PREFERENCE-RESTRICTED PARKING FUNCTIONS

ALAN KAPPLER AND JAYDEN THADANI

ABSTRACT. We define and enumerate different types of preference-restricted parking functions — parking functions of length c such that all cars prefer spots in some $S \subset [c]$. These include new, combinatorial proofs of difficult results from [CJPS08] and [BK77]. We describe the connections of initial segment restrictions to Abel's binomial theorem, hyperplane arrangements and polytopes. We also demonstrate procedures to sample from them, and then use them to sample from defect-d preference lists. Finally, the appendices contain detailed exposition on prime parking functions and Abel's binomial theorem as they relate to this work.

CONTENTS

- Classical results
- Combinatorial proof

preference-restricted parking functions as a paradigm, something between parking functions and **u**-parking functions. Maybe cut out results that are too specific to a particular restriction — use the narrative as a guide for what to cut

how to unify DIE proofs? Leave a comment that these are similar?

add your author details in alphabetical order of last name — LaTeX doesn't do it automatically

modular asymptotics

TODO LIST

Work on this after, but lead with the cool things you have done.

1. Introduction

Imagine c cars driving down a one-way street with c parking spots. Then, for $i \in [c]$, each ith car has a preferred parking spot π_i. In order, the cars try to park. Each car goes to its preferred parking spot. If the spot is unoccupied, it parks there. Else, if there are unoccupied spots further along, it parks in the next one. If not, it doesn't park at all. We say $\pi = (\pi_1, \dots, \pi_c)$ is a *parking function* if such a list of preferences leads to all cars parking. Naturally this definition gives rise to questions like

$PF_{c|s}$ where $\begin{cases} S = [s] \\ S = [s+1] \setminus \{2\} \\ S = \{j \in [qs-1] \mid j \equiv 1 \pmod q\} \end{cases}$

#{f: [c] → S | f is a parking function}.

- How many parking functions are there?
- Which preference lists are parking functions?
- Can these parking functions be neatly partitioned? By which cars go where (the *parking outcome*)? Or by invariance under some symmetry?

the answers to which are well known and detailed in the preliminaries section.

It's also only natural to ask the same questions about slightly different parking procedures. For example, what if there are only $s < c$ spots? At least $c - s$ cars won't get to park, but how many preference lists minimise this *defect*? Such *minimum-defect preference lists* are exactly the parking functions of length c such that all cars have preferences in the first s spots. What if g cars can park at each spot? These are parking functions of length c where all cars prefer to park in spots $s \equiv 1 \pmod{g}$.

Parking functions with restrictions on the cars' preferences are not just interesting for the stories they can tell; they can also be used to construct and sample from other preference lists of interest (like defect d preference lists). We call these central objects *preference-restricted parking functions*. We denote the set of all S-*restricted parking functions*, parking functions of length c with preferences restricted to $S \subset [c]$, by $\mathrm{PF}_{c|S}$.

Is this standard notation? If so, "following \cite{...}"

We will show that our motivating case $S = [s]$ can be enumerated in two ways. They are connected by Abel's binomial theorem and provide combinatorial insight for the results in [CJPS08] —

(I'd like to see notation like PF: [c] → S somewhere.)

Theorem 1.1. [cite]

Let $c, s \in \mathbb{N}$ with $c \leq S$...

$$\#\mathrm{PF}_{c|[s]} = s^c - \sum_{i=0}^{s-1} \binom{c}{i} (i+1)^{i-1} (s-i-1)^{c-i}$$
$$= \sum_{i=s}^{c} \binom{c}{i} (i+1)^{i-1} (s-i-1)^{c-i}.$$

Always start theorems with words

We show that there are indeed equivalence classes for these preference-restricted parking functions based on both preferences and parking outcome, echoing the classical results for parking functions. We also apply similar combinatorial arguments to enumerate prime parking functions.

Finally, we enumerate the "modular restriction" case with a much simpler proof than that initially given by Blake and Konheim [BK77] —

Theorem 1.2. *Let S be the set of the first s natural numbers j with $j \equiv 1 \pmod{g}$. Then the number of parking functions of length $gs - 1$ with gap g between possible preferred spots is $\#\mathrm{PF}_{gs-1|S} = s^{gs-2}$.*

an example would be helpful

In section 2 we recount answers to all of the "natural" enumerative questions above about parking functions and introduce prime parking functions and their breakpoints which are used later in the paper.

We answer all of the same questions and more for $[s]$-restricted parking functions in section 3. We prove theorem 1.1, enumerate the prime parking function case, and provide a combinatorial explanation for their connection to Abel's identity. We also describe interesting connections to hyperplane arrangements and polytopes. Specifically, we show that $\mathrm{PF}_{c|[3]}$ is the number of chambers in the C_c-Shi hyperplane arrangement, and $\mathrm{PF}_{c|[2]}$ is the number of chambers in a Linial-like hyperplane arrangement.

In section 4, we prove theorem 1.2 and a generalisation combinatorially. In section 5 we talk about the general case with random restrictions to $S \subset [c]$ — expected numbers of parking functions and their S_c-orbits. We provide asymptotic results for the latter.

Finally, section 6 details methods to sample from preference restricted parking functions and how to apply them to sample defect d preference lists. We show how $[s]$-restricted parking functions can be combined with difference types of parking functions to construct defect d parking functions, and thus, by sampling from $\mathrm{PF}_{c|[s]}$ with random walks, we can sample from defect d parking functions.

2. Preliminaries

2.1. **How many parking functions are there?** Pollak suggested an illuminating circle argument to enumerate parking functions, which we will recount since it inspires the proof of theorem 1.2.

Proposition 2.1 (Pollak's circle argument, from [Rio69]). *There are* $(c+1)^{c-1}$ *parking functions of length* c.

Proof. Consider c cars attempting to park in a circular parking lot with $c+1$ spots, in which cars circle clockwise around the lot until they find a parking spot. There are $(c+1)^c$ different preference functions that come out of this, and every car will be able to park.

There will be one empty spot; removing this spot yields a line of c parked cars in a linear lot with c spots. No cars will pass the empty spot, since otherwise at least one would have parked in it. Therefore, no cars will ever exit the parking lot: in other words, we have a parking function

Conversely, every parking function will yield circular parking functions of the form described. Adding an empty parking spot onto the end and closing up the circle yields a circular parking lot with clockwise movement; the orientation of this circle depends only on where the empty spot is placed. This can be done in $c+1$ ways, one for each of the $c+1$ possible spots.

Thus the total number of parking functions of length c is $\frac{1}{c+1}$ times the number of corresponding circular preference lists, or $\frac{(c+1)^c}{c+1} = (c+1)^{c-1}$. □

2.2. **Which preference lists are parking functions?** Whether a preference list is a parking function is entirely determined by its non-decreasing arrangement. Essentially, a preference list is a parking function if and only if at least i cars prefer to park in the first i spots for each i. Equivalently,

Proposition 2.2 (the Catalan condition, see [Yan15] for proof)**.** *A preference list $\pi \in [c]^c$ is a parking function if and only if its non-decreasing rearrangement π' gives $\pi'_i \leq i$ for all $i \in [c]$.*

This condition yields a natural bijection between non-decreasing parking functions and Catalan objects. For example, [ALW16] describes the natural bijection with Dyck paths by stepping East at each change and North otherwise. It also gives us a natural action of the symmetric group on parking functions — permuting which cars prefer which spots. Formally, *the natural action* of S_c on PF_c is given by $\sigma \cdot \pi = \pi \circ \sigma^{-1}$, thinking of π as a function with $\pi(i) = \pi_i$.

