Spring 2023

Parking Garage Functions

Felicia Elizabeth Flores
Bard College

Parking Garage Functions

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Felicia Elizabeth Flores

Annandale-on-Hudson, New York
May, 2023

Abstract

This project is about a generalization of parking functions called parking garage functions. Parking functions have been well studied, but the concept of parking garage functions is new and introduced in the project. Parking garage functions are sequences that represent the parking garage level preferences of cars which lead to all cars parking on a level after a systematic placement. We found a recursive formula for the number of sequences that are a parking garage function. We also found a closed formula for a subset of parking garage functions, descending parking garage functions, via a bijection between descending parking garage functions and Dyck paths which are paths on a rectangular grid which only take right and upward steps starting at the origin and remain under a positively sloped diagonal that goes through the origin.

Contents

Dedication

I dedicate this project to the undergraduates in the Bard College Mathematics Program that are women or people of color. It isn't impossible to be a mathematics major at Bard. You just have to work hard in your classes and even harder to find your confidence! In addition, do yourself a favor and build a group of math friends that believe in you even when you struggle to like I had with Darrion Thornburgh, Jen Lara, Hannah Kaufmann, and Julia Crager ♡♡♡♡

I also dedicate this project to my father who came to the United States as an undocumented immigrant from Peru for instilling in me the grit necessary to persevere in a field that has more failure than success.

Acknowledgments

I acknowledged my advisor, Lauren Rose, for foremost pushing me to be a mathematics major, but also for overall shaping me into the mathematician I am today. She taught me how to navigate being a women in the classroom, on a research team, as a presenter at a conference, as a tutor, as a math enrichment program director, and as a leader of an AWM student chapter. Without her, I would have struggled to overcome my imposter syndrome.

I also acknowledge Pamela Harris for inspiring the project topic and Alejandro Morales for his indispensable resource sharing and them both for their encouragement!

Lastly, I acknowledge Vegim Osmani to whom I got engaged to shortly before this project was due for all the ways in which he supports the family and I during demanding times in my mathematical career.

1
Introduction

The journey of this senior project began when I attended MathFest 2022 in Philadelphia. On August 5th, I attended an invited address by Pamela Harris entitled "Parking Functions: Choose Your Own Adventure". She is an Associate Professor in the Department of Mathematical Sciences at the University of Wisconsin at Milwaukee. Her talk was very accessible and visually enticing! As a double major in mathematics and studio arts the latter made me drawn to the material. She went through all the projects she had done with students in the past along with mentioning unsolved problems. I also appreciated seeing an invited address given so confidently by not only a member of Lathisms: Latinxs and Hispanics in the Mathematical Sciences, but the president and co-founder. I introduced myself to Pamela and began researching.

Parking functions are sequences that represent the parking space preferences of n cars for n spaces along a one-way street which lead to all n cars claiming a space after parking in a systematic way. Such a set of sequences has already been counted by both a recursive formula and closed formula.

My first motivating question was: How many sequences yield $n-1$ cars parking or $n-2$ cars parking or $n-3$ cars parking all the way to 1 car parking out of the n cars? Progress was being made. Then during a Discrete Math Workshop at Smith College on November 19th, 2023, I met Alejandro Morales. He is an assistant professor in the Department Mathematics and Statistics

at the University of Massachusetts, Amherst. As a collaborator and co-author with Pamela Harris along with being a researcher in the field of enumerative and algebraic combinatorics, he proved to be a great resource! He ended up finding and sending me a paper, [9], that answered my motivating question.

Determined to work on an unsolved problem, I returned to the paper that coincided with Pamela's original invited address, [6]. One of the unsolved problems mentioned had to do with parking multiple cars in a single spot. This idea then took on the name *clown functions* which are sequences that represent the clown car preferences of kn clowns for n cars with capacity k along a one-way sidewalk which lead to all kn clowns in a car after a systematic placement. However, I wanted to find a more direct connection with one-way streets for parking functions. This is how I ended up coming up with the new concept of **parking garage functions**.

Parking garage functions are sequences that represent the parking garage level preferences of kn cars for n levels with capacity k which lead to all kn cars parking on a level after a systematic placement with only upward movement. My new motivating question became: How many parking garage functions are there for a certain amount of levels at a certain capacity? I wrote a python code to output testing results to input into the On-line Encyclopedia of Integer Sequences. The result I got from OEIS gave a recursive formula for a scenario that can be interpreted as parking garage functions. In addition, I found that parking functions have numerous bijections. One was with Dyck paths and a variation of parking functions, descending parking functions. This led me to discover that there was a bijection between Dyck paths and descending parking garage functions. That was the only closed formula we have found so far that is close to answering my motivating question.

In Chapter 2, we introduce *lattice paths* on a rectangular grid of points in $\mathbb{Z}^2$ that only travel in steps to the right or upwards. We also introduce *lattice words* which are words over the alphabet $\{X, Y\}$ and prove a closed formula for the number of lattice words with a certain amount for X'S and certain number of Y's. Then we will introduce a slight variation of lattice paths called *lattice paths at zero*. Next, we prove a bijection between *lattice paths at zero* and *lattice words*.

In Chapter 3, we introduce *Dyck paths* which are lattice paths at zero under a positively sloped diagonal that goes through the origin. We give a closed formula for the number of Dyck paths with a certain final coordinate. We also introduce *Dyck words* which are lattice words with an added restriction on all prefixes. Next, we prove a bijection between *Dyck paths* and *Dyck words*.

In Chapter 4, we introduce sequences of natural numbers that represent the preferred parking space out of a one-way street for a certain amount of cars. *Parking functions* are the sequences which lead to all cars parking in all parking spaces available on a street given a systematic way of parking. In addition, we give the known recursive and closed formulas for the number of parking functions given a certain number of cars. Then we will introduce a slight variation of parking functions called *descending parking functions*. Next, we provide an argument for a bijection between *descending parking functions* and *Dyck words*.

In Chapter 5, we introduce a new concept called *parking garage functions* which are a generalization of *parking functions*. *Parking garage functions* are the sequences which lead to all cars parking in all parking spaces available on each level in a parking garage given a systematic way of parking. We developed python code which outputs a sequence. This led us to paper from 1977, [5], which gives us a recursive formula for a scenario that can be interpreted as parking garage functions. Then we will introduce a slight variation of parking garage functions called *descending parking garage functions*. Finally, we provide an argument for a bijection between *descending parking garage functions* and *Dyck words*.

2
Lattice Paths and Lattice Words

In this chapter, we introduce *lattice paths* on a rectangular grid of points in $\mathbb{Z}^2$ that only travel in steps to the right or upwards. We also introduce *lattice words* which are words over the alphabet $\{X, Y\}$ and prove a closed formula for the number of lattice words with a certain amount for X'S and certain number of Y's. Then we will introduce a slight variation of lattice paths called *lattice paths at zero*. Next, we prove a bijection between *lattice paths at zero* and *lattice words*.

2.1 Lattice Paths

In this section, we introduce *lattice paths* on a rectangular grid of points in $\mathbb{Z}^2$ that only travel in steps to the right or upwards.

The following definition is modified from [10].

Definition 2.1.1. [10] Let $k \in \mathbb{N}$ and $m, n \in \mathbb{N} \cup \{0\}$. A sequence of lattice points

$$P = \{(x_0, y_0), (x_1, y_1), \ldots, (x_k = m + x_0, y_k = n + y_0)\} \in \mathbb{Z}^2$$

is an (m, n)**-lattice path** if P satisfies the following for each $i = 1, 2, \ldots, k$:

$$(x_i, y_i) = (x_{i-1}, y_{i-1} + 1) \text{ or } (x_{i-1} + 1, y_{i-1}).$$

Let $LP_{(m,n)}$ denote the set of (m, n)-lattice paths. △

The following is an example of a $(2, 3)$-lattice path.

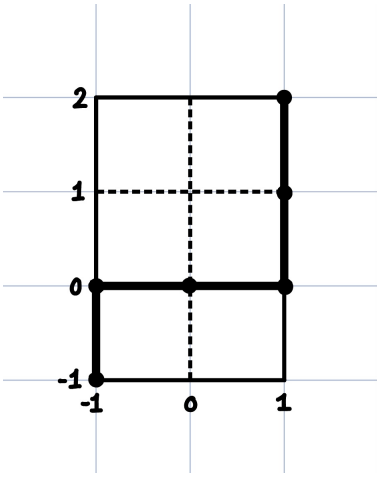

Figure 2.1.1: An example of a $(2, 3)$-lattice path.

Example 2.1.2. Consider the following path illustrated by Figure 2.1.1:

$$\{(-1,-1),(-1,0),(0,0),(1,0),(1,1),(1,2)\}.$$

Observe that

$$(-1,0) = (-1,-1+1), \text{ and}$$

$$(0,0) = (-1+1,0), \text{ and}$$

$$(1,0) = (0+1,0), \text{ and}$$

$$(1,1) = (1,0+1), \text{ and}$$

$$(1,2) = (1,1+1).$$

So, all the adjacent pairs of coordinates adhere to the restrictions in Definition 2.1.1.

Also, $(1,2) = (2+-1,3+-1)$.

Thus, the path is a $(2,3)$-lattice path. ◊

The following is an example of a path on a lattice that is not a $(2,2)$-lattice path.

Example 2.1.3. Consider the following path on a lattice illustrated by Figure 2.1.2:

$$\{(-1,-1),(-1,0),(0,0),(0,-1),(1,-1),(1,0),(1,1)\}.$$

Observe that $(0,-1) = (0,0-1)$.

