Fermionic formulas for superconformal characters and unrestricted Kostka polynomials

By

LIPIKA DEKA
B.A. (University of Cambridge, UK) June 1999
M.S. (University of Wisconsin, Madison, USA) June 2001

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

MATHEMATICS

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Anne Schilling (Chair)

Jesús A. De Loera

Greg Kuperberg

Committee in Charge

To my parents

Contents

Abstract

The problem of finding fermionic formulas for the many generalizations of Kostka polynomials and for the characters of conformal field theories has been a very exciting research topic for the last few decades. In this thesis we present new fermionic formulas for the unrestricted Kostka polynomials extending the work of Kirillov and Reshetikhin. We also present new fermoinic formulas for the characters of $N = 1$ and $N = 2$ superconformal algebras which extend the work of Berkovich, McCoy and Schilling.

Fermionic formulas for the unrestricted Kostka polynomials of type $A^{(1)}_{n-1}$ in the case of symmetric and anti-symmetric crystal paths were given by Hatayama et al. We present fermionic formulas for the unrestricted Kostka polynomials of type $A^{(1)}_{n-1}$ for all crystal paths based on Kirillov-Reshetihkin modules. Our formulas and method of proof even in the symmetric and anti-symmetric cases are different from the work of Hatayama et al. We interpret the fermionic formulas in terms of a new set of unrestricted rigged configurations. For the proof we give a statistics preserving bijection from this new set of unrestricted rigged configurations to the set of unrestricted crystal paths which generalizes a bijection of Kirillov and Reshetikhin.

We present fermionic formulas for the characters of $N = 1$ superconformal models $SM(p', 2p + p')$ and $SM(p', 3p' - 2p)$, and the $N = 2$ superconformal model with central charge $c = 3(1 - \frac{2p}{p'})$. The method used to derive these formulas is known as Bailey flow. We show Bailey flows from the nonunitary minimal model $M(p, p')$ with p, p' coprime positive integers to $N = 1$ and $N = 2$ superconformal algebras. We derive a new Ramond sector character formula for the $N = 2$ superconformal algebra with central charge $c = 3(1 - \frac{2p}{p'})$ and calculate its fermionic formula.

Acknowledgments

This work was supported in part by the U.S. National Science Foundation under Grants DMS-0200774.

Chapter 1

Introduction

1.1 Summary of the main results

Fermionic formulas have been widely researched in mathematics and physics. In this thesis we consider two problems regarding fermionic formulas that arise in the context of combinatorial representation theory and conformal field theory (CFT). This thesis is based on two papers that resulted from research performed with Prof. Anne Schilling during my years of graduate school at the University of California, Davis.

In Chapter 2, we present a new fermionic formula for the unrestricted Kostka polynomials. This work is based upon the paper "New Fermionic formula for the unrestricted Kostka polynomials" with Anne Schilling. An extended abstract of this paper has appeared in the proceedings of 17th International conference, *Formal Power Series and Algebraic Combinatorics* 2005, held at the University of Messina, Italy, in June 2005. The full version of the paper is available as a preprint at `http://front.math.ucdavis.edu/math.CO/0509194`. We have submitted this paper for publication to *The Journal of Combinatorial Theory, Series A*. Our results extend the work of Kerov, Kirillov and Reshetikhin [44, 47] who used the Bethe Ansatz to find a fermionic formula for the Kostka polynomials. This was first extended in [48] to generalized Kostka polynomials by establishing a bijection between the highest weight paths in the tensor products of Kirillov-Reshetikhin crystals of type A_n and rigged configurations. We prove our new formula for the unrestricted Kostka polynomial case by giving a statistics preserving algorithmic bijection between all crystal paths in the tensor products of Kirillov-Reshetikhin crystals of type A_n and the corresponding set of rigged configurations. An explicit description of the new set of rigged configurations is presented, which is called the set of unrestricted rigged configurations. Our formula when restricted to symmetric and anti-symmetric crystals is different from the results of Hatayama et al. [31] where fermionic formulas are given for the unrestricted Kostka polynomials in these special cases.

In Chapter 3, we present new fermionic formulas for the characters of $N = 1$ and $N = 2$ superconformal algebras using the method of Bailey construction. The work in Chapter 3 is based on the paper "Non Unitary minimal models, Bailey's lemma and $N = 1, 2$ superconformal algebras" with Anne Schilling. This paper is published in *Communications in Mathematical Physics*, Volume 260, number 3 (2005) 711-725. We show that there are Bailey flows from the nonunitary minimal models $M(p, p')$ for arbitrary coprime positive integers p, p' to $N = 1$ and $N = 2$ superconformal models. The superconformal models are also indexed by a pair of coprime positive integers (p, p'). Denote the $N = 1$ superconformal algebras by $SM(p, p')$ and $N = 2$ superconformal algebras by $\mathcal{A}(p, p')$. We find Bailey flows specifically from the model $M(p, p')$ to $SM(2p + p'p')$, $SM(p', 3p, -2p)$ and to $\mathcal{A}(p, p')$ with central charge given by $3(1 - \frac{2p}{p'})$. Using the known fermionic formulas for the minimal models $M(p, p')$ [11], we explicitly calculate the fermionic formulas for the characters of $SM(2p + p'p')$, $SM(p', 3p, -2p)$ and $\mathcal{A}(p, p')$. Moreover, we derive a new Ramond sector character for $N = 2$ superconformal algebras and calculate its fermionic formula.

The new bijection given in Chapter 2 as well as its inverse have been implemented as C++ programs and are included in Chapter 4. In early stages of the project on unrestricted Kostka polynomials, these programs were used extensively to produce data and to verify conjectures regarding the unrestricted rigged configurations. The progams have also been incorporated into MuPAD-Combinat [58] as a dynamic module by Francois Descouens. In Chapter 4, we describe three programs which were used to verify different parts of our conjectures for our results presented in Chapter 2. The programs in Sections 4.1, 4.2 and 4.3 are designed for use by anyone who would like to do calculations using the bijection or the inverse bijection. Working out the bijection for even a small example is time-consuming and is very tedious. Therefore, we believe that these programs are very helpful to anyone studying unrestricted Kostka polynomials. The program in Section 4.1 also calculates the unrestricted Kostka polynomials.

Having stated the main results, it is worth mentioning that the bridge between the two papers is fermionic formulas one of which appears in the context of combinatorial representation theory and the other appears in conformal field theories (CFTs). The following section provides a brief background on fermionic formulas, Kostka polynomials and CFTs.

1.2 Background and motivation

A partition $\lambda = (\lambda_1, \cdots, \lambda_k)$ is a k-tuple of positive integers satisfying $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k \geq 0$. Let $l(\lambda)$ be the length of the partition λ which is the number of nonzero parts. In symmetric functions theory, the ring of symmetric functions have various bases including the monomial symmetric functions, Schur functions, and Hall-Littlewood symmetric functions [55]. The Kostka polynomial $K_{\lambda\mu}(q)$, indexed by two partitions λ and μ is defined as the matrix elements of the transition matrix between the Schur functions $s_\lambda(X)$ and the Hall Littlewood symmetric functions $P_\mu(X;q)$. That is:

$$s_\lambda(X) = \sum_\mu K_{\lambda\mu}(q) P_\mu(X;q). \tag{1.2.1}$$

In representation theory, the Kostka polynomials $K_{\lambda\mu}(q)$ are a q-analog of the multiplicity of the irreducible sl_n representation V_λ, indexed by the highest weight λ, in the expansion of the L-fold tensor product $V_{(\mu_1)} \otimes \cdots \otimes V_{(\mu_L)}$. Here $\mu = (\mu_1, \cdots, \mu_L)$ is a partition and $V_{(\mu_i)}$ is the symmetric tensor representation of sl_n with weight μ_i. These polynomials have been generalized in many ways in algebraic combinatorics. In some generalizations, for example [48, 51, 53, 76, 77, 78, 79, 80], the components of the tensor product are replaced by tensor representations which are not always symmetric. In some other generalizations [30, 33, 61, 62, 69, 74], the representations of sl_n are replaced by representations of other Kac-Moody algebras [34]. There are many combinatorial descriptions of Kostka polynomials. Lascoux and Schützenberger [52] gave the description

$$K_{\lambda\mu}(q) = \sum_{t \in \mathcal{T}(\lambda,\mu)} q^{c(t)}, \tag{1.2.2}$$

where $\mathcal{T}(\lambda, \mu)$ is the set of semi-standard Young tableaux [27, 55] of shape λ and content μ and where $c(t)$ [55] is the charge statistic of the tableau $t \in \mathcal{T}(\lambda, \mu)$. This expression proved the non-negativity of the coefficients of the Kostka polynomials as conjectured by H.O. Foulkes [26].

In the mid 1980's, Kirillov and Reshetikhin [47] used the Bethe Ansatz to obtain a new expression for the Kostka polynomials known as a fermionic formula. A fermionic formula is a q-polynomial or a q-series that is a specific sum of products of the q-binomial coefficients

$$\begin{bmatrix} m \\ n \end{bmatrix}_q = \frac{(q)_m}{(q)_n (q)_{m-n}}, \tag{1.2.3}$$

where

$$(q)_m = \prod_{k=1}^{m} (1-q^k) \quad \text{for} \quad m \in \mathbb{Z}_{>0} \quad \text{and} \quad (q)_0 = 1. \tag{1.2.4}$$

When $\mu = (1^L)$, the fermionic formula for the Kostka polynomial looks like

$$K_{\lambda,\mu}(q) = q^{\frac{L(L-1)}{2}} M(\lambda, \mu; q^{-1}) \tag{1.2.5}$$

where

$$\begin{aligned} M(\lambda, \mu; q) &= \sum_{\{m\}} q^{c(\{m\})} \prod_{1 \le a \le n, i \ge 1} \begin{bmatrix} p_i^{(a)} + m_i^{(a)} \\ m_i^{(a)} \end{bmatrix}_q, \\ c(\{m\}) &= \frac{1}{2} \sum_{1 \le a,b \le n} C_{a,b} \sum_{i,j \ge 1} \min(i,j) m_i^{(a)} m_j^{(b)}, \\ p_i^{(a)} &= L\delta_{a1} - \sum_{1 \le b \le n} C_{ab} \sum_{j \ge 1} \min(i,j) m_j^{(b)}. \end{aligned}$$

The $\{m\}$-indexed sum is over the set $\{m_i^{(a)} \in \mathbb{Z}_{\ge 0} | 1 \le a \le n, i \ge 1\}$ such that for $1 \le a \le n, i \ge 1$,

$$p_i^{(a)} \ge 0, \quad \sum_{i \ge 1} i m_i^{(a)} = \sum_{j > a} \lambda_j.$$

Here $(C_{ab})_{1 \le a,b \le n}$ is the Cartan matrix for sl_{n+1} and the partition λ has at most $n+1$ nonzero parts. The importance of the fermionic formula lies in the fact that there are no minus signs. Therefore, it can be used to study the limiting behavior, which is a key ingredient in finding different formulas for the characters related to affine Lie algebras and Virasoro algebras.

Some examples of such applications can be found in [31, 45].

To prove that the fermionic formula of the Kostka polynomial is given by equation (1.2.5), Kirillov and Reshetikhin [47] gave a bijection between the set $T(\lambda, \mu)$ and a new combinatorial object called rigged configurations. Rigged configurations index the solutions of the Bethe Ansatz equations and they are sequences of partitions satisfying certain size restrictions along with some labellings called riggings for the parts of the partitions. This connection between fermionic formula and Kostka polynomial is the beginning of a whole new era of research in combinatorial representation theory.

In 1997 Nakayashiki and Yamada [60] gave a different representation of the Kostka polynomials in terms of paths:

$$K_{\lambda\mu}(q) = \sum_{p \in \overline{\mathcal{P}}(\lambda,\mu)} q^{E(p)}, \tag{1.2.6}$$

where a path $p \in \overline{\mathcal{P}}(\lambda, \mu)$ is a highest weight element of weight λ in Kashiwara's crystal base [37] corresponding to the tensor product representation $V_{(\mu_1)} \otimes V_{(\mu_2)} \otimes \cdots \otimes V_{(\mu_L)}$ of sl_n. The statistic $E(p)$ associated with a path p is called energy. This new representation was derived by realizing that paths are in bijection with the set of rigged configurations. This bijection is done by sending a path (which can be viewed as a word in the sl_n case) to its Robinson-Schensted [27] recording Q-tableau, which is then sent to the rigged configuration using Kirillov-Reshetikhin bijection. The path form of the Kostka polynomials is particularly important because this definition can be generalized to any Kac-Moody Lie algebras using the crystal base theory. Therefore, the Kostka polynomial for any Kac-Moody Lie algebra is defined as the generating function of paths when graded by the energy statistic and is called the "one dimensional sum" X. The fermionic formula M for the "one dinemsional sum" was conjectured in full generality by Hatayama et al. in [30, 31]. This is known as the famous $X = M$ *conjecture*. Although this conjecure in full generality is still open, many special cases have been proved in a series of papers [61, 62, 69, 74].

Similar to the Kostka polynomials, the unrestricted Kostka polynomials $X_{\lambda\mu}(q)$, indexed by two partitions λ and μ, can be defined as the matrix elements of the transition matrix between the monomial symmetric functions and the modified Hall-Littlewood symmetric functions [46, 55]. Let λ be a partition with $l(\lambda) \leq n$, and let $P_\lambda(X_n : q)$ and $Q_\lambda(X_n; q)$ be the Hall-Littlewood polynomials [55]. A modified Hall-Littlewood polynomial $Q'_\lambda(X_n; q)$ is defined to be

$$Q'_\lambda(X_n; q) = Q_\lambda(X_n/(1-q); q), \tag{1.2.7}$$

where the variables $X_n/(1-q)$ are the products $xq^{j-1}, j \geq 1$ for $x \in X_n := (x_1, \cdots, x_n)$. Note that $Q'_\lambda(X; 0) = s_\lambda(X)$ and $Q'_\lambda(X; 1) = h_\lambda(X)$ where $h_\lambda(X)$ is the complete homogeneous symmetric function [55]. With this notation the Kostka polynomial can also be defined as

$$Q'_\lambda(X; q) = \sum_\mu s_\mu(X) K_{\mu\lambda}(q).$$

The unrestricted Kostka polynomial, $X_{\lambda\mu}(q)$ is then defined as

$$Q'_\lambda(X; q) = \sum_\mu X_{\lambda\mu}(q) m_\mu(X).$$

Combinatorially [30, 31, 33, 76, 80],

$$X_{\lambda\mu}(q) = \sum_{p \in \mathcal{P}(\lambda,\mu)} q^{E(p)} \tag{1.2.8}$$

where $\mathcal{P}(\lambda, \mu)$ is the set of all unrestricted paths of weight λ. Unrestricted paths are elements in the crystal base of the tensor product representation $V_{(\mu_1)} \otimes \cdots \otimes V_{(\mu_L)}$ of sl_n. The unrestricted Kostka polynomials we described above correspond to type A_{n-1} Lie algebras. One should note that the set of unrestricted paths of weight λ contains the set of highest weight paths with the same weight vector.

A fermionic formula for the A_{n-1} unrestricted Kostka polynomials, when μ is a sequence of row partitions or a sequence of column partitions, was proved in [31, 46]. The existence of crystal bases have been conjectured in [32, 33] for all Kirillov-Reshetikhin modules. A Kirillov-Reshetikhin module, $W^{r,s}$ is a finite dimensional module over an affine Lie algebra, which corresponds to the weight vector $s\Lambda_r$, where Λ_r is the fundamental weight of the affine Lie algebra. The corresponding crystal is denoted by $B^{r,s}$. For $A^{(1)}_{n-1}$, the affine Lie algebra of type A_{n-1} [34], the existence of the Kirillov-Reshetikhin crystals are known [43, 80]. In the type $A^{(1)}_{n-1}$ case, the weight vector $s\Lambda_r$ is a rectangular partition of height r and width s. Having the crystal basis, it is natural to extend the definition of unrestricted Kostka polynomials to tensor products of Kirillov-Reshetikhin modules using the path definition (1.2.8). The fermionic formula for the unrestricted Kostka polynomials of type $A^{(1)}_{n-1}$ in this general set up has not been studied until now. In Chapter 2 we study these unrestricted Kostka polynomials for tensor products of all Kirillov-Reshetikhin modules of type $A^{(1)}_{n-1}$ and present new fermionic formulas.

Recently, fermionic expressions for generating functions of unrestricted paths for type $A^{(1)}_1$ have also surfaced in connection with box-ball systems. Takagi [83] establishes a bijection between box-ball systems and a new set of rigged configurations to prove a fermionic formula for the q-binomial coefficient. His set of rigged configurations coincides with our set in the type $A^{(1)}_1$ case. There is a generalization of Takagi's bijection to type $A^{(1)}_{n-1}$ case [50]. Hence our bijection composed with the generalized Takagi's bijection establishes a new connection between box-ball systems and the unrestricted Kostka polynomials.

One of the motivations to seek an explicit expression for unrestricted Kostka polynomials is their appearance in generalizations of Bailey's lemma [7]. Bailey's lemma is a powerful method to prove Rogers-Ramanujan type identities [64, 65, 68]. The Bailey transform of [4] starts with a seed identity and produces an infinite family of identities. The original Bailey lemma corresponds to type A_1. In [76] a type A_n generalization of Bailey's lemma was conjectured which was subsequently proven in [86]. A type A_2 Bailey chain, which yields an infinite family of identities, was given in [6]. In these generalizations a key ingredient was an explicit fermionic formula for the unrestricted Kostka polynomial. If the method we used in this thesis to find the new fermionic formula for the unrestricted Kostka polynomial can be generalized to other Kac-Moody algebras, it might trigger further progress towards generalizations of the Bailey's lemma to Kac-Moody algebras other than type A_n. Unrestricted rigged configurations for the simply laced type Lie algebras have already been studied in [70].

In the physics context, finding explicit formulas for the characters of the the solvable lattice models has been a fundamental problem. The minimal models denoted by $M(p,p')$ are conformal field theories (CFT) invented by Belvin, Polyakov and Zamolodchikov [13, 14]. These are conformally invariant two dimensional field theories, which describe second order phase transitions. The symmetry algebras of these theories are the infinite dimensional algebras known as the Virasoro algebras. The Virasoro algebra is generated by generators L_m satisfying

$$[L_n, L_m] = (n-m)L_{m+n} + \frac{c}{12}(n^3-n)\delta_{n+m,0}, \quad \text{for} \quad m,n \in \mathbb{Z}, \tag{1.2.9}$$

where c is the central charge given by

$$c(p,p') = 1 - \frac{6(p'-p)^2}{pp'}, \tag{1.2.10}$$

where $1 \leq p < p'$ and p, p' are coprime. Hence the minimal models are indexed by p, p'. The conformal dimension for this model is given by

$$\Delta_{r,s}^{p,p'} = \frac{(p'r-ps)^2 - (p'-p)^2}{4pp'}, \tag{1.2.11}$$

where

$$1 \leq r \leq p-1, 1 \leq s \leq p'-1.$$

The characters of these models are calculated in [16, 19, 67] as

$$\chi_{r,s}^{(p'p')}(q) = q^{\Delta_{r,s}^{p,p'} - c/24} \frac{1}{(q)_\infty} \sum_{j=-\infty}^{\infty} (q^{j(jpp'+rp'-sp)} - q^{(jp'+s)(jp+r)}), \tag{1.2.12}$$

where $(q)_\infty = \prod_{i=1}^{\infty}(1-q^i)$. This expression is derived by the Feigin and Fuchs construction [22] of a Fock space using bosonic generators and hence known as a bosonic formula. In 1993, Kedem et al. [17, 40, 41, 42] found a new expression for such characters in their study of the three state Potts models. The new formula had no minus signs like the bosonic form. They interpreted the new formula as the partition function of quasi-particles satisfying fermionic exclusion principles and called the new expression fermionic.

The fermionic expression for the minimal models $M(p,p')$ are calculated in [11, 84], which have the following form:

$$\sum_{\mathbf{m} \quad \text{restriction}} q^{\frac{1}{4}\mathbf{m}^t B\mathbf{m}+A\mathbf{m}} \prod_{j=1}^{n} \begin{bmatrix} ((1-B)\mathbf{m}+\frac{\mathbf{u}}{2})_a \\ m_a \end{bmatrix}_q , \tag{1.2.13}$$

where $\mathbf{m}$ is an n component vector of non-negative integers which may be subject to restrictions in the sum, B is an $n \times n$ matrix, and A and $\mathbf{u}$ are n component vectors. Kedem et al. showed in [17, 40, 41, 42] that any expression of this form can be interpreted physically. Due to this reason finding fermionic formulas is a very important problem in physics. The character identities obtained by equating the bosonic and fermionic expression for the CFT characters are known as Bose-Fermi identities.

As mentioned earlier, the fermionic formulas for minimal models are very well studied but the fermionic formulas for other CFTs are not yet known in full generality. The $N = 1$ and $N = 2$ superconformal algebras are two classes of CFTs where the symmetry algebras are extended Virasoro algebras. Berkovich, McCoy and Schilling demonstrated in [10] that some of the characters of $N = 1$ and $N = 2$ superconformal algebras can be obtained from the minimal models $M(p-1, p)$ by means of a construction known as the Bailey's lemma. Bailey's Lemma first appeared in the paper [7] in 1949. Bailey observed this important result while trying to clarify Rogers second proof of Rogers-Ramanujan (RR) identities (1917). The first and second RR identities are

$$\sum_{n=0}^{\infty} \frac{q^{n^2}}{(q)_n} = \frac{1}{(q)_\infty} \sum_{n=-\infty}^{\infty} (q^{n(10n+1)} - q^{(5n+2)(2n+1)}) = \prod_{n=1}^{\infty} \frac{1}{(1-q^{5n-1})(1-q^{5n-4})} \tag{1.2.14}$$

$$\sum_{n=0}^{\infty} \frac{q^{n(n+1)}}{(q)_n} = \frac{1}{(q)_\infty} \sum_{n=-\infty}^{\infty} (q^{n(10n+3)} - q^{(5n+1)(2n+1)}) = \prod_{n=1}^{\infty} \frac{1}{(1-q^{5n-2})(1-q^{5n-3})} \tag{1.2.15}$$

There are many different proofs of these identities for example [54, 64, 65, 68, 71] and there are many generalizations [2, 15, 29, 81, 82] in the theory of partitions. It is worth mentioning that the equality of the first two expressions of (1.2.14) is the Bose-Fermi identity for the minimal model $M(2, 5)$. Therefore, Bose-Fermi identities can be interpreted as a generalization of the RR-identities.

