid. However once we have defined the BMW_n algebra there is rarely a need to refer again to the BMW_n monoid.

Notice also that for special values of $q, r \in \mathbb{C}$ the BMW_n algebra can be made to retain or abandon various structural features. For example if $r = 1$ then the self-intersection relations become:



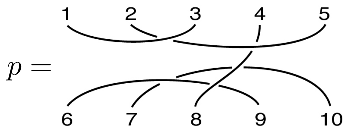
which is the Reidemeister move of type I. If q is also set to 1 then the untangling relation becomes:



and so if $q = r = 1$ then BMW_n is just the Brauer algebra $\mathbb{Z}[x]B_n$.

We next want to state and prove the BMW Basis theorem which says that the traces of the Brauer diagrams $\{\ T_d \mid d \in B_n\}$ form a basis for BMW_n. However before we can state the theorem we need to introduce a length function ℓ' on BMW_n.

To formally define the length function $\ell' : (BMW_n)_{\text{monoid}} \to \mathbb{N}$ we first define the crossings function c. If p is a picture of a tangle we define $c(p)$ to be the number of crossings of p. For example if:



then $c(p) = 5$. Clearly $c(p)$ depends on our choice of representative from the equivalence class, since we can apply the Reidemeister move of type II any number of times to add 2 to $c(p)$ each time. For this reason we define $\ell'(t) = \min\{c(p) \mid p \in t\}$ (recall that $t \in \mathcal{T}_n$ is an equivalence class of tangles). In other words, given a tangle $t \in \mathcal{T}_n$ we look at all possible pictures of it and count the number of crossings in each picture, $\ell'(t)$ is the minimum of this number.

Lemma 1.4. *Let d and t be elements of the Brauer and BMW monoids respectively. Then $\ell(d) = \ell'(T_d)$ and $\ell(\phi(t)) \leq \ell'(t)$.*

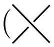
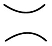
Proof. We first show that $\ell(\phi(t)) \leq \ell'(t)$. Let $\ell(\phi(t)) = a$. This means that there are a pairs of $\{i, j\}, \{k, l\}$ in $\phi(t)$ such that $i < k < j < l$. But if i is joined to j in $\phi(t)$ then i must be joined to j in t. Hence there are at least a pairs of dots $\{i, j\}, \{k, l\}$ in t such that $i < k < j < l$. Hence $\ell(\phi(t)) \leq \ell'(t)$. Now it is clear that $\ell'(T_d) \leq \ell(d)$ (this can be seen by tracing a nice diagram of d). But also $\phi(T_d) = d$ and so $\ell(d) \leq \ell'(T_d)$ (from above) and so $\ell'(T_d) = \ell(d)$. □

Because of the relations above we denote ℓ' by ℓ from now on. We can now state the main theorem:

BMW Basis Theorem. *Let $\mathbb{Z}[x]B_n$ be the Brauer algebra on $2n$ dots and BMW_n be the algebra as defined above. Then BMW_n is a free module of rank $(2n)!!$ and $\{T_d \mid d \in B_n\}$ is a basis for BMW_n, moreover there exists a length function, ℓ, on the BMW monoid such that $\ell(T_d) = \ell(d)$ and:*

$$t = \sum_{\ell(d) \leq \ell(t)} \lambda_d T_d \qquad\qquad \lambda_d \in \mathcal{A}$$

for all reachable tangles $t \in \mathcal{T}$.

It will take us a little while to prove this result. We first make an observation that in applying the untangling relation above, there is no chance of introducing a tangle which has a higher number of crossings than the tangle we started with. To see this let $t \in \mathcal{T}_n$ be a reachable tangle and let p be a picture of t such that $\ell(t) = c(p)$. Then if we let p' be the picture obtained from p by replacing a crossing ( or) with a or  then $c(p') < c(p)$. Note that this implies that if t' is the tangle of which p' is a representative then $\ell(t') < \ell(t)$. In other words, in applying the untangling relation we only ever introduce tangles whose number of crossings are either equal or less than the number of crossings in the original tangle.

The following two lemmas are important:

Lemma 1.5. *If $t \in \mathcal{T}_n$ is reachable and has double or self-intersection then it is possible to write $t = \sum_i \lambda_i t_i$ with $\lambda_i \in \mathcal{A}$ and $\ell(t_i) < \ell(t)$ for all i.*

Proof. Let $t \in \mathcal{T}_n$ be reachable and have a self-intersection. Let p be a picture of t such that $\ell(t) = c(p)$. Now we can focus on a vertex l which has a self-intersection and move around the loop applying the untangling relation whenever l passes over a strand. Thus we can write:

$$t = t' + \sum_i \lambda_i t_i \qquad \text{with } \ell(t_i) < \ell(t) \text{ and } \lambda_i \in \mathbb{Z}[q - q^{-1}]$$

with t' such that every strand that crosses the loop of l passes *over* l. We can then apply Reidemeister moves of types II and III to isolate the loop. It is now possible to write $t' = rs$ or $t' = r^{-1}s$ with $\ell(s) < \ell(t')$ by the self-intersection relations. Hence we have:

$$\begin{aligned} t &= \lambda s + \sum_i \lambda_i t_i \qquad \text{with } \ell(t_i) < \ell(t),\ \lambda_i \in \mathbb{Z}[q - q^{-1}] \text{ and } \lambda \in \mathbb{Z}[r, r^{-1}] \\ &= \sum_i \gamma_i t_i \qquad \text{with } \gamma_i \in \mathcal{A} \text{ and } \ell(t_i) < \ell(t) \text{ for some } t_i \text{ and } \gamma_i \end{aligned}$$

A very similar process yields such an expression for $t \in \mathcal{T}_n$ with a double intersection: We concentrate on the "loop" formed by the double intersection and move around it applying the untangling relation so that every strand which crosses into the loop crosses *over*. This allows us to write:

$$t = t' + \sum_i \lambda_i t_i \qquad \text{with } \ell(t_i) < \ell(t) \text{ for all } i$$

We then apply the Reidemeister moves of types II and II to see that t' is equivalent to a tangle with the double intersection by itself. There are then two possibilities: either we can apply the Reidemeister move of type II to remove double intersection or we can't. If we can the we see that $t' \sim t''$ with $c(t'') = c(t')$. Hence it is possible to write:

$$t = \sum_i \gamma_i t_i \qquad \text{with } \gamma_i \in \mathcal{A} \text{ and } \ell(t_i) < \ell(t)$$

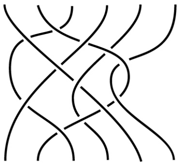
On the other hand, if we can't apply the Reidemeister move we can apply the untangling relation to one of the double intersections to write:

$$t' = t'' + (q - q^{-1})t_1 - (q + q^{-1})t_2 \qquad \text{with } \ell(t_1), \ell(t_2) < \ell(t)$$

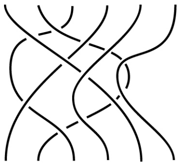
We can then apply the Reidemeister move of type II to see that $t'' \sim t'''$ with $c(t''') = c(t'') - 2$. Hence:

$$\begin{aligned} t &= t''' + (q - q^{-1})t_1 - (q - q^{-1})t_2 + \sum_i \lambda_i t_i \\ &= \sum_i \gamma_i t_i \qquad \text{with } \gamma_i \in \mathcal{A} \text{ and } \ell(t_i) < \ell(t) \end{aligned} \qquad \square$$

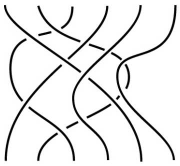
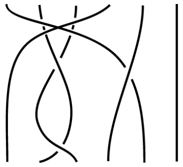
We illustrate this process with an example. Consider the following tangle with a double intersection:

$t =$ 

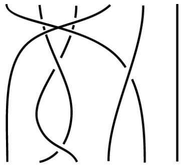
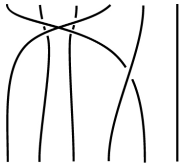
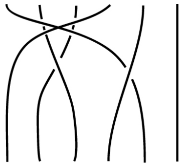
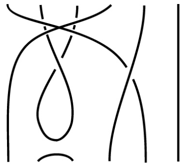
We can repeatedly apply the untangling relation to write:

$t =$  $+ \sum_i \lambda_i t_i \qquad \text{with } \ell(t_i) < \ell(t) \text{ for all } i$

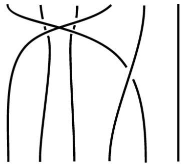
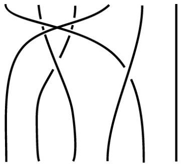
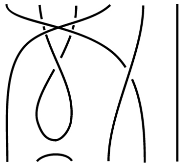
And we can apply the Reidemeister moves of types II and III to show that:

 $\sim$ 

And lastly:

 $=$  $+\ (q-q^{-1})$  $-\ (q-q^{-1})$ 

Hence:

$t\ =$  $+\ (q-q^{-1})$  $-\ (q-q^{-1})$  $+\sum_i \lambda_i t_i$

Which is the expression required by the lemma.

