

# Supplementary Material for “Inference for Bayesian nonparametric models with binary response data via permutation counting”

Dennis Christensen<sup>\*,†</sup>

## Appendix A: Proof of Proposition 2.2

In this section, we prove Proposition 2.2 from the main text, which we restate here as Proposition A.1.

**Proposition A.1.** *Let  $x \in [0, \infty)^n$ . Then  $w(x; \mathcal{B}) > 0$  if and only if  $x$  is  $\mathcal{B}$ -admissible.*

*Proof of Proposition A.1.* Let  $\sigma \in S_n$  be as in Definition 2.1 from the main text. Clearly, if  $x$  is  $\mathcal{B}$ -admissible, then by definition, there exists a permutation in  $S_n$ , namely  $\sigma$ , such that  $\sigma(x) \in \mathcal{B}$ .

We prove the converse by induction on  $n$ . The base case  $n = 1$  is trivial. For the inductive step, assume the proposition holds for  $\dim(\mathcal{B}) < n$  and suppose  $w(x; \mathcal{B}) > 0$ . Then there exists a permutation  $\tau$  such that  $\tau(x) \in \mathcal{B}$ . Let  $j$  be the index such that  $\sigma(1) = \tau(j)$ . We claim that we may assume that  $j = 1$  without loss of generality.

To prove the claim, it suffices to show that  $x_{\tau(1)} \in \mathcal{B}_j$ . Indeed, then we can redefine  $\tau$  to map 1 to  $\tau(j)$  and  $j$  to  $\tau(1)$ . To prove that  $x_{\tau(1)} \in \mathcal{B}_j$ , first assume that  $\mathcal{B}_j$  extends to the left, so  $\mathcal{B}_j = [0, t_j)$ . In this case, we have  $x_{\tau(1)} \in \mathcal{B}_1 \subseteq \mathcal{B}_j$ , so  $x_{\tau(1)} \in \mathcal{B}_j$ . On the other hand, if  $\mathcal{B}_j$  extends to the right, then  $\mathcal{B}_j = [t_j, \infty)$  and  $x_{\tau(1)} \in \mathcal{B}_j$  as  $x_{\tau(j)} \leq x_{\tau(1)}$ . In either case, we have that  $x_{\tau(1)} \in \mathcal{B}_j$ , so we may assume  $j = 1$ .

Now, let  $x_{-1} = (x_2, \dots, x_n)$  and  $\mathcal{B}_{-1} = \mathcal{B}_2 \times \dots \times \mathcal{B}_n$ . As  $\tau(x) \in \mathcal{B}$ , it must also be the case that  $\tau'(x_{-1}) \in \mathcal{B}_{-1}$ , where  $\tau'$  is the restriction of  $\tau$  to the domain  $\{2, \dots, n\}$ . Hence,  $w(x_{-1}; \mathcal{B}_{-1}) > 0$ , and so by induction hypothesis,  $x_{-1}$  is  $\mathcal{B}_{-1}$ -admissible. That is,  $\sigma'(x_{-1}) \in \mathcal{B}_{-1}$ , where  $\sigma'$  is the restriction of  $\sigma$  to the domain  $\{2, \dots, n\}$ . Since we also know that  $x_{\sigma(1)} = x_{\tau(1)} \in \mathcal{B}_1$ , we see that  $x$  is  $\mathcal{B}$ -admissible.  $\square$

## Appendix B: Proof of Proposition 3.1

In this section, we prove Proposition 3.1 from the main text, which we restate here as Proposition B.1.

**Proposition B.1.** *Let  $x$  be  $\mathcal{B}$ -admissible and let  $A$  be the matching matrix of  $x$ . Then, after permuting its columns if necessary,  $A$  is a complete block rectangular matrix.*

---

arXiv: [2010.00000](https://arxiv.org/abs/2010.00000)

<sup>\*</sup>Department of Mathematics, University of Oslo, Oslo, Norway

<sup>†</sup>Norwegian Defence Research Establishment (FFI), Kjeller, Norway [dennis.christensen@ffi.no](mailto:dennis.christensen@ffi.no)

*Proof of Proposition B.1.* If  $A$  contained a row of zeros, then this would mean that for some  $j \in \{1, \dots, n\}$ , we must have  $x_i \notin \mathcal{B}_i$  for all  $i = 1, \dots, n$ . But then  $x$  is not  $\mathcal{B}$ -admissible, a contradiction.