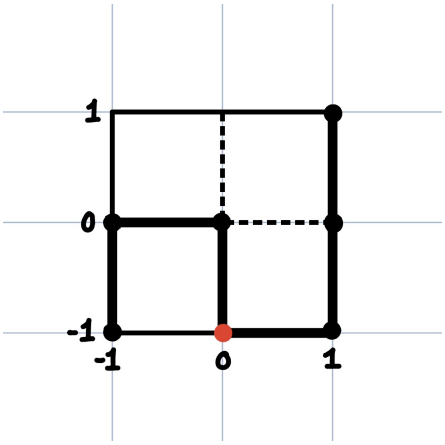

Figure 2.1.2: An example of a path on a lattice that is not a $(2, 2)$-lattice path. The flawed coordinate is emphasized by the color red.

So, not all of the adjacent pairs of coordinates adhere to the restrictions in Definition 2.1.1.

Thus, the path is not a $(2, 2)$-lattice path. ◊

2.2 Lattice Words and Counting

In this section, we also introduce *lattice words* which are words over the alphabet $\{X, Y\}$ and prove a closed formula for the number of lattice words with a certain amount for X'S and certain number of Y's.

We define two functions that output the number of X's in a word or the number of Y's in a word.

Definition 2.2.1. Let L be a word over the alphabet $\{X, Y\}$. Let $X(L)$ denote the number of X's in L and $Y(L)$ denote the number of Y's in L. △

The following definition is modified from [4].

Definition 2.2.2. [4] Let $m, n \in \mathbb{N}$. Let L be a $m + n$ length word over the alphabet $\{X, Y\}$. If $X(L) = m$ and $Y(L) = n$ we call L an (m, n)**-lattice word**. Let $LW_{(m,n)}$ denote the set of (m, n)-lattice words. △

The following is an example of a $(3, 2)$-lattice word.

Example 2.2.3. Consider the following $3 + 2 = 5$ length word over the alphabet $\{X, Y\}$: $L = XXXYY$.

Observe that $X(L) = 3$ and $Y(L) = 2$.

Thus, L is a $(3, 2)$-lattice word. ◊

The following is an example of a lattice word that is not a $(3, 2)$-lattice word.

Example 2.2.4. Consider the following $3 + 2 = 5$ length word over the alphabet $\{X, Y\}$: $XXYYY$. Denoted this word as L.

Observe that $X(L) = 2$ and $Y(L) = 3$.

Thus, L is not a $(3, 2)$-lattice word. Instead, it is a $(2, 3)$-lattice word. ◊

The following theorem is given in [10] without a proof. We provide one here.

Theorem 2.2.5. [10] *Let $m, n \in \mathbb{N}$. So, $|LW_(m, n)| = \binom{m+n}{m}$.*

Proof. Let $m, n \in \mathbb{N}$. Let $LW = a_1 a_2 \dots a_{m+n}$. Since m of the a_i's are a X and n of the a_i's are a Y, it follows that m of the a_i's can be selected out of the $m + n$ total number of a_i's to be X's while the rest are Y's in $\binom{m+n}{m}$ different ways. So, $|LW_{(m,n)}| = \binom{m+n}{m}$. □

2.3 Lattice Paths at Zero and Lattice Words Bijection

In this section, we introduce a slight variation of lattice paths called *lattice paths at zero*. Next, we prove a bijection between *lattice paths at zero* and *lattice words*.

By Definition 2.1.1, two elements of $LP_{(m,n)}$ can have a *path* with the same shape because their starting points differ. An example of such a case is illustrated in Figure 2.3.1.

As we will see later that lattice words only capture the shape of a lattice path. So, a bijection is only possible between the two if the starting point of the (m, n)-lattice paths considered is fixed. Thus, the following definition modifies the Definition 2.1.1.

Definition 2.3.1. Let $m, n \in \mathbb{N}$. an (m, n)-lattice path, denoted P, where

$$P = \{(x_0 = 0, y_0 = 0), (x_1, y_1), \dots, (x_{m+n} = m, y_{m+n} = n)\} \in (\mathbb{N} \cup \{0\})^2$$

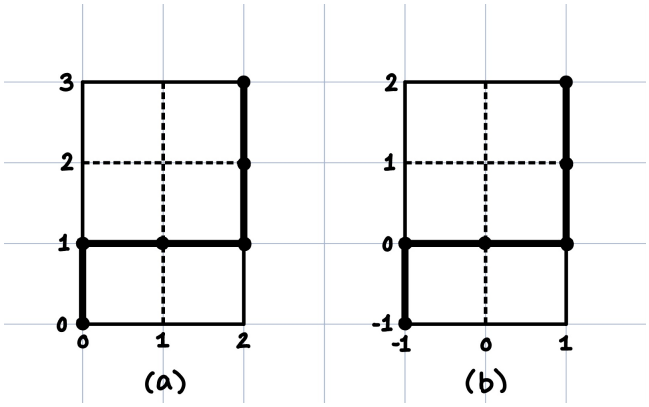

Figure 2.3.1: Observe that $(a), (b) \in LP_{(2,3)}$.

is an (m,n)**-lattice path at zero**. Let $LPZ_{(m,n)}$ denote the set of (m,n)-lattice paths at zero. △

Now observe that, in reference to Figure 2.3.1, $(a) \in LPZ_{(2,3)}$ and $(b) \notin LPZ_{(2,3)}$.

The following theorem is widely accepted, but a proof is lacking in the literature. We provide one here.

Theorem 2.3.2. *Let $m, n \in \mathbb{N}$. There is a bijection between $LPZ_{(m,n)}$ and $LW_{(m,n)}$.*

Proof. Let $m, n \in \mathbb{N}$. Let $F : LPZ_{(m,n)} \longrightarrow LW_{(m,n)}$ be a function defined by $F(\{(x_0 = 0, y_0 = 0), (x_1, y_1), \ldots, (x_{m+n} = m, y_{m+n} = n)\}) = a_1 a_2 \ldots a_{m+n}$ where

$$a_i = \begin{cases} X & \text{if } (x_i, y_i) = (x_{i-1} + 1, y_{i-1}) \\ Y & \text{if } (x_i, y_i) = (x_{i-1}, y_{i-1} + 1). \end{cases}$$

Note that $a_1 a_2 \ldots a_{m+n}$ is an (m,n)-lattice word since from $(0,0)$ to (m,n) there must be m adjacent pairs of coordinates where $(x_i, y_i) = (x_{i-1}+1, y_{i-1})$ and there must be n adjacent pairs of coordinates where $(x_i, y_i) = (x_{i-1}, y_{i-1}+1)$. So, there are m number of X's and n number of Y's in $a_1 a_2 \ldots a_{m+n}$. So, $a_1 a_2 \ldots a_{m+n} \in LW_{(m,n)}$.

First, we prove F is injective.

Let $LPZ = \{(x_0 = 0, y_0 = 0), (x_1, y_1), \ldots, (x_{m+n}, y_{m+n})\}, LPZ' = \{(x'_0 = 0, y'_0 = 0), (x'_1, y'_1), \ldots, (x'_{m+n}, y'_{m+n})\} \in LPZ_{(m,n)}$.

Assume $F(LPZ) = F(LPZ') = a_1, a_2 \ldots a_{m+n}$.

By construction $(x_0, y_0) = (0,0) = (x'_0, y'_0)$.

This is our base cases.

Let $z \in \{1, 2 \ldots, m+n-1\}$. Assume $(x_z, y_z) = (x'_z, y'_z)$.

There are now two cases.

For case one, assume $a_{z+1} = X$. So, by F it follows that $(x_{z+1}, y_{z+1}) = (x_z + 1, y_z)$ and $(x'_{z+1}, y'_{z+1}) = (x'_z + 1, y'_z)$. Since $(x_z, y_z) = (x'_z, y'_z)$ then $(x_{z+1}, y_{z+1}) = (x_z + 1, y_z) = (x'_z + 1, y'_z) = (x'_{z+1}, y'_{z+1})$.

For case two, assume $a_{z+1} = Y$. So, by F it follows that $(x_{z+1}, y_{z+1}) = (x_z, y_z + 1)$ and $(x'_{z+1}, y'_{z+1}) = (x'_z, y'_z + 1)$. Since $(x_z, y_z) = (x'_z, y'_z)$ then $(x_{z+1}, y_{z+1}) = (x_z, y_z + 1) = (x'_z, y'_z + 1) = (x'_{z+1}, y'_{z+1})$.

Thus, by the principle of mathematical induction, for all $z \in \{1, \ldots, m+n\}$ it is true that $(x_z, y_z) = (x'_z, y'_z)$.

So, $LPZ = LPZ'$.

Therefore, F is injective.

Now, we will prove F is surjective.

Let $LW = a_1 a_2 \ldots a_{m+n} \in LW_{(m,n)}$.

Construct an (m,n)-lattice path $LPZ = \{(x_0 = 0, y_0 = 0), (x_1, y_1), \ldots, (x_{m+n}, y_{m+n})\}$ where

$$(x_i, y_i) = \begin{cases} (x_{i-1} + 1, y_{i-1}) & \text{if } a_i = X \\ (x_{i-1}, y_{i-1} + 1) & \text{if } a_i = Y \end{cases}$$

Note that $\{(x_0 = 0, y_0 = 0), (x_1, y_1), \ldots, (x_{m+n}, y_{m+n})\}$ is an (m,n)-lattice paths at zero since $a_1 a_2 \ldots a_{m+n}$ must have m number of X's and n number of Y's. So, from $(0,0)$ to (m,n) there must be m adjacent pairs of coordinates where $(x_i, y_i) = (x_{i-1} + 1, y_{i-1})$ and there must be n adjacent pairs of coordinates where $(x_i, y_i) = (x_{i-1}, y_{i-1} + 1)$. So, $\{(x_0 = 0, y_0 = 0), (x_1, y_1), \ldots, (x_{m+n}, y_{m+n})\} \in LPZ_{(m,n)}$.

Now we compute $F(LPZ) = a'_1 a'_2 \ldots a'_{m+n}$.

There are now two cases.

For case one $a_i = X$. Then, by construction of LPZ, $(x_i, y_i) = (x_{i-1} + 1, y_{i-1})$. Next, by F, $a'_i = X$. So, $a_i = a'_i$.

