Enumeration of Messy Knot Mosaics

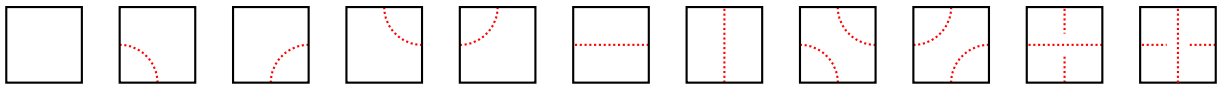
Jack Hanke **Richard Schank**
Northwestern University

Abstract

Lomonaco and Kauffman introduced a system of mosaics to model quantum knots. These systems of mosaics are composed of an $m \times n$ rectangular grid of 11 possible tiles. Oh and colleagues introduced a state matrix recursion method to exactly enumerate a subset of these mosaics that have the property of being suitably connected, which they call knot mosaics. We introduce and enumerate mosaics with the related property in which only some tiles must be suitably connected, which we call messy knot mosaics.

1 Introduction

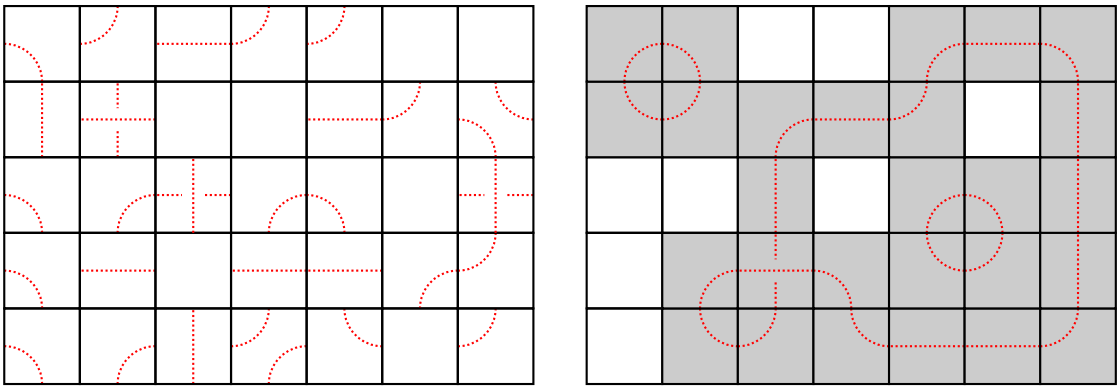
Lomonaco and Kauffman [3] introduced a model for quantum knots in which an $m \times n$ matrix is constructed using 11 distinct symbols called *tiles*. These tiles, diagrammed below, are composed of unit squares with dotted lines connecting 2 or 4 sides at their midpoint.



We denote the set of tiles $\mathbb{T} = \{T_0, \ldots, T_{10}\}$. A *mosaic* of size (m, n) is an $m \times n$ matrix made up of elements from $\mathbb{T}$. Figure 1a shows an example mosaic of size $(5, 7)$. We denote the set of all mosaics of size (m, n) as $\mathbb{M}^{(m,n)}$. As there are 11 elements in $\mathbb{T}$, there are 11^{mn} mosaics in $\mathbb{M}^{(m,n)}$. A *mosaic system* is then a subset of $\mathbb{M}^{(m,n)}$ with some property.

We are interested in mosaics with the property of being *suitably connected*, which is defined as follows. Consider an edge shared between two tiles in Figure 1a. The edge has either 0, 1, or 2 dotted lines drawn from its midpoint. Also note that the edges of the tiles on the boundary of the matrix are not shared by another tile. Therefore these edges only have 0 or 1 dotted lines drawn from their midpoint. A mosaic is suitably connected if all edges have 0 or 2 dotted lines drawn from their midpoint.

Lomonaco and Kauffman [3] call these *knot mosaics* because, other than the mosaic consisting of all T_0 tiles, the dotted lines form *knots*. Following the notation in [9] we denote the subset of mosaics of size (m, n) that are knot mosaics as $\mathbb{K}^{(m,n)}$. Figure 1b shows a knot mosaic of size $(5, 7)$ that contains 3 knots, with the tiles that make up the knots highlighted in gray. Note that a mosaic can contain knots isomorphic to the unknot, as well as knots that encompass other knots.



(a) A mosaic (b) A knot mosaic

Figure 1: Examples of mosaics of size $(5, 7)$ made of tiles in $\mathbb{T}$

Let $k_{m,n} = |\mathbb{K}^{(m,n)}|$ be the number of knot mosaics of size (m, n). First notice that if either m or n is 1, one can only construct a knot mosaic using T_0 tiles, so $k_{m,1} = k_{1,n} = 1$. Oh et al. [9] showed the following for $m, n \geq 2$.

Theorem 1.1 ([9]). *The number of knot mosaics of size (m, n) for $m, n \geq 2$ is $k_{m,n} = 2\,\|(X_{m-2} + O_{m-2})^{n-2}\|$, where X_{m-2} and O_{m-2} are $2^{m-2} \times 2^{m-2}$ matrices defined as*

$$X_{k+1} = \begin{bmatrix} X_k & O_k \\ O_k & X_k \end{bmatrix} \text{ and } O_{k+1} = \begin{bmatrix} O_k & X_k \\ X_k & 4O_k \end{bmatrix},$$

for $k = 0, 1, \ldots, m-3$, and $X_0 = O_0 = \begin{bmatrix} 1 \end{bmatrix}$. Here $\|N\|$ denotes the sum of elements of matrix N.

Oh and colleagues refer to these matrices X_k and O_k as *state matrices*. The authors utilize this state matrix recursion to bound the growth rate of knot mosaics $\delta = \lim_{n\to\infty} k_{n,n}^{\frac{1}{n^2}}$ [6, 8, 1], and Oh further adapts the method to solve problems in monomer and dimer tilings [5, 7]. An unexamined direction in this research program is modifying the suitably connected property. This motivates us to introduce *messy knot mosaics*.

2 Messy Knot Mosaics

Definition 2.1. A *messy knot mosaic* is a mosaic that contains at least one knot.

Figure 7 shows two examples of messy knot mosaic of size $(5, 7)$ that contains 3 knots[1], with the tiles that make up the knots highlighted in gray. All knot mosaics are messy knot mosaics.

[1]Certain permutations of $\{T_1, T_2, T_3, T_4\}$ and $\{T_7, T_8\}$ can make shapes that appear to be knots but have hanging connections, as seen in the $(0, 4)$ position in the right example in Figure 7. These are not considered knots by this paper and all referenced works.