Without loss of generality, we can assume that  $x_1 \leq \dots \leq x_n$ . This corresponds to permuting the columns of  $A$ , or equivalently, to hitting  $x$  with the permutation  $\sigma$  from Definition 2.1 from the main text. By the ordering of the  $\mathcal{B}_i$ , each column  $a_j$  of  $A$  takes the form

$$a_j = (\underbrace{0, \dots, 0}_{k_j^{(1)}}, \underbrace{1, \dots, 1}_{k_j^{(2)}}, \underbrace{0, \dots, 0}_{k_j^{(3)}})^T,$$

where

$$\begin{aligned} k_j^{(1)} &= \#\{i \mid x_j \geq t_i, i \leq n_0\} \\ k_j^{(2)} &= \#\{i \mid x_j < t_i, i \leq n_0\} + \#\{i \mid x_j \geq t_i, i > n_0\} \\ k_j^{(3)} &= \#\{i \mid x_j < t_i, i > n_0\}. \end{aligned}$$

Furthermore, since the  $x_i$  are in ascending order, we see that if  $j \leq j'$ , then  $k_j^{(1)} \leq k_{j'}^{(1)}$  and  $k_j^{(3)} \geq k_{j'}^{(3)}$ . This means that the matrix  $A$  comprises a sequence of descending rectangular blocks of ones, and can thus be described with parameters  $\alpha, \beta, \gamma, m$ , as in Definition 3.2 from the main text. Clearly,  $\alpha, \beta, \gamma, m$  satisfy condition (3.2) from the main text. It remains to show that they also satisfy condition (3.3) from the main text.

Assume for a contradiction that  $\sum_{r=1}^t \beta_r > \sum_{r=1}^t \alpha_r$  for some  $t \in \{1, \dots, k\}$ . Then  $a_{ij} = 0$  for all  $i \leq \sum_{s=1}^t \beta_r$  and  $j > \sum_{s=1}^t \alpha_s$ . That is,  $x_i \notin \mathcal{B}_j$  for all  $i \leq \sum_{s=1}^t \beta_r$  and  $j > \sum_{s=1}^t \alpha_s$ . But then  $x$  is not  $\mathcal{B}$ -admissible, since there exists no surjection from  $\{1, \dots, \sum_{r=1}^t \alpha_r\}$  to  $\{1, \dots, \sum_{r=1}^t \beta_r\}$ . The corresponding condition for  $\gamma$  can be proved similarly.  $\square$

## Appendix C: Proof of Theorem 3.1

In this section, we prove Theorem 3.1 from the main text, which we restate here as Theorem C.1.

**Theorem C.1.** *Let  $A$  be an  $m \times n$  complete block rectangular matrix. Then there exists an implementable algorithm for computing  $\text{perm}(A)$ , whose computational complexity grows polynomially with  $n$ .*

The first step is to introduce the notion of *subpermanents*.

### C.1 Subpermanents

To motivate the definition of subpermanents, we need a lemma which links permanents to counting permutations.

**Lemma C.1.** *Let  $A$  be an  $m \times n$   $(0, 1)$ -matrix, where  $m \leq n$ . Then  $\text{perm}(A) = \#\wp(A)$ , where*

$$\wp(A) = \{\tau \in S_{n,m} \mid a_{i,\tau(i)} = 1 \text{ for } i = 1, \dots, m\}. \quad (\text{C.1})$$

*Proof.* This follows from Definition 3.1 from the main text and the fact that  $A$  is a  $(0, 1)$ -matrix.  $\square$

The notion of subpermanents now arises by sorting the number of permutations in  $\wp(A)$  by how many entries from the *right-most* and *left-most* rectangular blocks are in each permutation.

**Definition C.1.** *Let  $A$  be an  $m \times n$  block rectangular matrix minimally parametrised by  $\alpha, \beta, \gamma, m$ , where  $m \leq n$  and  $k = \dim(\alpha) \geq 2$ . For  $0 \leq r \leq \alpha_1$  and  $0 \leq s \leq \alpha_k$ , define the set  $\wp_{rs}(A)$  by*

$$\wp_{rs}(A) = \left\{ \tau \in \wp(A) \mid \#\{i \mid \tau(i) \leq \alpha_1\} = r \text{ and } \#\{i \mid \tau(i) > \sum_{l=1}^{k-1} \alpha_l\} = s \right\} \quad (\text{C.2})$$

We define  $\text{perm}_{rs}(A) = \#\wp_{rs}(A)$  to be the  $r, s$ -subpermanent of  $A$ .