Lemma 1.6. *If $t \in \mathcal{T}_n$ has no double or self-intersection then it is possible to write $t = T_{\phi(d)} + \sum_i \lambda_i t_i$ with $\lambda_i \in \mathcal{A}$ and $\ell(t_i) < \ell(t)$ for all i.*

Proof. Let t be a tangle without double or self-intersection. Moving from left to right across the top and then left to right along the bottom we can follow each strand applying the untangling relation at any point at which the strand we are following passes *below* another. Repeating this process across the whole tangle allows us to write:

$$t = t' + \sum_i \lambda_i t_i \qquad \text{with } \ell(t_i) < \ell(t)$$

Now t' is uniquely determined by $\phi(t')(= \phi(t))$ and the property that a strand linking i to j passes over a strand linking k to m if and only if $i < k$, so we can conclude that $t' = T_{\phi(t)}$. Hence:

$$t = T_{\phi(t)} + \sum_i \lambda_i t_i \qquad \text{with } \ell(t_i) < \ell(t) \qquad \square$$

The proof is now relatively straightforward:

Proof of BMW Basis Theorem. Note that to prove the result we only need to show that we can express any reachable $t \in \mathcal{T}_n$ as a linear combination of the $\{T_d \mid d \in B_n\}$. We prove this result by induction on $\ell(t)$. The result is trivial if $\ell(t) = 0$ since $t = T_{\phi(t)}$ so assume that for all t such that $\ell(t) < k$ we can write:

$$t = \sum_{\ell(d) \leq \ell(t)} r_{d,t} T_d$$

Now let $s \in \mathcal{T}_n$ be reachable and assume that $\ell(s) = k$. If s has no self or double intersection we can write:

$$s = T_{\phi(s)} + \sum_i \lambda_i t_i$$

but since $\ell(t_i) < \ell(s) = k$ by induction we can write:

$$t_i = \sum_{\ell(d) \leq \ell(t_i)} r_{d,t_i} T_d$$

for all i. Hence:

$$\begin{aligned} s &= T_{\phi(s)} + \sum_i \lambda_i \sum_{\ell(d)\leq\ell(t_i)} r_{d,t_i} T_d \\ &= \sum_i \gamma_i T_{d_i} \qquad \text{for some } \gamma_i \in \mathcal{A} \text{ and } d_i \in B_n \end{aligned}$$

Similarly, if s has self or double intersection we can write:

$$\begin{aligned} s &= \sum_i \lambda_i t_i \qquad \text{with } \ell(t_i) < \ell(s) \\ &= \sum_i \lambda_i (\sum_{\ell(d)\leq\ell(t_i)} r_{d,t_i} T_d) \\ &= \sum_i \gamma_i T_{d_i} \qquad \text{for some } \gamma_i \in \mathcal{A} \text{ and } d_i \in B_n \end{aligned}$$ □

An alternative proof of this result shows that:

$$\mathrm{BMW}_n \otimes_{\mathcal{A}} \mathbb{Z}[x] \cong \mathbb{Z}[x] B_n$$

And hence $\mathrm{BMW}_n \otimes_{\mathcal{A}} \mathbb{Z}[x]$ has rank $(2n)!!$ as a $\mathbb{Z}[x]$-module. Since $\{T_d \mid d \in B_n\}$ is a linearly independent set with $(2n)!!$ elements they must form a basis. See [1].

2 Canonical Bases

2.1 The Main Theorem

In the have abstract setting we have a free $\mathbb{Z}[q,q^{-1}]$-algebra A with basis $\{\tau_i\}_{i\in\Lambda}$ such that Λ is interval finite (this just means that every interval $\{x\in\Lambda|x\leq k\}$ has a finite number of elements). We also have an endomorphism $^{-}:\mathbb{Z}[q,q^{-1}]\to\mathbb{Z}[q,q^{-1}]$ which sends q to q^{-1} together with a $\mathbb{Z}$–linear function $\iota:A\to A$ satisfying:

(i) $\iota^2=id_F$

(ii) $\iota(av)=\overline{a}\iota(v)$ (this is termed *anti-linearity*)

Such a function is called an *involution.*

It is an important problem of representation theory to find a "Canonical Basis" for the pair (A,ι). A Canonical Basis is a *unique* basis $\{C_i\}$ for A as a module satisfying the following three conditions:

(1) $C_j\in\sum_{i\leq j}\mathbb{Z}[q^{-1}]\tau_i$ for all j.

(2) $\iota(C_i)=C_i$. That is, $\{C_i\}$ is fixed by ι.

(3) $\pi(C_i)=\tau_i$ where π is the natural projection from $A=\sum\mathbb{Z}[q,q^{-1}]\tau_i$ to $\sum\mathbb{Z}\tau_i$.

The following theorem (from [2]) gives a particular set of conditions which guarantee the existence of a Canonical Basis. The proof is given because it is the base of the algorithm that we construct later on.

Main Theorem. *Let A be a free $\mathbb{Z}[q,q^{-1}]$-module with basis $\{\tau_i\}$, ι an involution on A and Λ an interval finite poset. Now suppose that $\iota(\tau_j)=\sum_{i\leq j}r_{ij}\tau_i$ with $r_{jj}=1$. That is, every $\iota(\tau_i)$ is expressible as a linear combination of elements all of which are less than τ_i (with the order inherited from Λ). Then:*

1. *$\sum_j r_{ij}\overline{r_{jk}}=\delta_{ik}$ (where δ is the Kronecker delta)*
2. *There exists a unique basis $\{C_j\}$ for A such that $\iota(C_j)=C_j$ for all i and $C_j=\sum_{i\leq j}p_{ij}\tau_i$ with $p_{ij}\in q^{-1}\mathbb{Z}[q^{-1}]$ for $i<j$, $p_{ii}=1$ and $p_{ij}=0$ if $j>i$.*

In other words a Canonical basis exists.

Proof. Since ι is an involution we have that $\iota^2(\tau_i)=\tau_i$. Also by assumption we can express

$\iota(\tau_i) = \sum_{j \leq i} r_{ji}$ we have:

$$\begin{aligned}
\tau_i &= \iota^2(\tau_i) \\
&= \iota(\sum_{j \leq i} r_{ji}\tau_i) \\
&= \sum_{j \leq i} \overline{r_{ji}}\iota(\tau_j) \\
&= \sum_{j \leq i} \overline{r_{ji}} \sum_{k \leq j} r_{kj}\tau_k \\
&= \sum_{k \leq j \leq i} \overline{r_{ji}} r_{kj}\tau_k
\end{aligned}$$

Since $\{\tau_i\}$ is a basis for A we have $\sum_{k \leq j \leq i} \overline{r_{ji}} r_{kj} = \delta_{ki}$.