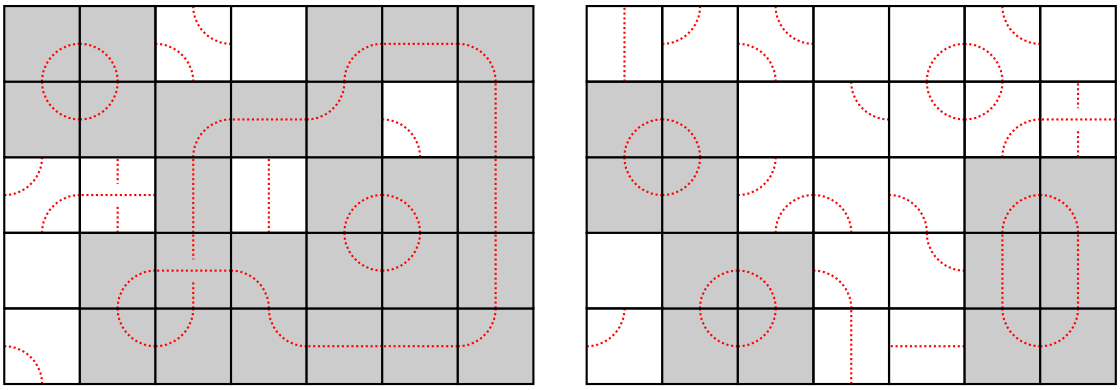


Figure 2: Messy knot mosaics

It turns out to be simpler to enumerate the number of mosaics that *do not* contain a knot. Therefore, let $\mathbb{S}^{(m,n)}$ be the set of mosaics that do not contain a knot, and let $|\mathbb{S}^{(m,n)}| = s_{m,n}$. Clearly the number of messy knot mosaics is then $11^{mn} - s_{m,n}$.

From the fact that the smallest knot is made of four tiles, shown in Figure 3, we can then conclude that $s_{n,1} = 11^n$, and $s_{2,2} = 11^4 - 1$. For $n, m \geq 2$, we first define the state matrices for messy knot mosaics.

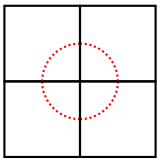


Figure 3: The smallest knot

Definition 2.2. Define $A(2) = \begin{bmatrix} 11^2 & 1 \\ -1 & 1 \end{bmatrix}$. We recursively define $A(k+1) \in \mathbb{N}^{2^k \times 2^k}$ given $A(k)$. Begin by writing $A(k) = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix}$, where the block matrices $A_{i,j}$ are square block matrices of size $2^{k-1} \times 2^{k-1}$. We then have

$$A(k+1) = \begin{bmatrix} 11A_{0,0} & \frac{1}{11}A_{0,0} & 11A_{0,1} & A_{0,1} \\ -\frac{1}{11}A_{0,0} & \frac{1}{11}A_{0,0} & 4A_{0,1} & A_{0,1} \\ 11A_{1,0} & -4A_{1,0} & 11A_{1,1} & A_{1,1} \\ A_{1,0} & -A_{1,0} & A_{1,1} & 11A_{1,1} \end{bmatrix}.$$

Construct $A(m)$ by starting with $k = 2$ and recursing until $k = m$.

Theorem 2.1. *The number of mosaics of size (m, n) that do not contain a knot is the $(0, 0)$ entry of $A(m)^n$.*

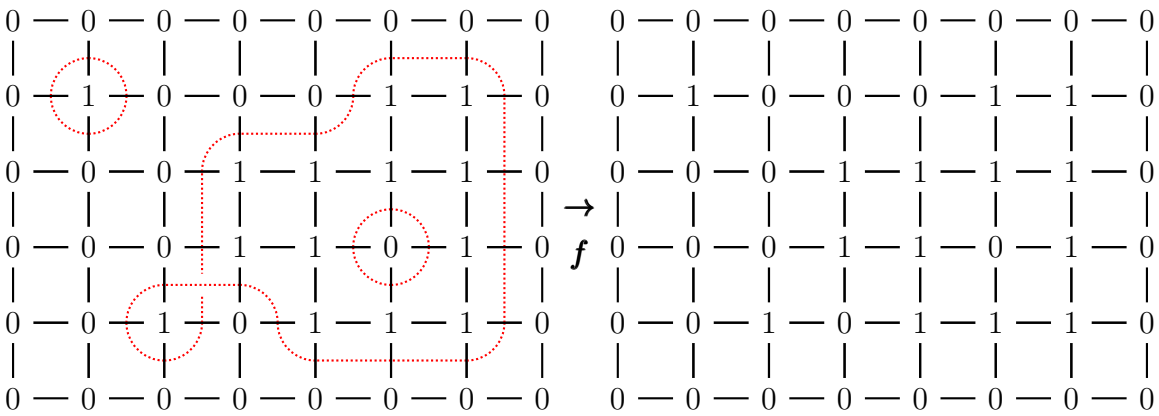


Figure 4: f applied to the left mosaic in Figure 7, resulting in a binary lattice

3 Preliminaries

We begin by defining a mapping f between $\mathbb{M}^{(m,n)}$ to a *binary lattice* of size (m, n). A binary lattice of size (m, n) is a rectangular lattice of $m+1$ by $n+1$ vertices, in which the boundary vertices are labeled 0, and the interior vertices are either 0 or 1. An example of a binary lattice of size $(5, 7)$ is shown on the right of Figure 4. Also let $\mathbb{L}^{(m,n)}$ be the set of all binary lattices of size (m, n), which gives $\left|\mathbb{L}^{(m,n)}\right| = 2^{(m-1)(n-1)}$.

Definition 3.1. $f : \mathbb{M}^{(m,n)} \to \mathbb{L}^{(m,n)}$ takes a mosaic and labels each vertex with the following rule. If the vertex is surrounded by an even number of knots (including 0 knots), label it 0. If the vertex is surrounded by an odd number of knots, label it 1. Removing the red dotted lines from the tiles gives the binary lattice.