Note that

$$\text{perm}(A) = \sum_{r=0}^{\alpha_1} \sum_{s=0}^{\alpha_k} \text{perm}_{rs}(A). \quad (\text{C.3})$$

Hence, if we can calculate all subpermanents of  $A$ , we obtain the permanent as well. It turns out that the subpermanents of  $A$  can be explicitly related to those of another block rectangular matrix which has fewer rows than  $A$ . This removal of rows will happen in a systematic fashion via series of moves, called *trimming*, *splitting* and *merging*, to be defined momentarily. By applying these moves repeatedly to  $A$ , we end up with a block rectangular matrix with a very simple parametrisation, whose subpermanents can be calculated explicitly. Having done so, we may then reverse the moves we performed and update the subpermanents accordingly. This will allow us to obtain the subpermanents of  $A$ . We now proceed by defining trimming, splitting and merging, and establishing the subpermanent relations.

## C.2 Trimming, splitting and merging

Throughout this section, let  $A$  be an  $m \times n$  complete block rectangular matrix minimally parametrised by  $\alpha, \beta, \gamma, m$ , where  $m \leq n$  and  $k = \dim(\alpha) \geq 2$ .

**Definition C.2.** *The operations trimming, splitting and merging are defined as follows.*

- If  $m > \beta_1 > 0$ , a top trim of  $A$  is the process of replacing  $A$  with the block rectangular matrix  $A^{\text{tt}}$  parametrised by  $\alpha, \beta^{\text{tt}}, \gamma, m^{\text{tt}}$ , where  $\beta^{\text{tt}} = (0, \beta_2, \dots, \beta_{k-1})$  and  $m^{\text{tt}} = m - \beta_1$ . This is equivalent to deleting the top  $\beta_1$  rows of  $A$ .

- If  $m > \gamma_{k-1} > 0$ , a bottom trim of  $A$  is the process of replacing  $A$  with the block rectangular matrix  $A^{\text{bt}}$  parametrised by  $\alpha, \beta, \gamma^{\text{bt}}, m^{\text{bt}}$ , where  $\gamma^{\text{bt}} = (\gamma_1, \dots, \gamma_{k-2}, 0)$  and  $m^{\text{bt}} = m - \gamma_{k-1}$ . This is equivalent to deleting the bottom  $\gamma_{k-1}$  rows of  $A$ .
- If  $m > \beta_{k-1} > 0$ , a top split of  $A$  is the process of replacing  $A$  with the block rectangular matrix  $A^{\text{ts}}$  parametrised by  $\alpha, \beta^{\text{ts}}, \gamma, m^{\text{ts}}$ , where  $\beta^{\text{ts}} = (\beta_1, \dots, \beta_{k-2}, 0)$  and  $m^{\text{ts}} = m - \beta_{k-1}$ . This is equivalent to deleting rows  $\sum_{r=1}^{k-2} \beta_r + 1, \dots, \sum_{r=1}^{k-1} \beta_r$  from  $A$ .
- If  $m > \gamma_1 > 0$ , a bottom split of  $A$  is the process of replacing  $A$  with the block rectangular matrix  $A^{\text{bs}}$  parametrised by  $\alpha, \beta, \gamma^{\text{bs}}, m^{\text{bs}}$ , where  $\gamma^{\text{bs}} = (0, \gamma_2, \dots, \gamma_{k-1})$  and  $m^{\text{bs}} = m - \gamma_1$ . This is equivalent to deleting rows  $m - \sum_{r=1}^{k-1} \gamma_r + 1, \dots, m - \sum_{r=1}^{k-2} \gamma_r$  from  $A$ .
- If  $\beta_1 = \gamma_1 = 0$ , then a left merge of  $A$  is the process of replacing  $\alpha, \beta, \gamma$  by

$$\alpha^{\text{lm}} = (\alpha_1 + \alpha_2, \alpha_3, \dots, \alpha_k), \quad \beta^{\text{lm}} = (\beta_2, \dots, \beta_{k-1}), \quad \gamma^{\text{lm}} = (\gamma_2, \dots, \gamma_{k-1}).$$

This is equivalent to merging the two left-most rectangular blocks when their vertical positions and heights match.

