POWERSFUL BIJECTIONS FOR PLANAR MAPS AND LATTICE WALKS

WEI CAI

ABSTRACT. The theory of the mating-of-trees that encodes graphs as walks is powerful in allowing one to reduce problems about Schramm-Loewner evolution (SLE) and Liouville quantum gravity (LQG), to problems about Brownian motions that are simpler and more well-studied. In this expository paper, we explore the theory of mating-of-trees bijection for random planar maps and their lattice walk analogs in \( \mathbb{Z}^2 \). In particular, we study the Mullin bijection between spanning-tree-decorated planar maps and random walk, as well as the bijection between site-percolated loopless triangulations and the Kreweras walk.

CONTENTS

1. Introduction 1
2. Spanning-tree-decorated planar maps 3
  2.1. Walking around a spanning tree 4
  2.2. Parenthesis systems and random walk in 2D 4
  2.3. Mullin’s Bijection 6
3. Site-percolated loopless triangulations 7
  3.1. Site-percolations on triangulations 8
  3.2. Depth First Search 9
  3.3. The Kreweras walk 12
  3.4. The bijection 13
4. Extension: Depth First Search Algorithm 16
Acknowledgments 17
References 17

1. INTRODUCTION

Interest in the problem of the enumerative theory of planar maps stems from the early sixties with William T. Tutte’s work on the enumeration of planar triangulations. Over the past years it has seen development in combinatorics, statistical physics, quantum gravity, enumerative topology and probability theory, which have started to interact intensely in the last decade.

Deriving closed-form formulas for the number of certain types of planar maps is difficult, but counting the corresponding lattice walks is often an easier and better-studied problem. It is also much easier to sample a walk using a computer than to sample a planar map. The problem hence is simplified into finding the right
random walk to construct a bijection from the planar maps. We will first develop some fundamental concepts about the definitions of maps and then we will move to investigate the connection between planar maps and lattice walks.

**Definition 1.1.** A **planar map** $M$ is a proper embedding of a connected graph $G = (V, E)$ into the two-dimensional, oriented sphere $S$, considered up to orientation preserving homeomorphisms of the sphere $S$, meaning the vertices are represented by distinct points and the edges are represented by arcs that only intersect at their endpoints and agree with the incidence relation of $G$.

There can be loops and multiple edges. Breaking an edge at the midpoint results in two half-edges, each incident to one of the endpoints. A **corner** is the angular sector delimited by two consecutive edges around a vertex. Each corner $c$ is incident to a vertex $v(c)$, to a face $f(c)$, and to two edges: in counterclockwise direction around $v(c)$, let $cw(c)$ denote the edge preceding $c$ and $ccw(c)$ denote the edge following $c$. The **degree** of a vertex or face is the number of incident corners. A map is a **$m$-angulation** if all its faces have degree $m$, and in the special cases $m = 3, 4$, we call the 3-angulations triangulations and 4-angulations quadrangulations.

The faces of a proper embedding $M$ of a connected graph $G$ in $S$ are simply connected. In particular the number of vertices $v(M)$, the number of faces $f(M)$ and the number of edges $e(M)$ of a planar map $M$ follow Euler’s formula:

$$v(M) + f(M) = e(M) + 2.$$  

In the context of this paper, we don’t need to distinguish between the abstract graph $G$ (the collection of vertices and edges) and the embedding $M$ of the graph. We will use $M$ in the rest of the paper to refer to the planar map.

We now discuss the fundamental idea of **duality**.

**Definition 1.2.** The **dual** of a planar map $M$, denoted $M^*$, is the map obtained by drawing a vertex $f^*$ of $M^*$ in each face $f$ of $M$ and drawing an edge $e^*$ of $M^*$ across each edge $e$ of $M$.

By construction, each face of $M^*$ then contains exactly one vertex of $M$. The superimposition of a map $M$ and its dual $M^*$ (with vertices created at the at the intersection of an edge $e$ of $M$ with its dual $e^*$) is a quadrangulation $\Delta(M)$ called the **derived map** of $M$. Faces of $\Delta(M)$ are in one-to-one correspondence with corners of $M$. An example of a dual map and a derived map is given in Figure 1.

**Theorem 1.3.** Duality is an involution on the set of planar maps. It preserves the number of edges, and exchanges the numbers of vertices and faces: that is, $M^{**} = M$, $e(M^*) = e(M)$, and $v(M^*) = f(M)$.

To better visualize planar maps, the notions of rooted maps and orientation naturally arise. We want to choose a point $x_0$ of $S$ in a face of $M$ and identify the punctured sphere $S^2 \setminus \{x_0\}$ with the plane, sending $x_0$ to infinity. In such a representation, all faces are homeomorphic to discs, except for the **exterior** or **outer** face containing $x_0$. Depending on the choice of $x_0$ we a priori get different drawings, but up to orientation-preserving homeomorphisms of the plane, only the choice of the face in which $x_0$ is chosen matters. Accordingly, let a **plane map** $(M, f)$ be a planar map $M$ with a marked face $f$, so that plane maps are in one-to-one
correspondence with equivalence classes of proper embeddings of connected graphs in the plane up to homeomorphisms of the oriented plane.

**Definition 1.4.** Let a rooted planer map \((M, c)\) be a planar map with a marked corner \(c\). The root face, root vertex and root edge of \((M, c)\) are then defined to be \(f(c), v(c), \) and \(ccw(c)\).