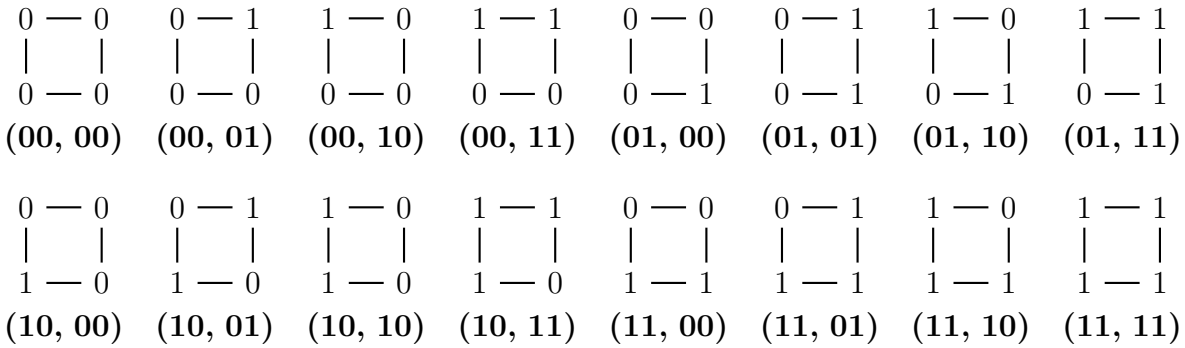
Similarly, define the *preimage* of a set L under f to be

$$f^{-1}(L) = \{m \in \mathbb{M}^{(m,n)} | f(m) \in L\}.$$

We want to compute $s_{m,n}$ by computing $f^{-1}(\ell)$ for each $\ell \in \mathbb{L}^{(m,n)}$, and then summing over all ℓ. We begin by finding a simple way to compute $f^{-1}(\ell)$ for a binary lattice ℓ by examining the structure of ℓ.

Definition 3.2. Let a *cell* be the portion of the binary lattice that an individual tile maps to, and let C be the set of unique cells.

For convenience, we give a pair of indexes to each of the $|C| = 2^4$ unique cells. The first index is the binary number formed by reading the bottom two vertices from left to right. The second index is the binary number formed by reading the top two vertices from left to right. Below is a diagram of all 2^4 cells with their indexes listed below.



Next let $u_{(i,j)}$ be the number of tiles in $\mathbb{T}$ that can map to cell (i, j). These values are simple to calculate, as each tile in $\mathbb{T}$ can only be part of a knot in certain ways. For example, $u_{(00,01)} = 1$, as cell $(00, 01)$ can only be formed by a mosaic with T_3 in that location. We can see that $u_{(01,10)} = 4$, as cell $(01, 10)$ can be from the T_7, T_8, T_9, or T_{10} tiles. Finally, we have $u_{(00,00)} = 11$, as any tile can fail to contribute to forming a knot. Table 1 summarizes the tiles in the preimage for each cell (i, j).

Cell (i, j)	**Preimage**	$\mathbf{u_{(i,j)}}$	**Cell (i, j)**	**Preimage**	$\mathbf{u_{(i,j)}}$
$(00, 00)$	$\mathbb{T}$	11	$(10, 00)$	T_1	1
$(00, 01)$	T_3	1	$(10, 01)$	T_7, T_8, T_9, T_{10}	4
$(00, 10)$	T_4	1	$(10, 10)$	T_6	1
$(00, 11)$	T_5	1	$(10, 11)$	T_2	1
$(01, 00)$	T_2	1	$(11, 00)$	T_5	1
$(01, 01)$	T_6	1	$(11, 01)$	T_4	1
$(01, 10)$	T_7, T_8, T_9, T_{10}	4	$(11, 10)$	T_1	1
$(01, 11)$	T_1	1	$(11, 11)$	$\mathbb{T}$	11

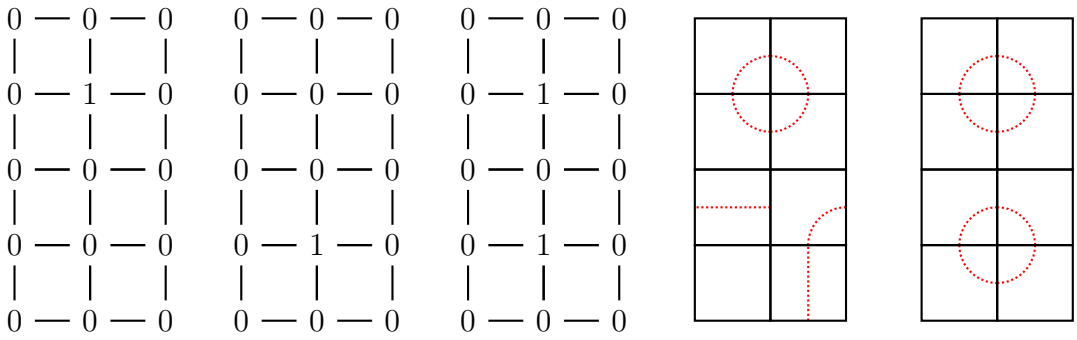
Table 1: Preimages of each unique cell

However, for some binary lattice ℓ the quantity

$$U(\ell) := \prod_{\text{Cell } (i,j) \in \ell} u_{(i,j)} \tag{1}$$

is not necessarily equal to $|f^{-1}(\{\ell\})|$, as $U(\ell)$ does not *just* count the number of mosaics that map to ℓ under f.

Example 3.1. Consider the following binary lattices for $s_{4,2}$.



$U(\ell_1)$ uniquely counts Mosaic 1, but both $U(\ell_1)$ and $U(\ell_2)$ count Mosaic 2, for which $f(\text{Mosaic 2}) = \ell_3$. This is because the bottom two rows of $(00, 00)$ cells in ℓ_1 could have come from 11 possible cells, though taken together 1 of those 11^4 permutations results in a new knot.

Definition 3.3. A knot is *specified* by a binary lattice ℓ if all mosaics in $f^{-1}(\{\ell\})$ contain the knot.

Example 3.2. From Example 3.1, ℓ_1 specifies the knot in the top 2 rows of Mosaic 1 and Mosaic 2, but not the knot in the bottom 2 rows of Mosaic 2.

Definition 3.4. Let $K : \mathbb{L}^{(m,n)} \to \mathbb{N}$ be the number of knots specified in ℓ.

Example 3.3. From Example 3.1, $K(\ell_1) = 1$, $K(\ell_2) = 1$, and $K(\ell_3) = 2$.

Definition 3.5. For a binary lattice ℓ, let $\mathbb{U}(\ell)$ be the set of binary lattices whose preimage mosaics under f are counted by $U(\ell)$. A binary lattice ℓ' is in $\mathbb{U}(\ell)$ if one can replace either some number of $(00, 00)$ or $(11, 11)$ cells in ℓ with other cells in C to create ℓ'. This corresponds with specifying new knots in the mosaics in the preimage of ℓ while leaving all knots specified by ℓ unchanged.