2.3. **Partitioning parking functions.** One natural way to partition parking functions is by their S_c-orbits. Each orbit corresponds to a non-decreasing parking function. Then, the following result falls out of their bijection with Catalan objects —

Proposition 2.3 (see [ALW16] for proof)**.** *The number of non-decreasing parking functions of length c is C_c, the cth Catalan number.*

However, we can also partition parking functions by their outcome — which parking functions result in the same cars parking in the same spots? Pinsky gives a formula in [Pin24] by considering how many earlier spots a car could prefer to end up in the spot it does.

2.4. **The number of ones.** It is a result of [Yan15] that parking functions admit an elegant partition based on the number of cars which prefer the first parking spot:

Proposition 2.4 (the number-of-ones enumerator)**.** *The enumerator for parking functions based on the number of cars preferring the first spot is $x(x+c)^{c-1}$. That is,*

$$\sum_{\pi \in \mathrm{PF}_c} x^{\#\{i|\pi(i)=1\}} = x(x+c)^{c-1}.$$

generating function

I might have this be "just in time"

A quick example. When $c=3$,

$x(x+3)^2 = x^3+6x^2+9x$, so there is one parking function $(1,1,1)$, six $(1,1,2)$ or $(1,1,3)$

and...

Why do we care about these? And what do they have to do with the other stuff?

2.5. **Prime parking functions.** Prime parking functions were introduced by Gessel in an unpublished manuscript; they have subsequently been used in a variety of sources such as [AL99] and [MHJ+23]. We provide a brief review, but for proofs and references for further reading, see appendix A.

Definition 2.5 (prime parking functions, PPF_c)**.** A prime parking function of length c is a parking function such that for every integer $i \in [c-1]$, at least $i+1$ cars have preferences in the first i spots. $i < c$

The set of all prime parking functions of length c is denoted PPF_c

Example?

This definition comes about in analogy to the Catalan condition on parking functions, where at least i cars had to have preferences in the first i spots; here our cars must prefer even earlier spots than before. Note that this means prime parking functions are permutation-invariant as well.

Naturally, an *S-restricted prime parking function* is a prime parking function that is also an S-restricted. We denote the set of all S-restricted prime parking functions of length c by $\mathrm{PPF}_{c|S}$.

Definition 2.6. A breakpoint in a parking function of length c is a parking spot $b \in [c]$ which no car has ever "passed"; equivalently, it is a spot in which at most one car has ever attempted to park.

Example?

In terms of the Catalan condition, the breakpoints are exactly the spots i where the number of cars attempting to park in the first i spots is exactly i. Thus, parking functions are prime if their only breakpoint is the last spot. The name prime is then motivated by the fact that we can decompose parking functions into primes between breakpoints —

Proposition 2.7. *A parking function is prime if and only if it has exactly one breakpoint. Furthermore, there exists a natural decomposition of any parking function π into b prime parking segments, where b is the number of breakpoints in π.*

We can also count prime parking functions using breakpoints and a circle argument, given in the appendix.

My preferred reason? $\#(\mathrm{PF}:[n]\to[n])$ has a nice structure, what about $\#(\mathrm{PF}:[m]\to[n])$?

Proposition 2.8. *The number of prime parking functions of length c is $(c-1)^{c-1}$.*

3. Initial segment restrictions — $[s]$-restricted parking functions

This is part of why someone cares.

There are two reasons to consider $[s]$-restricted parking functions of length c. We might want to model the case that certain parking spots, particularly those to the back of a linear parking lot may be undesirable, and thus, only the first s spots are preferred. However, $[s]$-restricted parking functions have another natural interpretation in terms of minimum-defect preference lists. Specifically, if c cars attempt to park in $s < c$ spots, the preference lists that

"all the spots are full"?

lead to the minimum number of cars being unable to park are exactly the $[s]$-restricted parking functions of length c. We can see this as follows — if we ignore the last $c - s$ spots, an $[s]$-restricted parking function is just a minimum-defect preference list. By adding back $c - s$ spots at the end of the parking lot, we can see that any minimum-defect preference list forms a parking functions.

3.1. **How many are there?** We have two different enumerations of $[s]$-restricted parking functions. The first is a special case of the formula for car parking numbers in [CJPS08], and the second is novel, proven using the DIE method pioneered in [BQ08].

Theorem 1.1.

$$\#\mathrm{PF}_{c|[s]} = s^c - \sum_{i=0}^{s-1} \binom{c}{i} (i+1)^{i-1}(s-i-1)^{c-i} \tag{\#}$$

$$= \sum_{i=s}^{c} \binom{c}{i} (i+1)^{i-1}(s-i-1)^{c-i}. \tag{\#}$$

Proof. ~~The first~~ count comes from eliminating all non-parking function preference lists from the s^c possible preference lists in $[s]^c$. If $\pi \in [s]^c$ is not a parking function, then there must be an unoccupied spot somewhere among the c spots. We claim further, that this unoccupied spot must be one of the first s spots. This follows exactly from the argument above. Since all of the cars prefer one of the first s spots, and at least $c - s$ cars will be unable to park in those first s spots, at least $c - s$ cars come to the last $c - s$ spots willing to park in the first available one.

Let i be the number of spots before the first unoccupied spot. Then there must be i cars that form a parking function on those first i spots. Choose those i cars in one of $\binom{c}{i}$ ways, and one of the $(i+1)^{i-1}$ parking functions of length i. None of the remaining $c - i$ cars can prefer any of the first $i+1$ spots since they would end up parking in the unoccupied spot $i+1$. However, they each can prefer any of the remaining $s - i - 1$ spots in one of $(s-i-1)^{c-i}$ ways. This makes for $\binom{c}{i}(i+1)^{i-1}(s-i-1)^{c-i}$ preference lists in $[s]^c$ so that the first unoccupied spot is $i+1$. Subtracting the sum over all possible i from the number of total preference lists gives us the first enumeration —

$$\#\mathrm{PF}_{c|[s]} = s^c - \sum_{i=0}^{s-1} \binom{c}{i} (i+1)^{i-1}(s-i-1)^{c-i}.$$

Another way to count $\mathrm{PF}_{c|[s]}$ is to start with all parking functions of length c and eliminate those with cars that prefer spots in $[c] \setminus [s]$. We do so by

means of an involution and alternating sum that results in us counting every parking function both positively and negatively unless it has no cars that prefer spots beyond the first s.

Suppose we choose a subset of size $i \geq s$ of our cars to be coloured indigo. These cars are to form a parking function on the first i parking spots (though the parking outcome may not result in them actually parking there). The remaining $c - i$ cars are coloured red and can prefer any of the $i + 1 - s$ "forbidden" spots in $[i + 1] \setminus [s]$. These preferences always form a parking function — the Catalan condition is satisfied up to spot i by the indigo cars since they form a parking function, and then the red cars all prefer one of the first $i + 1$ spots, satisfying the condition from $i + 1$ onwards. We count these parking functions positively when there are an even number of red cars and negatively otherwise. This count is

$$\sum_{i=s}^{c} \binom{c}{i} (i+1)^{i-1} (i+1-s)^{c-i} (-1)^{c-i}.$$

That is, it's exactly the second enumeration above, just with the $(-1)^{c-i}$ factored out.

We now create a sign-reversing involution on these subsets — we can change the paint on a certain car from red to indigo or back from indigo to red without changing the parking function so that we see that the parking function is counted both positively and negatively. If m is the maximum preference among all the cars, let car x prefer spot m. That is, for a parking function π, let $\pi_x = m \geq \pi_j$ for all $j \in [c]$. Note that $m > s$ as long as there are some red cars, since the red cars all prefer spots greater than s.

If car x is painted red, then $m \leq i+1$, just by definition. We can reconstruct the same parking function with car x painted indigo. We can add any preference in $[i + 1]$ to a parking function of length i and the list will form a parking function of length $i + 1$. To see this, just add the preference at the end. The first i cars will park in the first i spots on which they form the parking function, and then the added car will park in the last spot. Since parking functions are permutation invariant, the list will form a parking function no matter where we add the car. Thus, with car x, the indigo cars form a parking function of length $i + 1$. Since the red cars with car x formed a subset of $[i + 1] \setminus [s]$, without car x they certainly form a subset of the larger $[i + 2] \setminus [s]$.