**Definition 1.5.** A tree is a connected acyclic graph and a rooted tree is a tree with a vertex distinguished as the root vertex. A vertex \(v\) is an ancestor of another vertex \(v'\) in a tree \(T\) if \(v\) is on the (unique) path in \(T\) from \(v'\) to the root vertex of \(T\). When \(v\) is the first vertex encountered on that path, it is the father of \(v'\). A leaf is a vertex which is not a father.

**Definition 1.6.** Given a rooted planar map \(M\), a spanning tree of \(M\) is a subset \(T\) of the set of edges of \(M\) that forms a tree and that is incident to every vertex of \(M\) (contains every vertex of \(M\)). A spanning tree inherits its root from the map. A tree-rooted map \((M, T, c)\) is thus a rooted map \(M\) together with a distinguished spanning tree \(T\).

If \(M\) is rooted with root edge \(e_0\), then the dual map \(M^*\) is rooted with root edge \(e_0^*\) oriented from the right of \(e_0\) to the left of \(e_0\). If \(T\) is a spanning tree of \(M\), the dual tree is the spanning tree \(T^*\) of \(M^*\) consisting of the dual of the edges of \(M\) not in \(T\). \(T^*\) is a spanning tree of \(M^*\). Indeed, if \(T\) is connected, then \(T^*\) is acyclic and vice versa.

**Theorem 1.7.** Let \((T_1, T_2)\) be a partition of the edges of a planar map \(M\). Then \(T_1\) is a spanning tree of \(M\) if and only if \(T^*_2\) is a spanning tree of \(M^*\).

The trees \(T_1\) and \(T_2\) are called dual spanning trees (Figure 2) . It should be noted that the edges of \(T^*_2\) are not the duals of the edges of \(T_1\), but rather the duals of the edges not in \(T_1\). Here, the spanning tree and the corresponding dual spanning tree being mated together is a vivid illustration of the name of the theory “mating of trees”.

2. *Spanning-tree-decorated planar maps*

Tree-rooted maps commonly arise in counting problems. To further build some intuition about standard combinatorics of planar maps, we will start with the Mullin Bijection.
2.1. Walking around a spanning tree. Let \((M, T, c)\) be a tree-rooted planar map with \(n\) edges. The counterclockwise walk around \((T_1, c)\) starting and finishing at the root induces a total order, the order of appearance, on the vertex set and on the edge set of the tree by traveling on the border of the tree. Each edge is visited twice during the walk and four symbols can be used to record the four distinct types of moves during the walk:

- \(\uparrow\) first time following an edge of the spanning tree
- \(\downarrow\) second time following an edge of the spanning tree
- \(\rightarrow\) first time crossing an edge not in the spanning tree
- \(\leftarrow\) second time crossing an edge not in the spanning tree

We call this walk a counterclockwise contour code. An example is given in Figure 3. The transition between corners are: \(1 \rightarrow 2 \uparrow \rightarrow 4 \uparrow 5 \rightarrow 6 \uparrow 7 \uparrow 8 \leftarrow 9 \leftarrow 10 \leftarrow 11 \rightarrow 12 \rightarrow 13 \downarrow 14 \leftarrow 15 \uparrow 16 \leftarrow 17 \rightarrow 18 \leftarrow 19 \rightarrow 20 \downarrow 21 \rightarrow 22 \rightarrow 23 \downarrow 24 \leftarrow 25 \downarrow 26 \rightarrow 27 \downarrow 28 \leftarrow\), and thus the contour code is:

\(\rightarrow \uparrow \rightarrow \uparrow \uparrow \leftarrow \leftarrow \rightarrow \rightarrow \downarrow \rightarrow \rightarrow \rightarrow \rightarrow \downarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \righta
prefixes $w'$, $|w'|_a > |w'|_\bar{a}$. For instance, the word $aa\bar{a}a\bar{a}$ is a parenthesis system. A shuffle of two parenthesis systems (or parenthesis shuffle for short), is a word on the alphabet $\{a, \bar{a}, b, \bar{b}\}$ such that the subword of $w$ consisting of letters in $\{a, \bar{a}\}$ and the subword consisting of letters in $\{b, \bar{b}\}$ are parenthesis systems respectively. An example is the word $abb\bar{a}b\bar{a}a\bar{b}a\bar{b}a\bar{a}a\bar{b}$.

Shuffles of parenthesis systems can also be mapped to walks in the first quadrant. We consider two-dimensional walks with four distinct non-zero straight steps: $\{(1, 0), (-1, 0), (0, 1), (0, -1)\}$. We also impose the restriction that the start and end point of the walk must be the origin and the walk should remain in the quarter plane $\{(x, y) \in \mathbb{R}^2 : x, y \geq 0\}$ to make it correspond to the shuffle of parenthesis system. The correspondence can be obtained by considering each letter $a$ as a $(1, 0)$ step, $\bar{a}$ as a $(-1, 0)$ step, $b$ as a $(0, 1)$ step and $\bar{b}$ as a $(0, -1)$ step. For instance, we represent the word $abb\bar{a}b\bar{a}a\bar{b}a\bar{b}a\bar{a}a\bar{b}$ in Figure 4.

**Figure 4.** The random walk corresponding to the word $abb\bar{a}b\bar{a}a\bar{b}$.

**Corollary 2.1.** The number of parenthesis systems of size $n$ (total length $2n$) is the $n^{th}$ Catalan number $C_n = \frac{1}{n+1} \binom{2n}{n}$. From this, the number of shuffles of two parenthesis systems is $S_n = C_n C_{n+1}$.