Example 3.4. From Example 3.1, $\mathbb{U}(\ell_1) = \{\ell_1, \ell_3\}$, $\mathbb{U}(\ell_2) = \{\ell_2, \ell_3\}$, and $\mathbb{U}(\ell_3) = \{\ell_3\}$.

From the definition of $\mathbb{U}$, we have

$$U(\ell) = \sum_{\ell' \in \mathbb{U}(\ell)} |f^{-1}(\ell')|. \tag{2}$$

Therefore $s_{m,n} \neq \sum_{\ell \in \mathbb{L}^{(m,n)}} U(\ell)$, as the sum overcounts mosaics for all ℓ that have $\mathbb{U}(\ell) \neq \{\ell\}$.

Definition 3.6. Let $\ell^* \in \mathbb{L}^{(m,n)}$ be the binary lattice made up of all $(00, 00)$ cells.

ℓ^* specifies 0 knots and has $U(\ell^*) = 11^{mn} = \left|\mathbb{M}^{(m,n)}\right|$, which clearly overcounts $s_{m,n}$. Also note that $\mathbb{U}(\ell^*) = \mathbb{L}^{(m,n)}$.

Proposition 3.1. *By regrouping terms we have*

$$\sum_{\ell\in\mathbb{L}^{(m,n)}} U(\ell) = \sum_{\ell\in\mathbb{L}^{(m,n)}} \sum_{\ell'\in\mathbb{U}(\ell)} |f^{-1}(\ell')| = \sum_{\ell\in\mathbb{L}^{(m,n)}} \left(\binom{K(\ell)}{0} + \cdots + \binom{K(\ell)}{K(\ell)} \right) |f^{-1}(\ell)|. \quad (3)$$

Proof. The first equality follows directly from Equation 2. For the second equality, notice that the number of times $|f^{-1}(\ell)|$ appears in the second sum of Equation 3 is the number of times a binary lattice ℓ appears in the set

$$\bigcup_{\ell\in\mathbb{L}^{(m,n)}} \{\mathbb{U}(\ell)\}.$$

If ℓ has $K(\ell) > 0$, the definition of $\mathbb{U}$ gives that ℓ appears once in the $\mathbb{U}$ set for the binary lattice that specifies 0 knots (ie. ℓ^*). ℓ appears in the $\mathbb{U}$ set for each binary lattice that specifies 1 of the knots in ℓ. ℓ also appears in the $\mathbb{U}$ set for each binary lattice that specifies 2 of the knots in ℓ, and so on up to specifying $K(\ell)$ knots. As the number of times each subset of knots are the binomial coefficients, this gives the second equality for all $\ell \neq \ell^*$.

If $\ell = \ell^*$, we have that $K(\ell^*) = 0$, so $|f^{-1}(\ell^*)|$ is only counted once. As $\binom{0}{0} = 1$, this completes the proof. □

Proposition 3.2. *The number of mosaics of size (m, n) that do not contain a knot $s_{m,n}$ has*

$$s_{m,n} = \sum_{\ell\in\mathbb{L}^{(m,n)}} (-1)^{K(\ell)} U(\ell). \quad (4)$$

Proof. By the binomial theorem,

$$0 = (1-1)^{K(\ell)} = \left(\binom{K(\ell)}{0} - \binom{K(\ell)}{1} + \cdots + (-1)^{K(\ell)} \binom{K(\ell)}{K(\ell)} \right),$$

so all terms in Equation 3 where $\ell \neq \ell^*$ are 0. Therefore,

$$\sum_{\ell\in\mathbb{L}^{(m,n)}} (-1)^{K(\ell)} U(\ell) = \binom{0}{0} |f^{-1}(\ell^*)| = s_{m,n}.$$

□

It is important to note here that a knot that contains tiles T_7 or T_8 can appear to be two separate knots which we consider as 1 knot.

Example 3.5. Figure 5 shows a mosaic that appears to have 2 knots, but we only consider as 1 knot. A rule of thumb that can be followed is if knots that contain T_7 or T_8 tiles can be replaced by T_9 or T_{10} tiles to form a single knot, then the original "knots" we consider as 1 knot.

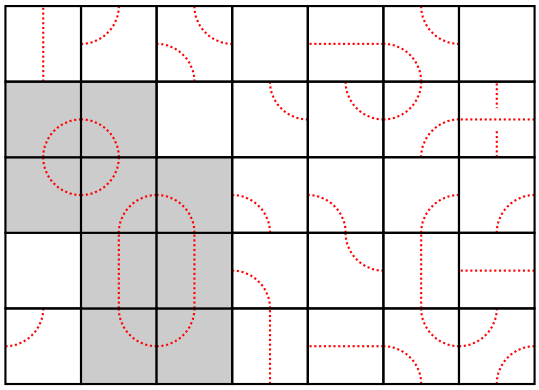


Figure 5: A messy knot mosaic with 1 knot

4 A Cell-Level Identity

Though Equation 4 does compute $s_{m,n}$, computing the number of knots specified in a binary lattice $K(\ell)$ requires examining the entire, global structure of ℓ. It will be more efficient to recover the $(-1)^{K(\ell)}$ term at the cell level. The idea is to add a coefficient $p_{(i,j)}$ to each value of $u_{(i,j)}$ so that the $(-1)^{K(\ell)}$ term is recovered.

Condition 4.1. *A set of cells $\mathcal{K}$ in a binary lattice ℓ meets this condition if*

$$\prod_{Cell\ (i,j)\in\mathcal{K}} p_{(i,j)} = -1, \tag{5}$$

with $p_{(i,j)} \in \{-1, 1\}$.

By definition of $K(\ell)$, if there exists values $p_{(i,j)}$ for which Condition 4.1 holds only for cells $\mathcal{K}$ that specify a knot, then the $(-1)^{K(\ell)}$ is recovered at the cell level.

Proposition 4.2. *There exists values $p_{(i,j)}$ for which Condition 4.1 holds for any set of cells that specify a knot.*

Proof. We can immediately see $p_{(00,00)} = p_{(11,11)} = 1$, as cells $(00, 00)$ and $(11, 11)$ can never be a member of cells that specify a knot, and so must be positive.