Bailey's lemma has been a very useful method for proving RR-type q-identities. Slater [81, 82] used this lemma extensively to prove 130 RR-type identities. In connecting the Bailey construction to physics the most remarkable step was achieved when Foda and Quano [23, 24] derived identities for the Virasoro characters using Bailey's lemma. The method used is a constructive procedure which starts from a polynomial generalization of a Bose-Fermi identity of one CFT and produces a Bose-Fermi identity for the character of another CFT. This is known as Bailey flow. Hence new fermionic formulas for CFTs can be calculated via Bailey flow from known fermionic formulas of another CFT. The Bailey flow from $M(p-1, p)$ to $M(p, p+1)$ is presented in [10, 24] and further flows to some special $N = 1$ and $N = 2$ supersymmetric models are given in [10]. This led us to investigate about Bailey flows from $M(p, p')$ with p, p' arbitrary coprime positive integers to other CFTs. In [10] it was conjectured that their methods, which was applied to the unitary case when $p = p' - 1$ can be applied to the general case. This is the problem we study in Chapter 3. We demonstrate new Bailey flows from $M(p, p')$ to $N = 1$ and $N = 2$ superconformal algebras and prove the conjectures of [10]. We present new Bose-Fermi identities for the characters of $N = 1$ and $N = 2$ superconformal algebras. These new identities can be thought of as the generalized RR-type identities for the $N = 1$ and $N = 2$ superconformal characters.

Chapter 2

Fermionic formulas for unrestricted Kostka polynomials

2.1 Introduction

The Kostka numbers $K_{\lambda\mu}$, indexed by the two partitions λ and μ, play an important role in symmetric function theory, representation theory, combinatorics, invariant theory and mathematical physics. The Kostka polynomials $K_{\lambda\mu}(q)$ are q-analogs of the Kostka numbers. There are several combinatorial definitions of the Kostka polynomials. For example Lascoux and Schützenberger [52] proved that the Kostka polynomials are generating functions of semi-standard tableaux of shape λ and content μ with charge statistic. In [60] the Kostka polynomials are expressed as generating function over highest-weight crystal paths with energy statistics. Crystal paths are elements in tensor products of finite-dimensional crystals. Dropping the highest-weight condition yields unrestricted Kostka polynomials [30, 31, 33, 76]. In the $A_1^{(1)}$ setting, unrestricted Kostka polynomials or q-supernomial coefficients were introduced in [75] as q-analogs of the coefficient of x^a in the expansion of $\prod_{j=1}^{N}(1+x+x^2+\cdots+x^j)^{L_j}$. An explicit formula for the $A_{n-1}^{(1)}$ unrestricted Kostka polynomials for completely symmetric and completely antisymmetric crystals was proved in [31, 46]. This formula is called fermionic as it is a manifestly positive expression.

In this chapter we give a new explicit fermionic formula for the unrestricted Kostka polynomials for all Kirillov–Reshetikhin crystals of type $A_{n-1}^{(1)}$. This fermionic formula can be naturally interpreted in terms of a new set of unrestricted rigged configurations for type $A_{n-1}^{(1)}$. Rigged configurations are combinatorial objects originating from the Bethe Ansatz, that label solutions of the Bethe equations. The simplest version of rigged configurations appeared in Bethe's original paper [8] and was later generalized by Kerov, Kirillov and Reshetikhin [44, 47] to models with $\mathrm{GL}(n)$ symmetry. Since the solutions of the Bethe equations label highest weight vectors, one expects a bijection between rigged configurations and semi-standard Young tableaux in the $\mathrm{GL}(n)$ case. Such a bijection was given in [47, 48]. Here we extend this bijection to a bijection Φ between the new set of unrestricted rigged configurations and unrestricted paths. It should be noted that Φ is defined algorithmically. In [70] the bijection was established in a different manner by constructing a crystal structure on the set of rigged configurations. Here we show that the crystal structures are compatible under the algorithmically defined Φ and use this to prove that Φ preserves the statistics.

The bijection Φ has been implemented as a C++ program and has been incorporated into the combinatorics package of MuPAD-Combinat by Francois Descouens [58]. The program is given in chapter 4.

This chapter is structured as follows. In Section 2.2 we review crystals of type $A_{n-1}^{(1)}$, highest weight paths, unrestricted paths and the definition of generalized Kostka polynomials and unrestricted Kostka polynomials as generating functions of highest weight paths and unrestricted paths respectively with energy statistics. In Section 2.3 we give our new definition of unrestricted rigged configurations (see Definition 2.3.3) and derive from this a fermionic expression for the generating function of unrestricted rigged configurations graded by cocharge (see Section 2.3.2). The statistic preserving bijection between unrestricted paths and unrestricted rigged configurations is established in Section 2.4 (see Definition 2.4.7 and Theorem 2.4.1). As a corolloray this yields the equality of the unrestricted Kostka polynomials and the fermionic formula of Section 2.3 (see Corolloray 2.4.2). The result that the crystal structures on paths and rigged configurations are compatible under Φ is stated in Theorem 2.4.14. Most of the technical proofs are relegated to the last three sections. An extended abstract of this chapter can be found in [18].

2.2 Unrestricted paths and Kostka polynomials

2.2.1 Crystals $B^{r,s}$ of type $A_{n-1}^{(1)}$

Kashiwara [37] introduced the notion of crystals and crystal graphs as a combinatorial means to study representations of quantum algebras associated with any symmetrizable Kac–Moody algebra. In this paper we only consider the Kirillov–Reshetikhin crystal $B^{r,s}$ of type $A_{n-1}^{(1)}$ and hence restrict to this case here.

As a set, the crystal $B^{r,s}$ consists of all column-strict Young tableaux of shape (s^r) over the alphabet $\{1, 2, \ldots, n\}$. As a crystal associated to the underlying algebra of finite type A_{n-1}, $B^{r,s}$ is isomorphic to the highest weight crystal with highest weight (s^r). We will define the classical crystal operators explicitly here. The affine crystal operators e_0 and f_0 are given explicitly in [80]. Since we do not use these operators we will omit the details.

Let $I = \{1, 2, \ldots, n-1\}$ be the index set for the vertices of the Dynkin diagram of type A_{n-1}, P the weight lattice, $\{\Lambda_i \in P \mid i \in I\}$ the fundamental roots, $\{\alpha_i \in P \mid i \in I\}$ the simple roots, and $\{h_i \in \mathrm{Hom}_{\mathbb{Z}}(P,\mathbb{Z}) \mid i \in I\}$ the simple coroots. As a type A_{n-1} crystal, $B = B^{r,s}$ is equipped with maps $e_i, f_i : B \longrightarrow B \cup \{0\}$ and $\mathrm{wt} : B \longrightarrow P$ for all $i \in I$ satisfying

$$f_i(b) = b' \Leftrightarrow e_i(b') = b \text{ if } b, b' \in B$$
$$\mathrm{wt}(f_i(b)) = \mathrm{wt}(b) - \alpha_i \text{ if } f_i(b) \in B$$
$$\langle h_i, \mathrm{wt}(b)\rangle = \varphi_i(b) - \varepsilon_i(b),$$

where $\langle \cdot,\cdot \rangle$ is the natural pairing. The maps f_i, e_i are known as the Kashiwara operators. Here for $b \in B$,

$$\varepsilon_i(b) = \max\{k \geq 0 \mid e_i^k(b) \neq 0\}$$
$$\varphi_i(b) = \max\{k \geq 0 \mid f_i^k(b) \neq 0\}.$$

Note that for type A_{n-1}, $P = \mathbb{Z}^n$ and $\alpha_i = \epsilon_i - \epsilon_{i+1}$ where $\{\epsilon_i \mid i \in I\}$ is the standard basis in P. Here $\mathrm{wt}(b) = (w_1, \ldots, w_n)$ is the weight of b where w_i counts the number of letters i in b.

Following [38] let us give the action of e_i and f_i for $i \in I$. Let $b \in B^{r,s}$ be a tableau of shape (s^r). The row word of b is defined by $\mathrm{word}(b) = w_r \cdots w_2 w_1$ where w_k is the word obtained by reading the k-th row of b from left to right. To find $f_i(b)$ and $e_i(b)$ we only consider the subword consisting of the letters i and $i+1$ in the word of b. First view each $i+1$ in the subword as an opening bracket and each i as a closing bracket. Then we ignore each adjacent pair of matched brackets successively. At the end of this process we are left with a subword of the form $i^p(i+1)^q$. If $p > 0$ (resp. $q > 0$) then $f_i(b)$ (resp. $e_i(b)$) is obtained from b by replacing the unmatched subword $i^p(i+1)^q$ by $i^{p-1}(i+1)^{q+1}$ (resp.

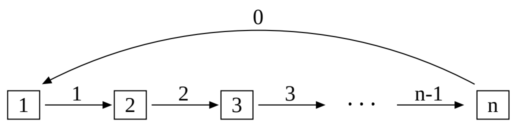

Figure 2.1: Crystal $B^{1,1}$.

$i^{p+1}(i+1)^{q-1}$). If $p=0$ (resp. $q=0$) then $f_i(b)$ (resp. $e_i(b)$) is undefined and we write $f_i(b)=0$ (resp. $e_i(b)=0$).

A crystal B can be viewed as a directed edge-colored graph whose vertices are the elements of B, with a directed edge from b to b' labeled $i \in I$, if and only if $f_i(b)=b'$. This directed graph is known as the crystal graph.

Example 2.2.1. The crystal graph for $B = B^{1,1}$ is given in Figure 2.1.

Given two crystals B and B', we can also define a new crystal by taking the tensor product $B \otimes B'$. As a set $B \otimes B'$ is just the Cartesian product of the sets B and B'. The weight function wt for $b \otimes b' \in B \otimes B'$ is $\mathrm{wt}(b \otimes b') = \mathrm{wt}(b) + \mathrm{wt}(b')$ and the Kashiwara operators e_i, f_i are defined as follows

$$e_i(b \otimes b') = \begin{cases} e_i b \otimes b' & \text{if } \varepsilon_i(b) > \varphi_i(b'), \\ b \otimes e_i b' & \text{otherwise,} \end{cases}$$

$$f_i(b \otimes b') = \begin{cases} f_i b \otimes b' & \text{if } \varepsilon_i(b) \geq \varphi_i(b'), \\ b \otimes f_i b' & \text{otherwise.} \end{cases}$$

This action of f_i and e_i on the tensor product is compatible with the previously defined action on $\mathrm{word}(b \otimes b') = \mathrm{word}(b)\mathrm{word}(b')$.

Example 2.2.2. Let $i = 2$ and

$$b \;=\; \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 2 & 3 \\ \hline \end{array} \otimes \begin{array}{|c|c|}\hline 2 & 3 \\ \hline 3 & 4 \\ \hline 4 & 5 \\ \hline \end{array}.$$

Then $\mathrm{word}(b) = 2312453423$, the relevant subword is $23-2--3-23$, and the unmatched subword is $2--------3$. Hence

$$f_2(b) \;=\; \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & 3 \\ \hline \end{array} \otimes \begin{array}{|c|c|}\hline 2 & 3 \\ \hline 3 & 4 \\ \hline 4 & 5 \\ \hline \end{array} \quad \text{and} \quad e_2(b) \;=\; \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 2 & 3 \\ \hline \end{array} \otimes \begin{array}{|c|c|}\hline 2 & 2 \\ \hline 3 & 4 \\ \hline 4 & 5 \\ \hline \end{array}.$$

2.2.2 Paths and unrestricted paths

Let $B = B^{r_k,s_k} \otimes B^{r_{k-1},s_{k-1}} \otimes \cdots \otimes B^{r_1,s_1}$.

A **highest weight path** or simply **path** is an element $b \in B$ such that $e_i(b) = 0$ for all $1 \leq i \leq n-1$. It is known that the weight vector of a highest weight element for type A_{n-1} is a partition with atmost n nonzero parts. Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ be a partition with atmost n nonzero parts, then the set of all highest weight paths of weight λ and shape B is defined as

$$\overline{\mathcal{P}}(B,\lambda) = \{b \in B \mid \mathrm{wt}(b) = \lambda \quad \text{and } e_i(b) = 0 \text{ for all } 1 \leq i \leq n-1\}.$$

An unrestricted path is an element in the tensor product of crystals $B = B^{r_k,s_k} \otimes B^{r_{k-1},s_{k-1}} \otimes \cdots \otimes B^{r_1,s_1}$. Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ be an n-tuple of nonnegative integers. The set of **unrestricted paths** is defined as

$$\mathcal{P}(B,\lambda) = \{b \in B \mid \mathrm{wt}(b) = \lambda\}.$$

Note that the weight of an unrestricted path need not be a partition.

Example 2.2.3. For $B = B^{1,1} \otimes B^{2,2} \otimes B^{3,1}$ of type A_3 the path

$$b \;=\; \begin{array}{|c|}\hline 1\\\hline\end{array} \otimes \begin{array}{|c|c|}\hline 1&3\\\hline 2&4\\\hline\end{array} \otimes \begin{array}{|c|}\hline 1\\\hline 2\\\hline 3\\\hline\end{array}$$

is a highest weight path of weight $\lambda = (2, 2, 2, 1)$. The path

$$b \;=\; \begin{array}{|c|}\hline 2\\\hline\end{array} \otimes \begin{array}{|c|c|}\hline 1&2\\\hline 2&4\\\hline\end{array} \otimes \begin{array}{|c|}\hline 1\\\hline 3\\\hline 4\\\hline\end{array}$$

is an unrestricted path of weight $\lambda = (2, 3, 1, 2)$.

There exists a crystal isomorphism $R : B^{r,s} \otimes B^{r',s'} \to B^{r',s'} \otimes B^{r,s}$, called the **combinatorial R-matrix**. Combinatorially it is given as follows. Let $b \in B^{r,s}$ and $b' \in B^{r',s'}$. The product $b \cdot b'$ of two tableaux is defined as the Schensted insertion of b' into b. Then $R(b \otimes b') = \tilde{b}' \otimes \tilde{b}$ is the unique pair of tableaux such that $b \cdot b' = \tilde{b}' \cdot \tilde{b}$.

The **local energy function** $H : B^{r,s} \otimes B^{r',s'} \to \mathbb{Z}$ is defined as follows. For $b \otimes b' \in B^{r,s} \otimes B^{r',s'}$, $H(b \otimes b')$ is the number of boxes of the shape of $b \cdot b'$ outside the shape obtained by concatenating (s^r) and $(s'^{r'})$.

Example 2.2.4. For

$$b \otimes b' = \begin{array}{|c|c|}\hline 1&2\\\hline 2&4\\\hline\end{array} \otimes \begin{array}{|c|}\hline 1\\\hline 3\\\hline 4\\\hline\end{array}$$

we have

$$b \cdot b' = \begin{array}{|c|c|c|}\hline 1&1&3\\\hline 2&2&4\\\hline 4\\\cline{1-1}\end{array} = \begin{array}{|c|}\hline 1\\\hline 2\\\hline 4\\\hline\end{array} \cdot \begin{array}{|c|c|}\hline 1&3\\\hline 2&4\\\hline\end{array} = \tilde{b}' \cdot \tilde{b}.$$

so that

$$R(b \otimes b') = \tilde{b}' \otimes \tilde{b} = \begin{array}{|c|}\hline 1\\\hline 2\\\hline 4\\\hline\end{array} \otimes \begin{array}{|c|c|}\hline 1&3\\\hline 2&4\\\hline\end{array}.$$

Since the concatentation of $\ydiagram{2,2}$ and $\ydiagram{1,1,1}$ is $\ydiagram{3,3,1}$, the local energy function $H(b \otimes b') = 0$.

Now let $B = B^{r_k,s_k} \otimes \cdots \otimes B^{r_1,s_1}$ be a k-fold tensor product of crystals. The **tail energy function** $\overleftarrow{D} : B \to \mathbb{Z}$ is given by

$$\overleftarrow{D}(b) = \sum_{1 \le i < j \le k} H_{j-1} R_{j-2} \cdots R_{i+1} R_i(b),$$

where H_i (resp. R_i) is the local energy function (resp. combinatorial R-matrix) acting on the i-th and $(i+1)$-th tensor factors of $b \in B$.

Definition 2.2.5. The generalized Kostka polynomial $K_{\lambda,\mu}(q)$ with $\mu = (R_k, \cdots, R_1)$ where R_j is a rectangular partition of height r_j and width s_j is the generating function of highest weight paths with the tail energy function

$$K_{\lambda,\mu}(q) = \sum_{b \in \overline{P}(\lambda,B)} q^{\overleftarrow{D}(b)}$$

$A^{(1)}_{n-1}$-unrestricted Kostka polynomials or supernomial coefficients were first introduced in [76] as generating functions of unrestricted paths graded by an energy function.

Definition 2.2.6. The q-**supernomial coefficient** or the **unrestricted Kostka polynomial** is defined as

$$X(B,\lambda) = \sum_{b \in \mathcal{P}(B,\lambda)} q^{\overleftarrow{D}(b)}.$$

2.3 Unrestricted rigged configurations and fermionic formula

Rigged configurations are combinatorial objects invented to label the solutions of the Bethe equations, which give the eigenvalues of the Hamiltonian of the underlying physical model [8]. Motivated by the fact that representation theoretically the eigenvectors and eigenvalues can also be labelled by Young tableaux, Kirillov and Reshetikhin [47] gave a bijection between tableaux and rigged configurations. This result and generalizations thereof were proven in [48].

In terms of crystal base theory, the bijection is between highest weight paths and rigged configurations. The new result of this paper is an extension of this bijection to a bijection between unrestricted paths and a new set of rigged configurations. The new set of unrestricted rigged configurations is defined in this section, whereas the bijection is given in section 2.4. In [70], a crystal structure on the new set of unrestricted rigged configurations is given, which provides a different description of the bijection.

2.3.1 Unrestricted rigged configurations

Let $B = B^{r_k,s_k}\otimes\cdots\otimes B^{r_1,s_1}$ and denote by $L = (L_i^{(a)} \mid (a,i)\in\mathcal{H})$ the multiplicity array of B, where $L_i^{(a)}$ is the multiplicity of $B^{a,i}$ in B. Here $\mathcal{H} = I\times\mathbb{Z}_{>0}$ and $I = \{1,2,\dots,n-1\}$ is the index set of the Dynkin diagram A_{n-1}. The sequence of partitions $\nu = \{\nu^{(a)} \mid a\in I\}$ is a (L,λ)-**configuration** if

$$\sum_{(a,i)\in\mathcal{H}} i m_i^{(a)}\alpha_a = \sum_{(a,i)\in\mathcal{H}} i L_i^{(a)}\Lambda_a - \lambda, \tag{2.3.1}$$

where $m_i^{(a)}$ is the number of parts of length i in partition $\nu^{(a)}$. Note that we do not require λ to be a dominant weight here. The **(quasi-)vacancy number** of a configuration is defined as

$$p_i^{(a)} = \sum_{j \geq 1} \min(i,j) L_j^{(a)} - \sum_{(b,j) \in \mathcal{H}} (\alpha_a | \alpha_b) \min(i,j) m_j^{(b)}.$$

Here $(\cdot|\cdot)$ is the normalized invariant form on the weight lattice P such that $(\alpha_i|\alpha_j)$ is the Cartan matrix. Let $\mathrm{C}(L,\lambda)$ be the set of all (L,λ)-configurations. We call $p_i^{(a)}$ quasi-vacancy number to indicate that they can actually be negative in our setting. For the rest of the paper we will simply call them vacancy numbers.

When the dependence of $m_i^{(a)}$ and $p_i^{(a)}$ on the configuration ν is crucial, we also write $m_i^{(a)}(\nu)$ and $p_i^{(a)}(\nu)$, respectively.

In the usual setting a rigged configuration (ν, J) consists of a configuration $\nu \in \mathrm{C}(L,\lambda)$ together with a double sequence of partitions $J = \{J^{(a,i)} \mid (a,i) \in \mathcal{H}\}$ such that the partition $J^{(a,i)}$ is contained in a $m_i^{(a)} \times p_i^{(a)}$ rectangle. In particular this requires that $p_i^{(a)} \geq 0$. For unrestricted paths we need a bigger set, where the lower bound on the parts in $J^{(a,i)}$ can be less than zero.

To define the lower bounds we need the following notation. Let $\lambda' = (c_1, c_2, \ldots, c_{n-1})^t$ where $c_k = \lambda_{k+1} + \lambda_{k+2} + \cdots + \lambda_n$. We also set $c_0 = c_1$. Let $\mathcal{A}(\lambda')$ be the set of tableaux of shape λ' such that the entries in column k are from the set $\{1, 2, \ldots, c_{k-1}\}$ and are strictly decreasing along each column.

Example 2.3.1. For $n = 4$ and $\lambda = (0,1,1,1)$, the set $\mathcal{A}(\lambda')$ consists of the following tableaux

$$\begin{array}{|c|c|c|}\hline 3&3&2\\ \hline 2&2\\ \cline{1-2} 1\\ \cline{1-1}\end{array}\quad \begin{array}{|c|c|c|}\hline 3&3&2\\ \hline 2&1\\ \cline{1-2} 1\\ \cline{1-1}\end{array}\quad \begin{array}{|c|c|c|}\hline 3&2&2\\ \hline 2&1\\ \cline{1-2} 1\\ \cline{1-1}\end{array}\quad \begin{array}{|c|c|c|}\hline 3&3&1\\ \hline 2&2\\ \cline{1-2} 1\\ \cline{1-1}\end{array}\quad \begin{array}{|c|c|c|}\hline 3&3&1\\ \hline 2&1\\ \cline{1-2} 1\\ \cline{1-1}\end{array}\quad \begin{array}{|c|c|c|}\hline 3&2&1\\ \hline 2&1\\ \cline{1-2} 1\\ \cline{1-1}\end{array}\,.$$

Note that each $t \in \mathcal{A}(\lambda')$ is weakly decreasing along each row. This is due to the fact that $t_{j,k} \geq c_k - j + 1$ since column k of height c_k is strictly decreasing and $c_k - j + 1 \geq t_{j,k+1}$ since the entries in column $k+1$ are from the set $\{1, 2, \ldots, c_k\}$.

Given $t \in \mathcal{A}(\lambda')$, we define the **lower bound** as

$$M_i^{(a)}(t) = -\sum_{j=1}^{c_a} \chi(i \geq t_{j,a}) + \sum_{j=1}^{c_{a+1}} \chi(i \geq t_{j,a+1}),$$

where $t_{j,a}$ denotes the entry in row j and column a of t, and $\chi(S) = 1$ if the the statement S is true and $\chi(S) = 0$ otherwise.