**Proof.** We first prove that the number of parenthesis systems of size $n$ is the $n^{th}$ Catalan number $C_n = \frac{1}{n+1} \binom{2n}{n}$. We use the standard terminology “left parenthesis” and “right parenthesis” instead of $a$ and $\bar{a}$ to better visualize a valid parenthesis system. We know $\binom{2n}{n}$ is the total number of writing $n$ left and $n$ right parentheses regardless of validity and we want to show $\binom{2n}{n}$ is the number of writing invalid parentheses system.

The parenthesis sequence is valid if when we read from left to right, there are at least as many left parentheses as right parentheses. Suppose a sequence $L$ is not valid, then there is a least $k$ where there is a right parenthesis at position $k$ and equally $k-1$ left and $k-1$ right parentheses before $k$. If we swap all left parentheses for right and all right for left in the first $k$ positions of $L$, we will get a collection of $n+1$ left parentheses and $n-1$ right parentheses. Conversely, given a sequence of $n+1$ left and $n-1$ right parentheses, let $k$ be the first position where there are more left parentheses than right parentheses up to that point. Flipping all left parentheses for right and all right for left in the first $k$ positions will result back in an invalid sequence of $n$ left parentheses and $n$ right parentheses since there are...
more right parentheses than left up to $k$. It’s not hard to see that the first and second mappings are inverse. Therefore, the number of invalid sequences of $n$ left and $n$ right parentheses is equal to the total number of sequences of $n + 1$ left and $n - 1$ right parentheses, which is $\binom{2n}{n+1}$. The number of valid parenthesis systems of size $n$ is $\binom{2n}{n} - \binom{2n}{n+1} = \frac{1}{n+1}\binom{2n}{n} = C_n$.

Given a parenthesis system $A$ of size $k$ on $\{a, \bar{a}\}$ and a parenthesis system $B$ of size $n - k$ on $\{b, \bar{b}\}$, we want to form a combined parenthesis system of size $n$. For each $k$, $0 \leq k \leq n$, there are $\binom{2k}{k}$ ways to insert all the letters of $A$ into the merged parenthesis system. Summing over all $k$, we get the result:

$$S_n = \sum_{k=0}^{n} \binom{2n}{2k} C_k C_{n-k}$$

$$= \sum_{k=0}^{n} \binom{2n}{2k} \frac{1}{k+1} \binom{2k}{k} \frac{1}{n-k+1} \binom{2n-2k}{n-k}$$

$$= \sum_{k=0}^{n} \frac{1}{k+1} \frac{1}{n-k+1} \frac{(2n)!}{(2k)! (2n-2k)!} \frac{(2k)!}{k!} \frac{(2n-2k)!}{(n-k)!}$$

$$= \binom{2n}{n} \sum_{k=0}^{n} \binom{n+1}{k} \frac{1}{k+1} \frac{1}{n-k+1} \binom{n+1}{k+1}$$

$$= \binom{2n}{n} \binom{2n+2}{n}$$

$$= \frac{(2n)!}{(n+1)!^2} \binom{2n+2}{n}$$

$$= \frac{(2n)!}{(n+1)! (n+1)! (n)! (n+2)!}$$

$$= \frac{1}{n+1} \binom{2n}{n} \frac{1}{n+2} \binom{2n+2}{n+1}$$

$$= C_n C_{n+1}.$$ 

The counterclockwise contour code can be seen as a word $w$ on the alphabet $\{\uparrow, \downarrow, \leftarrow, \rightarrow\}$ whose $i$th letter is the $i$th move, with the restriction that $|w|_\uparrow = |w|_\downarrow$ (resp. $|w|_\leftarrow = |w|_\rightarrow$) and for all prefixes $w'$ of $w$, $|w'|_\rightarrow \geq |w'|_\downarrow$ (resp. $|w'|_\uparrow \geq |w'|_\downarrow$). Mullin’s bijection essentially states that this encoding is one-to-one.

2.3. Mullin’s Bijection.

**Theorem 2.2.** The contour code is a bijection between tree-rooted planar maps $(M, T, c)$ with $n$ edges, and shuffles of parentheses systems on $\{\rightarrow, \leftarrow\}$ and $\{\uparrow, \downarrow\}$ of size $n$ (total length $2n$).

**Proof.** We first consider touring the spanning tree as described in Section 2.1. Recall that $\uparrow$ is the first time we follow an edge of the spanning tree, $\downarrow$ second time we follow an edge, $\rightarrow$ first time we cross an edge not in the spanning tree and $\leftarrow$ second time we cross an edge not in the spanning tree.

The reverse mapping can be constructed as follows: given a parenthesis shuffle $w$, we first create the tree corresponding to the subword of $w$ consisting of letters $\{\uparrow, \downarrow\}$, which will give the spanning tree. Next, we glue to this tree a head for each letter $\rightarrow$ and a tail for each letter $\leftarrow$. There is only one way to connect heads to
tails to ensure no edges intersect so that the result is a planar map. If the map $M$ has size $n$, then the corresponding parenthesis-shuffle $w$ has size $n$, since $|w|_{\uparrow}$ is the number of edges in the tree and $|w|_{\downarrow}$ is the number of edges not in the tree. A more detailed decomposition of the reverse mapping involves an in-depth discussion of tree orientation, vertex explosion and non-crossing partitions that are discussed in [2].

This mapping establishes a one-to-one correspondence between tree-rooted maps of size $n$ and parenthesis shuffles of size $n$. As a result, there are $C_nC_{n+1}$ tree-rooted maps of size $n$.

**Corollary 2.3.** The number of tree-rooted maps with $n$ edges is

$$\sum_{i+j=n \atop i,j \geq 0} \binom{2n}{2i} C_iC_j = C_nC_{n+1}$$

This means that the tree-rooted maps of size $n$ are in one-to-one correspondence with pairs of plane trees of size $n$ and $n+1$ respectively.