The second part of the Lemma (the existence of the unique basis $\{C_j\}$) is more difficult. We first note that:

$$\begin{aligned}
\sum_{j \leq k} p_{jk}\tau_j &= C_k = \iota(C_k) \\
&= \sum_{j \leq k} \overline{p_{jk}}\iota(\tau_j) \\
&= \sum_{j \leq k} \overline{p_{jk}} \sum_{i \leq j} r_{ij}\tau_i \\
&= \sum_{i \leq j \leq k} r_{ij}\overline{p_{jk}}\tau_i \\
\therefore \qquad p_{ik} &= \sum_{i \leq j \leq k} r_{ij}\overline{p_{jk}} \qquad (1)
\end{aligned}$$

So that the existence of the unique basis $\{C_j\}$ is equivalent to the system (1) having a unique solution for all p_{ik}. We also note that, since $r_{ii} = 1$, $p_{ik} - \overline{p_{ik}} = \sum_{i < j \leq k} r_{ij}\overline{p_{jk}}$. So the polynomial $\alpha_{(i,k]} = \sum_{i < j \leq k} r_{ij}\overline{p_{jk}}$ is of crucial importance in finding the p_{ik}'s.

We now show that $\alpha_{(i,k]} + \overline{\alpha_{(i,k]}} = 0$. To do this we use induction on $j \in \Lambda$. So assume that the p_{jk}'s are known for all $i < j \leq k$:

$$\begin{aligned}
\alpha_{(i,k]} &= \sum_{i < j \leq k} r_{ij}\overline{p_{jk}} \\
&= \sum_{i < j \leq k} r_{ij}\iota(\sum_{j \leq s \leq k} r_{js}\overline{p_{sk}}) \qquad \text{(By induction from (1))} \\
&= \sum_{j \leq s \leq k} (\sum_{i < j \leq k} r_{ij}\overline{r_{js}})p_{sk} \\
&= \sum_{i \leq s \leq k} (\sum_{i \leq j \leq s} r_{ij}\overline{r_j s} - r_{ii}\overline{r_{is}})p_{sk} \\
&= -\sum_{i \leq s \leq k} \overline{r_{is}}p_{sk} \qquad \text{(Since } \sum_{k \leq j \leq i} \overline{r_{ji}} r_{kj} = \delta_{ki}\text{)} \\
&= -\overline{\alpha_{(i,k]}}
\end{aligned}$$

Now setting $\alpha_{(i,k]} = \sum_j a_j q^j$ we see immediately that $\alpha_{(i,k]} + \overline{\alpha_{(i,k]}} = 2a_0 + \sum_{j \neq 0} a_j q^j$ and so $\alpha_{(i,k]}$ has no constant term. Also:

$$\begin{aligned} 0 &= \sum_j a_j q^j + \sum_j a_j q^{-j} \\ &= \sum_j a_j q^j + \sum_j a_{-j} q^j \\ &= \sum_j (a_j + a_{-j}) q^j \end{aligned}$$

So $-a_j = a_{-j}$. Furthermore:

$$\begin{aligned} \alpha_{(i,k]} &= \sum_{j<0} a_j q^j + \sum_{j>0} a_j q^j \\ &= \sum_{j>0} a_{-j} q^{-j} + \sum_{j>0} a_j q^j \\ &= \sum_{j>0} (-a_j) q^{-j} - \sum_{j>0} (-a_j) q^j \end{aligned}$$

So if we set $p_{ik} = \sum_{j>0} -a_j q^{-j}$ we see that $p_{ik} \in q^{-1}\mathbb{Z}[q^{-1}]$ and that p_{ik} is the required unique solution to the system in (1). □

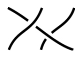
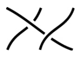
The theorem which has just been proved has an important corollary. The first is that if A has finite rank n (the only case that we will be considering) the above calculations can be reinterpreted as statements about matrices over $\mathbb{Z}[q^{-1}, q]$ and $\mathbb{Z}[q^{-1}]$:

Corollary 2.1. *Let A be as above and let $\iota : A \to A$ be an anti-linear involution described by right-multiplication by a matrix R with respect to a basis $\{\tau_i\}$. Then if R is unitriangular (that is upper triangular with 1's down the the diagonal) then $R\overline{R} = I$ and the system $P = R\overline{P}$ has a unique solution for $P \in Mat_{n\times n}(\mathbb{Z}[q^{-1}])$ under the conditions that P is unitriangular and $p_{ij} \in q^{-1}\mathbb{Z}[q^{-1}]$ if $i < j$.*

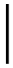
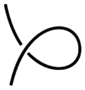
2.2 Applying the Main Theorem to the BMW Algebra

We are now in a position to apply the main theorem to the BMW Algebra. So we need to find an involution ι on the BMW algebra such that $\iota(\tau_i) = \tau_i + \sum_{\tau_j < \tau_i} \lambda_i \tau_j$ for all i under some partial order on $\{\tau_i\}$.

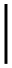
It turns out that the involution that concerns us in the BMW algebra is the involution which interchanges over and under crossings in the BMW monoid. For example, if we call this involution ι then:

$\iota($  $) =$ 

Note that we still need to discover ι's action on $\mathcal{A}$. We can use the relations of the BMW algebra to work out possibilities:

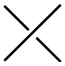
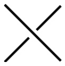
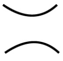
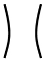
r^{-1}  $=$  (By self-intersection I)

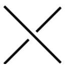
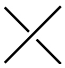
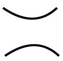
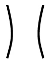
$= \iota(r$ $)$

$= \iota(r)$  (Since ι is an involution)

$\therefore \quad \iota(r) = r^{-1}$

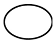
Similarly:

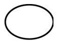
 $=$  $+ (q - q^{-1})$  $- (q - q^{-1})$  (By the untangling relation)

$\therefore$  $=$  $+ (\iota(q) - \iota(q^{-1}))$  $- (\iota(q) - \iota(q^{-1}))$ 

$\therefore \quad \iota(q) - \iota(q^{-1}) = q^{-1} - q$

So it makes sense to define $\iota(q) = q^{-1}$. Lastly:

$\iota(x) = \iota($  $)$

$=$ 

$= x$

So we define ι to be the involution on BMW$_n$ which interchanges over and under crossings in the monoid and takes q to q^{-1}, r to r^{-1} and x to x.

The reason that we can use this involution in conjunction with the main theorem is due to the following theorem:

Theorem 2.2. *Let d be an element of the Brauer monoid B_n and let ι be as above. Then it is possible to write:*

$$\iota(T_d) = T_d + \sum_{\ell(d') < \ell(d)} \lambda_{d'} T_{d'}$$

Proof. Let $d \in B_n$ by arbitrary. Since T_d has no self or double intersections (it is the trace of a nice diagram) we can apply Lemma 1.6 and write:

$$\iota(T_d) = T_{\phi(\iota(T_d))} + \sum_i \lambda_i t_i \qquad \text{with } \ell(t_i) < \ell(\iota(T_d))$$

but $\phi(\iota(T_d)) = d$ and $\ell(\iota(T_d)) = \ell(T_d) = \ell(d)$ so we can write:

$$\iota(T_d) = T_d + \sum_i \lambda_i t_i \qquad \text{with } \ell(t_i) < \ell(d)$$

Now, from the basis theorem we can write each t_i as:

$$t_i = \sum_{\ell(e) \leq \ell(t_i)} r_{e,t_i} T_e$$

So:

$$\begin{aligned}\iota(T_d) &= T_d + \sum_i \lambda_i(\sum_{\ell(e)\leq\ell(t_i)} r_{e,t_i}T_e) \\ &= T_d + \sum_{\ell(e)<\ell(d)} \gamma_e T_e\end{aligned}$$

for some $\gamma_i \in \mathcal{A}$. □

So we can use the main theorem with the BMW algebra, involution ι and partial order on the basis $T_d \leq T_e$ if $\ell(d) < \ell(e)$ or $d = e$ to conclude that a canonical basis exists.

2.3 Cells

One of the principle motivations for finding canonical bases for an algebra A is that they provide nice bases for expressing A as a direct sum of A-modules $A = M_1 \oplus M_2 \oplus \cdots \oplus M_n$ (such an expression is called a *representation*). In this section we describe how to find the left, right and two-sided cells of BMW$_n$ once a canonical basis has been found. These cells provide a straightforward means to find representations of BMW$_n$.