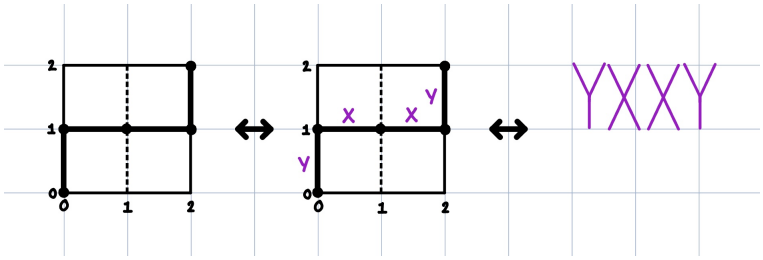

Figure 2.3.2: An example illustrating Theorem 2.3.2 using a $(2,2)$-lattice path at zero and a $(2,2)$-lattice word.

For case two $a_i = Y$. Then, by construction of LPZ, $(x_i, y_i) = (x_{i-1}, y_{i-1} + 1)$. Next, by F, $a'_i = Y$. So, $a_i = a'_i$.

Thus, $F(LPZ) = a'_1 a'_2 \dots a'_{m+n} = a_1 a_2 \dots a_{m+n} = LW$.

Therefore, F is surjective.

Hence, there is a bijection between $LPZ_{(m,n)}$ and $LW_{(m,n)}$. □

In Figure 2.3.2, we give an example illustrating Theorem 2.3.2 using a $(2,2)$-lattice path at zero and a $(2,2)$-lattice word.

3
Dyck Paths and Dyck Words

In this chapter, we introduce *Dyck paths* which are lattice paths at zero under a positively sloped diagonal that goes through the origin. We give a closed formula for the number of Dyck paths with a certain final coordinate. We also introduce *Dyck words* which are lattice words with an added restriction on all prefixes. Next, we prove a bijection between *Dyck paths* and *Dyck words*.

3.1 Dyck Paths and Counting

In this section, we introduce *Dyck paths* which are lattice paths at zero under a positively sloped diagonal that goes through the origin. We give a closed formula for the number of Dyck paths with a certain final coordinate.

The following definition is modified from [10].

Definition 3.1.1. [10] Let $n, k, x, y \in \mathbb{N}$. If an (n, kn)-lattice path at zero denoted P lies in the domain $y \leq k \cdot x$, we call P an (n, kn)**-Dyck path**. Let $DP_{(n,kn)}$ denote the set of (n, kn)-Dyck paths. △

The following is an example of a $(2, 4)$-Dyck path.

Example 3.1.2. Consider the following $(2, 4)$-lattice path at zero illustrated by 3.1.1:

$$\{(0, 0), (1, 0), (1, 1), (1, 2), (2, 2), (2, 3), (2, 4)\}.$$

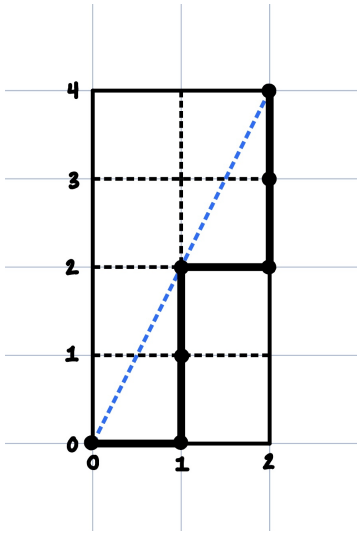

Figure 3.1.1: Example of a $(2,4)$-Dyck path.

The $(2,4)$-lattice path at zero lies in the domain $y \leq 2 \cdot x$ as illustrated by 3.1.1.

Hence, the $(2,4)$-lattice path at zero is a $(2,4)$-Dyck path. ◊

The following is an example of a $(2,4)$-lattice path at zero that is not a $(2,4)$-Dyck path.

Example 3.1.3. Consider the following $(2,4)$-lattice path at zero illustrated by 3.1.2:

$$\{(0,0),(1,0),(1,1),(1,2),(1,3),(2,3),(2,4)\}.$$

The $(2,4)$-lattice path at zero does not lie in the domain $y \leq 2 \cdot x$ as illustrated by 3.1.2.

Hence, the $(2,4)$-lattice path at zero is not a $(2,4)$-Dyck path. ◊

Remark 3.1.4. [10] Let C_n denote the nth Catalan number which is $\frac{1}{n+1}\binom{2n}{n}$. Then $DP_{(n,n)} = C_n$. ◊

The following proposition is proved in [10].

Proposition 3.1.5. [10] *Let* $n, k \in \mathbb{N}$. *Then* $|DP_{(n,kn)}| = \frac{1}{kn+1}\binom{(k+1)n}{n}$.

3.2 Dyck Words

In this section, we introduce *Dyck words* which are lattice words with an added restriction on all prefixes.

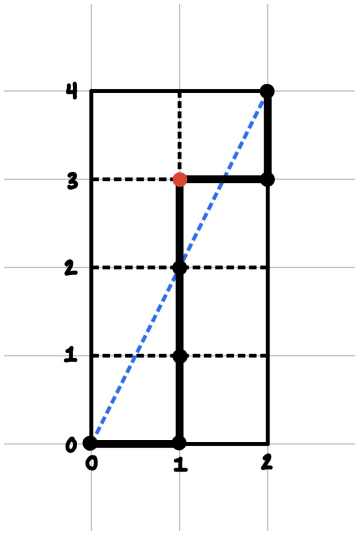

Figure 3.1.2: An example of a $(2, 4)$-lattice path at zero that is not a $(2, 4)$-Dyck path. The flawed coordinate is emphasized by the color red.

The following definition is modified from [4].

Definition 3.2.1. Let $n, k \in \mathbb{N}$. An (n, kn)**-Dyck word** is an (n, kn)-lattice word that has no prefix, denoted D_p, where $k \cdot X(D_p) < Y(D_p)$. Let $DW_{(n,kn)}$ denote the set of (n, kn)-Dyck words. △

The following is an example of a $(2, 2 \cdot 2)$-Dyck word.

Example 3.2.2. Consider the following $(2, 2 \cdot 2)$-lattice word: $XYYXYY$.

Observe that

$$2 \cdot X(X) \not< Y(X), \text{ and}$$

$$2 \cdot X(XY) \not< Y(XY), \text{ and}$$

$$2 \cdot X(XYY) \not< Y(XYY), \text{ and}$$

$$2 \cdot X(XYYX) \not< Y(XYYX), \text{ and}$$

$$2 \cdot X(XYYXY) \not< Y(XYYXY).$$

Thus, $XYYXYY$ is a $(2, 2 \cdot 2)$-Dyck word. ◊

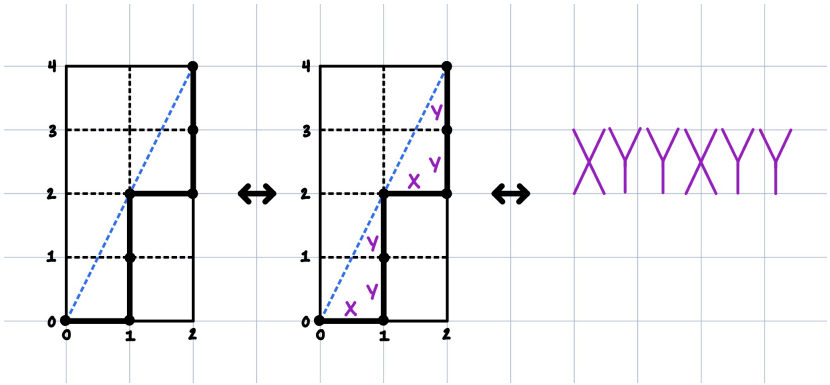

Figure 3.3.1: An example illustrating Theorem 3.3.1 using a $(2, 2 \cdot 2)$-Dyck path at zero and a $(2, 2 \cdot 2)$-Dyck word.

The following is an example of a $(2, 2 \cdot 2)$-lattice word that is not a $(2, 2 \cdot 2)$-Dyck word.

Example 3.2.3. Consider the following $(2, 2 \cdot 2)$-lattice word: $XYYYXY$.

Observe that $2 \cdot X(XYYY) < Y(XYYY)$.

Thus, $XYYYXY$ is not a $(2, 2 \cdot 2)$-Dyck word. ◊

3.3 Dyck Paths and Dyck Words Bijection

In this section, we prove a bijection between *Dyck paths* and *Dyck words*.

The following theorem is widely accepted, but a proof is lacking in the literature.

Theorem 3.3.1. *Let $n, k \in \mathbb{N}$. There is a bijection between $DP_{(n,kn)}$ and $DW_{(n,kn)}$.*

Proof. Similar to the proof for Theorem 2.3.2. □

In Figure 3.3.1, we give an example illustrating Theorem 3.3.1 using a $(2, 2 \cdot 2)$-Dyck path and a $(2, 2 \cdot 2)$-Dyck word.

By definition of bijection, the next corollary follows.

Corollary 3.3.2. *Let $m, n \in \mathbb{N}$. Then $|DP_{(n,kn)}| = |DW_{(n,kn)}|$.*

Proof. Let $n, k \in \mathbb{N}$. By Theorem 3.3.1, there is a bijection between $DP_{(n,kn)}$ and $DW_{(n,kn)}$. By definition of bijection, $|DP_{(n,kn)}| = |DW_{(n,kn)}|$. □

The next corollary follows.