Example 2.3.2. For the tableau $t = \begin{array}{|c|c|c|}\hline 3 & 3 & 2 \\ \hline 2 & 2 \\ \cline{1-2} 1 \\ \cline{1-1}\end{array}$ from example 2.3.1 some of the lower bounds are given by

$$M_1^{(1)}(t) = -1, M_4^{(1)}(t) = -1, M_1^{(2)}(t) = 0, M_3^{(2)}(t) = -1, M_2^{(3)}(t) = -1.$$

Let $M, p, m \in \mathbb{Z}$ such that $m \geq 0$. A (M, p, m)-quasipartition μ is a tuple of integers $\mu = (\mu_1, \mu_2, \ldots, \mu_m)$ such that $M \leq \mu_m \leq \mu_{m-1} \leq \cdots \leq \mu_1 \leq p$. Each μ_i is called a part of μ. Note that for $M = 0$ this would be a partition with at most m parts each not exceeding p.

Definition 2.3.3. An **unrestricted rigged configuration** (ν, J) associated to a multiplicity array L and weight λ is a configuration $\nu \in \mathrm{C}(L, \lambda)$ together with a sequence $J = \{J^{(a,i)} \mid (a, i) \in \mathcal{H}\}$ where $J^{(a,i)}$ is a $(M_i^{(a)}(t), p_i^{(a)}, m_i^{(a)})$-quasipartition for some $t \in \mathcal{A}(\lambda')$. Denote the set of all unrestricted rigged configurations corresponding to (L, λ) by $\mathrm{RC}(L, \lambda)$.

Remark 2.3.4.

1. Note that this definition is similar to the definition of level-restricted rigged configurations [73, Definition 5.5]. Whereas for level-restricted rigged configurations the

vacancy number had to be modified according to tableaux in a certain set, here the lower bounds are modified.

2. For type A_1 we have $\lambda = (\lambda_1, \lambda_2)$ so that $\mathcal{A} = \{t\}$ contains just the single tableau

$$t = \begin{array}{|c|}\hline \lambda_2 \\ \hline \lambda_2 - 1 \\ \hline \vdots \\ \hline 1 \\ \hline \end{array}.$$

In this case $M_i(t) = -\sum_{j=1}^{\lambda_2} \chi(i \geq t_{j,1}) = -i$. This agrees with the findings of [83].

The quasipartition $J^{(a,i)}$ is called **singular** if it has a part of size $p_i^{(a)}$. It is often useful to view an (unrestricted) rigged configuration (ν, J) as a sequence of partitions ν where the parts of size i in $\nu^{(a)}$ are labeled by the parts of $J^{(a,i)}$. The pair (i, x) where i is a part of $\nu^{(a)}$ and x is a part of $J^{(a,i)}$ is called a **string** of the a-th rigged partition $(\nu, J)^{(a)}$. The label x is called a **rigging**.

Example 2.3.5. Let $n = 4$, $\lambda = (2, 2, 1, 1)$, $L_1^{(1)} = 6$ and all other $L_i^{(a)} = 0$. Then

$$(\nu, J) \;=\; \begin{array}{l} \square\square\square \;\; -2 \\ \square \;\; 0 \end{array} \qquad \square\square \;\; 0 \qquad \square \;\; -1$$

is an unrestricted rigged configuration in $\mathrm{RC}(L, \lambda)$, where we have written the parts of $J^{(a,i)}$ next to the parts of length i in partition $\nu^{(a)}$. To see that the riggings form quasipartitions, let us write the vacancy numbers $p_i^{(a)}$ next to the parts of length i in partition $\nu^{(a)}$:

$$\begin{array}{l} \square\square\square \;\; 0 \\ \square \;\; 3 \end{array} \qquad \square\square \;\; 0 \qquad \square \;\; -1.$$

This shows that the labels are indeed all weakly below the vacancy numbers. For

$$\begin{array}{|c|c|c|}\hline 4 & 4 & 1 \\ \hline 3 & 3 \\ \cline{1-2} 2 \\ \cline{1-1} 1 \\ \cline{1-1} \end{array} \in \mathcal{A}(\lambda')$$

we get the lower bounds

$$\begin{array}{|c|c|c|l}\cline{1-3} & & & -2 \\ \cline{1-3} & \multicolumn{3}{|l}{-1} \\ \cline{1-1} \end{array} \quad \begin{array}{|c|c|l}\cline{1-2} & & 0 \\ \cline{1-2} \end{array} \quad \begin{array}{|c|l}\cline{1-1} & -1, \\ \cline{1-1} \end{array}$$

which are less or equal to the riggings in (ν, J).

Let $B = B^{r_k,s_k} \otimes \cdots \otimes B^{r_1,s_1}$ and L the corresponding multiplicity array. Let $(\nu, J) \in \mathrm{RC}(L, \lambda)$. Note that rewritting (2.3.1) we get

$$|\nu^{(a)}| = \sum_{j>a} \lambda_j - \sum_{j=1}^{k} s_j \max(r_j - a, 0). \tag{2.3.2}$$

Hence for large i, by definition of vacancy numbers we have

$$\begin{aligned} p_i^{(a)} &= |\nu^{(a-1)}| - 2|\nu^{(a)}| + |\nu^{(a+1)}| + \sum_j \min(i,j) L_j^{(a)} \\ &= \lambda_a - \lambda_{a+1} \end{aligned} \tag{2.3.3}$$

and

$$\begin{aligned} M_i^{(a)}(t) &= -\sum_{j=1}^{c_a} \chi(i \geq t_{j,a}) + \sum_{j=1}^{c_{a+1}} \chi(i \geq t_{j,a+1}) \\ &= -c_a + c_{a+1} = -\lambda_{a+1}. \end{aligned} \tag{2.3.4}$$

For a given $t \in \mathcal{A}(\lambda')$ define

$$\Delta p_i^{(a)}(t) = p_i^{(a)} - M_i^{(a)}(t).$$

We write $\Delta p_i^{(a)}$ for $\Delta p_i^{(a)}(t)$ when there is no cause of confusion. For large i, $\Delta p_i^{(a)}(t) = \lambda_a$.

From the definition of $p_i^{(a)}$ one may easily verify that

$$-p_{i-1}^{(a)} + 2p_i^{(a)} - p_{i+1}^{(a)} \geq m_i^{(a-1)} - 2m_i^{(a)} + m_i^{(a+1)}. \tag{2.3.5}$$

Let $t_{\cdot,a}$ denote the a-th column of t. Then it follows from the definition of $M_i^{(a)}(t)$ that

$$M_i^{(a)}(t) = M_{i-1}^{(a)}(t) - \chi(i \in t_{\cdot,a}) + \chi(i \in t_{\cdot,a+1}).$$

Hence (2.3.5) can be rewritten as

$$\begin{aligned} &- \Delta p_{i-1}^{(a)} + 2\Delta p_i^{(a)} - \Delta p_{i+1}^{(a)} - \chi(i \in t_{\cdot,a}) + \chi(i \in t_{\cdot,a+1}) \\ &\qquad + \chi(i+1 \in t_{\cdot,a}) - \chi(i+1 \in t_{\cdot,a+1}) \geq m_i^{(a-1)} - 2m_i^{(a)} + m_i^{(a+1)}. \end{aligned} \tag{2.3.6}$$

Lemma 2.3.6. *Suppose that for some $t \in \mathcal{A}(\lambda')$, $\Delta p_i^{(a)}(t) \geq 0$ for all $a \in I$ and i such that $m_i^{(a)} > 0$. Then there exists a $t' \in \mathcal{A}(\lambda')$ such that $\Delta p_i^{(a)}(t') \geq 0$ for all i and a.*

Proof. By definition $\Delta p_0^{(a)}(t) = 0$ and $\Delta p_i^{(a)}(t) = \lambda_a \geq 0$ for large i. By (2.3.6)

$$\begin{aligned} \Delta p_i^{(a)}(t) \geq \frac{1}{2}\{&\Delta p_{i-1}^{(a)}(t) + \Delta p_{i+1}^{(a)}(t) + \chi(i \in t_{\cdot,a}) - \chi(i \in t_{\cdot,a+1}) \\ &- \chi(i+1 \in t_{\cdot,a}) + \chi(i+1 \in t_{\cdot,a+1}) + m_i^{(a-1)} + m_i^{(a+1)}\} \end{aligned} \tag{2.3.7}$$

when $m_i^{(a)} = 0$. Hence $\Delta p_i^{(a)}(t) < 0$ is only possible if $m_i^{(a-1)} = m_i^{(a+1)} = 0$, column a of t contains $i+1$ but no i, and column $a+1$ of t contains i but no $i+1$. Let k be minimal such that $\Delta p_i^{(k)}(t) < 0$. Note that $k > 1$ since the first column of t contains all letters $1, 2, \dots, c_1$. Let $k' \leq k$ be minimal such that $\Delta p_i^{(a)}(t) = 0$ for all $k' \leq a < k$. Then inductively for $a = k-1, k-2, \dots, k'$ it follows from (2.3.7) that $m_i^{(a-1)} = 0$ and column a of t contains $i+1$ but no i. Construct a new t' from t by replacing all letters $i+1$ in columns $k', k'+1, \dots, k$ by i. This accomplishes that $\Delta p_j^{(a)}(t') \geq 0$ for all j and $1 \leq a < k$, $\Delta p_i^{(k)}(t') \geq 0$, and $\Delta p_j^{(a)}(t') \geq 0$ for all $a \geq k$ such that $m_j^{(a)} > 0$. Repeating the above construction, if necessary, eventually yields a new tableau t'' such that finally $\Delta p_j^{(a)}(t'') \geq 0$ for all j and a. □

2.3.2 Fermionic formula

The following statistics can be defined on the set of unrestricted rigged configurations. For $(\nu, J) \in \mathrm{RC}(L, \lambda)$ let

$$cc(\nu, J) = cc(\nu) + \sum_{(a,i) \in \mathcal{H}} |J^{(a,i)}|,$$

where $|J^{(a,i)}|$ is the sum of all parts of the quasipartition $J^{(a,i)}$ and

$$cc(\nu) = \frac{1}{2} \sum_{a,b \in I} \sum_{j,k \geq 1} (\alpha_a | \alpha_b) \min(j,k) m_j^{(a)} m_k^{(b)}.$$

Definition 2.3.7. The RC polynomial is defined as

$$M(L, \lambda) = \sum_{(\nu,J) \in \mathrm{RC}(L,\lambda)} q^{cc(\nu,J)}.$$

The RC polynomial is in fact S_n-symmetric in the weight λ. This is not obvious from its definition as both (2.3.1) and the lower bounds are not symmetric with respect to λ.

Let $\mathcal{SA}(\lambda')$ be the set of all nonempty subsets of $\mathcal{A}(\lambda')$ and set

$$M_i^{(a)}(S) = \max\{M_i^{(a)}(t) \mid t \in S\} \qquad \text{for } S \in \mathcal{SA}(\lambda').$$

By inclusion-exclusion the set of all allowed riggings for a given $\nu \in \mathrm{C}(L,\lambda)$ is

$$\bigcup_{S\in\mathcal{SA}(\lambda')} (-1)^{|S|+1}\{J \mid J^{(a,i)} \text{ is a } (M_i^{(a)}(S), p_i^{(a)}, m_i^{(a)})\text{-quasipartition}\}.$$

The q-binomial coefficient $\begin{bmatrix} m+p \\ m \end{bmatrix}$, defined as

$$\begin{bmatrix} m+p \\ m \end{bmatrix} = \frac{(q)_{m+p}}{(q)_m (q)_p}$$

where $(q)_n = (1-q)(1-q^2)\cdots(1-q^n)$, is the generating function of partitions with at most m parts each not exceeding p. Hence the polynomial $M(L,\lambda)$ may be rewritten as

$$M(L,\lambda) = \sum_{S\in\mathcal{SA}(\lambda')} (-1)^{|S|+1} \sum_{\nu\in\mathrm{C}(L,\lambda)} q^{cc(\nu)+\sum_{(a,i)\in\mathcal{H}} m_i^{(a)} M_i^{(a)}(S)} \times \prod_{(a,i)\in\mathcal{H}} \begin{bmatrix} m_i^{(a)} + p_i^{(a)} - M_i^{(a)}(S) \\ m_i^{(a)} \end{bmatrix}$$

called **fermionic formula**. This formula is different from the fermionic formulas of [31, 46] which exist in the special case when L is the multiplicity array of $B = B^{1,s_k} \otimes \cdots \otimes B^{1,s_1}$ or $B = B^{r_k,1} \otimes \cdots \otimes B^{r_1,1}$.

2.4 Bijection

In this section we define the bijection $\Phi : \mathcal{P}(B,\lambda) \to \mathrm{RC}(L,\lambda)$ from paths to unrestricted rigged configurations algorithmically. The algorithm generalizes the bijection of [48] to the unrestricted case. The main result is summarized in the following theorem.

Theorem 2.4.1. *Let $B = B^{r_k,s_k} \otimes \cdots \otimes B^{r_1,s_1}$, L the corresponding multiplicity array and $\lambda = (\lambda_1,\ldots,\lambda_n)$ a sequence of nonnegative integers. There exists a bijection Φ : $\mathcal{P}(B,\lambda) \to \mathrm{RC}(L,\lambda)$ which preserves the statistics, that is, $\overleftarrow{D}(b) = cc(\Phi(b))$ for all $b \in \mathcal{P}(B,\lambda)$.*

A different proof of Theorem 2.4.1 is given in [70] by proving directly that the crystal structure on rigged configurations and paths coincide. The results in [70] hold for all for all simply-laced types, not just type $A_{n-1}^{(1)}$. Hence Theorem 2.4.1 holds whenever there is a corresponding bijection for the highest weight elements (for example for type $D_n^{(1)}$ for symmetric powers [74] and antisymmetric powers [70]). Using virtual crystals and the method of folding Dynkin diagrams, these results can be extended to other affine root systems.

Here we use the crystal structure to prove that the statistics is preserved. It follows from Theorem 2.4.14 that the algorithmic definition for Φ of this thesis and the definition of [70] agree.

An immediate corollary of Theorem 2.4.1 is the relation between the fermionic formula for the RC polynomial of section 2.3 and the unrestricted Kostka polynomials of section 2.2.

Corollary 2.4.2. *With the same assumptions as in Theorem 2.4.1, $X(B,\lambda) = M(L,\lambda)$.*

2.4.1 Operations on crystals

To define Φ we first need to introduce certain maps on paths and rigged configurations. These maps correspond to the following operations on crystals:

1. If $B = B^{1,1} \otimes B'$, let $\mathrm{lh}(B) = B'$. This operation is called **left-hat**.

2. If $B = B^{r,s} \otimes B'$ with $s \geq 2$, let $\mathrm{ls}(B) = B^{r,1} \otimes B^{r,s-1} \otimes B'$. This operation is called **left-split**.

3. If $B = B^{r,1} \otimes B'$ with $r \geq 2$, let $\mathrm{lb}(B) = B^{1,1} \otimes B^{r-1,1} \otimes B'$. This operation is called **box-split**.

In analogy we define $\mathrm{lh}(L)$ (resp. $\mathrm{ls}(L)$, $\mathrm{lb}(L)$) to be the multiplicity array of $\mathrm{lh}(B)$ (resp. $\mathrm{ls}(B)$, $\mathrm{lb}(B)$), if L is the multiplicity array of B. The corresponding maps on crystal elements are given by:

1. Let $b = c \otimes b' \in B^{1,1} \otimes B'$. Then $\mathrm{lh}(b) = b'$.

2. Let $b = c \otimes b' \in B^{r,s} \otimes B'$, where $c = c_1 c_2 \cdots c_s$ and c_i denotes the i-th column of c. Then $\mathrm{ls}(b) = c_1 \otimes c_2 \cdots c_s \otimes b'$.

3. Let $b = \begin{array}{|c|}\hline b_1 \\ \hline b_2 \\ \hline \vdots \\ \hline b_r \\ \hline \end{array} \otimes b' \in B^{r,1} \otimes B'$, where $b_1 < \cdots < b_r$. Then $\mathrm{lb}(b) = \begin{array}{|c|}\hline b_r \\ \hline \end{array} \otimes \begin{array}{|c|}\hline b_1 \\ \hline \vdots \\ \hline b_{r-1} \\ \hline \end{array} \otimes b'$.

In the next subsection we define the corresponding maps on rigged configurations, and give the bijection in subsection 2.4.3.

2.4.2 Operations on rigged configurations

Suppose $L_1^{(1)} > 0$. The main algorithm on rigged configurations as defined in [47, 48] for admissible rigged configurations can be extended to our setting. For a tuple of nonnegative integers $\lambda = (\lambda_1, \dots, \lambda_n)$, let λ^- be the set of all nonnegative tuples $\mu = (\mu_1, \dots, \mu_n)$ such that $\lambda - \mu = \epsilon_r$ for some $1 \le r \le n$ where ϵ_r is the canonical r-th unit vector in $\mathbb{Z}^n$. Define $\delta : \mathrm{RC}(L, \lambda) \to \bigcup_{\mu \in \lambda^-} \mathrm{RC}(\mathrm{lh}(L), \mu)$ by the following algorithm. Let $(\nu, J) \in \mathrm{RC}(L, \lambda)$. Set $\ell^{(0)} = 1$ and repeat the following process for $a = 1, 2, \dots, n-1$ or until stopped. Find the smallest index $i \ge \ell^{(a-1)}$ such that $J^{(a,i)}$ is singular. If no such i exists, set $\mathrm{rk}(\nu, J) = a$ and stop. Otherwise set $\ell^{(a)} = i$ and continue with $a + 1$. Set all undefined $\ell^{(a)}$ to ∞.

The new rigged configuration $(\tilde{\nu}, \tilde{J}) = \delta(\nu, J)$ is obtained by removing a box from the selected strings and making the new strings singular again. Explicitly

$$m_i^{(a)}(\tilde{\nu}) = m_i^{(a)}(\nu) + \begin{cases} 1 & \text{if } i = \ell^{(a)} - 1 \\ -1 & \text{if } i = \ell^{(a)} \\ 0 & \text{otherwise.} \end{cases}$$

The partition $\tilde{J}^{(a,i)}$ is obtained from $J^{(a,i)}$ by removing a part of size $p_i^{(a)}(\nu)$ for $i = \ell^{(a)}$, adding a part of size $p_i^{(a)}(\tilde{\nu})$ for $i = \ell^{(a)} - 1$, and leaving it unchanged otherwise. Then $\delta(\nu, J) \in \mathrm{RC}(\mathrm{lh}(L), \mu)$ where $\mu = \lambda - \epsilon_{\mathrm{rk}(\nu, J)}$.

Proposition 2.4.3. *δ is well-defined.*

The proof is given in section 2.5.

Example 2.4.4. Let L be the multiplicity array of $B = B^{1,1} \otimes B^{2,1} \otimes B^{2,3}$ and $\lambda =$

$(2, 2, 2, 1, 1, 1)$. Then

$$(\nu, J) = \begin{array}{ll} \square\square & -1 \\ \square & 0 \end{array} \quad \begin{array}{ll} \square\square\square & 0 \\ \square & -1 \\ \square & -1 \end{array} \quad \begin{array}{ll} \square\square\square & 0 \end{array} \quad \begin{array}{ll} \square\square & -1 \end{array} \quad \begin{array}{ll} \square & -1 \end{array} \in \mathrm{RC}(L, \lambda).$$

Writing the vacancy numbers next to each part instead of the riggings we get

$$\begin{array}{ll} \square\square & -1 \\ \square & 0 \end{array} \quad \begin{array}{ll} \square\square\square & 0 \\ \square & -1 \\ \square & -1 \end{array} \quad \begin{array}{ll} \square\square\square & 1 \end{array} \quad \begin{array}{ll} \square\square & -1 \end{array} \quad \begin{array}{ll} \square & -1 \end{array}.$$

Hence $\ell^{(1)} = \ell^{(2)} = 1$ and all other $\ell^{(a)} = \infty$, so that

$$\delta(\nu, J) = \begin{array}{ll} \square\square & -1 \end{array} \quad \begin{array}{ll} \square\square\square & 0 \\ \square & -1 \end{array} \quad \begin{array}{ll} \square\square\square & 0 \end{array} \quad \begin{array}{ll} \square\square & -1 \end{array} \quad \begin{array}{ll} \square & -1 \end{array}.$$

Also $\mathrm{rk}(\nu, J) = 3$ and $cc(\nu, J) = 2$.

The inverse algorithm of δ denoted by δ^{-1} is defined as follows. Let $L_1^{(1)} = \overline{L}_1^{(1)} + 1, L_i^{(k)} = \overline{L}_i^{(k)}$ for all $i, k \neq 1$. Let $\overline{\lambda}$ be a weight and $\lambda = \overline{\lambda} + \epsilon_r$ for some $1 \leq r \leq n$. Define $\delta^{-1} : \mathrm{RC}(\overline{L}, \overline{\lambda}) \to \mathrm{RC}(L, \lambda)$ by the following algorithm. Let $(\overline{\nu}, \overline{J}) \in \mathrm{RC}(\overline{L}, \overline{\lambda})$. Let $s^{(r)} = \infty$. For $k = r - 1$ down to 1, select the longest singular string in $(\overline{\nu}, \overline{J})^{(k)}$ of length $s^{(k)}$ (possibly of zero length) such that $s^{(k)} \leq s^{(k+1)}$. With the convention $s^{(0)} = 0$ we have $s^{(0)} \leq s^{(1)}$ as well. $\delta^{-1}(\overline{\nu}, \overline{J}) = (\nu, J)$ is obtained from $(\overline{\nu}, \overline{J})$ by adding a box to each of the selected strings, and resetting their labels to make them singular with respect to the new vacancy number for $\mathrm{RC}(L, \lambda)$, and leaving all other strings unchanged.

Example 2.4.5. Let $n = 6$, $\overline{L}_1^{(2)} = \overline{L}_3^{(2)} = 1$ and $\overline{\lambda} = (1, 1, 1, 2, 2, 1)$.

$$(\nu, J) = \begin{array}{ll} \square\square\square & -1 \end{array}, \begin{array}{ll} \square\square\square & 0 \\ \square\square\square & 0 \end{array}, \begin{array}{ll} \square\square\square & -1 \\ \square\square & -1 \end{array}, \begin{array}{ll} \square\square & -1 \\ \square & -1 \end{array}, \begin{array}{ll} \square & 0 \end{array}$$

is a rigged configuration in $\mathrm{RC}(L, \lambda)$. For $r = 4$

$$\delta^{-1}(\nu, J) = \begin{array}{l} \square\square\square\ -1 \\ \square\ -1 \end{array}, \begin{array}{l} \square\square\square\square\ 0 \\ \square\square\square\ 0 \end{array}, \begin{array}{l} \square\square\square\square\ -2 \\ \square\square\ -1 \end{array}, \begin{array}{l} \square\square\ -1 \\ \square\ -1 \end{array}, \begin{array}{l} \square\ 0 \end{array}.$$

Proposition 2.4.6. δ^{-1} *is well defined.*

This proposition will also be proved in section 2.5.