Central to the construction of the cells is the notion of a *preorder*. A preorder is a binary relation $\leq$ on a set B satisfying:

1. $b \leq b$ for all b (reflixivity)
2. $a \leq b$ and $b \leq c$ implies that $a \leq c$ (transitivity)

(For those familiar with the notion of a *partial order* it should be noted that a partial order is a pre-order with the extra condition that $a \leq b$ and $b \leq a$ implies that $a = b$. A preorder is aptly named because if $\leq$ is a preorder on a set B we can define an equivalence relation by $a \sim b$ if $a \leq b$ and $b \leq a$. Then $\leq$ becomes a partial order on $B/\sim$.)

Given a canonical basis $\{C_i\}$ for BMW$_n$ we say that $C_x \leq_L C_y$ if there exists a C_z such that the coefficient of C_x in C_zC_y as a linear combination of the the $\{C_i\}$ is non-zero. Similarly we say that $C_x \leq_R C_y$ if there exists a C_z such that the coefficient of C_x in the product C_yC_z is nonzero. We also say that $C_x \leq_{LR} C_y$ if $C_x \leq_L C_y$ or $C_x \leq_R C_y$. For example, in BMW$_3$:

$$\begin{aligned}C_{14}C_{12} = ((-q^{-3} - q^{-1})x + ((q^{-2}+1)r^{-1} + (q^{-3}+q^{-1}) + (-q^{-2}-1)r))C_9 + \\ + (-q^{-2}x + (q^{-1}r^{-1} + (q^{-2}+1) - q^{-1}r))C_{14}\end{aligned}$$

And so we can conclude that $C_9, C_{14} \leq_L C_{12}$ and $C_9, C_{14} \leq_R C_{14}$. (See Section 3.2 for a justification of the above calculation).

Lemma 2.3. *$\leq_L$ is a preorder on $\{C_i\}$.*

Proof. From Lemma 1.3 we know that id_{BMW} (the picture identical to a nice diagram of the identity of B_n) is the identity is the identity of BMW_n. Also $\iota(id_{\mathrm{BMW}}) = id_{\mathrm{BMW}}$. Now C_{id}, the canonical basis element corresponding to T_{id} satisfies:

$$C_{id} = \sum_{\ell(d) \leq \ell(id)} r_{d,id} T_d \qquad \text{with } r_{id,id} = 1$$

But under the partial order on B_n and BMW_n id_{BMW} is the only element less than or equal to id_{BMW}. Hence we can conclude that:

$$C_{id} = id_{\mathrm{BMW}}$$

Hence $id_{\mathrm{BMW}} \in \{C_i\}$ where $\{C_i\}$ is the canonical basis. Now $id_{\mathrm{BMW}}\ C_i = C_i$ for all i and so $C_i \leq C_i$ for all i.

Now assume that $C_j \leq_L C_i$ and $C_k \leq C_j$. Then there exists a and b such that $C_a C_i = \sum_m \lambda_m C_m$ and $C_b C_j = \sum_n \lambda_n C_n$ with $\lambda_j \neq 0$ and $\lambda_k \neq 0$. Now:

$$\begin{aligned}
(C_b C_a) C_i &= C_b(\sum_m \lambda_m C_m) && \text{with } \lambda_j \neq 0 \\
&= \sum_m \lambda_m C_b C_m \\
&= \dots + \lambda_j C_b C_j + \dots \\
&= \dots + \lambda_j \sum_n \lambda_n C_n + \dots && \text{with } \lambda_k \neq 0 \\
&= \dots + \dots + \lambda_j \lambda_k C_k + \dots + \dots
\end{aligned}$$

And hence there exists a C_d such that $C_d C_i = \dots + \lambda C_k + \dots$ with $\lambda \neq 0$ and so $C_k \leq_L C_i$ and so $\leq_L$ is a preorder. □

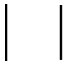
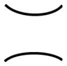
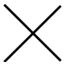
Given the preorder $\leq_L$ we can define an equivalence relation on $\{C_i\}$. We say that $C_i \sim C_j$ if $C_i \leq_L C_j$ and $C_j \leq_L C_i$. A set of equivalent basis elements Γ is called a *left-cell* of BMW_n (we similarly define right ($\leq_R$) and two-sided ($\leq_{LR}$) cells). As mentioned above, the importance of cells is that they lead to representations of BMW_n. If we set $F^\Gamma = \mathcal{A}\{C_i \leq_L \Gamma\}$ then F^Γ is a left ideal of BMW_n.

3 Results for $\mathbf{BMW}_2$ and $\mathbf{BMW}_3$

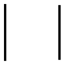
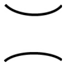
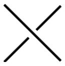
3.1 $\mathbf{BMW}_2$

As an example we first calculate the canonical basis for BMW_2 since the system is small enough to solve by hand:

There are $(2 \times 2)!! = 3 \times 1 = 3$ Brauer diagrams:

$b_1 =$  $b_2 =$  $b_3 =$ 

Their traces are:

$T_{b_1} = g_1 =$  $T_{b_2} = g_2 =$  $T_{b_3} = g_3 =$ 

Now we have:

$$\begin{aligned}
\overline{g_1} &= g_1 \\
\overline{g_2} &= g_2 \\
\overline{g_3} &= \qquad\qquad\qquad\qquad = g_3 + (q - q^{-1})g_2 - (q - q^{-1})h_1
\end{aligned}$$

Hence:

$$R = \begin{pmatrix} 1 & 0 & -(q - q^{-1}) \\ 0 & 1 & (q - q^{-1}) \\ 0 & 0 & 1 \end{pmatrix}$$

We rant to solve the system $P = R\overline{P}$ for:

$$P = \begin{pmatrix} 1 & p_{12} & p_{13} \\ 0 & 1 & p_{23} \\ 0 & 0 & 1 \end{pmatrix}$$

Hence:

$$\begin{aligned}
\begin{pmatrix} 1 & p_{12} & p_{13} \\ 0 & 1 & p_{23} \\ 0 & 0 & 1 \end{pmatrix} &= \begin{pmatrix} 1 & 0 & -(q - q^{-1}) \\ 0 & 1 & (q - q^{-1}) \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \overline{p_{12}} & \overline{p_{13}} \\ 0 & 1 & \overline{p_{23}} \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & \overline{p_{12}} & \overline{p_{13}} - (q - q^{-1}) \\ 0 & 1 & \overline{p_{23}} + q - q^{-1} \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}$$

We now can now reduce the problem to smaller equations:

$$\begin{aligned} & p_{12} = \overline{p_{12}} \\ \therefore \qquad & p_{12} = 0 \\ & p_{23} = \overline{p_{23}} - (q + q^{-1}) \\ \therefore \qquad & p_{23} = -q^{-1} \\ & p_{13} = \overline{p_{13}} + (q + q^{-1}) \\ \therefore \qquad & p_{13} = q^{-1} \end{aligned}$$

So $P = \begin{pmatrix} 1 & 0 & q^{-1} \\ 0 & 1 & -q^{-1} \\ 0 & 0 & 1 \end{pmatrix}$ is the unique solution. We have found the canonical basis:

$$\begin{aligned} C_1 &= h_1 \\ C_2 &= h_2 \\ C_3 &= q^{-1}h_1 - q^{-1}h_2 + h_3 \end{aligned}$$

Note that:

$$\begin{aligned} \overline{C_3} &= \overline{q^{-1}h_1} - \overline{q^{-1}h_2} + \overline{h_3} \\ &= qh_1 - qh_2 + h_3 + (q - q^{-1})h_2 - (q - q^{-1})h_1 \\ &= q^{-1}h_1 - q^{-1} + h_3 \\ &= C_3 \end{aligned}$$

So $\{C_1, C_2, C_3\}$ is indeed the canonical basis for BMW_2.

