



**HAL**  
open science

## Bilateral parking procedures

Philippe Nadeau

► **To cite this version:**

| Philippe Nadeau. Bilateral parking procedures. 2023. hal-04262134

**HAL Id: hal-04262134**

**<https://hal.science/hal-04262134v1>**

Preprint submitted on 27 Oct 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# BILATERAL PARKING PROCEDURES

PHILIPPE NADEAU

ABSTRACT. We introduce the class of *bilateral* parking procedures on  $\mathbb{Z}$ . These generalize the classical one as follows: Instead of having cars systematically park in the nearest available spot on their right, we also allow cars to use the nearest spot to their left. We will distinguish the subclass of *local* procedures, which has the striking property that for any procedure of the class, the number of corresponding *parking functions* of length  $r$  is given by  $(r+1)^{r-1}$ .

We show how to extend definitions and results to probabilistic procedures, in which the decision to park left or right is random. We finally describe how bilateral procedures can naturally be encoded by labeled binary forests, whose combinatorics explain and unify several results from the literature.

## 1. INTRODUCTION

Classical parking functions are one of the fundamental objects of enumerative and algebraic combinatorics. They are connected to various structures such as noncrossing partitions, hyperplane arrangements, and many others: see for instance the survey [16] and references therein. The corresponding parking procedure  $\mathcal{P}^{right}$  on  $\mathbb{Z}$  was originally defined as an elementary hashing procedure, cf. [10]. Since then, they have been generalized in a number of ways.

We recall the definition of the classical parking functions: Let  $r$  be a positive integer, and there are  $r$  cars that want to park on an empty street with spots labeled by  $1, 2, \dots, r$  from left to right. The cars arrive successively, and the  $i$ th car has a preferred spot  $a_i$ . If this spot is available, it parks there, and if not it parks in the *nearest available spot to the right*. The sequence  $(a_1, \dots, a_r)$  is called a *parking function* if at the end, all cars managed to park.

The main enumerative result is that the number of parking functions for  $r$  cars is given by the simple formula  $(r+1)^{r-1}$ . We have also the following characterization:  $a_1 \cdots a_r$  is a parking function if and only if for any  $k = 1, \dots, r$ , there are at least  $k$  indices  $i$  such that  $1 \leq a_i \leq k$ . Parking functions generalizations have often relied on the above characterization (often taken as the definition). This is the case of so-called  $G$ -parking functions and  $\cong$ -parking functions, see [16]. Recently, several variations were also considered by varying different aspects of the procedure, see for instance [3] and references therein.

In this work, we will consider the extensions of the parking procedure obtained by simply changing the “nearest available spot to the right” condition: we will also allow one to park to the nearest available spot *to the left*. This explains the name bilateral for our class of procedures.

Let us give already two examples of possible rules. We need only describe where to park when one’s preferred spot is occupied:

- *pclosest*: If the nearest available spot to the right is (weakly) closer than the nearest available spot to the left, park there, otherwise park to the left.

---

This research is partially supported by the French project ANR19-CE48-011-01 (COMBINÉ).

- $\mathcal{P}^{prime}$ : If the total number of cars parked between the nearest available spots to the left and to the right is a prime number, park to the right, otherwise park to the left.

We choose to describe the parking procedures  $\mathcal{P}$  that we want to consider, as functions associating to a preference word  $a_1 \cdots a_r$  a finite subset of  $\mathbb{Z}$ . In terms of parking,  $\mathcal{P}(a_1 \cdots a_r)$  is the set of occupied spots at the end of the procedure; if it is equal to  $\{1, \dots, r\}$  then we will say that  $a_1 \cdots a_r$  is a  $\mathcal{P}$ -parking function. Then some easy properties of the function  $\mathcal{P}$  then characterize what we mean by bilateral procedure.

We will then give two more properties to determine the subclass of *local* procedures: roughly put, these say that left/right decisions must be invariant under translation, and depend only on the cars that parked on the block. With such conditions we obtain the following striking enumerative result.

**Theorem 1.1.** *Let  $\mathcal{P}$  be a bilateral, local parking procedure. Then the number of  $\mathcal{P}$ -parking functions of length  $r$  is given by  $(r + 1)^{r-1}$ .*

In particular this holds for the classical procedure, for  $\mathcal{P}^{closest}$  and  $\mathcal{P}^{prime}$  defined above, and for an uncountable number of such procedures. One may interpret Theorem 1.1 as a kind of discrete *universality* result: a unique enumeration result holds for a large class of procedures where only light local conditions are imposed. The terminology is borrowed (and arguably stretched) from statistical physics.

The proof of this result will follow from Pollak’s argument in the classical case, based on a cyclic procedure derived from the original procedure.

We can naturally add randomness to a bilateral procedure: instead of choosing either left or right, choose probabilities adding to 1 for the two options. Now each preference word has a probability to be  $\mathcal{P}$ -parking, instead of being parking or not. For a particular procedure, we will see that these probabilities, suitably normalized, coincide with the family of remixed Eulerian numbers studied by Vasu Tewari and the author [12]. The notion of memoryless *local* procedure extends to the probabilistic setting, and a version of Theorem 1.1 holds in this case.

From the point of view of bijective combinatorics, we show that bilateral procedures are naturally encoded by certain pairs of labeled forests. This bijection naturally lifts the original parking procedure, and is directly connected to the outcome function of the procedure, which is the order in which the cars parked. We will recover some known results, including in the classical case.

Here is a short plan of the article. We first describe the parking procedures  $\mathcal{P}$  that we consider in Section 2. We will focus on bilateral procedures, and then introduce the subclass of local procedures in Section 3. We will then see that the enumeration  $(r + 1)^{r-1}$  is in a sense universal for local procedures, see Theorem 1.1. We then explain how to define a probabilistic version of our procedures in Section 4. We describe some natural connection with the combinatorics of binary trees via a natural encoding in Section 5. We finally define a “colored” version of the model in the last section.

## 2. BILATERAL AND LOCAL PARKING PROCEDURES

As stated in the introduction, the list of parking spot preferences of incoming cars is given by a preference word  $a_1 a_2 \cdots a_r$  with  $a_i \in \mathbb{Z}$ , read from left to right: the  $i$ th car wants to park in the spot  $a_i$ . A general parking procedure  $\mathcal{P}$  goes as follows: it is defined inductively, with no cars parked at the beginning. Assume the first  $i - 1$  cars have parked at distinct spots in  $\mathbb{Z}$ . If the spot

$a_i$  is available, the  $i$ th car parks there. If the spot is not available, then there is a rule in  $\mathcal{P}$  that determines an available parking spot for the car  $a_i$ . In the classical parking procedure, this rule consists in choosing systematically the nearest available spot to the right. More generally, bilateral procedures also allow the spot to be the nearest available spot to the left.

Let us point out that it is more customary to define parking procedures on an interval  $\{1, \dots, r\}$ , and say that the procedure fails as soon as a car cannot park. A parking function is then an entry word of length  $r$  where everyone managed to park. Here we define our procedures on  $\mathbb{Z}$ , which is more pleasing from a mathematical point of view since we don't have to deal with partially defined functions. Indeed all cars will find an available spot. A procedure can then be considered successful if the occupied spots at the end are  $\{1, 2, \dots, r\}$  (see Definition 2.5). This is how the boundary conditions are integrated to our setting.