- If  $\beta_{k-1} = \gamma_{k-1} = 0$ , then a right merge of  $A$  is the process of replacing  $\alpha, \beta, \gamma$  by

$$\alpha^{\text{rm}} = (\alpha_1, \dots, \alpha_{k-1}, \alpha_{k-1} + \alpha_k), \quad \beta^{\text{rm}} = (\beta_1, \dots, \beta_{k-2}), \quad \gamma^{\text{rm}} = (\gamma_1, \dots, \gamma_{k-2}).$$

This is equivalent to merging the two right-most rectangular blocks when their vertical positions and heights match. Note that merging does not actually change the matrix  $A$ , only its parametrisation.

Combining trimming, splitting and merging in the following way, we obtain the reduction algorithm.

**Lemma C.2.** *After applying the reduction algorithm,  $A$  is still a complete block rectangular matrix with a minimal parametrisation.*

*Proof.* It is easy to check that if conditions (3.2) and (3.3) from the main text were satisfied to begin with, then they are still satisfied after applying the algorithm. Hence,  $A$  is still block rectangular. Also, since trimming and splitting only remove rows, and merging only changes the parametrisation of  $A$ , it is not possible that either move will introduce a new row of zeros. Hence,  $A$  is still complete.

Assume for a contradiction that  $A$  is no longer minimally parametrised. Then for some  $1 \leq r \leq k-1$ , we have  $\beta_r = \gamma_r = 0$ . As  $A$  was minimally parametrised before applying the algorithm, we cannot have  $1 < r < k-1$ . Hence,  $r = 1$  or  $r = k-1$ . If  $r = k-1$ , then the right merge step in line 17 was not performed, a contradiction. Similarly, if  $r = 1$ , then the left merge step in line 8 was not performed, since none of the steps from line 10 to line 18 change the values of  $\beta_1, \gamma_1$ . Hence there exists no such  $r$ , and so the parametrisation of  $A$  is still minimal.  $\square$

**Algorithm 1** Reduction

---

**Require:** Matrix  $A$ 

```

1: if  $m > \beta_1 > 0$  then
2:    $A \leftarrow A^{\text{tt}}$ 
3: end if
4: if  $m > \gamma_1 > 0$  then
5:    $A \leftarrow A^{\text{bs}}$ 
6: end if
7: if  $\beta_1 = \gamma_1 = 0$  then
8:    $A \leftarrow A^{\text{lm}}$ 
9: end if
10: if  $m > \gamma_{k-1} > 0$  then
11:    $A \leftarrow A^{\text{bt}}$ 
12: end if
13: if  $m > \beta_{k-1} > 0$  then
14:    $A \leftarrow A^{\text{ts}}$ 
15: end if
16: if  $\beta_{k-1} = \gamma_{k-1} = 0$  then
17:    $A \leftarrow A^{\text{rm}}$ 
18: end if

```

---

Note that in each **if** statement of the reduction algorithm, the value of  $k + m$  is strictly decreased, so that if we repeatedly apply the algorithm, there will come a time where no move is possible. We say that the matrix  $A$  is *reduced* if this is the case.

**Lemma C.3.** *The matrix  $A$  is reduced if and only if exactly one of the following holds.*

1.  $k = 1$ ,
2.  $k = 2, \beta_1 = 0$  and  $\gamma_1 = m$ ,
3.  $k = 2, \beta_1 = m$  and  $\gamma_1 = 0$ ,
4.  $k = 3, \beta = (0, m)$  and  $\gamma = (m, 0)$ .

*Proof.* It is easy to check that if either condition holds, then no further reduction of  $A$  is possible. For the converse, we consider case by case. If  $k = 1$ , then we are done. Suppose that  $k = 2$ . If  $0 < \beta_1 < m$  or  $0 < \gamma_1 < m$ , then it would be possible to perform a trim or a split. Also, if  $\beta_1 = \gamma_1 = 0$ , then it would be possible to perform a left (or right) merge. Hence one of case 2 or 3 must hold.

Finally, assume that  $k \geq 3$ . By the same argument as above, one of the following must be true.

- $\beta = (0, \dots, 0, m)$  and  $\gamma = (m, 0, \dots, 0)$ ,
- $\beta = (m, 0, \dots, 0)$  and  $\gamma = (0, \dots, 0, m)$ .