Let $s \geq 2$. Suppose $B = B^{r,s} \otimes B'$ and L the corresponding multiplicity array. Note that $\mathrm{C}(L, \lambda) \subset \mathrm{C}(\mathrm{ls}(L), \lambda)$. Under this inclusion map, the vacancy number $p_i^{(a)}$ for ν increases by $\delta_{a,r}\chi(i < s)$. Hence there is a well-defined injective map $\mathrm{ls}_{rc} : \mathrm{RC}(L, \lambda) \to \mathrm{RC}(\mathrm{ls}(L), \lambda)$ given by the identity map $\mathrm{ls}_{rc}(\nu, J) = (\nu, J)$.

Suppose $r \geq 2$ and $B = B^{r,1} \otimes B'$ with multiplicity array L. Then there is an injection $\mathrm{lb}_{rc} : \mathrm{RC}(L, \lambda) \to \mathrm{RC}(\mathrm{lb}(L), \lambda)$ defined by adding singular strings of length 1 to $(\nu, J)^{(a)}$ for $1 \leq a < r$. Note that the vacancy numbers remain unchanged under lb_{rc}.

2.4.3 Bijection

The map $\Phi : \mathcal{P}(B, \lambda) \to \mathrm{RC}(L, \lambda)$ is defined recursively by various commutative diagrams. Note that it is possible to go from $B = B^{r_k,s_k} \otimes B^{r_{k-1},s_{k-1}} \otimes \cdots \otimes B^{r_1,s_1}$ to the empty crystal via successive application of lh, ls and lb.

Definition 2.4.7. Define that map $\Phi : \mathcal{P}(B, \lambda) \to \mathrm{RC}(L, \lambda)$ such that the empty path maps to the empty rigged configuration and such that the following conditions hold:

1. Suppose $B = B^{1,1} \otimes B'$. Then the following diagram commutes:

$$\begin{array}{ccc} \mathcal{P}(B,\lambda) & \xrightarrow{\Phi} & \mathrm{RC}(L,\lambda) \\ {\scriptstyle \mathrm{lh}}\big\downarrow & & \big\downarrow{\scriptstyle \delta} \\ \bigcup_{\mu\in\lambda^-} \mathcal{P}(\mathrm{lh}(B),\mu) & \xrightarrow[\Phi]{} & \bigcup_{\mu\in\lambda^-} \mathrm{RC}(\mathrm{lh}(L),\mu) \end{array}$$

2. Suppose $B = B^{r,s} \otimes B'$ with $s \geq 2$. Then the following diagram commutes:

$$\begin{array}{ccc} \mathcal{P}(B,\lambda) & \xrightarrow{\Phi} & \mathrm{RC}(L,\lambda) \\ {\scriptstyle \mathrm{ls}}\big\downarrow & & \big\downarrow{\scriptstyle \mathrm{ls}_{rc}} \\ \mathcal{P}(\mathrm{ls}(B),\lambda) & \xrightarrow[\Phi]{} & \mathrm{RC}(\mathrm{ls}(L),\lambda) \end{array}$$

3. Suppose $B = B^{r,1} \otimes B'$ with $r \geq 2$. Then the following diagram commutes:

$$\begin{array}{ccc} \mathcal{P}(B,\lambda) & \xrightarrow{\Phi} & \mathrm{RC}(L,\lambda) \\ {\scriptstyle \mathrm{lb}}\big\downarrow & & \big\downarrow{\scriptstyle \mathrm{lb}_{rc}} \\ \mathcal{P}(\mathrm{lb}(B),\lambda) & \xrightarrow[\Phi]{} & \mathrm{RC}(\mathrm{lb}(L),\lambda) \end{array}$$

Proposition 2.4.8. *The map Φ of Definition 2.4.7 is a well-defined bijection.*

The proof is given in section 2.6.

Example 2.4.9. Let $B = B^{1,1} \otimes B^{2,1} \otimes B^{2,3}$ and $\lambda = (2,2,2,1,1,1)$. Then

$$b = \boxed{3} \otimes \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline \end{array} \otimes \begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline \end{array} \in \mathcal{P}(B,\lambda)$$

and $\Phi(b)$ is the rigged configuration (ν, J) of Example 2.4.4. We have $\overleftarrow{D}(b) = cc(\nu, J) = 2$.

Example 2.4.10. Let $n = 4$, $B = B^{2,2} \otimes B^{2,1}$ and $\lambda = (2, 2, 1, 1)$. Then the multiplicity array is $L_1^{(2)} = 1$, $L_2^{(2)} = 1$ and $L_i^{(a)} = 0$ for all other (a, i). There are 7 possible unrestricted paths in $\mathcal{P}(B, \lambda)$. For each path $b \in \mathcal{P}(B, \lambda)$ the corresponding rigged configuration $(\nu, J) = \Phi(b)$ together with the tail energy and cocharge is summarized below.

$$b = \begin{array}{|c|c|}\hline 1&1\\\hline 2&2\\\hline\end{array} \otimes \begin{array}{|c|}\hline 3\\\hline 4\\\hline\end{array} \qquad (\nu, J) = \begin{array}{|c|}\hline \\\hline\end{array}\ 0 \quad \begin{array}{|c|}\hline \\\hline \\\hline\end{array}\begin{array}{l} -1\\ -1\end{array} \quad \begin{array}{|c|}\hline \\\hline\end{array}\ 0 \qquad \overleftarrow{D}(b) = 0 = cc(\nu, J)$$

$$b = \begin{array}{|c|c|}\hline 1&1\\\hline 2&4\\\hline\end{array} \otimes \begin{array}{|c|}\hline 2\\\hline 3\\\hline\end{array} \qquad (\nu, J) = \begin{array}{|c|}\hline \\\hline\end{array}\ -1 \quad \begin{array}{|c|}\hline \\\hline \\\hline\end{array}\begin{array}{l} 0\\ 0\end{array} \quad \begin{array}{|c|}\hline \\\hline\end{array}\ 0 \qquad \overleftarrow{D}(b) = 1 = cc(\nu, J)$$

$$b = \begin{array}{|c|c|}\hline 1&2\\\hline 2&3\\\hline\end{array} \otimes \begin{array}{|c|}\hline 1\\\hline 4\\\hline\end{array} \qquad (\nu, J) = \begin{array}{|c|}\hline \\\hline\end{array}\ 0 \quad \begin{array}{|c|}\hline \\\hline \\\hline\end{array}\begin{array}{l} 0\\ 0\end{array} \quad \begin{array}{|c|}\hline \\\hline\end{array}\ -1 \qquad \overleftarrow{D}(b) = 1 = cc(\nu, J)$$

$$b = \begin{array}{|c|c|}\hline 1&2\\\hline 2&4\\\hline\end{array} \otimes \begin{array}{|c|}\hline 1\\\hline 3\\\hline\end{array} \qquad (\nu, J) = \begin{array}{|c|}\hline \\\hline\end{array}\ 0 \quad \begin{array}{|c|}\hline \\\hline \\\hline\end{array}\begin{array}{l} 0\\ -1\end{array} \quad \begin{array}{|c|}\hline \\\hline\end{array}\ 0 \qquad \overleftarrow{D}(b) = 1 = cc(\nu, J)$$

$$b = \begin{array}{|c|c|}\hline 1&3\\\hline 2&4\\\hline\end{array} \otimes \begin{array}{|c|}\hline 1\\\hline 2\\\hline\end{array} \qquad (\nu, J) = \begin{array}{|c|}\hline \\\hline\end{array}\ 0 \quad \begin{array}{|c|}\hline \\\hline \\\hline\end{array}\begin{array}{l} 0\\ 0\end{array} \quad \begin{array}{|c|}\hline \\\hline\end{array}\ 0 \qquad \overleftarrow{D}(b) = 2 = cc(\nu, J)$$

$$b = \begin{array}{|c|c|}\hline 1&1\\\hline 2&3\\\hline\end{array} \otimes \begin{array}{|c|}\hline 2\\\hline 4\\\hline\end{array} \qquad (\nu, J) = \begin{array}{|c|}\hline \\\hline\end{array}\ -1 \quad \begin{array}{|c|c|}\hline &\\\hline\end{array}\ 0 \quad \begin{array}{|c|}\hline \\\hline\end{array}\ -1 \qquad \overleftarrow{D}(b) = 0 = cc(\nu, J)$$

$$b = \begin{array}{|c|c|}\hline 1&2\\\hline 3&4\\\hline\end{array} \otimes \begin{array}{|c|}\hline 1\\\hline 2\\\hline\end{array} \qquad (\nu, J) = \begin{array}{|c|}\hline \\\hline\end{array}\ -1 \quad \begin{array}{|c|c|}\hline &\\\hline\end{array}\ 1 \quad \begin{array}{|c|}\hline \\\hline\end{array}\ -1 \qquad \overleftarrow{D}(b) = 1 = cc(\nu, J)$$

The unrestricted Kostka polynomial in this case is $M(L, \lambda) = 2 + 4q + q^2 = X(B, \lambda)$.

2.4.4 Crystal operators on unrestricted rigged configurations

Let $B = B^{r_k,s_k} \otimes \cdots \otimes B^{r_1,s_1}$ and L be the multiplicity array of B. Let $\mathcal{P}(B) = \bigcup_\lambda \mathcal{P}(B, \lambda)$ and $\mathrm{RC}(L) = \bigcup_\lambda \mathrm{RC}(L, \lambda)$. Note that the bijection Φ of Definition 2.4.7 extends to a bijection from $\mathcal{P}(B)$ to $\mathrm{RC}(L)$. Let f_a and e_a for $1 \le a < n$ be the crystal operators acting on the paths in $\mathcal{P}(B)$. In [70] analogous operators $\tilde{f}_a$ and $\tilde{e}_a$ for $1 \le a < n$ acting on rigged configurations in $\mathrm{RC}(L)$ were defined.

Definition 2.4.11. [70, Definition 3.3]

1. Define $\tilde{e}_a(\nu, J)$ by removing a box from a string of length k in $(\nu, J)^{(a)}$ leaving all colabels fixed and increasing the new label by one. Here k is the length of the string with the smallest negative rigging of smallest length. If no such string exists, $\tilde{e}_a(\nu, J)$ is undefined.

2. Define $\tilde{f}_a(\nu, J)$ by adding a box to a string of length k in $(\nu, J)^{(a)}$ leaving all colabels fixed and decreasing the new label by one. Here k is the length of the string with the smallest nonpositive rigging of largest length. If no such string exists, add a new string of length one and label -1. If the result is not a valid unrestricted rigged configuration $\tilde{f}_a(\nu, J)$ is undefined.

Example 2.4.12. Let L be the multiplicity array of $B = B^{1,3} \otimes B^{3,2} \otimes B^{2,1}$ and let

$$(\nu, J) = \begin{array}{l} \square\square\square\square \; -3 \\ \square \; -1 \end{array} \quad \begin{array}{l} \square\square\square \; 0 \\ \square \; 1 \end{array} \quad \begin{array}{l} \square\square \; -1 \\ \square \; -1 \end{array} \in \mathrm{RC}(L).$$

Then

$$\tilde{f}_3(\nu, J) = \begin{array}{l} \square\square\square\square \; -3 \\ \square \; -1 \end{array} \quad \begin{array}{l} \square\square\square \; 1 \\ \square \; 1 \end{array} \quad \begin{array}{l} \square\square\square \; -2 \\ \square \; -1 \end{array}$$

$$\text{and} \qquad \tilde{e}_3(\nu, J) = \begin{array}{l} \square\square\square\square \; -3 \\ \square \; -1 \end{array} \quad \begin{array}{l} \square\square\square \; -1 \\ \square \; 0 \end{array} \quad \begin{array}{l} \square\square \; 1 \end{array} .$$

Define $\widetilde{\varphi}_a(\nu, J) = \max\{k \geq 0 \mid \tilde{f}_a(\nu, J) \neq 0\}$ and $\widetilde{\varepsilon}_a(\nu, J) = \max\{k \geq 0 \mid \tilde{e}_a(\nu, J) \neq 0\}$. The following Lemma is proven in [70].

Lemma 2.4.13. *[70, Lemma 3.6] Let $(\nu, J) \in \mathrm{RC}(L)$. For fixed $a \in \{1, 2, \ldots, n-1\}$, let $p = p_i^{(a)}$ be the vacancy number for large i and let $s \leq 0$ be the smallest nonpositive label in $(\nu, J)^{(a)}$; if no such label exists set $s = 0$. Then $\widetilde{\varphi}_a(\nu, J) = p - s$.*

Theorem 2.4.14. *Let $B = B^{r_k,s_k} \otimes \cdots \otimes B^{r_1,s_1}$ and L the multiplicity array of B. Then the following diagrams commute:*

$$\begin{array}{ccc} \mathcal{P}(B) & \xrightarrow{\Phi} & \mathrm{RC}(L) \\ {\scriptstyle f_a}\downarrow & & \downarrow{\scriptstyle \tilde{f}_a} \\ \mathcal{P}(B) & \xrightarrow[\Phi]{} & \mathrm{RC}(L) \end{array} \qquad \begin{array}{ccc} \mathcal{P}(B) & \xrightarrow{\Phi} & \mathrm{RC}(L) \\ {\scriptstyle e_a}\downarrow & & \downarrow{\scriptstyle \tilde{e}_a} \\ \mathcal{P}(B) & \xrightarrow[\Phi]{} & \mathrm{RC}(L). \end{array} \tag{2.4.1}$$

The proof of Theorem 2.4.14 is given in section 2.7. Note that Proposition 2.4.8 and Theorem 2.4.14 imply that the operators $\tilde{f}_a, \tilde{e}_a$ give a crystal structure on $\mathrm{RC}(L)$. In [70] it is shown directly that $\tilde{f}_a$ and $\tilde{e}_a$ define a crystal structure on $\mathrm{RC}(L)$.

2.4.5 Proof of Theorem 2.4.1

By Proposition 2.4.8 Φ is a bijection which proves the first part of Theorem 2.4.1. By Theorem 2.4.14 the operators $\tilde{f}_a$ and $\tilde{e}_a$ give a crystal structure on $\mathrm{RC}(L)$ induced by the crystal structure on $\mathcal{P}(B)$ under Φ. The highest weight elements are given by the usual rigged configurations and highest weight paths, respectively, for which Theorem 2.4.1 is known to hold by [48]. The energy function $\overleftarrow{D}$ is constant on classical components. By [70, Theorem 3.9] the statistics cc on rigged configurations is also constant on classical components. Hence Φ preserves the statistic.

2.5 Proof of Propositions 2.4.3 and 2.4.6

In this section we prove Propositions 2.4.3 and 2.4.6, namely that δ is a well-defined bijection. The following remark will be useful.

Remark 2.5.1. Let (ν, J) be admissible with respect to $t \in \mathcal{A}(\lambda')$. Suppose that $\Delta p_{i-1}^{(k)}(t)+$

$\Delta p_{i+1}^{(k)}(t) \geq 1$ and $\Delta p_i^{(k)}(t) = m_i^{(k)}(\nu) = 0$. Then by (2.3.6) there are five choices for the letters i and $i+1$ in columns k and $k+1$ of t:

1. $i+1$ in column k;
2. $i+1$ in column k and $k+1$, i in column $k+1$;
3. i in column $k+1$;
4. i in column k and $k+1$, $i+1$ in column k;
5. $i+1$ in column k, i in column $k+1$.

In cases 1 and 2 we have $m_i^{(k-1)}(\nu) = 0$. Changing letter $i+1$ to i in column k to form a new tableau t' has the effect $M_i^{(k)}(t') = M_i^{(k)}(t) - 1$, $M_i^{(k-1)}(t') = M_i^{(k-1)}(t) + 1$ and all other lower bounds remain unchanged. In cases 3 and 4 we have $m_i^{(k+1)}(\nu) = 0$. Changing letter i to $i+1$ in column $k+1$ to form a new tableau t' has the effect $M_i^{(k)}(t') = M_i^{(k)}(t) - 1$, $M_i^{(k+1)}(t') = M_i^{(k+1)}(t) + 1$ and all other lower bounds remain unchanged. Finally in case 5 either $m_i^{(k-1)}(\nu) = 0$ or $m_i^{(k+1)}(\nu) = 0$. Changing $i+1$ to i in column k (resp. i to $i+1$ in column $k+1$) has the same effect as in case 1 (resp. case 3).

This shows that under the replacement $t \mapsto t'$ we have $\Delta p_i^{(k)}(t') > 0$ and by Lemma 2.3.6 (ν, J) is admissible with respect to some tableau t''.

Let λ be a weight such that $\lambda_r > 0$ for a given $1 \leq r \leq n$. Set $\overline{\lambda} = \lambda - \epsilon_r$. Recall that $c_k = \lambda_{k+1} + \lambda_{k+2} + \cdots + \lambda_n$ is the height of the k-th column of $t \in \mathcal{A}(\lambda')$. Let us define the map $\mathcal{D}_r : \mathcal{A}(\lambda') \to \mathcal{A}(\overline{\lambda}')$ with $\overline{t} = \mathcal{D}_r(t)$ as follows. If $t_{1,r} < c_{r-1}$ then

$$\overline{t}_{i,k} = \begin{cases} t_{i+1,k} & \text{for } 1 \leq k \leq r-1 \text{ and } 1 \leq i < c_k, \\ t_{i,k} & \text{for } r \leq k \leq n \text{ and } 1 \leq i \leq c_k. \end{cases} \tag{2.5.1}$$

If $t_{1,r} = c_{r-1}$ then there exists $1 \leq j \leq c_r$ such that $t_{i,r} = t_{i-1,r} - 1$ for $2 \leq i \leq j$ and $t_{j+1,r} < t_{j,r} - 1$ if $j < c_r$. In this case

$$\bar{t}_{i,k} = \begin{cases} t_{i+1,k} & \text{for } 1 \leq k \leq r-1 \text{ and } 1 \leq i < c_k, \\ t_{i,r} - 1 & \text{for } k = r \text{ and } 1 \leq i \leq j, \\ t_{i,r} & \text{for } k = r \text{ and } j < i \leq c_r, \\ t_{i,k} & \text{for } r < k \leq n \text{ and } 1 \leq i \leq c_k. \end{cases} \tag{2.5.2}$$

Note that by definition the entries of $\mathcal{D}_r(t)$ are strictly decreasing along columns. Let $\overline{c}_k = \overline{\lambda}_{k+1} + \cdots + \overline{\lambda}_n$. Then we have $\overline{c}_k = c_k - 1$ for $1 \leq k \leq r-1$ and $\overline{c}_k = c_k$ for $r \leq k \leq n$. Again by definition $\bar{t}_{j,1} \in \{1, 2, \cdots, \overline{c}_1\}$ for all $1 \leq j \leq \overline{c}_1$ and $\bar{t}_{j,k} \in \{1, 2, \cdots, \overline{c}_{k-1}\}$ for all $2 \leq j \leq \overline{c}_k$ and $1 \leq k \leq n$. Therefore, $\mathcal{D}_r(t) \in \mathcal{A}(\overline{\lambda}')$.

Example 2.5.2. Let $t = \begin{array}{|c|c|c|} \hline 3 & 3 & 2 \\ \hline 2 & 1 & \multicolumn{1}{c}{} \\ \cline{1-2} 1 & \multicolumn{2}{c}{} \\ \cline{1-1} \end{array}$ and $r = 3$. Then $\mathcal{D}_r(t) = \begin{array}{|c|c|c|} \hline 2 & 1 & 1 \\ \hline 1 & \multicolumn{2}{c}{} \\ \cline{1-1} \end{array}$.

We will use the following lemma and remark in the proofs.

Lemma 2.5.3. *Let $B = B^{r_l,s_l} \otimes \cdots \otimes B^{r_1,s_1}$ with $r_l = 1 = s_l$. Let $(\overline{\nu}, \overline{J}) = \delta(\nu, J)$ and let* $\mathrm{rk}(\nu, J) = r$. *For $1 < k < r$ let $i = t_{1,k}$. Then one of the following conditions hold:*

1. $m_i^{(k)}(\nu) = 0$ *or*
2. $m_i^{(k)}(\nu) = 1$*, in which case δ selects the part of length i in $\nu^{(k)}$.*

Proof. Note that $i = t_{1,k} \geq c_k$. By (2.3.2) we have $|\nu^{(k)}| \leq c_k$, so that either $m_i^{(k)}(\nu) = 0$ or $i = c_k$ and $\nu^{(k)}$ consists of just one part of size i. In this case $m_i^{(k)}(\nu) = 1$ and δ has to select this single part. □

Remark 2.5.4. By (2.3.2) we have

$$|\nu^{(r)}| = |\nu^{(r-1)}| - \lambda_r + \sum_{i\geq 1} s_i \chi(r_i \geq r)$$
$$|\nu^{(r+1)}| = |\nu^{(r-1)}| - \lambda_r - \lambda_{r+1} + 2\sum_{i\geq 1} s_i \chi(r_i \geq r) - \sum_{i\geq 1} s_i \delta_{r_i,r}.$$

Note that for $a > 0$

$$\sum_{i\geq 1} \min(a,i) L_i^{(r)} = \sum_{i\geq 1} s_i \chi(s_i \leq a)\delta_{r_i,r} + \sum_{i\geq 1} a\chi(s_i > a)\delta_{r_i,r}.$$

Then if $|\nu^{(r-1)}| = c_{r-1} - k$ for some $k \geq 0$ it follows that

$$-2|\nu^{(r)}| + |\nu^{(r+1)}| + \sum_{i\geq 1} \min(a,i) L_i^{(r)} = -2\lambda_{r+1} - c_{r+1} + k - \sum_{i\geq 1} \max(s_i - a, 0)\delta_{r_i,r}.$$

Proof of Proposition 2.4.3. To prove that δ is well-defined it needs to be shown that $(\overline{\nu}, \overline{J}) = \delta(\nu, J) \in \mathrm{RC}(\overline{L}, \overline{\lambda})$. Here $\overline{L}$ is given by $\overline{L}_1^{(1)} = L_1^{(1)} - 1$, $\overline{L}_i^{(a)} = L_i^{(a)}$ for all other i, a, and $\overline{\lambda} = \lambda - \epsilon_r$ where $r = \mathrm{rk}(\nu, J)$.