Let $(M,c)$ be a rooted planar map, $T$ be a spanning tree of $M$ and $T^*$ be the dual spanning tree of $M^*$. It can further be observed that counterclockwise walk around the spanning tree $T$ is simultaneously a clockwise walk around $T^*$. Using the same symbol as the counterclockwise contour code, we can define the clockwise contour code of the rooted map $(M^*,c)$ with the tree $T^*$.

**Proposition 2.5.** The total order induced by the counterclockwise walk around $T$ starting from $c$ in $M$ is identical to the total order induced by the clockwise walk around $T^*$ starting from $c$ in $M^*$, as illustrated in Figure 5. In particular the counterclockwise contour code of $(M,c)$ with $T$ and the clockwise contour code of $(M^*,c)$ with $T^*$ are mapped onto the other by exchanging $\uparrow \leftrightarrow \leftarrow$ and $\downarrow \leftrightarrow \rightarrow$ (first time crossing an edge exchanged with first time going along an edge and second time crossing an edge exchanged with second time going along an edge).

**Proof.** If we draw the contour walk as a curve $C$ traveling between the spanning tree $T$ and the dual spanning tree $T^*$ in the superimposition of $M$ and $M^*$, the intersections of the curve $C$ with edges of $M\setminus T$ and $M^*\setminus T^*$ create a quadrangle of four vertices in the middle of every half-edge of $M$ and $M^*$ that is not in $T$ or $T^*$. In the superimposition of $M\setminus T$, $M^*\setminus T^*$ and $C$, each of these new vertices is adjacent to three other new vertices (two along $C$ and one along an edge of $M\setminus T$ or $M^*\setminus T^*$) and to one vertex of $M$ or $M^*$. We call each quadrangle $Q$.

We first prove that the dual spanning tree to $T^*$ is $T$ itself. If $T^{**}$ is the dual spanning tree to $T^*$, then the edges of $T^{**}$ correspond to edges of $M^{**}$ which don’t cross an edge of $T^*$. The edges of $T^*$ are the edges of $M^*$ which don’t cross an edge of $T$. Therefore, $T^{**}$ is exactly $T$. Hence, the quadrangle $Q$ used in building the contour walk for $(M,T)$ is the same as the quadrangle used when doing the contour walk with $(M^*,T^*)$, except now when we build the contour, passage through a quadrangle bisected by $T^*$ corresponds to $\uparrow/\downarrow$ and passage through one bisected by $T$ corresponds to $\leftarrow/\rightarrow$. \hfill \Box

3. Site-percolated loopless triangulations

In this section, we will explore a different bijection between maps and walks.
3.1. **Site-percolations on triangulations.** As defined above, a *triangulation* is a planar map in which every face has degree 3 and a *cubic map* is a map such that every vertex has degree 3. A *loopless triangulation* is a triangulation without self-loops, but multiple edges between two vertices are still allowed. A *near-triangulation* is a planar map in which all faces except one have degree 3 and we can always arbitrarily choose the outer face to be the face which doesn’t have degree 3. A near-triangulation such that the boundary of the root face is simple (that is, without cut points) is a *triangulation with a simple boundary*. We denote the set of rooted loopless triangulations with a simple boundary by $T$.

A *site-percolation configuration* on a planar map $M$ is any coloring of its vertices in black or white. A *site-percolated planar map* (or *percolated map* for short) $(M, \sigma)$ consists of a planar map $M$ and a site-percolation configuration $\sigma$. An edge of $(M, \sigma)$ is *unicolor* if its endpoints are of the same color, and *bicolor* otherwise. Keeping only unicolor edges of $(M, \sigma)$ gives a disjoint union of planar maps called *percolation clusters* that could be either *black* or *white*.

Now suppose $M$ is a near-triangulation. Each inner triangle of $(M, \sigma)$ is either *unicolor*, meaning all of its vertices have the same color, or *bicolor*, meaning it is incident to two bicolor edges. Drawing in each bicolor triangle a curve joining the middle of the two incident bicolor edges results in a set of disjoint curves, which we call *percolation interfaces*. If a percolation interface is a cycle, we call it a *percolation cycle*. Otherwise, it is a path starting and ending on the boundary of $M$, which we call a *percolation path*. The percolation interfaces separate the clusters of black and white vertices of $M$, as shown in Figure 6. Note percolation interfaces are also simple paths and cycles on the dual map $M^\ast$. The *length* of a percolation interface is the number of triangles of $M$ it crosses.

We say a site-percolated near-triangulation $(M, \sigma)$ satisfies the *root-interface condition* if the root-edge is oriented from a white vertex to a black vertex, and no other outer edge goes from a white vertex to a black vertex in the counterclockwise direction around the root face. We denote the set of such $(M, \sigma)$ by $T_P$.

For any $(M, \sigma) \in T_P$, the percolation path connects the root-edge to another bicolor outer edge and the latter is called the *top-edge*. The white and black endpoints of the top-edge are called the *top-left* and *top-right* vertices respectively.
The white and black outer vertices are called the left vertices and right vertices respectively, while the unicolor white and unicolor black outer edges are called left edges and right edges. An example is given in Figure 6.

3.2. Depth First Search. We now introduce a popular algorithm called the depth-first search algorithm (DFS) and DFS trees.