Assume that the former is true. By Lemma C.2, we know that the parametrisation is minimal. This forces  $k = 3$  and so case 4 holds. Finally, if the latter is true, then again minimality of the parametrisation forces  $\beta = (m, 0)$  and  $\gamma = (0, m)$ . However, this parametrisation will yield a matrix with a row of zeros, which contradicts Lemma C.2.  $\square$

In Figure 1, we see how the matrix from Figure 3a from the main text is affected by the reduction algorithm. Note that the final parametrisation belongs to case 3 in Lemma C.3.

We will now derive formulae for the subpermanents of a reduced matrix  $A$ . In Definition C.1, we only defined subpermanents for matrices with  $k = \dim(\alpha) \geq 2$ . In condition 1 in Lemma C.3,  $A$  is an  $m \times n$  matrix of ones, parametrised with  $k = 1$ . We therefore modify the reduction algorithm slightly by declaring that if  $k = 2$  and  $\beta_1 = \gamma_1 = 0$ , then we ignore the left and right merge moves in lines 7 and 16. This allows us to change condition 1 in Lemma C.3 to the following.

1'.  $k = 2$  and  $\beta_1 = \gamma_1 = 0$ .

Note that this parametrisation is no longer minimal.

**Notation.** For whole numbers  $i, j$ , if  $j \leq i$ , we let

$$\binom{i}{j} = \frac{i!}{j!(i-j)!}, \quad \left\{ \begin{matrix} i \\ j \end{matrix} \right\} = \frac{i!}{(i-j)!},$$

and declare that  $\binom{i}{j} = \left\{ \begin{matrix} i \\ j \end{matrix} \right\} = 0$  if  $j > i$ .

**Lemma C.4.** The subpermanents of a reduced matrix  $A$  take the following form.

1'. If  $k = 2$  and  $\beta_1 = \gamma_1 = 0$ , then

$$\text{perm}_{rs}(A) = \begin{cases} \binom{m}{r} \left\{ \begin{matrix} \alpha_1 \\ r \end{matrix} \right\} \left\{ \begin{matrix} \alpha_2 \\ s \end{matrix} \right\} & \text{if } r + s = m, \\ 0 & \text{otherwise.} \end{cases}$$

2. If  $k = 2, \beta_1 = 0$  and  $\gamma_1 = m$ , then

$$\text{perm}_{rs}(A) = \begin{cases} \left\{ \begin{matrix} \alpha_2 \\ m \end{matrix} \right\} & \text{if } r = 0, s = m, \\ 0 & \text{otherwise.} \end{cases}$$

3. If  $k = 2, \beta_1 = m$  and  $\gamma_1 = 0$ , then

$$\text{perm}_{rs}(A) = \begin{cases} \left\{ \begin{matrix} \alpha_1 \\ m \end{matrix} \right\} & \text{if } r = m, s = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Figure 1: Reducing the matrix in Figure 3a.

4. If  $k = 3, \beta = (0, m)$  and  $\gamma = (m, 0)$ , then

$$\text{perm}_{rs}(A) = \begin{cases} \left\{ \begin{smallmatrix} \alpha_2 \\ m \end{smallmatrix} \right\} & \text{if } r = s = 0, \\ 0 & \text{otherwise.} \end{cases}$$

*Proof.* We prove the first case for illustration. First note that  $\wp_{rs}(A) = \emptyset$  when  $r+s \neq m$ . When  $r+s = m$ , every permutation  $\tau \in \wp_{rs}(A)$  can be constructed as follows.

- (i) Choose a subset  $I$  of size  $r$  from  $\{1, \dots, m\}$ .
- (ii) Assign values  $\tau(i)$  for all  $i \in I$  such that  $\tau(i) \leq \alpha_1$ .
- (iii) Assign values  $\tau(i)$  for all  $i \notin I$  such that  $\tau(i) > \alpha_1$ .

There are  $\binom{m}{r}$  ways to perform step (i),  $\left\{ \begin{smallmatrix} \alpha_1 \\ r \end{smallmatrix} \right\}$  ways to perform step (ii), and  $\left\{ \begin{smallmatrix} n-\alpha_1 \\ s \end{smallmatrix} \right\} = \left\{ \begin{smallmatrix} \alpha_2 \\ s \end{smallmatrix} \right\}$  ways to perform step (iii).  $\square$

We now show how the subpermanents of  $A$  are affected by trimming, splitting and merging.