Corollary 3.3.3. *Let $n, k \in \mathbb{N}$. Then $|DW_{(n,kn)}| = \frac{1}{kn+1}\binom{(k+1)n}{n}$.*

Proof. By Corollary 3.3.2, $|DP_{(n,kn)}| = |DW_{(n,kn)}|$. By Theorem 3.1.5, $|DP_{(n,kn)}| = \frac{1}{kn+1}\binom{(k+1)n}{n}$. Thus, $|DW_{(n,kn)}| = |DP_{(n,kn)}| = \frac{1}{kn+1}\binom{(k+1)n}{n}$. □

4
Parking Functions

In this chapter we introduce sequences of natural numbers that represent the preferred parking space out of a one-way street for a certain amount of cars. *Parking functions* are the sequences which lead to all cars parking in all parking spaces available on a street given a systematic way of parking. In addition, we give the known recursive and closed formulas for the number of parking functions given a certain number of cars. Then we will introduce a slight variation of parking functions called *descending parking functions*. Next, we provide an argument for a bijection between *descending parking functions* and *Dyck words*.

4.1 Parking Functions and Counting

In this section, we build up to a formal definition of a *parking function*. In addition, we give the known recursive and closed formulas for the number of parking functions given a certain number of cars.

The folowing set up and definitions are modified from [1, 2].

Remark 4.1.1. For $n \in \mathbb{N}$, let $[n] = \{1, \ldots, n\}$. $\Diamond$

Consider n parking spaces on a one-way street arranged in a line numbered 1 to n from west to east as illustrated in Figure 4.1.1.

Suppose there are $n \in \mathbb{N}$ cars, denoted $c_1, c_2, c_3, \ldots, c_n$, as illustrated in Figure 4.1.2.

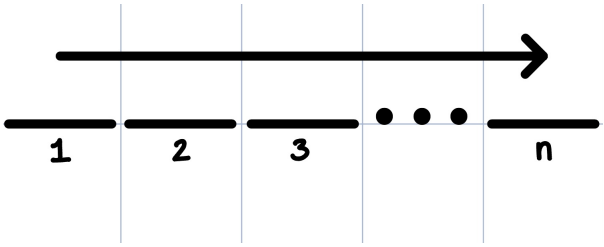

Figure 4.1.1: n parking spaces on a one-way street arranged in a line numbered 1 to n from west to east.

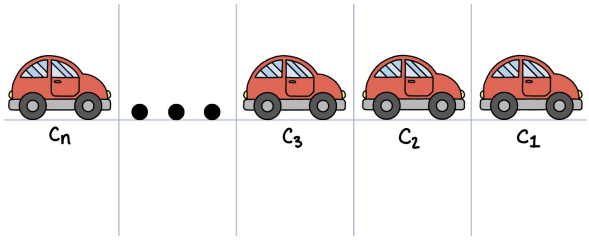

Figure 4.1.2: n cars denoted $c_1, c_2, c_3, \ldots, c_n$.

For all $i \in [n]$, each car c_i has a non-distinct parking space preference, denoted $a_i \in [n]$. This leads us to the following definition.

Definition 4.1.2. Let $n \in \mathbb{N}$. Let $A = (a_1, a_2, a_3, \ldots, a_n) \in [n]^n$. Then A is an **n-parking space preference sequence**. Let $PSPS_n$ be the set of n-parking space preference sequences. △

A closed formula is given for the number of n-parking space preference sequences, but a formal proof is lacking in literature. We provide one here.

Proposition 4.1.3. *Let $n \in \mathbb{N}$. Then $|PSPS_n| = n^n$.*

Proof. Let $n \in \mathbb{N}$. Observe that by Definition 4.1.2, $PSPS_n = [n]^n$. So, $|PSPS_n| = |[n]^n|$. Thus, $|PSPS_n| = |[n]^n| = n^n$. □

Parking Rules: For all $i \in [n]$ and in increasing order, car c_i starts at parking space 1 and drives toward its preferred parking space a_i. If a_i is unoccupied, then c_i parks. Otherwise, c_i proceeds forward until it reaches the next available parking space. If every parking space numbered from a_i up to and including n is taken, then c_i is unable to park. Figure 4.1.3 illustrates the scene before parking attempts commence.

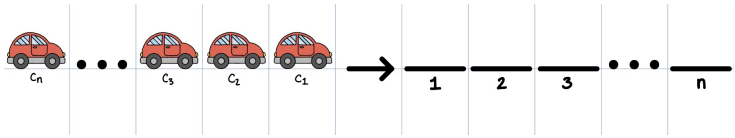

Figure 4.1.3: The scene before parking attempts commence.

Remark 4.1.4. Let $n \in \mathbb{N}$. Let $A = (a_1, a_2, a_3, \ldots, a_n)$ be an n-parking space preference sequence. From the parking rules it is evident that if parking space $i \in [n]$ is occupied after all cars have parked then there must exist some $a_j \in A$ where $j \in [n]$ such that $a_j \leq i$. ◊

We are now ready to define a *parking function*.

Definition 4.1.5. Let $n \in \mathbb{N}$. Let $A = (a_1, a_2, a_3, \ldots, a_n)$ be an n-parking space preference sequence. Let $B = (b_1, b_2, b_3, \ldots, b_n)$ be a permutation of A where $b_1 \leq b_2 \leq b_3 \leq \ldots \leq b_n$. If $b_i \leq i$ for all $i \in [n]$ then A is an **n-parking function**. Let PF_n denote the set of n-parking functions. △

The following is an example of a 3-parking function.

Example 4.1.6. Consider the following 3-parking space preference sequence: $A = (2, 2, 1)$. Observe that $B = (b_1 = 1, b_2 = 2, b_3 = 2)$ is a permutation of A where $b_1 \leq b_2 \leq b_3$. Furthermore,

$$b_1 = 1 \leq 1, \text{ and}$$

$$b_2 = 2 \leq 2, \text{ and}$$

$$b_3 = 2 \leq 3.$$

Hence, A is a 3-parking function. ◊

The following steps demonstrate why $(2, 2, 1)$ is a 3-parking function visually:

1. Observe that car c_1 prefers parking space 2 as given by $(\underline{2}, 2, 1)$. Parking space 2 is available so c_1 parks in space 2 as illustrated by Figure 4.1.4.

2. Observe that car c_2 also prefers parking space 2 as given by $(2, \underline{2}, 1)$. Parking space 2 is not available so c_2 must continue forward to find the next empty space to park in which is 3 as illustrated by Figure 4.1.5.

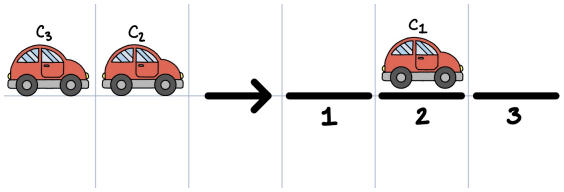

Figure 4.1.4: c_1 parks in space 2.

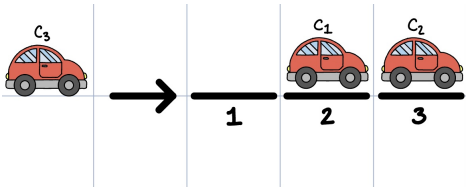

Figure 4.1.5: c_2 parks in space 3.

3. Observe that car c_3 prefers parking space 1 as given by $(2, 2, \underline{1})$. Parking space 1 is available so c_3 parks in space 1 as illustrated by Figure 4.1.6.

The following is an example of a 3-parking space preference sequence that is not a 3-parking function.

Example 4.1.7. Consider the following 3-parking space preference sequence: $A = (3, 1, 3)$. Observe that $B = (b_1 = 1, b_2 = 3, b_3 = 2)$ is a permutation of A where $b_1 \leq b_2 \leq b_3$. Furthermore, $b_2 = 3 \not\leq 2$.

Hence, A is not a 3-parking function. ◊

The following steps demonstrate why $(3, 1, 3)$ is not a 3-parking function visually:

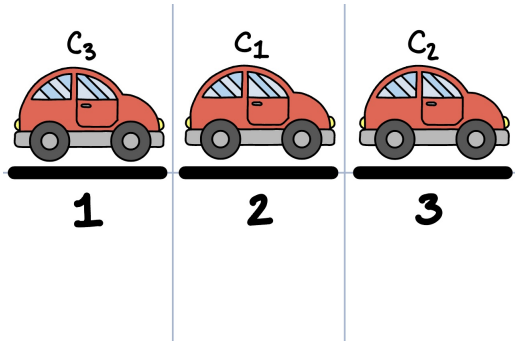

Figure 4.1.6: c_3 parks in space 1.

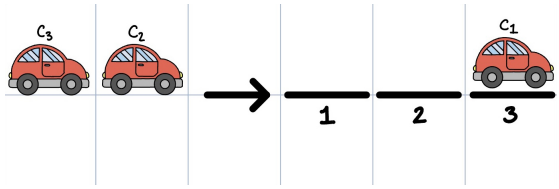

Figure 4.1.7: c_1 parks in space 3.

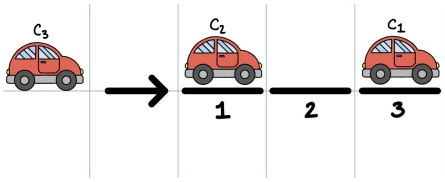

Figure 4.1.8: c_2 parks in space 1.

1. Observe that c_1 prefers parking space 3 as given by $(\underline{3}, 1, 3)$. Parking space 3 is available so c_1 parks in space 3 as illustrated by Figure 4.1.7.

2. Observe that c_2 prefers parking space 1 as given by $(3, \underline{1}, 3)$. Parking space 1 is available so c_2 parks in space 1 as illustrated by Figure 4.1.8.

3. Observe that c_3 also prefers parking space 3 as given by $(3, 1, \underline{3})$. Parking space 3 is not available so c_3 must continue forward to find an empty space to park in, but falls off the edge before doing so as illustrated by Figure 4.1.9.

Let $n \in \mathbb{N}$. Recursive and closed formulas for the number of n-parking functions are known. The following theorem gives a recursive formula for the number of n-parking functions. A thorough explanation can be found in [7].