For the remaining values of $p_{(i,j)}$, we examine the cells that map to the smallest knot, shown in Figure 3.

$$\begin{array}{ccccc} 0 & - & 0 & - & 0 \\ | & & | & & | \\ 0 & - & 1 & - & 0 \\ | & & | & & | \\ 0 & - & 0 & - & 0 \end{array} \qquad\qquad \begin{array}{ccccc} 1 & - & 1 & - & 1 \\ | & & | & & | \\ 1 & - & 0 & - & 1 \\ | & & | & & | \\ 1 & - & 1 & - & 1 \end{array}$$

Figure 6: Portions of binary lattices associated with the smallest knot

Condition 4.1 amounts to the following two equations

$$\begin{aligned} p_{(00,01)}p_{(00,10)}p_{(01,00)}p_{(10,00)} &= -1 \\ p_{(11,10)}p_{(11,01)}p_{(10,11)}p_{(01,11)} &= -1, \end{aligned} \tag{6}$$

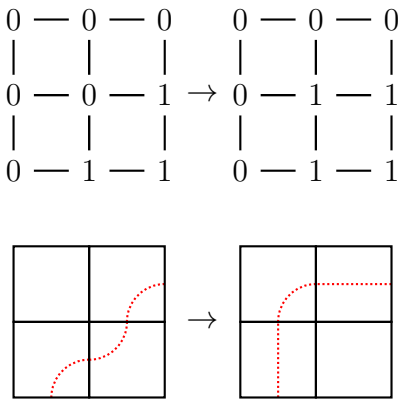
one for each portion of a binary lattice in Figure 6. We refer to the equations above as *constraints*, as they constrain the possible assignments of $p_{(i,j)}$. Let the constraints in Equation 6 be numbered 1 and 2.

Next note that in a binary lattice, a specified knot corresponds with a set of edge-and-corner-wise connected 0's or 1's, that if 0 are not edge-and-corner-wise connected to the boundary 0's. This is illustrated in Figure 4, where 3 knots correspond with 3 edge-and-corner connected regions, excluding the region that contains the boundary.

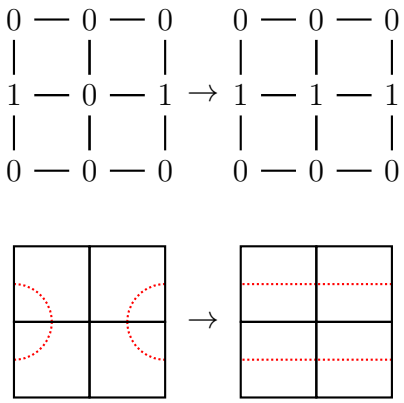
Let's suppose, without loss of generality, that the knot we are concerned with is specified by an edge-and-corner-wise connected set of 1's.

The idea is to describe all possible ways to build larger knots from smaller knots by changing a neighboring 0 to a 1. This flipping of the parity of the vertex corresponds with changing the identity of the four surrounding cells, and consequently the associated tiles. As the surrounding 8 vertices are held fixed, this creates 2^8 constraints to solve for. If $p_{(i,j)}$ exist that satisfy all $2^8 + 2$ constraints, then one can build any edge-and-corner-wise connected region (and therefore any knot) through some combination of the 2^8 possible flips to the region's neighbors. This gives that Condition 4.1 holds for all cells that specify a binary lattice.

However, there is a complication to this argument. Some of the 2^8 possible surrounding vertices may themselves not be edge-and-corner-wise connected, which affects whether we want the sign to be preserved or to change. Therefore, the flipping of parity of a single vertex results in 2 distinct types of constraints.



(a) Type 1 parity flip and associated tiles



(b) Type 2 parity flip and associated tiles

Figure 7: Parity flips for binary lattices

Let a constraint of *Type 1* be a parity flip that does not change the number of edge-and-corner-wise connected regions of 1s $\mod 2$. Therefore, we want the signs of the left and right hand side to be equal. For example, Figure 7a represents the constraint

$$p_{(00,00)}p_{(00,01)}p_{(00,01)}p_{(01,11)} = p_{(00,01)}p_{(00,11)}p_{(01,01)}p_{(01,11)}. \tag{7}$$

Now let a constraint of *Type 2* be a parity flip that does change the number of edge-and-corner-wise connected regions of 1s $\mod 2$. Therefore, we want the signs of the left and right hand side to change. For example, Figure 7b represents the constraint

$$p_{(00,10)}p_{(00,01)}p_{(10,00)}p_{(01,00)} = -p_{(00,11)}p_{(00,11)}p_{(11,00)}p_{(11,00)}. \tag{8}$$

In Figure 7b the transformation corresponds with *either* two distinct knot joining into one knot *or* one knot splitting into two distinct knot. In either case, we want the sign of the product to change.

After defining all 2^8 constraints, we combine them with the constraints in Equation **??**. We also assume that $p_{(00,01)}p_{(00,10)}p_{(01,00)} = 1$ and $p_{(10,00)} = -1$ is the solution for the first constrain in Equation 6.

We can then use software to verify that all $2^8 + 2$ constraints admit a solution.

□

If we let

$$v_{(i,j)} := p_{(i,j)}u_{(i,j)},$$

Proposition 4.2 allows us to rewrite the term in the sum of Equation 4 to

$$(-1)^{K(\ell)}U(\ell) = (-1)^{K(\ell)} \prod_{\text{Cell } (i,j)\in\ell} v_{(i,j)} = \prod_{\text{Cell } (i,j)\in\ell} p_{(i,j)}u_{(i,j)} \ \forall \ell. \tag{9}$$

We summarize the values of $v_{(i,j)}$ in Table 2.

Cell (i, j)	$\mathbf{u_{(i,j)}}$	$\mathbf{p_{(i,j)}}$	$\mathbf{v_{(i,j)}}$	**Cell (i, j)**	$\mathbf{u_{(i,j)}}$	$\mathbf{p_{(i,j)}}$	$\mathbf{v_{(i,j)}}$
$(00, 00)$	11	1	11	$(10, 00)$	1	-1	-1
$(00, 01)$	1	1	1	$(10, 01)$	4	1	4
$(00, 10)$	1	1	1	$(10, 10)$	1	1	1
$(00, 11)$	1	1	1	$(10, 11)$	1	1	1
$(01, 00)$	1	1	1	$(11, 00)$	1	1	1
$(01, 01)$	1	1	1	$(11, 01)$	1	1	1
$(01, 10)$	4	-1	-4	$(11, 10)$	1	-1	-1
$(01, 11)$	1	1	1	$(11, 11)$	11	1	11

Table 2: Values of $p_{(i,j)}$ and $v_{(i,j)}$

Equation 9 allows us to simplify Equation 4 to

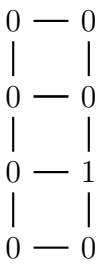
$$s_{m,n} = \sum_{\ell\in\mathbb{L}^{(m,n)}} \prod_{\text{Cell } (i,j)\in\ell} v_{(i,j)}. \tag{10}$$

$s_{m,n}$ can be calculated more efficiently than in Equation 10 using the state matrix recursion introduced in Theorem 1.1.