Definition 3.1. Let $G$ be a connected graph and $v_0$ a vertex. A depth-first search of $G$ starting at $v_0$ is a visit of its vertices by a “chip” according to the following rule. At the beginning of the process, the chip is placed at $v_0$ and the vertex $v_0$ is considered “visited”, while all the other vertices are “unvisited”. We repeat the following step, and let $u$ denote the current position of the chip:

- If there exists some edge between $u$ and an unvisited neighbor $v$, we choose such an edge $e = \{u, v\}$ and move the chip from $u$ to $v$. Then we mark $v$ as visited, call $u$ the parent of $v$ and $e$ the parent edge of $v$.
- If there is no unvisited neighbor for the current vertex $u$ and if $u \neq v_0$, the chip moves back to the parent of $u$ and explores any other unvisited neighbor of the parent of $u$. On the other hand, if $u = v_0$, which means all reachable vertices have been visited, the depth first search finishes.

It should be noted that the tree associated to a depth first search is the spanning tree of $G$ made of the set $T$ of all the parent edges.

Next, we define a set of DFS trees and a set of percolation configurations.

Definition 3.2. Let $M$ be a near-triangulation in $\mathcal{T}$ and let $M^*$ be the dual map. Let $v_0$ be the root vertex of $M^*$. An inner coloring of $M$ is a coloring of the inner vertices of $M$ in black or white. We denote the set of inner colorings of $M$ by $\text{Perc}_M$ and the set of DFS trees $T$ of $M^*$ rooted at $v_0$ such that the root-edge $e_0^*$ of $M^*$ is in $T$ by $\text{DFS}_{M^*, v_0}$.

It can been proven that a spanning tree $T$ of $G$ can be obtained by a DFS of $G$ starting at $v_0$ if and only if for any two adjacent vertices of $G$, one of the vertices
is the ancestor of the other in the tree $T$. We call such a tree a $v_0$-DFS tree of $G$. It’s not hard to discover that for all $M \in T$, the trees in $DFS_{M^*,v_0}$ contain no edge incident to $v_0$ apart from the root-edge $e^*_0$. Otherwise $v_0$ would have several children and the characterization of $v_0$-DFS trees would be violated. This implies that during a DFS of $M^*$ starting at $v_0$, the chip doesn’t visit $v_0$ except at the first and last step. For each visited vertex $v \neq v_0$, we consider the incident edges $e_1, e_2, e_3$ in clockwise order around $v$. We call $e_1$ the parent edge of $v$, $e_2$ the left forward edge of $v$ and $e_3$ the right forward edge. The face $f$ containing the corner between $e_2$ and $e_3$ is the forward face of $v$.

We now define the bijection $\Delta_M$ between $\text{Perc}_M$ and $DFS_{M^*,v_0}$.

**Definition 3.3.** Given an inner coloring $\sigma^o \in \text{Perc}_M$, we consider the corresponding coloring of the faces of $M^*$ with the convention that the faces of $M^*$ dual to the outer vertices of $M$ are colored white. We define $\Delta_M(\sigma^o)$ as the spanning tree of $M^*$ obtained by the DFS of $M^*$ defined as follows:

1. The chip starts at the root vertex $v_0$ of $M$ and first moves along the root edge $e^*_0$ of $M^*$.
2. If several edges are possible when moving to an unvisited neighbor $v$ from the current position $u$, if the forward face $f$ is black, the chip moves along the forward left edge. Otherwise, the chip moves along the forward right edge. This is explicitly illustrated in Figure 7.

We can now state the bijection between DFS trees and percolation configurations.

**Definition 3.4.** Given a $v_0$-DFS tree $T^* \in DFS_{M^*,v_0}$, and the dual spanning tree $T$ of $M$, we define a coloring $\Lambda_M(T^*)$ of $M$ as follows. Let $u$ be an inner vertex of $M$. Let $e$ be the parent-edge of $u$ in the spanning tree $T$ of $M$. The edge $e^* \in M^* \setminus T^*$ joins a vertex $v_1$ to one of its descendents $v_2$ in $T^*$, if the edge $e^*$ is on the left of the path of $T^*$ from $v_0$ to $v_2$ at $v_1$, we color $u$ white. Otherwise, we color $u$ black. An example is illustrated in Figure 8.

From the above mapping, we obtain the following theorem

**Theorem 3.5.** The mapping $\Delta_M$ is a bijection from $\text{Perc}_M$ to $DFS_{M^*,v_0}$ and $\Lambda_M$ is its inverse. An example is shown in Figure 9. Moreover, for any site-percolation configuration $\sigma$ of $M$ that satisfies the root-interface condition and for any inner coloring $\sigma^o \in \text{Perc}_M$ that agrees with coloring of $\sigma$, the tree $T^* = \Delta_M(\sigma^o)$ satisfies these properties:
Figure 8. An edge $e^*$ of $M^*$ is incident to the face $u^*$ such that the cycle inside $T^* \cup \{e^*\}$ separates $u^*$ from the root face, with $v_1$, $v_2$ being the endpoints of $e^*$ and $v_1$ being the ancestor of $v_2$. $u$ is colored white if $e^*$ is on the left of $T^*$ at $v_1$ and black otherwise.

Figure 9. Left: An inner coloring $\sigma^\circ \in \text{Perc}_M$ represented as a coloring of the faces of $M^*$ not incident to the root vertex $v_0$. Right: The $v_0$-DFS tree $T^* = \Delta_M(\sigma^\circ)$.

(1) The percolation path of $(M, \sigma)$ is contained in $T^*$.
(2) For any percolation cycle $C$ of $(M, \sigma)$, every edge of $C$ except one is in $T^*$.
(3) Consider the coloring of the faces of $M^*$ corresponding to the configuration $\sigma$. Any edge of $T^*$ separating a black face and a white face of $M^*$ has the white face on its left when oriented from parent to child.