Theorem 4.1.8. *Let $n \in \mathbb{N}$. Then,*

$$|PF_n| = \sum_{i=1}^{n} i \binom{n-1}{i-1} |PF_{i-1}| \cdot |PF_{n-i}|.$$

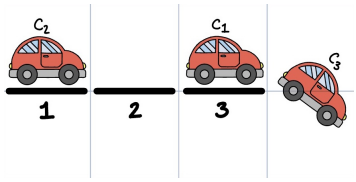

Figure 4.1.9: c_3 doesn't park.

The next theorem gives a closed formula for the number of n-parking functions. A thorough explanation can be found in [8].

Theorem 4.1.9. *Let $n \in \mathbb{N}$. Then, $|PF_n| = (n+1)^{n-1}$.*

4.2 Descending Parking Functions and Dyck Words Bijection

In this section, we introduce a slight variation of parking functions called *descending parking functions*. Next, we provide an argument for a bijection between *descending parking functions* and *Dyck words*.

We now define descending parking functions.

Definition 4.2.1. Let $n \in \mathbb{N}$. Let $A = (a_1, a_2, a_3, \ldots, a_n)$ be an n-parking function. If $a_1 \geq a_2 \geq a_3 \geq \ldots \geq a_n$, then A is an **n-descending parking function**. Let DPF_n be the set of n-parking functions. △

The following theorem is often claimed and explained, such as in [3], but without proof.

Theorem 4.2.2. *Let $n, k \in \mathbb{N}$. There is a bijection between $DW_{(n,n)}$ and DPF_n.*

We will demonstrate the proof of this theorem by defining two functions and using an example to support them being inverses of one another. The example can then be generalized, but we will not include the details here.

We will use the following function as part our first function.

Definition 4.2.3. Let $n \in \mathbb{N}$. Let P be an n-descending parking function. Let $i \in [n]$. Let $N(P, i)$ denote the number of elements in P equal to i. △

We now define our first function where the input is an n-descending parking functions and the output is an (n, n)-Dyck word.

Definition 4.2.4. Let $n \in \mathbb{N}$. Let $F : DPF_n \longrightarrow DW_{(n,n)}$ be a function defined by $F(P = (p_0, p_1, \ldots, p_{n-1})) = Xb_nXb_{n-1}Xb_{n-2}X \ldots Xb_1)$ where b_i is an $N(P, i)$-length word over the alphabet $\{Y\}$. △

We now define a second function where the input is an (n, n)-Dyck word and the output is an n-descending parking function.

Recall from Definition 2.2.1 that if L is a word over the alphabet $\{X, Y\}$, then $X(L)$ denotes the number of X's in L.

Definition 4.2.5. Let $n \in \mathbb{N}$. Let $G : DW_{(n,n)} \longrightarrow DPF_n$ be a function defined by $G(a_0a_1 \dots a_{n+n-1}) = (p_0, p_1, \dots, p_{n+n-1})$ where

$$p_i = \begin{cases} \text{empty} & \text{if } a_i = X \\ n + 1 - X(a_i) & \text{if } a_i = Y. \end{cases}$$

△

We now demonstrate how these two functions are inverses through two examples.

Let $n \in \mathbb{N}$. Let $P \in DPF_n$. The first is an example of how $G(F(P)) = P$.

Example 4.2.6. Consider the following 3-descending parking function: $P = (2, 2, 1)$. Observe that $F((2, 2, 1)) = XXYYXY$. Furthermore, $G(XXYYXY) = (2, 2, 1)$.

Hence, $G(F(P)) = P$. ◊

Let $n \in \mathbb{N}$. Let $A \in DW_{(n,n)}$. The following is an example of how $F(G(A)) = A$.

Example 4.2.7. Consider the following $(3, 3)$-Dyck word: $A = XYXXYY$. Observe that $G(XYXXYY) = (3, 1, 1)$. Furthermore, $F((3, 1, 1)) = XYXXYY$.

Hence, $F(G(A)) = A$. ◊

By definition of bijection, the next corollary follows.

Corollary 4.2.8. *Let $n \in \mathbb{N}$. Then $|DPF_n| = |DW_{(n,n)}|$.*

Proof. Let $n \in \mathbb{N}$. By Theorem 4.2.2, there is a bijection between DPF_n and $DW_{(n,n)}$. By definition of bijection, $|DPF_n| = |DW_{(n,n)}|$. □

The next corollary follows.

Corollary 4.2.9. *Let $n \in \mathbb{N}$. Then $|DPF_n| = \frac{1}{n+1}\binom{2n}{n}$.*

Proof. By Corollary 4.2.8, $|DPF_n| = |DW_{(n,n)}|$. By Corollary 3.3.3, $|DW_{(n,n)}| = \frac{1}{n+1}\binom{2n}{n}$. Thus, $|DPF_n| = |DW_{(n,n)}| = \frac{1}{n+1}\binom{2n}{n}$. □

5
Parking Garage Functions

In this chapter we introduce a new concept called *parking garage functions* which are a generalization of *parking functions*. *Parking garage functions* are the sequences which lead to all cars parking in all parking spaces available on each level in a parking garage given a systematic way of parking. We developed python code which outputs a sequence. This led us to paper from 1977, [5], which gives us a recursive formula for a scenario that can be interpreted as parking garage functions. Then we will introduce a slight variation of parking garage functions called *descending parking garage functions*. Finally, we provide an argument for a bijection between *descending parking garage functions* and *Dyck words*.

5.1 Introduction to Parking Garage Functions

In this section, we build up to a formal definition of a *parking garage function*.

The folowing set up and definitions are modified from that of *parking functions* in Section 4.1.

Remark 5.1.1. For $n, k \in \mathbb{N}$, let $[kn] = \{1, \ldots, kn\}$. $\Diamond$

Consider n parking garage levels with capacity k in an upwardly one-way parking garage numbered 1 to n from bottom to top as illustrated in Figure 5.1.1.

Suppose there are $kn \in \mathbb{N}$ cars, denoted $c_1, c_2, c_3, \ldots, c_{kn}$, as illustrated in Figure 5.1.2.

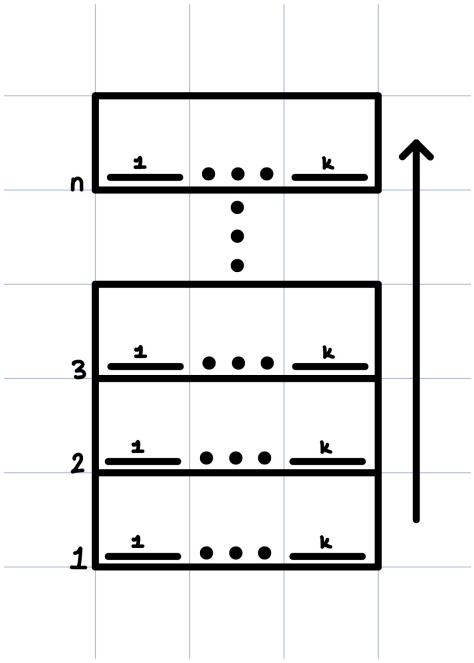

Figure 5.1.1: n parking garage levels with capacity k in an upwardly one-way parking garage numbered 1 to n from bottom to top.

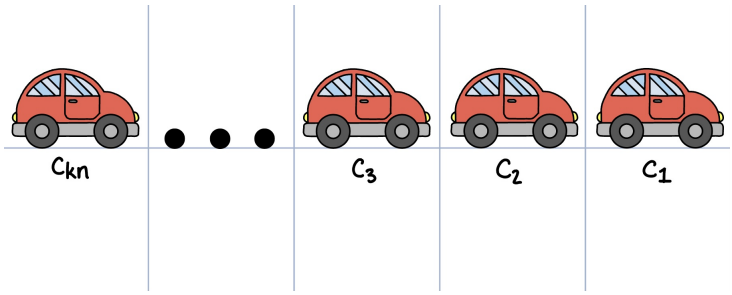

Figure 5.1.2: kn cars denoted $c_1, c_2, c_3, \ldots, c_{kn}$.

For all $i \in [kn]$, each car c_i has a non-distinct parking garage level preference, denoted $a_i \in [n]$. This leads us to the following definition.

Definition 5.1.2. Let $n \in \mathbb{N}$. Let $A = (a_1, a_2, a_3, \ldots, a_{kn}) \in [n]^{kn}$. Then A is an (n, kn)-**parking garage level preference sequence**. Let $PGLPS_{(n,kn)}$ be the set of (n, kn)-parking garage level preference sequences. △

The following is a closed formula for the number of (n, kn)-parking garage level preference sequences.

Proposition 5.1.3. *Let $n \in \mathbb{N}$. Then $|PGLPS_{(n,kn)}| = n^{kn}$.*

Proof. Let $n \in \mathbb{N}$. Observe that by Definition 5.1.2, $PGLPS_{(n,kn)} = [n]^{kn}$. So, $|PGLPS_{(n,kn)}| = |[n]^{kn}|$. Thus, $|PGLPS_{(n,kn)}| = |[n]^{kn}| = n^{kn}$. □

Parking Rules: For all $i \in [kn]$ and in increasing order, car c_i starts at parking garage level 1 and drives upwards to its preferred parking garage level a_i. If a_i is not at full capacity, k, then c_i parks. Otherwise, c_i proceeds upwards until it reaches the next parking garage level not at full capacity, k. If every parking garage level numbered from a_i up to and including n is at full capacity, k, then c_i is unable to park. Figure 5.1.3 illustrates the scene before parking attempts commence.

Remark 5.1.4. Let $n, k \in \mathbb{N}$. Let $A = (a_1, a_2, a_3, \ldots, a_{kn})$ be an (n, kn)-parking garage level preference sequence. From the parking rules it is evident that if parking garage level $i \in [n]$ has a car parked on it after all cars have parked then there must exist some $a_j \in A$ where $j \in [kn]$ such that $a_j \leq \lceil \frac{i}{k} \rceil$. ◊

We are now ready to define a *parking garage function*.