3.2 BMW$_3$

In the case of BMW$_3$ the calculations are much more involved. The matrix involved has $(2\times3)!! = 15$ rows and columns which necessitates solving 105 equations. For this reason a function was written in MAGMA to assist in the calculation. Here is our algorithm in pseudocode:

```
function CalculateCanonicalBasis(R)
    for $i$ from $(n-1)$ down to 1 do
        for $k$ from $(i+1)$ to $n$ do                         (1)
            $\alpha_{(i,k]} = \sum_{i<j\leq k} r_{ij}\overline{p_{jk}}$          (2)
            if $\alpha_{(i,k]} = 0$ then
                $p_{ik} = 0$
            else
                $p_{ik} = \sum_{j>0} -a_j q^{-j}$                  (3)
            end if
        end for
    end for
end function
```

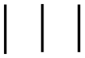
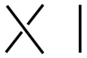
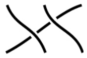
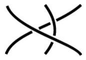
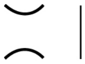
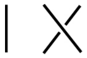
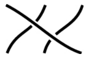
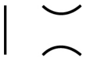
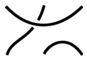
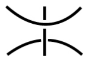
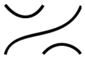
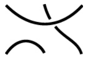
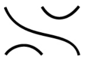
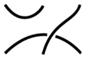
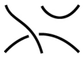
We make the following observations:

1 The two for loops have i moving from $(n-1)$ down to 1 and k from $(i+1)$ to n. This means that we start in the bottom right hand corner of the matrix P and work our way up row by row. Note that this progression is similar to the induction used in the proof of the main theorem.

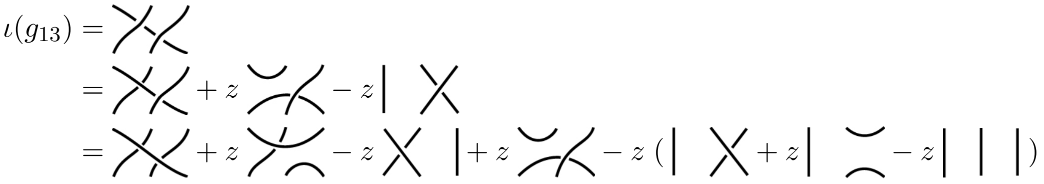
2 We use the same notation in the pseudocode to that of the proof of the main theorm. Note that, from the observation above, all the p_{ik}'s are know for $j > i$ and so it is possible to calculate $\alpha_{(i,k]}$.

3 This is the explicit form of the unique solution in the main theorem with $\alpha_{(i,k]} = \sum_j a_j q^j$.

Just as in the case of BMW$_2$ it is necessary to calculate the matrix of the involution before the algorithm is applied. Here is the basis of traces for BMW$_3$ (note that we have arranged the elements so that $g_i \leq g_j$ implies that $i \leq j$:

$\ell(g_i) = 0$	$\ell(g_i) = 1$	$\ell(g_i) = 2$	$\ell(g_i) = 3$
$g_1 =$ 	$g_6 =$ 	$g_{12} =$ 	$g_{15} =$ 
$g_2 =$ 	$g_7 =$ 	$g_{13} =$ 	
$g_3 =$ 	$g_8 =$ 	$g_{14} =$ 	
$g_4 =$ 	$g_9 =$ 		
$g_5 =$ 	$g_{10} =$ 		
	$g_{11} =$ 		

It is then a routine but lengthy process to calculate $\iota(g_i)$ in terms of the g_i for all i. To illustrate the process we calculate $\iota(g_{13})$ (as usual $z = q - q^{-1}$):



$$= g_{13} + zg_8 - zg_6 + zg_{10} - zg_7 - z^2g_3 + z^2g_1$$

Repeating this process for all the g_i yields the following table of the involution:

$$\begin{aligned}
\overline{g_1} &= g_1 \\
\overline{g_2} &= g_2 \\
\overline{g_3} &= g_3 \\
\overline{g_4} &= g_4 \\
\overline{g_5} &= g_5 \\
\overline{g_6} &= -zg_1 + zg_2 + g_6 \\
\overline{g_7} &= -zg_1 + zg_3 + g_7 \\
\overline{g_8} &= -zg_3 + zg_4 + g_8 \\
\overline{g_9} &= zg_2 - zg_5 + g_9 \\
\overline{g_{10}} &= zg_2 - zg_4 + g_{10} \\
\overline{g_{11}} &= -zg_3 + zg_5 + g_{11} \\
\overline{g_{12}} &= z^2g_1 - z^2g_3 - zg_6 - zg_7 + zg_9 + zg_{11} + g_{12} \\
\overline{g_{13}} &= z^2g_1 - z^2g_3 - zg_6 - zg_7 + zg_8 + zg_{10} + g_{13} \\
\overline{g_{14}} &= z^2g_2 + z^2g_3 - z^2g_4 - z^2g_5 - zg_8 + zg_9 + zg_{10} - zg_{11} + g_{14} \\
\overline{g_{15}} &= (-z^3 - z)g_1 + z^3g_3 + zg_4 + zg_5 + z^2g_6 + z^2g_7 - z^2g_8 - z^2g_{11} - zg_{12} - zg_{13} + zg_{14} + g_{15}
\end{aligned}$$

Note that when this is hand calculated mistakes almost inevitably occur. However Corollary 2.1 states that $R\overline{R} = I$ which is useful in verifying that the involution has indeed been calculated correctly. It is then possible to either hand calculate the canonical basis using the above algorithm or implement it in a computer algebra program. In MAGMA (see B.1):

```
> CalculateCanonicalBasis(Matrix_ZZ_to_ZQ(InvoMatrix));
[    1     0     0     0     0  q^-1  q^-1     0     0     0     0  q^-2  q^-2     0  q^-3]
[    0     1     0     0     0 -q^-1     0     0 -q^-1 -q^-1     0     0     0  q^-2 -q^-1]
[    0     0     1     0     0     0 -q^-1  q^-1     0     0  q^-1 -q^-2 -q^-2  q^-2 -q^-3]
[    0     0     0     1     0     0     0 -q^-1     0  q^-1     0     0     0 -q^-2     0]
[    0     0     0     0     1     0     0     0  q^-1     0 -q^-1     0     0 -q^-2     0]
[    0     0     0     0     0     1     0     0     0     0     0  q^-1  q^-1     0  q^-2]
[    0     0     0     0     0     0     1     0     0     0     0  q^-1  q^-1     0  q^-2]
[    0     0     0     0     0     0     0     1     0     0     0     0 -q^-1  q^-1 -q^-2]
[    0     0     0     0     0     0     0     0     1     0     0 -q^-1     0 -q^-1     0]
[    0     0     0     0     0     0     0     0     0     1     0     0 -q^-1 -q^-1     0]
[    0     0     0     0     0     0     0     0     0     0     1 -q^-1     0  q^-1 -q^-2]
[    0     0     0     0     0     0     0     0     0     0     0     1     0     0  q^-1]
[    0     0     0     0     0     0     0     0     0     0     0     0     1     0  q^-1]
[    0     0     0     0     0     0     0     0     0     0     0     0     0     1 -q^-1]
[    0     0     0     0     0     0     0     0     0     0     0     0     0     0     1]
```