**2.1. Parking procedures.** Let  $\text{Fin}(\mathbb{Z}) = \{I \subset \mathbb{Z} \mid \#I < +\infty\}$  be the collection of finite subsets of  $\mathbb{Z}$ . A parking procedure will be given by a function  $\mathcal{P} : \mathbb{Z}^* \rightarrow \text{Fin}(\mathbb{Z})$ . The subset  $\mathcal{P}(a_1 \cdots a_r)$  represents the set of occupied spots after cars with preferences  $a_1, \dots, a_r \in \mathbb{Z}$  have parked successively. This leads to the following definition:

**Definition 2.1** (*Parking procedure*). A function  $\mathcal{P} : \mathbb{Z}^* \rightarrow \text{Fin}(\mathbb{Z})$  is a parking procedure if  $\mathcal{P}(\epsilon) = \emptyset$  and for any  $r \geq 1$  and any word  $a_1 \cdots a_r$ ,

- (1) The subset  $\mathcal{P}(a_1 \cdots a_r)$  has cardinality  $r$ , and for any  $i < r$

$$\mathcal{P}(a_1 \cdots a_i) \subset \mathcal{P}(a_1 \cdots a_{i+1});$$

- (2) If  $a_r \notin \mathcal{P}(a_1 \cdots a_{r-1})$ , then  $\mathcal{P}(a_1 \cdots a_r) = \mathcal{P}(a_1 \cdots a_{r-1}) \sqcup \{a_r\}$ .

The first condition indicates that everyone manages to park (and does not move afterwards). The second condition expresses that one parks at their favorite spot if it is available. Note that at this stage cars may possibly park at any available spot, possibly very far, when their desired spot is not available. We will soon restrict to bilateral procedures, but in this section we choose to give definitions at this level of generality where they make sense.

*Remark 2.2.* These conditions capture reasonable conditions of what a “real-life” parking procedure on  $\mathbb{Z}$  should satisfy. Note however that in a recent work [9], cars can be dislodged from their parking spot by a later car, which is not covered by our setup.

**2.2. Memoryless parking procedures.** Definition 2.1 includes the possibility that the parking decisions may depend on the whole sequence of preferred spots  $a_1, \dots, a_i$ . A natural subclass of parking procedures, which includes the classical one  $\mathcal{P}^{\text{right}}$  as well as  $\mathcal{P}^{\text{prime}}$ ,  $\mathcal{P}^{\text{closest}}$  from the introduction, consists of those where the parking decisions only depend on the set of occupied spots at the moment where one wants to park. This can be expressed as follows:

**Definition 2.3.** A parking procedure  $\mathcal{P}$  is *memoryless* if there exists a function  $M_{\mathcal{P}} : \text{Fin}(\mathbb{Z}) \times \mathbb{Z} \rightarrow \mathbb{Z}$  such that  $\mathcal{P}(a_1 \cdots a_r a) = \mathcal{P}(a_1 \cdots a_r) \sqcup \{M_{\mathcal{P}}(\mathcal{P}(a_1 \cdots a_r), a)\}$  for any  $a_1, \dots, a_r, a$ .

The function  $M_{\mathcal{P}}$  characterizes the parking procedure  $\mathcal{P}$  by immediate induction. By definition of our procedures, it necessarily satisfies  $M(S, a) \notin S$  for any  $S, a$ , and  $M(S, a) = S \sqcup \{a\}$  whenever  $a \notin S$ . Conversely, a memoryless parking procedure is given by *any* family of integers  $M(S, a) \in \mathbb{Z}$  for  $a, S$  such that  $a \in S$ , that satisfy  $M(S, a) \notin S$ . In words,  $M(S, a)$  specifies the spot where a car with preferred spot  $a$  parks, when  $a$  is in the set  $S$  of occupied spots.

Let us now give an example of a parking procedure that is *not* memoryless. The reader will have no trouble finding other examples, even simpler to express than the following one. We include it here since it was the starting point and motivation for this work.

**Example 2.4.** The procedure  $\mathcal{P}^{LBS}$  is introduced and studied by Vasu Tewari and the author in [13]. It is defined inductively as follows: Let  $W = a_1 \cdots a_r$  be any word and  $a$  such that  $a \in \mathcal{P}^{LBS}(W)$ . We need to define  $\mathcal{P}^{LBS}(Wa)$ , which is done as follows:

Let  $I$  be the block of  $\mathcal{P}^{LBS}(W)$  such that  $a \in I$ . Define  $j \in \{1, \dots, r\}$  to be maximal such that  $a_j \in I$ : in words, the  $j$ th car is the last car that parked on  $I$ . Then  $\mathcal{P}^{LBS}(Wa)$  is defined by parking in the nearest spot available left of  $I$  if  $a < a_j$ , and right of  $I$  if  $a \geq a_j$ .

Thus one needs to record, for each block, where the last car that parked there had wanted to park: the procedure is not memoryless. To give an explicit example, it sends both words 12 and 21 to the set  $S = \{1, 2\}$  (this is true for any procedure of course), but sends 121 to  $\{0, 1, 2\}$  and 211 to  $\{1, 2, 3\}$ .

2.2.1.  *$\mathcal{P}$ -parking functions.* We finally come to our definition our parking functions associated to a parking procedure.

**Definition 2.5** ( *$\mathcal{P}$ -parking functions*). Let  $\mathcal{P}$  be a parking procedure. A word  $a_1 \cdots a_r$  is said to be parking for  $\mathcal{P}$ , or a  *$\mathcal{P}$ -parking function*, if  $\mathcal{P}(a_1 \cdots a_r) = \{1, \dots, r\}$ .

It is clear that this coincides with the usual notion of parking functions when  $\mathcal{P} = \mathcal{P}^{right}$ . Let  $\text{Park}(\mathcal{P})$  be the set of parking functions for  $\mathcal{P}$ , and  $\text{Park}_r(\mathcal{P})$  those of length  $r$ . The standard enumeration result recalled in the introduction is  $\#\text{Park}_r(\mathcal{P}^{right}) = (r+1)^{r-1}$ .

An immediate remark is that for any  $\mathcal{P}$ -parking procedure, all  $r!$  permutations of the word  $12 \cdots r$  are in  $\text{Park}_r(\mathcal{P})$ : indeed these are the cases where everyone finds their preferred spots available.

2.2.2. *Functions  $\text{ls}_{\mathcal{P}}$  and  $\pi_{\mathcal{P}}(W)$ .* Let us define two functions associated to a parking procedure  $\mathcal{P}$ .

The function  $\text{ls}_{\mathcal{P}}$  from  $\mathbb{Z}^*$  to  $\mathbb{Z}$  associates to a preference word the spot in  $\mathbb{Z}$  where the last car parks. Explicitly,  $\text{ls}_{\mathcal{P}}(a_1 \cdots a_i) = k$  where  $\{k\} = \mathcal{P}(a_1 \cdots a_i) \setminus \mathcal{P}(a_1 \cdots a_{i-1})$ . If  $W = a_1 \cdots a_r$  is any word, then the *outcome function*  $\pi_{\mathcal{P}}(W)$  is defined as follows:

$$\pi_{\mathcal{P}}(W) : \mathcal{P}(W) \rightarrow \{1, \dots, r\}, \quad i \mapsto \text{ls}(a_1 a_2 \cdots a_i)$$

is a bijection. It associates to an occupied spot  $k$  the integer  $i$  such that the  $i$ th car parked in  $l$ . When  $W$  is  $\mathcal{P}$ -parking, we have that  $\pi_{\mathcal{P}}(W)$  is a permutation of  $\{1, \dots, r\}$ .

A natural question, well studied in the case of the classical parking procedure, is to compute, given a permutation  $\sigma$ , the number of (parking) words  $W$  such that  $\pi_{\mathcal{P}}(W) = \sigma$ . This naturally partitions parking words of length  $r$  in  $r!$  classes. We will come back to this question in Section 5.