If car x is painted indigo, then $m \leq i$ in order for the indigo cars to form a parking function. Thus, we can reconstruct the same parking function with car x repainted red. The remaining indigo cars will still form a parking function on $i - 1$ cars — the jth smallest preference will still be at most j after removing the maximum preference. Since $m \leq i$ was the maximum

Start by saying "We give a combinatorial interpretation of the summands"

nice!

Get an example!

be the first car that prefers?

This is a long proof. I'm wondering if you can break it up with some lemmas. This is my favorite part of the paper, and I think it deserves its own subsection-

preference, none of the red cars preferences will exceed i, and car x will have preference exceeding s since it preferred spot m. That is, unless $m \leq s$.

This is our exception — we cannot recolour to have one more or one fewer red car if none of the cars prefer a spot greater than s. These are the $[s]$-restricted parking functions of length c. They are only counted once — positively, as a subset of the first term in which there are no cars coloured red. Any other parking function is counted positively once when x is coloured so that there are an even number of red cars, and negatively once when x is coloured so that there are an odd number of red cars. Thus, the sum counts $\mathrm{PF}_{c|[s]}$. □

Say earlier. Or better yet, just make a choice.

Remark 3.1. Though this enumeration may appear to depend on the choice of x, it does not. No matter what x is chosen such that car x prefers m, the proof shows that car x can be recoloured, and thus, that the parking function is positively and negatively, unless $m \leq s$, in which case the parking function is counted only once, again, regardless of the choice of x.

Remark 3.2. One could enumerate the $[s]$-restricted parking functions by inclusion-exclusion as

$$\#\mathrm{PF}_c - \#\Big|\bigcup_{j\in[c]} A_j\Big|$$

for $A_j = \{\pi \in \mathrm{PF}_c \mid \pi_j \in [c] \setminus [s], (\pi_1, \ldots, \pi_{j-1}, \pi_{j+1}, \ldots, \pi_c) \in \mathrm{PF}_{c-1}\}$. This count appears to be similar to the second enumeration above, but it is difficult to pin down the relation between them exactly.

At the end of the paper, you might ask this question.

3.2. **Partitioning restricted parking functions.** We can extend the known enumeration of non-decreasing parking functions (or S_c-orbits) to the $[s]$-restricted case. Proving this just involves noticing a simple recurrence relation.

When $c \geq s$ ...

Proposition 3.3. *The number of non-decreasing $[s]$-restricted parking functions of length c is $C(c, s-1)$, the $(c, s-1)$th entry in Catalan's triangle.*

have you defined this notation?

Proof. Let $\mathrm{PF}^{\uparrow}_{c|[s]}$ be the set of non-decreasing $[s]$-restricted parking functions of length c. Note that we already have $\#\mathrm{PF}^{\uparrow}_{c|[s]} = C(c, s-1)$ ~~on the "sides" of Catalan's triangle (where~~ when ~~$c = s$~~ or $s = 0$). That is, $\#\mathrm{PF}^{\uparrow}_{c|[c]} = C_c = C(c, c-1)$ and $\#\mathrm{PF}^{\uparrow}_{c|[1]} = 1 = C(c, 0)$.

by Theorem [ref] ...

$C(c, -1)$?

Suppose $\pi \in \mathrm{PF}^{\uparrow}_{c|[s]}$. If $\pi_c = s$, then $(\pi_1, \ldots, \pi_{c-1}) \in \mathrm{PF}^{\uparrow}_{c-1|[s]}$. That is, π is a non-decreasing $[s]$-restricted parking function of length $c-1$ with an s tacked on the end. If $\pi_c \neq s$, then there cannot be any s in π, and thus, π is

$[s-1]$-restricted as well. Thus,

$$\#\mathrm{PF}^{\uparrow}_{c|[s]} = \#\mathrm{PF}^{\uparrow}_{c-1|[s]} + \#\mathrm{PF}^{\uparrow}_{c|[s-1]}.$$

However, this identity holds for Catalan's triangle as well [Bai96]. That is, $C(c,s) = C(c-1,s) + C(c,s-1)$. Thus, we can recursively extend the result from the initial conditions where $c = s$ or $s = 0$, to all non-negative integers c and s. □

We can also ask about the parking outcomes of $[s]$-restricted parking functions. Which parking outcomes are even possible from restricted parking functions?

Proposition 3.4. *The possible parking outcomes of* $\mathrm{PF}_{c|[s]}$ *are exactly the* $\binom{c}{s-1}(s-1)!$ *functions* $\sigma : [c] \to [n]$ *where the last* $c-s+1$ *spots are filled by cars in ascending order. That is,* $\sigma^{-1}(s) \dots \sigma^{-1}(s)$ *is an increasing sequence.*

Proof. Note that any parking outcome σ of $\mathrm{PF}_{c|[s]}$ must have $\sigma^{-1}(s), \dots, \sigma^{-1}(c)$ increasing. All cars attempt to park in one of the first $[s]$ spots. The first car to prefer spot s or attempt to park there gets to park there; the unparked cars fill the first empty spot they come across, and thus, fill them in order.

We can achieve any parking outcome σ with $\sigma^{-1}(s), \dots, \sigma^{-1}(c)$ increasing as follows. Let cars $\sigma^{-1}(s), \dots, \sigma^{-1}(c)$ prefer spot s. Thus, they all end up in the spots specified by σ. Allow the remaining cars (that are not the $\sigma^{-1}(j)$th car for $j \in [c] \setminus [s]$) to prefer their parking outcome — let car j prefer $\sigma(j)$. Since there are no collisions, they all end up in the spots specified by σ as well. □

Remark 3.5. These parking outcomes can be thought of as orbits of $\mathrm{Sym}_{[c]\setminus[s-1]}$, the symmetric group of $[c] \setminus [s-1]$, acting on S_c by

$$\tau \cdot \sigma(i) = \begin{cases} \tau \circ \sigma(i) & \sigma(i) \in [c] \setminus [s-1] \\ \sigma(i) & \sigma(i) \in [s-1] \end{cases}$$

for $\tau \in \mathrm{Sym}_{[c]\setminus[s-1]}$ and $\sigma \in S_c$.

We can also modify Pinsky's formula to get exact counts for how many $[s]$-restricted parking functions result in each of these parking outcomes.

Proposition 3.6 (inspired by [Pin24])**.** *The number of* $[s]$*-restricted parking functions of length* c *with parking outcome* σ *is*

$$\prod_{i=1}^{c} \max(0, l_{c,i}(\sigma) - \max(0, i-s)).$$

Here, suppose σ^{-1} is written in word form as $\sigma^{-1}(1)\sigma^{-1}(2)\dots\sigma^{-1}(c)$ indicating the cars that parked in spots $1, 2, \dots, c$. Then $l_{c,i}(\sigma)$ is the length of the longest string $\sigma_k^{-1}, \dots, \sigma_i^{-1}$ such that $\sigma_j^{-1} < \sigma_i^{-1}$ for any $j \in \{k, \dots, i-1\}$.

Proof. We will first ignore the restriction on the cars preferences and recount Pinsky's result.

We know that car σ_i^{-1} ends up in spot i. However, car $\sigma^{-1}(i)$ could have preferred a spot before i. In particular, it could have preferred exactly that string of spots before i that were occupied and didn't have any unoccupied spots between it and spot i while car $\sigma^{-1}(i)$ was trying to park. This is what $l_{c,i}(\sigma)$ counts.