Proof. The following proof is outlined in [3]. We first prove that $\Delta_M$ is injective. Let $\sigma_1^\circ$, $\sigma_2^\circ$ be distinct inner colorings of $M$ and let $F$ be the set of faces of $M^*$ having different colors in $\sigma_1^\circ$ and $\sigma_2^\circ$. Let $u$ be the first vertex of $M^*$ incident to a face in $F$ encountered during the DFS of $M^*$ corresponding to $\Delta_M(\sigma_1^\circ)$. It is clear that $u$ is also the first vertex of $M^*$ incident to a face in $F$ encountered during the DFS corresponding to $\Delta_M(\sigma_2^\circ)$. It can also be observed that the parent-edge $e_1$ of $u$ is the same in $\Delta_M(\sigma_1^\circ)$ and $\Delta_M(\sigma_2^\circ)$. The trees $\Delta_M(\sigma_1^\circ)$ and $\Delta_M(\sigma_2^\circ)$ each contain a different forward edge of $u$ due to their different colorings as a direct result of Definition 3.3 (2). Hence $\Delta_M(\sigma_1^\circ) \neq \Delta_M(\sigma_2^\circ)$ and $\Delta_M$ is injective.

Next we want to show $\Delta_M \circ \Lambda_M = \text{Id}$. Let $T^* \in \text{DFS}_{M^*,v_0}$, $\sigma^\circ = \Lambda_M(T^*)$, and $T' = \Delta_M(\sigma^\circ)$. Suppose by contradiction that $T' \neq T^*$. Let $e$ be the first edge in $T' \setminus T^*$ added to $T'$ during the DFS corresponding to $\Delta_M(\sigma^\circ)$. Let $u$, $v$ be the endpoints of $e$, with $u$ being the parent of $v$ in $T'$. Note that the path $P$ from $v_0$ to
unmatched Kreweras walks words that are in one-to-one correspondence with a Kreweras walk as the set with a has a matched a and at most one matching the subwalk a are as many w c no more { the alphabet of steps: Starting from the origin, staying in the first quadrant and consisting of three kinds w spondence with the site-percolations on triangulations defined above. The set of the Kreweras walk.

C the percolation interface v C a vertex C the face f C e the edge incident to u C e left-forward and right-forward edges respectively. Equivalently, e1, e2 and e3 are the edges incident to u in clockwise order, with e1 in P and f being the face of M∗ between e2 and e3. We know that for vertices u and v, one is the ancestor of the other in T and moreover v cannot be an ancestor of u in T∗ because it is not on P. Hence v is a descendent of u in T∗. Let Q be the path of T∗ from u to v. Observe that Q \ {e} forms a cycle of M∗ and P and f are on different sides of this cycle because the paths P, Q \ T∗ cannot cross. In particular, f is not incident to v0 and hence f is the dual of an inner face of M. In fact, u is the first vertex incident to f encountered during the DFS corresponding to ∆M(σ̂). Thus by definition of ∆M, the face f is black if e = e2 and white if e = e3. However, by definition of ΛM, the face f is white if e = e2 and black if e = e3, which is a contradiction. Hence T′ = T∗ and M M ΛM = Id. Given M is injective, we see that M is a bijection and M is the inverse mapping.

Finally, we proceed to prove the three properties about percolation interfaces. From the definitions, the chip will visit all the vertices on the percolation path from v0 to its other end before visiting any other vertex, which proves (1). Similarly, given any percolation cycle C of (M, σ), we consider the first time the DFS reaches a vertex v of M∗ on the cycle. In the next steps, the chip will follow the edges of the cycle C starting at v without visiting any vertex not on C until it reaches the second neighbor of v on C and this proves (2). To prove (3), the direction in which the percolation interface C is followed is such that the black faces are on the right and the white faces are on the left, which concludes the proof.

3.3. The Kreweras walk. In this section, we consider the set of walks in correspondence with the site-percolations on triangulations defined above. The set of Kreweras walks, denoted K, is referred to as the set of lattice walks in the Z2 plane starting from the origin, staying in the first quadrant and consisting of three kinds of steps: \{a = (1, 0), b = (0, 1), c = (−1, −1)\}. In addition, we denote the finite words that are in one-to-one correspondence with a Kreweras walk as the set \(W\) on the alphabet \{a, b, c\} with the restriction that any prefix \(w′\) of the word contains no more c’s than a’s and than b’s. Equivalently, \(|w′|_a \geq |w′|_b\), \(|w′|_b \geq |w′|_c\).

For a word \(w = w_1w_2...w_n \in W\), we say \(w_i\) is an a-step (resp. b-step, c-step) if \(w_i = a\) (resp. \(w_i = b\), \(w_i = c\)). An a-step \(w_i\) and a c-step \(w_k\) are matching if there are as many a-steps and c-steps in each subword \(w_iw_{i+1}...w_k\) for \(i < k\), and there are strictly more a-steps than c-steps in \(w_iw_{i+1}...w_j\) for \(i \leq j \leq k - 1\). Visually, the subwalk \(w_iw_{i+1}...w_k\) stays strictly to the right of \(w_1w_2...w_i\) and \(w_1w_2...w_k\).

Each a-step or b-step has at most one matching c-step and those without a matching c-step are called unmatched. Each c-step has at most one matching a-step and at most one matching b-step, the absence of one of which makes the c-step unmatched. In the special case of \(w \in K\), every c-step is matched. If a walk \(w \in K\) has \(x\) unmatched a-steps and \(y\) unmatched b-steps, then it ends at the point \((x, y)\).

