Michigan Technological University

Digital Commons @ Michigan Tech

Dissertations, Master's Theses and Master's Reports

2021

Major Index over Descent Distributions of Standard Young Tableaux

Emily Anible
Michigan Technological University, eeanible@mtu.edu

Copyright 2021 Emily Anible

Recommended Citation

Anible, Emily, "Major Index over Descent Distributions of Standard Young Tableaux", Open Access Master's Thesis, Michigan Technological University, 2021.
https://doi.org/10.37099/mtu.dc.etdr/1339

Follow this and additional works at: https://digitalcommons.mtu.edu/etdr

 Part of the Discrete Mathematics and Combinatorics Commons

MAJOR INDEX OVER DESCENT DISTRIBUTIONS OF STANDARD YOUNG TABLEAUX

By

Emily E. Anible

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Mathematical Sciences

MICHIGAN TECHNOLOGICAL UNIVERSITY

2021

© 2021 Emily E. Anible

This thesis has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Mathematical Sciences.

Department of Mathematical Sciences

Thesis Advisor:	*Dr. William J. Keith*
Committee Member:	*Dr. David Hemmer*
Committee Member:	*Dr. Melissa Keranen*
Committee Member:	*Dr. Fabrizio Zanello*
Department Chair:	*Dr. Jiguang Sun*

Contents

List of Figures

Acknowledgments

My sincerest thanks to my thesis advisor, William Keith, without whose guiding hand throughout my study I would not have succeeded.

Thanks to my committee members: David Hemmer, Melissa Keranen, and Fabrizio Zanello.

Thank you to my dad, who has been by my side every step of the way.

I would also like to thank the following:

Ann Humes and Teresa Woods, for their constant support and sage advice in developing my skills as an instructor.

Brooke, Charlie, Corey, Hannah, Joy, Kai, Marshal, Prangya, Ryan, Rob, Tim, Sasha, Simone, Will, and Yasasya, colleagues and friends whose companionship throughout this journey has been an incredible boon.

My three brothers Jonathon, Tait, and Wyatt.

Abstract

This thesis concerns the generating functions $f_{\lambda,k}(q)$ for standard Young tableaux of shape λ with precisely k descents, aiming to find closed formulas for a general form given by Kirillov and Reshetikhin in 1988. Throughout, we approach various methods by which further closed forms could be found.

In Chapter 2 we give closed formulas for tableaux of any shape and minimal number of descents, which arise as principal specializations of Schur functions. We provide formulas for tableaux with three parts and one more than minimal number of descents, and demonstrate that the technique is extendable to any number of parts.

In Chapter 3 we aim to reduce the complexity of Kirillov and Reshetikhin's formula by identifying the summands contributing a nonzero amount to the polynomial. While the resulting formulas are lengthy, they greatly reduce the computation time for specified partition shapes and numbers of descents.

In Chapter 4 we investigate an apparent relation among $f_{\lambda,k}(q)$ and $f_{\lambda,k-1}(q)$ and discuss how this may lead to a greater insight of the distribution of these statistics.

Included appendices give a library of utilities in SageMath and Mathematica to generate the polynomials $f_{\lambda,k}$ and demonstrate Chapter 4's relationships.

Chapter 1

Introduction

A weakly decreasing sequence $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r)$ of positive integers whose sum is n is called a *partition* of n, denoted $\lambda \vdash n$. For an introduction to partitions and further definitions, we refer the reader to Andrews' and Macdonald's texts ([1] and [2]). One may describe this partition graphically using its *Ferrers' diagram* or *Young diagram*, a left-aligned arrangement of boxes in a grid where the number of boxes in the first row is λ_1, the number of boxes in the second is λ_2, and so on. We note that in Russian or French texts, these diagrams may appear inverted. For example, the partition $\lambda = (5, 4, 2, 1)$ of 12 may be represented as the following:

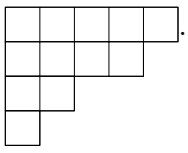

Figure 1.1: Partition of shape $\lambda = (5, 4, 2, 1)$

Occasionally we will write λ in *frequency notation* as $\lambda = a_1^{e_1} a_2^{e_2} \dots a_k^{e_k}$ with a_i strictly decreasing, where e_i is the number of *parts* of size a_i in λ. In this example, $\lambda = 5^1 4^1 2^1 1^1$. By mirroring the Young diagram of λ through its diagonal, we obtain the *conjugate partition* $\lambda' = (\lambda'_1, \dots, \lambda'_{\lambda_1})$. To each box in row i and column j of the Young diagram of a partition λ, we associate a positive integer given by the number of boxes directly below it and to its right, including itself. We denote this value $h_{i,j}$, the *hooklength* of box (i, j). Filling the Young diagram of $\lambda = (5, 4, 2, 1)$ above with the hooklength of each box, we have

8	6	4	3	1
6	4	2	1	
3	1			
1				

.

Figure 1.2: Partition of shape $\lambda = (5, 4, 2, 1)$ with its hooklengths

We may also *skew* a partition λ by another, μ, with the restriction that $\mu_i \leq \lambda_i$ for all i. The skew partition $\lambda \setminus \mu$ is the set of boxes in the Young diagram of λ but not in μ. Skewing $\lambda = (5, 4, 2, 1)$ by $\mu = (3, 2, 1)$, we have the following *skew diagram*, with hooklengths filled:

			3	1
		2	1	
	1			
1				

.

Figure 1.3: $(5, 4, 2, 1) \setminus (3, 2, 1)$ with its hooklengths.

For each partition shape λ, we may choose to fill its Young diagram with integers $1, 2, \ldots, n$, with the restriction that the numbers must increase across columns and down rows. The set of all fillings obeying these restrictions is called the set of *standard Young tableaux* of shape λ, denoted $\mathrm{SYT}(\lambda)$. If we instead loosen this restriction to allow for weakly increasing integers along the rows of λ, we obtain the set of *semistandard Young tableaux* of λ, $\mathrm{SSYT}(\lambda)$. We may fill the Young diagrams associated to skew partitions in the same way, giving the sets of *skew-semistandard* and *skew-standard Young tableaux*. However, we will mainly concern ourselves with $\mathrm{SYT}(\lambda)$ in this paper. Originally written by Frame, Robinson, and Thrall [3], the number of standard Young tableaux of shape λ, $|SYT(\lambda)|$, is given by the following:

$$f^\lambda := |SYT(\lambda)| = \frac{n!}{\prod_{h_{(i,j)} \in \lambda} h_{(i,j)}}. \tag{1.1}$$

Upon the standard Young tableaux of shape λ, we may define a *statistic*: a function $f : \mathrm{SYT}(\lambda) \to \mathbb{Z}$ which associates each tableau with an integer. We note that though the statistics we are interested in are integer-valued, but they need not be in general. One such tableau statistic is the *descent number*. Within a tableau τ, we call an occurrence of the integer $i + 1$ in a row below i a *descent*. The set of i for which $i + 1$ appears in a row lower is denoted by $\mathrm{Des}(\tau)$, the *descent set* of τ. The descent number, then, is given by $\mathrm{des}(\tau) := |\mathrm{Des}(\tau)|$. We note that for all $\tau \in \mathrm{SYT}(\lambda)$, we have $r - 1 \leq \mathrm{des}(\tau) \leq n - \lambda_1$. Another important statistic is the *major index* of a tableau, given by $\mathrm{maj}(\tau) := \sum_{i \in \mathrm{Des}(\tau)} i$. For example, the following SYT of

$\lambda = (5, 4, 2, 1)$ has descent set $\{2, 6, 9\}$ and major index 17:

1	2	5	6	12
3	4	9	11	
7	8			
10				

.

Figure 1.4: A standard Young tableaux of shape $(5, 4, 2, 1)$.

For a statistic $f(s)$ on the elements s of a set S, its *generating function* is $\sum_{s \in S} q^{f(s)}$, with q an indeterminate. A generating function, then, is an element of the polynomial ring $\mathbb{Z}[q]$ if S is finite and $f(s)$ is restricted to nonnegative integer values. One example is the polynomial $f_\lambda := \sum_{\tau \in \mathrm{SYT}(\lambda)} q^{\mathrm{maj}(\tau)}$, the generating function for the major index on the set $\mathrm{SYT}(\lambda)$. We will often say this "counts" the number of $\tau \in \mathrm{SYT}(\lambda)$ with major index m on the coefficient of q^m, since each tableau τ contributes precisely $q^{\mathrm{maj}(\tau)}$ to the polynomial. The polynomial f_λ is a quintessential example of a *q-analog*, a polynomial in q which, when one lets $q \to 1$, gives some other combinatorially interesting value. In this case, f_λ is a q-analog of f^λ. When one has a q-analog, it is desirable to provide a simple representation for the polynomial, referred to as a *closed form*. Richard Stanley gave a closed-form representation for f_λ in [3] as

$$f_\lambda := \sum_{\tau \in \mathrm{SYT}(\lambda)} q^{\mathrm{maj}(\tau)} = \frac{q^{\sum_i (i-1)\lambda_i} [n]_q!}{\prod_{h_{i,j} \in \lambda} [h_{i,j}]_q}, \tag{1.2}$$

where $[n]_q := 1 + q + \cdots + q^{n-1} = \frac{1-q^n}{1-q}$ and $[n]_q! := [n]_q [n-1]_1 \ldots [1]_q$ are q-analogs of n and $n!$. Letting $|q| < 1$, we can think of these polynomials as existing within $\mathbb{Z}[[q]]$,

the ring of formal power series in q.

Another important object is the q-binomial coefficient, a q-analog of the binomial coefficient $\binom{M+N}{N}$. We write this as

$$\begin{bmatrix} M+N \\ N \end{bmatrix}_q = \frac{[M+N]!_q}{[M]!_q[N]!_q} = \frac{(1-q^{M+1})(1-q^{M+2})\dots(1-q^{M+N})}{(1-q)(1-q^2)\dots(1-q^N)}. \tag{1.3}$$

By convention, if $M \geq 0$, then $\begin{bmatrix} M \\ 0 \end{bmatrix}_q = 1$ and if $M < N$, then $\begin{bmatrix} M \\ N \end{bmatrix}_q = 0$. The q-binomial coefficient has several useful, though not entirely obvious properties. First, $\begin{bmatrix} M+N \\ N \end{bmatrix}_q$ is a polynomial in q with positive integer coefficients which can be seen by iterating an extension to the Pascal relation for q-binomial coefficients, the q-Pascal relation $\begin{bmatrix} M+N \\ N \end{bmatrix}_q = \begin{bmatrix} M+N-1 \\ N \end{bmatrix}_q + q^M \begin{bmatrix} M+N-1 \\ N-1 \end{bmatrix}_q$, to its boundary conditions. An elegant interpretation, given in [4], describes $\begin{bmatrix} M+N \\ N \end{bmatrix}_q$ in terms of the number of M-dimensional subspaces of an $(M+N)$-dimensional vector space over the field of order q (a prime power). This description is used to interpret the coefficient of q^n in the resulting polynomial as the number of partitions of n into at most M parts with each part of size at most N (i.e. the partitions of n in the $M \times N$ box). To refer to the coefficient of q^n for a polynomial $f(q)$, we will typically write $[q^n]f(q)$.

Definition 1 *A polynomial $f(q) = \sum_{i=m}^{M} a_i q^i$ with coefficients $a_i \in \mathbb{R}$ is said to be symmetric if $[q^{m+i}]f(q) = [q^{M-i}]f(q)$ for all i. Equivalently, $f(q)$ is symmetric if $f(q) = q^A f(q^{-1})$ for some A. A symmetric polynomial $f(q)$ has central degree $(M+m)/2$ and two symmetric polynomials with the same central degree are said to*

be concentric.

The q-binomial is also a symmetric polynomial. Since $[q^i]\begin{bmatrix} M+N \\ N \end{bmatrix}_q$ is equal to the number of partitions of size i in the $M \times N$ box, we may obtain $[q^{MN-i}]\begin{bmatrix} M+N \\ N \end{bmatrix}_q$ by taking the complement of each given partition of size i in the box, then rotating the box 180 degrees about its center. The resulting partition shape has size $MN - i$, and is in one-to-one correspondence with its complement, so these two sets of partitions are equinumerous.

Definition 2 *A polynomial $f(q) = \sum_{i=m}^{M} a_i q^i$ with coefficients $a_i \in \mathbb{R}$ is said to be unimodal if the sequence of coefficients $(a_m, a_{m+1}, \ldots, a_M)$ is unimodal; that is, if $a_m \leq a_{m+1} \leq \cdots \leq a_j \geq a_{j+1} \geq \cdots \geq a_M$ for some $m \leq j \leq M$.*

Unlike the symmetricity of q-binomial coefficients, it is rather difficult to combinatorially show unimodality. The first injective map proving this fact was given by Kathleen O'Hara in 1990 [5]. O'Hara's theorem gives a *symmetric chain decomposition* of a partially ordered set corresponding to the q-binomial coefficient, separating elements into co-centered maximal length chains. Doron Zeilberger gave an algebraic interpretation of her theorem in terms of integer partitions in [6], which we write here as in [7]:

Theorem 1 (The KOH Theorem) *Let $\lambda = (\lambda_1, \lambda_2, \ldots) \vdash N$ and $Y_i = \sum_{j=1}^{i} \lambda_j$,*

with $Y_0 = 0$. *Then*

$$\begin{bmatrix} M+N \\ N \end{bmatrix}_q = \sum_{\lambda \vdash N} q^{\sum_{i=1}^{\infty} \lambda_i^2 - \lambda_i} \prod_{j \geq 1} \begin{bmatrix} j(M+2) - Y_{j-1} - Y_{j+1} \\ \lambda_j - \lambda_{j+1} \end{bmatrix}_q . \tag{1.4}$$

To see that (1.4) yields the desired property, we note that the product of two symmetric and unimodal polynomials is also a symmetric and unimodal polynomial. Algebraically, one can confirm that each term in the sum over partitions of N has the same central degree. By iterating the theorem downward to degenerate cases and instances of $\begin{bmatrix} A \\ 1 \end{bmatrix}_q = 1 + q + q^2 + \cdots + q^{A-1}$ for some integer A, one obtains both the symmetricity and unimodality of $\begin{bmatrix} M+N \\ N \end{bmatrix}_q$. We also define the class of *symmetric functions* in n variables known as the *Schur polynomials*. Symmetric functions are named as such because any permutation of their variables yields the same function – not necessarily because they are symmetric as in Definition 1. A *composition* μ of n is a sequence of positive integers whose sum is n. The *content* of a tableau $\tau \in \mathrm{SSYT}(\lambda \backslash \rho)$ is the composition $t = (t_1, t_2, \dots)$ of $|\tau|$, where t_j gives the number of times j appears in τ. For future reference, the set of (skew-) semistandard Young tableaux of shape $\lambda \backslash \rho$ with content μ is denoted as $\mathrm{SSYT}(\lambda \backslash \rho, \mu)$. Let $x_1, \dots, x_n$ be variables and denote $x^\tau = x_1^{t_1} x_2^{t_2} \dots x_n^{t_n}$. The *Schur polynomial indexed by* $\lambda \backslash \rho$ is defined as

$$s_{\lambda \backslash \rho}(x_1, \dots, x_n) = \sum_{\tau \in \mathrm{SSYT}(\lambda \backslash \rho)} x^\tau . \tag{1.5}$$

It is a fact that when ρ is the empty partition, the *principal specialization* of this Schur polynomial, $s_\lambda(1, q, \ldots, q^n)$, is always unimodal [2]. While there are many proofs for the unimodality of $s_\lambda(1, q, \ldots, q^n)$, Goodman, O'Hara, and Stanton ([8]) were able to conclude this fact using the *Kirillov-Reshetikhin formula* below (1.17). Schur polynomials have myriad uses in algebraic combinatorics, but we are primarily concerned with the *Jacobi-Trudi identities* applied to the principal specialization ([2], page 41):

Theorem 2 *For* $\lambda = (\lambda_1, \ldots, \lambda_r)$, $\rho = (\rho_1, \ldots, \rho_r)$, $0 \leq \rho_i \leq \lambda_i$,

$$s_{\lambda\backslash\rho}(1, q, \ldots, q^n) = \det\left(\begin{bmatrix} n-1+\lambda_i-\rho_j-i+j \\ n-1 \end{bmatrix}_q\right)_{1\leq i,j\leq r} \tag{1.6}$$

$$= \det\left(q^{\binom{\lambda'_i-\rho'_j-i+j}{2}}\begin{bmatrix} n \\ \lambda'_i-\rho'_j-i+j \end{bmatrix}_q\right)_{1\leq i,j\leq \lambda_1}. \tag{1.7}$$

The main focus of this paper concerns the generating function $f_{\lambda,k}(q)$ for the major index over $\mathrm{SYT}(\lambda, k)$, the set of standard Young tableaux of shape λ with exactly k descents:

$$f_{\lambda,k}(q) := \sum_{\substack{\tau\in\mathrm{SYT}(\lambda) \\ \mathrm{des}(\tau)=k}} q^{\mathrm{maj}(\tau)}. \tag{1.8}$$

The motivation for the study of these polynomials arises from the work of Cheng, Elizalde, Kasraoui, and Sagan [9] concerning polynomials $A_{n,k}(q)$, the major index

statistic of certain pattern-avoiding permutations of length n with exactly k descents. Sagan conjectured that the $A_{n,k}(q)$ were unimodal, a fact proven by William Keith in [10]. The key fact is that the $A_{n,k}(q)$ can be written as a sum of concentric polynomials $f_{\lambda,k}(q)$ shifted by some amount, reminiscent of the concentric polynomials in (1.4). However, this requires us to prove that the $f_{\lambda,k}(q)$ themselves are unimodal.

1.1 q-Kostka polynomials and the Kirillov-Reshetikhin Formula

The *q-Kostka polynomial* associated to $\mathrm{SSYT}(\lambda, \mu)$ is

$$K_{\lambda,\mu}(q) = \sum_{\tau \in \mathrm{SSYT}(\lambda,\mu)} q^{\mathrm{charge}(\tau)}, \tag{1.9}$$

where $\mathrm{charge}(\tau)$ is the *charge statistic* on semistandard Young tableaux. A full reference on the charge statistic for semistandard Young Tableaux may be found in [11]. To restrict to standard tableaux, let the content $\mu = 1^{|\lambda|}$, the all ones composition, so then

$$K_{\lambda,1^{|\lambda|}}(q) = \sum_{\tau \in \mathrm{SYT}(\lambda)} q^{\mathrm{charge}(\tau)}. \tag{1.10}$$

We will limit our definition of the charge statistic to standard young tableaux. For $\tau \in \mathrm{SYT}(\lambda)$, define the *standard word* or *reading word* of τ, $\mathrm{rw}(\tau)$, as a concatenated list of the entries in the tableau beginning with the last row. For instance, consider the tableau τ from Figure 1.4:

$$\tau = \begin{array}{|c|c|c|c|c|} \hline 1 & 2 & 5 & 6 & 12 \\ \hline 3 & 4 & 9 & 11 \\ \cline{1-4} 7 & 8 \\ \cline{1-2} 10 \\ \cline{1-1} \end{array}.$$

This tableau has reading word $\mathrm{rw}(\tau) = (10, 7, 8, 3, 4, 9, 11, 1, 2, 5, 6, 12)$. Let the *index of a letter* i in a reading word be the number of times a letter $j < i$ has the letter $j+1$ to its right. The *charge of a reading word* is the sum of the indices of all letters in the word. For instance, the index of 10 is 6, since among the numbers $i = 1, 2, \ldots, 9$, precisely six of them have $i+1$ to their right in the word. In this example, the indices for each letter in $\mathrm{rw}(\tau)$ are

$\mathrm{rw}(\tau) = ($	10	7	8	3	4	9	11	1	2	5	6	12	$)$
Indices:	6	4	5	1	2	6	7	0	1	3	4	8	$= 47$

Finally, define the *charge* of a standard Young tableau τ to be the charge of its reading word. In our example, $\mathrm{charge}(\tau) = \mathrm{charge}(\mathrm{rw}(\tau)) = 47$. We can perform this computation in a slightly different way. Notice that for $i \in \{1, \ldots, |\lambda| - 1\}$, i is a descent in τ if and only if it appears to the right of $i + 1$ in $\mathrm{rw}(\tau)$: the entries of

τ which contribute to the charge and those which contribute to the major index are mutually exclusive and partition $\{1, \ldots, |\lambda| - 1\}$:

$$\mathrm{maj}(\tau) = \sum_{i \in \mathrm{Des}(\tau)} i, \tag{1.11}$$

$$\mathrm{charge}(\tau) = \sum_{i \in \{1,\ldots,|\lambda|-1\} \setminus \mathrm{Des}(\tau)} (|\lambda| - i). \tag{1.12}$$

Note that for a given tableau $\tau \in \mathrm{SYT}(\lambda)$, this gives

$$\mathrm{maj}(\tau) + \mathrm{charge}(\tau) = \binom{|\lambda|}{2}. \tag{1.13}$$

This allows us write f^λ in terms of $K_{\lambda,1^{|\lambda|}}(q)$:

$$f^\lambda(q^{-1}) = q^{-\binom{|\lambda|}{2}} K_{\lambda,1^{|\lambda|}}(q). \tag{1.14}$$

As an example, consider the two polynomials for $\lambda = (3, 3, 1)$:

$$f^{(3,3,1)}(q) = q^6 + q^7 + 2q^8 + 2q^9 + 3q^{10} + 3q^{11} + 3q^{12} + 2q^{13} + 2q^{14} + q^{15} + q^{16}$$

$$K_{(3,3,1),1^7}(q) = q^5 + q^6 + 2q^7 + 2q^8 + 3q^9 + 3q^{10} + 3q^{11} + 2q^{12} + 2q^{13} + q^{14} + q^{15}$$

A key observation is that (1.13) holds at the level of each tableau, so we may refine this polynomial by various classes of the tableaux. For instance, consider the charge

statistic over standard Young tableaux of shape λ with exactly k descents, or

$$K^k_{\lambda,1^{|\lambda|}}(q) = \sum_{\substack{\tau \in \mathrm{SYT}(\lambda) \\ \mathrm{des}(\tau)=k}} q^{\mathrm{charge}(\tau)}. \tag{1.15}$$

Thus, we may write $f_{\lambda,k}(q)$ in terms of this new polynomial, as seen in [8]:

$$f_{\lambda,k}(q^{-1}) = q^{-\binom{|\lambda|}{2}} K^k_{\lambda,1^{|\lambda|}}(q) \tag{1.16}$$

Both of these families of q-Kostka polynomials were studied by Kirillov and Reshetikhin in [12], who gave a formula for both in terms of a summation over admissible sequences of partitions, (α), together with certain statistics on these sequences, $c(\alpha)$ and $P^a_i(\alpha)$:

$$K_{\lambda,1^{|\lambda|}}(q) = \sum_{\alpha=(\mu',\alpha^1,\alpha^2,\dots)} q^{c(\alpha)} \prod_{a,i} \begin{bmatrix} P^a_i(\alpha) + \alpha^a_i - \alpha^a_{i+1} \\ \alpha^a_i - \alpha^a_{i+1} \end{bmatrix}_q \tag{1.17}$$

$$K^k_{\lambda,1^{|\lambda|}}(q) = \sum_{\substack{\alpha=(\mu',\alpha^1,\alpha^2,\dots) \\ \alpha^1_1=k}} q^{c(\alpha)} \prod_{a,i} \begin{bmatrix} P^a_i(\alpha) + \alpha^a_i - \alpha^a_{i+1} \\ \alpha^a_i - \alpha^a_{i+1} \end{bmatrix}_q. \tag{1.18}$$

Note that the latter differs from the former in a restriction on the first part of the second partition in the sequence. The construction of this formula decomposes these q-Kostka polynomials into symmetric polynomials with the same central degree. Since each term in the decomposition is nonnegative, this also suffices to show the unimodality of these polynomials, and thus $f_{\lambda,k}(q)$ as well. This, along with the

relation established in [10], is enough to show the unimodality of Sagan's $A_{n,k}(q)$. Though K-R provides a general formula for all $f_{\lambda,k}$, writing a single formula down quickly becomes computationally taxing. For fixed λ and k, one must sum over all admissible sequences of partition, α, which grows rapidly as one increases k and the number of parts of λ. In Chapter 3, we will discuss an avenue to reducing the computational load through restricting the admissibility conditions imposed upon the sequences α. If one were to find closed forms for $f_{\lambda,k}$ instead, we could altogether avoid computation. As we will see in the next section, such closed forms typically have fewer terms than the number of admissible sequences of $K^k_{\lambda,1^{|\lambda|}}(q)$. Further, these closed forms tend to be combinatorially interesting, allowing for additional interpretations of the polynomials.