Let us first show that $\overline{\lambda}$ indeed has nonnegative entries. Assume the contrary that $\overline{\lambda}_r < 0$. This can happen only if $\lambda_r = 0$. Suppose $t \in \mathcal{A}(\lambda')$ is such that $M_j^{(k)}(t) \leq p_j^{(k)}(\nu)$ for all j, k. By (2.3.3), $p_i^{(r)}(\nu) = -\lambda_{r+1}$ for large i. Let ℓ be the size of the largest part in $\nu^{(r)}$, so that $m_j^{(r)}(\nu) = 0$ for $j > \ell$. By definition of vacancy numbers, $p_i^{(r)}(\nu) \geq p_j^{(r)}(\nu)$ for $i \geq j \geq \ell$. Also we have $M_j^{(r)}(t) \geq -\lambda_{r+1}$ for all j. Hence, $-\lambda_{r+1} \leq M_j^{(r)}(t) \leq p_j^{(r)}(\nu) \leq p_i^{(r)}(\nu) = -\lambda_{r+1}$ implies

$$M_i^{(r)}(t) = M_j^{(r)}(t) = p_j^{(r)}(\nu) = p_i^{(r)}(\nu) \quad \text{for all } \ell \leq j \leq i. \tag{2.5.3}$$

This means that the string of length ℓ in $(\nu, J)^{(r)}$ is singular and $\Delta p_j^{(r)}(t) = 0$ for all $j \geq \ell$. We claim that $m_j^{(r-1)}(\nu) = 0$ for $j > \ell$. By (2.3.6) we get

$$\begin{aligned} S :=& -\chi(j \in t_{\cdot,r}) + \chi(j \in t_{\cdot,r+1}) + \chi(j+1 \in t_{\cdot,r}) - \chi(j+1 \in t_{\cdot,r+1}) \\ \geq& m_j^{(r-1)}(\nu) + m_j^{(r+1)}(\nu) \end{aligned}$$

for $j > \ell$. Clearly, $m_j^{(r-1)}(\nu) = 0$ unless $1 \leq S \leq 2$. If $S = 2$ we have $j+1 \in t_{\cdot,r}$ and $j \in t_{\cdot,r+1}$ which implies $M_j^{(r)}(t) = M_{j+1}^{(r)}(t) + 1$, a contradiction to (2.5.3). Hence $S = 2$ is not possible. Similarly, we can show that $S = 1$ is not possible. This proves that $m_j^{(r-1)}(\nu) = 0$ for $j > \ell$. Hence $\ell^{(r-1)} \leq \ell$ which contradicts the assumption that $r = \mathrm{rk}(\nu, J)$ since $(\nu, J)^{(r)}$ has a singular string of length ℓ. Therefore $\lambda_r > 0$.

Next we need to show that $(\overline{\nu}, \overline{J})$ is admissible, which means that the parts of $\overline{J}$ lie between the corresponding lower bound for some $\overline{t} \in \mathcal{A}(\overline{\lambda}')$ and the vacancy number. Let $t \in \mathcal{A}(\lambda')$ be such that (ν, J) is admissible with respect to t. By the same arguments as in the proof of Proposition 3.12 of [48] the only problematic case is when

$$m_{\ell-1}^{(k)}(\nu) = 0, \quad \Delta p_{\ell-1}^{(k)}(t) = 0, \quad \ell^{(k-1)} < \ell \quad \text{and } \ell \text{ finite} \tag{2.5.4}$$

where $\ell = \ell^{(k)}$.

Assume that $\Delta p_{\ell-2}^{(k)}(t) + \Delta p_\ell^{(k)}(t) \geq 1$ and (2.5.4) holds. By Remark 2.5.1 with $i = \ell-1$, there exists a new tableau t' such that $\Delta p_{\ell-1}^{(k)}(t') > 0$ so that the problematic case is avoided.

Hence assume that $\Delta p_{\ell-2}^{(k)}(t) + \Delta p_\ell^{(k)}(t) = 0$ and (2.5.4) holds. Let $\ell' < \ell$ be maximal such that $m_{\ell'}^{(k)}(\nu) > 0$. If no such ℓ' exists, set $\ell' = 0$.

Suppose that there exists $\ell' < j < \ell$ such that $\Delta p_{j-1}^{(k)}(t) > 0$. Let i be the maximal such j. Then by Remark 2.5.1 we can find a new tableau t' such that $\Delta p_i^{(k)}(t') > 0$ and (ν, J)

is admissible with respect to t'. Repeating the argument we can achieve $\Delta p_{\ell-1}^{(k)}(t'') > 0$ for some new tableau t'', so that the problematic case does not occur.

Hence we are left to consider the case $\Delta p_i^{(k)}(t) = 0$ for all $\ell' \leq i \leq \ell$. If $m_i^{(k-1)}(\nu) = 0$ for all $\ell' < i < \ell$, then by the same arguments as in the proof of Proposition 3.12 of [48] we arrive at a contradition since $\ell^{(k-1)} \leq \ell'$, but the string of length ℓ' in $(\nu, J)^{(k)}$ is singular which implies that $\ell^{(k)} \leq \ell' < \ell$. Hence there must exist $\ell' < i < \ell$ such that $m_i^{(k-1)}(\nu) > 0$ and $\ell^{(k-1)} = i$. By (2.3.6) the same five cases as in Remark 2.5.1 occur as possibilities for the letters i and $i+1$ in columns k and $k+1$ of t. In cases 3, 4 and case 5 if $m_i^{(k-1)}(\nu) = 2$, we have $m_i^{(k+1)}(\nu) = 0$. Replace i in column $k+1$ by $i+1$ in t to get a new tableau t'. In all other cases $m_i^{(k-1)}(\nu) = 1$; replace the letter $i+1$ in column k by i to obtain t'. The replacement $t \mapsto t'$ yields $\Delta p_i^{(k)}(t') > 0$ in all cases. The change of lower bound $M_i^{(k-1)}(t') = M_i^{(k-1)}(t) + 1$ in cases 1, 2 and 5 when $m_i^{(k-1)}(\nu) \neq 2$ will not cause any problems since $m_i^{(k-1)}(\nu) = 1$ so that after the application of δ there is no part of length i in the $(k-1)$-th rigged partition. Then again repeated application of Remark 2.5.1 achieves $\Delta p_{\ell-1}^{(k)}(t'') > 0$ for some tableau t'', so that the problematic case does not occur.

Let t'' be the tableau we constructed so far. Note that in all constructions above, either a letter $i+1$ in column k is changed to i, or a letter i in column $k+1$ is changed to $i+1$. In the latter case $i+1 \leq \ell \leq |\nu^{(k)}| \leq c_k$. Hence t'' satisfies the constraint that $t''_{i,k} \in \{1, 2, \ldots, c_{k-1}\}$ for all i, k.

Now let $\bar{t} = \mathcal{D}_r(t'')$. We know $\bar{t} \in \mathcal{A}(\overline{\lambda}')$. We will show that the parts of $\overline{J}$ lie between the corresponding lower bound with respect to $\bar{t} \in \mathcal{A}(\overline{\lambda}')$ and the vacancy number.

If $t''_{1,r} < c_{r-1}$ then by Lemma 2.5.3 $M_i^{(k)}(\bar{t}) \leq M_i^{(k)}(t'')$ for all k and i such that $m_i^{(k)}(\overline{\nu}) > 0$. Hence by Lemma 2.3.6 we have that $(\overline{\nu}, \overline{J})$ is admissible with respect to $\bar{t}$.

Let $t''_{1,r} = c_{r-1}$. Then there exists j as in the definition of $\mathcal{D}_r$. We claim that

(i) $m_i^{(r-1)}(\nu) = 0$ for $i > c_{r-1} - j$ and $m_{c_{r-1}-j}^{(r-1)}(\nu) \leq 1$.

(ii) If $m_{c_{r-1}-j}^{(r-1)}(\nu) = 1$, then $\ell^{(r-1)} = c_{r-1} - j$.

Note that $M_i^{(r-1)}(\bar{t}) = M_i^{(r-1)}(t'') + 1$ for $c_{r-1} - j \leq i < c_{r-1}$ and $M_i^{(k)}(\bar{t}) \leq M_i^{(k)}(t'')$ for all other k and i such that $m_i^{(k)}(\overline{\nu}) > 0$. Hence if the claim is true using Lemma 2.5.3 we have $M_i^{(k)}(\bar{t}) \leq M_i^{(k)}(t'')$ for all k and i such that $m_i^{(k)}(\overline{\nu}) > 0$. Therefore by Lemma 2.3.6 we have that $(\overline{\nu}, \overline{J})$ is admissible with respect to $\bar{t}$.

It remains to prove the claim. Note that if $|\nu^{(r-1)}| < c_{r-1} - j$ then our claim is trivially true. Let $|\nu^{(r-1)}| = c_{r-1} - k$ for some $0 \leq k \leq j$. If all parts of $\nu^{(r-1)}$ are strictly less than $c_{r-1} - j$, again our claim is trivially true. Let the largest part in $\nu^{(r-1)}$ be $c_{r-1} - p \geq c_{r-1} - j$ for some $k \leq p \leq j$. Let a be the largest part in $\nu^{(r)}$.

First suppose $a > c_{r-1} - p$ and $a = c_r - q$ for some $0 \leq q < c_r$. Then $a = c_{r-1} - (\lambda_r + q)$ which implies that

$$M_a^{(r)}(t'') \geq -(c_r - \lambda_r - q) + (c_{r+1} - q) = \lambda_r - \lambda_{r+1}.$$

This means $p_a^{(r)}(\nu) \leq M_a^{(r)}(t'')$ since $p_b^{(r)}(\nu) \geq p_a^{(r)}(\nu)$ for all $b \geq a$ and $p_b^{(r)} = \lambda_r - \lambda_{r+1}$ for large b. If $p_a^{(r)}(\nu) < M_a^{(r)}(t'')$, it contradicts that $p_a^{(r)}(\nu) \geq M_a^{(r)}(t'')$. If $p_a^{(r)}(\nu) = M_a^{(r)}(t'')$, it contradicts the fact that $r = \mathrm{rk}(\nu, J)$ since we get a singular part of length a in $\nu^{(r)}$ which is larger than the largest part in $\nu^{(r-1)}$. Therefore $a > c_{r-1} - p$ is not possible.

Hence $a \le c_{r-1} - p$. Using Remark 2.5.4 we get,

$$\begin{aligned} p_a^{(r)}(\nu) &= Q_a(\nu^{(r-1)}) - 2|\nu^{(r)}| + Q_a(\nu^{(r+1)}) + \sum_{i\ge 1} \min(a,i) L_i^{(r)} \\ &\le a + p - k - 2|\nu^{(r)}| + |\nu^{(r+1)}| + \sum_{i\ge 1} \min(a,i) L_i^{(r)} \\ &= a + p - 2\lambda_{r+1} - c_{r+1} - \sum_{i\ge 1} \max(s_i - a, 0)\delta_{r_i,r}. \end{aligned} \tag{2.5.5}$$

Since $p_a^{(r)}(\nu) \ge M_a^{(r)}(t'') \ge -\lambda_{r+1}$ we get

$$c_r - (p - \sum_{i\ge 1} \max(s_i - a, 0)\delta_{r_i,r}) \le a \le c_r.$$

Hence $a = c_r - q$ for $0 \le q \le p - \sum_{i\ge 1} \max(s_i - a, 0)\delta_{r_i,r}$. Then from (2.5.5) with $a = c_r - q$ we get

$$p_a^{(r)}(\nu) \le p - q - \lambda_{r+1} - \sum_{i\ge 1} \max(s_i - a, 0)\delta_{r_i,r} \le \lambda_r - \lambda_{r+1}, \tag{2.5.6}$$

where we used that $0 \le p - q \le \lambda_r$ which follows from $a = c_r - q \le c_{r-1} - p$.

If $a > c_{r-1} - j$, as in the case $a > c_{r-1} - p$ we have

$$M_a^{(r)}(t'') \ge -(c_r - \lambda_r - q) + (c_{r+1} - q) = \lambda_r - \lambda_{r+1} \ge p_a^{(r)}(\nu).$$

Hence we get a contradiction unless $p_a^{(r)}(\nu) = M_a^{(r)}(t'')$. By (2.5.6) and the fact that $0 \le p - q \le \lambda_r$ we know $p_a^{(r)}(\nu) = \lambda_r - \lambda_{r+1}$ happens only when $p - q = \lambda_r$ and $\sum_{i\ge 1} \max(s_i - a, 0)\delta_{r_i,r} = 0$. This means the largest part in $\nu^{(r-1)}$ is of length $c_{r-1} - p = c_r - q = a$. Since we have a singular string of length a in $\nu^{(r)}$ this contradicts the fact that $r = \mathrm{rk}(\nu, J)$.

If $a \leq c_{r-1}-j$ then $M_a^{(r)}(t'') \geq -(c_r-j)+(c_{r+1}-q) = j-q-\lambda_{r+1} \geq p_a^{(r)}(\nu)$ because of (2.5.6) and the fact that $j \geq p$. Again we get a contradiction unless $p_a^{(r)}(\nu) = M_a^{(r)}(t'')$. But this happens only when $p_a^{(r)}(\nu) = j-q-\lambda_{r+1}$ which gives $p = j$ because $p_a^{(r)}(\nu)$ attains the right hand side of (2.5.6). This means the largest part in $\nu^{(r-1)}$ is $c_{r-1}-j$. Furthermore, for large i we have $p_i^{(r)} = \lambda_r - \lambda_{r+1} \geq j-q-\lambda_{r+1}+(c_{r-1}-j-a) = \lambda_r - \lambda_{r+1}$ which shows that besides $c_{r-1}-j$ all parts in $\nu^{(r-1)}$ have to be less than or equal to a. But the part of length a in $\nu^{(r)}$ is singular, so we have to have $c_{r-1}-j > a$ and $\ell^{(r-1)} = c_{r-1}-j$ else it will contradict the fact that $r = \mathrm{rk}(\nu, J)$. This proves our claim.

Hence $(\overline{\nu}, \overline{J})$ is admissible with respect to $\overline{t} \in \mathcal{A}(\overline{\lambda}')$ and therefore δ is well-defined. $\square$

Example 2.5.5. Let L be the multiplicity array of $B = (B^{1,1})^{\otimes 4}$ and $\lambda = (0,1,0,1,2)$. Let

$$(\nu, J) = \begin{array}{|c|c|c|l}\cline{1-3} & & & -1 \\ \cline{1-3} & \multicolumn{3}{l}{2} \\ \cline{1-1} \end{array} \quad \begin{array}{|c|c|l}\cline{1-2} & & 0 \\ \cline{1-2} & \multicolumn{2}{l}{0} \\ \cline{1-1} \end{array} \quad \begin{array}{|c|c|l}\cline{1-2} & & -1 \\ \cline{1-2} & \multicolumn{2}{l}{-1} \\ \cline{1-1} \end{array} \quad \begin{array}{|c|c|l}\cline{1-2} & & -1 \\ \cline{1-2} \end{array} \in \mathrm{RC}(L, \lambda).$$

Let $t = \begin{array}{|c|c|c|c|}\hline 4 & 4 & 3 & 3 \\ \hline 3 & 2 & 2 & 2 \\ \hline 2 & 1 & 1 \\ \cline{1-3} 1 \\ \cline{1-1} \end{array}$ be the corresponding lower bound tableau. Then

$$\delta(\nu, J) = \begin{array}{|c|c|c|l}\cline{1-3} & & & -1 \\ \cline{1-3} \end{array} \quad \begin{array}{|c|c|l}\cline{1-2} & & 0 \\ \cline{1-2} \end{array} \quad \begin{array}{|c|c|l}\cline{1-2} & & -1 \\ \cline{1-2} \end{array} \quad \begin{array}{|c|l}\cline{1-1} & -1 \\ \cline{1-1} \end{array}.$$

Note that in this example $\ell = \ell^{(4)} = 2$ and it satisfies (2.5.4) with $k = 4$. Also $\Delta p_{\ell-2}^{(4)}(t) + \Delta p_{\ell}^{(4)}(t) = 0$ with $\Delta p_i^{(4)}(t) = 0$ for all $0 \leq i \leq \ell$. Since $m_1^{(3)}(\nu) = 1$ and $2 \in t_{.,4}$ this is an example where we get the new tableau t' by replacing the $2 \in t_{.,4}$ by 1 and then the corresponding lower bound tableau for $\delta(\nu, J)$ is $\mathcal{D}_5(t') = \begin{array}{|c|c|c|c|}\hline 3 & 2 & 2 & 1 \\ \hline 2 & 1 & 1 \\ \cline{1-3} 1 \\ \cline{1-1} \end{array}$.

Proof of Proposition 2.4.6. Similar to Proposition 2.4.3 we need to show that for $(\overline{\nu}, \overline{J}) \in$

$\mathrm{RC}(\overline{L}, \overline{\lambda})$ we have $\delta^{-1}(\overline{\nu}, \overline{J}) = (\nu, J) \in \mathrm{RC}(L, \lambda)$ where $\lambda = \overline{\lambda} + \epsilon_r$. Clearly λ has nonnegative parts, so it suffices to show that (ν, J) is admissible which means that the parts of J lie between the corresponding lower bound with respect to some $t \in \mathcal{A}(\lambda')$ and the vacancy number. Let $\overline{t} \in \mathcal{A}(\overline{\lambda}')$ be a tableau such that $(\overline{\nu}, \overline{J})$ is admissible with respect to $\overline{t}$. By similar argument as in the proof of Propostion 2.4.3 the only problematic case occurs when

$$m_{s+1}^{(k)}(\overline{\nu}) = 0, \quad \Delta p_{s+1}^{(k)}(\overline{t}) = 0, \quad s < s^{(k+1)} \quad \text{and } s \text{ finite} \tag{2.5.7}$$

where $s = s^{(k)}$.

Assume that $\Delta p_s^{(k)}(\overline{t}) + \Delta p_{s+2}^{(k)}(\overline{t}) \geq 1$ and (2.5.7) holds. By Remark 2.5.1 with $i = s+1$ there exists a new tableau $\overline{t}'$ such that $\Delta p_{s+1}^{(k)}(\overline{t}') > 0$ so that the problematic case is avoided.

Hence assume that $\Delta p_s^{(k)}(\overline{t}) + \Delta p_{s+2}^{(k)}(\overline{t}) = 0$ and (2.5.7) holds. Let $s' > s$ be minimal such that $m_{s'}^{(k)}(\overline{\nu}) > 0$. If no such s' exists, set $s' = \infty$.

Suppose that there exists $s' > j > s$ such that $\Delta p_{j+1}^{(k)}(\overline{t}) > 0$. Let i be the minimal such j. Then by Remark 2.5.1 we can find a new tableau $\overline{t}'$ such that $\Delta p_i^{(k)}(\overline{t}') > 0$ and $(\overline{\nu}, \overline{J})$ is admissible with respect to $\overline{t}'$. Repeating the argument we can achieve $\Delta p_{s+1}^{(k)}(\overline{t}'') > 0$ for some new tableau $\overline{t}''$, so that the problematic case does not occur.

Hence we are left to consider the case $\Delta p_i^{(k)}(\overline{t}) = 0$ for all $s' \geq i \geq s$. First let us suppose $k < r-1$. If $m_i^{(k+1)}(\overline{\nu}) = 0$ for all $s' > i > s$, then by the similar arguments as in the proof of Proposition 2.4.3 we arrive at a contradiction since $s^{(k+1)} \geq s'$, but the string of length s' in $(\overline{\nu}, \overline{J})^{(k)}$ is singular which implies that $s^{(k+1)} > s^{(k)} \geq s' > s$. Hence there must exist $s' > i > s$ such that $m_i^{(k+1)}(\overline{\nu}) > 0$ and $s^{(k+1)} = i$. By (2.3.6) the same five cases as in Remark 2.5.1 occur as possibilities for the letters i and $i+1$ in columns k and $k+1$ of $\overline{t}$. In cases 1, 2 and case 5 if $m_i^{(k+1)}(\overline{\nu}) = 2$, we have $m_i^{(k-1)}(\overline{\nu}) = 0$. Replace $i+1$ in column k by i in $\overline{t}$ to get a new tableau $\overline{t}'$. In all other cases $m_i^{(k-1)}(\overline{\nu}) = 1$;

replace the letter i in column $k+1$ by $i+1$ to obtain $\bar{t}'$. The replacement $\bar{t} \mapsto \bar{t}'$ yields $\Delta p_i^{(k)}(\bar{t}') > 0$ in all cases. The change of lower bound $M_i^{(k+1)}(\bar{t}') = M_i^{(k+1)}(\bar{t}) + 1$ in cases 3, 4 and 5 when $m_i^{(k+1)} \neq 2$ will not cause any problems since $m_i^{(k+1)} = 1$ so that after the application of δ^{-1} there is no part of length i in the $(k+1)$-th rigged partition. Then again repeated application of Remark 2.5.1 achieves $\Delta p_{s+1}^{(k)}(\bar{t}'') > 0$ for some tableau $\bar{t}''$, so that the problematic case does not occur.

Now let us consider the case $k = r-1$. Note that $s' = \infty$ here. Else $s^{(r-1)} > s$, a contradiction. So, $\Delta p_i^{(r-1)}(\bar{t}) = 0$ for $i > s$ which implies $m_i^{(r-1)}(\bar{\nu}) = 0$ for $i > s$, else $s^{(r-1)} > s$. Then by (2.3.6) with $i \geq s+1$ and $k = r-1$ we have

$$\begin{aligned} -\chi(i \in \bar{t}_{\cdot,r-1}) &+ \chi(i \in \bar{t}_{\cdot,r}) + \chi(i+1 \in \bar{t}_{\cdot,r-1}) - \chi(i+1 \in \bar{t}_{\cdot,r}) \\ &\geq m_i^{(r-2)}(\bar{\nu}) + m_i^{(r)}(\bar{\nu}) \geq 0. \end{aligned} \tag{2.5.8}$$

If $s+1 \in \bar{t}_{\cdot,r}$ by (2.5.8) with $i = s+1$ there are seven choices for the letters $s+1$ and $s+2$ in columns $r-1$ and r of $\bar{t}$.