And hence the canonical basis is:

$$
\begin{aligned}
C_1 &= g_1\\
C_2 &= g_2\\
C_3 &= g_3\\
C_4 &= g_4\\
C_5 &= g_5\\
C_6 &= q^{-1}g_1 - q^{-1}g_2 + g_6\\
C_7 &= q^{-1}g_1 - q^{-1}g_3 + g_7\\
C_8 &= q^{-1}g_3 - q^{-1}g_4 + g_8\\
C_9 &= -q^{-1}g_2 + q^{-1}g_5 + g_9\\
C_{10} &= -q^{-1}g_2 + q^{-1}g_4 + g_{10}\\
C_{11} &= q^{-1}g_3 - q^{-1}g_5 + g_{11}\\
C_{12} &= q^{-2}g_1 - q^{-2}g_3 + q^{-1}g_6 + q^{-1}g_7 - q^{-1}g_9 - q^{-1}g_{11} + g_{12}\\
C_{13} &= q^{-2}g_1 - q^{-2}g_3 + q^{-1}g_6 + q^{-1}g_7 - q^{-1}g_8 - q^{-1}g_{10} + g_{13}\\
C_{14} &= q^{-2}g_2 + q^{-2}g_3 - q^{-2}g_4 - q^{-2}g_5 + q^{-1}g_8 - q^{-1}g_9 - q^{-1}g_{10} + q^{-1}g_{11} + g_{14}\\
C_{15} &= q^{-3}g_1 - q^{-1}g_2 - q^{-3}g_3 + q^{-2}g_6 + q^{-2}g_7 - q^{-2}g_8 - q^{-2}g_{11} + q^{-1}g_{12} + q^{-1}g_{13} - q^{-1}g_{14} + g_{15}
\end{aligned}
$$

3.3 Cells

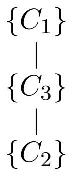
Theoretically, calculation of the left cells is easy: we fix an i and left multiply by C_j for all j keeping all of the resultant basis elements in a set S_i. Then we have $C_j \leq C_i$ for all $C_j \in S_i$. Repeating this process for all i yields the relations of the preorder and the cells can be calculated.

For example, in BMW$_2$ we have:

$$
\begin{aligned}
C_1C_1 &= C_1 & C_1C_2 &= C_2\\
C_2C_1 &= C_2 & C_2C_2 &= xC_2\\
C_3C_1 &= C_3 & C_3C_2 &= (q^{-1} - xq^{-1} + r^{-1})C_2
\end{aligned}
$$

$$
\begin{aligned}
C_1C_3 &= C_3\\
C_2C_3 &= (q^{-1} - xq^{-1} + r^{-1})C_2\\
C_3C_3 &= (q^{-2}x + ((-q^{-1} - q)r^{-1} + (-q^{-2} + 1)))C_2 + (q^{-1} + q)C_3
\end{aligned}
$$

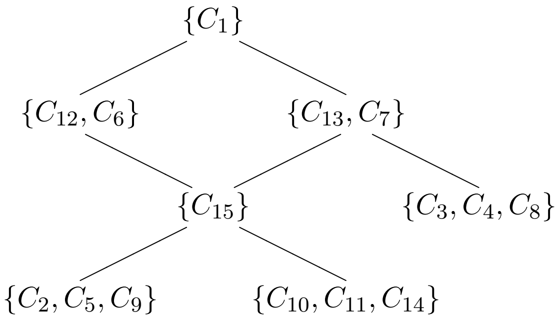
Hence $C_1, C_2, C_3 \leq_L C_1$, $C_2 \leq_L C_2$ and $C_2, C_3 \leq_L C_3$ and so the cells are $\{C_1\}$, $\{C_2\}$ and $\{C_3\}$. The Hasse diagram looks like:



However beyond BMW$_2$ the computation required is substantial—simply multiplying C_{14} and C_{15} in BMW$_3$ involves almost as much work as calculating the full table of the involution! For this reason an algorithm was written in MAGMA to perform and simplify multiplication within the algebra. This algorithm is outlined in Appendix B.2. Using this algorithm it is easy to calculate the left cells:

```
> CalcAllCellRelations("left");
Calculating LEFT cell relations...
 C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15 <= C1.
 C2 C5 C9 <= C2.
 C3 C4 C8 <= C3.
 C3 C4 C8 <= C4.
 C2 C5 C9 <= C5.
 C2 C5 C6 C9 C10 C11 C12 C14 C15 <= C6.
 C2 C3 C4 C5 C7 C8 C9 C10 C11 C13 C14 C15 <= C7.
 C3 C4 C8 <= C8.
 C2 C5 C9 <= C9.
 C10 C11 C14 <= C10.
 C10 C11 C14 <= C11.
 C2 C5 C6 C9 C10 C11 C12 C14 C15 <= C12.
 C2 C3 C4 C5 C7 C8 C9 C10 C11 C13 C14 C15 <= C13.
 C2 C10 C11 C14 <= C14.
 C2 C5 C9 C10 C11 C14 C15 <= C15.
```

Hence the cells are $\{C_1\}$, $\{C_2, C_5, C_9\}$, $\{C_3, C_4, C_8\}$, $\{C_6, C_{12}\}$, $\{C_7, C_{13}\}$, $\{C_{10}, C_{11}, C_{14}\}$ and $\{C_{15}\}$. The Hasse diagram looks like:

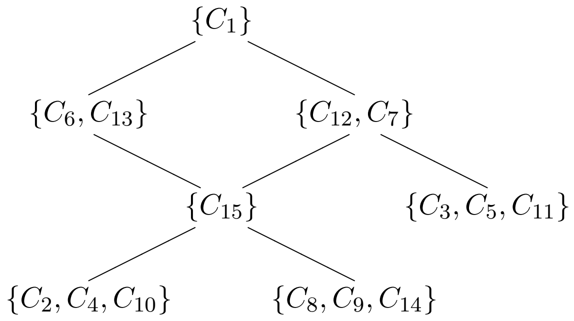


Similarly for the right cells:

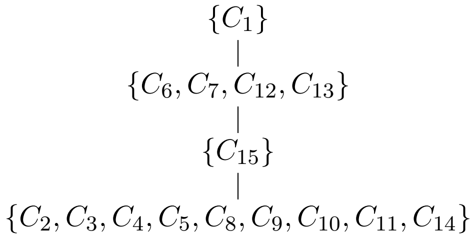
```
> CalcAllCellRelations("right");
Calculating RIGHT cell relations...
 C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15 <= C1.
 C2 C4 C10 <= C2.
```

```
C3 C5 C11 <= C3.
C2 C4 C10 <= C4.
C3 C5 C11 <= C5.
C2 C4 C6 C8 C9 C10 C13 C14 C15 <= C6.
C2 C3 C4 C5 C7 C8 C9 C10 C11 C12 C14 C15 <= C7.
C8 C9 C14 <= C8.
C8 C9 C14 <= C9.
C2 C4 C10 <= C10.
C3 C5 C11 <= C11.
C2 C3 C4 C5 C7 C8 C9 C10 C11 C12 C14 C15 <= C12.
C2 C4 C6 C8 C9 C10 C13 C14 C15 <= C13.
C8 C9 C14 <= C14.
C2 C4 C8 C9 C10 C14 C15 <= C15.
```

And so the Hasse diagram is:



It is now possible to calculate the two-sided cells. The Hasse diagram looks like:


$\{C_1\}$
$\{C_6, C_7, C_{12}, C_{13}\}$
$\{C_{15}\}$
$\{C_2, C_3, C_4, C_5, C_8, C_9, C_{10}, C_{11}, C_{14}\}$

A Avenues for Further Work

A.1 A Closer Look at the Tangle Monoid

The tangle monoid $\mathcal{T}_n$ was introduced to motivate and provide a general framework for the BMW monoid. It was stated in Section 1.2 that the tangle monoid is not finitely generated. It seems that this is quite difficult to show. In trying to show this some terms from ring theory were useful (this make sense since every ring is a monoid under multiplication). We say that $d \in \mathcal{T}_n$ is a *unit* if it is invertible and that $f, g \in \mathcal{T}_n$ are *associates* if $f = ug$ for some unit $u \in \mathcal{T}_n$. We also say that $f \in \mathcal{T}_n$ is *irreducible* if f is not a unit and if $f = gh$ implies that either g or h is a unit. The following lemma seems to provide an effective way to approach the problem:

Lemma A.1. *Let M be a monoid that contains an infinite set of irreducible elements such that no two are associates. Then M is not finitely generated.*