2.3. **Bilateral parking procedures.** In this work, we will focus on *bilateral* parking procedures: when one's preferred parking spot is not available, then the chosen spot must be either the nearest available spot to the left or the nearest available spot to the right –note that even though our goal is not to define real-life parking procedures, the constraint is fairly natural in order for cars to have a small displacement. All procedures considered until now ( $\mathcal{P}^{right}$ ,  $\mathcal{P}^{closest}$ ,  $\mathcal{P}^{prime}$ ,  $\mathcal{P}^{LBS}$ ) are bilateral

Let us give a mathematical definition and fix some terminology and notations in the process. Given a subset  $S$  of  $\mathbb{Z}$ , we say that a discrete interval  $I = \{t, t+1, \dots, u\}$  with  $t \leq u$  in  $\mathbb{Z}$  is a *block*

of  $S$  if it is included in  $S$ , and maximal for this property with respect to inclusion. For instance, if  $S = \{2, 3, 5, 6, 7, 8, 9, 12\}$ , its blocks are  $\{2, 3\}$ ,  $\{5, 6, 7, 8, 9\}$  and  $\{12\}$ .

**Definition 2.6** (*Bilateral parking procedure*). A parking procedure  $\mathcal{P}$  is *bilateral* if the following holds: For any word  $W$  and letter  $a$  such that  $a \in \mathcal{P}(W)$ , let  $I = \{t, t + 1, \dots, u\}$  be the block of  $\mathcal{P}(W)$  such that  $a \in I$ , one has  $\text{ls}_{\mathcal{P}}(Wa) \in \{t - 1, u + 1\}$ .

Thus to define a bilateral procedure  $\mathcal{P}$ , it suffices to determine a rule that picks either the spot *left* of the block ( $\text{ls}_{\mathcal{P}}(Wa) = t - 1$ ) or *right* ( $\text{ls}(Wa) = u + 1$ ) whenever  $a \in \mathcal{P}(W)$ . We define the direction  $\text{Dir}_{\mathcal{P}}$  by  $\text{Dir}_{\mathcal{P}}(Wa) \in \{\text{left}, \text{right}\}$  accordingly, and will more conveniently define bilateral procedures by specifying  $\text{Dir}_{\mathcal{P}}$ . For memoryless procedures, we will naturally abuse notation and write  $\text{Dir}_{\mathcal{P}}(S, a)$  for  $\text{Dir}_{\mathcal{P}}(Wa)$  when  $S = \mathcal{P}(W)$ .

**Example 2.7.** Let  $k \in \mathbb{Z}_{>0}$ . The *k-Naples parking procedure*  $\mathcal{P}_+^{Nap,k}$  is introduced in [4] and further studied in [5]. The procedure is defined on positive integers, i.e. goes from  $(\mathbb{Z}_{>0})^*$  to  $\text{Fin}(\mathbb{Z}_{>0})$ . It is a variation of the classical parking procedure where one allows cars to back up, up to  $k$  spots, to find an available spot.

We give an extension  $\mathcal{P}^{Nap,k}$  to  $\mathbb{Z}^*$  here –note that in order to define parking functions, only the original procedure plays a role, so our notion coincides with the one from [5],[4]. When a driver finds its preferred spot  $a$  occupied, it goes to the next available spot to the right if  $a \leq 0$ . If  $a > 0$ , it checks spots  $a - 1, a - 2, \dots, a - k$  in order and parks to the first available one, say  $a - j$ , if it exists and satisfies  $a - j > 0$ ; otherwise it takes the first available spot to the right.

Using our notations,  $\mathcal{P}^{Nap,k}$  is the memoryless procedure characterized as follows: for  $S \in \text{Fin}(\mathbb{Z})$  and  $a \in S$ , let  $I = \{t, t + 1, \dots, u\}$  be the block that contains  $a$ . Then  $\text{Dir}(S, a) = \text{left}$  if  $t > 1$  and  $a - t < k$ , and  $\text{Dir}(S, a) = \text{right}$  otherwise.

### 3. LOCAL PARKING PROCEDURES

**3.1. Shift invariance and local decision.** We now come to two extra constraints on a procedure  $\mathcal{P}$ . Informally, we require it to be invariant under translation, and that left/right decisions must depend only on the subsequence of cars that parked on the block.

**3.1.1. Shift invariance.** Let  $\tau : i \mapsto i + 1$  be the shift on  $\mathbb{Z}$ . It extends to subsets of  $\mathbb{Z}$  or words in  $\mathbb{Z}^*$  naturally: for instance  $\tau(\{2, 3, 5\}) = \{3, 4, 6\}$  and  $\tau(523) = 634$ .

**Definition 3.1** (*Shift invariance*). A bilateral parking procedure  $\mathcal{P}$  is said to be *shift invariant* if for any word  $W$ , one has  $\text{Dir}_{\mathcal{P}}(\tau(W)) = \text{Dir}_{\mathcal{P}}(W)$ .

This condition can be equivalently written as  $\mathcal{P}(\tau(W)) = \tau(\mathcal{P}(W))$ , which allows this definition to be extended to non necessarily bilateral procedures.

**Example 3.2.** The procedures  $\mathcal{P}^{\text{right}}$ ,  $\mathcal{P}^{\text{prime}}$ ,  $\mathcal{P}^{\text{closest}}$ ,  $\mathcal{P}^{\text{LBS}}$  are all shift invariant. The  $k$ -Naples procedure  $\mathcal{P}^{Nap,k}$  for  $k \geq 1$  is not shift-invariant since it sends both 11 and 22 to  $\{1, 2\}$ .

Another simple case of a procedure that is not invariant is the (bilateral, memoryless) procedure  $\mathcal{P}^{\text{evenodd}}$  defined as follows: if the desired spot  $a_i$  is occupied, park right if  $a_i$  is even, and left if  $a_i$  is odd.

3.1.2. *Local decision.* If  $I$  is any subset of  $\mathcal{P}(W)$ , let  $W|_I$  be the subword of  $W$  given by the letters  $a_i$  such that the  $i$ th car parked in  $S$ . We will use this in the case where  $I$  is a block of  $S$ .

**Definition 3.3** (*Local decision*). A parking procedure is called locally decided if for any  $W, a$  and  $I$  a block of  $\mathcal{P}(W)$ , we have that  $\text{ls}_{\mathcal{P}}(Wa) = \text{ls}_{\mathcal{P}}(W|_I)$ .

All examples of procedures defined until now are locally decided. It is easy to define one that is not, and we will see such a procedure in Section 3.3.

3.2. **Local procedures.** We now combine the two previous properties to obtain a class of particular interest.

**Definition 3.4** (*Local parking procedures*). A parking procedure  $\mathcal{P}$  is *local* if it is both shift-invariant and locally decided.

As we shall see in the next section, this is a class with a striking enumerative property in the bilateral case. Let us give two immediate properties of the definition. First, the next property shows that in order to study words resulting in a fixed set of parking spots  $S$ , it suffices to know parking functions.