5 Proof of Theorem 2.1

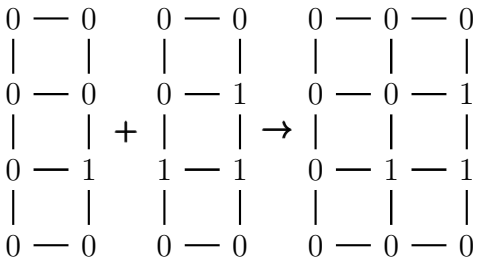
TODO induction and definition for state matrix

Our solution accomplishes this by building $m \times n$ vertex labelings for fixed m using vertex labelings of $m \times 1$ columns of cells. These columns have vertex labelings such that the top and bottom edge all labeled 0. One of these columns for $m = 3$ is below.



It is useful to define an index for vertex labelings of these $m \times 1$ columns. To do this, consider reading a column of vertex labelings from bottom to top, ignoring the first and last 0's. If we interpret these sequences for the left and right column as binary numbers b_{left} and b_{right}, the index in base ten is the pair (b_{left}, b_{right}). For example, for the above $m = 3$ column, we have sequences 00 for the left column and 10 for the right column, so the index is $(0, 2)$.

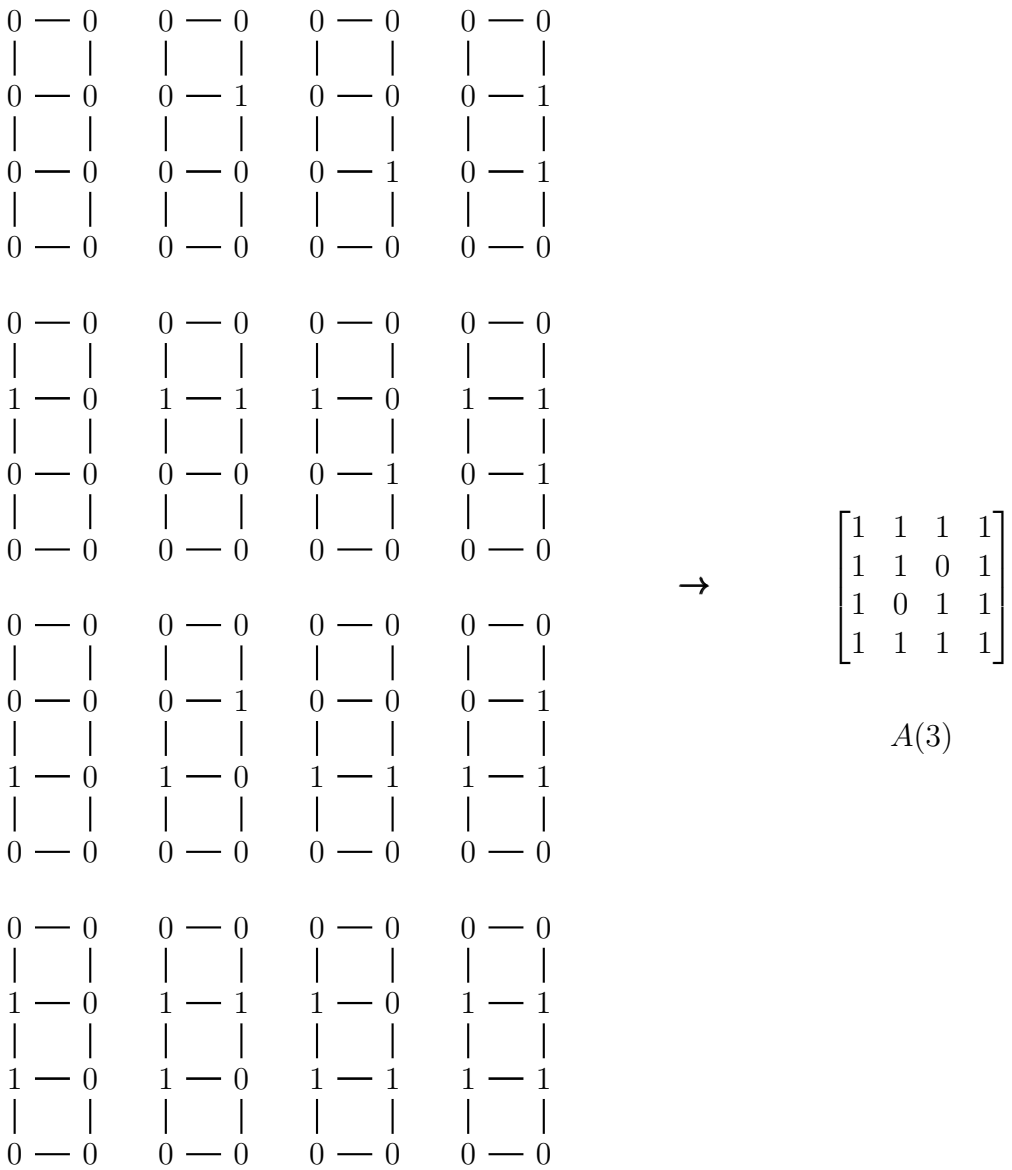
Next consider a column with index $(0, b_1)$ for some $b_1 \in [0, 2^{m-1} - 1]$. For reasons we will see later, let's call this the *starting column*. Now consider appending a column with index (b_1, b_2) for some $b_2 \in [0, 2^{m-1} - 1]$. For example, below is a diagram for appending our starting column $(0, 2)$ with column $(2, 3)$.



Notice that in the above example we have created a vertex labeling for an $m \times 2$ grid of cells, in which the left-most vertex column labels are all 0 (left index of 0). Also note that we did not create either illegal cell labelings with column $(2, 3)$. However, we could append the column $(2, 1)$, which would create an illegal cell labeling.