1.2 Closed forms and beyond

In this paper, we aim to find closed forms for the major index distribution over $\mathrm{SYT}(\lambda, k)$, $f_{\lambda,k}(q)$, for several families of partition shapes and fixed k. Further, we explore potential avenues for extending these results to larger families and descent numbers. Recall Stanley's refinement of Frame-Robinson-Thrall's formula for the

number of SYT by major index:

$$\sum_{\tau \in \mathrm{SYT}(\lambda)} q^{\mathrm{maj}(\tau)} = \frac{q^{\sum_i (i-1)\lambda_i} [n]_q!}{\prod_{h_{i,j} \in \lambda} [h_{i,j}]_q}. \tag{1.19}$$

As an ultimate goal, finding a unifying closed formula for any λ and k would give a (closed) refinement of (1.19) by descent number, perhaps with the inclusion of an additional parameter.

One interesting facet of $f_{\lambda,k}$ was revealed in Keith's study of the polynomials in [10], and is explored further herein. The polynomials for extremal descent numbers yield principal specializations of Schur polynomials up to multiplication by a power of q. Keith showed the case for $f_{\lambda,k}$ for k maximal:

Theorem 3 *(Keith [10]) Let $\lambda = (\lambda_1, \ldots, \lambda_r) \vdash n, \alpha = a(\alpha) = (\lambda_2, \ldots, \lambda_r)$. Then*

$$f_{\lambda, n-\lambda_1} = q^{\binom{n-\lambda_1+1}{2}} s_{\alpha'}(1, q, \ldots, q^{\lambda_1 - 1}). \tag{1.20}$$

In Chapter 2, we show the complementary statement, $f_{\lambda,k}$ for k minimal:

Theorem 4 *Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r)$. Then*

$$f_{\lambda, r-1} = q^{\binom{r}{2}\lambda_r} \det \left(\begin{bmatrix} \lambda_i - \lambda_r + (r-i) \\ r - j \end{bmatrix}_q \right)_{1 \le i,j \le r}. \tag{1.21}$$

While this alone is fascinating, it is possible that the polynomials for non-extremal descent numbers can be expressed in terms of linear combinations of certain principal specializations or their shifts by a power of q. In fact, for two-rowed standard (and skew-standard) Young tableaux, there is a determinantal form which arises as a principal specialization of a Schur function:

Theorem 5 *(Keith [10]) Let $\lambda = (\lambda_1, \lambda_2)$ and $\mu = (\mu_1)$ with $0 \leq \mu_1 < \lambda_1$. Then*

$$f_{\lambda\setminus\mu,i}(q) = q^{i^2}\left(\begin{bmatrix}\lambda_1-\mu_1\\ i\end{bmatrix}_q\begin{bmatrix}\lambda_2\\ i\end{bmatrix}_q - \begin{bmatrix}\lambda_1+1\\ i\end{bmatrix}_q\begin{bmatrix}\lambda_2-\mu_1-1\\ i\end{bmatrix}_q\right) \tag{1.22}$$

$$= q^{i^2} s_{(\lambda_1-i,\lambda_2-i)\setminus\mu}(1, q, \ldots, q^{i+1}). \tag{1.23}$$

Other work, notably [13] by George Wang, yields a formula $f_{\lambda,k}$ for partitions of the form $\lambda = (\lambda_1, \lambda_2, 2^{h_2-2}, 2^{h_1-h_2})$: precisely those with *Durfee square* of size 2. In this manuscript, we give a closed formula for $f_{\lambda,k}(q)$ for standard Young tableau with three parts and one more than the minimal number of descents:

Theorem 6 *Let $\lambda = (n, k, j) \vdash N$. Then we have*

$$\begin{aligned} f_{\lambda,3} &= q^{-1}\big(f_{(n+1,k+1),3} - f_{(n+1,j),3} + f_{(k,j),3}\big) + q^{6j-8} f_{(n-j+2,k-j+2),3} \\ &\quad + \sum_{i=1}^{j-1} q^{6i-9}\big(f_{(n-i+3,k-i+3),3} - f_{(n-i+3,j-i+2),3} + f_{(k-i+2,j-i+2),3}\big) \\ &\quad - q^{3j+2k+1}(1+q^{n-k+1}+q^{n-j+2})\frac{(1-q^{j-1})(1-q^{k-j+1})(1-q^{n-k+1})(1-q^{n-j+2})}{(1-q)^2(1-q^2)(1-q^3)}. \end{aligned}$$

In terms of principal specializations of Schur functions (using the shorthand $s_\beta = s_\beta(1, q, \ldots, q^4)$*), we have*

$$\begin{aligned} f_{\lambda,3} = {} & q^8(s_{(n-2,k-2)} - s_{(n-2,j-3)} + s_{(k-3,j-3)}) + q^{6j+1}s_{(n-j-1,k-j-1)} \\ & + \sum_{i=1}^{j-1} q^{6i}(s_{(n-i,k-i)} - s_{(n-i,j-i-1)} + s_{(k-i-1,j-i-1)}) \\ & - q^{3j+2k+1}(1 + q^{n-k+1} + q^{n-j+2})\frac{(1-q^{j-1})(1-q^{k-j+1})(1-q^{n-k+1})(1-q^{n-j+2})}{(1-q)^2(1-q^2)(1-q^3)}. \end{aligned}$$

This theorem notably proves conjectures Conjectures 14 and 15 of [10], extending current work on the subject. Further, the ability for us to write $f_{(n,k,j),3}(q)$ (nearly) as a q-linear combination of Schur polynomials indicates that future extensions to larger numbers of descents or parts may take on a similar form.

Finally, several of the formulas we prove and conjecture herein have several regularities. We will see symmetry and equivalence (up to a shift) among these formulas, typically demonstrated by partition (or tableau) conjugation and complementation in some region. These are often the most noticeable with rectangular tableaux:

Theorem 7 *Let* $\lambda = (n, \ldots, n) = n^r \vdash nr$*. Then*

$$f_{\lambda,r}(q) = q^{\binom{r}{2}n}\left(\begin{bmatrix} n+r \\ r \end{bmatrix}_q - \begin{bmatrix} nr+1 \\ 1 \end{bmatrix}_q\right) = f_{\lambda,r-1}(q)\left(\begin{bmatrix} n+r \\ r \end{bmatrix}_q - \begin{bmatrix} nr+1 \\ 1 \end{bmatrix}_q\right). \quad (1.24)$$

We note that 1.24 demonstrates a pleasant result: since $f_{n^r,r}(q)$ is unimodal ([10]),

the difference of q-binomials $\begin{bmatrix} n+r \\ r \end{bmatrix}_q - \begin{bmatrix} nr+1 \\ 1 \end{bmatrix}_q$ is also unimodal. This is precisely the trivial case of F. Bergeron's *ad-bc* conjecture, which states

Conjecture 1 *(Bergeron [14]) Fix any positive integers a, b, c, d such that a is the smallest and $ad = bc$. Then the coefficients of the symmetric polynomial*

$$\begin{bmatrix} b+c \\ b \end{bmatrix}_q - \begin{bmatrix} a+d \\ d \end{bmatrix}_q \tag{1.25}$$

are nonnegative and unimodal.

Bergeron's conjecture, along with problems concerning the unimodality of differences of certain q-binomial coefficients and the strict unimodality of q-binomial coefficients have seen much interest interest in recent years ([14], [15], [16] [7], [17], [18]). It would be interesting to determine if closed forms for $f_{n^r,k}(q)$ with $k > r$ yield related non-trivial results in this vein.

Another theorem gives a relation upon tableaux whose partition shapes are complementary within the $r \times (\lambda_1 + 1)$ box:

Theorem 8 *Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r)$ and $\rho = (\lambda_1 + 1 - \lambda_r, \lambda_1 + 1 - \lambda_{r-1}, \ldots, \lambda_1 + 1 - \lambda_2, 1)$. Then*

$$f_{\lambda,r-1} = q^{(r-1)|\lambda| - \binom{r}{2}(\lambda_1+1)} f_{\rho,r-1}. \tag{1.26}$$

As an example, consider the partition $\lambda = (5, 3, 2)$ and its complement in the 3×6 box, $\rho = (4, 3, 1)$:

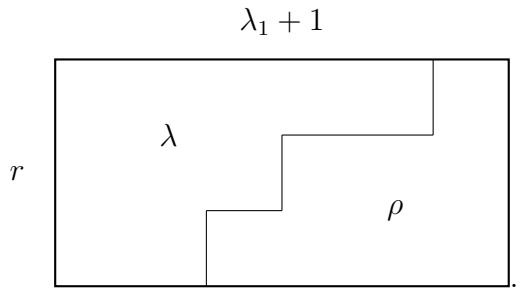

Figure 1.5: Complementary partitions in the $r \times (\lambda_1 + 1)$ box.

The distributions $f_{\lambda,2}(q)$ and $f_{\rho,2}(q)$ are given by the following:

$$
\begin{aligned}
f_{\lambda,2} = q^6 \det\left(\begin{bmatrix} \lambda_i + 1 - i \\ 3 - j \end{bmatrix}_q \right)_{1 \le i,j \le 3} &= q^6 \begin{vmatrix} \begin{bmatrix} 5 \\ 2 \end{bmatrix}_q & \begin{bmatrix} 5 \\ 1 \end{bmatrix}_q & \begin{bmatrix} 5 \\ 0 \end{bmatrix}_q \\ \begin{bmatrix} 2 \\ 2 \end{bmatrix}_q & \begin{bmatrix} 2 \\ 1 \end{bmatrix}_q & \begin{bmatrix} 2 \\ 0 \end{bmatrix}_q \\ \begin{bmatrix} 0 \\ 2 \end{bmatrix}_q & \begin{bmatrix} 0 \\ 1 \end{bmatrix}_q & \begin{bmatrix} 0 \\ 0 \end{bmatrix}_q \end{vmatrix} \\
&= q^6 \left(\begin{bmatrix} 5 \\ 2 \end{bmatrix}_q \begin{bmatrix} 2 \\ 1 \end{bmatrix}_q - \begin{bmatrix} 5 \\ 1 \end{bmatrix}_q \right) \\
&= q^7 + 2q^8 + 3q^9 + 3q^{10} + 3q^{11} + 2q^{12} + q^{13}
\end{aligned}
$$

$$f_{\rho,2} = q^3 \det\left(\begin{bmatrix} \rho_i + 2 - i \\ 3 - j \end{bmatrix}_q \right)_{1 \le i,j \le 3} = q^3 \begin{vmatrix} \begin{bmatrix} 5 \\ 2 \end{bmatrix}_q & \begin{bmatrix} 5 \\ 1 \end{bmatrix}_q & \begin{bmatrix} 5 \\ 0 \end{bmatrix}_q \\ \begin{bmatrix} 3 \\ 2 \end{bmatrix}_q & \begin{bmatrix} 3 \\ 1 \end{bmatrix}_q & \begin{bmatrix} 3 \\ 0 \end{bmatrix}_q \\ \begin{bmatrix} 0 \\ 2 \end{bmatrix}_q & \begin{bmatrix} 0 \\ 1 \end{bmatrix}_q & \begin{bmatrix} 0 \\ 0 \end{bmatrix}_q \end{vmatrix}$$

$$= q^3 \left(\begin{bmatrix} 5 \\ 2 \end{bmatrix}_q \begin{bmatrix} 3 \\ 1 \end{bmatrix}_q - \begin{bmatrix} 5 \\ 1 \end{bmatrix}_q \begin{bmatrix} 3 \\ 2 \end{bmatrix}_q \right)$$

$$= q^5 + 2q^6 + 3q^7 + 3q^8 + 3q^9 + 2q^{10} + q^{11}.$$

Note that $(r-1)|\lambda| - \binom{r}{2}(\lambda_1 + 1) = 2(10) - 3(6) = 2$, so Theorem 8 holds for this small example.

These relations arise frequently enough, even experimentally, to raise significant curiosity regarding potential hidden structures among the $f_{\lambda,k}(q)$. Indeed, Chapter 4 discusses one such possible structure based on polynomial division among families of tableaux.

1.3 Tableau threads

Throughout the paper, we make frequent use of the reference to consecutive runs of integers in the filling of a tableau. To this end, we define the *threads* of a tableau. Let $\lambda = (\lambda_1, \ldots, \lambda_r) \vdash n$ and $\tau \in \mathrm{SYT}(\lambda, k)$ for $r - 1 \le k \le n - \lambda_1$ have descent set $\mathrm{Des}(\tau) = \{d_1, \ldots, d_k\}$. Define the ith *thread* of τ as the sequence of consecutive positive integers $T_i(\tau) := (1 + d_{i-1}, \ldots, d_i)$, with $d_0 = 0$ and $d_{k+1} = n$. For $i > (k+1)$

or $i < 1$, let $T_i(\tau) = \emptyset$. As an example, consider the following tableaux τ:

$$\tau = \begin{array}{|c|c|c|c|c|c|c|c|c|}\hline 1 & 2 & 5 & 6 & 7 & 10 & 11 & 17 \\ \hline 3 & 4 & 8 & 9 & 15 & 16 \\ \cline{1-6} 12 & 13 & 14 \\ \cline{1-3}\end{array}.$$

Its descent set is $\mathrm{Des}(\tau) = \{2, 7, 11\}$, and so its threads are $T_1(\tau) = (1, 2)$, $T_2(\tau) = (3, \ldots, 7)$, $T_3(\tau) = (8, \ldots, 11)$, and $T_4(\tau) = (12, \ldots, 17)$. As a descent set does not identify a tableau, neither does the list of threads. Consider

$$\mu = \begin{array}{|c|c|c|c|c|c|c|c|c|}\hline 1 & 2 & 4 & 5 & 6 & 7 & 10 & 11 \\ \hline 3 & 8 & 9 & 15 & 16 & 17 \\ \cline{1-6} 12 & 13 & 14 \\ \cline{1-3}\end{array}.$$

The tableaux τ and μ both have the same shape, descent set, and threads, yet are not the same tableau. However, if we instead specify which portion of each thread is in each row, we may fully identify a tableau using its threads. Let $T_i^m(\tau)$ be the portion of $T_i(\tau)$ in the mth row of τ for $1 \leq i \leq k+1$ and $1 \leq m \leq r$. Then

$$\tau = \left(T_i^m(\tau) \right)_{\substack{1 \leq m \leq r \\ 1 \leq i \leq k+1}}. \tag{1.27}$$

The example tableau, τ, can be written in this way as

$$\tau = \begin{bmatrix} (1,2) & (5,6,7) & (10,11) & (17) \\ \emptyset & (3,4) & (8,9) & (15,16) \\ \emptyset & \emptyset & \emptyset & (12,13,14) \end{bmatrix}.$$

This is evidently reversible given a valid doubly-indexed list of threads T_i^m, as one may write the entries τ by reading the entries in the table left-to-right, top-to-bottom. For specificity, a valid list of threads has the following properties:

1. Each positive integer $1, \ldots, n$ appears exactly once among all threads;
2. Each non-empty entry T_i^m is a list of increasing consecutive positive integers;
3. The entries of non-empty T_i are less than those in T_j iff $i < j$;
4. The entries of non-empty T_i^m are less than those in T_i^n iff $m > n$.

This list of threads corresponds to a $\tau \in \mathrm{SYT}(\lambda, k)$ if the total number of entries across all sequences in row i gives λ_i for $1 \leq i \leq r$ and the largest positive integer in column j is the jth descent, d_j, for $1 \leq j \leq \mathrm{des}(\tau)$. Further, define $|T_i|$ (resp. $|T_i^m|$) as the *length* of the ith thread (resp. in the mth row) of τ, the total number of boxes of τ in the ith thread (resp. and in the mth row).

Chapter 2

Closed formulas and symmetries

2.1 Minimum descents

In [10], Keith provided a formula for $f_{\lambda,n-\lambda_1}(q)$, the polynomial for major index over descent of tableaux of shape $\lambda = (\lambda_1, \ldots, \lambda_r)$ with maximum number of descents. Here, we communicate a similar proof for $f_{\lambda,r-1}(q)$ and provide a formula regarding conjugate tableaux.

Theorem 4 *Let* $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r)$. *Then*

$$\begin{aligned} f_{\lambda,r-1} &= q^{\binom{r}{2}\lambda_r} \det\left(\begin{bmatrix} \lambda_i - \lambda_r + (r-i) \\ r-j \end{bmatrix}_q\right)_{1\le i,j\le r} \\ &= q^{\binom{r}{2}\lambda_r} \det\left(\begin{bmatrix} r-1+\lambda_i-\lambda_r-i+j \\ r-1 \end{bmatrix}_q\right)_{1\le i,j\le r} \\ &= \det\left(\begin{bmatrix} r \\ \lambda_i' - i + j \end{bmatrix}_q q^{\binom{\lambda_i'-i+j}{2}}\right)_{1\le i,j\le \lambda_1}. \end{aligned}$$

Proof. Suppose $r = 1$. Then $f_{\lambda,0} = 1$, as there is a single SYT of shape (λ_1) with major index 0. We have

$$\det\left(\begin{bmatrix} 1 \\ 1-i+j \end{bmatrix}_q q^{\binom{1-i+j}{2}}\right)_{1\le i,j\le\lambda_1} = 1,$$

as the matrix has $\begin{bmatrix}1\\1\end{bmatrix}_q q^{\binom{1}{2}} = 1$ on the diagonal, $\begin{bmatrix}1\\0\end{bmatrix}_q q^{\binom{0}{2}} = 1$ on the subdiagonal, and zeroes elsewhere.

Suppose now that $r > 1$, and let $\lambda = (\lambda_1, \ldots, \lambda_r)$. Consider the columns of the partition, labelled $\{1, 2, \ldots, \lambda_1\}$. Let S be a collection of the columns of a tableau $\tau \in \mathrm{SYT}(\lambda, r-1)$ corresponding to the columns inhabited by $T_r(\tau)$, the last thread of the tableau, excluding the first column. That is, a collection of columns S is valid if S contains $2, 3, \ldots, \lambda_r$, and if the bottom entry of a column in S is in row i of the tableau, then all columns to the right whose bottom entries are also in row i are also in the column. If $|T_r(\tau)| = 1$, then the first box in the last row of τ has the label n and so $S = \emptyset$.

Consider now $S \cup \{1\}$, the collection of columns corresponding to the entire last thread of τ. The removal of this thread removes the boxes $n-|S|, n-|S|+1, \ldots, n$, eliminates a single descent from τ, and reduces the major index by $n - (|S| + 1)$. Denote the resulting partition shape as $\lambda^{\downarrow(S\cup\{1\})}$. Performing this removal for all possible final threads, we have the recurrence

$$\begin{aligned} f_{\lambda,r-1} &= q^{n-1} f_{\lambda^{\downarrow\{1\}},r-2} + \sum_{\substack{S\subseteq\{2,\ldots,\lambda_1\} \\ S\neq\emptyset}} q^{n-(|S|+1)} f_{\lambda^{\downarrow(S\cup\{1\})},r-2} \\ &= \sum_{S\subseteq\{2,\ldots,\lambda_1\}} q^{n-(|S|+1)} f_{\lambda^{\downarrow(S\cup\{1\})},r-2}. \end{aligned}$$

Marking S in a dark gray and $\{1\}$ in a light gray, consider the following filling of $\lambda = (7, 3, 3, 2)$, where $S = \{2, 3, 6, 7\}$:

1	2	3	7	8	14	15	.
4	5	6					
9	10	13					
11	12						

Figure 2.1: SYT of shape $(7, 3, 3, 2)$ with last thread in columns $\{1, 2, 3, 6, 7\}$.

By induction, the claim of the theorem is now

$$\begin{aligned} f_{\lambda,r-1} &= \det\left(\begin{bmatrix} r \\ \lambda_i' - i + j \end{bmatrix}_q q^{\binom{\lambda_i'-i+j}{2}}\right)_{1\le i,j\le \lambda_1} \\ &= \sum_{S\subseteq\{2,\dots,\lambda_1\}} q^{n-(|S|+1)} \det\left(\begin{bmatrix} r-1 \\ \lambda_i' - \chi(i) - i + j \end{bmatrix}_q q^{\binom{\lambda_i'-\chi(i)-i+j}{2}}\right)_{1\le i,j\le \lambda_1}. \end{aligned}$$

Consider terms in which $\lambda_1 \notin S$, and pair with the terms given by $S \cup \{\lambda_1\}$. Denote δ_{i,λ_1} as the Dirac delta, such that δ_{i,λ_1} is 1 if $i = \lambda_1$, 0 otherwise. Then we have

$$\begin{aligned} &q^{n-(|S|+1)} \det\left(\begin{bmatrix} r-1 \\ \lambda_i' - \chi(i) - i + j \end{bmatrix}_q q^{\binom{\lambda_i'-\chi(i)-i+j}{2}}\right)_{1\le i,j\le \lambda_1} \\ &+ q^{n-(|S|+2)} \det\left(\begin{bmatrix} r-1 \\ \lambda_i' - \chi(i) - i + j - \delta_{i,\lambda_1} \end{bmatrix}_q q^{\binom{\lambda_i'-\chi(i)-i+j-\delta_{i,\lambda_1}}{2}}\right)_{1\le i,j\le \lambda_1} \\ &= q^{n-(|S|+2)} \left[q \det\left(\begin{bmatrix} r-1 \\ \lambda_i' - \chi(i) - i + j \end{bmatrix}_q q^{\binom{\lambda_i'-\chi(i)-i+j}{2}}\right)_{1\le i,j\le \lambda_1} \right. \\ &\left. + \det\left(\begin{bmatrix} r-1 \\ \lambda_i' - \chi(i) - i + j - \delta_{i,\lambda_1} \end{bmatrix}_q q^{\binom{\lambda_i'-\chi(i)-i+j-\delta_{i,\lambda_1}}{2}}\right)_{1\le i,j\le \lambda_1} \right]. \end{aligned}$$