Definition 5.1.5. Let $n \in \mathbb{N}$. Let $A = (a_1, a_2, a_3, \ldots, a_{kn})$ be an (n, kn)-parking garage level preference sequence. Let $B = (b_1, b_2, b_3, \ldots, b_{kn})$ be a permutation of A where $b_1 \leq b_2 \leq b_3 \leq \ldots \leq b_{kn}$. If $b_i \leq \lceil \frac{i}{k} \rceil$ for all $i \in [kn]$ then A is an (n, kn)-**parking garage function**. Let $PGF_{(n,kn)}$ denote the set of (n, kn)-parking garage functions. △

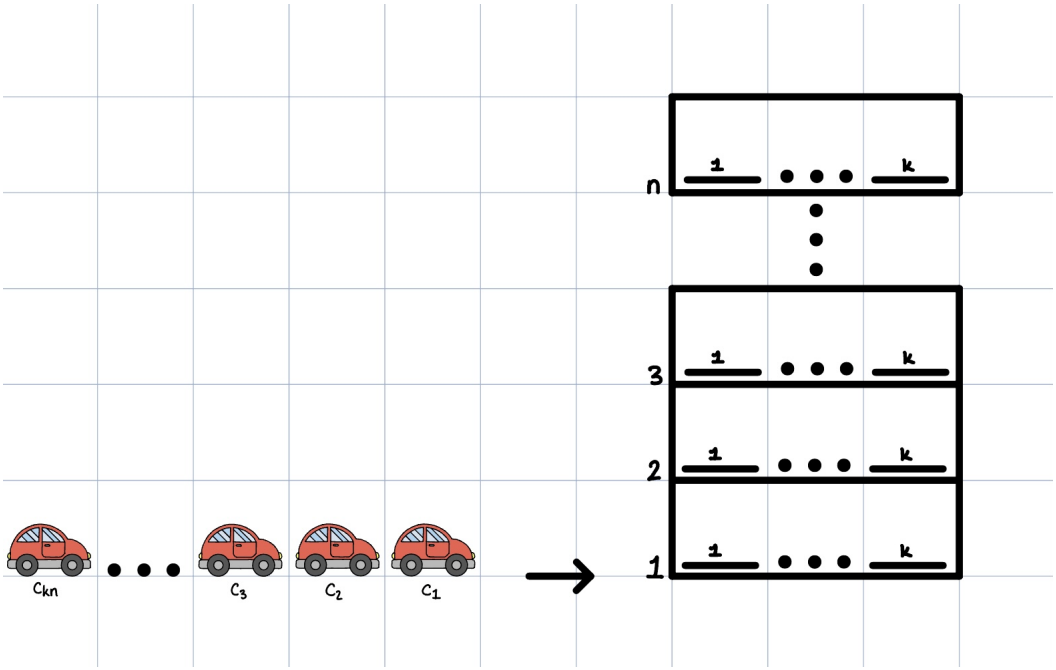

Figure 5.1.3: The scene before parking attempts commence.

Remark 5.1.6. n-parking functions are the same as (n, kn)-parking garage function where $k = 1$. ◊

The following is an example of a $(2, 3 \cdot 2)$-parking garage function.

Example 5.1.7. Consider the following $(2, 3 \cdot 2)$-parking garage level preference sequence: $A = (2, 2, 1, 1, 1, 1)$. Observe that $B = (b_1 = 1, b_2 = 1, b_3 = 1, b_4 = 1, b_5 = 2, b_6 = 2)$ is a permutation of A where $b_1 \leq b_2 \leq b_3 \leq b_4 \leq b_5 \leq b_6$. Furthermore,

$$b_1 = 1 \leq 1 = \left\lceil \frac{1}{3} \right\rceil, \text{ and}$$
$$b_2 = 1 \leq 1 = \left\lceil \frac{2}{3} \right\rceil, \text{ and}$$
$$b_3 = 1 \leq 1 = \left\lceil \frac{3}{3} \right\rceil, \text{ and}$$
$$b_4 = 1 \leq 2 = \left\lceil \frac{4}{3} \right\rceil, \text{ and}$$
$$b_5 = 2 \leq 2 = \left\lceil \frac{5}{3} \right\rceil, \text{ and}$$
$$b_6 = 2 \leq 2 = \left\lceil \frac{6}{3} \right\rceil.$$

Hence, A is a $(2, 3 \cdot 2)$-parking garage function. ◊

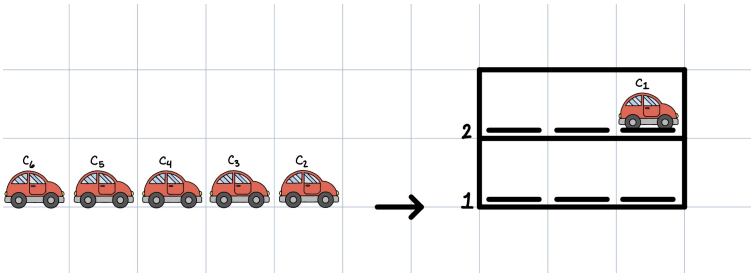

Figure 5.1.4: c_1 parks on level 2.

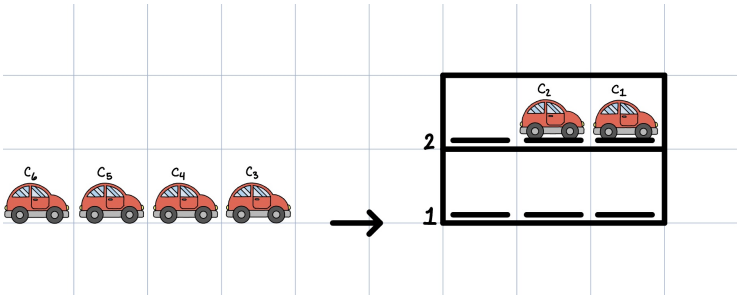

Figure 5.1.5: c_2 parks on level 2.

The following steps demonstrate why $(2, 2, 1, 1, 1, 1)$ is a $(2, 3 \cdot 2)$-parking garage function visually:

1. Observe that car c_1 prefers parking garage level 2 as given by $(\underline{2}, 2, 1, 1, 1, 1)$. Parking garage level 2 is not at full capacity so c_1 parks on level 2 as illustrated by Figure 5.1.4.

2. Observe that car c_2 prefers parking garage level 2 as given by $(2, \underline{2}, 1, 1, 1, 1)$. Parking garage level 2 is not at full capacity so c_2 parks on level 2 as illustrated by Figure 5.1.5.

3. Observe that car c_3 prefers parking garage level 1 as given by $(2, 2, \underline{1}, 1, 1, 1)$. Parking garage level 1 is not at full capacity so c_3 parks on level 1 as illustrated by Figure 5.1.6.

4. Observe that car c_4 prefers parking garage level 1 as given by $(2, 2, 1, \underline{1}, 1, 1)$. Parking garage level 1 is not at full capacity so c_4 parks on level 1 as illustrated by Figure 5.1.7.

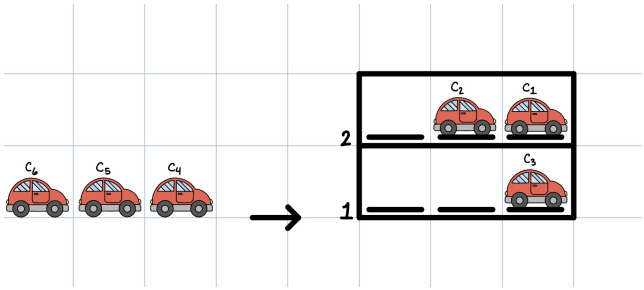

Figure 5.1.6: c_3 parks on level 1.

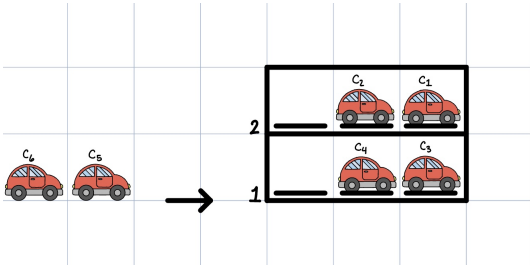

Figure 5.1.7: c_4 parks on level 1.

5. Observe that car c_5 prefers parking garage level 1 as given by $(2, 2, 1, 1, \underline{1}, 1)$. Parking garage level 1 is not at full capacity so c_5 parks on level 1 as illustrated by Figure 5.1.8.

6. Observe that car c_6 also prefers parking garage level 1 as given by $(2, 2, 1, 1, 1, \underline{1})$. Parking garage level 1 is at full capacity so c_5 must continue upward to find the next level not at full capacity to park on which is 2 as illustrated by Figure 5.1.9.

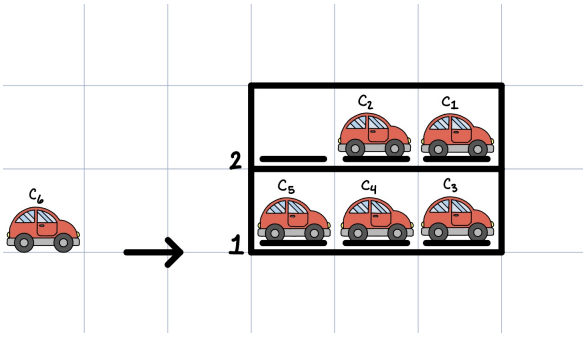

Figure 5.1.8: c_5 parks on level 1.

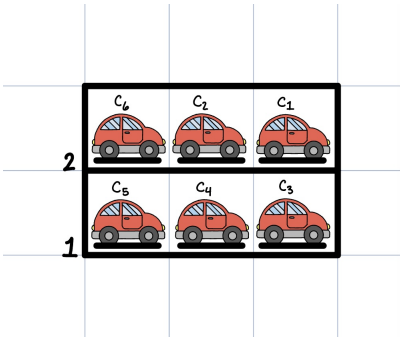

Figure 5.1.9: c_6 parks on level 2.