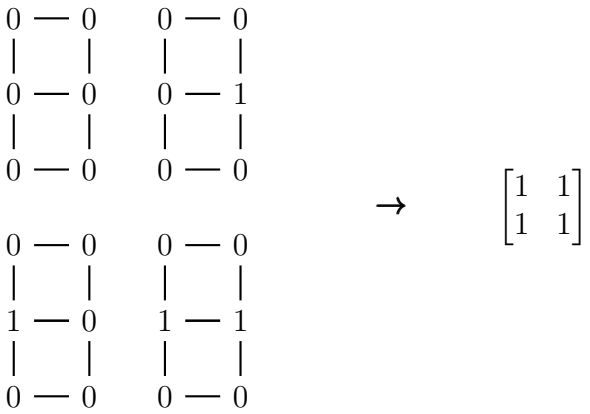
Finally, if we further appended a column with label $(b_2, 0)$, we would have created a vertex labeling with a boundary of all 0's. For this reason, let's call a column with index $(b, 0)$ for $b \in [0, 2^{m-1} - 1]$ an *ending column*. This vertex labeling would correspond with 1 polygon mosaic if the middle column had index $(2, 3)$, but not if the middle column had index $(2, 1)$.

This motivates the creation of a matrix $A(m)$ to every column index (i, j), where $A(m)_{i,j} = 1$ if the column labeling corresponds with a legal polygon mosaic, and 0 otherwise. For example, the $A(3)$ matrix corresponds with the following labeled columns.

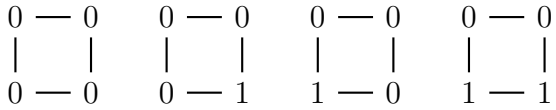


$A(m)$ has the property that the 0-th row represets all starting columns, and the 0-th column represents all ending columns. Even more importantly, notice that $A(m)^2_{i,j}$ represents the number of $m \times 2$ grids with left-most index i and right-most index j that correspond with a legal polygon mosaic. In general, $(A(m)^n)_{i,j}$ represents this quantity for an $m \times n$ grid of cells, and so if we know $A(m)$ for some m, then $(A(m)^n)_{0,0} = p_{m,n}$.

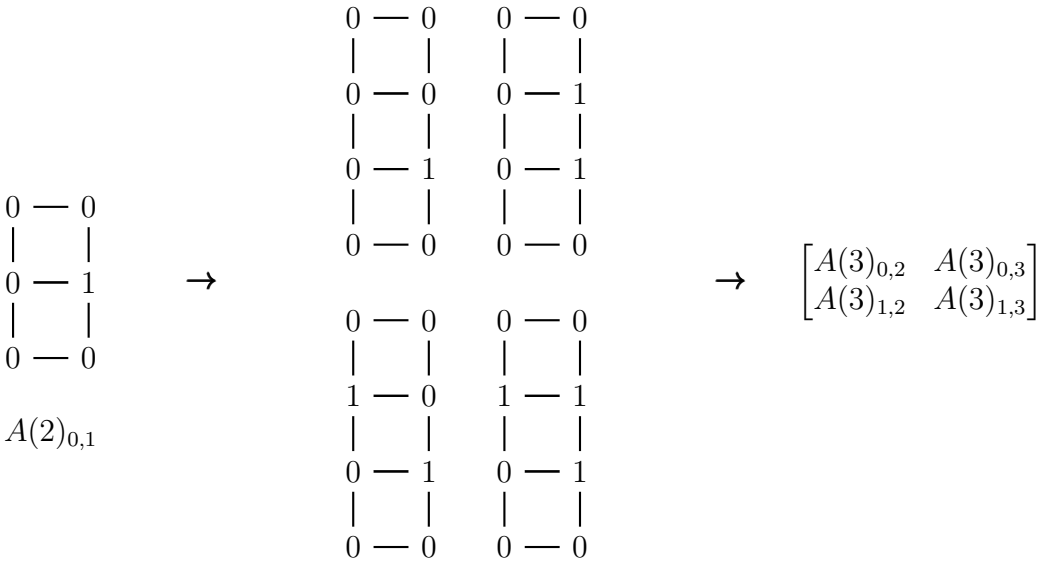
The final component of the proof is constructing $A(m)$ for any m. Begin by calculating $A(2)$ by identifying the number of legal polygon mosaics that correspond with each vertex coloring index $\{(0,0),(0,1),(1,0),(1,1)\}$, like so.



Next consider an arbitrary value $A(k)_{i,j}$ for any $k \geq 2$. This value is 1 if the $k \times 1$ column with index (i, j) can be part of a polygon mosaic, and 0 otherwise. We can determine that specific values of $A(k+1)$ are multiples of $A(k)_{i,j}$ by considering the following operation on an arbitrary column with index (i, j). Copy the column four times and replace the top two 0's of each column with the bottom row of labels from one of the four cell labelings below.



For example, for the $m = 2$ column with index $(0, 1)$, this operation looks like the following.



This operation results in 4 new columns that are represented in $A(k+1)$. In our example, specifically we get the following.

$$\begin{bmatrix} A(3)_{0,2} & A(3)_{0,3} \\ A(3)_{1,2} & A(3)_{1,3} \end{bmatrix} = \begin{bmatrix} 1A(2)_{i,j} & 1A(2)_{i,j} \\ 0A(2)_{i,j} & 1A(2)_{i,j} \end{bmatrix},$$

Critically, this transformation *only* changes the identity of the top two tiles. This implies that the same value coefficients computed by comparing $A(2)$ and $A(3)$ can be used for any $m \times 1$ column, as long as both column indices (i, j) are congruent mod 2. Furthermore, if one writes $A(k)$ as the block matrix

$$A(k) = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix},$$

where $A_{\hat{i},\hat{j}} \in \mathbb{R}^{2^{k-2} \times 2^{k-2}}$, then all column's represented in $A_{\hat{i},\hat{j}}$ have indices $(i, j) \equiv (\hat{i}, \hat{j})$ mod 2. This allows us to write that in general, if $A(k) = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix}$, then

$$\begin{bmatrix} V_{0,0}A_{0,0} & V_{0,1}A_{0,0} & V_{0,2}A_{0,1} & V_{0,3}A_{0,1} \\ V_{1,0}A_{0,0} & V_{1,1}A_{0,0} & V_{1,2}A_{0,1} & V_{1,3}A_{0,1} \\ V_{2,0}A_{1,0} & V_{2,1}A_{1,0} & V_{2,2}A_{1,1} & V_{2,3}A_{1,1} \\ V_{3,0}A_{1,0} & V_{3,1}A_{1,0} & V_{3,2}A_{1,1} & V_{3,3}A_{1,1} \end{bmatrix}. \tag{11}$$

where $V \in \mathbb{R}^{4\times 4}$ can be found after directly computing $A(2)$ and $A(3)$, then solving the following equation.