**Proposition 3.5.** *Let  $\mathcal{P}$  be a bilateral, local procedure. Let  $S \in \text{Fin}(\mathbb{Z})$  of size  $r$ . We denote its blocks  $I_1, \dots, I_m$  with  $I_j = \{t_j + 1, \dots, t_j + r_j\}$  for any  $j$ . Then a word  $W$  satisfies  $\mathcal{P}(W) = S$  if and only if it is a shuffle of words  $W_1, \dots, W_m$  where  $\tau^{-t_j}W_j$  is  $\mathcal{P}$ -parking of length  $r_j$ .*

*Thus the total number of such words  $W$  is*

$$\#\{W \mid \mathcal{P}(W) = S\} = \binom{r}{r_1, \dots, r_m} \#\text{Park}_{r_1}(\mathcal{P}) \#\text{Park}_{r_2}(\mathcal{P}) \cdots \#\text{Park}_{r_m}(\mathcal{P}).$$

As  $\text{Park}_r(\mathcal{P}) = (r + 1)^{r-1}$  by Theorem 1.1, proved in the next section, we have an explicit formula.

The second property is that in the *memoryless* case, the class of local procedures is particularly nice to describe.

**Proposition 3.6.** *A local, memoryless, bilateral parking procedure  $\mathcal{P}$  is determined by the data of  $m_{\mathcal{P}}(r, i) := M_{\mathcal{P}}(\{1, \dots, r\}, i) \in \{0, r + 1\}$  for all  $1 \leq i \leq r$ , or equivalently of the directions  $\text{Dir}_{\mathcal{P}}(r, i) := \text{Dir}_{\mathcal{P}}(\{1, \dots, r\}, i)$ .*

*Conversely, any choice of boolean values  $D(r, i) \in \{\text{left}, \text{right}\}$  for all  $1 \leq i \leq r$  determines a local, memoryless, bilateral parking procedure  $\mathcal{P}$  such that  $\text{Dir}_{\mathcal{P}} = D$ .*

The proof follows readily from the definitions. Here are the explicit values  $\text{Dir}_{\mathcal{P}}(r, i)$  in our running examples:

- For the usual procedure  $\mathcal{P}^{\text{right}}$ , we have  $\text{Dir}(r, i) = \text{right}$  for any  $r, i$ .
- For  $\mathcal{P}^{\text{closest}}$ , we have  $\text{Dir}(r, i) = \text{left}$  if  $i \leq r/2$  and  $\text{Dir}(r, i) = \text{right}$  if  $i > r/2$ .
- For  $\mathcal{P}^{\text{prime}}$ , we have  $\text{Dir}(r, i) = \text{right}$  if  $r$  is prime and  $\text{Dir}(r, i) = \text{left}$  if  $r$  is composite.

3.3. **Enumeration.** We will now prove Theorem 1.1, which says that for any bilateral local procedure  $\mathcal{P}$ , the number of  $\mathcal{P}$ -parking functions of length  $r$  is given by  $(r + 1)^{r-1}$ ; the first values for  $r = 1, 2, 3, 4$  are 1, 3, 16, 125. Before we give the proof of this result, let us note that neither condition in the definition of a local procedure (Shift invariance and local decision) can be removed as a hypothesis:

*Remark 3.7.* The procedures  $\mathcal{P}^{evenodd}$  and  $\mathcal{P}^{Nap,k}$  (for  $k > 0$ ) are locally decided, but not shift invariant.  $\mathcal{P}^{evenodd}$  has only 2 parking functions of length 2, while  $\mathcal{P}^{Nap,k}$  has 4, so the result of the theorem does not hold for either of the procedures.

*Remark 3.8.* Now consider the following memoryless procedure  $\mathcal{P}^{far}$ : if one's preferred spot  $a$  is occupied, let  $R$ , *resp.*  $L$ , be the number of cars already parked to the right, *resp.* to the left, of  $a$ . Then one parks in the nearest spot available to the right if  $R \leq L$ , and to the left if  $R > L$ . In our mathematical language: let  $a, W$  such that  $a \in S := \mathcal{P}(W)$ , and  $I = \{s, s+1, \dots, t\}$  the block containing  $a$ . Define  $R = \#\{i \in S \mid i > a\}$  and  $L = \#\{i \in S \mid i < a\}$ . Then  $\text{Dir}_{\mathcal{P}} = \textit{left}$  if  $L \geq R$  while  $\text{Dir}_{\mathcal{P}} = \textit{right}$  if  $L < R$ . It is shift invariant, but not locally decided, and direct enumeration shows that there are only 14 parking functions of length 3 for  $\mathcal{P}^{far}$ , so the conclusion of Theorem 1.1 does not hold.

The proof is based on an argument of Pollak in the classical case, as found in [14],[8]. Let  $\mathbb{Z}_{[r+1]}^i$  be the set of words of length  $i \leq r$  and with letters in  $\{1, \dots, r+1\}$ . By identifying  $\{1, \dots, r+1\}$  with  $\mathbb{Z}/(r+1)\mathbb{Z}$ , the cyclic group  $\mathbb{Z}/(r+1)\mathbb{Z}$  then acts on  $\{1, \dots, r+1\}$  by rotation and thus  $\mathbb{Z}_{[r+1]}^i$  by acting on each letter. Explicitly, let  $\rho = \rho_r$  be defined as

$$\rho : \mathbb{Z}/(r+1)\mathbb{Z} \rightarrow \mathbb{Z}/(r+1)\mathbb{Z}, \quad i \mapsto i+1.$$

which we extend naturally to words by  $\rho(a_1 \cdots a_i) = \rho(a_1) \cdots \rho(a_i)$ . Thus each orbit of words has the form  $\{W, \rho(W), \dots, \rho^r(W)\}$ ; we call such orbits ‘‘cyclic’’. Equivalently, they correspond to the cosets of  $(1, \dots, 1)$  in  $\mathbb{Z}_{[r+1]}^i \simeq (\mathbb{Z}/(r+1)\mathbb{Z})^i$  considered as an additive group.

**Proposition 3.9** (Parking cyclic lemma). *Let  $\mathcal{P}$  be a bilateral, local procedure, and  $r \geq 1$ . There is exactly one  $\mathcal{P}$ -parking function in each cyclic orbit in  $\mathbb{Z}_{[r+1]}^r$ .*

*Proof of Theorem 1.1.* It is clear that  $\text{Park}_r \subset \mathbb{Z}_{[r+1]}^r$ , the latter being of cardinal  $(r+1)^r$ . Proposition 3.9 then tells us that a fraction  $1/(r+1)$  of these words are parking functions, which proves the desired enumeration.  $\square$

For the proof of Proposition 3.9, we will introduce a cyclic version of the procedure  $\mathcal{P}$ . By a *cyclic parking procedure*, we mean a procedure with domain  $\mathbb{Z}_{[r+1]}^{\leq r}$  and codomain  $\text{Fin}(\{1, \dots, r+1\})$  that satisfies the properties of Definition 2.1. Cyclic intervals are well-defined as sets of consecutive integers, so we can also define cyclic blocks as the maximal such intervals in a set  $S$ . Bilateral cyclic parking procedures thus make sense, and can be specified by a direction function  $\text{Dir}$ .

*Proof of Proposition 3.9.* We define the cyclic procedure  $\mathcal{P}_r$  by induction, by specifying its direction function  $\text{Dir}_r$ . Suppose that it is defined on all words of length up to  $i < r$ , and let  $W$  have length  $i$ . Let  $S := \mathcal{P}_r(W)$ ,  $a \in S$ , and let  $I$  be the cyclic block of  $S$  containing  $a$  and  $V = W|_I$ . By applying a suitable rotation  $\rho^k$ , we have  $\rho^k(I) = \{1, \dots, j\}$  for a certain  $j$ .