If there were no restriction, then the answer would just be the product of all $l_{c,i}$. However, if $i > s$, then we need to subtract off the $\max(0, i-s)$ invalid preferences $s+1, \dots, i$. If $l_{c,i}(\sigma) - \max(0, i-s)$ is negative for any i, then $\sigma^{-1}(s), \dots, \sigma^{-1}(c)$ cannot be increasing. Thus, our answer is just the product of all $\max(0, l_{c,i} - \max(0, i-s))$. □

3.3. **The number of ones.** Finally, we have an enumerator for the number of ones in $[s]$-restricted parking functions:

Theorem 3.7. *The enumerator for $[s]$-restricted parking functions of length c based on the number of cars preferring the first spot is*

$$\sum_{\pi \in \mathrm{PF}_{c|[s]}} x^{\#\{i|\pi(i)=1\}} = (s-1+x)^c - \sum_{i=0}^{s-1} \binom{c}{i} x(i+x)^{i-1}(s-1-i)^{c-i}$$

$$= \sum_{i=s}^{c} \binom{c}{i} x(i+x)^{i-1}(s-1-i)^{c-i}.$$

Proof. The proof proceeds very similarly to that of theorem 1.1; we note here only the places where it is different. Instead of considering all preference lists, we consider those with i 1s ~~for arbitrary i~~; these will be represented by the coefficient of x^i in our formula. Summing these gives the enumerator for the number of 1s in any set.

The enumerator for some number of independently-made choices on different subsets of our cars is given by the product of the enumerators for each individual choice. Since the enumerator for one car's preference is $s-1+x$, the enumerator for all $[s]$-restricted preference functions of length c is $(s-1+x)^c$, as all choices are independent.

As shown in proposition 2.4, the enumerator for parking functions over the first i spots is $x(i+x)^{i-1}$; all other calculations in the first equality never include a choice of how many 1s to have, and so we can simply multiply by the same constants used in theorem 1.1.

For the second equality, the definition-involution-exception proof works similarly to before, with a fine-grained focus on the number of 1s added in. Each choice of placement for the i indigo cars is given by $x(i+x)^{i-1}$ since these cars form a parking function on the first i spots; red cars all prefer forbidden spots, so they will never choose 1. The remainder of the proof follows in the same manner. □

A section on prime parking functions!

3.4. **Prime parking functions.** Restricted parking functions have a close relationship with restricted prime parking functions. For example, the number of S-restricted prime parking functions is the number of T-restricted parking functions for a related subset T.

Proposition 3.8. *The number of S-restricted prime parking functions of length c is*

$$\#\mathrm{PPF}_{c|S} = \#\mathrm{PF}_{c|T}$$

where $T = \{1\} \cup \{i+1 \mid i \in S, 1 < i < c\}$. In particular, the number of prime parking functions of length c is equal to

$$\#\mathrm{PPF}_c = \#\mathrm{PF}_{c|[c]\setminus\{2\}},$$

while the number of prime parking functions of length c with preferences restricted to $[s]$ for $s < c$ is equal to if $s = c-1$, this matches above

$$\#\mathrm{PPF}_{c|[s]} \quad \#\mathrm{PF}_{c|[s+1]\setminus\{2\}}.$$

Corollary.

Proof. Note first that no cars will prefer spot c in a prime parking function of length c; by definition, all c cars will have preferences in $[c-1]$. Thus we only consider preferences less than c.

There is a natural bijection between $S\setminus\{c\}$ and T, given by the function $f : S\setminus\{c\} \to T$ which sends 1 to 1 and all other elements $x \in S\setminus\{1, c\}$ to $x+1$. This bijection induces a bijection between $S\setminus\{c\}$-restricted preference lists and T-restricted preference lists.

We claim that an $S\setminus\{c\}$-restricted preference list is a prime parking function if and only if the corresponding T-restricted preference list is a parking function. This is true since for $i > 2$, the first preference list will have at least i preferences less than i iff the second preference list has at least i preferences of at most i. Additionally, the first list has at least 2 preferences less than 2 iff the second list has at least 2 preferences of at most 2, since $2 \notin T$. □

We can also directly enumerate $[s]$-restricted prime parking functions. Our enumeration of $\mathrm{PPF}_{c|[s]}$ follows Theorem 1.1 very closely —

Theorem 3.9. Start with words

$$\#\mathrm{PPF}_{c|[s]} = s^c - (s-1)^c - \sum_{i=1}^{s} \binom{c}{i}(i-1)^{i-1}(s-i)^{c-i}$$

$$= \sum_{i=s+1}^{c} \binom{c}{i}(i-1)^{i-1}(s-i)^{c-i}$$

Keep splitting these up.

Proof. Again, the first count is the number of preference lists in $[s]^c$ less those which are not parking functions. In the previous proof we were effectively doing casework on the least i for which fewer than i cars prefer the first i spots. Analogously, if $\pi \in [s]^c$ is not a prime parking function, there must be some least i for which i or fewer cars prefer the first i spots. Notice that as long as $i > 1$, for i to be the least such spot, exactly i cars must prefer the first i spots — if fewer than i cars prefer the first i spots, then at most $i-1$ cars can prefer the first $i-1$ spots. Thus, if $i \neq 1$, i is a breakpoint. However, for $i = 1$, no cars may prefer the first spot at all.

After subtracting out the $(s-1)^c$ preference lists where no car prefers the first spot, we can assume that if $i \in [s]$ is the least spot such that i or fewer cars prefer the first i spots, then it is a breakpoint. Since more than j cars prefer the first j spots, for all $j < i$, the i cars that prefer the first i spots form a prime parking function on them in one of $\#\mathrm{PPF}_i = (i-1)^{i-1}$ ways. The remaining $c-i$ cars can prefer any of the remaining $s-i$ spots in $[s] \setminus [i]$ in one of $(s-i)^{c-i}$ ways. With $\binom{c}{i}$ ways to choose the i cars, there are $\binom{c}{i}(i-1)^{i-1}(s-i)^{c-i}$ preference lists in $[s]^c$ so that the first breakpoint is i. Subtracting the sum over all possible i, from the preference lists where at least one car prefers the first spot we get

$$\#\mathrm{PPF}_{c|[s]} = s^c - (s-1)^c - \sum_{i=1}^{s} \binom{c}{i}(i-1)^{i-1}(s-i)^{c-i}$$

We can also count $\mathrm{PPF}_{c|[s]}$ with the involution approach used above, starting with all prime parking functions of length c and eliminating those with cars that prefer spots in $[c] \setminus [s]$, counting each both positively and negatively until only our restricted prime parking functions remain.

Choose a subset of size $i \geq s$ of our cars to be coloured indigo. These cars are to form a prime parking function on the first i parking spots (though the parking outcome may not result in them actually parking there). The remaining $c-i$ cars are coloured red and can prefer any of the $i-s$ forbidden spots in $[i] \setminus [s]$. These preferences always form a prime parking function — the extra-strong Catalan condition is satisfied up to spot $i-1$ by the indigo cars since they form a prime parking function, and then the red cars all

prefer one of the first i spots, satisfying the condition from i onwards. We count these parking functions positively with an even number of red cars and negatively otherwise. This count is

$$\sum_{i=s}^{c} \binom{c}{i} (i-1)^{i-1} (i-s)^{c-i} (-1)^{c-i},$$

the second enumeration with the sign factored out.

We use the same involution as before: repaint car x, where m is the maximum preference among all the cars, and x prefers spot m; that is, for a parking function π, let $\pi_x = m \geq \pi_j$ for all $j \in [c]$. Note that $m > s$ as long as there are some red cars, since the red cars all prefer spots greater than s.