$$\begin{bmatrix} A(3)_{0,0} & A(3)_{0,1} & A(3)_{0,2} & A(3)_{0,3} \\ A(3)_{1,0} & A(3)_{1,1} & A(3)_{1,2} & A(3)_{1,3} \\ A(3)_{2,0} & A(3)_{2,1} & A(3)_{2,2} & A(3)_{2,3} \\ A(3)_{3,0} & A(3)_{3,1} & A(3)_{3,2} & A(3)_{3,3} \end{bmatrix} = \begin{bmatrix} V_{0,0}A(2)_{0,0} & V_{0,1}A(2)_{0,0} & V_{0,2}A(2)_{0,1} & V_{0,3}A(2)_{0,1} \\ V_{1,0}A(2)_{0,0} & V_{1,1}A(2)_{0,0} & V_{1,2}A(2)_{0,1} & V_{1,3}A(2)_{0,1} \\ V_{2,0}A(2)_{1,0} & V_{2,1}A(2)_{1,0} & V_{2,2}A(2)_{1,1} & V_{2,3}A(2)_{1,1} \\ V_{3,0}A(2)_{1,0} & V_{3,1}A(2)_{1,0} & V_{3,2}A(2)_{1,1} & V_{3,3}A(2)_{1,1} \end{bmatrix}$$

This is solved by

$$V = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

which completes the proof.

The method detailed in Theorem **??** generalizes to other tile sets, which we tabularize in Section **??** without proof. Interestingly, the method not only generalizes to other tile sets, but can also be augmented to enumerate the more complicated "messy" polygon mosaics.

As demonstrated in the proof of Theorem 2.1, the enumeration of both polygon mosaics and mosaics that do not contain a polygon share the same structure, and only differ by the identity of the matrices $A(2), V$. We summarize these matrices for various collections of tiles below.

TODO induction for matrix powers

TODO complexity

Equation 10 has $mn2^{(m-1)(n-1)}$ multiplications.

Theorem 2.1 uses $4^2 + 4^3 + 4^4 + 4^{m-1} = 4^2\frac{4^{m-2}-1}{4-1}$ multiplications for the construction of $A(m)$, and using the naive matrix multiplication algorithm we have $2^{3(m-1)} \lceil \log_2(n) \rceil$ multiplications to compute $A(m)^n$.

If we set $m = n$, calculating $s_{m,m}$ with Equation 10 is $O(m^2 2^{m^2})$, while Theorem 2.1 is $O(\log(m)8^m)$.

6 Extensions

Hong and Oh [2] study the mosaic system with the tile set $\mathbb{T}^* = \{T_0, \ldots, T_7\}$. This tile set constructs shapes we call *polygons*[2]. If we let $p_{m,n}$ be the number of polygon mosaics of size (m, n), Hong and Oh showed the following results[3].

Theorem 6.1 ([2])**.** *The number of polygon mosaics of size* (m, n) $p_{m,n}$ *for* $m, n \geq 2$ *has*

$$2^{m+n-3}\left(\frac{17}{10}\right)^{(m-2)(n-2)} \leq p_{m,n} \leq 2^{m+n-3}\left(\frac{31}{16}\right)^{(m-2)(n-2)}.$$

Though not stated in Hong and Oh [2], $p_{m,n}$ is exactly enumerated by Theorem 1.1 by replacing the 4 in the definition of O_{k+1} with a 0.The array $p_{n,m}$ is A181245 on the OEIS [4, OEIS].

TODO

7 Acknowledgements

The authors would like to thank Michael Maltenfort for the edits, improvements and ideas for this paper.

[2]Polygons are more commonly called "self-avoiding polygons" in the literature to emphasize their relationship with self-avoiding walks.

[3]The authors did not consider the mosaic containing all T_0 tiles a polygon mosaic, and so define $p_{m,n}$ as one less than what we define.

References

[1] Dooho Choi et al. "Quantum knot mosaics and bounds of the growth constant". In: *Reviews in Mathematical Physics* 36.10 (2024), p. 2450025. DOI: 10.1142/S0129055X24500259. eprint: https://doi.org/10.1142/S0129055X24500259. URL: https://doi.org/10.1142/S0129055X24500259.

[2] Kyungpyo Hong and Seungsang Oh. "Bounds on Multiple Self-avoiding Polygons". In: *Canadian Mathematical Bulletin* 61.3 (Sept. 2018), pp. 518–530. ISSN: 1496-4287. DOI: 10.4153/cmb-2017-072-x. URL: http://dx.doi.org/10.4153/CMB-2017-072-x.

[3] Samuel J. Lomonaco and Louis H. Kauffman. "Quantum knots and mosaics". In: *Quantum Information Processing* 7.2 (2008), pp. 85–115. DOI: 10.1007/s11128-008-0076-7. URL: https://doi.org/10.1007/s11128-008-0076-7.

[4] OEIS Foundation Inc. *The On-Line Encyclopedia of Integer Sequences*. Published electronically at http://oeis.org.

[5] Seungsang Oh. "Domino tilings of the expanded Aztec diamond". In: *DISCRETE MATHEMATICS* 341.4 (Apr. 2018), pp. 1185–1191. ISSN: 0012-365X. DOI: 10.1016/j.disc.2017.10.016.

[6] Seungsang Oh. "Quantum knot mosaics and the growth constant". In: *Topology and its Applications* 210 (2016), pp. 311–316. ISSN: 0166-8641. DOI: https://doi.org/10.1016/j.topol.2016.08.011. URL: https://www.sciencedirect.com/science/article/pii/S0166864116301857.

[7] Seungsang Oh. "State matrix recursion method and monomer-dimer problem". In: *DISCRETE MATHEMATICS* 342.5 (May 2019), pp. 1434–1445. ISSN: 0012-365X. DOI: 10.1016/j.disc.2019.01.022.

[8] Seungsang Oh and Youngin Kim. "Growth rate of quantum knot mosaics". In: *Quantum Information Processing* 18.8 (2019), p. 238. DOI: 10.1007/s11128-019-2353-z. URL: https://doi.org/10.1007/s11128-019-2353-z.

[9] Seungsang Oh et al. "Quantum knots and the number of knot mosaics". In: *Quantum Information Processing* 14.3 (2015), pp. 801–811. DOI: 10.1007/s11128-014-0895-7. URL: https://doi.org/10.1007/s11128-014-0895-7.

8 Appendix

TODO