Expand each determinant as a sum over permutations, and consider $\sigma = (\sigma_1, \dots, \sigma_{\lambda_1}) \in \mathfrak{S}_{\lambda_1}$. The terms in the sum that correspond to this permutation

are:

$$
\begin{aligned}
&q^{n-(|S|+2)}\left[q\prod_{i=1}^{\lambda_1}\begin{bmatrix} r-1 \\ \beta_i \end{bmatrix}_q q^{\binom{\beta_i}{2}} + \prod_{i=1}^{\lambda_1}\begin{bmatrix} r-1 \\ \beta_i - \delta_{i,\lambda_1} \end{bmatrix}_q q^{\binom{\beta_i-\delta_{i,\lambda_1}}{2}}\right] \\
&= q^{n-(|S|+2)}\left[\prod_{i=1}^{\lambda_1-1}\begin{bmatrix} r-1 \\ \beta_i \end{bmatrix}_q q^{\binom{\beta_i}{2}}\right]\left[q\begin{bmatrix} r-1 \\ \beta_{\lambda_1} \end{bmatrix}_q q^{\binom{\beta_{\lambda_1}}{2}} + \begin{bmatrix} r-1 \\ \beta_{\lambda_1}-1 \end{bmatrix}_q q^{\binom{\beta_{\lambda_1}-1}{2}}\right] \\
&= q^{n-(|S|+2)}\left[\prod_{i=1}^{\lambda_1-1}\begin{bmatrix} r-1 \\ \beta_i \end{bmatrix}_q q^{\binom{\beta_i}{2}}\right]\left[q^{\beta_{\lambda_1}}\begin{bmatrix} r-1 \\ \beta_{\lambda_1} \end{bmatrix}_q + \begin{bmatrix} r-1 \\ \beta_{\lambda_1}-1 \end{bmatrix}_q\right] q^{\binom{\beta_{\lambda_1}-1}{2}} \\
&= q^{n-(|S|+2)}\left[\prod_{i=1}^{\lambda_1-1}\begin{bmatrix} r-1 \\ \beta_i \end{bmatrix}_q q^{\binom{\beta_i}{2}}\right]\begin{bmatrix} r \\ \beta_{\lambda_1} \end{bmatrix}_q q^{\binom{\beta_{\lambda_1}-1}{2}},
\end{aligned}
$$

since $\begin{bmatrix} n \\ k \end{bmatrix}_q = q^k \begin{bmatrix} n-1 \\ k \end{bmatrix}_q + \begin{bmatrix} n-1 \\ k-1 \end{bmatrix}_q$. So, we now have the claim

$$
f_{\lambda,r-1} = \sum_{S\subseteq\{2,\ldots,\lambda_1-1\}} q^{n-(|S|+2)} \det\left(\begin{bmatrix} r-1+\delta_{i,\lambda_1} \\ \lambda_i' - \chi(i) - i + j \end{bmatrix}_q q^{\binom{\lambda_i'-\chi(i)-i+j-\delta_{i,\lambda_1}}{2}}\right)_{1\le i,j\le\lambda_1}.
$$

The sets S now never contain λ_1, and the last row of each matrix has r in the upper index of the q-binomial coefficient. Continue this matching process for indices $\lambda_1 - 1, \lambda_1 - 2, \ldots, 2$, until finally, first noting that $\det A = \det A^T$ and multiplying

row i by $q^{\lambda'_i-1}$ multiplies the determinant by the same factor, we have:

$$\begin{aligned}
&q^{n-\lambda_1} \det\left(\begin{bmatrix}\lambda_1\\ \lambda'_i-i+j\end{bmatrix}_q q^{\binom{\lambda'_i-i+j-1}{2}}\right)_{1\le i,j\le\lambda_1}\\
&= q^{\sum_{i=1}^{\lambda_1}\lambda'_i-1} \det\left(\begin{bmatrix}\lambda_1\\ \lambda'_i-i+j\end{bmatrix}_q q^{\binom{\lambda'_i-i+j-1}{2}}\right)_{1\le i,j\le\lambda_1}\\
&= \det\left(\begin{bmatrix}\lambda_1\\ \lambda'_i-i+j\end{bmatrix}_q q^{\binom{\lambda'_i-i+j-1}{2}+(\lambda'_i-1)}\right)_{1\le i,j\le\lambda_1}\\
&= \det\left(\begin{bmatrix}\lambda_1\\ \lambda'_i-i+j\end{bmatrix}_q q^{\binom{\lambda'_i-i+j}{2}+i-j}\right)_{1\le i,j\le\lambda_1}\\
&= q^{\binom{\lambda_1+1}{2}-\binom{\lambda_1+1}{2}} \det\left(\begin{bmatrix}\lambda_1\\ \lambda'_i-i+j\end{bmatrix}_q q^{\binom{\lambda'_i-i+j}{2}}\right)_{1\le i,j\le\lambda_1}\\
&= \det\left(\begin{bmatrix}\lambda_1\\ \lambda'_i-i+j\end{bmatrix}_q q^{\binom{\lambda'_i-i+j}{2}}\right)_{1\le i,j\le\lambda_1}\\
&= f_{\lambda,r-1}.
\end{aligned}$$

□

2.2 Symmetries

It is no coincidence that the proofs for maximum and minimum descents are incredibly similar. In fact, if we consider the set of tableaux of shape λ with k descents, we may simply conjugate the Young diagrams to obtain a set of Young tableaux with a conjugate number of descents. This yields the following relation:

Theorem 9 *Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r) \vdash n$. Let M be the sum of the smallest major index and largest major index among all* $\mathrm{SYT}(\lambda, k)$. *Then*

$$f_{\lambda,k} = q^{\binom{n}{2}-M} f_{\lambda',n-k-1}. \tag{2.1}$$

Equivalently, letting $k = (r-1) + i$:

$$f_{\lambda,(r-1)+i} = q^{\binom{n}{2}-M} f_{\lambda',(n-r)-i}. \tag{2.2}$$

Proof. Let $\tau \in \mathrm{SYT}(\lambda, k)$ with descent set $\mathrm{Des}(\tau) = \{d_1, \ldots, d_k\}$. Consider the placement of box $d_i + 1$ in the Young diagram of τ for $1 \leq i \leq k$. Since d_i is a descent, $d_i + 1$ must appear in a row lower than d_i, and either in the same column or one to the left of d_i. In the conjugate tableau τ', the box $d_i + 1$ will now be in the same row or above d_i, in a column to the right. Hence, d_i is not a descent in the conjugate tableau. Similarly, a box which isn't a descent in τ will become a descent in τ'. Further, $\mathrm{maj}(\tau') = \binom{n}{2} - \mathrm{maj}(\tau')$, since $\mathrm{Des}(\tau') = \{1, 2, \ldots, n-1\} \backslash \mathrm{Des}(\tau)$. $\square$

While writing down M explicitly can be tedious for most λ and k, the case of minimum (and maximal) descent fillings of λ and its conjugate is straightforward, allowing us to conclude the theorem at the start of the chapter without a lengthy proof. To do so, we rely on the following lemma:

Lemma 1 *Let $\lambda = (\lambda_1, \ldots, \lambda_r)$. Among $\mathrm{SYT}(\lambda, r-1)$, a tableau with maximal major index τ_{min} is constructed by filling $\{1, 2, \ldots, \lambda_1\}$ across the first row, $\{\lambda_1+1, \ldots, \lambda_1+\lambda_2\}$ across the second row, and so on. Further, $\mathrm{maj}(\tau_{min}) = \sum_{i=1}^{r}(r-i)\lambda_i$.*

Proof. Let $\tau \in \mathrm{SYT}(\lambda, r-1)$ be the described tableau. Each thread $T_i(\tau)$ of τ is contained entirely to row i for $1 \leq i \leq r$. Suppose a tableau μ of the same shape with larger major index and the same number of descents exists. Then one thread $T_i(\mu)$ must be longer, in which case $T_i(\mu)$ extends to a row above row i. However, since thread $T_i(\mu)$ begins in the first column of row i, this creates a gap, which must be filled by the next thread, and so on, eventually leaving an unfilled collection of boxes along the outer edge of the tableau. This may only be filled by a new thread, $T_{r+1}(\mu)$, but then the tableau would have r descents. Hence, τ is has the maximum major index among tableau of the same shape and minimal of descents, and has major index

$$\begin{aligned}\mathrm{maj}(\tau) &= \lambda_1 + (\lambda_1+\lambda_2) + (\lambda_1+\lambda_2+\lambda_3) + \cdots + (\lambda_1+\cdots+\lambda_{r-1}) \\ &= (r-1)\lambda_1 + (r-2)\lambda_2 + \cdots + \lambda_{r-1} \\ &= \sum_{i=1}^{r}(r-i)\lambda_i.\end{aligned}$$

□

Theorem 10 *Let* $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r) \vdash n$. *Then*

$$f_{\lambda,n-\lambda_1} = q^{\binom{n+1}{2}-n\lambda_1} f_{\lambda',\lambda_1-1}, \tag{2.3}$$

$$f_{\lambda,r-1} = q^{nr-\binom{n+1}{2}} f_{\lambda',n-r}. \tag{2.4}$$

Proof. We know that $f_{\lambda,n-\lambda_1}$ and f_{λ',λ_1-1} are equivalent up to a shift by a power of q, so we only need to find the the minimum degree of both polynomials.

Consider the filling τ of λ with minimal descents and minimal major index. The major index of τ is simply $\sum_{i=1}^{\lambda_1} \binom{\lambda'_i}{2} = \sum_{i=1}^{r}(i-1)\lambda_i$, given by the power of q in Stanley's formula for the distribution of major index over SYT(λ). The minimum major index of a filling τ_2 of λ with maximal number of descents can be found by conjugating τ_2: τ_2' is a tableau with maximal number of descents and maximal major index, given by the previous lemma. That is, τ_2 has entries $\{1, 2, \ldots, \lambda'_1\}$ down the first column, $\{\lambda'_1 + 1, \ldots, \lambda'_1 + \lambda'_2\}$ down the second column, and so on. Every entry counts towards the major index except the last entry in each column, so $\mathrm{maj}(\gamma) = \binom{n+1}{2} - \sum_{i=1}^{\lambda_1}\sum_{j=1}^{i}\lambda'_j$. Algebraically, we simplify the shift from $f_{\lambda,n-\lambda_1}$ to f_{λ',λ_1-1} as

$$\begin{aligned}\left(\binom{n+1}{2} - \sum_{i=1}^{\lambda_1}\sum_{j=1}^{i}\lambda'_j\right) - \sum_{i=1}^{\lambda_1}(i-1)\lambda'_i &= \binom{n+1}{2} - \left(\sum_{i=1}^{\lambda_1}(\sum_{j=1}^{i}\lambda'_j) + (i-1)\lambda'_i\right) \\ &= \binom{n+1}{2} - \sum_{i=1}^{\lambda_1}\lambda_1 \cdot \lambda'_i \\ &= \binom{n+1}{2} - n\lambda_1.\end{aligned}$$

Similarly, we write the shift from $f_{\lambda,r-1}$ to $f_{\lambda',n-r}$ as

$$\sum_{i=1}^{r}(i-1)\lambda_i - \left(\binom{n+1}{2} - \sum_{i=1}^{r}\sum_{j=1}^{i}\lambda_j\right) = nr - \binom{n+1}{2}.$$

□

Consider now the partition $\lambda = (\lambda_1, \ldots, \lambda_r)$ placed at the top-left corner of the $r \times (\lambda_1 + 1)$ box. The complement of λ in this box forms a second partition, $\rho = (\lambda_1 + 1 - \lambda_r, \lambda_1 + 1 - \lambda_{r-1}, \ldots, \lambda_1 + 1 - \lambda_2, 1)$, of size $|\rho| = r(\lambda_1 + 1) - |\lambda|$. While these two partition shapes may be wildly different, their generating functions for major index over minimal descents are related.

Theorem 8 *Let* $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r)$ *and* $\rho = (\lambda_1 + 1 - \lambda_r, \lambda_1 + 1 - \lambda_{r-1}, \ldots, \lambda_1 + 1 - \lambda_2, 1)$. *Then*

$$f_{\lambda,r-1} = q^{(r-1)|\lambda| - \binom{r}{2}(\lambda_1+1)} f_{\rho,r-1}.$$

Proof. The conjugate partitions corresponding to λ and ρ are

$$\lambda' = (\lambda'_1, \ldots, \lambda'_{\lambda_1}) \text{ and}$$

$$\rho' = (\rho'_1, \ldots, \rho'_{\rho_1}) = (r, r - \lambda'_{\lambda_1}, r - \lambda'_{\lambda_1 - 1}, \ldots, r - \lambda'_{\lambda_r + 1}).$$

Applying the determinant forms for each generating function, we have

$$\begin{aligned} f_{\lambda,r-1} &= \det\left(\begin{bmatrix} r \\ \lambda'_i - i + j \end{bmatrix}_q q^{\binom{\lambda'_i - i + j}{2}}\right)_{1\le i,j\le \lambda_1}, \\ f_{\rho,r-1} &= \det\left(\begin{bmatrix} r \\ \rho'_i - i + j \end{bmatrix}_q q^{\binom{\rho'_i - i + j}{2}}\right)_{1\le i,j\le \rho_1} \\ &= \det\left(\begin{bmatrix} r \\ r - \lambda'_{\lambda_1 - i + 2} - i + j \end{bmatrix}_q q^{\binom{r - \lambda'_{\lambda_1 - i + 2} - i + j}{2}}\right)_{1\le i,j\le \lambda_1 - \lambda_r + 1}. \end{aligned}$$

For the first generating function, we reverse the order of rows and columns in the matrix, sending entry (i, j) to position $(\lambda_1 - i + 1, \lambda_1 - j + 1)$ for $1 \le i, j \le \lambda_1$. Note that while swapping the position of two rows or two columns of a matrix inverts the sign of the determinant, performing this for all rows and columns is an even number of operations. Hence, the sign of the resulting determinant is unchanged and we have

$$f_{\lambda,r-1} = \det\left(\begin{bmatrix} r \\ \lambda'_{\lambda_1 - i + 1} + i - j \end{bmatrix}_q q^{\binom{\lambda'_{\lambda_1 - i + 1} + i - j}{2}}\right)_{1\le i,j\le \lambda_1}$$

The diagonal entry is $q^{\binom{r}{2}}$ when $\lambda'_{\lambda_1 - i + 1} = r$. This happens for indices $\lambda_1 - \lambda_r + 1 \le i \le \lambda_1$, i.e. in the rows corresponding to $\lambda'_1, \lambda'_2, \ldots, \lambda'_{\lambda_r}$. The entries to left of the diagonal in these rows, then, is zero. Evaluating the determinant by minors, we have

$$f_{\lambda,r-1} = q^{\binom{r}{2}\lambda_r} \det\left(\begin{bmatrix} r \\ \lambda'_{\lambda_1 - i + 1} + i - j \end{bmatrix}_q q^{\binom{\lambda'_{\lambda_1 - i + 1} + i - j}{2}}\right)_{1\le i,j\le \lambda_1 - \lambda_r}.$$

Now, consider the generating function for $f_{\rho,r-1}$. In the first row of the corresponding matrix, $\rho'_1 = r$, so every entry is 0 except for the one in the first column, which is

$q^{\binom{r}{2}}$. Hence, we have

$$\begin{aligned} f_{\rho,r-1} &= q^{\binom{r}{2}} \det\left(\begin{bmatrix} r \\ r-\lambda'_{\lambda_1-i+1}-i+j \end{bmatrix}_q q^{\binom{r-\lambda'_{\lambda_1-i+1}-i+j}{2}} \right)_{1\le i,j\le \lambda_1-\lambda_r} \\ &= q^{\binom{r}{2}} \det\left(\begin{bmatrix} r \\ \lambda'_{\lambda_1-i+1}-i+j \end{bmatrix}_q q^{\binom{r-\lambda'_{\lambda_1-i+1}-i+j}{2}} \right)_{1\le i,j\le \lambda_1-\lambda_r}. \end{aligned}$$

To show the two determinants are equivalent up to a shift, we only need to shift each entry of the matrix corresponding to $f_{\rho,r-1}$ the appropriate amount. Expanding the power of q in each entry of the matrices, we see that the entries in column i of the matrix corresponding to ρ need to be multiplied by $q^{\frac{1}{2}(r-1)(2(i+\lambda'_{\lambda_1-i+1})-r)}$, and those in row j by $q^{(1-r)j}$. Multiplying any row or column of a matrix by a scalar multiplies the determinant by that scalar, so the determinants are equal up to a shift by the following power of q:

$$\begin{aligned} &\sum_{i=1}^{\lambda_1-\lambda_r} \big[\frac{1}{2}(r-1)(2(i+\lambda'_{\lambda_1-i+1})-r)-(r-1)i\big] + \binom{r}{2}(\lambda_r-1) \\ &\quad = \frac{1}{2}(r-1)\sum_{i=1}^{\lambda_1-\lambda_r}\big[2\lambda'_{\lambda_1-i+1}-r\big] + \binom{r}{2}(\lambda_r-1) \\ &\quad = (r-1)\big[\lambda'_{\lambda_1}+\cdots+\lambda'_{\lambda_r+1}\big] - \frac{1}{2}r(r-1)(\lambda_1-\lambda_r) + \binom{r}{2}(\lambda_r-1) \\ &\quad = (r-1)(|\lambda|-r\lambda_r) + \binom{r}{2}(2\lambda_r-\lambda_1-1) \\ &\quad = (r-1)|\lambda| - \binom{r}{2}(\lambda_1+1). \end{aligned}$$

□

Note, then, that these two generating functions are equal if and only if $|\lambda| = \frac{r(\lambda_1+1)}{2}$: if each partition fills exactly one half of the $r \times (\lambda_1 + 1)$ box. It would be interesting to find a combinatorial explanation for this relation. Consider the final thread in each partition shape. In a tableau of shape λ, this thread is required to traverse the last row, then may continue on to any of the higher rows. For the i^{th} row, we have a number of choices: we may skip the row, or continue the thread on that row in any of $\lambda_i - \lambda_{i+1}$ places. Hence, we have a total of $\sum_{i=1}^{r-1} \lambda_i - \lambda_{i+1} + 1 = \lambda_1 - \lambda_r + r - 1$ choices. For a tableau of shape ρ, we similarly have a total of $\sum_{i=1}^{r-1} \rho_i - \rho_{i+1} + 1 = \sum_{i=1}^{r-1}(\lambda_r + 1 - \lambda_{r-i+1} - (\lambda_r + 1 - \lambda_{r-i}) + 1) = \sum_{i=1}^{r-1} \lambda_{r-i} - \lambda_{r-i+1} + 1 = \lambda_1 - \lambda_r + r - 1$ choices for the last thread. To prove this relation combinatorially, one would need to show that these choices (for all threads) are equinumerous and that they have the same distribution with respect to the major index. Now, consider the shift in the relation. The first term in the exponent of the shift, $(r-1)|\lambda|$, could be considered as an "impossible" maximal major index, wherein all $r-1$ descents in a tableau of shape λ are in the largest box. The second term, $\binom{r}{2}(\lambda_1 + 1)$, is the major index of the one tableau that fills the $r \times (\lambda_1 + 1)$ box.

2.3 Minimum-plus-one descents

2.3.1 Three rowed tableaux

For general three-rowed partitions $\lambda = (\lambda_1, \lambda_2, \lambda_3) \vdash N$, finding a suitable closed formula for $f_{\lambda,i}$ for $i > 2$ can become quite tedious. Here we provide a combinatorial argument yielding a closed form for $i = 3$, after proving a lemma used within.

Lemma 2 *Let $A, B, C \in \mathbb{N}$. Then*

$$\sum_{i=0}^{A} q^{B-C+i} \begin{bmatrix} B+i \\ C \end{bmatrix}_q = \begin{bmatrix} A+B+1 \\ C+1 \end{bmatrix}_q - \begin{bmatrix} B \\ C+1 \end{bmatrix}_q. \tag{2.5}$$

Proof. We interpret the summation combinatorially. The left-hand side gives the partitions in the $(B-C+i)\times C$ box with an extra strip of shape $1\times(B-C+i)$ appended to each. Ranging over i, this indexes the partitions in the $(A+B-C)\times(C+1)$ box by the size of their first part. The first parts considered are those of lengths $B-C$ to $A+B-C$, so we have all partitions in that box except those with first part of size $B-(C+1)$ or lesser, which is precisely given by the right-hand side. □

Theorem 6 *Let $\lambda = (n, k, j) \vdash N$. Then we have*

$$\begin{aligned} f_{\lambda,3} = q^{-1}\big(f_{(n+1,k+1),3} - f_{(n+1,j),3} + f_{(k,j),3}\big) + q^{6j-8} f_{(n-j+2,k-j+2),3} \\ + \sum_{i=1}^{j-1} q^{6i-9}\big(f_{(n-i+3,k-i+3),3} - f_{(n-i+3,j-i+2),3} + f_{(k-i+2,j-i+2),3}\big) \\ - \begin{bmatrix} j-1 \\ 1 \end{bmatrix}_q \sum_{\alpha=(k,j)}^{(n,k)} q^{2\alpha_1+3\alpha_2+1} \begin{bmatrix} \alpha_1 - \alpha_2 + 1 \\ 1 \end{bmatrix}_q . \end{aligned}$$

Proof. Let $\tau \in \mathrm{SYT}(\lambda, 3)$. We will consider the three distinct configurations of the last thread, $T_4(\tau)$. Let $f^i_{\lambda,3}$ be the major index over descent polynomial for tableaux of shape λ with 3 descents wherein the last thread has one of three following behaviors:

1. $f^1_{\lambda,3}$: the last thread covers the entire last row;
2. $f^2_{\lambda,3}$: the last thread does not contribute to the last row;
3. $f^3_{\lambda,3}$: the last thread contributes to a portion of the last row.

Note that since the last thread has to exhibit exactly one of these behaviors, we have $f_{\lambda,3} = f^1_{\lambda,3} + f^2_{\lambda,3} + f^3_{\lambda,3}$.

1. $|T^3_4(\tau)| = j$

   If the last thread has length j in the last row, then it covers the row in its entirety. $T_4(\tau)$ may also cover up to $k - j$ boxes in the second row and up to $n - k$ in the first row, as illustrated:

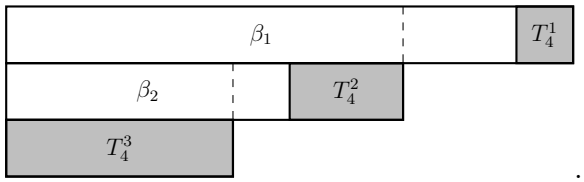

Figure 2.2: A tableau whose last thread covers the entire last row.