If car x is painted red, then $m \leq i$, just by definition. We can reconstruct the same parking function with car x painted indigo. We can add any preference in $[i]$ to a prime parking function of length i and the list will form a prime parking function of length $i+1$. To see this, just add the preference at the end. The first i cars will park in the first i spots on which they form the prime parking function, meaning none of the first $i-1$ spots are breakpoints; then the added car will pass spot i and park in spot $i+1$, meaning spot i is not a breakpoint either. Prime parking functions are also permutation invariant, so the list will form a prime parking function no matter where we add the car. Thus, with car x, the indigo cars form a prime parking function of length $i+1$. Since the red cars with car x formed a subset of $[i] \setminus [s]$, without car x they also form a subset of the larger $[i+1] \setminus [s]$.

If car x is painted indigo, then $m \leq i$ in order for the indigo cars to form a parking function. Thus, we can reconstruct the same parking function with car x repainted red. The remaining indigo cars will still form a prime parking function on $i-1$ cars — the jth smallest preference will still be less than j after removing the maximum preference. Since $m \leq i$ was the maximum preference, none of the red cars' preferences will exceed i, and car x will have preference exceeding s since it preferred spot m — unless $m \leq s$.

This is our exception — we cannot recolour to have one more or one fewer red car if none of the cars prefer a spot greater than s. These are the $[s]$-restricted prime parking functions of length c. They are only counted once — positively, as a subset of the first term in which there are no cars coloured red. Any other parking function is counted positively once when x is coloured so that there are an even number of red cars, and negatively once when x is coloured so that there are an odd number of red cars. Thus, the sum counts $\mathrm{PPF}_{c|[s]}$. □

3.5. **Abel's binomial theorem.** In enumerating $[s]$-restricted parking functions and their prime variants, we demonstrated two different counts for

each — one by excluding all of the restricted preference lists that aren't the desired type of parking function, and another by counting (by involution) all of the desired parking functions that are also restricted as desired. In each case, both counts are connected by Abel's binomial theorem.

Abel's binomial theorem has many equivalent statements, but perhaps most common is the following version:

Start with words

Theorem 3.10 (Abel's binomial theorem)**.**

$$(x+y+n)^n = \sum_{i=0}^{n} \binom{n}{i} x(x+i)^{i-1}(y+n-i)^{n-i}.$$

Example environment.

For example, we can rewrite Theorem 1.1 as

$$s^c = \sum_{i=0}^{c} \binom{c}{i} (i+1)^{i-1}(s-i-1)^{c-i}$$

which is just given by Abel's theorem with $x = 1$, $y = s-c-1$, $n = c$ and $k = i$.

Example environment.

We can also rewrite Theorem 3.9 as

$$-(s-1)^c = \sum_{i=0}^{c} \binom{c}{i} (i-1)^{i-1}(s-i)^{c-i},$$

$$\#\mathrm{PPF}_{c|[s]} = s^c - (s-1)^c - \sum_{i=1}^{s} \binom{c}{i}(i-1)^{i-1}(s-i)^{c-i}$$
$$= \sum_{i=s+1}^{c} \binom{c}{i}(i-1)^{i-1}(s-i)^{c-i}$$

noting that the $-s^c$ term appears when $i = 0$ as $(-1)^{-1}(s-0)^{c-0}$. This is Abel's theorem with $x = -1$, $y = s-c$, $n = c$ and $i = i$. These are perhaps the two most natural-to-interpret cases of Abel's identity, from a combinatorial point of view: they alone have the $i = 0$ case output an integer, as $(-1)^{-1}$ and 1^{-1} respectively.

Example environment.

Last but not least, our enumerator for parking functions with restricted preferences based on their number of ones may be rewritten as

$$(s-1+x)^c = \sum_{i=0}^{c} \binom{c}{i} x(i+x)^{i-1}(s-1-i)^{c-i}$$

which is given by Abel's theorem with $x = x$, $y = s-c-1$, $n = c$, and $i = i$. Notably, this recovers the complete identity! To see this, note that the above statement is true for all values of x and for the c integer values of s from 1 to c. Subtracting s^c yields on both sides a polynomial which is of degree $c-1$ in s on both sides (true through expanding out each term).

Two polynomials of degree $c-1$ are equal at all values if they are equal for c different values. Thus we have equality for all values of x and c; substituting $x = x$, $s = y+c+1$, $c = n$, and $i = i$ gives us the initial identity.

In particular, our claim is proven for all values of x and all integer values of s from 1 to c; since both sides of our identity are polynomials of degree c in the variable s, only one further value of s needs to be verified in order to show the full theorem as a consequence.

A combinatorial proof of Abel's binomial theorem, much inspired by [Sha91], can be found in appendix B along with a brief description of the connections between parking functions and reluctant functions.

4. Modular restrictions

4.1. **Enumerative results.** In [BK77], Blake and Konheim use generating functions and methods from complex analysis to prove the following result. We give a much simpler combinatorial proof that is generalisable to the case of c cars with $c+2$ spots too.

Theorem 1.2. *Let S be the set of the first s natural numbers j with $j \equiv 1 \pmod{g}$. Then the number of parking functions of length $gs-1$ with gap g between possible preferred spots is $\#\mathrm{PF}_{gs-1|S} = s^{gs-2}$.*

$\{j \in [gs-1] \mid j \equiv 1 \pmod{g}\}$

Proof. Consider $gs-1$ cars attempting to park in a circular parking lot with gs spots, in which cars can only park in s spots spaced evenly around the circle with gap g between them. There are s^{gs-1} different preference functions that come out of this, and every car will be able to park.

There will be one empty spot. This spot will appear directly before a car where preferences are legal, since the only way for a full spot to appear in front of an empty spot is for someone to have preferred the full spot. Thus removing this spot yields a parking function on $gs-1$ spots; preferences will be 1 mod g by virtue of the position of the spot removed.

Conversely, every modularly-restricted parking function will yield circular parking functions of the form described through adding an empty spot. The orientation of this circle depends only on where the empty spot is placed; this can be done in s ways, one for each of the s spots which can be preferred.

Thus the total number of parking functions with modular restrictions is $\frac{1}{s}$ times the number of corresponding circular preference lists, or $\frac{s^{gs-1}}{s} = s^{gs-2}$. □

Theorem 4.1. *Let S be the set of the first s natural numbers j with $j \equiv 1 \pmod{g}$. Then the number of parking functions of length $gs-2$ cars with gap g between possible preferred spots is*

$$\#\mathrm{PF}_{gs-2|S} = s^{gs-3} - \frac{1}{2}\sum_{i=1}^{s-1} \binom{gs-2}{gi-1} i^{gi-2}(s-i)^{g(s-i)-2}$$

Proof. Consider $gs-2$ cars attempting to park in a circular parking lot with gs spots, in which cars can only park in s spots spaced evenly around the circle with gap g between them. There are s^{gs-2} different preference functions that come out of this, and every car will be able to park.

There will be two empty spots, which may be adjacent or nonadjacent. If the two are adjacent, they will both be prior to a preferable spot; removing them will yield a modularly-restricted parking function of length $gs-2$, and conversely every such parking function yields s different preference functions on a circular lot. There will therefore be $s \cdot \#\mathrm{PF}_{gs-2|S}$ circular preference functions of this type.

If the two are nonadjacent, both will be prior to distinct preferable spots and will have a gap between them. The first empty spot may be placed in one of s different ways. This empty spot could have a gap of of $g, 2g, \ldots, (s-2)g$ or $(s-1)g$; we denote this gap as gi.