**Lemma C.5.** *The subpermanents of  $A$  are related to those of  $A^{\text{tt}}, A^{\text{bt}}, A^{\text{ts}}, A^{\text{bs}}, A^{\text{lm}}$  and  $A^{\text{rm}}$  via the following relationships.*

$$\begin{aligned} \text{perm}_{rs}(A) &= \left\{ \begin{smallmatrix} \alpha_k + \gamma_{k-1} - s \\ \gamma_{k-1} \end{smallmatrix} \right\} \text{perm}_{r, s-\gamma_{k-1}}(A^{\text{bt}}) \\ &= \left\{ \begin{smallmatrix} \alpha_1 + \beta_1 - r \\ \beta_1 \end{smallmatrix} \right\} \text{perm}_{r-\beta_1, s}(A^{\text{tt}}) \\ &= \sum_{l=0}^{\alpha_k} \left\{ \begin{smallmatrix} \sum_{q=2}^{k-1} \alpha_q - (m - \gamma_1 - r - l) \\ \gamma_1 + l - s \end{smallmatrix} \right\} \left\{ \begin{smallmatrix} \alpha_k - l \\ s - l \end{smallmatrix} \right\} \binom{\gamma_1}{s-l} \text{perm}_{rl}(A^{\text{bs}}) \\ &= \sum_{l=0}^{\alpha_1} \left\{ \begin{smallmatrix} \sum_{q=2}^{k-1} \alpha_q - (m - \beta_{k-1} - l - s) \\ \beta_{k-1} + l - r \end{smallmatrix} \right\} \left\{ \begin{smallmatrix} \alpha_1 - l \\ r - l \end{smallmatrix} \right\} \binom{\beta_{k-1}}{r-l} \text{perm}_{ls}(A^{\text{ts}}) \\ &= \sum_{l=r}^{\alpha_1 + \alpha_2} \frac{\left\{ \begin{smallmatrix} \alpha_2 \\ l-r \end{smallmatrix} \right\} \left\{ \begin{smallmatrix} \alpha_1 \\ r \end{smallmatrix} \right\}}{\left\{ \begin{smallmatrix} \alpha_1 + \alpha_2 \\ l \end{smallmatrix} \right\}} \binom{l}{r} \text{perm}_{ls}(A^{\text{lm}}) \\ &= \sum_{l=s}^{\alpha_{k-1} + \alpha_k} \frac{\left\{ \begin{smallmatrix} \alpha_{k-1} \\ l-s \end{smallmatrix} \right\} \left\{ \begin{smallmatrix} \alpha_k \\ s \end{smallmatrix} \right\}}{\left\{ \begin{smallmatrix} \alpha_{k-1} + \alpha_k \\ l \end{smallmatrix} \right\}} \binom{l}{s} \text{perm}_{rl}(A^{\text{rm}}). \end{aligned}$$

*Proof.* For the bottom trim, observe that every permutation  $\tau \in \wp_{r, s-\gamma_{k-1}}(A^{\text{bt}})$  may be extended to a permutation  $\bar{\tau} \in \wp_{rs}(A)$  by defining  $\bar{\tau}(i) = \tau(i)$  for  $1 \leq i \leq \sum_{l=1}^{k-2} \gamma_l$ , and choosing values  $\bar{\tau}(\sum_{l=1}^{k-2} \gamma_l + 1), \dots, \bar{\tau}(\sum_{l=1}^{k-1} \gamma_l)$  distinctly amongst the indices  $\{\sum_{l=1}^{k-1} \alpha_l + 1, \dots, n\}$ . There are  $\left\{ \begin{smallmatrix} \alpha_k + \gamma_{k-1} - s \\ \gamma_{k-1} \end{smallmatrix} \right\}$  ways to do this, and every permutation in  $\wp_{rs}(A)$  can be recovered this way, thus yielding the result. The corresponding relation for the top trim is proven similarly.