1. $s+1$ in both columns $r-1$ and r;

2. Both $s+1, s+2$ in column r;

3. Both $s+1, s+2$ in columns $r-1, r$;

4. $s+1$ in columns $r-1, r$ and $s+2$ in column $r-1$;

5. $s+1$ in column r;

6. $s+1$ in column r and $s+2$ in columns $r-1, r$;

7. $s+1$ in column r and $s+2$ in column $r-1$.

First note that by (2.5.8) $m_{s+1}^{(r-2)}(\overline{\nu}) = m_{s+1}^{(r)}(\overline{\nu}) = 0$ for cases 1, 2 and 3. For case 4 we have $m_{s+1}^{(r)}(\overline{\nu}) = 0$ again, else $p_{s+1}^{(r-1)}(\overline{t}) > p_s^{(r-1)}(\overline{t}) = M_s^{(r-1)}(\overline{t}) = M_{s+1}^{(r-1)}(\overline{t})$, contradiction to $\Delta p_{s+1}^{(r-1)}(\overline{t}) = 0$. In cases 5 and 6 either $m_{s+1}^{(r)}(\overline{\nu}) = 0$ or $m_{s+1}^{(r-2)}(\overline{\nu}) = 0$ by (2.5.8). When $m_{s+1}^{(r-2)}(\overline{\nu}) = 0$ and $m_{s+1}^{(r)}(\overline{\nu}) > 0$ in case 5 we have $m_i^{(r-2)}(\overline{\nu}) = 0$ for all $i > s+1$, else $p_{s+1}^{(r-1)}(\overline{\nu}) \geq p_s^{(r-1)}(\overline{\nu}) + 2 = M_s^{(r-1)}(\overline{t}) + 2 \geq M_{s+1}^{(r-1)}(\overline{t}) - 1 + 2 > M_{s+1}^{(r-1)}(\overline{t})$, a contradiction. In case 7 by the same string of inequalities either $m_{s+1}^{(r)}(\overline{\nu}) = 0$ or $m_{s+1}^{(r-2)}(\overline{\nu}) = 0$.

When $m_{s+1}^{(r)}(\overline{\nu}) = 0$ we construct a new tableau $\overline{t}'$ from $\overline{t}$ by replacing $s+1$ in column r by the smallest number $i > s+1$ that does not appear in column r of $\overline{t}$. The effect of this change is $M_{s+1}^{(r)}(\overline{t}') = M_{s+1}^{(r)}(\overline{t}) + 1$ and $M_{s+1}^{(r-1)}(\overline{t}') = M_{s+1}^{(r-1)}(\overline{t}) - 1$. Since $m_{s+1}^{(r)}(\overline{\nu}) = 0$ the first change does not create any problem. When $m_{s+1}^{(r)}(\overline{\nu}) > 0$ in cases 6 and 7 we change the $s+2$ in column $r-1$ to $s+1$. The effect of this replacement is $M_{s+1}^{(r-2)}(\overline{t}') = M_{s+1}^{(r-2)}(\overline{t}) + 1$ and $M_{s+1}^{(r-1)}(\overline{t}') = M_{s+1}^{(r-1)}(\overline{t}) - 1$. Since $m_{s+1}^{(r-2)}(\overline{\nu}) = 0$ there is no problem. When $m_{s+1}^{(r)}(\overline{\nu}) > 0$ in case 5 we replace the smallest $\overline{t}_{j,r-1} > s+1$ by $s+1$. This has the effect that $M_i^{(r-2)}(\overline{t}') = M_i^{(r-2)}(\overline{t}) + 1$ for $s+1 \leq i < \overline{t}_{j,r-1}$. Since we have $m_i^{(r-2)} = 0$ for all $i \geq s+1$ we do not have any problem. In all cases, replacing $\overline{t}$ by $\overline{t}'$ the problematic case (2.5.7) is avoided and we have $\Delta p_i^{(k)}(\overline{t}') \geq 0$ for all other i, k such that $m_i^{(k)}(\overline{\nu}) > 0$.

Let us consider the case $s+1 \not\in \overline{t}_{\cdot,r}$. Note that $M_s^{(r-1)}(\overline{t}) \geq M_{s+1}^{(r-1)}(\overline{t})$. We have $m_i^{(r)}(\overline{\nu}) = 0 = m_i^{(r-2)}(\overline{\nu})$ for all $i > s$, else $p_{s+1}^{(r-1)}(\overline{\nu}) > p_s^{(r-1)}(\overline{\nu}) = M_s^{(r-1)}(\overline{t}) \geq M_{s+1}^{(r-1)}(\overline{t})$, contradiction to $\Delta p_{s+1}^{(r-1)}(\overline{t}) = 0$. Using (2.5.8) for $i = s+1, k = r-1$ we have four possible cases for the choice of the letters $s+1$ and $s+2$ in columns $r-1$ and r of $\overline{t}$.

1. $s+2$ in column $r-1$;

2. $s+2$ in columns $r-1$ and r;

3. $s+1$ and $s+2$ in column $r-1$;

4. no $s+1, s+2$ in both columns $r-1$ and r.

We first argue that case 3 cannot occur. Suppose case 3 holds. Then $M_{s+1}^{(r-1)}(\overline{t}) = M_s^{(r-1)}(\overline{t}) - 1$ and $M_{s+2}^{(r-1)}(\overline{t}) = M_{s+1}^{(r-1)}(\overline{t}) - 1$. But we also have $\Delta p_i^{(r-1)}(\overline{t}) = 0$ for $i > s$ and $m_i^{(r-1)}(\overline{\nu}) = m_i^{(r-2)}(\overline{\nu}) = m_i^{(r)}(\overline{\nu})$ for $i > s$. Note that $\Delta p_i^{(r-1)}(\overline{t}) = 0$ implies that $p_{s+2}^{(r-1)}(\overline{\nu}) = p_{s+1}^{(r-1)}(\overline{\nu}) - 1 = p_s^{(r-1)}(\overline{\nu}) - 2$. On the other hand $m_i^{(r-1)}(\overline{\nu}) = m_i^{(r-2)}(\overline{\nu}) = m_i^{(r)}(\overline{\nu})$ implies that $p_{s+2}^{(r-1)}(\overline{\nu}) \geq p_s^{(r-1)}(\overline{\nu})$ and $p_{s+1}^{(r-1)}(\overline{\nu}) \geq p_s^{(r-1)}(\overline{\nu})$ which yields a contradiction.

In cases 1 and 2 we replace the letter $s+2$ in column $r-1$ to $s+1$ to get a new tableau $\overline{t}'$. The change from $\overline{t}$ to $\overline{t}'$ yields $\Delta p_{s+1}^{(r-1)}(\overline{t}') > 0$ without any other change. In case 4 if there exists $\overline{t}_{j,r-1} > s+2$ for some j then we replace the smallest such $\overline{t}_{j,r-1}$ by $s+1$ to construct $\overline{t}'$. Then again we get $\Delta p_{s+1}^{(r-1)}(\overline{t}') > 0$ without any other change since $m_i^{(r-2)}(\overline{\nu}) = 0$ for all $i > s$. On the other hand if $\overline{t}_{1,r-1} \leq s$ then $\overline{c}_{r-1} \leq s \leq |\overline{\nu}^{(r-1)}| \leq \overline{c}_{r-1}$ implies $\overline{t}_{1,r-1} = s$. Note that $\overline{t}_{1,r-2} \geq s$. Here we will avoid the problematic case (2.5.7) by constructing a new tableau $t \in \mathcal{A}(\lambda')$. Let

$$t_{i,k} = \begin{cases} \overline{c}_1 + 1 & \text{for } k = 1 = i \\ \overline{c}_{k-1} + 1 & \text{for } 2 \leq k \leq r-2 \text{ and } i = 1, \\ s+1 & \text{for } k = r-1 \text{ and } i = 1, \\ \overline{t}_{i-1,k} & \text{for } 1 \leq k \leq r-1 \text{ and } 1 < i \leq \overline{c}_k, \\ \overline{t}_{i,k} & \text{for } r \leq k \leq n \text{ and } 1 \leq i \leq \overline{c}_k. \end{cases} \tag{2.5.9}$$

Note that $c_k = \overline{c}_k + 1$ for $1 \leq k \leq r-1$ and $c_k = \overline{c}_k$ for $r \leq k \leq n$. Clearly $t_{i,k} \in \{1, 2, \ldots, c_{k-1}\}$ for all i, k. Column-strictness of t follows since $\overline{t}_{1,1} < \overline{c}_1 + 1$ and $\overline{t}_{1,k} < \overline{c}_k + 1 \leq \overline{c}_{k-1} + 1$ for $2 \leq k \leq r-1$ and $s+1 > \overline{t}_{1,r}$. Hence $t \in \mathcal{A}(\lambda')$. Note that we have $M_{s+1}^{(r-1)}(t) = M_{s+1}^{(r-1)}(\overline{t}) - 1 < p_{s+1}^{(r-1)}(\overline{\nu})$, so the problematic case (2.5.7) is avoided. The fact that (ν, J) is admissible with respect to t is shown later.

Let us now define $t \in \mathcal{A}(\lambda')$ in all other cases. Let $\overline{t}'' \in \mathcal{A}(\overline{\lambda}')$ be the tableau we constructed from $\overline{t}$ so far except in the last case. Note that in all constructions above, either a letter $i+1$ in column k is changed to i, or a letter i in column $k+1$ is changed to $i+1$. In the latter case $m_i^{(k+1)} = 0$ means $i+1 \leq s^{(k+1)} \leq |\nu^{(k+1)}| \leq \overline{c}_{k+1} \leq \overline{c}_k$. Hence $\overline{t}''$ satisfies the constraint that $\overline{t}''_{i,k} \in \{1, 2, \ldots, \overline{c}_{k-1}\}$ for all i, k.

Let us define a new tableau t from $\overline{t}''$ in the following way:

$$t_{i,k} = \begin{cases} \overline{c}_1 + 1 & \text{for } k = 1 = i \\ \overline{c}_{k-1} + 1 & \text{for } 2 \leq k \leq r-1 \text{ and } i = 1, \\ \overline{t}''_{i-1,k} & \text{for } 1 \leq k \leq r-1 \text{ and } 1 < i \leq \overline{c}_k, \\ \overline{t}''_{i,k} & \text{for } r \leq k \leq n \text{ and } 1 \leq i \leq \overline{c}_k. \end{cases} \tag{2.5.10}$$

Similarly as in (2.5.9) we have $t \in \mathcal{A}(\lambda')$.

Next we show that (ν, J) is admissible with respect to t, that is, the parts of J lie between the corresponding lower bound with respect to $t \in \mathcal{A}(\lambda')$ and the vacancy number. Note that $s^{(k)} + 1 \leq |\nu^{(k)}| \leq c_k \leq c_{k-1}$. We distinguish the three cases $s^{(k)} + 1 < c_k$, $s^{(k)} + 1 = c_k = c_{k-1}$ and $s^{(k)} + 1 = c_k < c_{k-1}$.

If $s^{(k)} + 1 < c_k$ for all $1 \leq k \leq r-1$, then $M_i^{(k)}(t) = M_i^{(k)}(\overline{t}'')$ for all i, k such that $m_i^{(k)}(\nu) > 0$. If $s^{(k)} + 1 = c_{k-1}$ for some $1 \leq k \leq r-2$, then $M_{s^{(k)}+1}^{(k)}(t) = M_{s^{(k)}+1}^{(k)}(\overline{t}'')$ since $c_{k-1} \geq c_k$. Also if $s^{(r-1)} + 1 = c_{r-2}$, then $M_{s^{(r-1)}+1}^{(r-1)}(t) = M_{s^{(r-1)}+1}^{(r-1)}(\overline{t}'') - 1$. In both cases (ν, J) is admissible since $M_i^{(k)}(t) \leq M_i^{(k)}(\overline{t}'')$ for all i, k such that $m_i^{(k)}(\nu) > 0$.

Now suppose $s^{(k)} + 1 = c_k < c_{k-1}$ for some $1 \leq k < r-1$. Then $M_{s^{(k)}+1}^{(k)}(t) =$

$M^{(k)}_{s^{(k)}+1}(\bar{t}'')+1$. Suppose k is minimal satisfying this condition. Note that in this situation, $s^{(k)} = c_k - 1 = \bar{c}_k$. This means $|\overline{\nu}^{(k)}| = \bar{c}_k$ which implies by definition of $|\overline{\nu}^{(k)}|$ that $|\overline{\nu}^{(a)}| = \bar{c}_a$ for $a \geq k$. Using this we get

$$\bar{c}_k = s^{(k)} \leq s^{(k+1)} \leq \cdots \leq s^{(a)} \leq \cdots \leq s^{(r-1)} \leq |\overline{\nu}^{(r-1)}| = \bar{c}_{r-1} \leq \bar{c}_k.$$

This implies $\bar{c}_a = s^{(a)} = s^{(a+1)} = \bar{c}_{a+1}$ for all $k \leq a \leq r-2$. When $s^{(a)} = s^{(a+1)}$ we have $p^{(a)}_{s^{(a)}+1}(\nu) = p^{(a)}_{s^{(a)}+1}(\overline{\nu})$. Hence we only need to worry when $\Delta p^{(k)}_{s^{(k)}+1}(\bar{t}'') = 0$. Let ℓ be the largest part in $\overline{\nu}^{(k-1)}$. If $\ell > s^{(k)}$ then by definition $p^{(k)}_{s^{(k)}+1}(\overline{\nu}) > p^{(k)}_{s^{(k)}}(\overline{\nu})$. But we have $M^{(k)}_{s^{(k)}}(\bar{t}'') \geq M^{(k)}_{s^{(k)}+1}(\bar{t}'')$, hence $\Delta p^{(k)}_{s^{(k)}+1}(\bar{t}'') > 0$. Suppose $\ell \leq s^{(k)}$, then $p^{(k)}_{s^{(k)}+1}(\overline{\nu}) \geq p^{(k)}_{s^{(k)}}(\overline{\nu})$ since $m^{(k)}_i(\overline{\nu}) = 0$ for $i > s^{(k)}$. If $s^{(k)}+1 \in \bar{t}''_{\cdot,k}$ then $M^{(k)}_{s^{(k)}}(\bar{t}'') = M^{(k)}_{s^{(k)}+1}(\bar{t}'')+1$ and we get $\Delta p^{(k)}_{s^{(k)}+1}(\bar{t}'') > 0$. If $s^{(k)}+1 \notin \bar{t}''_{\cdot,k}$ then there exists $\bar{t}''_{j,k} > s^{(k)}+1$ for some j and we replace the smallest such $\bar{t}''_{j,k}$ by $s^{(k)}+1$ to get a new tableau t' from $t \in \mathcal{A}(\lambda')$. This has the effect that $M^{(k)}_{s^{(k)}+1}(t') = M^{(k)}_{s^{(k)}+1}(t) - 1 = M^{(k)}_{s^{(k)}+1}(\bar{t}'')$ so that $\Delta p^{(k)}_{s^{(k)}+1}(t') \geq 0$.

This proves that (ν, J) is admissible with respect to t or $t' \in \mathcal{A}(\lambda')$. Hence δ^{-1} is well-defined. $\square$

Example 2.5.6. Let $\overline{L}$ be the multiplicity array of $B = (B^{1,1})^{\otimes 4}$ and $\overline{\lambda} = (0,1,1,1,1)$. Let

$$(\overline{\nu}, \overline{J}) = \begin{array}{|c|c|c|l}\cline{1-3} &&& -1\\ \cline{1-3} & \multicolumn{3}{l}{1}\\ \cline{1-1}\end{array} \quad \begin{array}{|c|c|l}\cline{1-2} && -1\\ \cline{1-2} & \multicolumn{2}{l}{0}\\ \cline{1-1}\end{array} \quad \begin{array}{|c|l}\cline{1-1} & -1\\ \cline{1-1} & -1\\ \cline{1-1}\end{array} \quad \begin{array}{|c|l}\cline{1-1} & 0\\ \cline{1-1}\end{array} \in \mathrm{RC}(\overline{L}, \overline{\lambda}).$$

Let $\bar{t} = \begin{array}{|c|c|c|c|}\hline 4&4&3&2\\ \hline 3&2&1\\ \cline{1-3} 2&1\\ \cline{1-2} 1\\ \cline{1-1}\end{array}$ be the corresponding lower bound tableau. Then with $r = 3$,

$$\delta^{-1}(\overline{\nu}, \overline{J}) = \begin{array}{|c|c|c|l}\cline{1-3} &&& -1\\ \cline{1-3} & \multicolumn{3}{l}{1}\\ \cline{1-1} & \multicolumn{3}{l}{1}\\ \cline{1-1}\end{array} \quad \begin{array}{|c|c|c|l}\cline{1-3} &&& -1\\ \cline{1-3} & \multicolumn{3}{l}{1}\\ \cline{1-1}\end{array} \quad \begin{array}{|c|l}\cline{1-1} & -1\\ \cline{1-1} & -1\\ \cline{1-1}\end{array} \quad \begin{array}{|c|l}\cline{1-1} & 0\\ \cline{1-1}\end{array} \; .$$

Note that in this example we have $k = r-1 = 2$ and $s = s^{(2)} = 2$ which satisfies (2.5.7). Also $s+1 = 3 \in \bar{t}_{\cdot,r}$, hence this is the situation when $k = r-1$ in (2.5.7) with $\Delta p_i^{(r-1)}(\bar{t}) = 0$ for all $i > s$ and since $s+1 \in \bar{t}_{\cdot,r}$ this is case 7 discussed in the proof. So we get the corresponding lower bound tableau for (ν, J) by replacing $3 \in \bar{t}_{\cdot,r}$ by 4 and then doing the construction defined in (2.5.10). The lower bound tableau we get is $\begin{array}{|c|c|c|c|}\hline 5&5&4&2\\\hline 4&4&1\\\cline{1-3} 3&2\\\cline{1-2} 2&1\\\cline{1-2} 1\\\cline{1-1}\end{array}$.

2.6 Proof of Proposition 2.4.8

In this section a proof of Proposition 2.4.8 is given stating that the map Φ of Definition 2.4.7 is a well-defined bijection.

The proof proceeds by induction on B using the fact that it is possible to go from $B = B^{r_k,s_k} \otimes B^{r_{k-1},s_{k-1}} \otimes \cdots \otimes B^{r_1,s_1}$ to the empty crystal via successive application of lh, ls and lb. Suppose that B is the empty crystal. Then both sets $\mathcal{P}(B,\lambda)$ and $\mathrm{RC}(L,\lambda)$ are empty unless λ is the empty partition, in which case $\mathcal{P}(B,\lambda)$ consists of the empty partition and $\mathrm{RC}(L,\lambda)$ consists of the empty rigged configuration. In this case Φ is the unique bijection mapping the empty partition to the empty rigged configuration.

Consider the commutative diagram (1) of Definition 2.4.7. By induction

$$\Phi : \bigcup_{\mu\in\lambda^-} \mathcal{P}(\mathrm{lh}(B),\mu) \longrightarrow \bigcup_{\mu\in\lambda^-} \mathrm{RC}(\mathrm{lh}(L),\mu)$$

is a bijection. By Propositions 2.4.3 and 2.4.6 δ is a bijection, and by definition it is clear that lh is a bijection as well. Hence $\Phi = \delta^{-1} \circ \Phi \circ \mathrm{lh}$ is a well-defined bijection.

Suppose that $B = B^{r,1} \otimes B'$ with $r \geq 2$. By induction Φ is a bijection for $\mathrm{lb}(B) = B^{1,1} \otimes B^{r-1,1} \otimes B'$. Hence to prove that (3) uniquely determines Φ for B it suffices to show that Φ restricts to a bijection between the image of $\mathrm{lb} : \mathcal{P}(B,\lambda) \longrightarrow \mathcal{P}(\mathrm{lb}(B),\lambda)$ and the image of $\mathrm{lb}_{rc} : \mathrm{RC}(L,\lambda) \longrightarrow \mathrm{RC}(\mathrm{lb}(L),\lambda)$. Let $b = \boxed{b_r} \otimes \begin{array}{|c|}\hline b_1 \\ \hline \vdots \\ \hline b_{r-1} \\ \hline\end{array} \otimes b' \in \mathcal{P}(\mathrm{lb}(B),\lambda)$ with $b_{r-1} < b_r$. Let $(\nu, J) = \Phi(b)$ which is in $\mathrm{RC}(\mathrm{lb}(L),\lambda)$. We will show that $(\nu,J)^{(a)}$ has a singular string of length one for $1 \le a \le r-1$.

By induction we know for $(\overline{\nu}, \overline{J}) = \Phi(\bar{b})$ where $\bar{b} = \boxed{b_{r-1}} \otimes \begin{array}{|c|}\hline b_1 \\ \hline \vdots \\ \hline b_{r-2} \\ \hline\end{array} \otimes b' \in \mathrm{lb}(B^{r-1,1} \otimes B')$ with $b_{r-2} < b_{r-1}$, $(\overline{\nu}, \overline{J})^{(a)}$ has a singular string of length one for $1 \le a \le r-2$. Let $\overline{b}' = \begin{array}{|c|}\hline b_1 \\ \hline \vdots \\ \hline b_{r-1} \\ \hline\end{array} \otimes b'$ and $(\overline{\nu}', \overline{J}') = \Phi(\overline{b}')$. This "unsplitting" on the rigged configuration side removes the singular string of length one from $(\overline{\nu}, \overline{J})^{(a)}$ for $1 \le a \le r-2$ yielding $(\overline{\nu}', \overline{J}')$.

Let $\overline{s}^{(a)}$ be the length of the selected strings by δ^{-1} associated with b_{r-1}. Note that $\overline{s}^{(a)} = 0$ for $1 \le a \le r-2$. Now let $s^{(a)}$ be the selected strings by δ^{-1} associated with b_r. Since $b_{r-1} < b_r$ we have by construction that $s^{(a+1)} \le \overline{s}^{(a)}$. In particular $s^{(r-1)} \le \overline{s}^{(r-2)} = 0$ and therefore, $s^{(r-1)} = 0$. This implies that $s^{(a)} = 0$ for $1 \le a \le r-1$. Hence $(\nu,J)^{(a)}$ has a singular string of length one for $1 \le a \le r-1$.