For this i, there will be $\binom{gs-2}{gi-1}$ ways to choose the cars that go in the $gi-1$ filled spots. These are filled by a modularly-restricted parking function on $gi-1$ spots, which by the above theorem can occur in i^{gi-2} ways. Finally, the remaining $g(s-i)-1$ spots are filled by a modularly-restricted parking function; this can be done in $(s-i)^{g(s-i)-2}$ ways. We are overcounting by a factor of 2, once for each empty spot, so the total number of circular preference functions with nonadjacent empty spots is

$$\frac{1}{2}s \cdot \sum_{i=1}^{s-1} \binom{gs-2}{gi-1} i^{gi-2}(s-i)^{g(s-i)-2}.$$

Thus we have the decomposition

$$s^{gs-2} = s \cdot \#\mathrm{PF}_{gs-2|S} + \frac{1}{2}s \cdot \sum_{i=1}^{s-1} \binom{gs-2}{gi-1} i^{gi-2}(s-i)^{g(s-i)-2};$$

solving for $\#\mathrm{PF}_{gs-2|S}$ gives our desired formula. □

This technique may be extended as necessary to $gs-3$, $gs-4$, etc. spots; however, the extent of the casework and nesting of sums required renders such formulas rather infeasible to calculate and use.

Note that given s possible preferences equal to 1 mod g on $gs-1$-spot parking lot, and given randomly chosen preferences, the probability that we find a parking function is $\frac{s^{gs-2}}{s^{gs-1}} = \frac{1}{s}$. Again, considering any other cases gives significant complications; however, we conjecture that when keeping i and g constant and varying s, the probability that a restricted preference function on $gs-i$ spots is a parking function will remain $O\left(\frac{1}{s}\right)$.

5. Random restrictions

Suppose that we randomly sample a subset of $[c]$, say $S \in \mathcal{P}([c])$. What do the parking functions in $\mathrm{PF}_{c|S}$ look like?

Lemma 5.1 (Partition by image size)**.** *The number of parking functions π on $[c]$ such that the image of π has i elements is $S(c,i)P(c,i-1)$, where $S(n,k)$ is a Stirling number of the second kind and $P(n,k) = \frac{n!}{k!}$ counts permutations. In particular, $\sum_{i=1}^{c} S(c,i)P(c,i-1) = (c+1)^{c-1}$.*

Proof. Our cars will go into i partitions, each of which we will assign to a distinct preference. The number of ways to partition c cars into i partitions is $S(c,i)$.

We proceed by a circle argument: given these i partitions, imagine placing them in distinct spots on a circular parking lot of size $c+1$. There are $(c+1)(c)(c-1)\cdots(c-i+2)$ ways to make this choice. This will lead to all cars parking since there are more cars than spots, with one open spot remaining; removing this open spot yields a linear parking lot, containing a parking function of length c.

Due to the $c+1$ different possible open spots for each parking function, and by symmetry on the circular parking lot, we are overcounting parking functions by a factor of $c+1$. Thus we are left with $c(c-1)\cdots(c-i+2) = P(c,i-1)$ ways to make a parking function out of our partitions. This gives the desired total of $S(c,i)P(c,i-1)$. □

We can now count the sum over all subsets $S \subseteq [c]$ of $\#\mathrm{PF}_{c|S}$. This sum is the size of the set of ordered pairs (S,π), where π is a parking function on c cars, all of whose preferences lie in S.

We may count the elements of this set through casework on the size of the image of π. For a given $i \in [c]$, there are $S(c,i)P(c,i-1)$ parking functions π with an image of size i, and for each such parking function, there are 2^{c-i} choices of S which are supersets of the image of $[c]$. Thus our total number is

$$\sum_{i=0}^{c} 2^{c-i} S(c,i)P(c,i-1).$$

In particular, the average number of parking functions under a randomly chosen restriction to $S \in \mathcal{P}([c])$ will be $\sum_{i=0}^{c} 2^{-i} S(c,i)P(c,i-1)$; intriguingly, this corresponds to half of A007889, which counts the number of regions in the c-dimensional Linial hyperplane arrangement.

6. Sampling and decompositions

In order to understand a set of objects better, it's often useful to find a way to sample randomly from that set. [DH17] notably uses this technique for the entire set of ordinary parking functions PF_n, analyzing from this procedure an assortment of statistics on parking functions.

We consider a general procedure for sampling near-uniformly from a set of parking functions with restricted preferences. This may be applied in two different ways to sample from the set of preference lists with a given defect d.

Definition 6.1. Given a preference list π of length c, simulating a parking process on a lot of size c yields some number of cars which are not able to park. The size of this set of cars is called the **defect** of the preference list.

Consider a preference list of length c with defect $d \geq 0$. We may consider this as a preference list of length c on $[c+d]$, such that every car parks and every car has preferences in $[c]$. This preference list will have exactly d empty parking spots, and the last $d+1$ spots will be full; this occurs because the cth spot must be filled, so that d cars can leave the first c spots.

We wish to partition this set of preference lists into a relatively small number of subsets which can be more easily sampled.

Proposition 6.2. *The number of preference lists of length c with defect d may be expressed as*

$$\sum_{i=1}^{c-d} \binom{c}{i+d} \cdot \#\mathrm{PF}_{i+d|[i]} \cdot d(c-i)^{c-i-d-1}$$

or as

$$\sum_{i=1}^{c-d} \binom{c}{i+d} \cdot \#\mathrm{PPF}_{i+d|[i]} \cdot (d+1)(c-i+1)^{c-i-d-1}.$$

Proof. For the first decomposition, we use casework on the location of the last empty spot. This may occur as early as the dth spot, or as late as the $c-1$th spot. In particular, let i denote the number of parking spots which occur after this last empty spot and up to the cth spot; i will range from 1 to $c-d$.

There will be $i+d$ cars which park after the last empty spot, and $c-i-d$ which park before it; there are $\binom{c}{i+d}$ ways to make the split. After the empty spot, we will have a parking function of length $i+d$ in which all cars have preferences in the first i spots, since we have all spots filled; the number of these is $\#\mathrm{PF}_{i+d|[i]}$.

Finally, before the last empty spot, we have $c-i-d$ cars which must all park in the first $c-i-1$ spots. In general, given $k \leq n$ cars which must all park in n spots, there are $(n-k+1)(n+1)^{k-1}$ ways to achieve this; this elegant result is shown in [CJPS08] among other places.

In our case, there are $d(c-i)^{c-i-d-1}$ such functions, for a total of $\binom{c}{i+d} \cdot \#\mathrm{PF}_{i+d|[i]} \cdot d(c-i)^{c-i-d-1}$ functions for a particular i Summing over all feasible i gives

$$\sum_{i=1}^{c-d} \binom{c}{i+d} \cdot \#\mathrm{PF}_{i+d|[i]} \cdot d(c-i)^{c-i-d-1}.$$

Alternatively, we use casework on the location of the last breakpoint before spot $c+d$. This may again occur as early as the dth spot or as late as the $c-1$th spot; define i similarly to above. (Note that empty spots are counted as breakpoints here.)

There will again be $\binom{c}{i+d}$ ways to select cars which park after the breakpoint. The cars after the breakpoint will again form a parking function of length $i+d$ with all preferences in the first i spots; here we have the additional constraint that there is only one breakpoint in the function, meaning that it is prime. The number of these is then $\#\mathrm{PPF}_{i+d|[i]}$.

Up to the last breakpoint, we have $c-i-d$ cars which must all park in the first $c-i$ spots. There are $(d+1)(c-i+1)^{c-i-d-1}$ ways to do this, for a product of $\binom{c}{i+d} \cdot \#\mathrm{PPF}_{i+d|[i]} \cdot (d+1)(c-i+1)^{c-i-d-1}$ and a sum over all possible i of

$$\sum_{i=1}^{c-d} \binom{c}{i+d} \cdot \#\mathrm{PPF}_{i+d|[i]} \cdot (d+1)(c-i+1)^{c-i-d-1}.$$

□

Sampling from one of $c-d$ subsets, with weighted probabilities, is significantly easier than sampling from all preference lists with a given defect uniformly. We now need only sample uniformly from each of our subsets.