We then define  $\text{Dir}_r(Wa) := \text{Dir}(\rho^k(Wa))$ . Let us remark that at this point, the local property has not been used, and the definition makes thus sense for any procedure. Note that  $\mathcal{P}$  being local implies that this is also equal to  $\text{Dir}(\rho^k(Va))$ . Now we claim that:

- (1) For any  $W$ ,  $\mathcal{P}_r(\rho(W)) = \rho(\mathcal{P}_r(W))$ .
- (2)  $W$  is a  $\mathcal{P}$ -parking word if and only if  $\mathcal{P}_r(W) = \{1, \dots, r\}$ .



The two claims together imply the proposition: the first claim implies that each orbit contains exactly one word  $W$  with  $\mathcal{P}_r(W) = \{1, \dots, r\}$ , while the second one claims that these are precisely the  $\mathcal{P}$ -parking words. Let us now prove these claims.

The first claim follows readily from our definition of  $\mathcal{P}_r$ : keeping the notations from the above paragraph, one needs to show that  $\text{Dir}_r(Wa) = \text{Dir}_r(\rho(Wa))$  for any  $a, W$  with  $a \in \mathcal{P}_r(W)$ , and indeed both are given by  $\text{Dir}(\rho^k(Wa)) = \text{Dir}(\rho^{k-1}\rho(Wa))$ .

For the second claim, note that  $r+1$  is missing from  $\mathcal{P}_r(W)$  at the end if and only if it was missing at each step. If  $W'$  is a  $\mathcal{P}$ -parking word, let us show that  $\mathcal{P}(W) = \mathcal{P}_r(W)$  for each prefix. We need to show that  $\text{Dir}(Wa) = \text{Dir}_r(Wa)$  for each prefix  $Wa$  of  $W'$  with  $a \in \mathcal{P}(W)$ . Since  $\mathcal{P}$  is local,  $\text{Dir}(Wa) = \text{Dir}(Va)$  while  $\text{Dir}_r(Wa) = \text{Dir}(\rho^k(Va))$ . Now remark that we have  $\rho^k(Va) = \tau^j(Va)$  for a certain  $j$  as words in  $\{1, \dots, r\}$ , and we can conclude by shift-invariance.  $\square$

This gives a better understanding of why the procedure  $\mathcal{P}^{far}$  fails to satisfy the result of Proposition 3.9. Two cosets have no parking function in them, namely  $\{131, 242, 313, 424\}$  and  $\{133, 244, 311, 422\}$ , the other fourteen having one.

**3.4. Extending Theorem 1.1 and Proposition 3.9.** Following up on the previous remark, the problem is essentially that the parking decisions in  $\mathcal{P}^{far}$  depend on information on the position (left or right) of cars that are outside of the block where one wants to park. Such information is not stable under a cyclic rotation of the blocks, which is why the cyclic version  $\mathcal{P}_r^{far}$  used in the proof of Proposition 3.9 behaves differently from the original procedure..

For a bilateral, shift-invariant procedure, the local decision property removes this issue. Since it is also easy to state and has some nice structural properties (see Propositions 3.5,3.6), we chose to state the theorem at this level of generality. But there are weaker properties that also work, the key being to be able to define the cyclic version. Analyzing the proof of Proposition 3.9, one can weaken the hypothesis as follows.

**Proposition 3.10.** *Let  $\mathcal{P}$  be a bilateral procedure that satisfies the following property:*

*For any  $r \geq 1$ , any word  $W$  of length at most  $r$  with letters in  $\{1, \dots, r\}$ , and any cyclic rotation  $R = \rho_r^k$  such that  $\mathcal{P}(R(W))$  does not contain  $r+1$ , then  $\text{Dir}(Wa) = \text{Dir}(R(Wa))$  for any  $a \in \mathcal{P}(W)$ . Then there is exactly one  $\mathcal{P}$ -parking function in each cyclic orbit in  $\mathbb{Z}_{[r+1]}^r$ , and thus there are  $(r+1)^{r-1}$   $\mathcal{P}$ -parking words of length  $r$ .*

In particular, note that the property is satisfied if the parking rule is a function of the total number of parked cars –equivalently, if the parking rule for the  $i$ th car depends on  $i$ . For this class of procedures, the previous result is in fact hinted at in the seminal article [10], as we will pinpoint in the next section.

#### 4. PROBABILISTIC PARKING

To add probabilities to the setting, the usual manner is to study properties of  $\mathcal{P}$ -parking words picked uniformly at random for instance. In the classical case, this is a very natural problem from the hashing viewpoint, and the problem and was studied extensively: see [6, 7]. One can also probabilize the procedures themselves, which is the point of view we develop here.

**4.1. Definition.** A probabilistic procedure is defined at the end of [10]: Given  $q \in [0, 1]$ , consider the following procedure  $\mathcal{P}^{KW,q}$ : when one's spot is occupied, park at the nearest spot on the right (resp. left) with probability  $q$  (resp.  $1 - q$ ). A more generalized procedure  $\mathcal{P}^{KW,(q)_i}$  is in fact defined at the very end of the paper: if the  $i$ th driver's spot is occupied then the probabilities are  $q_i$  and  $1 - q_i$  for a fixed family  $\mathbf{q} = (q_i)_{i \geq 1}$ . Notice that if each  $q_i$  is in  $\{0, 1\}$ , we get a class of deterministic procedure that satisfies the hypothesis of of Proposition 3.10.

We can then develop the notion of a bilateral probabilistic parking procedure  $\mathcal{P}$  in much the same way as we did in the deterministic case. In that case  $\mathcal{P}(W)$  will be the data of a finitely supported "probability measure" on  $\text{Fin}(\mathbb{Z})$ ; we write it  $\mathcal{P}_W$ . By this we will just mean a function on  $\text{Fin}(\mathbb{Z})$  with values in  $[0, 1]$  such that  $\mathcal{P}_W$  is zero outside of a finite number of subsets, and the sum of  $\mathcal{P}_W(S)$  over  $S \in \text{Fin}(\mathbb{Z})$  is 1.

To define a bilateral probabilistic procedure  $\mathcal{P}$ , we fix  $\text{Dir}_{\mathcal{P}}(W, S, a)$  a real number in  $[0, 1]$ , representing the probability to go right when  $a \in S$  after having read  $\mathcal{P}_W$ . This is defined when  $W$  has  $\#S$  letters, all of them belonging to  $S$ . Let us define a graph  $G(\mathcal{P})$  with vertices all such pairs  $(W, S)$ , with outgoing edges at each vertex labeled by  $\mathbb{Z}$ . Then we have the following edges for any  $W, S, a$ :

- if  $a \notin S$ ,  $(W, S) \xrightarrow{a} (Wa, S \sqcup \{a\})$
- if  $a \in S$ , let  $I = \{s, \dots, t\}$  the block that contains  $a$ . Then we have an edge  $(W, S) \xrightarrow{a} (Wa, S \cup \{t+1\})$  with weight  $\text{Dir}_{\mathcal{P}}(W, S, a)$  and an edge  $(W, S) \xrightarrow{a} (Wa, S \cup \{s-1\})$  with weight  $1 - \text{Dir}_{\mathcal{P}}(W, S, a)$ .

The value  $\mathcal{P}_W$  is then the total weight of all paths in  $G(\mathcal{P})$  going from  $(\epsilon, \emptyset)$  to  $(W, S)$ . It is clear by induction that the sum of  $\mathcal{P}_W(S)$  over  $S \in \text{Fin}(\mathbb{Z})$  is 1, which justifies thinking about  $\mathcal{P}_W$  as a probability measure.