Conversely, let $(\nu, J) \in \mathrm{lb}_{rc}(\mathrm{RC}(L,\lambda))$, that is, $(\nu,J)^{(a)}$ has singular string of length one for $1 \le a \le r-1$. Let $b = \Phi^{-1}(\nu,J) = \boxed{b_r} \otimes \begin{array}{|c|}\hline b_1 \\ \hline \vdots \\ \hline b_{r-1} \\ \hline\end{array} \otimes b' \in \mathcal{P}(\mathrm{lb}(B),\lambda)$. We want to show that $b_{r-1} < b_r$. Let $(\overline{\nu}, \overline{J}) = \delta(\nu, J)$ and $\ell^{(a)}$ be the length of the selected string in $(\nu,J)^{(a)}$ by δ. Then $\ell^{(a)} = 1$ for $1 \le a \le r-1$ and the change of vacancy numbers from

(ν, J) to $(\overline{\nu}, \overline{J})$ is given by

$$p_i^{(a)}(\overline{\nu}) = p_i^{(a)}(\nu) - \chi(\ell^{(a-1)} \leq i < \ell^{(a)}) + \chi(\ell^{(a)} \leq i < \ell^{(a+1)}). \tag{2.6.1}$$

This implies that $(\overline{\nu}, \overline{J})^{(r-1)}$ has no singular string of length less than $\ell^{(r)}$ since $\ell^{(r-1)} = 1$. Let $(\overline{\nu}', \overline{J}') = \mathrm{lb}_{rc}(\overline{\nu}, \overline{J})$. Denote by $\overline{\ell}^{(a)}$ the length of the singular string selected by δ in $(\overline{\nu}', \overline{J}')^{(a)}$. Then by induction $\overline{\ell}^{(a)} = 1$ for $1 \leq a \leq r-2$ and by (2.6.1) we get $\overline{\ell}^{(a)} \geq \ell^{(a+1)}$ for $a \geq r-1$. Therefore $\overline{\ell}^{(a)} \geq \ell^{(a+1)}$ for all $1 \leq a \leq n$. Hence $b_{r-1} < b_r$. This proves that Φ in (3) is uniquely determined.

Let us now consider the case $B = B^{r,s} \otimes B'$ where $s \geq 2$. Any map Φ satisfying (2) is injective by definition and unique by induction. To prove the existence and surjectivity it suffices to prove that bijection Φ maps the image of $\mathrm{ls} : \mathcal{P}(B, \lambda) \longrightarrow \mathcal{P}(\mathrm{ls}(B), \lambda)$ to the image of $\mathrm{ls}_{rc} : \mathrm{RC}(L, \lambda) \longrightarrow \mathrm{RC}(\mathrm{ls}(L), \lambda)$. Let $b = c_1 \otimes c \otimes b' \in \mathrm{ls}(\mathcal{P}(B, \lambda))$ where $c = c_2 c_3 \cdots c_s$ and c_i denotes the $(i-1)$-th column of $c \in B^{r,s-1}$. Let $c_1 = \begin{array}{|c|}\hline a_1 \\ \hline \vdots \\ \hline a_r \\ \hline\end{array} \in B^{r,1}$ and $c_2 = \begin{array}{|c|}\hline b_1 \\ \hline \vdots \\ \hline b_r \\ \hline\end{array}$, so that we have $a_i \leq b_i$ for $1 \leq i \leq r$. Let $(\nu, J) = \Phi(b)$. We want to show that $(\nu, J) \in \mathrm{ls}_{rc}(\mathrm{RC}(L, \lambda))$. To do that by definition of ls_{rc} it is enough to show that $(\nu, J)^{(r)}$ has no singular string of length less than s.

Let us introduce some further notation. Let $\overline{b} = c \otimes b'$ and $(\overline{\nu}_0, \overline{J}_0) = \Phi(c_3 \cdots c_s \otimes b')$. Define $(\overline{\nu}_i, \overline{J}_i) = (\mathrm{lb}_{rc}^{-1} \circ \delta^{-1})^{i-1} \circ \delta^{-1}(\overline{\nu}_0, \overline{J}_0)$ for $1 \leq i \leq r$ and let $\overline{s}_i^{(a)}$ be the length of the singular strings associated to b_i. Similarly define $(\nu_i, J_i) = (\mathrm{lb}_{rc}^{-1} \circ \delta^{-1})^{i-1} \circ \delta^{-1}(\nu_0, J_0)$ for $1 \leq i \leq r$ and let $s_i^{(a)}$ be the length of the singular strings associated to a_i where

$(\nu_0, J_0) = \Phi(\bar{b})$. The change of vacancy number from $(\overline{\nu}_0, \overline{J}_0)$ to $(\overline{\nu}_i, \overline{J}_i)$ is given by

$$p_k^{(a)}(\overline{\nu}_i) = p_k^{(a)}(\overline{\nu}_0) + \sum_{m=1}^{i} \chi(\overline{s}_m^{(a-1)} < k \leq \overline{s}_m^{(a)}) - \sum_{m=1}^{i} \chi(\overline{s}_m^{(a)} < k \leq \overline{s}_m^{(a+1)}), \tag{2.6.2}$$

and the change of vacancy number from $(\overline{\nu}_0, \overline{J}_0)$ to (ν_i, J_i) is given by

$$\begin{aligned} p_k^{(a)}(\nu_i) = p_k^{(a)}(\overline{\nu}_0) &+ \sum_{m=1}^{r} \chi(\overline{s}_m^{(a-1)} < k \leq \overline{s}_m^{(a)}) - \sum_{m=1}^{r} \chi(\overline{s}_m^{(a)} < k \leq \overline{s}_m^{(a+1)}) \\ &- \delta_{a,r}\chi(k < s-1) + \sum_{m=1}^{i} \chi(s_m^{(a-1)} < k \leq s_m^{(a)}) - \sum_{m=1}^{i} \chi(s_m^{(a)} < k \leq s_m^{(a+1)}). \end{aligned} \tag{2.6.3}$$

Using this we will show that $s_i^{(a)} > \overline{s}_i^{(a)}$ for all $a \geq i$ and $1 \leq i \leq r$ by induction on i. Note that by (2.6.2) in $(\nu_0, J_0)^{(a)}$ the strings of length $\overline{s}_i^{(a)} + 1$ remain singular for all i, a. Since $a_1 \leq b_1$ we have $s_1^{(a)} > \overline{s}_1^{(a)}$ for all a, this starts the induction. Let $s_i^{(a)} > \overline{s}_i^{(a)}$ for all a and for $1 \leq i \leq k$. Then by induction hypothesis and (2.6.3) in $(\nu_k, J_k)^{(a)}$ the strings of length $\overline{s}_i^{(a)} + 1$ remain singular for all a and $k+1 \leq i \leq r$, which implies that $s_{k+1}^{(a)} \geq \overline{s}_{k+1}^{(a)} + 1$. Hence $s_{k+1}^{(a)} > \overline{s}_{k+1}^{(a)}$ which proves our claim by induction. In particular $s_r^{(r)} > \overline{s}_r^{(r)}$. By induction $(\overline{\nu}_r, \overline{J}_r)^{(r)}$ has no singular string of length strictly less than $s-1$, so $\overline{s}_r^{(r)} \geq s-1$ which implies $s_r^{(r)} \geq s$. But note that by construction of the algorithm $s_r^{(a)} = 0$ for $1 \leq a \leq r-1$ and the change of vacancy numbers from (ν_{r-1}, J_{r-1}) to $(\nu_r, J_r) = (\nu, J)$ is given by,

$$p_k^{(a)}(\nu) = p_k^{(a)}(\nu_{r-1}) + \chi(s_r^{(a-1)} < k \leq s_r^{(a)}) - \chi(s_r^{(a)} < k \leq s_r^{(a+1)}).$$

This implies that $(\nu, J)^{(r)}$ has no singular string less than $s_r^{(r)}$ which means $(\nu, J)^{(r)}$ has no singular string less than s and we are done.

Conversely let $(\nu, J) \in \mathrm{ls}_{rc}(\mathrm{RC}(L,\lambda))$ and $b = \Phi^{-1}(\nu, J) = c_1 \otimes c \otimes b'$, same notation as before. We will show that $a_i \leq b_i$ for $1 \leq i \leq r$. Set $(\nu_i, J_i) = (\delta \circ \mathrm{lb})^{r-i}(\nu, J)$ for $1 \leq i \leq r$ and set $(\nu_0, J_0) = \delta \circ (\delta \circ \mathrm{lb})^{r-1}(\nu, J)$. Let us denote the length of the string selected by δ in $(\nu_i, J_i)^{(a)}$ by $\ell_i^{(a)}$. Similarly set $(\overline{\nu}, \overline{J}) = \mathrm{ls}_{rc}(\nu_0, J_0)$ and $(\overline{\nu}_i, \overline{J}_i) = (\delta \circ \mathrm{lb})^{r-i}(\overline{\nu}, \overline{J})$ for $1 \leq i \leq r$ and $(\overline{\nu}_0, \overline{J}_0) = \delta \circ (\delta \circ \mathrm{lb})^{r-1}(\overline{\nu}, \overline{J})$. Denote the length of the string selected by δ in $(\overline{\nu}_i, \overline{J}_i)^{(a)}$ by $\overline{\ell}_i^{(a)}$. We claim that $\ell_i^{(a)} > \overline{\ell}_i^{(a)}$ for all $1 \leq i \leq r$ and all $i \leq a \leq n$. We will show this by reverse induction on i.

First note that the change in vacancy number from (ν, J) to (ν_i, J_i) is given by

$$p_k^{(a)}(\nu_i) = p_k^{(a)}(\nu) - \sum_{m=i+1}^{r} \chi(\ell_m^{(a-1)} \leq k < \ell_m^{(a)}) + \sum_{m=i+1}^{r} \chi(\ell_m^{(a)} \leq k < \ell_m^{(a+1)}). \quad (2.6.4)$$

The change in vacancy number from (ν, J) to $(\overline{\nu}_i, \overline{J}_i)$ is given by

$$\begin{aligned} p_k^{(a)}(\overline{\nu}_i) = p_k^{(a)}(\nu) &- \sum_{m=1}^{r} \chi(\ell_m^{(a-1)} \leq k < \ell_m^{(a)}) + \sum_{m=1}^{r} \chi(\ell_m^{(a)} \leq k < \ell_m^{(a+1)}) \\ &+ \delta_{a,r}\chi(k < s-1) - \sum_{m=i+1}^{r} \chi(\overline{\ell}_m^{(a-1)} \leq k < \overline{\ell}_m^{(a)}) + \sum_{m=i+1}^{r} \chi(\overline{\ell}_m^{(a)} \leq k < \overline{\ell}_m^{(a+1)}). \end{aligned} \quad (2.6.5)$$

(2.6.4) implies that $\ell_i^{(a)} < \ell_{i-1}^{(a)}$ and the string of length $\ell_j^{(a)} - 1$ remains singular in $(\nu_i, J_i)^{(a)}$ for $i+1 \leq j \leq r$. Recall that $(\nu, J)^{(r)}$ has no singular string of length less than s. So, $\ell_r^{(r)} \geq s$. By construction of the algorithm $\overline{\ell}_r^{(a)} = 1$ for $1 \leq a \leq r-1$. By induction $(\overline{\nu}, \overline{J})^{(r)}$ has no singular string of length less than $s-1$ and hence by (2.6.5) $s - 1 \leq \overline{\ell}_r^{(r)} < \ell_r^{(r)}$ since the string of length $\ell_r^{(r)} - 1 \geq s-1$ is singular. Now by using (2.6.4) the algorithm of δ acting on $(\overline{\nu}, \overline{J})$ gives that $\overline{\ell}_r^{(a)} < \ell_r^{(a)}$ for $a \geq r$. This starts the induction. Suppose $\ell_i^{(a)} > \overline{\ell}_i^{(a)}$ for all $k \leq i \leq r$ and all $i < a \leq n$. Induction hypothesis along with (2.6.5) implies that in $(\overline{\nu}_{k-1}, \overline{J}_{k-1})^{(a)}$ we have $\overline{\ell}_i^{(a)} < \overline{\ell}_{i-1}^{(a)}$ for $i \geq k+1$ and the string of length $\ell_j^{(a)} - 1$ remains singular for $1 \leq j \leq k-1$. Therefore $\overline{\ell}_{k-1}^{(a)} = 1$ for $1 \leq a \leq k-2$ and in $(\overline{\nu}_{k-1}, \overline{J}_{k-1})^{(k-1)}$, the smallest singular string we know is of length $\ell_{k-1}^{(k-1)} - 1$. Hence $\overline{\ell}_{k-1}^{(k-1)} \leq \ell_{k-1}^{(k-1)} - 1 < \ell_{k-1}^{(k-1)}$. Then by using (2.6.5) the algorithm of δ acting on $(\overline{\nu}_k, \overline{J}_k)$ gives that $\overline{\ell}_{k-1}^{(a)} < \ell_{k-1}^{(a)}$ for $a > k-1$. This proves our claim.

But $\ell_i^{(a)} > \overline{\ell}_i^{(a)}$ for all $1 \leq i \leq r$ and all $i \leq a \leq n$ implies $a_i \leq b_i$. So we are done.

2.7 Proof of Theorem 2.4.14

In this section we prove that the crystal operators on paths and rigged configurations commute with the bijection Φ.

The following Lemma is a result of [48, Lemma 3.11] about the convexity of the vacancy numbers.

Lemma 2.7.1. *(**Convexity**) Let* $(\nu, J) \in \mathrm{RC}(L)$.

1. *For all* $i, k \geq 1$ *we have* $-p_{k-1}^{(i)}(\nu) + 2p_k^{(i)}(\nu) - p_{k+1}^{(i)}(\nu) \geq m_k^{(i-1)}(\nu) - 2m_k^{(i)}(\nu) + m_k^{(i+1)}(\nu)$.

2. *Let* $m_k^{(i)}(\nu) = 0$ *for* $a < k < b$. *Then* $p_k^{(i)}(\nu) \geq \min(p_a^{(i)}(\nu), p_b^{(i)}(\nu))$.

3. *Let* $m_k^{(i)}(\nu) = 0$ *for* $a < k < b$. *If* $p_a^{(i)}(\nu) = p_{a+1}^{(i)}(\nu)$ *and* $p_{a+1}^{(i)}(\nu) \leq p_b^{(i)}(\nu)$ *then* $p_{a+1}^{(i)}(\nu) = p_k^{(i)}(\nu)$ *for all* $a \leq k \leq b$.

4. *Let* $m_k^{(i)}(\nu) = 0$ *for* $a < k < b$. *If* $p_b^{(i)}(\nu) = p_{b-1}^{(i)}(\nu)$ *and* $p_{b-1}^{(i)}(\nu) \leq p_a^{(i)}(\nu)$ *then* $p_{b-1}^{(i)}(\nu) = p_k^{(i)}(\nu)$ *for all* $a \leq k \leq b$.

Proof. The proof of (1) is given in [49, Appendix] (see also (2.3.5)), (2) follows from repeated use of (1), and the proof of (3) and (4) follow from (1) and (2). $\square$

Lemma 2.7.2. *Let $B = B^{1,1} \otimes B'$ and let L and L' be the multiplicity arrays of B and B'. For $1 \le i < n$ the following diagrams commute if $\tilde{f}_i$ is always defined:*

$$\begin{array}{ccc} \mathrm{RC}(L) & \xrightarrow{\delta} & \mathrm{RC}(L') \\ {\scriptstyle \tilde{f}_i}\big\downarrow & & \big\downarrow{\scriptstyle \tilde{f}_i} \\ \mathrm{RC}(L) & \xrightarrow[\delta]{} & \mathrm{RC}(L') \end{array} \qquad \begin{array}{ccc} \mathrm{RC}(L) & \xrightarrow{\delta} & \mathrm{RC}(L') \\ {\scriptstyle \tilde{e}_i}\big\downarrow & & \big\downarrow{\scriptstyle \tilde{e}_i} \\ \mathrm{RC}(L) & \xrightarrow[\delta]{} & \mathrm{RC}(L') \end{array} \tag{2.7.1}$$

Proof. We prove (2.7.1) for $\tilde{f}_i$ here; the proof for $\tilde{e}_i$ is similar. Let us introduce some notation. Let $(\nu, J) \in \mathrm{RC}(L)$ and let $\ell^{(a)}$ be the length of the singular string selected by δ in $(\nu, J)^{(a)}$ for $1 \le a < n$. Let $(\overline{\nu}, \overline{J}) = \delta(\nu, J)$ and $(\tilde{\nu}, \tilde{J}) = \tilde{f}_i(\nu, J)$. Let $\tilde{\ell}^{(a)}$ be the length of the singular string selected by δ in $(\tilde{\nu}, \tilde{J})^{(a)}$ for $1 \le a < n$ and ℓ (respectively $\overline{\ell}$) be the length of the string selected by $\tilde{f}_i$ in $(\nu, J)^{(i)}$ (respectively in $(\overline{\nu}, \overline{J})^{(i)}$). A string of length k and label x_k in $(\nu, J)^{(a)}$ is denoted by (k, x_k).

Using the definition of $\tilde{f}_i$ it is easy to see that the diagram (2.7.1) commutes trivially except when $\ell^{(i-1)} - 1 \le \ell \le \ell^{(i)}$. We list the nontrivial cases as follows:

(a) $\ell^{(i-1)} < \infty, \ell^{(i)} = \infty, \ell + 1 \ge \ell^{(i-1)}$.

(b) $\ell^{(i)} < \infty, \ell^{(i-1)} \le \ell + 1 \le \ell^{(i)}$.

(c) $\ell^{(i)} < \infty$ and $\ell^{(i)} = \ell$.

Note that since $\tilde{f}_i$ fixes all the colabels, the singular strings (except the new string of length $\ell + 1$) remain singular under the action of $\tilde{f}_i$. Let (ℓ, x_ℓ) be the string selected by $\tilde{f}_i$ in $(\nu, J)^{(i)}$. The new string of length $\ell+1$ can be singular in $(\tilde{\nu}, \tilde{J})^{(i)}$ only if $p^{(i)}_{\ell+1}(\nu) = x_\ell + 1$. Also note that by the definition of $\tilde{f}_i$ if $m^{(i)}_k(\nu) > 0$ and (k, x_k) is a string in $(\nu, J)^{(i)}$ then

$$\begin{aligned} x_\ell < x_k \le p^{(i)}_k(\nu), &\quad \text{if } k > \ell, \\ x_\ell \le x_k \le p^{(i)}_k(\nu), &\quad \text{if } k < \ell. \end{aligned} \tag{2.7.2}$$

Let us now consider all the nontrivial cases.

Case (a): If the new string of length $\ell+1$ in $(\tilde{\nu}, \tilde{J})^{(i)}$ is nonsingular, then (2.7.1) commutes trivially. Let us consider the case when the new string of length $\ell+1$ in $(\tilde{\nu}, \tilde{J})^{(i)}$ is singular. We have $p_{\ell+1}^{(i)}(\nu) = x_\ell + 1$ and since $\ell^{(i-1)} < \infty, \ell^{(i)} = \infty$ we have $p_j^{(i)}(\overline{\nu}) = p_j^{(i)}(\nu) - 1$ for $j \geq \ell^{(i-1)}$. In particular $p_{\ell+1}^{(i)}(\overline{\nu}) = p_{\ell+1}^{(i)}(\nu) - 1 = x_\ell$. The labels in $(\overline{\nu}, \overline{J})^{(i)}$ are the same as in $(\nu, J)^{(i)}$. Hence $\overline{\ell} = \ell$, but the result is not a valid rigged configuration since $p_{\ell+1}^{(i)}(\overline{\nu}) - 2 < x_\ell - 1$. So, $\tilde{f}_i(\overline{\nu}, \overline{J})$ is undefined, which contradicts the assumptions of Lemma 2.7.2.

Case (b): If the new string of length $\ell+1$ in $(\tilde{\nu}, \tilde{J})^{(i)}$ is singular, we show that the following conditions hold:

(i) $p_{\ell^{(i)}-1}^{(i)}(\overline{\nu}) \leq x_\ell$;

(ii) $m_j^{(i+1)}(\nu) = 0$ for $\ell < j < \ell^{(i)}$.

The above conditions imply that diagram (2.7.1) with $\tilde{f}_i$ commutes for the following reason. Condition (i) implies that $\tilde{f}_i$ acts on the new string of length $\ell^{(i)} - 1$ in $(\overline{\nu}, \overline{J})^{(i)}$. Condition (ii) implies that if $\ell^{(i+1)} < \infty$ then $\tilde{\ell}^{(i+1)} = \ell^{(i+1)}$. Hence $\tilde{\ell}^{(a)} = \ell^{(a)}$ for $a \neq i$ and $\tilde{\ell}^{(i)} = \ell + 1$. This gives $\tilde{f}_i \circ \delta(\nu, J) = \delta \circ \tilde{f}_i(\nu, J)$.

If the new string of length $\ell + 1$ in $(\tilde{\nu}, \tilde{J})^{(i)}$ is nonsingular then the diagram (2.7.1) with $\tilde{f}_i$ commutes if $\tilde{f}_i$ acts on the same string of length ℓ in $(\overline{\nu}, \overline{J})^{(i)}$ as it did on $(\nu, J)^{(i)}$. In this case if $(\ell^{(i-1)} - 1, p_{\ell^{(i)}-1}^{(i)}(\overline{\nu}))$ is the new string created by δ we need to show that $x_\ell < p_{\ell^{(i)}-1}^{(i)}(\overline{\nu})$.