Sampling from the $\binom{c}{i+d}$ ways to select $i+d$ elements of $[c]$ is fairly trivial. Sampling from the $(n-k+1)(n+1)^{k-1}$ ways to park $k<n$ cars in n spots can also be done with a fairly simple procedure, inspired by the method of [DH17] for sampling a random parking function:

(1) Pick a random element $\pi \in (\mathbb{Z}/(n+1)\mathbb{Z})^k$.
(2) Viewing elements of $\mathbb{Z}/(n+1)\mathbb{Z}$ as spots in a circular parking lot, let cars 1 through k prefer spots $\pi(1)$ through $\pi(k)$ and run our parking algorithm.
(3) Randomly choose one of the $n-k+1$ empty spots remaining and remove it, leaving a linear parking lot of n spots in which k cars park successfully.

Every successful preference list is represented $n+1$ times by this algorithm, once for each empty spot from $\mathbb{Z}/(n+1)\mathbb{Z}$ that could have been removed. Thus our choice is uniform over all successful lists.

Finally, we'd like to find a way to sample from either parking functions with preferences restricted to an initial segment or prime parking functions with a similar restriction. By the natural bijection given in Proposition 3.8, we only need a method to sample from restricted parking functions in general– a question which is of interest in its own right.

For the defect-1 case, our task is relatively easy through sampling prime parking functions: we need only look at $\mathrm{PPF}_{c|[c-1]}$, which is equivalent to PPF_c in general since no prime parking function has preferences in its last spot. Similar to our procedure for parking functions in general, we may sample from prime parking functions by taking c random preferences in $\mathbb{Z}/(c-1)\mathbb{Z}$ and creating a corresponding prime parking function through the method of proposition 2.8. However, the task is much harder for most preference restrictions; there is no clear decomposition in general, so we need to worry more about sampling from restricted parking functions themselves.

Given $S \subseteq [c]$, one approach to sampling from $\mathrm{PF}_{c|S}$ is to repeatedly sample from the set S^c of all restricted preference lists, stopping when one arrives at a parking function. The expected length of this procedure is relatively well-studied for initial-segment restricted parking functions, in particular by [SY99].

there are some good bounds here, we need to write them up though

When not looking at initial-segment restrictions, though, this process can get out of hand. Take $\mathrm{PF}_{c|S}$, for example where $S = \{1, c\}$. There are 2^c elements in $\{1, c\}^c$ to sample from, but only $\binom{c}{0} + \binom{c}{1} = c + 1$ of these preference lists are parking functions (namely those with at most one car preferring spot c). It takes exponential time for such an algorithm to stop!

One alternative approach involves a lazy random walk along parking functions. Given a subset $S \subseteq [c]$ such that $1 \in S$, and given a parking function $\pi \in \mathrm{PF}_{c|S}$, we may randomly select another parking function $\pi' \in \mathrm{c} \mid \mathrm{S}$ as follows:

(1) With probability $\frac{1}{2}$, let $\pi' = \pi$.
(2) Otherwise, randomly select an $i \in [c]$, and either 'up' or 'down' with probability $\frac{1}{2}$.
(3) Let $\pi_i(j)$ be the preference function generated by replacing the ith preference with the value j.
(4) If our direction is 'up,' let k be the smallest element of S such that $k > \pi(i)$ and $\pi_i(k)$ is a parking function; if it's 'down', let k be the largest element of S such that $k < \pi(i)$ and $\pi_i(k)$ is a parking function.
(5) If there exists such a k, let $\pi' = \pi_i(k)$; otherwise, let $\pi' = \pi$.

This process always yields a parking function, giving a Markov chain with set of states $\mathrm{PF}_{c|S}$.

Proposition 6.3. *Applying the above Markov chain repeatedly to any initial distribution on parking functions yields a sequence which converges in distribution to the uniform distribution on* $\mathrm{PF}_{c|S}$.

Proof. This Markov chain is aperiodic and irreducible. Aperiodicity occurs when for every state π in the chain, the gcd of the lengths of all possible random walks which start and end at π is 1. This condition is satisfied here due to our first step, which ensures a cycle of length 1 always exists.

Irreducibility occurs when there always exists a walk of finite length from any π to any π'. To show this, we need only prove that there exists a walk from any π to $(1, 1, 1, \ldots, 1)$ and vice versa. The first direction can be achieved by reducing each preference with 'down' until it reaches 1, noting that reducing a preference on a parking function keeps it as a parking function. The opposite direction can be achieved simply by reversing the order of each step with 'up'.

By the Fundamental Theorem of Markov Chains, as described by [Bis22], any aperiodic and irreducible Markov chain has exactly one probability distribution over it which is preserved under a step of the chain, and any initial distribution over our set of parking functions converges in distribution to this stationary distribution under repeated application of our Markov chain. We'd like to show that this stationary distribution is uniform.

Consider, therefore, the uniform distribution on $\mathrm{PF}_{c|S}$. Note that any 'up' step that moves us to a different parking function is reversible by a 'down' step from that parking function, and both have the same probability of occurring. This means our transition matrix T is symmetric, and in particular that the sums of its rows $\sum_j T_{ij}$ equal 1 just as the sums of its columns do. Because of this, a uniform distribution where each state has probability p will be mapped to a distribution where each state has probability $\sum_j T_{ij} p = p$. That is, the uniform distribution is stationary, and so every other distribution converges to it. □

This means that by following our procedure a sufficient number of times, starting at a particular parking function, we can arrive at a distribution arbitrarily close to uniform on parking functions. The rate of convergence is much harder to determine– even for parking functions in general, the structure of the Markov chain is hard to grasp. The authors of this paper, however, suspect that the rate of convergence is relatively quick– or at least significantly better than the worst-case scenario outlined above.

Definition 6.4. Given $\epsilon > 0$, the mixing time $\tau(\epsilon)$ for Markov chain M is the smallest number N such that any initial distribution iterated through N steps of the Markov chain has distance at most ϵ from the unique stationary

distribution. The distance between two distributions is defined here by

$$\|D_1 - D_2\| = \frac{1}{2} \sum_{v \in M} |D_1(v) - D_2(v)|.$$

Conjecture 6.5. *There exist absolute constants $\delta, k \in \mathbb{R}^+$, $n \in \mathbb{N}$, such that for all $0 < \epsilon < \delta$, $c \in \mathbb{N}^{>1}$, $S \subseteq [c]$, the mixing time for the Markov chain above corresponding to $\mathrm{PF}_{c|S}$ is bounded above by*

$$\tau(\epsilon) \leq k \log(1/\epsilon) \cdot c^N.$$

6.1. **$[s]$-restricted parking functions.**

6.2. **Defect-$[d]$ preference lists.**

Appendix A. Prime parking functions

Proposition 2.7. *A parking function is prime if and only if it has exactly one breakpoint. Furthermore, there exists a natural decomposition of any parking function π into b prime parking segments, where b is the number of breakpoints in π.*

Proof. Given some $i \in [c-1]$, if at least $i+1$ cars have preferences in the first i spots, then at least one of those cars will not park in the first i spots. This car will therefore cross the point between the ith and $i+1$th parking spots, meaning that this spot will not be a breakpoint. Conversely, if at most i cars have preferences in the first i spots, all of the first i spots will be filled by said cars since we have a parking function. Thus no car will leave spot i, meaning that the point between the ith and $i+1$th parking spots will be a breakpoint.

Therefore, at least $i+1$ cars have preferences in the first i spots if and only if there is no breakpoint directly after the ith spots. By extension, a parking function is prime if and only if there are no breakpoints at all.