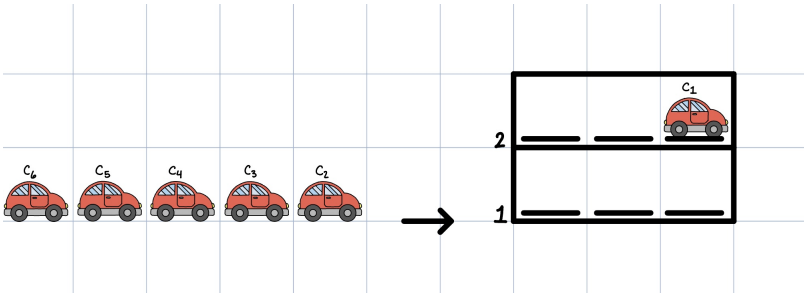

Figure 5.1.10: c_1 parks on level 2.

The following is an example of a $(2, 3 \cdot 2)$-parking garage level preference sequence that is not a $(2, 3 \cdot 2)$-parking garage function.

Example 5.1.8. Consider the following $(2, 3 \cdot 2)$-parking garage level preference sequence: $A = (2, 2, 2, 1, 1, 2)$. Observe that $B = (b_1 = 1, b_2 = 1, b_3 = 2, b_4 = 2, b_5 = 2, b_6 = 2)$ is a permutation of A where $b_1 \leq b_2 \leq b_3 \leq b_4 \leq b_5 \leq b_6$.

Furthermore, $b_3 = 2 \not\leq 1 = \lceil \frac{3}{3} \rceil$.

Hence, A is not a $(2, 3 \cdot 2)$-parking garage function. ◊

The following steps demonstrate why $(2, 2, 2, 1, 1, 2)$ is not a $(2, 3 \cdot 2)$-parking garage function visually:

1. Observe that car c_1 prefers parking garage level 2 as given by $(\underline{2}, 2, 2, 1, 1, 2)$. Parking garage level 2 is not at full capacity so c_1 parks on level 2 as illustrated by Figure 5.1.10.

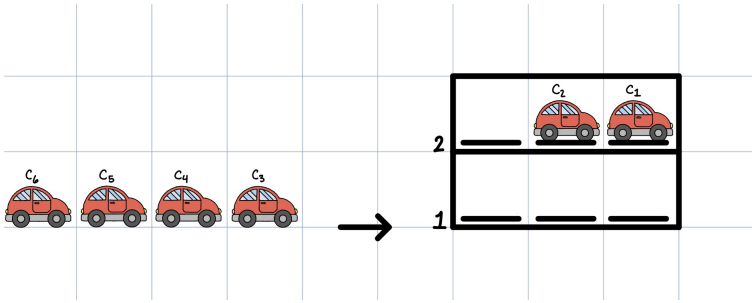

Figure 5.1.11: c_2 parks on level 2.

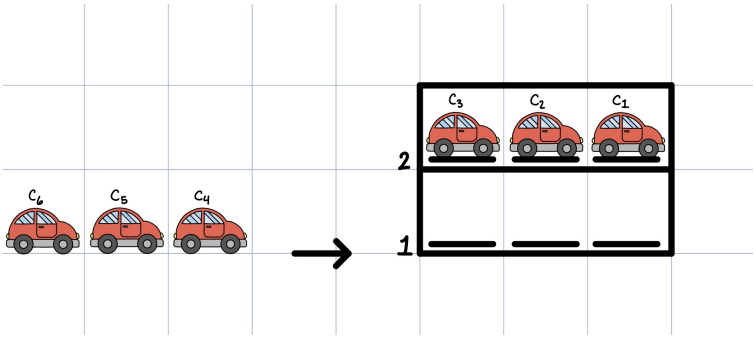

Figure 5.1.12: c_3 parks on level 2.

2. Observe that car c_2 prefers parking garage level 2 as given by $(2, \underline{2}, 2, 1, 1, 2)$. Parking garage level 2 is not at full capacity so c_2 parks on level 2 as illustrated by Figure 5.1.11.

3. Observe that car c_3 prefers parking garage level 2 as given by $(2, 2, \underline{2}, 1, 1, 2)$. Parking garage level 2 is not at full capacity so c_3 parks on level 2 as illustrated by Figure 5.1.12.

4. Observe that car c_4 prefers parking garage level 1 as given by $(2, 2, 2, \underline{1}, 1, 2)$. Parking garage level 1 is not at full capacity so c_4 parks on level 1 as illustrated by Figure 5.1.13.

5. Observe that car c_5 prefers parking garage level 1 as given by $(2, 2, 2, 1, \underline{1}, 2)$. Parking garage level 1 is not at full capacity so c_5 parks on level 1 as illustrated by Figure 5.1.14.

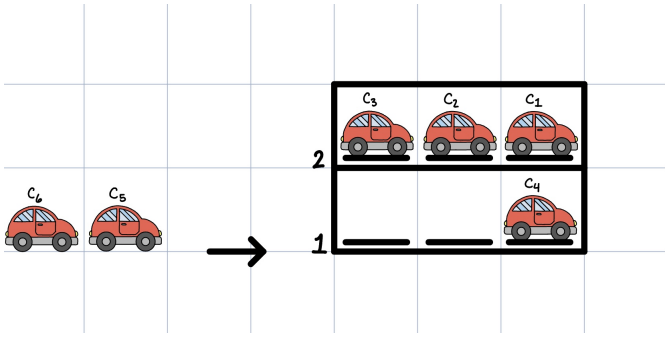

Figure 5.1.13: c_4 parks on level 1.

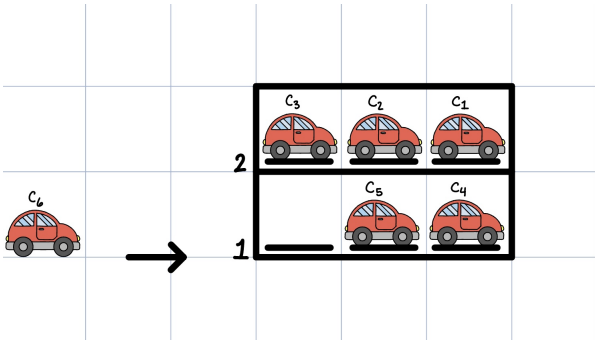

Figure 5.1.14: c_5 parks on level 1.

6. Observe that car c_6 prefers parking garage level 2 as given by $(2, 2, 2, 1, 1, \underline{2})$. Parking garage level 2 is at full capacity so c_6 must continue upward to find the next level not at full capacity to park on, but falls off the top level before doing so as illustrated by Figure 5.1.9.

5.2 Python Code Leading to Discovery

In this section, a python code with three functions including a main will be given. The main created a sequence that when input into the the On-line Encyclopedia of Integer Sequences, had a match with a sequence in a paper from 1977, [5]. Such paper gives us a recursive formula for a scenario that can be interpreted as parking garage functions.

The following is the python code:

```
import itertools

#Input: pf ((n,cn)-parking garage level preference sequence), c (capacity of
                                        each level) and n (number of levels).
```

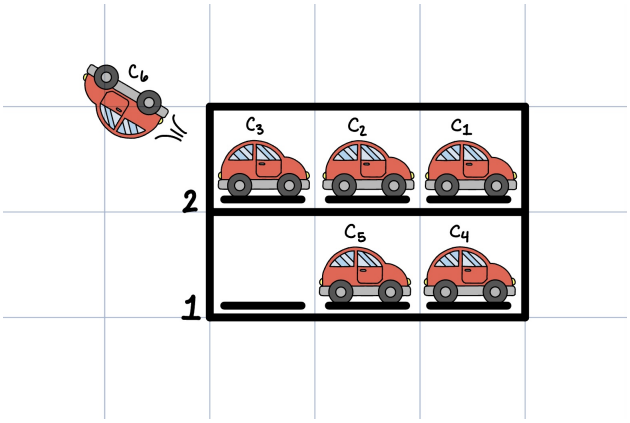

Figure 5.1.15: c_6 doesn't park.

```
#Output: Number of parked cars yielded by the (n,cn)-parking garage level
                                        preference sequence.
def num_of_Parked_Cars(pf, c, n):
    # take list pf and sort in ascending order
    new_pf = sorted(pf)
    # create list of possible preferences
    levels = [j for j in range(1, n + 1)]
    # start counter for parked cars
    parked_Cars = 0
    j = 0
    # compare every nth group of c to n
    for i in range(0, c * n, c):
        x = 0
        while x < c:
            if new_pf[i + x] < levels[j] or new_pf[i + x] == levels[j]:
                parked_Cars = parked_Cars + 1
            else:
                break;
            x = x + 1
        j = j + 1
        # return number of parked cars
    return parked_Cars

#Input: k_f (capacity of each level) and n_f (number of levels).
#Output: Cardinality of the set of (n_f,k_f \cdot n_f)-parking garage functions.
def num_of_Parking_Garage_Functions(k_f,n_f):
    # number of levels
    n = int(n_f)
    print('Levels =', n)
    # capacity of each level
    k = int(k_f)
    print('Capacity =', k)
    # number of cars is kn
    print('Number of cars =', k * n)
    # create list of possible preferences
    levels = [j for j in range(1, n + 1)]
    # create list of lists of possible preferences for each car
    somelists = []
    for v in range(1, k * n + 1):
        somelists.append(levels)
```

```
    # counter for number of parking garage functions out of all preference
                                                vectors
    valid_pgf = 0
    # evaluate each element of cartesian product (every possible combination of
                                                values) from somelists for being a
                                                parking garage function or not
    i = 0
    for v in itertools.product(*somelists):
        #if i % 100000 == 0:
            #print(i)
        v_lis = list(v)
        parked_Cars = num_of_Parked_Cars(v_lis, k, n)
        if k * n == parked_Cars:
            valid_pgf = valid_pgf + 1
        i += 1
    # Return number of valid parking garage functions
    return valid_pgf


#Test for the number of:
#(1,4 \cdot 1)-parking garage functions,
#(2,4 \cdot 2)-parking garage functions,
#(3,4 \cdot 3)-parking garage functions,
#(4,4 \cdot 4)-parking garage functions
#and (5,4 \cdot 5)-parking garage functions.
for i in range(1,5):
    valid_pgf = num_of_Parking_Garage_Functions(4,i)
    print(valid_pgf)
```

The output of the main function is:

1

11

378

27213

3378680

This would mean that

$$|PGF_{(1,4\cdot 1)}| = 1, \text{ and}$$

$$|PGF_{(2,4\cdot 2)}| = 11, \text{ and}$$

$$|PGF_{(3,4\cdot 3)}| = 378, \text{ and}$$

$$|PGF_{(4,4\cdot 4)}| = 27213, \text{ and}$$

$$|PGF_{(5,4\cdot 5)}| = 3378680.$$

We then get a match with a sequence in a paper from 1977, [5], when we input $1, 11, 378, 27213, 3378680$ into the On-line Encyclopedia of Integer Sequences as illustrated in 5.2.1.