Remove this thread, and write the remaining partition shape after this removal as $\beta = (\beta_1, \beta_2)$, so then $(k, j) \subseteq \beta \subseteq (n, k)$. Removing a thread T_4 that yields the partition shape β removes a single descent at position $|\beta|$ from the tableau, so we have the following summation for the tableaux in this case:

$$f^1_{\lambda,3} = \sum_{(k,j)\subseteq\beta\subseteq(n,k)} q^{|\beta|} f_{\beta,2}.$$

Applying the major index over descent formula for two-rowed tableaux from [10] and reindexing the sum with $\alpha_1 = \beta_1 - k$ and $\alpha_2 = \beta_2 - j$, we have

$$\begin{aligned} f^1_{\lambda,3} &= \sum_{\beta_1=k}^{n} \sum_{\beta_2=j}^{k} q^{\beta_1+\beta_2+4} \left(\begin{bmatrix} \beta_1 \\ 2 \end{bmatrix}_q \begin{bmatrix} \beta_2 \\ 2 \end{bmatrix}_q - \begin{bmatrix} \beta_1 + 1 \\ 2 \end{bmatrix}_q \begin{bmatrix} \beta_2 - 1 \\ 2 \end{bmatrix}_q \right) \\ &= q^{k+j+4} \sum_{\alpha_1=0}^{n-k} \sum_{\alpha_2=0}^{k-j} q^{\alpha_1+\alpha_2} \left(\begin{bmatrix} k + \alpha_1 \\ 2 \end{bmatrix}_q \begin{bmatrix} j + \alpha_2 \\ 2 \end{bmatrix}_q \right. \\ &\quad \left. - \begin{bmatrix} (k+1) + \alpha_1 \\ 2 \end{bmatrix}_q \begin{bmatrix} (j-1) + \alpha_2 \\ 2 \end{bmatrix}_q \right). \end{aligned}$$

This sum telescopes, eliminating the first term for all but $\alpha_1 = 0$ or $\alpha_2 = k - j$

and the second for all but $\alpha_1 = n-k$ or $\alpha_2 = 0$. Taking care not to introduce extra terms, we have

$$\begin{aligned} f^1_{\lambda,3} = q^{k+j+4}\Bigg(& \begin{bmatrix} k \\ 2 \end{bmatrix}_q \sum_{i=0}^{k-j-1} q^i \begin{bmatrix} j+i \\ 2 \end{bmatrix}_q + q^{k-j} \begin{bmatrix} k \\ 2 \end{bmatrix}_q \sum_{i=0}^{n-k} q^i \begin{bmatrix} k+i \\ 2 \end{bmatrix}_q \\ & - \begin{bmatrix} j-1 \\ 2 \end{bmatrix}_q \sum_{i=0}^{n-k-1} q^i \begin{bmatrix} k+1+i \\ 2 \end{bmatrix}_q \\ & - q^{n-k} \begin{bmatrix} n+1 \\ 2 \end{bmatrix}_q \sum_{i=0}^{k-j} q^i \begin{bmatrix} (j-1)+i \\ 2 \end{bmatrix}_q \Bigg). \end{aligned}$$

After an application of the identity $\sum_{i=0}^{A} q^{B-2+i} \begin{bmatrix} B+i \\ 2 \end{bmatrix}_q = \begin{bmatrix} A+B+1 \\ 3 \end{bmatrix}_q - \begin{bmatrix} B \\ 3 \end{bmatrix}_q$ and simplifying, we have

$$\begin{aligned} f^1_{\lambda,3} = {} & q^{k+6} \begin{bmatrix} k \\ 2 \end{bmatrix}_q \left(\begin{bmatrix} n+1 \\ 3 \end{bmatrix}_q - \begin{bmatrix} j \\ 3 \end{bmatrix}_q \right) - q^{j+5} \begin{bmatrix} j-1 \\ 2 \end{bmatrix}_q \left(\begin{bmatrix} n \\ 3 \end{bmatrix}_q - \begin{bmatrix} k+1 \\ 3 \end{bmatrix}_q \right) \\ & - q^{n+7} \begin{bmatrix} n+1 \\ 2 \end{bmatrix}_q \left(\begin{bmatrix} k \\ 3 \end{bmatrix}_q - \begin{bmatrix} j-1 \\ 3 \end{bmatrix}_q \right). \end{aligned}$$

Applying the q-Pascal identity $q^{m-r}\begin{bmatrix} m-1 \\ r-1 \end{bmatrix}_q = \begin{bmatrix} m \\ r \end{bmatrix}_q - \begin{bmatrix} m-1 \\ r \end{bmatrix}_q$ thrice yields

$$\begin{aligned}
f^1_{\lambda,3} &= q^8\left(\begin{bmatrix} k+1 \\ 3 \end{bmatrix}_q - \begin{bmatrix} k \\ 3 \end{bmatrix}_q\right)\left(\begin{bmatrix} n+1 \\ 3 \end{bmatrix}_q - \begin{bmatrix} j \\ 3 \end{bmatrix}_q\right) \\
&\quad - q^8\left(\begin{bmatrix} j \\ 3 \end{bmatrix}_q - \begin{bmatrix} j-1 \\ 3 \end{bmatrix}_q\right)\left(\begin{bmatrix} n+1 \\ 3 \end{bmatrix}_q - \begin{bmatrix} k+1 \\ 3 \end{bmatrix}_q\right) \\
&\quad - q^8\left(\begin{bmatrix} n+2 \\ 3 \end{bmatrix}_q - \begin{bmatrix} n+1 \\ 3 \end{bmatrix}_q\right)\left(\begin{bmatrix} k \\ 3 \end{bmatrix}_q - \begin{bmatrix} j-1 \\ 3 \end{bmatrix}_q\right) \\
&= q^8\left(\begin{bmatrix} n+1 \\ 3 \end{bmatrix}_q\begin{bmatrix} k+1 \\ 3 \end{bmatrix}_q - \begin{bmatrix} n+2 \\ 3 \end{bmatrix}_q\begin{bmatrix} k \\ 3 \end{bmatrix}_q - \begin{bmatrix} n+1 \\ 3 \end{bmatrix}_q\begin{bmatrix} j \\ 3 \end{bmatrix}_q\right. \\
&\quad \left. + \begin{bmatrix} n+2 \\ 3 \end{bmatrix}_q\begin{bmatrix} j-1 \\ 3 \end{bmatrix}_q + \begin{bmatrix} k \\ 3 \end{bmatrix}_q\begin{bmatrix} j \\ 3 \end{bmatrix}_q - \begin{bmatrix} k+1 \\ 3 \end{bmatrix}_q\begin{bmatrix} j-1 \\ 3 \end{bmatrix}_q\right) \\
&= q^{-1}\big(f_{(n+1,k+1),3} - f_{(n+1,j),3} + f_{(k,j),3}\big).
\end{aligned}$$

2. $|T^3_4(\tau)| = 0$.

   Because there must be a descent from $T_3(\tau)$ to $T_4(\tau)$, if the last thread covers none of the last row it must start in the second row. As $T_3(\tau)$ must cover the entire last row of the tableau, we have $1 \leq |T^2_4| \leq k - j$. Further, $0 \leq |T^1_4| \leq n - k$, since T_4 covers the rightmost boxes in row 2. We may remove this entire strip. Note, however, that the last box of T_3 is fixed to be in the rightmost box of row one after removal of T_4, so we may also remove this box. This leaves tableaux of shape $(k-1, j, j) \subseteq \beta \subseteq (n-1, k-1, j)$ across all τ, with major index $|\beta| + 1$ less than their respective τ. Hence, the tableaux in this case are

given by

$$f^2_{\lambda,3} = \sum_{(k-1,j,j)\subseteq\beta\subseteq(n-1,k-1,j)} q^{|\beta|+1} f_{\beta,2}.$$

Applying the minimum descent formula for tableaux with three rows, we have

$$\begin{aligned} f^2_{\lambda,3} &= \sum_{\beta} q^{|\beta|+1+3\beta_3} \det\left(\begin{bmatrix} \beta_s - \beta_3 - s + t + 2 \\ 2 \end{bmatrix}_q \right)_{1\le s,t\le 3} \\ &= q^{4j+1} \sum_{\beta_1=k-1}^{n-1} \sum_{\beta_2=j}^{k-1} q^{\beta_1+\beta_2} \left(\begin{bmatrix} \beta_1 - j + 2 \\ 2 \end{bmatrix}_q \begin{bmatrix} \beta_2 - j + 2 \\ 2 \end{bmatrix}_q \right. \\ &\quad \left. - \begin{bmatrix} \beta_1 - j + 3 \\ 2 \end{bmatrix}_q \begin{bmatrix} \beta_2 - j + 1 \\ 2 \end{bmatrix}_q \right). \end{aligned}$$

Now, we fix β_1 and evaluate each of terms with respect to the β_2 sum. For instance, notice that $\sum_{\beta_2=j}^{k-1} q^{\beta_2-j} \begin{bmatrix} \beta_2-j+2 \\ 3 \end{bmatrix}_q$ indexes the partitions in the $3 \times (k-1-j)$ box by the size of their first part, yielding $\begin{bmatrix} k-j+2 \\ 3 \end{bmatrix}_q$. Repeating this for the other term gives

$$\begin{aligned} & q^{5j+1+\beta_1} \left(\begin{bmatrix} \beta_1 - j + 2 \\ 2 \end{bmatrix}_q \sum_{\beta_2=j}^{k-1} q^{\beta_2-j} \begin{bmatrix} \beta_2 - j + 2 \\ 2 \end{bmatrix}_q \right. \\ & \quad \left. - q \begin{bmatrix} \beta_1 - j + 3 \\ 2 \end{bmatrix}_q \sum_{\beta_2=j}^{k-1} q^{\beta_2-j-1} \begin{bmatrix} \beta_2 - j + 1 \\ 2 \end{bmatrix}_q \right) \\ & = q^{5j+1+\beta_1} \left(\begin{bmatrix} \beta_1 - j + 2 \\ 2 \end{bmatrix}_q \begin{bmatrix} k - j + 2 \\ 3 \end{bmatrix}_q - q \begin{bmatrix} \beta_1 - j + 3 \\ 2 \end{bmatrix}_q \begin{bmatrix} k - j + 1 \\ 3 \end{bmatrix}_q \right). \end{aligned}$$

With this, we close the β_1 sum by repeating this interpretation then simplifying:

$$\begin{aligned}
f^2_{\lambda,3} &= q^{6j+1} \sum_{\beta_1=k-1}^{n-1} \Bigg(q^{\beta_1-j} \begin{bmatrix} k-j+2 \\ 3 \end{bmatrix}_q \begin{bmatrix} \beta_1-j+2 \\ 2 \end{bmatrix}_q \\
&\quad - q^{\beta_1-j+1} \begin{bmatrix} k-j+1 \\ 3 \end{bmatrix}_q \begin{bmatrix} \beta_1-j+3 \\ 2 \end{bmatrix}_q \Bigg) \\
&= q^{6j+1} \Bigg(\begin{bmatrix} k-j+2 \\ 3 \end{bmatrix}_q \Bigg(\begin{bmatrix} n-j+2 \\ 3 \end{bmatrix}_q - \begin{bmatrix} k-j+1 \\ 3 \end{bmatrix}_q \Bigg) \\
&\quad - \begin{bmatrix} k-j+1 \\ 3 \end{bmatrix}_q \Bigg(\begin{bmatrix} n-j+3 \\ 3 \end{bmatrix}_q - \begin{bmatrix} k-j+2 \\ 3 \end{bmatrix}_q \Bigg) \Bigg) \\
&= q^{6j+1} \Bigg(\begin{bmatrix} n-j+2 \\ 3 \end{bmatrix}_q \begin{bmatrix} k-j+2 \\ 3 \end{bmatrix}_q - \begin{bmatrix} n-j+3 \\ 3 \end{bmatrix}_q \begin{bmatrix} k-j+1 \\ 3 \end{bmatrix}_q \Bigg) \\
&= q f_{(n-1,k-1,j,j),3} \\
&= q^{6j-8} f_{(n-j+2,k-j+2),3}.
\end{aligned}$$

If $k = j$, then T_4 is forced to begin on the third row for the tableau to have three descents, hence this term would be zero. We note that it's rather interesting that the distribution of the major index of all tableaux in this case is equivalent (up to a shift by a power of q) to the distribution of a single partition shape.

3. $0 < |T_4^3(\tau)| < j$:

   If the last thread starts in the last row in a column other than the first, it can have length at most $k - j$ in the second row and at most $n - k$ in the first row. Removing this thread, the resulting partition shape (for all τ) is among $(k, j, 1) \subseteq \beta \subseteq (n, k, j - 1)$. However, the tableaux of this shape will form a subset of $\mathrm{SYT}(\beta, 2)$: in particular, those whose last thread, $T_3(\tau)$, has some

entries in the first or second row. In other words, we can write these as the tableaux of shape β except for those whose last thread stays in the third row. Call this exceptional set of tableaux S^*_β. Thus, we have

$$f^3_{\lambda,3} = \sum_{(k,j,1)\subseteq\beta\subseteq(n,k,j-1)} q^{|\beta|} f_{\beta,2} - q^{|\beta|} \sum_{\kappa\in S^*_\beta} q^{\mathrm{maj}(\kappa)}.$$

The major index over descent polynomial in S^*_β can be rewritten, however. Since the third thread in β is confined to the last row, we may remove it, removing $\beta_1 + \beta_2$ from the major index. This leaves the two-rowed tableau (β_1, β_2), with a single descent and no further restrictions on the placement of either thread. Hence, we have

$$f^3_{\lambda,3} = \sum_{(k,j,1)\subseteq\beta\subseteq(n,k,j-1)} q^{|\beta|} f_{\beta,2} - q^{|\beta|+\beta_1+\beta_2} f_{(\beta_1,\beta_2),1}.$$

The polynomial $f_{(\beta_1,\beta_2),1}$ can be constructed as follows, for fixed $(k, j) \subseteq (\beta_1, \beta_2) \subseteq (n, k)$. There is a single descent, so it must occur in the first row. Hence, the first thread exists only on the first row and the second thread must span the entirety of the second row. The second thread may fill the remaining portion of the first row, but starting only from columns $\beta_2 + 1$ to β_1. Hence, the major index will be entirely determined by the position of the last entry in the first thread in the first row, between columns β_2 and β_1. That is,

$$\sum_{(k,j,1)\subseteq\beta\subseteq(n,k,j-1)} q^{|\beta|+\beta_1+\beta_2} f_{(\beta_1,\beta_2),1} = \sum_{(k,j,1)\subseteq\beta\subseteq(n,k,j-1)} q^{|\beta|+\beta_1+\beta_2}(q^{\beta_2}+\cdots+q^{\beta_1})$$
$$= \sum_{(k,j,1)\subseteq\beta\subseteq(n,k,j-1)} q^{2\beta_1+3\beta_2+\beta_3} \begin{bmatrix} \beta_1-\beta_2+1 \\ 1 \end{bmatrix}_q .$$

Closing the sum over β_3 yields

$$\sum_{(k,j,1)\subseteq\beta\subseteq(n,k,j-1)} q^{2\beta_1+3\beta_2+\beta_3} \begin{bmatrix} \beta_1-\beta_2+1 \\ 1 \end{bmatrix}_q$$
$$= \begin{bmatrix} j-1 \\ 1 \end{bmatrix}_q \sum_{\beta_1=k}^{n} \sum_{\beta_2=j}^{k} q^{2\beta_1+3\beta_2+1} \begin{bmatrix} \beta_1-\beta_2+1 \\ 1 \end{bmatrix}_q .$$

We then apply the q-Pascal identity $q^A \begin{bmatrix} A+B-1 \\ B-1 \end{bmatrix}_q = \begin{bmatrix} A+B \\ B \end{bmatrix}_q - \begin{bmatrix} A+B-1 \\ B \end{bmatrix}_q$ and the summation identity $\sum_{R=0}^{j} q^{iR} \begin{bmatrix} A-R-1 \\ i-1 \end{bmatrix}_q = \begin{bmatrix} A \\ i \end{bmatrix}_q - q^{i(j+1)} \begin{bmatrix} A-j-1 \\ i \end{bmatrix}_q$, then cancel like terms to obtain:

$$\begin{bmatrix} j-1 \\ 1 \end{bmatrix}_q \sum_{\beta_1=k}^{n} \sum_{\beta_2=j}^{k} q^{2\beta_1+3\beta_2+1} \begin{bmatrix} \beta_1-\beta_2+1 \\ 1 \end{bmatrix}_q$$
$$= \begin{bmatrix} j-1 \\ 1 \end{bmatrix}_q \sum_{\beta_1=k}^{n} \sum_{\beta_2=j}^{k} q^{2\beta_1+3\beta_2+1} \left(\begin{bmatrix} \beta_1-\beta_2+2 \\ 2 \end{bmatrix}_q - q^2 \begin{bmatrix} \beta_1-\beta_2+1 \\ 2 \end{bmatrix}_q \right)$$
$$= \begin{bmatrix} j-1 \\ 1 \end{bmatrix}_q \sum_{\beta_1=k}^{n} q^{2\beta_1+1} \left(q^{3j} \begin{bmatrix} \beta_1-j+3 \\ 3 \end{bmatrix}_q - q^{3(k+1)} \begin{bmatrix} \beta_2-k+2 \\ 3 \end{bmatrix}_q \right.$$
$$\left. + q^{3(k+1)+2} \begin{bmatrix} \beta_1-k+1 \\ 3 \end{bmatrix}_q - q^{3j+2} \begin{bmatrix} \beta_1-j+2 \\ 3 \end{bmatrix}_q \right).$$

Applying the q-Pascal identity again and closing the sum over β_2 for each term

using the identity $\sum_{\beta_1=k}^{n} q^{\beta_1-A}\begin{bmatrix}\beta_1-A+B\\ B\end{bmatrix}_q = \begin{bmatrix}n-\beta_1+B\\ B\end{bmatrix}_q - \begin{bmatrix}k-\beta_1-1+B\\ B\end{bmatrix}_q$ yields

$$
\begin{aligned}
&\sum_{\beta_1=k}^{n}\sum_{\beta_2=j}^{k} q^{2\beta_1+3\beta_2+1}\begin{bmatrix}\beta_1-\beta_2+1\\ 1\end{bmatrix}_q \\
&= q^{5j+1}\left(\begin{bmatrix}n-j+5\\ 5\end{bmatrix}_q - q\begin{bmatrix}n-j+4\\ 5\end{bmatrix}_q - \begin{bmatrix}k-j+4\\ 5\end{bmatrix}_q + q\begin{bmatrix}k-j+3\\ 5\end{bmatrix}_q\right) \\
&\quad - q^{5j+5}\left(\begin{bmatrix}n-j+4\\ 5\end{bmatrix}_q - q\begin{bmatrix}n-j+3\\ 5\end{bmatrix}_q - \begin{bmatrix}k-j+3\\ 5\end{bmatrix}_q + q\begin{bmatrix}k-j+2\\ 5\end{bmatrix}_q\right) \\
&\quad - q^{5k+6}\left(\begin{bmatrix}n-k+4\\ 5\end{bmatrix}_q - q\begin{bmatrix}n-k+3\\ 5\end{bmatrix}_q\right) \\
&\quad + q^{5k+10}\left(\begin{bmatrix}n-k+3\\ 5\end{bmatrix}_q - q\begin{bmatrix}n-k+2\\ 5\end{bmatrix}_q\right).
\end{aligned}
$$

Finally, expand every q-binomial as a ratio of polynomials in q and simplify algebraically to obtain

$$
\begin{aligned}
&\begin{bmatrix}j-1\\ 1\end{bmatrix}_q \sum_{\beta_1=k}^{n}\sum_{\beta_2=j}^{k} q^{2\beta_1+3\beta_2+1}\begin{bmatrix}\beta_1-\beta_2+1\\ 1\end{bmatrix}_q \\
&= q^{3j+2k+1}(1+q^{n-k+1}+q^{n-j+2})\frac{(1-q^{j-1})(1-q^{k-j+1})(1-q^{n-k+1})(1-q^{n-j+2})}{(1-q)^2(1-q^2)(1-q^3)}.
\end{aligned}
$$

Consider now the first term in the sum. Expanding $f_{\beta,2}$ using the generating function for tableau with minimum number of descents, we have

$$\begin{aligned}\sum_{\beta} q^{|\beta|} f_{\beta,2} &= \sum_{\beta} q^{\beta_1+\beta_2+4\beta_3} \det\left(\begin{bmatrix}\beta_s-\beta_3-s+t+2\\ 2\end{bmatrix}_q\right)_{1\le s,t\le 3}\\ &= \sum_{\beta} q^{\beta_1+\beta_2+4\beta_3}\left(\begin{bmatrix}\beta_1-\beta_3+2\\ 2\end{bmatrix}_q\begin{bmatrix}\beta_2-\beta_3+2\\ 2\end{bmatrix}_q\right.\\ &\quad\left.-\begin{bmatrix}\beta_1-\beta_3+3\\ 2\end{bmatrix}_q\begin{bmatrix}\beta_2-\beta_3+1\\ 2\end{bmatrix}_q\right).\end{aligned}$$

Fix β_1 and β_3 and evaluate the sums with respect to β_2. Applying the same combinatorial arguments as before, we have

$$\begin{aligned}\sum_{\beta} q^{|\beta|} f_{\beta,2} &= \sum_{\beta_1,\beta_3} q^{\beta_1+5\beta_3}\begin{bmatrix}\beta_1-\beta_3+2\\ 2\end{bmatrix}_q\left(\begin{bmatrix}k-\beta_3+3\\ 3\end{bmatrix}_q-\begin{bmatrix}j-\beta_3+2\\ 3\end{bmatrix}_q\right)\\ &\quad-q^{\beta_1+5\beta_3+1}\begin{bmatrix}\beta_1-\beta_3+3\\ 2\end{bmatrix}_q\left(\begin{bmatrix}k-\beta_3+2\\ 3\end{bmatrix}_q-\begin{bmatrix}j-\beta_3+2\\ 3\end{bmatrix}_q\right).\end{aligned}$$

Now, fix β_3 and range over β_1, simplifying again to obtain

$$\begin{aligned}\sum_{\beta} q^{|\beta|} f_{\beta,2} &= \sum_{\beta_3} q^{6\beta_3}\left[\left(\begin{bmatrix}n-\beta_3+3\\ 3\end{bmatrix}_q\begin{bmatrix}k-\beta_3\\ 3\end{bmatrix}_q-\begin{bmatrix}n-\beta_3+4\\ 3\end{bmatrix}_q\begin{bmatrix}k-\beta_3+2\\ 3\end{bmatrix}_q\right)\right.\\ &\quad-\left(\begin{bmatrix}n-\beta_3+3\\ 3\end{bmatrix}_q\begin{bmatrix}j-\beta_3+2\\ 3\end{bmatrix}_q-\begin{bmatrix}n-\beta_3+4\\ 3\end{bmatrix}_q\begin{bmatrix}j-\beta_3+1\\ 3\end{bmatrix}_q\right)\\ &\quad\left.+\left(\begin{bmatrix}k-\beta_3+2\\ 3\end{bmatrix}_q\begin{bmatrix}j-\beta_3+2\\ 3\end{bmatrix}_q-\begin{bmatrix}k-\beta_3+3\\ 3\end{bmatrix}_q\begin{bmatrix}j-\beta_3+1\\ 3\end{bmatrix}_q\right)\right].\end{aligned}$$

Applying the identity for $f_{(\lambda_1,\lambda_2)\setminus(\mu_1),i}(q)$ given by Theorem 5 with $\mu_1 = 0$ yields

$$\sum_{\beta} q^{|\beta|} f_{\beta,2} = \sum_{\beta_3=1}^{j-1} q^{6\beta_3-9}(f_{(n-\beta_3+3,k-\beta_3+3),3} - f_{(n-\beta_3+3,j-\beta_3+2),3} + f_{(k-\beta_3+2,j-\beta_3+2),3}).$$

□

At this point, we have a formula for $f_{\lambda,k}$ for $k = 2, 3$. To obtain $f_{\lambda,4}$, we can extend the above argument in the same way, writing $f_{\lambda,4} = \sum_{i=1}^{4} f^i_{\lambda,4}$. The sums will now be more lengthy, involving different $f_{\mu,3}$ (for some other partition μ). There is nothing preventing us from doing this, but as the number of descents increases this process quickly becomes unwieldy. To find a reasonable closed form for general $f_{\lambda,k}$, one would hope that their length doesn't increase so drastically as k increases.

2.3.2 Rectangular tableaux

Consider the family of standard young tableaux in the shape of a *rectangular partition* $\lambda = (n, n, \ldots, n) = n^r \vdash nr$, which we call *rectangular tableaux*. Rectangular tableaux appear in combinatorics and representation theory quite often as their relatively simple shape allows for a more obtainable grasp on any underlying structures, for instance in connection to the *cyclic sieving phenomenon* ([19], [20]). Here, we

provide two different proof methods to obtain a closed formula for $f_{(n^r),r}(q)$. The first constructs a recursive argument on the tableaux in $\mathrm{SYT}(n^r, r)$ and smaller partition shapes, while the latter combinatorially identifies the tableaux with a family of partitions.