Proof. Let $\{r_1, r_2, \dots\}$ be the infinite set of irreducible elements and assume for contradiction that M is finitely generated on generators $\{u_1, u_2, \dots, u_m, g_1, g_2, \dots, g_n\}$ with the u_i units and the g_i non-units. Then each $r_i = ug_i$ for some unit or product of units u (because if $r_i = ug_ig_j$ then $r_i = (ug_i).g_j$ which contradicts r_i's irreducibility). There are only finitely many generators which implies that $r_i = ur_j$ for some i, j and unit u. But this contradicts the fact that none of the r_i are associates. Hence M is not finitely generated. □

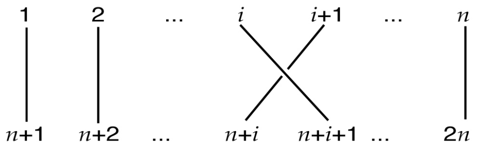
It is then a matter of showing that $\mathcal{T}_n$ has an infinite number of irreducible elements, no two of which are associates. In classifying the units the following seemed true (although it evaded a watertight proof):

Lemma A.2. *An element $d \in \mathcal{T}_n$ is invertible if an only if it is expressible as a product:*

$$d = \prod_{j=1}^{n} s_{i_j}^{p_j} \qquad p_j \in \mathbb{Z}$$

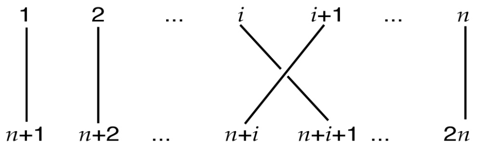
with:

$s_i =$



$i \in \{1, 2, \dots, (n-1)\}$

$s_i^{-1} =$



$i \in \{1, 2, \dots, (n-1)\}$

Idea for Proof. The implication one way is straightforward. If $d = \prod_{j=1}^{n} s_{i_j}^{p_j}$ then:

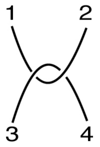
$$\begin{aligned} d(\prod_{j=n}^{1} s_{i_j}^{-p_j} &= \prod_{j=1}^{n} s_{i_j}^{p_j} (\prod_{j=n}^{1} s_{i_j}^{-p_j} \\ &= s_{i_1}^{p_1} s_{i_2}^{p_2} \dots s_{i_n}^{p_n} s_{i_n}^{-p_n} \dots s_{i_2}^{-p_2} s_{i_1}^{-p_1} \\ &= id \end{aligned}$$

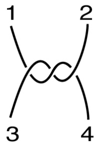
Hence d is invertible and $d^{-1} = \prod_{j=n}^{1} s_{i_j}^{-p_j}$. The implication the other way seems difficult. One important observation appears to be that if $d \in \mathcal{T}_n$ is invertible then there exists some sequence of Reidemeister moves which show that a nice picture of id is equivalent to d concatenated with d^{-1}. Hence as pictures we have:

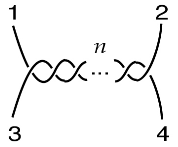
$$id_p \sim r_1 \sim r_2 \sim \cdots \sim r_n \sim d_p d_p^{-1}$$

Where $d_p d_p^{-1}$ denotes the concatenation of d and d^{-1} as pictures. So, to prove the result, we ned to show that any sequence of Reidemeister moves on id_p can be translated into a statement about s_i's and s_i^{-1}'s. For example the second Reidemeister move translates into the fact that $s_i s_i^{-1} = 1$. □

It seems that the following set is an infinite set of irreducible elements of $\mathcal{T}_2$, no two of which are associates:

$l_2 =$ 

$l_3 =$ 

$\dots$ $l_n =$ 

To see that they are all irreducible the following general argument appears useful. Assume that $l_i = gh$ for some $g, h \in \mathcal{T}_2$. Then there exists some sequence of Reidemeister moves that take $g_p h_p$ to l_i. But this implies that there exists a chain of Reidemeister moves that exhibits l_i as the concatenation of g_p and h_p. Now the double-intersection cannot be split and so must appear either above or below the central line in the concatenation. And so either g or h is a unit.

A generalisation appears possible for arbitrary n by considering analogues of l_3, l_5 etc. if n is odd and l_2, l_4 etc if n is even.

A.2 Generalisation of the Software for Arbitrary n

Due to time constraints the MAGMA software was written with $n = 3$ hardwired in the code. Most of the code wouldn't require that much work before it would work for arbitrary n. However some routines would provide some challenge. The two biggest challenges would be in rewriting the `StrandStep` routine (the current implementation just enumerates all possibilities) and in getting the basis of $g_i' s$ there initially (this is also hard-coded in the current implementation). Note also that the current software takes approximately 5 minutes to calculate all of the two sided cell-relations and so some degree of optimisation would be necessary even for $n = 4$.

A.3 A Rougher Order on B_n

Given two partial orders $\leq_1$ and $\leq_2$ we say that $\leq_1$ is *rougher* than $\leq_2$ if $a \leq_1 b$ implies $a \leq_2 b$ but there exists c, d such that $c \leq_2 d$ but $c \not\leq_1 d$. For a geometric interpretation one can imagine that the Hasse diagram of $\leq_1$ is equal to that of $\leq_2$ except with some lines removed.

The partial order used by Fishel and Grojnowski ([1]) is not very rough in that two Brauer diagrams d and e are comparable so long as $\ell(d) \neq \ell(e)$. Jie Du has suggested that it might be interesting to look at rougher orderings on B_n such that the BMW Basis Theorem and Theorem 2.2 still hold.

One rougher ordering which does not violate the theorems is to say the $d \leq e$ if it is possible to transform e into d by replacing a number of crossings in a nice diagram with horizontal or vertical loops. It would be interesting to see if this is the roughest possible ordering and whether a nice charaterisation of this ordering exists in terms of generators.

B Implementation of the Algorithms in MAGMA

B.1 Calculating the Canonical Basis

```
// Almost all work is done over the ring of Laurent polynomials Z[q,q^-1]
// so we initialise it here. In magma there is no way that I know of of
// implementing Laurent polynomails, so we work in the ring of Laurent power
// series.

ZQ<q> := LaurentSeriesRing(Integers());
ZZ<z> := PolynomialRing(Integers());

// Functions from ZZ to ZQ.

ZZ_to_ZQ := hom < ZZ -> ZQ | (q-q^-1) >;

Matrix_ZZ_to_ZQ := function (M);
    MQ := Matrix([[ ZZ_to_ZQ(M[i,j]) : j in [1..Ncols(M)] ] : i in [1..Nrows(M)] ]);
    return MQ;
end function;

// Calculates a canonical basis of an involution represented by a matrix M.

CalculateCanonicalBasis := function(M);
   error if Ncols(M) ne Nrows(M), "ERROR: Matrix must be square.";
   error if M*MatrixInvolution_q(M) ne IdentityMatrix(ZQ,Nrows(M)),
        "ERROR: The matrix must be satisfy M*Mbar = I.";
   n := Nrows(M);
   P := IdentityMatrix(ZQ,n);
   for i in [n-1..1 by -1] do
      for k in [i+1..n] do;
         Sum := &+[ M[i,j]*Involution_q(P[j,k]) : j in [i + 1..n]];
         Coeffs, low_power := Coefficients(Sum);
         if low_power eq 0 then P[i,k] := 0;
         else P[i,k] := &+[ q^(i)*Coeffs[i-low_power+1] : i in [low_power..-1]];
         end if;
      end for;
   end for;
   return P;
end function;
```

B.2 Calculating the Cells

The whole file is too long to include in full. The following initialisation of variables are ommited:

`g[i]` These are elements of the the standard trace basis on page 24 in terms of the s_i's and e_i's.

`c[i]` These are the elements of the canonical basis in terms of the s_i's and e_i' s.

The following functions are easy to implement:

`SimplifyMonoid` This function takes an element over BMW_3 and simplifies it using the simplification rules within the monoid. It is handy to use MAGMA ’s `RewriteMonoid` here.

`FullySimplify` This function takes a set of relations as well as an element of `BMW` and simplifies it as much as possible based on the relations and the `SimplifyMonoid` routine.

`MoveUpToG` Moves one step from BMW up to the polynomial ring $G[g_1, g_2, \ldots, g_{15}]$.