A c-step \(w_k\) is of type a if \(w_k\) has a matched b-step \(w_j\) and either no matching a-step or a matching a-step \(w_i\) with \(i < j\). Similarly, a c-step \(w_k\) is of type b if \(w_k\) has a matched a-step \(w_i\) and either no matching b-step or a matching b-step \(w_j\) with \(j < i\).
Corollary 3.6. The number of walks of length $3n$ beginning and ending at the origin is

\[
k_n = \frac{4^n}{(n + 1)(2n + 1)} \binom{3n}{n}
\]

Proof. Unlike the proof provided in Corollary 2.1, this closed-form expression has no simple proof. A constructive proof was shown by Bousquet-Melou in 2015 [6], but a simpler derivation and a more direct combinatorial explanation are yet to be provided. \qed

3.4. The bijection.

Definition 3.8. For a walk $w \in K$, $(M, \sigma) = \Phi(w)$ is constructed in the following way: For $w = w_1w_2...w_n$, we define $\Phi(w) = \phi_{w_n} \circ \phi_{w_{n-1}} \circ \cdots \circ \phi_{w_1}(M, \sigma_0)$, where $(M, \sigma_0) \in \mathcal{T}_P$ is a percolated map with a single root-edge going from a white vertex to a black vertex, and for each step $w_i \in \{a, b, c\}$, $\phi_{w_i}$ obey the following rules, which are further illustrated in Figure 10. An example of the bijection $\Phi$ is presented in Figure 11.

- If $w_i = a$, the map $\phi_a(M, \sigma)$ is obtained by gluing a triangle with two white vertices and one black vertex to the top-edge of $(M, \sigma)$.
- If $w_i = b$, the map $\phi_b(M, \sigma)$ is obtained by gluing a triangle with two black vertices and one white vertex to the top-edge of $(M, \sigma)$.
- If $w_i = c$, for $(M, \sigma) \in \mathcal{T}_P$ having both a left edge and a right edge, we define $\phi_c(M, \sigma)$ as follows: Let $e_l$ be the left edge incident to the top-left vertex $v_l$, and let $e_r$ be the right edge incident to the top-right vertex $v_r$. Let $P$ be the percolation path of $(M, \sigma)$ and consider $P$ as starting at the root-edge and ending at the top-edge $e$. By definition of $\mathcal{T}_P$, the inner triangles $t_l$ and $t_r$ incident to $e_l$ and $e_r$ respectively are on $P$; one of them $t \in \{t_l, t_r\}$ is the last triangle on $P$ incident to a left or right edge. If $t = t_r$, then we recolor the vertex $v_l$ in white and glue the edges $e$ and $e_l$ together so that $v_l$ becomes an inner white vertex. Symmetrically, if $t = t_l$, then we recolor the vertex $v_r$ in black and glue the edges $e$ and $e_r$ together so that $v_r$ becomes an inner black vertex.

Next we introduce some more detailed corresponding concepts between the steps of $w \in K$ and the vertices, faces and edges of $(M, \sigma)$, which follow naturally from the construction of $\Phi(w)$ and will be used in statement of Theorem 3.10.

Definition 3.9. Let $w = w_1w_2...w_n \in K$ and $(M, \sigma) = \Phi(w) = \phi_{w_n} \circ \cdots \circ \phi_{w_1}(M, \sigma_0)$.

- We call the in-edges of $(M, \sigma)$ the edges which are neither a left edge, a right edge nor the top-edge. Let $E$ be the set of in-edges of $(M, \sigma)$. We define the mapping $\eta_E$ from $\{1, 2, \ldots, n\}$ to $E$ as follows: For each $i$, applying $\phi_{w_i}$ makes the top edge an in-edge $e$ of $(M, \sigma)$ and we set $\eta_E(i) = e$.
- Let $V$ and $F$ be the sets of inner vertices and inner triangles of $M$ respectively. We define the mapping $\eta_{V,F}$ from $\{1, 2, \ldots, n\}$ to $V \cup F$ as follows: if $w_i$ is an $a$-step or $b$-step, then applying $\phi_{w_i}$ adds one inner triangle $f$ to $(M, \sigma)$ and we set $\eta_{V,F}(i) = f$. If $w_i$ is a $c$-step, then applying $\phi_{w_i}$ adds one inner vertex $v$ to $(M, \sigma)$ and we set $\eta_{V,F}(i) = v$.
- For an unmatched $a$-step (resp. $b$-step) $w_i$, the triangle $\eta_{V,F}(i)$ is incident to a left (resp. right) edge $e$ of $(M, \sigma)$ and we set $\eta_{LR}(i) = e$. 


Theorem 3.10. The mapping $\Phi$ is a bijection between $K$ and $T_P$. For a walk $w \in K$ and its image $\Phi(w) = (M, \sigma)$, we have:

1. The mapping $\eta_E$ gives a one-to-one correspondence between the steps of $w$ and the in-edges of $(M, \sigma)$.
2. The mapping $\eta_{VF}$ gives a one-to-one correspondence between the $a$-steps and $b$-steps of $w$ and the inner triangles $M$. The mapping $\eta_{VF}$ also gives a one-to-one correspondence between the $c$-steps of $w$ of types $a$ (resp. $b$) with the white (resp. black) inner vertices of $(M, \sigma)$.
3. The mapping $\eta_{LR}$ gives a one-to-one correspondence between the unmatched $a$-steps (resp. $b$-steps) of $w$ and the left (resp. right) edges of $(M, \sigma)$.
Theorem 3.12. We also define the mapping \( \Omega \) as defined above is a bijection between \( \mathcal{K} \) and \( \mathcal{C}_T \).