Theorem 7 *Let $\lambda = (n, \ldots, n) = n^r \vdash nr$. Then*

$$f_{n^r,r} = q^{\binom{r}{2}n}\left(\begin{bmatrix} n+r \\ r \end{bmatrix}_q - \begin{bmatrix} nr+1 \\ 1 \end{bmatrix}_q\right).$$

Proof 1. Consider a tableau $\tau \in \mathrm{SYT}(\lambda, r)$. The last thread in the tableaux, T_r, either covers the entire last row or it doesn't. If it does, we may remove this last thread entirely, shifting the major index of the tableau by $(r-1)n$. If not, then the last thread has length $1 \leq \beta_1 \leq n-1$ in the last row. If we remove this thread, then we obtain a subset of $\mathrm{SYT}(n^{r-1}(n-\beta_1)^1, r-1)$: precisely those tableaux whose last thread continues past the last row. Let β_2 be the length of the last thread in this new tableau in the penultimate row, so then $1 \leq \beta_2 \leq \beta_1$. Remove this thread as well, obtaining a tableau in $\mathrm{SYT}((n^{r-2}, n-\beta_2), r-2)$. Summing over all possible configurations of β_1, β_2, we have

$$f_{n^r,r} = q^{(r-1)n} f_{(n^{r-1}),r-1} + \sum_{\beta_1=1}^{n-1} q^{nr-\beta_1} \sum_{\beta_2=1}^{\beta_1} q^{n(r-1)-\beta_2} f_{(n^{r-2},n-\beta_2),r-2}.$$

The same procedure can be performed on all of the rectangular tableau with $r-1$

rows, $r-2$ rows, etc. Iterating on the first term, we obtain

$$\begin{aligned} f_{(n^r),r} &= q^{(r-1)n+(r-2)n+\cdots+2n} f_{(n,n),2} \\ &\quad + \sum_{\alpha=0}^{r-3} q^{\sum_{i=1}^{\alpha}(r-\alpha)n} \sum_{\beta_1=1}^{n-1} q^{(r-\alpha)n-\beta_1} \sum_{\beta_2=1}^{\beta_1} q^{(r-\alpha-1)n-\beta_2} f_{(n^{r-\alpha-2},n-\beta_2),r-\alpha-2} \\ &= q^{\binom{r}{2}n-n} f_{(n,n),2} \\ &\quad + \sum_{\alpha=0}^{r-3} \sum_{\beta_1=1}^{n-1} \sum_{\beta_2=1}^{\beta_1} q^{\binom{r}{2}n-\binom{r-\alpha}{2}n+(r-\alpha)n+(r-\alpha-1)n-\beta_1-\beta_2} f_{(n^{r-\alpha-2},n-\beta_2),r-\alpha-2}. \end{aligned}$$

We now aim to simplify $f_{n^{r-\alpha-2},r-\alpha-2}$. Let $\mu = (n, n, \ldots, n, n-\beta_2)$ be a partition with $(r-\alpha-1)$ parts, and consider $\tau \in \mathrm{SYT}(\mu, r-\alpha-2)$. Because τ has the minimal number of descents, every thread starts in the first column of the tableau. Since these are the only threads in the tableau, a thread starting in row k may only occur in rows k or $k-1$. Write the length of the kth thread in row k beyond the $(n-\beta_2)$nd column as ω_k. For all k, $0 \leq \omega_k \leq \beta_2$ and the ω_k must be weakly decreasing, so $\omega = (\omega_1, \ldots, \omega_{\beta_2})$ forms a partition (perhaps with trailing zeroes) in the $(r-\alpha-1) \times \beta_2$ box.

The tableau $\pi \in \mathrm{SYT}(\mu, r-\alpha-2)$ with minimum major index has first thread of length $n-\beta_2$ and other threads of length n, hence $\mathrm{maj}(\pi) = (n-\beta_2)+(2n-\beta_2)+\cdots+((r-\alpha-2)n-\beta_2) = \binom{r-\alpha-1}{2}n-(r-\alpha-2)\beta_2$. Note that the corresponding partition ω for π is the empty partition. For other tableaux τ, the first descent occurs in the $n-\beta_2+\omega_1$ box, the second in the $(n-\beta_2+\omega_1)+(n-\beta_2+\omega_2+(\beta_2-\omega_1)) = 2n-\beta_2+\omega_2$

box, and so on. Thus, $\mathrm{maj}(\tau) = \mathrm{maj}(\pi) + |\omega|$, so we have

$$f_{(n^{r-\alpha-2},n-\beta_2),r-\alpha-2} = q^{\binom{r-\alpha-1}{2}n-(r-\alpha-2)\beta_2} \begin{bmatrix} (r-\alpha-2)+\beta_2 \\ \beta_2 \end{bmatrix}_q .$$

Applying this to our current formulation and rearranging, we have

$$\begin{aligned} f_{(n^r),r} &= q^{\binom{r}{2}n-n} f_{(n,n),2} + \sum_{\alpha=0}^{r-3} \sum_{\beta_1=1}^{n-1} \sum_{\beta_2=1}^{\beta_1} q^{\binom{r}{2}n-\binom{r-\alpha}{2}n+(r-\alpha)n+(r-\alpha-1)n-\beta_1-\beta_2} \\ &\quad \cdot q^{\binom{r-\alpha-1}{2}n-(r-\alpha-2)\beta_2} \begin{bmatrix} (r-\alpha-2)+\beta_2 \\ \beta_2 \end{bmatrix}_q \\ &= q^{\binom{r}{2}n-n} f_{(n,n),2} + \sum_{\alpha=0}^{r-3} q^{\binom{r}{2}n-\binom{r-\alpha}{2}n+\binom{r-\alpha-1}{2}n} \\ &\quad \cdot \sum_{\beta_1=1}^{n-1} \sum_{\beta_2=1}^{\beta_1} q^{(n-\beta_1)+(r-\alpha-1)(2n-\beta_2)} \begin{bmatrix} (r-\alpha-2)+\beta_2 \\ \beta_2 \end{bmatrix}_q . \end{aligned}$$

We may close the inner double sum by proving the identity

$$\sum_{\beta_1=1}^{n-1} \sum_{\beta_2=1}^{\beta_1} q^{-\beta_1-(C-1)\beta_2} = q^{C(1-n)} \begin{bmatrix} C+(n-1) \\ n-1 \end{bmatrix}_q - q^{1-n} \begin{bmatrix} n \\ 1 \end{bmatrix}_q .$$

Consider the $C \times (n-1)$ lattice box and replace $q \to q^{-1}$ in the summation. The sum fills the first column with a strip of length β_1, then fills a rectangle of dimensions $\beta_2 \times C-1$ to its right. The q^{-1}-binomial then "eats away" a partition in the $(C-2)\times\beta_2$ box created by this filling. That is, these are the conjugates of the partitions in the $C \times (n-1)$ box with at least two parts, i.e. the partitions *not* in the $(n-1) \times 1$

box, with $q \to q^{-1}$. Replacing q and shifting the q-binomials appropriately yields the desired identity. Hence, we have

$$\begin{aligned}
f_{(n^r),r} &= q^{\binom{r}{2}n-n} f_{(n,n),2} + \sum_{\alpha=0}^{r-3} q^{\binom{r}{2}n-\binom{r-\alpha}{2}n+\binom{r-\alpha-1}{2}n+n+(r-\alpha-1)2n} \\
&\quad \cdot \left(q^{-(r-\alpha)(n-1)} \begin{bmatrix} r-\alpha+n-1 \\ n-1 \end{bmatrix}_q - q^{1-n} \begin{bmatrix} n \\ 1 \end{bmatrix}_q \right) \\
&= q^{\binom{r}{2}n-n} f_{(n,n),2} + \sum_{\alpha=0}^{r-3} q^{\binom{r}{2}n+r-\alpha} \begin{bmatrix} r-\alpha+n-1 \\ n-1 \end{bmatrix}_q - q^{\binom{r+1}{2}n-n+1-\alpha n} \begin{bmatrix} n \\ 1 \end{bmatrix}_q \\
&= q^{\binom{r}{2}n-n} f_{(n,n),2} + q^{\binom{r}{2}n} \left(\begin{bmatrix} n+r \\ r \end{bmatrix}_q - \begin{bmatrix} n+2 \\ 2 \end{bmatrix}_q \right) \\
&\quad - q^{\binom{r+1}{2}n-n+1-n(r-3)} \frac{1-q^{n(r-2)}}{1-q^n} \begin{bmatrix} n \\ 1 \end{bmatrix}_q \\
&= q^{\binom{r}{2}n} \left(q^2 \begin{bmatrix} n \\ 2 \end{bmatrix}_q + \begin{bmatrix} n+r \\ r \end{bmatrix}_q - \begin{bmatrix} n+2 \\ 2 \end{bmatrix}_q - q^{2n+1} \begin{bmatrix} n(r-2) \\ 1 \end{bmatrix}_q \right) \\
&= q^{\binom{r}{2}n} \left(\begin{bmatrix} n+r \\ r \end{bmatrix}_q - q^{2n+1} \begin{bmatrix} n(r-2) \\ 1 \end{bmatrix}_q - \begin{bmatrix} 2n+1 \\ 1 \end{bmatrix}_q \right) \\
&= q^{\binom{r}{2}n} \left(\begin{bmatrix} n+r \\ r \end{bmatrix}_q - \begin{bmatrix} nr+1 \\ 1 \end{bmatrix}_q \right).
\end{aligned}$$

□

Proof 2. We begin by identifying partitions which do not correspond to a tableau in $\mathrm{SYT}(n^r, r)$.

Let $\mu = (\mu_1, \ldots, \mu_I, \mu_{I+1}) = (n, \ldots, n, \mu_{I+1})$ for $0 \leq I \leq n-1$, with $0 \leq \mu_{I+1} \leq n-1$. Attempt to form a tableau with r descents by letting μ_i denote the length of the i^{th} thread, τ_i, in the i^{th} row. The first I parts of the tableau, then, are filled, leaving a partially filled $(I+1)$st row. The τ_{I+2} thread may not be placed in row $(I+1)$,

else it would be a part of τ_{I+1}. Further, it may not be placed at the start of row $(I+2)$, since $\mu_{I+2} = 0$. Hence, μ does not correspond to any tableau $\tau \in \mathrm{SYT}(n^r, r)$. There is one partition μ for all sizes $0 \leq i \leq nr$, so their generating function is $\begin{bmatrix} nr+q \\ 1 \end{bmatrix}_q$.

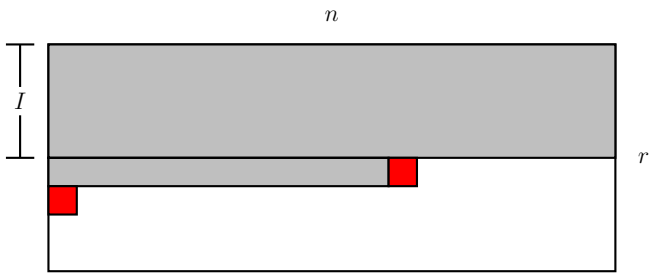

Figure 2.3: Rectangular tableau of shape (n^r) with r descents whose $(I+1)$st thread partially fills the $(I+1)$st row.

Now, consider the partitions $\mu = (\mu_1, \ldots, \mu_I, \mu_{I+1}, \ldots, \mu_{I+R}) = (n, \ldots, n, \mu_{I+1}, \ldots, \mu_{I+R})$ with $0 \leq I \leq r-2$ and $2 \leq R \leq r-I$, s.t. $1 \leq \mu_{I+m} \leq n-1$ for $1 \leq m \leq R$, precisely the partitions in the $n \times r$ box except for the previous set. For one such μ, form the first I threads, $\tau_1, \ldots, \tau_I$, across the first I rows. The thread τ_{I+1} finishes in column μ_{I+1}, as it cannot proceed to a row above. The thread τ_{I+2}, then, must fill out μ_{I+2}, then proceed to the column above. If not, then there would be a gap which could not be filled by subsequent threads, similar to the previous case. This process continues until we reach τ_{I+R}, which finishes in row $(I+R-1)$. Thus, τ_{I+R+1} is forced to cover the rest of row $(I+R)$. Subsequent

threads $\tau_{I+R+2}, \ldots, \tau_r$ span the entire row they occupy.

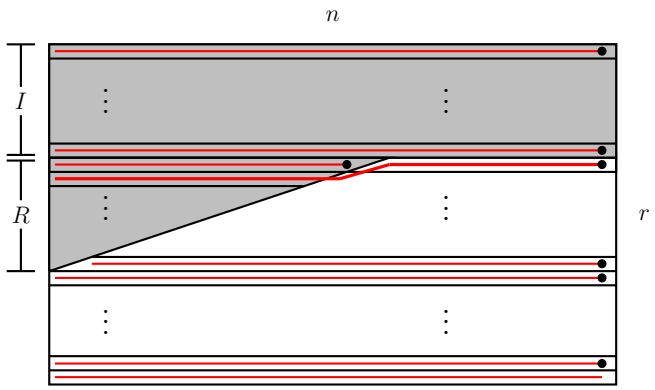

Figure 2.4: Threads of a rectangular tableau of shape (n^r) with r descents.

This identifies a tableau which has exactly r descents. The major index of this tableau is $(n+2n+\cdots+In)+[(In+\mu_{I+1})+((I+1)n+\mu_{I+2})+\cdots+(I+R-1)n+\mu_{I+R})]+[(I+R)n+\cdots+(r-1)n] = \binom{r}{2}n+|\mu|$. This process can be reversed by writing $\mu = (\mu_1, \ldots, \mu_r)$ s.t. μ_i is the length of τ_i in row i, disregarding any trailing zeroes. Hence, we can write $f_{n^r,r}(q)$ as

$$f_{n^r,r}(q) = \sum_{\mu} q^{\binom{r}{2}n+|\mu|} = q^{\binom{r}{2}n}\left(\begin{bmatrix} n+r \\ r \end{bmatrix}_q - \begin{bmatrix} nr+1 \\ 1 \end{bmatrix}_q\right).$$

□

Chapter 3

Computation through the Kirillov-Reshetikhin Formula

3.1 Admissible sequences

As discussed previously, the Kirillov-Reshetikhin formula $K^k_{\lambda,1^{|\lambda|}}(q)$ gives a way to compute $f_{\lambda,k}$ by using the charge statistic. Let λ be a partition and let $\mu = 1^{|\lambda|}$ be the partition of size $|\lambda|$ comprised of all ones. Let $\alpha = (\mu', \alpha^1, \alpha^2, \dots) = ((|\lambda|), \alpha^1, \alpha^2, \dots)$ be a sequence of partitions such that for $i \geq 1$, $|\alpha^i| = \sum_{j=i+1}^{\infty} \lambda_j$. We will write the

partitions as $\alpha^i = (\alpha_1^i, \alpha_2^i, \dots)$ for $i \geq 1$. For any such sequence α, define

$$c(\alpha) := \sum_{a,i} \frac{1}{2}(\alpha_i^{a-1} - \alpha_i^a)(\alpha_i^{a-1} - \alpha_i^a - 1).$$

For a given sequence α and any $a, i \geq 1$, define

$$P_i^a(\alpha) := \sum_{s=1}^{i} (\alpha_s^{a-1} - 2\alpha_s^a + \alpha_s^{a+1}).$$

Using the above definitions, Kirillov and Reshetikhin give the following formula in [12]:

$$K^k_{\lambda,1^{|\lambda|}}(q) = \sum_{\substack{\alpha=(\mu',\alpha^1,\alpha^2,\dots) \\ \alpha_1^1=k}} q^{c(\alpha)} \prod_{a,i} \begin{bmatrix} P_i^a(\alpha) + \alpha_i^a - \alpha_{i+1}^a \\ \alpha_i^a - \alpha_{i+1}^a \end{bmatrix}_q.$$

As discussed in the first chapter, $K_{\lambda,1^{|\lambda|}}$ is useful for our purposes because it generates a polynomial with the same distribution as the major index for tableaux in $\mathrm{SYT}(\lambda, k)$. That is,

$$f_{\lambda,k}(q^{-1}) = q^{-\binom{|\lambda|}{2}} K^k_{\lambda,1^{|\lambda|}}(q).$$

With this formula in hand and enough computational resources, one may generate formulae for any $f_{\lambda,k}$. However, as the number of parts (or the number of descents) increases, the number of potential sequences α increases exponentially faster.

Our aim in this chapter is to reduce the computational complexity for three-rowed tableaux with at most three descents, demonstrating that the current written formula for $K^k_{\lambda,1^{|\lambda|}}(q)$ has potential to be simplified in the future for larger partition shapes and descent numbers. This may be done by reducing the number of sequences the formula's summation ranges over, considering only those which have nonzero contribution to the total sum. That is, we desire to find the *admissible sequences* α for given λ and k. Call this set A^λ_k.

Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r)$. The set of admissible sequences A^λ_k for $k < (r-1)$ or $k > |\lambda| - \lambda_1$ is clearly empty, as there are no standard Young tableaux of shape λ with that many descents. For $r = 3$ and $k = 2$, we have the following:

Theorem 11 *Let $\lambda = (n, k, j)$ and $k = 2$. Then*

$$A_2^{(n,k,j)} = \{((N), (2^{j+m}, 1^{k-j-2m}), (1^j)) : 0 \leq m \leq \frac{k-j}{2}\}.$$

Proof. We first show that all admissible sequences α have $\alpha^2 = (1^j)$.

By definition, $\alpha^1 = (2, \ldots) \vdash (k+j)$ and $\alpha^2 \vdash j$. Suppose that α^2 has more than one part, and fix i such that α^2_i is the last such part. For each sequence α, its contribution to the charge polynomial $K^2_{\lambda,1^{|\lambda|}}(q)$ is zero if the product given by that sequence is

zero. Consider the terms in the product:

$$\begin{bmatrix} P_i^2(\alpha) + \alpha_i^2 - \alpha_{i+1}^2 \\ \alpha_i^2 - \alpha_{i+1}^2 \end{bmatrix}_q = \begin{bmatrix} P_i^2(\alpha) + \alpha_i^2 - 1 \\ \alpha_i^2 - 1 \end{bmatrix}_q.$$

Now, compute the maximum value of $P_i^2(\alpha)$. Note that up to index i, the maximum value of all α_m^1 is 2, the minimum value of all α_m^2 is α_i^2, and α_m^3 is zero as $\alpha^3 \vdash 0$. So,

$$\begin{aligned} P_i^2(\alpha) &= \sum_{m=1}^{i} (\alpha_m^1 - 2\alpha_m^2 + \alpha_m^3) \\ &= \sum_{m=1}^{i} (\alpha_m^1 - 2\alpha_m^2) \\ &\leq \sum_{m=1}^{i} (2 - 2\alpha_i^2) \\ &= 2i(1 - \alpha_i^2) < 0. \end{aligned}$$

Hence, any sequence α with $\alpha^2 \neq (1^j)$ contributes nothing to $K^2_{\lambda,1^{|\lambda|}}(q)$. Suppose now that $\alpha^2 = (1^j)$, but α^1 contains $j - y$ parts of size 2 for $y \geq 1$. By definition, $\alpha^2 = (2, \dots) \vdash (k + j)$, so there are no larger parts. Fix $i = j - y + 1$ and consider $P^2_{j-y+1}(\alpha)$:

$$\begin{aligned} P^2_{j-y+1}(\alpha) &= \sum_{m=1}^{j-y+1} (\alpha_m^1 - 2\alpha_m^2 + \alpha_m^3) \\ &= (j - y)2 + 1 - (j - y + 1)2 = -1. \end{aligned}$$

Thus, the q-binomial in which P^2_{j-y+1} appears is zero, so the contribution from such a sequence is zero, and so a sequence α only contributes to $K^2_{\lambda,1^{|\lambda|}}(q)$ if $\alpha^1 = (2^j, \dots) \vdash (k+j)$ and $\alpha^2 = (1^j)$. $\square$ Now, let $k = 3$.

Theorem 12 *Let $\lambda = (n, k, j)$ and $k = 3$. Then*

$$A_3^{(n,k,j)} \subseteq \{((N), (3, \alpha^1_2, \dots, \alpha^1_r), (1^j)) \\ : \alpha^1 \vdash (k+j), \min(j, r) \geq (k+j-n), \alpha^1_1 + \alpha^1_2 + \cdots + \alpha^1_j \geq 2j\}.$$

That is, a sequence α may only contribute to $K^3_{(n,k,j),1^{n+k+j}}(q)$ if $\alpha^2 = (1^j)$, $\min(j, r) \geq (k+j-n)$, and the first j parts of α^1 sum to at least $2j$.

Proof. Similar to the proof for 2 descents, all admissible sequences α here must have $\alpha^2 = (1^j)$. Suppose $\alpha^2 \neq (1^j)$, and fix i to be the index of the last part in α^2 larger than one. The only difference here is that in $P^2_i(\alpha)$, the maximum value of α^1_m is now 3 instead of 2, which still yields a negative sum and thus contributes nothing to the final polynomial.

Now, suppose that $\alpha^2 = (1^j)$ and $\alpha^1 = (3, \alpha^1_2, \dots, \alpha^1_r)$. Then

$$\begin{aligned} P_r^1(\alpha) &= \sum_{m=1}^{r} (\alpha_m^0 - 2\alpha_m^1 + \alpha_m^2) \\ &= (n + k + j) - 2(k + j) + \min(j, r) \\ &= \min(j, r) - (k + j - n), \end{aligned}$$

which is less than zero if $\min(j, r) < (k + j - n)$. Further,

$$\begin{aligned} P_j^2(\alpha) &= \sum_{m=1}^{j} (\alpha_m^1 - 2\alpha_m^2) \\ &= \sum_{m=1}^{j} (\alpha_m^1) - 2j \\ &= (\alpha_2^1 + \cdots + \alpha_j^1) - (2j - 3). \end{aligned}$$

Since $\alpha_1^1 = 3$, this imposes the desired restriction on the first j parts of α^1. □

When we extend to $k > 3$ descents, the identification of admissible sequences increases in complexity. Not only may the last partition in a sequence take on several forms, but this choice of partition distorts the available options for the previous partition in the sequence. However, this refinement for small k allows us to throw away the vast majority of sequences α considered in the original theorem as the size of j increases.

For example, in a small case like $\lambda = (9, 4, 4)$ and $k = 3$, we are left with a total of 3 admissible sequences out of the 25 available. For $\lambda = (10, 10, 10)$, we instead keep only five out of 1386 sequences.

3.2 Closed formulas

While reducing the number of sequences considered by the summation in $K^k_{\lambda,1^{|\lambda|}}(q)$ is beneficial for computation, this also allows us to construct closed (albeit quite verbose) formulas. We will demonstrate the process for 2 and 3 descents, noting that there is nothing but an investment of time preventing one from writing down a closed formula for higher descents. However, we will see that these formulae are lengthy and it may not be a worthwhile investment to approach simplification from this angle.