`MoveUpToC` Moves one step from G up to the polynomial ring $C[C_1, C_2, \ldots, C_{15}]$.

`CalcAllCellRelations` Goes through the algorithm described in Section 3.3.

```
// This file provides routines to compute the cells of BMW_n.

MQ<s1,S1,s2,S2,e1,e2> := Monoid < s1,S1,s2,S2,e1,e2 | s1*S1, S1*s1, s2*S2, S2*s2, s1*s2*s1=s2*s1*s2 >;

// Prepare the rewrite monoid...
RMQ<t1,T1,t2,T2,f1,f2> := RWSMonoid(MQ : MaxRelations := 100);

// Establish the monoid simplification function.
MQ_to_RMQ := func < x | RMQ!Eltseq(x) >;
RMQ_to_MQ := func < x |  MQ!Eltseq(x) >;
Simp := func < x | RMQ_to_MQ(MQ_to_RMQ(x)) >;

// Create the Algebra
Q<q> := LaurentSeriesRing(Integers());
R<r> := LaurentSeriesRing(Q);
A<x> := PolynomialRing(R);
BMW<a1,A1,a2,A2,b1,b2> := FreeAlgebra(A,MQ);

// Once we have everything we work in two polynomial rings which
// represent our work over the basis of traces and over the canonical basis.
// The structure lost by the fact that the polynomial ring is commutative
// doesn’t matter since we do all the necessary multiplication in BMW.
// The polynomial rings merely facilitate our expression of products in terms
// of the canonical bases. (To compute the cells).
// Trace basis ring:
G<g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15> := PolynomialRing(A,15);
// Canonical basis ring:
C<C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14,C15> := PolynomialRing(A,15);

// LOAD THE RELATIONS:

// Self Intersection:
SI := [];
SI[1] := a1*b1=r^-1*b1;
SI[2] := b1*a1=r^-1*b1;
// etc...[omitted]

// Double Intersection
DI := [];
DI[1] := a1^2 = 1 + (q-q^-1)*a1-(q-q^-1)*r^-1*b1;
// etc...[omitted]
// Does a single step within and element of the BMW monoid and reports:
// where it got to, whether it changed direction (ie through and ei) and
// whether it passed under a strand (this is generally bad).
StrandStep := function( Strand, Depth, Down, Element);

   // Make sure that we're not at the end of the line.
   if Down eq -1 then
      if Depth eq 1 then
         return Strand, Depth, Down, 1;
      end if;
   else
      if Depth eq (#Element+1) then
         return Strand, Depth, Down, 1;
      end if;
   end if;

   // Get the correct letter.
   seq := Eltseq(Element);
   if Down eq 1 then
      CurrentLetter := MQ!([seq[Depth]]);
   else
   CurrentLetter := MQ!([seq[Depth-1]]);
   end if;

   // Now go through _all_ possibilities (not very elegant programming!!!).
   CrossAbove := [];
   if Down eq 1 then
      case Strand:
         when 1:
            case CurrentLetter:
               when s1: Strand:=2; Depth+:=1; Down:=1; CrossAbove:=[Depth-1];
               when s2: Strand:=1; Depth+:=1; Down:=1; CrossAbove:=[Depth-1];
               when S1: Strand:=2; Depth+:=1; Down:=1; CrossAbove:=[Depth-1, Eltseq(s1)[1], Eltseq(e1)[1], -1];
               when S2: Strand:=1; Depth+:=1; Down:=1; CrossAbove:=[Depth-1];
               when e1: Strand:=2;         Down:=-1; CrossAbove:=[Depth];
               when e2: Strand:=1; Depth+:=1; Down:=1; CrossAbove:=[Depth-1];
               end case;
//...
// There was a lot of code here that has been omitted. It just goes through all the possibilities.
//...
   end case;
   end if;
   return Strand, Depth, Down, CrossAbove;
end function;

// Writes and element as T_d + \sum { \lambda_i e_i } where all the e_i are shorter than
// T_d. A very important routine. (d is in the Brauer monoid and T is the trace function).

TraceElement := function (Element)
   ProcElt := Element; Chaff := BMW!0;
   BlockArray := [0 : i in [1..#Element]];
   StrandArray := [0 : i in [1..6]];

   CurStartStrand := 1;
   Strand := CurStartStrand; Depth := 1; Dir := 1;

   StrandArray[CurStartStrand] := 1;

   repeat
      OldStrand := Strand;
      OldDepth := Depth;

      Strand, Depth, Dir, Cross := StrandStep(Strand, Depth, Dir, ProcElt);

      // Make sure that we're not at the end of a strand.
      if (OldStrand eq Strand) and (OldDepth eq Depth) then
         // Update the strand that we ended up on.
         if Dir eq -1 then // We've ended up the top.
            StrandArray[Strand] := 1;
         else // We've ended down the bottom.
            StrandArray[3+Strand] := 1;
         end if;
         // Get a new strand to start on and make sure there is one:
         CurStartStrand := Index(StrandArray,0);
         if CurStartStrand eq 0 then
            continue; // We've finished.
         end if;
         // Mark it as used:
         StrandArray[CurStartStrand] := 1;
         Strand := CurStartStrand;
         if Strand le 3 then
            Depth := 1;
            Dir := 1;
         else
            Depth := #Element+1;
            Strand := Strand - 3;
            Dir := -1;
         end if;

         continue;
      end if;

      // Did we cross under???
      // Reminder:
      // Cross[1] = depth at which under cross occurred.
      // Cross[2] = element that fixes the problem (should be an si or an Si).
      // Cross[3] = appropriate horizontal element for the relation.
      // Cross[4] = whether the relation appears in its normal or rotated form.
      if (#Cross ne 1) then // there's been an under crossing
         if (BlockArray[Cross[1]]) eq 0 then
            seq := Eltseq(ProcElt);
            Insert(~seq, Cross[1], Cross[1], [Cross[2]]);
            Chaff +:= Cross[4]*((q-q^-1)*BMW!(MQ!(Remove(seq,Cross[1])))-
               (q-q^-1)*BMW!(MQ!(Insert(seq,Cross[1],Cross[1],[Cross[3]]))));
            ProcElt := MQ!(seq);
            // Should this block be marked as OK??
            if OldStrand ne Strand then
               BlockArray[Cross[1]] := 1;
            end if;
         end if;
      else
         if OldStrand ne Strand then // We've crossed a crossing without going under. So we should mark it as OK.
            BlockArray[Cross[1]] := 1;
         end if;
      end if;

   until 0 notin BlockArray;
```

```
   return ProcElt, Chaff;

end function;

// This is the full simplification algorithm. Given an Element it reduces it until
// it is a product of traces of (not necessarily nice) diagrams in the Brauer monoid.
// It is sometimes necessary to give it further simplification.
TraceReduce := function (Element)
   input := Element; output := BMW!0;
   repeat
      input := FullySimplify(input, Rels);
      support := Support(input);
      CurElt := support[1];
      CurCoeff := MonomialCoefficient(input,CurElt);
      // Check that there are other elements in the support;
      if #support eq 1 then
         input := BMW!0;
      else
         input := &+[ MonomialCoefficient( input, support[i]) * BMW!(support[i]) : i in [2..#support] ];
      end if;
      Wheat, Chaff := TraceElement(CurElt);
      output +:= CurCoeff*BMW!(Wheat);
      input +:= CurCoeff*BMW!(Chaff);
   until input eq BMW!0;
   return output;
end function;
```

References

[1] F. Fishel and I. Grojnowski, *Canonical Bases for the Brauer Centralizer Algebra*, Mathematical Research Letters 2, 15-26 (1995).

[2] J. Du, B. Parshall and J. P. Wang, *GL_n: Quantum, Rational and Discreet*, unfinished.

[3] H. Barcelo and A. Ram, *Combinatorial Representation Theory*, in Math. Sci. Res. Inst. Publ., 38 , Cambridge Univ. Press, Cambridge, 1999, pp. 23–90.