In [5], there is a brief discussion on transforming their problem to a parking problem where the balls in their paper are interpreted as cars and the cells with a standard capacity are interpreted as a parking lot with each row having a standard capacity. For this paper, we reimagine the cells with a standard capacity as a parking garage with each level having a standard capacity.

This leads us to modifying a lemma in [5] to the following theorem. The following theorem gives a recursive formula for the number of (n, kn)-parking garage functions.

Theorem 5.2.1. *Let $n, k \in \mathbb{N}$. Then,*

$$|PGF_{k,n}| = \sum_{i=1}^{n} i\binom{kn-1}{ki-1}|PGF_{k,ki-1}| \cdot |PGF_{k,k(n-i)}|.$$

5.3 Descending Parking Garage Functions and Dyck Words Bijection

In this section, we introduce a slight variation of parking garage functions called *descending parking garage functions*. Next, we provide an argument for a bijection between *descending parking garage functions* and *Dyck words*.

We now define descending parking garage functions.

Figure 5.2.1: Match on OEIS.

Definition 5.3.1. Let $n, k \in \mathbb{N}$. Let $A = (a_1, a_2, a_3, \ldots, a_k n)$ be an (n, kn)-parking garage function. If $a_1 \geq a_2 \geq a_3 \geq \ldots \geq a_k n$, then A is an (n, kn)**-descending parking garage function**. Let $DPGF_{(n,kn)}$ be the set of (n, kn)-parking garage functions. △

The following theorem was first conjectured since their is a bijection between descending parking functions and Dyck words.

Theorem 5.3.2. *Let $n, k \in \mathbb{N}$. There is a bijection between $DW_{(n,n)}$ and $DPGF_{(n,kn)}$.*

We will demonstrate the proof of this theorem by defining two functions and using an example to support them being inverses of one another. The example can then be generalized, but we will not include the details here.

We will use the following function as part our first function.

Definition 5.3.3. Let $n, k \in \mathbb{N}$. Let P be an (n, kn)-descending parking garage function. Let $i \in [n]$. Let $N_g(P, i)$ denote the number of elements in P equal to i. △

We now define our first function where the input is an (n, kn)-descending parking garage functions and the output is an (n, kn)-Dyck word.

Definition 5.3.4. Let $n, k \in \mathbb{N}$. Let $F_g : DPGF_{(n,kn)} \longrightarrow DW_{(n,kn)}$ be a function defined by $F_g(P = (p_0, p_1, \ldots, p_{kn-1})) = Xb_nXb_{n-1}Xb_{n-2}X \ldots Xb_1)$ where b_i is an $N_g(P, i)$-length word over the alphabet $\{Y\}$. △

We now define a second function where the input is an (n, kn)-Dyck word and the output is an (n, kn)-descending parking garage function.

Recall from Definition 2.2.1 that if L is a word over the alphabet $\{X, Y\}$, then $X(L)$ denotes the number of X's in L.

Definition 5.3.5. Let $n, k \in \mathbb{N}$. Let $G_g : DW_{(n,kn)} \longrightarrow DPGF_{(n,kn)}$ be a function defined by $G_g(a_0a_1 \ldots a_{kn+n-1}) = (p_0, p_1, \ldots, p_{kn+n-1})$ where

$$p_i = \begin{cases} \text{empty} & \text{if } a_i = X \\ n + 1 - X(a_i) & \text{if } a_i = Y. \end{cases}$$

△

We now demonstrate how these two functions are inverses through two examples.

Let $n, k \in \mathbb{N}$. Let $P \in DPGF_{(n,kn)}$. The first is an example of how $G_g(F_g(P)) = P$.

Example 5.3.6. Consider the following $(2, 3 \cdot 2)$-descending parking garage function: $P = (2, 2, 1, 1, 1, 1)$. Observe that $F_g((2, 2, 1, 1, 1, 1)) = XYYXYYYY$. Furthermore, $G_g(XYYXYYYY) = (2, 2, 1, 1, 1, 1)$.

Hence, $G_g(F_g(P)) = P$. ◊

Let $n, k \in \mathbb{N}$. Let $A \in DW_{(n,kn)}$. The following is an example of how $F_g(G_g(A)) = A$.

Example 5.3.7. Consider the following $(2, 3 \cdot 2)$-Dyck word: $A = XYYYXYYY$. Observe that $G_g(XYYYXYYY) = (2, 2, 2, 1, 1, 1)$. Furthermore, $F_g((2, 2, 2, 1, 1, 1)) = XYYYXYYY$.

Hence, $F_g(G_g(A)) = A$. ◊

By definition of bijection, the next corollary follows.

Corollary 5.3.8. *Let $n, k \in \mathbb{N}$. Then $|DPGF_{(n,kn)}| = |DW_{(n,kn)}|$.*

Proof. Let $n, k \in \mathbb{N}$. By Theorem 5.3.2, there is a bijection between $DPGF_{(n,kn)}$ and $DW_{(n,kn)}$. By definition of bijection, $|DPGF_{(n,kn)}| = |DW_{(n,kn)}|$. □

The next corollary follows.

Corollary 5.3.9. *Let $n, k \in \mathbb{N}$. Then $|DPGF_{(n,kn)}| = \frac{1}{kn+1}\binom{(k+1)n}{n}$.*

Proof. By Corollary 5.3.8, $|DPGF_{(n,kn)}| = |DW_{(n,kn)}|$. By Corollary 3.3.3, $|DW_{(n,kn)}| = \frac{1}{kn+1}\binom{(k+1)n}{n}$. Thus, $|DPGF_{(n,kn)}| = |DW_{(n,kn)}| = \frac{1}{kn+1}\binom{(k+1)n}{n}$. □

6
Future Work

Suggestions for future work include attempting to take results known for parking functions and adapt them to parking garage functions, which is just a new concept that is a generalization of parking functions. In particular, we would like to find a closed formula for $|PGF_{(n,kn)}|$. More projects ideas include:

1. Can we find the closed formula of defective parking garage functions?

2. There exists research on k-naples parking functions which allow for cars to be able to move backwards by k parking spaces. What can be said about k-naples parking garage functions?

3. k-naples parking functions also have a bijection with lattice paths. Is there a bijection between k-naples parking garage functions and a certain type of lattice path?

4. There is also a bijection between parking functions and binary trees as well as triangulations of convex polygons. Is their a bijection between parking garage functions and certain graphs or triangulations of polygons?

Bibliography

[1] Alex Christensen and Pamela E. Harris and Zakiya Jones and Marissa Loving and Andrés Ramos Rodríguez and Joseph Rennie and Gordon Rojas Kirby, *A generalization of parking functions allowing backward movement*, The Electronic Journal of Combinatorics **27** (2020).

[2] Ayomikun Adeniran and Steve Butler and Galen Dorpalen-Barry and Pamela E. Harris and Cyrus Hettle and Qingzhong Liang and Jeremy L. Martin and Hayan Nam, *Enumerating Parking Completions Using Join and Split*, The Electronic Journal of Combinatorics **27** (2020).

[3] Drew Armstrong and Nicholas A. Loehr and Gregory S. Warrington, *RATIONAL PARKING FUNCTIONS AND CATALAN NUMBERS*, Annals of Combinatorics **20** (2016), 21–58.

[4] Francine Blanchet-Sadri and Kun Chen and Kenneth Hawes, *Dyck Words, Lattice Paths, and Abelian Borders*, International Journal of Foundations of Computer Science **33** (2022), 203-226.

[5] Ian F. Blake and Alan G. Konheim, *Big Buckets Are (Are Not) Better!*, Journal of the Association for Computing Machinery **24** (1977), 591-606.

[6] Joshua Carlson and Alex Christensen and Zakiya Jones and and Andrés Ramos Rodríguez, *Parking Functions: Choose Your Own Adventure*, The College Mathematics Journal **52** (2021).

[7] Kimberly P. Hadaway and Pamela E. Harris, *Honk! Honk!, Part 1: An Introduction to Parking Functions*, Girls' Angle Bulletin **14** (December 2020/January 2021), no. 2, 15-20.

[8] ———, *Honk! Honk!, Part 2: An Introduction to Parking Functions*, Girls' Angle Bulletin **14** (February/March 2021), no. 3, 14-20.

[9] Peter J. Cameron and Daniel Johannsen and Thomas Prellberg and Pascal Schweitzer, *Counting Defective Parking Functions*, The Electronic Journal of Combinatorics **15** (2008).

[10] Yukiko Fukukawa, *COUNTING GENERALIZED DYCK PATHS*, arXiv: Combinatorics (2013).