We begin with tableaux of shape $\lambda = (n, k, j) \vdash N$ with 2 descents. The admissible sequences α considered by the Kirillov-Reshetikhin formula are $\alpha(m) =$

$((N),(2^{j+m},1^{k-j-2m}),(1^j))$. Then we have

$$\begin{aligned}
c(\alpha(m)) &= \sum_{a,i\geq 1}\frac{1}{2}(\alpha_i^{a-1}-\alpha_i^a)(\alpha_i^{a-1}-\alpha_i^a-1)\\
&= \sum_{i\geq 1}\frac{1}{2}(\alpha_i^0-\alpha_i^1)(\alpha_i^0-\alpha_i^1-1)+\frac{1}{2}(\alpha_i^1-\alpha_i^2)(\alpha_i^1-\alpha_i^2-1)\\
&= \left(\frac{1}{2}(N-2)(N-3)+(j+m-1)\cdot\frac{1}{2}(-2)(-3)+(k-j-2m)\cdot\frac{1}{2}(2)\right)\\
&\quad +\left(j\cdot\frac{1}{2}(1)(0)+m\cdot\frac{1}{2}(2)(1)+(k-j-2m)\frac{1}{2}(1)(0)\right)\\
&= \frac{1}{2}(N-2)(N-3)+3(j+m-1)+(k-j-2m)+m\\
&= \frac{1}{2}N(N-5)+k+2j+2m
\end{aligned}$$

and

$$P_i^a(\alpha(m)) = \sum_{s=1}^{i}(\alpha_s^{a-1}-2\alpha_s^a+\alpha_s^{a+1}) = \sum_{s=1}^{i}\psi_a^m(s),$$

where for $m < \lfloor \frac{k-j}{2} \rfloor$ we define

$$\psi_1^m(s) := \begin{cases} N-3 & , s = 1 \\ -3 & , s = 2 \text{ to } j \\ -4 & , s = j+1 \text{ to } j+m \\ -2 & , s = j+m+1 \text{ to } k-m \end{cases},$$

$$\psi_2^m(s) := \begin{cases} 0 & , s = 1 \text{ to } j \\ 2 & , s = j+1 \text{ to } j+m \\ 1 & , s = j+m+1 \text{ to } k-m \end{cases}$$

and for $m = \frac{k-j}{2}$ we similarly define

$$\psi_1^{\frac{k-j}{2}}(s) := \begin{cases} N-3 & , s = 1 \\ -3 & , s = 2 \text{ to } j \\ -4 & , s = j+1 \text{ to } j+m \end{cases},$$

$$\psi_2^{\frac{k-j}{2}}(s) := \begin{cases} 0 & , s = 1 \text{ to } j \\ 2 & , s = j+1 \text{ to } j+m \end{cases}.$$

Now, evaluate $K^2_{\lambda,1^N}(q)$ using the admissible sequences and algebraically simplify. Letting $\delta = k - j - 1 \pmod 2$ and $A = \frac{1}{2}N(N-5) + k + 2j$, we have

$$K^2_{\lambda,1^N}(q) = q^A \sum_{\substack{m=0 \\ \alpha=\alpha(m)}}^{\frac{k-j}{2}} q^{2m} \prod_{i\geq 1} \begin{bmatrix} P_i^1(\alpha) + (\alpha_i^1 - \alpha_{i+1}^1) \\ (\alpha_i^1 - \alpha_{i+1}^1) \end{bmatrix}_q \begin{bmatrix} P_i^2(\alpha) + (\alpha_i^2 - \alpha_{i+1}^2) \\ (\alpha_i^2 - \alpha_{i+1}^2) \end{bmatrix}_q$$

$$= q^A \Bigg[\Bigg(\sum_{m=0}^{\frac{k-j-1}{2}} q^{2m} \begin{bmatrix} N-3j-4m+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-2k-j+1 \\ 1 \end{bmatrix}_q \Bigg) + \delta q^{k-j} \begin{bmatrix} N-2k-j+2 \\ 2 \end{bmatrix}_q \Bigg]$$

$$= q^A \Bigg[\begin{bmatrix} N-2k-j+1 \\ 1 \end{bmatrix}_q \Bigg(\sum_{m=0}^{\frac{k-j-1}{2}} q^{2m} \begin{bmatrix} N-3j-4m+1 \\ 1 \end{bmatrix}_q \Bigg) + \delta q^{k-j} \begin{bmatrix} N-2k-j+2 \\ 2 \end{bmatrix}_q \Bigg].$$

The remaining inner sum can be simplified as follows:

$$\sum_{m=0}^{\lfloor \frac{k-j-1}{2} \rfloor} q^{2m} \begin{bmatrix} N-3j-4m+1 \\ 1 \end{bmatrix}_q = \sum_{m=0}^{\lfloor \frac{k-j-1}{2} \rfloor} q^{2m} \begin{bmatrix} n+k-2j-4m+1 \\ 1 \end{bmatrix}_q$$

$$= \begin{bmatrix} n-j+\delta+2 \\ 1 \end{bmatrix}_q (1 + q^2 + \cdots + q^{k-j-\delta-1}).$$

To see this, the left-hand side is comprised of concentric strands of polynomials $q^{2m}(1+q+\cdots+q^{n+k-2j-4m})$ for $0 \leq m \leq \frac{k-j-1-\delta}{2}$. The right-hand side is comprised of strands of polynomials $1+q+\cdots+q^{n-j+1+\delta}$ shifted by $2m$ for $0 \leq m \leq \frac{k-j-1-\delta}{2}$, resulting in a polynomial which increases every other degree for each m before stabilizing, up to the central degree. Note also that the first strand comprising this polynomial doesn't end until after the last one begins, since $n-j+1+\delta \geq k-j+1+\delta > k-j-1-\delta$.

Hence, we have

$$K^2_{\lambda,1^N}(q) = q^A\Bigg(\begin{bmatrix} N-2k-j+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-k-2j+\delta+2 \\ 1 \end{bmatrix}_q \frac{1-q^{k-j-\delta+1}}{1-q^2} + \delta q^{k-j} \begin{bmatrix} N-2k-j+2 \\ 2 \end{bmatrix}_q\Bigg).$$

Thus, we have the following result:

$$f_{(n,k,j),2} = q^{k+2j}\Bigg(\begin{bmatrix} N-2k-j+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-k-2j+\delta+2 \\ 1 \end{bmatrix}_q \frac{1-q^{k-j+1-\delta}}{1-q^2} + \delta q^{k-j} \begin{bmatrix} N-2k-j+2 \\ 2 \end{bmatrix}_q\Bigg).$$

Let us now consider the tableaux of the same shape with 3 descents. The admissible sequences here are $\alpha(m_3, m_2, m_1) = ((N), (3^{m_3+1}, 2^{m_2}, 1^{m_1}), (1^j))$, with the restriction that $\sum_{i=1}^{j} \alpha_i^1 \geq 2j$ and $\min(j, r) \geq N - 2n$. Note that $0 \leq m_3 \leq \lfloor \frac{k+j-3}{3} \rfloor$, $0 \leq m_2 \leq \lfloor \frac{k+j-3-3m_3}{3} \rfloor$, and $m_1 = k+j-2m_2-3m_3-3$. Also, note that $r = m_1+m_2+m_3+1$. We will ignore the argument of $\alpha(m_3, m_2, m_1)$ going forward for brevity. The contribution of a sequence α can be placed into one of 16 different cases. We will say that an admissible sequence α is of type (A, B), where

$$A = \begin{cases} 1 & \text{if } r \leq j \\ 2 & \text{if } m_2 + m_3 + 1 \leq j \leq r \\ 3 & \text{if } m_3 + 1 \leq j \leq m_2 + m_3 \\ 4 & \text{if } 1 \leq j \leq m_3 \end{cases}, \qquad B = \begin{cases} 1 & \text{if } m_1, m_2 > 0 \\ 2 & \text{if } m_2 > 0,\, m_1 = 0 \\ 3 & \text{if } m_1 > 0,\, m_2 = 0 \\ 4 & \text{if } m_1, m_2 = 0 \end{cases}.$$

The A-type of a given admissible sequene α is dependent on the relationship between the length of α^1 and j, whereas the B-type is dependent on the content of sequence α^1. To be safe, if there are any sequences marked as admissible that contribute zero, mark their type as (0,0) and discard them from all subsequent computations.

First, we compute $c(\alpha)$ for each sequence type. This yields

$$c(\alpha) = ((N-3)(N-4) + 2k + 2j - 6) + 2C_\alpha,$$

where

$$C_\alpha = \begin{cases} \alpha \in (1, B): & 3m_3 - m_1 + j \\ \alpha \in (2, B): & 4m_3 + m_2 + 1 \\ \alpha \in (3, B): & 5m_3 + 2m_2 - 2j + 3 \\ \alpha \in (4, B): & 6m_3 + 2m_2 - 2j + 6 \end{cases}.$$

For each sequence α, its product in the Kirillov-Reshetikhin formula will range over all $a, i \geq 1$. For an admissible sequence α, almost all terms will go to one, dependent entirely on the B-type of α. The only ones that don't are given as the contributions in M_α, where

$$M_\alpha = \begin{cases} \alpha \in (A,1) & : \begin{bmatrix} P^1_{m_3+1}(\alpha)+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} P^1_{m_3+m_2+1}(\alpha)+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} P^1_{m_3+m_2+m_1+1}(\alpha)+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} P^2_j(\alpha)+1 \\ 1 \end{bmatrix}_q \\ \alpha \in (A,2) & : \begin{bmatrix} P^1_{m_3+1}(\alpha)+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} P^1_{m_3+m_2+1}(\alpha)+2 \\ 2 \end{bmatrix}_q \begin{bmatrix} P^2_j(\alpha)+1 \\ 1 \end{bmatrix}_q \\ \alpha \in (A,3) & : \begin{bmatrix} P^1_{m_3+1}(\alpha)+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} P^1_{m_3+m_1+1}(\alpha)+2 \\ 2 \end{bmatrix}_q \begin{bmatrix} P^2_j(\alpha)+1 \\ 1 \end{bmatrix}_q \\ \alpha \in (A,4) & : \begin{bmatrix} P^1_{m_3+1}(\alpha)+3 \\ 3 \end{bmatrix}_q \begin{bmatrix} P^2_j(\alpha)+1 \\ 1 \end{bmatrix}_q \end{cases}.$$

Now, we must determine the effect that the A-type of an admissible sequence will have on its contribution. Define $\Psi^\alpha_{a,A}(s)$ as the sth term of $P^a_i(\alpha)$ when α is of type (A,B). Then we have the following:

$$\psi^\alpha_{1,1}(s) = \begin{cases} N-5 & , s = 1 \\ -5 & , s = 2 \text{ to } m_3+1 \\ -3 & , s = m_3+2 \text{ to } m_2+m_3+1 \\ -1 & , s = m_2+m_3+2 \text{ to } r \\ 1 & , s = r+1 \text{ to } j \end{cases},$$

$$\psi^{\alpha}_{2,1}(s) = \begin{cases} 1 & , s = 1 \text{ to } m_3 + 1 \\ 0 & , s = m_3 + 2 \text{ to } m_2 + m_3 + 1 \\ -1 & , s = m_2 + m_3 + 2 \text{ to } r \\ -2 & , s = r + 1 \text{ to } j \end{cases},$$

$$\psi^{\alpha}_{1,2}(s) = \begin{cases} N - 5 & , s = 1 \\ -5 & , s = 2 \text{ to } m_3 + 1 \\ -3 & , s = m_3 + 2 \text{ to } m_2 + m_3 + 1 \\ -1 & , s = m_2 + m_3 + 2 \text{ to } j \\ -2 & , s = j + 1 \text{ to } r \end{cases},$$

$$\psi^{\alpha}_{2,2}(s) = \begin{cases} 1 & , s = 1 \text{ to } m_3 + 1 \\ 0 & , s = m_3 + 2 \text{ to } m_2 + m_3 + 1 \\ -1 & , s = m_2 + m_3 + 2 \text{ to } j \\ 1 & , s = j + 1 \text{ to } r \end{cases},$$

$$\psi^{\alpha}_{1,3}(s) = \begin{cases} ,N-5 & ,\ s=1 \\ -5 & ,\ s=2 \text{ to } m_3+1 \\ -3 & ,\ s=m_3+2 \text{ to } j \\ -4 & ,\ s=j+1 \text{ to } m_2+m_3+1 \\ -2 & ,\ s=m_2+m_3+2 \text{ to } r \end{cases},$$

$$\psi^{\alpha}_{2,3}(s) = \begin{cases} 1 & ,\ s=1 \text{ to } m_3+1 \\ 0 & ,\ s=m_3+2 \text{ to } j \\ 2 & ,\ s=j+1 \text{ to } m_2+m_3+1 \\ 1 & ,\ s=m_2+m_3+2 \text{ to } r \end{cases},$$

$$\psi^{\alpha}_{1,4}(s) = \begin{cases} N-5 & ,\ s=1 \\ -5 & ,\ s=2 \text{ to } j \\ -6 & ,\ s=j+1 \text{ to } m_3+1 \\ -4 & ,\ s=m_3+2 \text{ to } m_2+m_3+1 \\ -2 & ,\ s=m_2+m_3+2 \text{ to } r \end{cases},$$

$$\psi^{\alpha}_{2,4}(s) = \begin{cases} 1 & , s = 1 \text{ to } j \\ 3 & , s = j+1 \text{ to } m_3 + 1 \\ 2 & , s = m_3 + 2 \text{ to } m_2 + m_3 + 1 \\ 1 & , s = m_2 + m_3 + 2 \text{ to } r \end{cases}.$$

This, combined with the previous observations, allow us to determine the contribution of a sequence α based solely off of its type (A, B):

$$M_\alpha = \begin{cases} \alpha \in (1,1): & \begin{bmatrix} N-5(m_3+1)+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-5(m_3+1)-3m_2+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-5(m_3+1)-3m_2-m_1+1 \\ 1 \end{bmatrix}_q \\ & \cdot \begin{bmatrix} 3(m_3+1)+2m_2+m_1-2j+1 \\ 1 \end{bmatrix}_q \\ \alpha \in (1,2): & \begin{bmatrix} N-5(m_3+1)+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-5(m_3+1)-3m_2+2 \\ 2 \end{bmatrix}_q \begin{bmatrix} 3(m_3+1)+2m_2+m_1-2j+1 \\ 1 \end{bmatrix}_q \\ \alpha \in (1,3): & \begin{bmatrix} N-5(m_3+1)+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-5(m_3+1)-3m_2-m_1+2 \\ 2 \end{bmatrix}_q \begin{bmatrix} 3(m_3+1)+2m_2+m_2-2j+1 \\ 1 \end{bmatrix}_q \\ \alpha \in (1,4): & \begin{bmatrix} N-5(m_3+1)+3 \\ 3 \end{bmatrix}_q \begin{bmatrix} 3(m_3+1)+2m_2+m_1-2j+1 \\ 1 \end{bmatrix}_q \\ \alpha \in (2,1): & \begin{bmatrix} N-5(m_3+1)+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-5(m_3+1)-3m_2+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-6(m_3+1)-4m_2-2m_1+j+1 \\ 1 \end{bmatrix}_q \\ & \cdot \begin{bmatrix} 2(m_3+1)+m_2-j+1 \\ 1 \end{bmatrix}_q \\ \alpha \in (2,2): & \begin{bmatrix} N-5(m_3+1)+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-5(m_3+1)-3m_2+2 \\ 2 \end{bmatrix}_q \begin{bmatrix} 2(m_3+1)+m_2-j+1 \\ 1 \end{bmatrix}_q \\ \alpha \in (2,3): & \begin{bmatrix} N-5(m_3+1)+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-6(m_3+1)-4m_2-2m_1+j+2 \\ 2 \end{bmatrix}_q \begin{bmatrix} 2(m_3+1)+m_2-j+1 \\ 1 \end{bmatrix}_q \\ \alpha \in (2,4): & \begin{bmatrix} N-5(m_3+1)+3 \\ 3 \end{bmatrix}_q \begin{bmatrix} 2(m_3+1)+m_2-j+1 \\ 1 \end{bmatrix}_q \\ \alpha \in (3,1): & \begin{bmatrix} N-5(m_3+1)+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-6(m_3+1)-4m_2+j+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-6(m_3+1)-4m_2-2m_1+j+1 \\ 1 \end{bmatrix}_q \\ & \cdot \begin{bmatrix} m_3+2 \\ 1 \end{bmatrix}_q \\ \alpha \in (3,2): & \begin{bmatrix} N-5(m_3+1)+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-6(m_3+1)-4m_2+j+2 \\ 2 \end{bmatrix}_q \begin{bmatrix} m_3+2 \\ 1 \end{bmatrix}_q \\ \alpha \in (3,3): & \begin{bmatrix} N-5(m_3+1)+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-6(m_3+1)-4m_2-2m_1+j+2 \\ 2 \end{bmatrix}_q \begin{bmatrix} m_3+2 \\ 1 \end{bmatrix}_q \\ \alpha \in (3,4): & \begin{bmatrix} N-5(m_3+1)+3 \\ 3 \end{bmatrix}_q \begin{bmatrix} m_3+2 \\ 1 \end{bmatrix}_q \\ \alpha \in (4,1): & \begin{bmatrix} N-6(m_3+1)+j+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-6(m_3+1)-4m_2+j+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-6(m_3+1)-4m_2-2m_1+j+1 \\ 1 \end{bmatrix}_q \\ & \cdot \begin{bmatrix} j+1 \\ 1 \end{bmatrix}_q \\ \alpha \in (4,2): & \begin{bmatrix} N-6(m_3+1)+j+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-6(m_3+1)-4m_2+j+1 \\ 2 \end{bmatrix}_q \begin{bmatrix} j+1 \\ 1 \end{bmatrix}_q \\ \alpha \in (4,3): & \begin{bmatrix} N-6(m_3+1)+j+1 \\ 1 \end{bmatrix}_q \begin{bmatrix} N-6(m_3+1)-4m_2-2m_1+j+2 \\ 2 \end{bmatrix}_q \begin{bmatrix} j+1 \\ 1 \end{bmatrix}_q \\ \alpha \in (4,4): & \begin{bmatrix} N-6(m_3+1)+j+3 \\ 3 \end{bmatrix}_q \begin{bmatrix} j+1 \\ 1 \end{bmatrix}_q \end{cases}.$$

Finally, we have

$$K^3_{\lambda,1^N}(q) = q^{(N-3)(N-4)+2k+2j-6} \sum_\alpha q^{2C_\alpha} M_\alpha.$$

Performing the required substitution $q \to q^{-1}$ and shifting appropriately yields the final closed form for $f_{(n,k,j),3}$ from this process.

Chapter 4

Relations among $f_{\lambda,k}$

As we've seen throughout this manuscript, many of the major index over descent polynomials for a given partition shape λ arise as polynomial multiples of $f_{\mu,k}$ for a combinatorially related partition shape μ, often giving new representations for $f_{\lambda,k}$ not apparent from more general formulae.

To this end, it may be advantageous to build $f_{\lambda,k}$ from smaller tableau shapes or from tableaux with fewer descents to find further relations among these polynomials. That is, take $f_{\mu,k-i}$ for some $\mu \subseteq \lambda$ and $i \geq 0$, and construct $f_{\lambda,k}$ by performing a number of combinatorial operations on the tableaux in $\mathrm{SYT}(\mu, k-i)$ which shift the major index by a specified amount. Algebraically, we write this as $f_{\lambda,k}(q) = g(q) \cdot f_{\mu,k-i}(q)$, where $g(q)$ is a nonnegative polynomial in q. Unfortunately, we do not seem to always

be able to do this. However, if we set $\mu = \lambda$ and $k - i = k - 1$, a curious pattern arises:

Conjecture 2 *Let* $\lambda = (n, k, j)$. *Then* $f_{\lambda,3} = g(q) f_{\lambda,2}$ *if and only if* $(n+3)(k+2)(j+1) \equiv 0 \pmod 6$.

Further, let $\lambda = (\lambda_1, \ldots, \lambda_r)$. *Then* $f_{\lambda,r} = g(q) f_{\lambda,r-1}$ *if and only if* $\prod_i (\lambda_i + r - i + 1) \equiv 0 \pmod{r!}$.

To explore this relation, we define a slightly modified polynomial long division by the following algorithm. Let $f_a(q)$ and $f_b(q)$ be symmetric, nonnegative polynomials in q. We 'divide' f_b by f_a as follows:

1. Let $r_b(q) = f_b(q)$ and $g_b(q) = 0$. Define $\text{mindeg}(f(q))$ as the minimum degree of a polynomial $f(q)$.
2. While $r_b(q) - q^{\text{mindeg}(r_b(q)) - \text{mindeg}(f_a(q))} f_a$ has nonnegative coefficients, do the following:
   (a) subtract $q^{\text{mindeg}(r_b(q)) - \text{mindeg}(f_a(q))} f_q$ from $r_b(q)$;
   (b) add $q^{\text{mindeg}(r_b(q)) - \text{mindeg}(f_a(q))}$ to $g_b(q)$.
3. The result is $f_b(q) = g_b(q) \cdot f_a(q) + r_b(q)$.

Because f_a and f_b are symmetric, this algorithm is the same as Euclidean division

with the caveat that we halt the process once $q^A r_b(q)$ no longer dominates $f_a(q)$ for any $A \in \mathbb{Z}$, that is once $[q^n]q^A r_b(q) < [q^n]f_a(q)$ for some integer n and any integer A. Equivalently, we halt the standard division algorithm immediately before the quotient or remainder obtains a negative coefficient. While this implies that $\deg(r_b(q))$ is not necessarily strictly less than $\deg(f_a(q))$, the algorithm does guarantee that both $g_b(q)$ and $r_b(q)$ have nonnegative coefficients. Experimental data suggests the following properties of $g_b(q)$ and $r_b(q)$, the 'quotient' and 'remainder' polynomials from this process:

Conjecture 3 *Let $\lambda = (n, k, j)$. Applying the modified division algorithm to $f_{\lambda,3}(q)$ and $f_{\lambda,2}$ yields*

$$f_{\lambda,3}(q) = g_\lambda(q) \cdot f_{\lambda,2}(q) + r_\lambda(q), \tag{4.1}$$

where $r_\lambda(q)$ is always an Euler product (or zero) and $g_\lambda(q)$ can be constructed from the 'quotient' polynomials $g_\mu(q)$ for some partition μ dominated by λ.

Consider $\lambda = (3n + a, 3n + a, 2)$. While we have a closed formula for $f_{\lambda,3}(q)$ in this case through Theorem 6 of our manuscript and Theorem 5 in [10], additional interpretations may prove useful. When $j = 2$, we have the following forms for $f_{\lambda,3}$, split into three cases based on the congruence class of $3n + a \pmod 3$:

Conjecture 4 *The following relations hold:*

$$f_{(3n,3n,2),3} = \left[g_{(3n,3n,1)} + q^2 \frac{1-q^{6n}}{1-q^6} + \sum_{i=1}^{n} [q^{3i+1} \frac{1-q^{3i}}{1-q}] \right] f_{(3n,3n,2),2},$$

$$f_{(3n+1,3n+1,2),3} = \left[g_{(3n+1,3n+1,1)} + q^2 + q^4 \frac{1-q^{6n}}{1-q^6} + \sum_{i=1}^{n} [q^{3i+2} \frac{1-q^{3i+1}}{1-q}] \right] f_{(3n+1,3n+1,2),2},$$

$$f_{(3n+2,3n+2,2),3} = \left[g_{(3n+2,3n+2,1)} + q^2 + (q^3+q^5) \frac{1-q^{6n}}{1-q^3} \right.$$
$$\left. - q^3 \frac{1-q^{6n}}{1-q^6} + \sum_{i=0}^{n} [q^{3i+3} \frac{1-q^{3i+2}}{1-q}] \right] f_{(3n+2,3n+2,2),2},$$

where

$$g_{(3n+a,3n+a,1)} = q^3 \frac{(1-q^{3n+a})(1-q^{3n+a-1})}{(1-q)(1-q^3)}.$$

We refrain from closing the first sum to display the similarity among these conjectured, but note that $\sum_{i=1}^{n} q^{3i+1} \frac{1-q^{3i}}{1-q} = q^4 \frac{(1-q^{3n})(1-q^{3n+3})}{(1-q)(1-q^6)}$ and so the formula for $f_{(3n,3n,2),3}$ given is correct by applying Theorem 3 and 5 in [10]. The corresponding summations in the other two formulas, however, do not yield a single Euler product, nor a unimodal or symmetric polynomial. Since $(3n+3)(3n+2)(3) \equiv (3n+4)(3n+3)(3) \equiv (3n+5)(3n+4)(3) \equiv 0 \pmod 6$, we expect the remainder polynomial to be zero in each case. Further, if we increase the size of the last part of λ, it appears to have a predictable effect on the resulting quotient polynomial associated with the new partition shape.

Conjecture 5

$$
\begin{aligned}
f_{(3n,3n,j),3} &= \left[g_{(3n,3n,1)} + \frac{1-q^{j-1}}{1-q} \left(\sum_{i=1}^{n} [q^{3i+1} \frac{1-q^{3i}}{1-q} + q^{6i-4}] \right) \right] f_{\lambda,2}, \\
f_{(3n+1,3n+1,j),3} &= \left[g_{(3n+1,3n+1,1)} + \frac{1-q^{j-1}}{1-q} \left(q^{6n-2} \frac{1-q^{6n}}{1-q^6} + \sum_{i=0}^{n} q^{3i+2} \frac{1-q^{3i+1}}{1-q} \right) \right] f_{\lambda,2}, \\
f_{(3n+2,3n+2,j),3} &= \left[g_{(3n+2,3n+2,1)} + \frac{1-q^{j-1}}{1-q} \left(q^2 + q^5 \frac{1-q^{6n}}{1-q^3} - q^3 \frac{1-q^{6n}}{1-q^6} \right. \right. \\
&\quad \left. \left. + \sum_{i=0}^{n} [q^{3i+3} \frac{1-q^{3i+2}}{1-q}] \right) - q^{6n+3} \left(\frac{1-q^{j-1}}{1-q} - \frac{1-q^{1-q^{3\lfloor \frac{j+1}{3} \rfloor}}}{1-q^3} \right) \right] f_{\lambda,2} + r_\lambda.
\end{aligned}
$$

While we expect the first two to have zero remainder, the last may not since $(3n+5)(3n+4)(j+1) \equiv 0, 2, 4 \pmod 6$, dependent on the value of j. If we instead choose to increase the size of the first part or, if the first is larger than the second, the size of the second part instead, similar patterns seem to arise.