**Definition 4.1** (Parking probability). Let  $\mathcal{P}$  be a probabilistic parking procedure, and  $W$  a word of length  $r$ . Then the *parking probability*  $\text{Park}_{\mathcal{P}}(W)$  is defined by  $\mathcal{P}_W(\{1, \dots, r\})$ .

The notion of a memoryless procedure extends directly: here we need just fix real numbers  $\text{Dir}_{\mathcal{P}}(S, a)$  with no dependence on  $W$ , and the graph  $G(\mathcal{P})$  has correspondingly vertices indexed simply by  $\text{Fin}(S)$  and its edges are simply defined by dropping  $W$  from the definitions above. Finally  $\mathcal{P}_W(S)$  is the total weight of all paths in  $G(\mathcal{P})$  from  $\emptyset$  to  $S$  and labeled by the word  $W$ .

**4.2. Enumerative properties.** The notion of local procedure extends readily to the probabilistic setting (DETAIL), and we have the following extension of Theorem 1.1.

**Theorem 4.2.** *Let  $\mathcal{P}$  be a local, probabilistic parking procedure, and  $r \geq 1$ . Then the sum of  $\text{Park}_{\mathcal{P}}(W)$  over all words  $W$  of length  $r$  is  $(r+1)^{r-1}$ .*

This was noticed to hold for the procedure  $\mathcal{P}^{KW,q}$  in [10]. The key to the theorem is the following extension of Proposition 3.9:

**Proposition 4.3.** *Let  $\mathcal{P}$  be a local, probabilistic parking procedure, and  $r \geq 1$ . Then the sum of  $\text{Park}_{\mathcal{P}}(W)$  over any cyclic orbit in  $\mathbb{Z}_{[r+1]}^r$  is equal to 1. Thus the sum of  $\text{Park}_{\mathcal{P}}(W)$  over all words of length  $r$  is  $(r+1)^{r-1}$ .*

The local hypothesis can be weakened to the one in Proposition 3.10. In particular, the result applies to the procedures  $\mathcal{P}^{KW,(q)_i}$ , and thus we recover as a special case the result mentioned at the very end of [10].

**4.3. Abelian procedures.** A local, memoryless procedure  $\mathcal{P}$  is completely specified by the numbers  $p_{\mathcal{P}}(r, i) = \text{Dir}_{\mathcal{P}}(r, i) \in [0, 1]$  giving the probability to go right when a car with preferred spot  $ih$  wants to park on the block  $\{1, \dots, r\}$ .

**Example 4.4.** For the procedure  $\mathcal{P}^{KW,q}$ , we have  $p(r, i) = q$  for all  $r, i$ .

**Example 4.5.** Let us define the procedure  $\mathcal{P}^{P,q}$ , with  $q \in [0, +\infty]$ , by  $p(r, i)$  are given by  $\frac{[i]}{[r+1]}$ . This is the probability that the car parks in  $r + 1$  after a biased random walk on  $\{1, \dots, r\}$  with right probability  $1/(1 + q)$ . This leads to the rich combinatorics of *remixed Eulerian numbers*, introduced by the author with Vasu Tewari [11],[12]. The interpretation as a parking procedure is a reformulation of the “sequential process” given in [12, Section2].

**Definition 4.6.** A (probabilistic) parking procedure  $\mathcal{P}$  is called *abelian* if  $\mathcal{P}_{a_1 \dots a_r} = \mathcal{P}_{a_{\sigma_1} \dots a_{\sigma_r}}$  for any letters  $a_1, \dots, a_r$  and any permutation  $\sigma$ .

$\mathcal{P}^{right}$  is known to be abelian, as can be seen immediately from the characterization of parking words in the introduction. More generally, the procedures  $\mathcal{P}^{P,q}$  are known to be abelian, cf. [12].

ADD SOMETHING HERE

**Proposition 4.7.** *The procedures  $\mathcal{P}^q$ ,  $q \in [0, \infty]$ , are the only local memoryless procedures that are abelian.*

In the deterministic case, it follows that a local memoryless procedure  $\mathcal{P}$  is abelian if and only if it is either  $\mathcal{P}^{right}$  or its symmetric version  $\mathcal{P}^{left}$ .

*Proof.* Let  $\mathcal{P}$  be a local memoryless abelian procedure, and define  $p_{r,i} := p_{\mathcal{P}}(r, i)$ . Define  $q$  by  $p_{1,1} = \frac{1}{1+q}$ . We need to show that  $p_{r,i} = \frac{[i]}{[r+1]}$  for all  $r, i$ . We prove this by induction on  $r$ . It holds for  $r = 1$  by definition, and we assume the claim holds for  $r - 1$  with  $r \geq 2$ .

Consider the word  $W_{r,i} = 12 \dots (r - 1)ri$  for any  $1 \leq i \leq r$ . By definition its parking probability is  $p_{r,i}$ . Now, for  $i \neq r$ , consider the word  $12 \dots (r - 1)ir$ , obtained by exchanging the last two letters in  $W_{r,i}$ . Its parking probability is given by  $p_{r-1,i}p_{r,r}$ , so by abelianity and the induction hypothesis we obtain:

$$(4.1) \quad p_{r,i} = \frac{[i]}{[r]} p_{r,r} \text{ for } i = 1, \dots, r - 1.$$

Consider now the word  $12 \dots (r - 2)rr(r - 1)$ , obtained from  $W_{r,r}$  by moving  $n - 1$  to the end. Its parking probability is easily computed as  $\frac{1}{1+q} + \frac{q}{1+q} p_{r,r-1}$ , the two terms corresponding to the second car with desired spot  $r$  going right or left. By abelianity.

$$(4.2) \quad p_{r,r} = \frac{1}{1+q} + \frac{q}{1+q} p_{r,r-1}.$$

One can now easily solve the system of equations (4.1),(4.2) and get the desired result  $p_{r,i} = \frac{[i]}{[r+1]}$  which is thus proven by induction.  $\square$

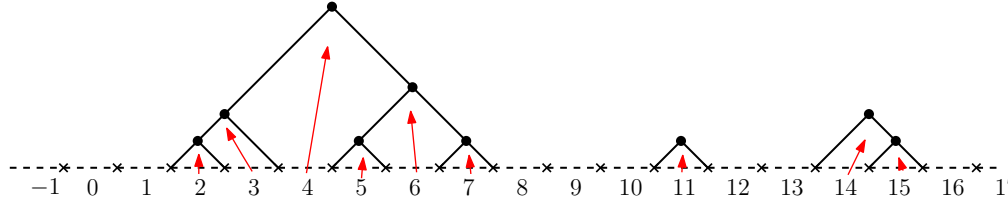
## 5. ENCODING WITH BINARY TREES

In this section we will define a general lift of the parking procedure: given any parking procedure  $\mathcal{P}$ , we define an injective function  $\widehat{\mathcal{P}}$  also defined on  $\mathbb{Z}^*$  with a natural projection  $\Pi$  such that  $\Pi \circ \widehat{\mathcal{P}} = \mathcal{P}$ .

5.1. **Definition of  $\widehat{\mathcal{P}}$ .** Recall that a finite, plane, binary *tree* is defined recursively as either empty or consisting of a node, a left (sub)tree and a right (sub)tree. Its size is its number of nodes. These are in bijection with *complete* binary trees, where by attaching extra leaves. A forest is then usually defined as a set of trees.

**Definition 5.1.** An *indexed forest*  $F$  is the data of a subset  $S$  in  $\mathbb{Z}$ , and a binary tree of size  $|I|$  for each block  $I$  of  $S$ . The set  $S$  is the *support* of  $F$ . Binary trees of size  $r$  are identified with indexed forests with support  $\{1, \dots, r\}$ .