Given b breakpoints, we can divide $[c]$ into b intervals, each of which ends at its corresponding breakpoint. A given interval contains ℓ_j spots for $j \in [b]$. It will have ℓ_j cars parked in it, since π is a parking function; since no cars will enter or exit the interval, it being bounded by breakpoints, there will be ℓ_j cars with preferences in the interval as well. This may be considered a parking function of length ℓ_j. Furthermore, since this interval has no breakpoints within it, it is a prime parking function; thus we have our decomposition. □

Proposition 2.8. *The number of prime parking functions of length c is $(c-1)^{c-1}$.*

Proof. Consider c cars attempting to park in a circular parking lot with $c-1$ spots; there are $(c-1)^c$ different preference functions that come out of this.

The first $c-1$ cars will all park. There will be at least one breakpoint since no car will pass the location where the $c-1$th car parked.

The last car will circle around indefinitely after the first $i-1$ cars fill their respective spots. It will therefore cross each breakpoint. Take the last breakpoint which this cth car crosses; we may break the circle at this point, forming a parking function of length $c-1$, and place the cth car at the end.

This will yield a prime parking function; Every parking spot is filled, and every breakpoint from the first $c-1$ cars is crossed by the last car.

Conversely, every prime parking function will yield circular parking functions of the form described. The last car will always park at the end, since otherwise the point ahead of the last car's parking outcome would be a breakpoint. The remaining $c-1$ cars form a parking function which can be wrapped around our circle; this can be done in $c-1$ ways, one for each spot that can correspond to the first spot in our prime parking function.

Thus the total number of prime parking functions of length c is $\frac{1}{c-1}$ times the number of corresponding circular preference lists, or $\frac{(c-1)^c}{c-1} = (c-1)^{c-1}$. □

Appendix B. Abel's binomial theorem

briefly describe the connection between binomial-type polynomials like $x(x \pm an)^{n-1}$ and ball-distribution

Theorem 3.10 (Abel's binomial theorem)**.**

$$(x+y+n)^n = \sum_{i=0}^{n} \binom{n}{i} x(x+i)^{i-1}(y+n-i)^{n-i}.$$

Proof sketch. We can prove the identity for all $x, y \in \mathbb{N}$ by analogy to words.

The left hand side clearly counts the number of words of length-n in the alphabet $A \sqcup B \sqcup [n]$ where $\#A = x$ and $\#B = y$. We notice that the positions in the length-n word can be partitioned in two by the following greedy algorithm

(1) If there are no characters from A in the word, then $S_x = \emptyset$.
(2) For all $i \in [n]$, if $f(i) \in A$, then add i to S_x.
(3) For all $i \in [n]$, if $f(i) \in S_x$, then add i to S_x.
(4) Repeat step (3) until there are no more i such that $f(i) \in S_x$ but $i \notin S_x$.
(5) Partition $[n]$ into S_x and $S_y = [n] \setminus S_x$

This is equivalent to thinking of a word as a function $f : [n] \to A \sqcup B \sqcup [n]$ and choosing the subset S_x of all s such that $f(s) \in A$ or $f(\cdots f(s)) \in A$. For the reader familiar with the theory of *reluctant functions* as exposited in [MR70], S_x is the largest subset of $[n]$ for which f is a reluctant function on $A \sqcup S_x$.

decide whether to give background or just [MR70]

We will show that the right hand side corresponds to the number of ways to choose S_x and place characters in it so that the reluctant condition holds. Given any choice of S_x, call $\#S_x = k$. We can choose the characters in positions in S_y freely from $B \sqcup S_y$. That is, we can do so in one of $(y+n-k)^{n-k}$ ways. In [Sha91], Shapiro shows crucially (in proving the more general Hurwitz formula), that we can also choose almost all of the characters in S_x freely. We can freely choose characters from $A \sqcup S_x$ for all but the last position — $\max S_x$.

Let the word correspond to a function f. We choose $f(S_x \setminus \{\max S_x\})$ from $A \sqcup S_x$ in one of $(x+k)^{k-1}$ ways. If there are no cycles in f as defined so far, define $f(\max S_x) \in A$ (in one of x ways). Else, order all of the cycles of f in S_x by the largest i contained by them, and then unravel them as in [Sha91]. In the digraph representation of f, this means breaking cycles at the edge mapping to their maximum element and then mapping that vertex to the maximum element of the cycle with the next largest maximum element instead. Mapping the last vertex to some character in A, we have converted f into a forest of trees all rooted at A, and thus, a reluctant function. We can recover this arbitrary choice from any reluctant function we get. After getting S_x from the greedy algorithm, look at the path from $\max S_x$ to A. Excluding $\max S_x$, start a new cycle every time a new maximum is reached. Thus, the $x(x+k)^{k-1}$ ways to perform this procedure count exactly the number of ways to choose the characters so that the reluctant condition holds on S_x. Multiplying by the choices for S_y and the number of ways to choose S_x itself, we get $\binom{n}{k}x(x+k)^{k-1}(y+n-k)^{n-k}$. When summed over all k, this is exactly Abel's binomial theorem. □

Remark B.1. Note that though the identity above isn't the "complete" form of Abel's binomial theorem, we can recover the complete form by substituting $x = X/Z$ and $y = Y/Z - n$ to get

$$(X+Y)^n = \sum_{k=0}^{n} \binom{n}{k} X(X+kZ)^{k-1}(Y-kZ)^{n-k}.$$

References

[AL99] Christos A. Athanasiadis and Svante Linusson. A simple bijection for the regions of the Shi arrangement of hyperplanes. *Discrete Math.*, 204(1-3):27–39, 1999.

[ALW16] Drew Armstrong, Nicholas A. Loehr, and Gregory S. Warrington. Rational parking functions and Catalan numbers. *Ann. Comb.*, 20(1):21–58, 2016.

[Bai96] D. F. Bailey. Counting arrangements of 1's and -1's. *Mathematics Magazine*, 69(2):128–131, 1996.

[Bis22] Somenath Biswas. Various proofs of the fundamental theorem of markov chains, 2022.

[BK77] Ian F. Blake and Alan G. Konheim. Big buckets are (are not) better! *J. Assoc. Comput. Mach.*, 24(4):591–606, 1977.

[BQ08] Arthur T. Benjamin and Jennifer J. Quinn. An alternate approach to alternating sums: A method to DIE for. *The College Mathematics Journal*, 39(3):191–202, 2008.

[CJPS08] Peter J. Cameron, Daniel Johannsen, Thomas Prellberg, and Pascal Schweitzer. Counting defective parking functions. *Electron. J. Combin.*, 15(1):Research Paper 92, 15, 2008.

[DH17] Persi Diaconis and Angela Hicks. Probabilizing parking functions. *Adv. in Appl. Math.*, 89:125–155, 2017.

[MHJ+23] Lucas Chaves Meyles, Pamela E. Harris, Richter Jordaan, Gordon Rojas Kirby, Sam Sehayek, and Ethan Spingarn. Unit-interval parking functions and the permutohedron, 2023.

[MR70] Ronald Mullin and Gian-Carlo Rota. On the foundations of combinatorial theory III: Theory of binomial enumeration. *Graph theory and its applications*, 1970.

[Pin24] Ross G. Pinsky. The distribution on permutations induced by a random parking function, 2024.

[Rio69] John Riordan. Ballots and trees. *Journal of Combinatorial Theory*, 6(4):408–411, 1969.

[Sha91] Louis W. Shapiro. Voting blocks, reluctant functions, and a formula of Hurwitz. *Discrete Mathematics*, 87(3):319–322, 1991.

[SY99] J. Spencer and Catherine H. Yan. A ball-distribution problem and the branching process, 1999.

[Yan15] Catherine H. Yan. Parking functions. In *Handbook of enumerative combinatorics*, Discrete Math. Appl. (Boca Raton), pages 835–893. CRC Press, Boca Raton, FL, 2015.

(A. Kappler) Harvey Mudd College, United States
Email address: akappler@g.hmc.edu

(J. Thadani) Harvey Mudd College, United States
Email address: jthadani@hmc.ed.