One potential approach to a proof of these conjectures and similar results is a combinatorial mapping $\phi : \mathrm{SYT}(\lambda, 2) \to \mathrm{SYT}(\lambda, 3)$ wherein one considers each tableaux in $\mathrm{SYT}(\lambda, 2)$ and tracks how a descent may be added and the major index of the resulting tableaux. If the modified division algorithm above produces a remainder, we hope that this corresponds to some number of tableaux in $\mathrm{SYT}(\lambda, 3)$ that cannot be constructed from those with two descents.

Preliminary computation suggests that similar patterns arise when considering tableaux with more descents or of partitions into more parts, giving hope that there

is an underlying structure to this modified division. A method that allows us to procedurally construct the major index over descent polynomials for larger standard Young tableaux would allow us to negate the intense computational complexity that comes with the larger size, a great boon when studying distributions on these objects. Contained within Appendix A is SageMath ([21]) code written to demonstrate this relationship for tableaux of any size.

References

[1] G. E. Andrews, *The Theory of Partitions*. Encyclopedia of Mathematics and its Applications, Cambridge University Press, 1984.

[2] I. Macdonald, *Symmetric functions and Hall polynomials*. Oxford University Press, 1979.

[3] R. P. Stanley and S. Fomin, *Enumerative Combinatorics*, vol. 2 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1999.

[4] R. P. Stanley, *Enumerative Combinatorics: Volume 1*. USA: Cambridge University Press, 2nd ed., 2011.

[5] K. M. O'Hara, "Unimodality of Gaussian coefficients: A constructive proof," *Journal of Combinatorial Theory, Series A*, vol. 53, no. 1, pp. 29–52, 1990.

[6] D. Zeilberger, "Kathy O'Hara's constructive proof of the unimodality of the Gaussian polynomials," *Am. Math. Monthly*, vol. 96, p. 590–602, Aug. 1989.

[7] F. Zanello, "On Bergeron's positivity problem for q -binomial coefficients," *The Electronic Journal of Combinatorics*, vol. 25, 2018. P2.17.

[8] D. S. Frederick M Goodman, Kathleen M O'Hara, "A unimodality identity for a Schur function," *Journal of Combinatorial Theory, Series A*, vol. 60, no. 1, pp. 143–146, 1992.

[9] S.-E. Cheng, S. Elizalde, A. Kasraoui, and B. E. Sagan, "Inversion polynomials for 321-avoiding permutations," *Discrete Mathematics*, vol. 313, no. 22, pp. 2552–2565, 2013.

[10] W. J. Keith, "Families of major index distributions: Closed forms and unimodality," *The Electronic Journal of Combinatorics*, vol. 26, 2019. P3.58.

[11] L. Butler and A. M. Society, *Subgroup Lattices and Symmetric Functions*. American Mathematical Society: Memoirs of the American Mathematical Society, American Mathematical Society, 1994.

[12] A. Kirillov and N. Reshetikhin, "The Bethe Ansatz and the combinatorics of Young tableaux," *J Math Sci*, vol. 40, pp. 925–955, 1988.

[13] G. Wang, "Enumerating quasi-Yamanouchi tableaux of Durfee size two," 2018.

[14] F. Bergeron, "The q-Foulkes conjecture." Talk delivered at Bowdoin College, ME, 2016.

[15] I. Pak and G. Panova, "Strict unimodality of q-binomial coefficients," *Comptes Rendus Mathematique*, vol. 351, pp. 415–418, 2013.

[16] R. Stanley and F. Zanello, "A generalization of a 1998 unimodality conjecture of Reiner and Stanton," *arXiv: Combinatorics*, 2017.

[17] T. Wagner, "Integer partitions under certain finiteness conditions," *Open Access Dissertation*, 2021.

[18] V. Dhand, "A combinatorial proof of strict unimodality for q-binomial coefficients," *Discret. Math.*, vol. 335, pp. 20–24, 2014.

[19] B. Rhoades, "Cyclic sieving, promotion, and representation theory," *Journal of Combinatorial Theory, Series A*, vol. 117, no. 1, pp. 38–76, 2010.

[20] Rhee, Donguk, "Cyclic sieving phenomenon of promotion on rectangular tableaux," Master's thesis, 2012.

[21] The Sage Developers, *SageMath, the Sage Mathematics Software System (Version 9.0)*, 2021. `https://www.sagemath.org`.

[22] M. Fulmek, "Viewing determinants as nonintersecting lattice paths yields classical determinantal identities bijectively," *The Electronic Journal of Combinatorics*, vol. 19, 2012. P21.

[23] W. R. Inc., "Mathematica, Version 12.3.1." Champaign, IL, 2021.

Appendix A

SageMath Code

This appendix gives SageMath ([21]) code to write $f_{\lambda,k}(q)$ using $K^k_{\lambda,1^{|\lambda|}}(q)$ in varying ways. For example, we may generate all nonzero $f_{\lambda,k}(q)$ for partitions λ dominated by another partition, say Λ. We may also write the admissible sequences α that arise in computation of $K^k_{\lambda,1^{|\lambda|}}(q)$ and write explicitly their contribution to $f_{\lambda,k}(q)$. The intention is for this code to be used in a notebook environment, e.g. *Jupyter*, though it can be minimally modified to run elsewhere. We note that the space and time complexity of all subsequent functions can likely be improved significantly by a more proficient programmer.

A.1 Major index over descent via $\mathrm{SYT}(\lambda, k)$

A.1.1 Code

```
import sys
from sage.all import *
import argparse
from datetime import datetime
import time
import numpy

R, q = objgen(QQ['q'])
q    = gen(QQ['q'])

# Generate all partitions whose Young diagrams fit inside of the given partition's.
def gen_partitions_inside(shape=[]):
    ptn = Partition(shape)
    r = len(shape)
    n = ptn.size()
    print("Calculating partitions within {}, a partition of {}, with {} rows".format(shape,n,r))

    new = deepcopy(shape)

    insideList=[ptn]

    for i in range(0,n):
        lenIn = len(insideList)
        for j in range(0,lenIn):
            lil = insideList[j]
            if (lil.size() == n-i):
                insideList.extend(lil.down_list())
                insideList = sorted(list( dict.fromkeys(insideList) ))

    insideList = sorted(list( dict.fromkeys(insideList) ))
    print("Partitions inside {}: {}".format(ptn,insideList))
    return insideList
```

```
# Generate all partitions whose Young diagram fit inside of the given partition's,
# with the condition that the partitions also have the same number of parts.
def gen_partitions_inside_same_len(shape=[]):
    ptn = Partition(shape)
    r = len(shape)
    n = ptn.size()
    print("Calculating partitions with {} rows within {}, a partition of {}, with {} rows".format(r,shape,n,r))

    new = deepcopy(shape)

    insideList=[ptn]
    out = []

    for i in range(0,n):
        lenIn = len(insideList)
        for j in range(0,lenIn):
            lil = insideList[j]
            if (lil.size() == n-i):
#                print("working on lil: {}".format(lil))
                insideList.extend(lil.down_list())
                insideList = sorted(list( dict.fromkeys(insideList) ))

    for ptn in insideList:
        if len(ptn) == r:
            out.append(ptn)

    out = sorted(list( dict.fromkeys(out) ))

    print("Partitions inside {} with length {}: {}".format(ptn,r,out))
    return out


# Given a partition shape, generate all associated nonzero f_{lambda, k}(q).
# If extraInfo is true, the returned list will contain the name of the polynomial (f_{lambda, k}) for each.
def gen_all_flambda(shape, extraInfo):
    min = len(shape)-1                          #minimum number of descents
    max = sum(shape)-shape[0]                   #maximum number of descents
```

```
    polylist = [0 for n in range(min,max+1)]   #polynomial ↩
       list... we'll store our flambda in here.
    s = StandardTableaux(shape)

    print("---Generating Major Index over Descent polynomials↩
        for ", shape,"---")

    for tbx in s:
        i = tbx.standard_number_of_descents()
        polylist[i-min] += q**(tbx.standard_major_index())

    outlist = []

    for i in range(0,len(polylist)):
        f = "f_{{{},{}}}".format(shape, i+min)

        if extraInfo:
            outlist.append([f, polylist[i]])
        else:
            outlist.append([polylist[i]])
        print("{}= {}".format(f,polylist[i]))

    print("\n")
    return outlist

# Given a partition shape "shape", generate the nonzero f_{↩
   lambda, k}(q) polynomials for all SYT of shape
# lambda inside "shape".
def gen_all_flambda_inside(shape, extraInfo):
    insideList = gen_partitions_inside(shape)
    outList = []

    for i in range(0,len(insideList)):
        ptn = insideList[i]
        if len(ptn)>0:
            out = gen_all_flambda(ptn, extraInfo)
            outList.append(out)

    return outList

# Given a partition shape "shape", generate the nonzero f_{↩
   lambda, k}(q) polynomials for all SYT of shape
# lambda inside "shape" with the same number of parts.
def gen_all_flambda_inside_same_len(shape, extraInfo):
    insideList = gen_partitions_inside_same_len(shape)
    outList = []
```

```
    for ptn in insideList:
        if len(ptn)>0:
            out = gen_all_flambda(ptn, extraInfo)
            outList.append(out)

    return outList

# Helper function to handle extraInfo.
def librarize(inList):
    out = []
    for i in inList:
        for j in i:
            out.append(j[-1])
    return out

# Given a partition shape "shape" and number of descents "↩
   number_of_descents", return
# SYT(shape, number_of_descents) and print f_{shape, ↩
   number_of_descents}.
def flambdai_tbx(shape, number_of_descents):
    print("--Generating tableaux of shape {} with {} descents↩
       . Please hold...--".format(shape,number_of_descents))
    s = StandardTableaux(shape).list()
    out = []

    for tbx in s:
        if tbx.standard_number_of_descents() == ↩
           number_of_descents:
            out.append(tbx)

    out = sorted(out, key=lambda x: x.standard_major_index())

    for tbx in out:
        tbx.pp()
        print("")
    return out


# Given a partition shape "shape", number of descents "↩
   number_of_descents", and major index "major_index", return
# subset of SYT(shape, number_of_descents) with specified ↩
   major index and print out the tableaux.
def flambdai_tbx_fixed_maj(shape, number_of_descents, ↩
   major_index):
    print("--Generating tableaux of shape {} with {} descents↩
        and major index {}. Please hold, ...--".format(shape,↩
        number_of_descents, major_index))
```

```
    s = StandardTableaux(shape).list()
    out = []

    for tbx in s:
        if tbx.standard_number_of_descents() == ↩
           number_of_descents and tbx.standard_major_index() ↩
           == major_index:
            out.append(tbx)

    out = sorted(out, key=lambda x: x.standard_major_index())

    for tbx in out:
        tbx.pp()
        print("")

    return out

# Given a tableau "tableau", write out the threads of tableau↩
    .
def tableau_strands(tableau):

    mark = []
    tbx = StandardTableau(tableau)
    des = tbx.standard_descents()

    out = []
    strand = []
    for i in range(1,tbx.size()+1):
        if i in des:
            strand.append(i)
            out.append(strand)
            strand = []
            continue
        if i == tbx.size():
            strand.append(i)
            out.append(strand)
            strand=[]
            continue
        else:
            strand.append(i)
            continue
    return out
```

A.1.2 Examples

```
# flambdai_tbx([3,2,1], 3)

--Generating tableaux of shape [3, 2, 1] with 3 descents. Please hold...--
  1  4  6
  2  5
  3

  1  3  6
  2  4
  5

  1  4  5
  2  6
  3

  1  3  5
  2  6
  4

  1  3  5
  2  4
  6

  1  2  5
  3  6
  4

  1  3  4
  2  5
  6

  1  2  4
  3  5
  6
```

```
 # gen_all_flambda([4,2,2], True)
```

```
---Generating Major Index over Descent polynomials for  [4, 2, 2] ---
f_{[4, 2, 2],2}= q^10 + q^9 + 2*q^8 + q^7 + q^6
f_{[4, 2, 2],3}= q^16 + 2*q^15 + 4*q^14 + 5*q^13 + 6*q^12 + 5*q^11 + 4*q^10
    + 2*q^9 + q^8
f_{[4, 2, 2],4}= q^20 + q^19 + 3*q^18 + 3*q^17 + 4*q^16 + 3*q^15 + 3*q^14
        + q^13 + q^12


# gen_all_flambda_inside([3,2,1], False)

Calculating partitions within [3, 2, 1], a partition of 6, with 3 rows
Partitions inside [3, 2, 1]: [[], [1], [1, 1], [1, 1, 1], [2], [2, 1],
 [2, 1, 1], [2, 2], [2, 2, 1], [3], [3, 1], [3, 1, 1], [3, 2], [3, 2, 1]]

---Generating Major Index over Descent polynomials for  [1] ---
f_{[1],0}= 1


---Generating Major Index over Descent polynomials for  [1, 1] ---
f_{[1, 1],1}= q


---Generating Major Index over Descent polynomials for  [1, 1, 1] ---
f_{[1, 1, 1],2}= q^3


---Generating Major Index over Descent polynomials for  [2] ---
f_{[2],0}= 1


---Generating Major Index over Descent polynomials for  [2, 1] ---
f_{[2, 1],1}= q^2 + q


---Generating Major Index over Descent polynomials for  [2, 1, 1] ---
f_{[2, 1, 1],2}= q^5 + q^4 + q^3


---Generating Major Index over Descent polynomials for  [2, 2] ---
f_{[2, 2],1}= q^2
f_{[2, 2],2}= q^4


---Generating Major Index over Descent polynomials for  [2, 2, 1] ---
f_{[2, 2, 1],2}= q^6 + q^5 + q^4
```

```
f_{[2, 2, 1],3}= q^8 + q^7


---Generating Major Index over Descent polynomials for  [3] ---
f_{[3],0}= 1


---Generating Major Index over Descent polynomials for  [3, 1] ---
f_{[3, 1],1}= q^3 + q^2 + q


---Generating Major Index over Descent polynomials for  [3, 1, 1] ---
f_{[3, 1, 1],2}= q^7 + q^6 + 2*q^5 + q^4 + q^3


---Generating Major Index over Descent polynomials for  [3, 2] ---
f_{[3, 2],1}= q^3 + q^2
f_{[3, 2],2}= q^6 + q^5 + q^4


---Generating Major Index over Descent polynomials for  [3, 2, 1] ---
f_{[3, 2, 1],2}= q^8 + 2*q^7 + 2*q^6 + 2*q^5 + q^4
f_{[3, 2, 1],3}= q^11 + 2*q^10 + 2*q^9 + 2*q^8 + q^7
```

A.2 Major index over descent via $K^k_{\lambda,1^{|\lambda|}}(q)$

A.2.1 Code

```
from sage.all import *
from sage.combinat.q_analogues import q_binomial
import sys
import argparse
from datetime import datetime
import time
import numpy
from sage.rings.polynomial.polydict import PolyDict
import gc
```

```
R, q = QQ['q'].objgen()
q    = QQ['q'].gen()
R, q = objgen(QQ['q'])
q    = gen(QQ['q'])


#  Binomial(n,k) defined as
#     binomial(n,k) if integers n,k>=0
#     binomial(n,k) for all non-integers
#     0             if one of integers n,k<0
def Binomial(n, k):
    if n in ZZ and k in ZZ:
        if n >= 0 and k >= 0:
            return binomial(n, k)
        return 0
    return binomial(n, k)

# q-Analog of Binomial(n,k).
def QBinomial(n, k):
    if n in ZZ and k in ZZ:
        if n >= 0 and k >= 0:
            return q_binomial(n, k)
        return 0
    return q_binomial(n, k)

#  Generates valid sequences used to generate the Kostka ↩
   polynomial representing the distribution of the charge ↩
   statistic
#  across all SYT of shape lambda with exactly k descents.
#
#           Let lambda = (\lambda_1, ..., \lambda_r) be a ↩
   partition of n.
#           Let mu = (1,1,...,1) be a partition of n, so then↩
    mu' = (n).
#
#           A valid sequence of partitions is alpha = (alpha↩
   ^0, alpha^1, alpha^2, ..., alpha^r)
#           subject to the following constraints:
#               alpha^0 = mu' = (n)
#               For i=1, the first part of alpha^1, is always↩
    k.
#               For i>=1, alpha^i is a partition of sum_{j>= ↩
   i+1} lambda_j.
#                   Note that this means alpha^r is the empty↩
    partition.
#                   This is to prevent indexing issues.
```

```
def gen_alpha_seqs_new(shape, k):
    ptn = Partition(shape)
    mu = Partition([1]*ptn.size())
    muConj = list(mu.conjugate())
    r = len(ptn)

    bigg = []

    for i in range(0,r):
        bigg.append([]) ## seq0, seq1, ... seqr

    bigg[0].append([muConj])   ## In seq0, create an initial ↩
       sequence alpha = (mu')

    # Build everything else.
    for i in range(1,r):

        # Generate all of the partitions of size sum_{j>=i+1}↩
            \lambda_j.
        # In sage, that's sum_{j>=i \lambda_j}.
        ptnsize = sum( ptn[j] for j in range(i,r))
        ptns = Partitions(ptnsize)
        # For every sequence alpha, the first part of alpha_1↩
            in alpha is k.
        if i==1:
            valid = []

            for par in ptns:
                p = list(par)
                if p[0]==k:
                    valid.append(p)

            for par in valid:
                p = list(par)
                for alpha in bigg[i-1]:   # For every ↩
                   sequence of partitions alpha in Seq(i-1)
                    current = copy(alpha)
                    current.append(p)
                    bigg[i].append(current)

        if i>1:
            for par in ptns:
                p = list(par)
                for alpha in bigg[i-1]:   # For every ↩
                   sequence of partitions alpha in Seq(i-1)
                    current = copy(alpha)
                    current.append(p)
```

```
                    bigg[i].append(current)
        gc.collect()

    res = []
    [res.append(x) for x in bigg[len(bigg)-1] if x not in res]

    return res

#  Given a valid sequence alpha for the fixed descent charge statistic Kostka polynomial,
#  computes P^{a}_i(alpha), with sequence index 'a' and part index 'i', where
#           P^{a}_i(alpha) = sum_{j=1}^i ( alpha^{a-1}_j - 2*alpha^{a}_j + alpha^{a+1}_j )
#
def p_alpha(alpha, a, i):
    total = 0
    for j in range(0,i+1):
        total += alpha_part(alpha,a-1,j) - 2*alpha_part(alpha,a,j) + alpha_part(alpha,a+1,j)
    return total

# Given a valid sequence alpha for the fixed descent charge statistic Kostka polynomial,
# computes  c(alpha), a charge-like weighting for the sequence, where
#           c(alpha) = sum_{a=1}^{r} sum_{i=1}^{length(alpha^{a})} binomial( alpha^{a-1}_i - \alpha^{a}_i , 2)
#
def c_weight_alpha(alpha):
    weight = 0

    for a in range(1,len(alpha)+1):
        alphaA = alpha[a] if a<len(alpha) else [0]
        alphaAMinus = alpha[a-1]

        imax = max( len(alphaA), len(alphaAMinus))
        for i in range(0,imax+1):
            currentpart = alphaA[i] if i<len(alphaA) else 0
            prevpart    = alphaAMinus[i] if i<len(alphaAMinus) else 0

            bino = binomial(prevpart - currentpart , 2)
            weight += bino
    return weight
```

```
# Given a sequence alpha of partition, returns the part alpha^a_i if possible.
# If impossible, returns 0.
#
def alpha_part(alpha, a, i):
    return alpha[a][i] if a<len(alpha) and i<len(alpha[a]) else 0

#  Given a partition lambda and integer value k, generates the Kostka polynomial representing
#  the distribution of the charge statistic across all SYT of shape lambda with exactly k descents.
def charge_poly_fixed_desc(ptn, k):
    if len(ptn) == 1:
        return 0*q
    shape = []
    for part in ptn:
        shape.append(part)

    sequences = gen_alpha_seqs_new(shape, k)
    poly = 0

    for alpha in sequences:
        poly += q**(c_weight_alpha(alpha)) * prodterm(alpha)
    return poly

# Given a sequence alpha, generate the terms in the product of KR formula.
def prodterm(alpha):
    prodterm = 1
    for a in range(1,len(alpha)):
        for i in range(0,max(len(alpha[a-1]),len(alpha[a]))):
            qupper = p_alpha(alpha,a,i) + alpha_part(alpha,a,i) - alpha_part(alpha,a,i+1)
            qlower = alpha_part(alpha,a,i) - alpha_part(alpha,a,i+1)
            prodterm = prodterm * QBinomial(qupper,qlower)

    return prodterm

# Given a sequence alpha, generate the terms in the product of KR formula, with information
# about the q-binomials used.
def prodterm_info(alpha):
    prodterm = 1
    fancy = []
```

```
    for a in range(1,len(alpha)):
        fancy.append("  (a={})".format(a))
#         for i in range(0,max(len(alpha[a-1]),len(alpha[a]))↩
   ):
        for i in range(0, max(len(alpha[1]),len(alpha[2]))):
            qupper = p_alpha(alpha,a,i) + alpha_part(alpha,a,↩
               i) - alpha_part(alpha,a,i+1)
            qlower = alpha_part(alpha,a,i) - alpha_part(alpha↩
               ,a,i+1)
            term = QBinomial(qupper, qlower)
            fancyterm = [qupper, qlower]
            fancy.append(fancyterm)
            prodterm = prodterm * term
    return [prodterm, fancy]

# Given a partition shape "shape", generate the nonzero f_{↩
   lambda, k}(q) polynomials for all SYT of shape
# lambda inside "shape".
def gen_partitions_inside(shape=[]):
    ptn = Partition(shape)
    r = len(shape)
    n = ptn.size()
    print("Calculating partitions within {}, a partition of ↩
       {}, with {} rows".format(shape,n,r))

    new = deepcopy(shape)

    insideList=[ptn]

    for i in range(0,n):
        lenIn = len(insideList)
        for j in range(0,lenIn):
            lil = insideList[j]
            if (lil.size() == n-i):
                insideList.extend(lil.down_list())
                insideList = sorted(list( dict.fromkeys(↩
                   insideList) ))

    insideList = sorted(list( dict.fromkeys(insideList) ))
    print("Partitions inside {}: {}".format(ptn,insideList))
    return insideList

# Use KR formla to generate all nonzero f_{ptn, k}
def gen_all_flambda_by_kostka(ptn):
    minn = len(ptn)-1
    maxx = sum(ptn)-ptn[0]
    polylist = [0 for n in range(minn,maxx+1)]
```

```
    outlist = []

    for des in range(minn,maxx+1):
        polylist[des-minn] = q**Binomial(sum(ptn),2)*↩
           charge_poly_fixed_desc(ptn, des)(q**(-1))

    for i in range(0,len(polylist)):
        f = "f_{{{},{}}}".format(ptn, i+minn)
        outlist.append([f, polylist[i]])
        print("{}= {}".format(f,polylist[i]))
    print("\n")

    return outlist

# Use KR formla to generate all nonzero f_{lambda, k} for all↩
    partitions lambda inside 'shape'.
def gen_all_flambda_inside_by_kostka(shape):
    ptns = gen_partitions_inside(shape)
    outlist = []

    for ptn in ptns:
        if len(ptn)>=1:
            out = gen_all_flambda_by_kostka(ptn)
            outlist.append(out)
    return outlist

# Libary helper function (removes extraneous info when trying↩
    to write list of all polynomials)
def librarize(inList):
    out = []
    for i in inList:
        for j in i:
            out.append(j[-1])
    return out
```

A.3 Admissible sequence contributions in $K^{k}_{\lambda,1^{|\lambda|}}(q)$

To run the code in this section, we need to utilize the code from the previous section. It has been omitted here to save space.