Proof of Theorem 3.10. We will provide a brief summary of the result proven by Bernardi [1]. For a non-root outer edge \( e \) of \( M \), we denote by \( \text{Perc}_M^e \) the set of site-percolation configurations of \( M \) satisfying the root-interface condition such that \( e \) is bicolor, and such that the percolation path visits every inner triangle of \( M \) incident to an outer edge. For a non-root outer edge \( e^* \) of \( M^* \) incident to \( v_0 \), we denote by \( \text{DFS}_{M^*}^{e^*} \) the set of trees \( T^* \in \text{DFS}_{M^*} \) such that the non-root vertex \( v_0 \) incident to \( e^* \) is the descendant in \( T^* \) of every vertex of \( M^* \) adjacent to \( v_0 \). For \( T^* \in \text{DFS}_{M^*} \), we denote by \( \Lambda_M^e(T^*) \) the unique site-percolation configuration of \( M \) that extends the inner-coloring \( \sigma^0 = \Lambda_M(T^*) \) and satisfies the root-interface condition with a bicolor.

Recall that \( \mathcal{C}_T \) is the set of triples \( (M^*, e^*, T^*) \) where \( M^* \) is the dual of a near-triangulation \( M \in T \), \( e^* \) is an edge of \( M^* \) incident to the root-vertex \( v_0 \) and \( T^* \) is in \( \text{DFS}_{M^*} \). We call \( e^* \) the head edge and the point of \( e^* \) distinct from \( v_0 \) the head vertex. We also define \( M_0^* \) to be the rooted map with one vertex and one self-loop, with \( e_0^* \) as the root-edge and \( T_0^* \) as the unique spanning tree. For \( w = w_1 w_2 \ldots w_n \in K \), the image \( \Omega(w) \) is defined as the triple \( (M^*, e_0^*, T^*) = \Omega_{w_n} \circ \ldots \circ \Omega_{w_2} \circ \Omega_{w_1}(M_0^*, e_0^*, T_0^*) \).

It can be shown that the composition of the bijection \( \Omega \) with the mapping \( \Lambda_M^e \) is equal to the mapping \( \Phi \). More precisely, for a walk \( w \in K \), if \( \Omega(w) = (M^*, e^*, T^*) \), then \( \Phi(w) = (M, \sigma) \), where \( \sigma = \Lambda_M^e(T^*) \). This shows that \( \Phi \) is a bijection between \( \mathcal{K} \) and \( \mathcal{T}_p \), which concludes the proof. \( \square \)

As a consequence, we get the following corollary.
**Corollary 3.13.** The number of cubic maps of size $n$ with a distinguished DFS tree is

$$d_n = k_n = \frac{4^n}{(n + 1)(2n + 1)} \binom{3n}{n}$$

4. **Extension: Depth First Search Algorithm**

On top of being an algorithm for constructing a spanning tree of a graph, the DFS algorithm is one of the several essential graph algorithms for traversing graphs and answering theoretical questions in combinatorial theory.

We revisit the DFS algorithm proposed by Tarjan [7] as a fundamental method to traverse a graph $G = (V, E)$. DFS follows paths as far as possible: from a vertex $v$ already reached, we proceed to any neighboring vertex $w$ which has not been visited; then we go from $w$ to another vertex not yet visited as long as this is possible. If we cannot move on, we backtrack as much as necessary until we are back to the initial vertex $v_0$. In this way, one constructs maximal paths starting at $v_0$. In the process, we label the vertices with numbers $nr$ according to the order in which they are visited, and we call the vertex from which a vertex $w$ is accessed $p(w)$. $u(vw)$ is used to denote whether the edge between vertex $v$ and $w$ has been reached.

**Algorithm 1** Depth First Search

```plaintext
Require: $G = (V, E)$ be a graph and $v_0 \in V$ a vertex of $G$
for $v \in V$ do
    $nr(v) \leftarrow 0$
    $p(v) \leftarrow 0$
end for
for $e \in E$ do
    $u(e) \leftarrow false$
end for
$i \leftarrow 1$
$v \leftarrow v_0$
$nr(v_0) \leftarrow 1$
repeat
    while there exists $w \in A_v$ with $u(vw) = false$ do
        $u(vw) \leftarrow true$ for some $w \in A_v$ with $u(vw) = false$
        if $nr(w) = 0$ then
            $p(w) \leftarrow v$
            $i \leftarrow i + 1$
            $nr(w) \leftarrow i$
            $v \leftarrow w$
        end if
        $v \leftarrow p(v)$
    end while
until $v = v_0$ and $u(v_0w) = true$ for all $w \in A_{v_0}$
```

It is not hard to see that each edge in the connected component of $v_0$ is used exactly once in each direction during the execution of Algorithm 1. DFS indeed can be used to traverse a connected graph and solve real-world problems such as
finding the exit of a maze, rendering it one of the most powerful graph traversal algorithms.

Acknowledgments

I would like to sincerely thank my mentor, Charley Devlin, for clarifying my doubts on the topics discussed in this paper, guiding me along the way and providing me with extremely detailed feedback on my draft. I would also like to express my gratitude to Professor Ewain Gwynne for his lecture series on site percolation that gave inspiration to the second part of this paper. Finally, I am grateful for the hard work of Professor Peter May in organizing the 2023 University of Chicago Math REU program during which this paper was written.

References