Next, we prove the result for the top split. Given  $1 \leq l \leq \alpha_1$ , we may extend every permutation  $\tau \in \wp_{ls}(A^{\text{ts}})$  to a permutation  $\bar{\tau} \in \wp_{rs}(A)$  as follows. First, we define  $\bar{\tau}(i) = \tau(i)$  for  $1 \leq i \leq \sum_{q=1}^{k-2} \beta_q$  and  $\bar{\tau}(i) = \tau(i - \beta_{k-1})$  for  $\sum_{q=1}^{k-1} \beta_q < i \leq m$ . To define  $\bar{\tau}(i)$  for  $\sum_{q=1}^{k-2} \beta_q < i \leq \sum_{q=1}^{k-1} \beta_q$ , we do the following.

- (i) Choose a subset  $I \subseteq \{\sum_{q=1}^{k-2} \beta_q + 1, \dots, \sum_{q=1}^{k-1} \beta_q\}$  of size  $r - l$ .
- (ii) For all  $i \in I$ , assign values  $\bar{\tau}(i)$  such that  $\bar{\tau}(i) \leq \alpha_1$ . Note that  $l$  such values out of the possible  $\alpha_1$  are already taken.
- (iii) For all  $i \notin I$ , assign values  $\bar{\tau}(i)$  such that  $\alpha_1 < \bar{\tau}(i) \leq \sum_{q=1}^{k-1} \alpha_q$ . Note that  $m - \beta_{k-1} - l - s$  such values out of the possible  $\sum_{q=2}^{k-1} \alpha_q$  are already taken.

There are  $\binom{\beta_{k-1}}{r-l}$  ways to perform step (i),  $\{\alpha_1 - l\}$  ways to perform step (ii) and  $\{\sum_{q=2}^{k-1} \alpha_q - (m - \beta_{k-1} - l - s)\}$  ways to perform step (iii). Any permutation in  $\wp_{rs}(A)$  can be recovered this way. Thus, summing over  $l = 0, \dots, \alpha_1$ , we get the result. The corresponding relation for the bottom split is proven similarly.

For the left merge, fix  $l$  and let  $\tau \in \wp_{ls}(A^{\text{lm}})$ . Let  $I = \tau^{-1}(\{1, \dots, \alpha_1 + \alpha_2\})$  and let  $\tau'$  be the restriction of  $\tau$  to  $\{1, \dots, m\} \setminus I$ . By assumption,  $\#I = l$ . Note that there are exactly  $\text{perm}_{ls}(A) / \{\alpha_1 + \alpha_2\}$  permutations in  $\wp_{ls}(A^{\text{lm}})$  which yield the same  $\tau'$ . Given  $\tau'$ , we may then extend it to a permutation  $\bar{\tau} \in \wp_{rs}(A)$  as follows.

- (i) Choose a subset  $J \subseteq I$  of size  $r$ .
- (ii) Assign values  $\bar{\tau}(i)$  such that  $\bar{\tau}(i) \leq \alpha_1$  for all  $i \in J$ .
- (iii) Assign values  $\bar{\tau}(i)$  such that  $\alpha_1 < \bar{\tau}(i) \leq \alpha_1 + \alpha_2$  for all  $i \in I \setminus J$ .

There are  $\binom{l}{r}$  ways to perform step (i),  $\{\alpha_1\}$  ways to perform step (ii), and  $\{\alpha_2\}$  ways to perform step (iii). For all permutations  $\tau$  in  $\wp_{rs}(A)$ , there exists an  $l$  such that  $\tau$  can be recovered this way. Hence, summing over  $l = r, \dots, \alpha_1 + \alpha_2$ , we get the result. The corresponding relation for the right merge is proven similarly.  $\square$

*Proof of Theorem C.1.* Consider the following procedure. First, reduce  $A$  by repeatedly applying the reduction algorithm until termination, and keep track of all the parameters involved in memory. Next, calculate the subpermanents of the resulting reduced matrix using Lemma C.4. Then for every move performed in reverse order, update the subpermanents using the relations in Lemma C.5. Once completed, we end up with all subpermanents of the original matrix  $A$ . The permanent of  $A$  is then the sum of the subpermanents, as shown in (C.3).

Now, reducing  $A$  requires at most  $k+m \leq 2n$  applications of the reduction algorithm. Further, each calculation involved in updating the subpermanents can be computed in polynomial time. Thus, the complexity of the procedure above grows polynomially with  $n$ , as required.  $\square$

In practice, we compute all subpermanents on a log scale, and use the `LogSumExp` function to compute the relations in Lemma [C.5](#).