A.3.1 Code

```
# Return minimal charge + associated sequence(s) for a
# given partition shape and number of descents
def minimal_charge(shape, desc):
    sequences = gen_alpha_seqs_new(shape, desc)

    minCharge = float('inf')
    minAlpha = []

    for alpha in sequences:
        charge = c_weight_alpha(alpha)
        prod = prodterm(alpha)
        if prod == 0:
            continue
        minCharge = min(charge, minCharge )

        if (charge == minCharge):
            newMinAlpha = [alpha]

        for beta in minAlpha:
            if (c_weight_alpha(beta) == charge):
                newMinAlpha.append(beta)

        minAlpha = newMinAlpha

    return [minCharge, minAlpha]

# Return maximal charge + associated sequence(s) for a given ↩
   partition shape
# and number of descents
def maximal_charge(shape, desc):
    sequences = gen_alpha_seqs_new(shape, desc)

    maxCharge = float('-inf')
    maxAlpha = []

    for alpha in sequences:
        charge = c_weight_alpha(alpha)
        prod = prodterm(alpha)
        if prod == 0:
            continue

        maxCharge = max(charge, maxCharge )
```

```
        if (charge == maxCharge):
            newMaxAlpha = [alpha]

        for beta in maxAlpha:
            if (c_weight_alpha(beta) == charge):
                newMaxAlpha.append(beta)

        maxAlpha = newMaxAlpha

    return [maxCharge, maxAlpha]

# Breaks down construction of f_{shape, desc} by writing ↩
   contribution of each term in KR
def flambda_dissect_by_charge(shape, desc):
    sequences = gen_alpha_seqs_new(shape, desc)

    fullgf = []
    majorbydescpoly = 0
    print("========== Charge(Alpha) Weight Breakdown for {} ↩
       with {} descents ==========\n "
        .format(shape,desc))
    for alpha in sequences:
        charge = c_weight_alpha(alpha)
        fancyprod = prodterm_info(alpha)
        product = fancyprod[0]*(q**0)
        maj = q**((binomial(sum(shape),2)) - charge)
        majpoly = maj * product(q**(-1))
        ff = R(majpoly)
        if majpoly == 0:
            continue
        else:
            print("Charge poly: {}".format(product * q**↩
               charge))
            shift = ff.exponents()[0] if ff != 0 else 0
            print("Sequence: {}".format(alpha))
            print("CWeight: {}".format(charge))
            print("Contr. Product: {} *{}".format(shift, ↩
               fancyprod[1]))
            print("Contr. Poly: {}".format(majpoly))
            majorbydescpoly += majpoly

            fullgf.append([shift, fancyprod[1]])

            print("---------------------------\n ")
```

```
    print("Generating Function f_({},{}): ".format(shape, desc))
    print("{}*{} ".format(q**fullgf[0][0], fullgf[0][1]))
    for i in range(1,len(fullgf)):
        print("+ {}*{}".format(q**fullgf[i][0], fullgf[i][1]))
    print("=\n {}".format(majorbydescpoly))
    print("\n==========================================================\n ")
    return [fullgf, majorbydescpoly]

# Applies flambda_dissect_by_charge to partitions whose Ferrer's diagram fits inside of shape's,
# for all valid numbers of descents.
def dissect_flambda_inside_by_charge(shape):
    ptns = gen_partitions_inside(shape)
    outlist = []

    for ptn in ptns:
        if len(ptn)>=1:
            for des in range( len(ptn)-1, ptn.size()-ptn[0]+1):
                out = flambda_dissect_by_charge(ptn, des)
                outlist.append(out)
    return outlist

# Writes admissible sequences in KR construction of f_{shape,desc}.
def admissible_sequences(shape, desc):
    sequences = gen_alpha_seqs_new(shape, desc)

    zero =[]
    print("=========== Admissible Sequences for {} with {} descents ==========\n "
        .format(shape,desc))
    for alpha in sequences:
        charge = c_weight_alpha(alpha)
        fancyprod = prodterm_info(alpha)
        product = fancyprod[0]*(q**0)
        maj = q**((binomial(sum(shape),2)) - charge)
        majpoly = maj * product(q**(-1))
        ff = R(majpoly)
        if majpoly == 0:
            zero.append([alpha,fancyprod[1]])
            continue
        else:
```

```
            print("Sequence: {}".format(alpha))
            print("")
            print("---------Product: {}".format(prodterm_info↩
               (alpha)[1]))
            print("\n")
    print("\n")
    return
```

A.3.2 Examples

```
# flambda_dissect_by_charge([4,2,2],4)

========== Charge(Alpha) Weight Breakdown for [4, 2, 2] with 4 descents ==

Charge poly: q^16 + q^15 + 2*q^14 + 2*q^13 + 3*q^12 + 2*q^11 + 2*q^10 + q^9
  + q^8
Sequence: [[8], [4], [2]]
CWeight: 8
Contr. Product: 12 *['  (a=1)', [6, 4], '  (a=2)', [2, 2]]
Contr. Poly: q^20 + q^19 + 2*q^18 + 2*q^17 + 3*q^16 + 2*q^15 + 2*q^14
  + q^13 + q^12
---------------------------

Charge poly: q^14 + q^13 + q^12 + q^11 + q^10
Sequence: [[8], [4], [1, 1]]
CWeight: 10
Contr. Product: 14 *['  (a=1)', [5, 4], [2, 0], '  (a=2)', [2, 0], [1, 1]]
Contr. Poly: q^18 + q^17 + q^16 + q^15 + q^14
---------------------------

Generating Function f_([4, 2, 2],4):
q^12*['  (a=1)', [6, 4], '  (a=2)', [2, 2]]
+ q^14*['  (a=1)', [5, 4], [2, 0], '  (a=2)', [2, 0], [1, 1]]
=
 q^20 + q^19 + 3*q^18 + 3*q^17 + 4*q^16 + 3*q^15 + 3*q^14 + q^13 + q^12

================================================================================
```

```
# admissible_sequences([5,3,2],4)
```

```
================ Admissible Sequences for [5, 3, 2] with 4 descents ==========

Sequence: [[10], [4, 1], [2]]

---------Product: ['  (a=1)', [7, 3], [3, 1], '  (a=2)', [2, 2], [1, 0]]


Sequence: [[10], [4, 1], [1, 1]]

---------Product: ['  (a=1)', [6, 3], [3, 1], '  (a=2)', [2, 0], [2, 1]]
```

A.4 Relationships among $f_{\lambda,k}(q)$

The following code performs the division algorithm discussed in Chapter 4 for $f_{\lambda,3}(q)$ and $f_{\lambda,2}(q)$. We note that outside of implementation, there is nothing preventing one from slightly modifying the code to accommodate relations among different numbers of descents or even among different partition shapes. Each major function here has a *mathematicaOutput* toggle, which will form lists compliant with Mathematica's syntax ([23]) for those who prefer processing polynomials in the Wolfram language. As with the previous section, we use many of the same underlying functions given in Appendix A.2.

A.4.1 Code

```
#
#  poly: a polynomial in q
#      returns string of polynomial with "nice" ordering of ↩
   terms.
#
def poly_proper(poly):
    poly = (poly*q**0).subs(q=z)
```

```
    return " + ".join(map(str,sorted([f for f in poly.↩
       operands()],key=lambda exp:exp.degree(z)))).replace("z↩
       ","q")

#   Relationship test for min vs. min+1 descents.
#   f3 -  f_{lambda, 3}
#   f2 -  f_{lambda, 2}
#
#   printType - Printing Style Toggles
#     full       - Print f3 = q f2 + r
#     quotient   - Print quotient  g
#     remainder  - Print remainder r
#
#   useIndicator
#     enable to display a black square for non-multiples, ↩
   white square for multiples.
def relationship_test(f3, f2, printType, useIndicator):
    poly = R(f3)
    g = 0
    while ((poly.coefficients() != []) and (min(poly.↩
       coefficients()) > 0)):

        term = q**(min(poly.exponents()) - min(f2.exponents()↩
           ))
        test = poly - term*f2

        if( (test.coefficients() ==[]) or (min(test.↩
           coefficients()) > 0)):
            poly = test
            g+= term
        else:
            break

    r = f3-g*f2

    propR = poly_proper(r)
    propG = poly_proper(g)
    indicator = '   ' if ((r)!=0 and useIndicator) else '   '↩
        if (r== 0 and useIndicator) else ' '

    if(printType == 'full'):
        print("{}f_3 = ({})f_2+({})".format( indicator,  str(↩
           propG).replace('*',''), str(propR).replace('*','')↩
           ))

    if(printType == 'quotient'):
```

```
        print("{}({}),".format( indicator, str(propG).replace↩
            ('*','')))

    if(printType == 'remainder'):
        print("{}({}),".format( indicator, str(propR.replace(↩
            '*',''))))

    return [R(g),R(r)]


# Example Remainder Search for min vs. min+1 descents.
#     Fix j, range over k.
#     For each k, give remainders when n is in congruence ↩
   classes mod (modulus)
#     continue to n=modulus*nprime+modulus where nprime=capN
#     mathematicaOutput formats things nicely to copy-paste ↩
   into Mathematica.
def hunt_remainder(j, modulus, capK, capN, mathematicaOutput)↩
   :
    print("--j={}".format(j))
    for k in range(j,capK):
        print("---Letting: k={}".format(k))
        for ncong in range(0,modulus):
            print("----Class:   n={} (mod {})".format(ncong, ↩
               modulus), end='')
            first = True
            for nprime in range((k/modulus).floor(),capN):
                n = modulus*nprime+ncong
                if (n >= k and k>= j):
                    flambda = gen_all_flambda_by_kostka([n,k,↩
                       j])
                else:
                    continue
                if (len(flambda) < 2):
                    continue

                if(first):
                    print(", so n= {}, {}, {}, {}, ...".↩
                       format(n, n+modulus, n+2*modulus, n+3*↩
                       modulus))
                    if(mathematicaOutput):
                        print("list$J{}K{}N{}mod{}=".format(j↩
                           ,k,ncong,modulus),end='{\n')
                first = False

                f2 = R(flambda[0][1])
                f3 = R(flambda[1][1])
```

```
                if(not mathematicaOutput):
                    print("[{},{},{}]:".format(n,k,j),end='')
                p = relationship_test(f3, f2, 'remainder', ↩
                   False)
            if(mathematicaOutput):
                print("}")
            else:
                print("")
        print("")

# Example Quotient Search for min vs. min+1 descents.
# (modulus*n+congclass,modulus*n+congclass,j) Quotient Search
#     Fix n,k, range over j.
#     For each j, give quotients and generate list of ↩
   differences between successive j.
#     Do so for n=k=modulus*nprime+congclass up to nprime=↩
   nprimeCap
#     mathematicaOutput formats things nicely to copy-paste ↩
   into Mathematica.
def hunt_quotient_nnj(modulus, congclass, nprimeCap, jCap, ↩
   mathematicaOutput):
    for nprime in range(0,nprimeCap+1):
        n = modulus*nprime + congclass
        k = n
        quotients=[]

        print("Letting n,k={}*{}+{}={}".format(modulus,nprime↩
           ,congclass,n))
        first = True

        for j in range(1,min(n+1,jCap+1)):
            if (n<k or k<j):
                continue

            flambda = gen_all_flambda_by_kostka([n,k,j])
            if (len(flambda) < 2):
                    continue

            if(first):
                if(mathematicaOutput):
                    print("list$NK{}$=".format(n),end='{\n')
                first = False

            f2 = R(flambda[0][1])
            f3 = R(flambda[1][1])
```

```
            if(not mathematicaOutput):
                print("[{},{},{}]:".format(n,k,j),end='')
            p = relationship_test(f3, f2, 'quotient', True)

            quotients.append(p[0])
        if(mathematicaOutput):
            print("}")
        diff = [poly_proper(t - s) for s, t in zip(quotients,↩
            quotients[1:])]
        print("\nSubsequent Differences:")
        if(mathematicaOutput):
            print("diff$NK{}=".format(n),end='{\n')
        print(',\n'.join(map(str, diff)),end=('\n}\n\n----' ↩
           if mathematicaOutput else '\n\n----'))
    print("")
```

A.4.2 Examples

```
# flist = gen_all_flambda_by_kostka([5,5,3])
# fmin = R(flist[0][1])
# fminplusone = R(flist[1][1])
# relationship_test(fminplusone, fmin, 'full', True)

■f_3 = (q^2 + 2q^3 + 2q^4 + 3q^5 + 4q^6 + 3q^7 + 3q^8 + 3q^9 + q^10
  + q^11)f_2+(q^23 + q^25)

[q^11 + q^10 + 3*q^9 + 3*q^8 + 3*q^7 + 4*q^6 + 3*q^5 + 2*q^4 + 2*q^3 + q^2,
 q^25 + q^23]


# hunt_quotient_nnj(3, 2, 2, 4, False)

Letting n,k=3*0+2=2
[2,2,1]:■(),
[2,2,2]:□(q^2 + q^3 + q^4),

Subsequent Differences:
q^2 + q^3 + q^4

----Letting n,k=3*1+2=5
[5,5,1]:■(q^3 + q^4 + q^5 + 2q^6 + q^7),
[5,5,2]:□(q^2 + q^3 + 2q^4 + 2q^5 + 3q^6 + 2q^7 + 2q^8 + q^9 + q^10),
```

```
[5,5,3]:■(q^2 + 2q^3 + 2q^4 + 3q^5 + 4q^6 + 3q^7 + 3q^8 + 3q^9 + q^10
    + q^11),
[5,5,4]:■(q^2 + 2q^3 + 3q^4 + 3q^5 + 5q^6 + 4q^7 + 4q^8 + 4q^9 + 3q^10
  + q^11 + q^12),

Subsequent Differences:
q^2 + q^4 + q^5 + q^6 + q^7 + 2*q^8 + q^9 + q^10,
q^3 + q^5 + q^6 + q^7 + q^8 + 2*q^9 + q^11,
q^4 + q^6 + q^7 + q^8 + q^9 + 2*q^10 + q^12

----Letting n,k=3*2+2=8
[8,8,1]:■(q^3 + q^4 + q^5 + 2q^6 + 2q^7 + 2q^8 + 3q^9 + 2q^10 + q^11
  + 2q^12 + q^13),
[8,8,2]:□(q^2 + q^3 + 2q^4 + 2q^5 + 3q^6 + 3q^7 + 4q^8 + 4q^9 + 4q^10
    + 3q^11
  + 3q^12 + 2q^13 + 2q^14 + q^15 + q^16),
[8,8,3]:■(q^2 + 2q^3 + 2q^4 + 3q^5 + 4q^6 + 4q^7 + 5q^8 + 6q^9 + 5q^10
    + 5q^11
  + 5q^12 + 3q^13 + 3q^14 + 3q^15 + q^16 + q^17),
[8,8,4]:■(q^2 + 2q^3 + 3q^4 + 3q^5 + 5q^6 + 5q^7 + 6q^8 + 7q^9 + 7q^10
    + 6q^11
  + 7q^12 + 5q^13 + 4q^14 + 4q^15 + 3q^16 + q^17 + q^18),

Subsequent Differences:
q^2 + q^4 + q^5 + q^6 + q^7 + 2*q^8 + q^9 + 2*q^10 + 2*q^11 + q^12 + q^13
  + 2*q^14 + q^15 + q^16,
q^3 + q^5 + q^6 + q^7 + q^8 + 2*q^9 + q^10 + 2*q^11 + 2*q^12 + q^13 + q^14
  + 2*q^15 + q^17,
q^4 + q^6 + q^7 + q^8 + q^9 + 2*q^10 + q^11 + 2*q^12 + 2*q^13 + q^14 + q^15
  + 2*q^16 + q^18

----


# hunt_remainder(3, 4, 4, 4, False)

--j=3
---Letting: k=3
----Class:   n=0 (mod 4), so n= 4, 8, 12, 16, ...
[4,3,3]: (18 + q),
[8,3,3]: (q^21 + q^22 + q^23 + 2q^24 + 2q^25 + 2q^26 + 2q^27 + q^28 + q^29
  + q^30),
[12,3,3]: ().

----Class:   n=1 (mod 4), so n= 5, 9, 13, 17, ...
```

```
[5,3,3]: (q^18 + q^19 + q^20 + q^21),
[9,3,3]: (),
[13,3,3]: (q^27 + q^29 + q^30 + q^31 + q^32 + 2q^33 + q^34 + 2q^35 + 2q^36
  + 2q^37 + q^38 + 2q^39 + q^40 + q^41 + q^42 + q^43 + q^45).

----Class:   n=2 (mod 4), so n= 6, 10, 14, 18, ...
[6,3,3]: (),
[10,3,3]: (q^24 + q^26 + q^27 + q^28 + q^29 + 2q^30 + q^31 + q^32 + q^33
  + q^34 + q^36),
[14,3,3]: (q^27 + q^28 + q^29 + 2q^30 + 2q^31 + 2q^32 + 3q^33 + 3q^34 + 3q^35
  + 4q^36 + 4q^37 + 4q^38 + 4q^39 + 3q^40 + 3q^41 + 3q^42 + 2q^43 + 2q^44
  + 2q^45 + q^46 + q^47 + q^48).

----Class:   n=3 (mod 4), so n= 3, 7, 11, 15, ...
[3,3,3]: (),
[7,3,3]: (q^21 + q^23 + q^24 + q^25 + q^27),
[11,3,3]: (q^24 + q^25 + q^26 + 2q^27 + 2q^28 + 2q^29 + 3q^30 + 3q^31 + 3q^32
  + 3q^33 + 2q^34 + 2q^35 + 2q^36 + q^37 + q^38 + q^39),
[15,3,3]: ().
```

Appendix B

Mathematica Code

This appendix gives Mathematica (Wolfram language [23]) code to write $f_{\lambda,k}(q)$ by generating the set $\mathrm{SYT}(\lambda, k)$ and printing it out either as a list of diagrammatic tableaux by major index or as the polynomial $f_{\lambda,k}(q)$. We note that the tools in Appendix A could be written in the Wolfram language instead, though Mathematica does not include many of the typical statistics on tableaux by default.

B.1 Major index over descent via $\mathrm{SYT}(\lambda, k)$

B.1.1 Code

```
Needs["Combinatorica`"]

(*
Handle combinatorial issues with 0^0.
*)
Unprotect[Power];
Power[0 | 0., 0 | 0.] = 1;
Protect[Power];

(*
Fixes combinatorial issues with Mathematica's QBinomial,
returns zero if we have negative index.
*)
GenQBinomial[a_, b_, c_] := If[(a < b) || (a < 0), 0, QBinomial[a, b, c]]

(*
Given a tableau Tbx, return the number of descents of Tbx
*)
TabDes[Tbx_] := Sum[
  Which[Position[Tbx, i][[1]][[1]] < Position[Tbx, i + 1][[1]][[1]], 1,
   Position[Tbx, i][[1]][[1]] ≥ Position[Tbx, i + 1][[1]][[1]], 0],
  {i, 1, Max[Tbx] - 1}]

(*
Given a tableau Tbx, return the major index of Tbx
*)
TabMaj[Tbx_] := Sum[Which[Position[Tbx, i][[1]][[1]] < Position[Tbx, i + 1][[1]][[1]], i,
   Position[Tbx, i][[1]][[1]] ≥ Position[Tbx, i + 1][[1]][[1]], 0],
  {i, 1, Max[Tbx] - 1}]
```

```
(*
Given a partition part and integer des, return f_{part,des}, the major index statistic
 over all SYT of shape part with exactly des descents.
*)
FLambdaI[part_, des_] := Sum[Which[TabDes[Jay] ≠ des, 0, TabDes[Jay] == des,
    q^TabMaj[Jay]], {Jay, Tableaux[part]}];

(*
Diagramatically write out all SYT of shape part with given number of descents.
*)
InfoFLambdaI[part_, des_] := SortBy[DeleteCases[Table[
       Which[TabDes[Jay] ≠ des, 0, TabDes[Jay] == des, {Jay, q^TabMaj[Jay]}],
       {Jay, Tableaux[part]}], 0], Last] // Transpose // TableForm;
(*
Given a polynomial poly and a desired maximum power PowersBound, returns EulerProduct
 that matches up to bound.
  Sequence {a_i}, 1\leq i\leq bound, with
      a_i = j if 1/(1-q^i)^j appears and a_i =-j if (1-q^i)^j appears.
    Gives shift q^k if first nonzero term has degree k.
*)
QProdMake[Poly_, PowersBound_] := Module[{i, n},
  LocalF = Poly;
  FirstTerm = Poly /. q → 0;
  FirstDegree = 0;
  While[FirstTerm == 0, LocalF = Simplify[LocalF / q];
   FirstTerm = LocalF /. q → 0; FirstDegree = FirstDegree + 1];
  LFCoeffList = PadRight[CoefficientList[Series[LocalF / FirstTerm,
      {q, 0, PowersBound}], q], PowersBound];
  For[n = 1, n < PowersBound + 1 - FirstDegree, n++, If[! IntegerQ[LFCoeffList[[n]]],
    Print["Non-integer coefficients after division by leading coefficient"];
    Return[Null]]];
  EtaPowers = Table[0, {n, 1, PowersBound - FirstDegree}];
  For[n = 1, n < PowersBound + 1 - FirstDegree, n++,
   EtaPowers[[n]] = Coefficient[
     Series[LocalF / FirstTerm - Product[(1 - q^k) ^ (-EtaPowers[[k]]), {k, 1, n - 1}],
      {q, 0, PowersBound}], q^n]
  ];
  Print["First term ", FirstTerm * q^FirstDegree, " ; Qn powers ", EtaPowers];
  Return[Null]
 ]
```

```
(*
Computer-readable form of QProdMake. Spits out list of powers (used in QProdMake).
   Drops the shift term.
*)
PowerList[Poly_, PowersBound_] := Module[{i, n},
  LocalF = Poly;
  FirstTerm = Poly /. q → 0;
  FirstDegree = 0;
  While[FirstTerm == 0, LocalF = Simplify[LocalF / q]; FirstTerm = LocalF /. q → 0;
   FirstDegree = FirstDegree + 1];
  LFCoeffList = PadRight[CoefficientList[
     Series[LocalF / FirstTerm, {q, 0, PowersBound}], q], PowersBound];
  For[n = 1, n < PowersBound + 1 - FirstDegree, n++,
   If[! IntegerQ[LFCoeffList[[n]]], Return[Null]]];
  EtaPowers = Table[0, {n, 1, PowersBound - FirstDegree}];
  For[n = 1, n < PowersBound + 1 - FirstDegree, n++,
   EtaPowers[[n]] =
    Coefficient[Series[LocalF / FirstTerm - Product[(1 - q^k) ^ (- EtaPowers[[k]]),
        {k, 1, n - 1}], {q, 0, PowersBound}], q^n]
  ];
  Return[EtaPowers]
 ]
```

B.1.2 Examples

```
TabMaj[{{1, 4, 6}, {2, 5}, {3}}]
```

7

```
TabDes[{{1, 4, 6}, {2, 5}, {3}}]
```

3

```
FLambdaI[{3, 2, 1}, 3]
```

$q^7 + 2\,q^8 + 2\,q^9 + 2\,q^{10} + q^{11}$

```
InfoFLambdaI[{3, 2, 1}, 3]
```

1 4 6 2 5 3	1 3 6 2 4 5	1 4 5 2 6 3	1 3 5 2 4 6	1 3 5 2 6 4	1 2 5 3 6 4	1 3 4 2 5 6	1 2 4 3 5 6
q^7	q^8	q^8	q^9	q^9	q^{10}	q^{10}	q^{11}

```
QProdMake[FLambdaI[{3, 2, 1}, 2], 20]
```

```
First term q^4 ; Qn powers {2, -1, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

Note that $f_{(3,2,1),2}(q) = q^4 \frac{(1-q^2)\cdot(1-q^4)}{(1-q)^2}$.