In the example below, the support is  $\{2, 3, 4, 5, 6, 7, 11, 14, 15\}$ . Elements of the support correspond bijectively to nodes of the forest, as illustrated by the arrows in the figure: this is the *canonical labeling* of the nodes that will be important in what follows, which we will use to specify nodes.



Given a procedure  $\mathcal{P}$  and a word  $W$  of length  $r$ , we will attach a pair  $(P, Q)$  of labeled forests with the same underlying indexed forest of size  $r$ . The projection  $\Pi(P, Q)$  is defined as the support of this indexed forest  $F$ . The labels are on the nodes of the common shape, and we will write  $p_i$ , *resp.*  $q_i$  for the label in  $P$ , *resp.*  $Q$  of the node  $i \in \Pi(P, Q)$ .  $P$  will be bijectively labeled by the multiset of letters of  $W$ , while  $Q$  is a *decreasing forest*: it has labels  $\{1, \dots, r\}$  and each node has greater label than all its descendants.

Let us first define the forest  $Q$ . Given  $\mathcal{P}, W$ , consider the bijection  $\pi_{\mathcal{P}}(W)$  defined in Section 2, which encodes the order in which parking spots were filled. Let  $S = \mathcal{P}(W)$ . Then one can naturally encode  $\pi_{\mathcal{P}}(W)$  as an indexed decreasing forest with support  $S$ . This defines the forest  $Q$ , and its underlying forest  $F$ . We note  $i \rightarrow q_i$  this labeling. To construct  $P$ , simply label  $F$  by having node  $i$  get the label  $p_i = W_{q_i}$ .

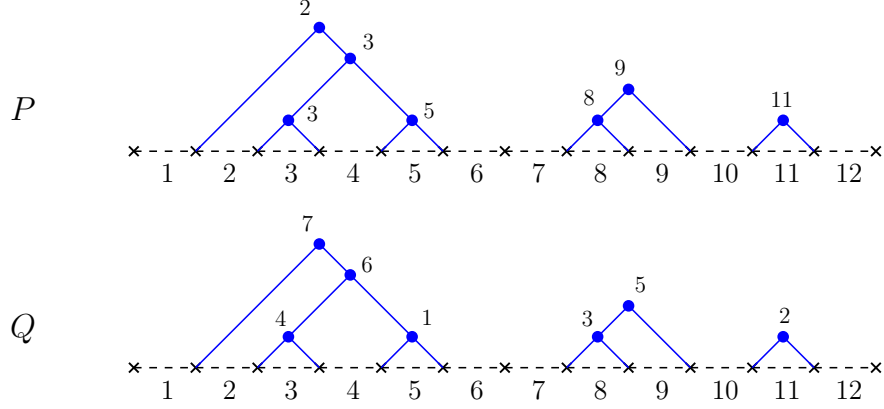
**Definition 5.2.** We define  $\widehat{\mathcal{P}} : \mathbb{Z}^* \mapsto (P, Q)$  to be this construction.

Here is this correspondence for  $\mathcal{P}^{LBS}$  with the word  $W = 5.11.8.3.9.3.2$  from our previous example:

**Proposition 5.3.** *Let us list some immediate properties for any local procedure  $\mathcal{P}$ :*

- *A word  $W$  is  $\mathcal{P}$ -parking if the common shape of  $(P, Q)$  is a tree.*
- *The  $\mathcal{P}$ -correspondence is injective.*
- *The canonical labeling of a node is the spot where the corresponding car ended up parking.*

Following up on this last point, one sees that  $\sum_i |p_i - i|$  corresponds to the total *displacement* of the parking process.



**5.2. Correspondence.** The framework is particularly nice if for any  $(P, Q)$  in the image of  $\widehat{\mathcal{P}}$ , all  $(P, Q')$  with  $Q'$  a decreasing forest of the same shape is also in the image of  $\widehat{\mathcal{P}}$ . In that case the image is determined once we know what labeled forests  $P$  can occur: we say that  $\widehat{\mathcal{P}}$  is a *correspondence*, and that such admissible labeled forests are  *$\mathcal{P}$ -forest*.

In general, local procedures do not give rise to correspondences. It holds however in the memoryless case, where in fact one need only assume that the procedure is locally decided (see Definition 3.3):

**Proposition 5.4.** *Let  $\mathcal{P}$  be a locally decided memoryless procedure. Then  $\widehat{\mathcal{P}}$  is a correspondence with  $\mathcal{P}$ -trees given by the following condition: if  $i$  is any node with interval  $[l_i, r_i]$ , then the possible values of its label from the following set :*

$$(5.1) \quad \{j \mid \text{Dir}_{\mathcal{P}}(\{l_i, \dots, i-1\}, j) = \text{right}\} \cup \{i\} \cup \{j \mid \text{Dir}_{\mathcal{P}}(\{i+1, \dots, r_i\}, j) = \text{left}\}.$$

*Proof.* Suppose  $W \mapsto (P, Q)$  Exchange  $q_i = k$  and  $q_j = k+1$  in  $Q$  when  $i, j$  are incomparable, let  $Q'$  be the result. Then  $W' \mapsto Q'$  where  $W'$  is obtained from  $W$  by swapping letters at positions  $k, k+1$ . Indeed this is true for up to position  $k$  because it's locally decided, and then the memoryless condition takes care of the rest. The rest of the statement is easy and left to the reader.  $\square$

Now to determine the fibers of the outcome function  $\pi_{\mathcal{P}}$ , one has to simply count the number of  $\mathcal{P}$ -trees of a given shape. We thus get immediately from the previous proposition:

**Proposition 5.5.** *Let  $\mathcal{P}$  be a locally decided memoryless procedure, and  $\sigma$  a permutation of  $\{1, \dots, r\}$ . Let  $T$  be the binary tree underlying the decreasing tree of  $\sigma$ .*

*Then the number of  $\mathcal{P}$ -parking functions with outcome  $\sigma$  is given by the product over all  $i$  of the cardinality of the sets (5.1). If  $\mathcal{P}$  is moreover shift-invariant (thus a local procedure), then we have explicitly*

$$(5.2) \quad \prod_{i \in T} (1 + R_{i-l_i} + L_{r_i-i}),$$

where  $R_k$  is the number of values  $i \in \{1, \dots, k\}$  such that  $\text{Dir}_{\mathcal{P}}(r, i) = \text{right}$ , and  $L_k = k - R_k$ .

The  $\mathcal{P}^{\text{right}}$ -forests for the classical procedure  $\mathcal{P}^{\text{right}}$  have labels in  $[l_i, i]$  for any node  $i$ . The specialization of Proposition 5.5 then gives the well-known result given in [15, Exercise 5.49(d)].

**Example 5.6.** For the  $k$ -Naples procedure  $\mathcal{P}^{Nap,k}$  the set (5.1) is given by

$$\{l_i + k - 1, l_i + k, \dots, i - 1\} \cup \{i\} \cup \{i + 1, \dots, i + k\}$$

if  $l_i > 1$ , and by

$$\{1, \dots, i - 1\} \cup \{i\} \cup \{i + 1, \dots, i + k\}$$

if  $l_i = 1$ .

We thus get the main result of [5], Theorem 3.12. Note that  $l_i = 1$  means that node  $i$  lies on the leftmost branch of the tree  $T$ . It is then immediate to get the recursive formula for  $\#\text{Park}_r(\mathcal{P}^{Nap,k})$  given in [4].

*Remark 5.7.* We thus get a bijection  $\Phi = \widehat{\mathcal{P}}^{right}$  between classical parking functions and binary trees with a double labeling. Parking functions are also bijectively connected to families of labeled binary trees in (at least) two other ways: first, one uses the standard representation of parking functions as labeled Dyck paths, composed with any bijection between Dyck paths and binary trees. This was done recently for example in [1, Section 6]. Another known bijection is with *Shi trees*, as defined by Gessel, see for instance [2] and references therein.

These bijections are however not naturally related to our correspondence  $\Phi$ , in the sense that the number of labelings of a given tree differ for  $r = 3$  in all three cases: If we write the number of labelings in a nonincreasing fashion for all 5 trees of these sizes, we get 6, 4, 3, 2, 1 in our correspondence, 6, 3, 3, 3, 1 using any of the other bijections.

Using standard bijection between binary and plane trees,  $\Phi$  can be seen to be equivalent to a bijection between parking functions and Cayley trees due to Knuth, as described by Yan [16].

**5.3. The probabilistic case.** (Rough sketch) Each path in the graph  $G(\mathcal{P})$  naturally gives rise to a pair  $(P, Q)$  by using the above construction, carrying the weight of the path. If we group all pairs  $(P, Q)$  ending in  $(W, S)$ , we get an interpretation for  $\mathcal{P}_W(S)$ . In the case of  $\mathcal{P}^{NT,q}$ , this gives rise naturally to the interpretation of  $A_c(q)$  as counting *Postnikov trees* [12].

## 6. COLORED VERSION

Let  $C$  be any set, and consider the alphabet  $A = \mathbb{Z} \times C$ . The set  $C$  represents some extra information. In terms of cars, one might consider its brand, its color, or the age of the driver. Let  $\text{val} : A = \mathbb{Z} \times S \rightarrow \mathbb{Z}$  be the projection to the first factor, which represents the preferred spot. Now a preference list, will be a word  $a_1 \dots a_r$  in  $A^*$ , the  $i$ th car having preferred spot  $\text{val} a_i$ . The set  $C$  can then be used as extra source of information in order to decide where to park.

We have as before (bilateral) parking procedures,

$$\mathcal{P} : A^* \rightarrow \text{Fin}(\mathbb{Z})$$

defined exactly as in Section 2: one simply has to replace  $\mathbb{Z}^*$  by  $A^*$  and the conditions  $a \in \mathcal{P}(W)$  by  $\text{val}(a) \in \mathcal{P}(W)$ .

If  $a = (i, s)$ , extend the shift  $\tau$  by defining  $\tau(a) = (i + 1, s)$  and extend to words. Then the shift-invariance and local decision property are immediately extended to procedures on  $A^*$ , so we have the notion of local procedures in this case.

One can require  $\mathcal{P}$  to be only a *partial* function, that is, to be defined on a subset  $L \subset A^*$ . It is reasonable to require that  $L$  be closed under deleting a letter at any position, that is, that  $L$  be

closed under taking subwords. This ensures that the words  $W|_I$  are well-defined in the definition of a local procedure.

Also,  $L$  should be closed under (cyclic) shifts: if  $W$  is a word of  $L$  of length  $i < r$ , the word  $\rho_r(W)$  is also in  $L$ . We can now state the extension of Proposition 3.9.

**Proposition 6.1.** *Let  $L \subseteq A^*$  be closed under subwords and cyclic shifts. Let  $\mathcal{P}$  be a colored local procedure defined on  $L$ . Then any cyclic class in  $L \cap A_{r+1}^r$  contains exactly one parking word.*

We can extend  $\mathcal{P}^{LBS}$  to this colored setting. This correspondence is introduced and studied in [13]. We pick  $C = \mathbb{Z}_{>0}$ , and only consider the language  $L$  of words in  $A$  with *distinct letters*. Order  $A = \mathbb{Z} \times S$  lexicographically. Let  $W, a, I$  such that  $I$  is a block of  $\mathcal{P}^{LBS}(W)$  and  $a \in I$ . Let  $j \in \{1, \dots, r\}$  be maximal such that  $a_j \in I$  (“last car that parked on the interval”). Then if  $a < a_j$  the car parks left, and if  $a > a_j$  park right. Since we only consider words with distinct letters, this is well-defined.

#### ACKNOWLEDGEMENTS

I would like to thank everyone I’ve bothered with this in the past couple of years.

#### REFERENCES

- [1] N. Bergeron, R. S. González D’León, S. X. Li, C. Y. A. Pang, and Y. Vargas. Hopf algebras of parking functions and decorated planar trees. *Adv. in Appl. Math.*, 143:Paper No. 102436, 62, 2023.
- [2] O. Bernardi. Deformations of the braid arrangement and trees. *Adv. Math.*, 335:466–518, 2018.
- [3] J. Carlson, A. Christensen, P. E. Harris, Z. Jones, and A. Ramos Rodríguez. Parking functions: choose your own adventure. *College Math. J.*, 52(4):254–264, 2021.
- [4] A. Christensen, P. E. Harris, Z. Jones, M. Loving, A. Ramos Rodríguez, J. Rennie, and G. R. Kirby. A generalization of parking functions allowing backward movement. *Electron. J. Combin.*, 27(1):Paper No. 1.33, 18, 2020.
- [5] L. Colmenarejo, P. E. Harris, Z. Jones, C. Keller, A. Ramos Rodríguez, E. Sukarto, and A. R. Vindas-Meléndez. Counting  $k$ -Naples parking functions through permutations and the  $k$ -Naples area statistic. *Enumer. Comb. Appl.*, 1(2):Paper No. S2R11, 16, 2021.
- [6] P. Diaconis and A. Hicks. Probabilizing parking functions. *Adv. in Appl. Math.*, 89:125–155, 2017.
- [7] P. Flajolet, P. Poblete, and A. Viola. On the analysis of linear probing hashing. *Algorithmica*, 22(4):490–515, 1998. Average-case analysis of algorithms.
- [8] D. Foata and J. Riordan. Mappings of acyclic and parking functions. *Aequationes Math.*, 10:10–22, 1974.
- [9] P. E. Harris, B. M. Kamau, J. C. M. Mori, and R. Tian. On the outcome map of MVP parking functions: Permutations avoiding 321 and 3412, and Motzkin paths, 2022.
- [10] A.G. Konheim and B. Weiss. An occupancy discipline and applications. *SIAM J. Appl. Math.*, 14:1266–1274, 1966.
- [11] P. Nadeau and V. Tewari. A  $q$ -analogue of an algebra of Klyachko and Macdonald’s reduced word identity, 2021.
- [12] P. Nadeau and V. Tewari. Remixed Eulerian numbers, 2022.
- [13] P. Nadeau and V. Tewari. Forest polynomials and the class of the permutahedral variety, 2023.
- [14] J. Riordan. Ballots and trees. *J. Combinatorial Theory*, 6:408–411, 1969.
- [15] R. P. Stanley. Enumerative combinatorics. Vol. 2. 62:xii+581, 1999. With a foreword by Gian-Carlo Rota and appendix 1 by Sergey Fomin.
- [16] C. H. Yan. Parking functions. In *Handbook of enumerative combinatorics*, Discrete Math. Appl. (Boca Raton), pages 835–893. CRC Press, Boca Raton, FL, 2015.

UNIV LYON, CNRS, UNIVERSITÉ CLAUDE BERNARD LYON 1, UMR 5208, INSTITUT CAMILLE JORDAN, 43 BLVD. DU 11 NOVEMBRE 1918, F-69622 VILLEURBANNE CEDEX, FRANCE

Email address: nadeau@math.univ-lyon1.fr