Let us now consider the proof of conditions (i) and (ii) in the case when the new string of length $\ell + 1$ in $(\tilde{\nu}, \tilde{J})^{(i)}$ is singular. Note that $p_{\ell+1}^{(i)}(\nu) = x_\ell + 1 \leq x_j$ for $j > \ell$ and $m_j^{(i)}(\nu) > 0$ by (2.7.2). In particular if $m_{\ell+1}^{(i)}(\nu) > 0$ and $(\ell+1, x_{\ell+1})$ is a string in $(\nu, J)^{(i)}$ then $p_{\ell+1}^{(i)}(\nu) \leq x_{\ell+1} \leq p_{\ell+1}^{(i)}(\nu)$. This implies $p_{\ell+1}^{(i)}(\nu) = x_{\ell+1}$, hence $(\ell + 1, x_{\ell+1})$ is a singular string which is a contradiction if $\ell^{(i-1)} \leq \ell+1 < \ell^{(i)}$. If $\ell+1 = \ell^{(i)}$, it is easy to see that (2.7.1) commutes. Hence we may assume that $\ell+1 < \ell^{(i)}$, so that $m^{(i)}_{\ell+1}(\nu) = 0$. Let $k > \ell$ be smallest so that $m^{(i)}_k(\nu) > 0$. Then by Lemma 2.7.1 (2) we have

$$p^{(i)}_{\ell+1}(\nu) \geq \min(p^{(i)}_\ell(\nu), p^{(i)}_k(\nu)). \tag{2.7.3}$$

If $p^{(i)}_\ell(\nu) > p^{(i)}_k(\nu)$ then by (2.7.3) we get $p^{(i)}_{\ell+1}(\nu) \geq p^{(i)}_k(\nu)$. But

$$\begin{aligned} p^{(i)}_{\ell+1}(\nu) &\leq x_k < p^{(i)}_k(\nu) \quad \text{if } \ell < k < \ell^{(i)}, \\ p^{(i)}_{\ell+1}(\nu) &\leq x_k = p^{(i)}_k(\nu) \quad \text{if } k = \ell^{(i)}. \end{aligned} \tag{2.7.4}$$

Hence $k = \ell^{(i)}$ which implies $p^{(i)}_{\ell+1}(\nu) = p^{(i)}_{\ell^{(i)}}(\nu) < p^{(i)}_\ell(\nu)$ and $m^{(i)}_j(\nu) = 0$ for $\ell < j < \ell^{(i)}$. But now using Lemma 2.7.1 (1) we get the following contradiction:

$$0 > -p^{(i)}_\ell(\nu) + 2p^{(i)}_{\ell+1}(\nu) - p^{(i)}_{\ell+2}(\nu) \geq m^{(i-1)}_{\ell+1}(\nu) + m^{(i+1)}_{\ell+1}(\nu) \geq 0.$$

Hence $p^{(i)}_\ell(\nu) \leq p^{(i)}_k(\nu)$ and by (2.7.3) we $p^{(i)}_{\ell+1}(\nu) \geq p^{(i)}_\ell(\nu)$. Recall that we have

$$\begin{aligned} p^{(i)}_{\ell+1}(\nu) &= x_\ell + 1 \leq p^{(i)}_\ell(\nu), && \text{if } \ell^{(i-1)} < \ell < \ell^{(i)} \text{ or } (\ell, x_\ell) \text{ is nonsingular,} \\ p^{(i)}_{\ell+1}(\nu) &= x_\ell + 1 = p^{(i)}_\ell(\nu) + 1, && \text{if } \ell = \ell^{(i-1)} - 1 \text{ and } (\ell, x_\ell) \text{ is singular.} \end{aligned}$$

This gives us two possible situations:

1. $p^{(i)}_{\ell+1}(\nu) = p^{(i)}_\ell(\nu)$ if $\ell^{(i-1)} < \ell < \ell^{(i)}$ or (ℓ, x_ℓ) is nonsingular,

2. $p^{(i)}_{\ell+1}(\nu) = p^{(i)}_\ell(\nu) + 1$ if $\ell = \ell^{(i-1)} - 1$ and (ℓ, x_ℓ) is singular.

In situation (1) using Lemma 2.7.1 (3) we get $p^{(i)}_{\ell+1}(\nu) = p^{(i)}_j(\nu)$ for $\ell+1 < j \leq k$.

Using (2.7.4) this implies $k = \ell^{(i)}$ and by convexity we get condition (ii). Also this gives $p^{(i)}_{\ell^{(i)}-1}(\nu) = p^{(i)}_{\ell+1}(\nu) = x_\ell + 1$ and hence $p^{(i)}_{\ell^{(i)}-1}(\overline{\nu}) = x_\ell$, which proves condition (i).

In situation (2) we have

$$\begin{aligned} p^{(i)}_{\ell+1}(\overline{\nu}) &= p^{(i)}_{\ell+1}(\nu) - 1 && \text{since } \ell^{(i-1)} = \ell+1 < \ell^{(i)} \\ &= p^{(i)}_{\ell}(\nu) && \text{since } p^{(i)}_{\ell+1}(\nu) = p^{(i)}_{\ell}(\nu) + 1 \\ &= p^{(i)}_{\ell}(\overline{\nu}) && \text{since } \ell < \ell^{(i-1)}. \end{aligned}$$

Also note that $p^{(i)}_k(\overline{\nu}) = p^{(i)}_k(\nu) - 1$ if $k < \ell^{(i)}$ and $p^{(i)}_k(\overline{\nu}) = p^{(i)}_k(\nu) + 1$ if $k = \ell^{(i)}$. Now using (2.7.4) and (2.7.2) we get $p^{(i)}_{\ell+1}(\overline{\nu}) < p^{(i)}_k(\overline{\nu})$. Since $m^{(i)}_\ell(\overline{\nu}) = m^{(i)}_\ell(\nu) > 0$ using Lemma 2.7.1 (3) we get $p^{(i)}_{\ell+1}(\overline{\nu}) = p^{(i)}_\ell(\overline{\nu}) = p^{(i)}_j(\overline{\nu})$ for $\ell+2 \le j \le k$. This contradicts that $p^{(i)}_{\ell+1}(\overline{\nu}) < p^{(i)}_k(\overline{\nu})$, hence situation (2) cannot occur.

Now let us consider the case when the new string of length $\ell+1$ in $(\tilde{\nu}, \tilde{J})^{(i)}$ is nonsingular. If $\ell+1 = \ell^{(i)}$ the commutation of (2.7.1) is again fairly easy to see. Hence assume that $\ell+1 < \ell^{(i)}$. Then we have $p^{(i)}_{\ell^{(i)}-1}(\overline{\nu}) = p^{(i)}_{\ell^{(i)}-1}(\nu) - 1$. If $m^{(i)}_{\ell^{(i)}-1}(\nu) > 0$ and $(\ell^{(i)}-1, x_{\ell^{(i)}-1})$ is a string in $(\nu, J)^{(i)}$ then $x_{\ell^{(i)}-1} < p^{(i)}_{\ell^{(i)}-1}(\nu)$ since $\ell^{(i-1)} \le \ell+1 \le \ell^{(i)} - 1 < \ell^{(i)}$. Hence by (2.7.2) we have $x_\ell < x_{\ell^{(i)}-1} < p^{(i)}_{\ell^{(i)}-1}(\nu)$ which implies $x_\ell < p^{(i)}_{\ell^{(i)}-1}(\overline{\nu})$ and we are done.

If $m^{(i)}_{\ell^{(i)}-1}(\nu) = 0$ let $\ell \le j < \ell^{(i)} - 1$ be smallest such that $m^{(i)}_j(\nu) > 0$. By Lemma 2.7.1 (2) we get

$$p^{(i)}_{\ell^{(i)}-1}(\nu) \ge \min(p^{(i)}_j(\nu), p^{(i)}_{\ell^{(i)}}(\nu)). \tag{2.7.5}$$

Note that if $\ell < j < \ell^{(i)}$ then the string (j, x_j) in $(\nu, J)^{(i)}$ is nonsingular and therefore $p^{(i)}_j(\nu) > x_j > x_\ell$ by (2.7.2). Also if $(\ell^{(i)}, x_{\ell^{(i)}})$ is the singular string $p^{(i)}_{\ell^{(i)}}(\nu) = x_{\ell^{(i)}} > x_\ell$ by (2.7.2). So $\min(p^{(i)}_j(\nu), p^{(i)}_{\ell^{(i)}}(\nu)) \ge x_\ell + 1$. Hence by (2.7.5) $p^{(i)}_{\ell^{(i)}-1}(\nu) \ge x_\ell + 1$.

Suppose $p^{(i)}_{\ell^{(i)}-1}(\nu) = x_\ell + 1$. Since $p^{(i)}_j(\nu) > x_\ell + 1$ we get by (2.7.5) $x_\ell + 1 = p^{(i)}_{\ell^{(i)}-1}(\nu) \geq p^{(i)}_{\ell^{(i)}}(\nu) \geq x_\ell + 1$ which implies $p^{(i)}_{\ell^{(i)}-1}(\nu) = p^{(i)}_{\ell^{(i)}}(\nu)$. Since $p^{(i)}_{\ell^{(i)}-1}(\nu) = x_\ell + 1 \leq p^{(i)}_a(\nu)$ for all $j < a < \ell^{(i)}$ by Lemma 2.7.1 (4) we get $p^{(i)}_j(\nu) = x_\ell + 1$ which is a contradiction. Hence $p^{(i)}_{\ell^{(i)}-1}(\nu) > x_\ell + 1$ and we get $x_\ell < p^{(i)}_{\ell^{(i)}-1}(\overline{\nu})$ as desired.

Let us consider the case $j = \ell$. If the string (ℓ, x_ℓ) is nonsingular by similar argument as in the previous case we have that $p^{(i)}_{\ell^{(i)}-1}(\nu) \geq x_\ell + 1$. Suppose $p^{(i)}_{\ell^{(i)}-1}(\nu) = x_\ell + 1$. By (2.7.5) if $p^{(i)}_{\ell^{(i)}-1}(\nu) \geq p^{(i)}_{\ell^{(i)}}(\nu) \geq x_\ell + 1$ we get as before that $p^{(i)}_{\ell^{(i)}-1}(\nu) = p^{(i)}_{\ell^{(i)}}(\nu)$. Using Lemma 2.7.1 (4) we can show as before that $p^{(i)}_{\ell+1}(\nu) = x_\ell + 1$ which is a contradiction since the string of length $\ell + 1$ is not singular in $(\tilde{\nu}, \tilde{J})^{(i)}$. By (2.7.5) if $p^{(i)}_{\ell^{(i)}-1}(\nu) \geq p^{(i)}_\ell(\nu) \geq x_\ell + 1$ we get $p^{(i)}_{\ell^{(i)}-1}(\nu) = p^{(i)}_\ell(\nu) = x_\ell + 1$. This implies that $p^{(i)}_{\ell^{(i)}-1}(\nu) \leq p^{(i)}_a(\nu)$ for all $a > \ell$. If we use this in Lemma 2.7.1 (1) for $k = \ell^{(i)} - 1$ we get $p^{(i)}_{\ell^{(i)}-1}(\nu) = p^{(i)}_{\ell^{(i)}}(\nu)$ and then using Lemma 2.7.1 (4) we get $p^{(i)}_{\ell+1}(\nu) = x_\ell + 1$ which is a contradiction as before.

Hence the only case left to be considered is when $j = \ell = \ell^{(i-1)} - 1$ and the string (ℓ, x_ℓ) is singular in $(\nu, J)^{(i)}$. Here $\min(p^{(i)}_\ell(\nu), p^{(i)}_{\ell^{(i)}}(\nu)) = p^{(i)}_\ell(\nu)$ and therefore by (2.7.5) $p^{(i)}_{\ell^{(i)}-1}(\nu) \geq x_\ell$. Suppose $p^{(i)}_{\ell^{(i)}-1}(\nu) = x_\ell$. Since $p^{(i)}_{\ell^{(i)}}(\nu) \geq x_\ell + 1$ we have $p^{(i)}_{\ell^{(i)}-1}(\nu) < p^{(i)}_{\ell^{(i)}}(\nu)$. Also, $p^{(i)}_{\ell^{(i)}-1}(\nu) \geq \min(p^{(i)}_\ell(\nu), p^{(i)}_{\ell^{(i)}}(\nu)) = p^{(i)}_\ell(\nu) = x_\ell = p^{(i)}_{\ell^{(i)}-1}(\nu)$. Using this in Lemma 2.7.1 (1) for $k = \ell^{(i)} - 1$ we get the following contradiction:

$$0 > -p^{(i)}_{\ell^{(i)}-2}(\nu) + 2p^{(i)}_{\ell^{(i)}-1}(\nu) - p^{(i)}_{\ell^{(i)}}(\nu) \geq m^{(i-1)}_{\ell^{(i)}-1}(\nu) + m^{(i+1)}_{\ell^{(i)}-1}(\nu) \geq 0. \tag{2.7.6}$$

Hence $p^{(i)}_{\ell^{(i)}-1}(\nu) > x_\ell$. Suppose $p^{(i)}_{\ell^{(i)}-1}(\nu) = x_\ell + 1$. Here $p^{(i)}_{\ell^{(i)}-1}(\nu) \leq p^{(i)}_{\ell^{(i)}}(\nu)$. If $p^{(i)}_{\ell^{(i)}-1}(\nu) = p^{(i)}_{\ell^{(i)}}(\nu)$ as before we can show that $p^{(i)}_{\ell+1}(\nu) = x_\ell + 1$, which is a contradiction. Suppose $p^{(i)}_{\ell^{(i)}-1}(\nu) < p^{(i)}_{\ell^{(i)}}(\nu)$ then $p^{(i)}_{\ell^{(i)}-2}(\nu) \geq \min(p^{(i)}_\ell(\nu), p^{(i)}_{\ell^{(i)}}(\nu)) = p^{(i)}_\ell(\nu) = x_\ell = p^{(i)}_{\ell^{(i)}-1}(\nu) - 1$. If $p^{(i)}_{\ell^{(i)}-2}(\nu) > p^{(i)}_{\ell^{(i)}-1}(\nu)$ we again get the contradiction (2.7.6). If $p^{(i)}_{\ell^{(i)}-2}(\nu) = p^{(i)}_{\ell^{(i)}-1}(\nu)$ using Lemma 2.7.1 (1) for $k = \ell^{(i)} - 1$ we get $p^{(i)}_{\ell^{(i)}}(\nu) =$

$p^{(i)}_{\ell^{(i)}-1}(\nu)$ which is a contradiction to our assumption. Hence $p^{(i)}_{\ell^{(i)}-1}(\nu) > x_\ell + 1$ giving $x_\ell < p^{(i)}_{\ell^{(i)}-1}(\overline{\nu})$.

Case (c): Note that since $\tilde{f}_i$ acts on the string (ℓ, x_ℓ) in $(\nu, J)^{(i)}$ we have

$$p^{(i)}_{\ell+1}(\nu) \geq x_\ell + 1 = p^{(i)}_\ell(\nu) + 1. \tag{2.7.7}$$

If $\tilde{f}_i$ and δ select the same string of length ℓ in $(\nu, J)^{(i)}$ then $m^{(i)}_\ell(\nu) = 1$. But if $\tilde{f}_i$ and δ select different strings of length ℓ in $(\nu, J)^{(i)}$ then $m^{(i)}_\ell(\nu) > 1$. We will consider each of these two cases separately.

If $m^{(i)}_\ell(\nu) > 1$ let (ℓ, x_ℓ) be the string selected by $\tilde{f}_i$ and $(\ell, p^{(i)}_\ell(\nu))$ be the string selected by δ in $(\nu, J)^{(i)}$. Note that $x_\ell \leq p^{(i)}_\ell(\nu)$. To prove that the diagram (2.7.1) with $\tilde{f}_i$ commutes it is enough to show that $\tilde{f}_i$ acts on the same string (ℓ, x_ℓ) in $(\overline{\nu}, \overline{J})^{(i)}$ as it did in $(\nu, J)^{(i)}$. Hence it suffices to show that the new label in $(\overline{\nu}, \overline{J})^{(i)}$ satisfies $p^{(i)}_{\ell-1}(\overline{\nu}) \geq x_\ell$. Note that

$$\begin{aligned} p^{(i)}_{\ell-1}(\overline{\nu}) &= p^{(i)}_{\ell-1}(\nu) - 1 && \text{if } \ell > \ell^{(i-1)},\\ p^{(i)}_{\ell-1}(\overline{\nu}) &= p^{(i)}_{\ell-1}(\nu) && \text{if } \ell = \ell^{(i-1)}. \end{aligned}$$

If $m^{(i)}_{\ell-1}(\nu) > 0$ let $(\ell-1, x_{\ell-1})$ be a string in $(\nu, J)^{(i)}$. Then

$$\begin{aligned} x_\ell \leq x_{\ell-1} &< p^{(i)}_{\ell-1}(\nu) && \text{if } \ell > \ell^{(i-1)},\\ x_\ell \leq x_{\ell-1} &\leq p^{(i)}_{\ell-1}(\nu) && \text{if } \ell = \ell^{(i-1)}, \end{aligned}$$

which implies $p^{(i)}_{\ell-1}(\overline{\nu}) \geq x_\ell$.

If $m^{(i)}_{\ell-1}(\nu) = 0$ let $j < \ell - 1$ be largest such that $m^{(i)}_j(\nu) > 0$ and (j, x_j) be a string in $(\nu, J)^{(i)}$. Then by Lemma 2.7.1 (2) we have $p^{(i)}_{\ell-1}(\nu) \geq \min(p^{(i)}_j(\nu), p^{(i)}_\ell(\nu))$.

If $p_j^{(i)}(\nu) \leq p_\ell^{(i)}(\nu)$ then using (2.7.2) we have

$$p_{\ell-1}^{(i)}(\nu) \geq p_j^{(i)}(\nu) > x_j \geq x_\ell \qquad \text{if } \ell^{(i-1)} \leq j < \ell - 1,$$
$$p_{\ell-1}^{(i)}(\nu) \geq p_j^{(i)}(\nu) \geq x_j \geq x_\ell \qquad \text{if } j < \ell^{(i-1)}.$$

Hence $p_{\ell-1}^{(i)}(\overline{\nu}) \geq x_\ell$ unless

$$p_{\ell-1}^{(i)}(\nu) = p_j^{(i)}(\nu) = x_j = x_\ell \leq p_\ell^{(i)}(\nu) \qquad \text{with } j < \ell^{(i-1)} \leq \ell - 1. \tag{2.7.8}$$

But if this happens by Lemma 2.7.1 we get $p_{\ell^{(i-1)}}^{(i)}(\nu) = p_{\ell^{(i-1)}-1}^{(i)}(\nu) \leq p_{\ell^{(i-1)}+1}^{(i)}(\nu)$. Note that here $m_{\ell^{(i-1)}}^{(i)}(\nu) = 0$ and $m_{\ell^{(i-1)}}^{(i-1)}(\nu) > 0$. Using all these we get the following contradiction:

$$0 \geq -p_{\ell^{(i-1)}-1}^{(i)}(\nu) + 2p_{\ell^{(i-1)}}^{(i)}(\nu) - p_{\ell^{(i-1)}+1}^{(i)}(\nu) \geq m_{\ell^{(i-1)}}^{(i-1)}(\nu) + m_{\ell^{(i-1)}}^{(i-1)}(\nu) \geq 1.$$

This shows that (2.7.8) can not happen.

If $p_j^{(i)}(\nu) > p_\ell^{(i)}(\nu)$ then $p_{\ell-1}^{(i)}(\nu) \geq \min(p_j^{(i)}(\nu), p_\ell^{(i)}(\nu)) = p_\ell^{(i)}(\nu) \geq x_\ell$. Again $p_{\ell-1}^{(i)}(\overline{\nu}) \geq x_\ell$ unless

$$p_{\ell-1}^{(i)}(\nu) = p_\ell^{(i)}(\nu) = x_\ell \qquad \text{with } \ell^{(i-1)} \leq \ell - 1. \tag{2.7.9}$$

But this implies by Lemma 2.7.1 that $p_j^{(i)}(\nu) = p_\ell^{(i)}(\nu) = x_\ell$ which is a contradiction to our assumption. Hence (2.7.9) does not occur. This completes the proof when $m_\ell^{(i)}(\nu) > 1$.

If $m_\ell^{(i)}(\nu) = 1$ we claim that

(i) $p_{\ell+1}^{(i)}(\nu) = x_\ell + 1 = p_\ell^{(i)}(\nu) + 1$,

(ii) $p^{(i)}_{\ell-1}(\overline{\nu}) = x_\ell$,

(iii) If $\ell^{(i+1)} < \infty$ then $\ell + 1 \leq \ell^{(i+1)}$.

It is easy to see that diagram (2.7.1) with $\tilde{f}_i$ commutes if our claim is true. Condition (i) implies that the new string $(\ell+1, x_\ell - 1)$ in $(\tilde{\nu}, \tilde{J})^{(i)}$ is singular and $\tilde{\ell}^{(i)} = \ell+1$. Condition (iii) implies that $\tilde{\ell}^{(i+1)} = \ell^{(i+1)}$. On the other hand condition (ii) implies $\overline{\ell} = \ell - 1$, the new string created by δ in $(\overline{\nu}, \overline{J})^{(i)}$.

Let us prove our claims now. Using Lemma 2.7.1 (1) we have

$$(p^{(i)}_\ell(\nu) - p^{(i)}_{\ell-1}(\nu)) + (p^{(i)}_\ell(\nu) - p^{(i)}_{\ell+1}(\nu)) \geq m^{(i-1)}_\ell(\nu) - 2 + m^{(i+1)}_\ell(\nu).$$

which can be rewritten as

$$(p^{(i)}_\ell(\nu) + 1 - p^{(i)}_{\ell-1}(\nu)) + (p^{(i)}_\ell(\nu) + 1 - p^{(i)}_{\ell+1}(\nu)) \geq m^{(i-1)}_\ell(\nu) + m^{(i+1)}_\ell(\nu) \geq 0. \quad (2.7.10)$$

Suppose $\ell^{(i-1)} < \ell = \ell^{(i)}$. If $m^{(i)}_{\ell-1}(\nu) > 0$ then the string $(\ell-1, x_{\ell-1})$ is nonsingular and hence by (2.7.2) $p^{(i)}_\ell(\nu) = x_\ell \leq x_{\ell-1} < p^{(i)}_{\ell-1}(\nu)$. If $m^{(i)}_{\ell-1}(\nu) = 0$ let $j < \ell - 1$ be largest such that $m^{(i)}_j(\nu) > 0$. Note that $p^{(i)}_j(\nu) \geq x_\ell = p^{(i)}_\ell(\nu)$, so by Lemma 2.7.1 (2) we have $p^{(i)}_{\ell-1}(\nu) \geq \min(p^{(i)}_j(\nu), p^{(i)}_\ell(\nu)) = p^{(i)}_\ell(\nu)$. Hence $p^{(i)}_{\ell-1}(\nu) > p^{(i)}_\ell(\nu)$ unless

$$p^{(i)}_{\ell-1}(\nu) = p^{(i)}_\ell(\nu) = p^{(i)}_j(\nu) = x_\ell \text{with } j < \ell^{(i-1)} < \ell. \quad (2.7.11)$$

But if this happens by Lemma 2.7.1 we get $p^{(i)}_{\ell^{(i-1)}}(\nu) = p^{(i)}_{\ell^{(i-1)}-1}(\nu) = p^{(i)}_{\ell^{(i-1)}+1}(\nu)$ which gives us the following contradiction since $m^{(i-1)}_{\ell^{(i-1)}}(\nu) > 0$:

$$0 \geq -p^{(i)}_{\ell^{(i-1)}-1}(\nu) + 2p^{(i)}_{\ell^{(i-1)}}(\nu) - p^{(i)}_{\ell^{(i-1)}+1}(\nu) \geq m^{(i-1)}_{\ell^{(i-1)}}(\nu) + m^{(i-1)}_{\ell^{(i-1)}}(\nu) \geq 1.$$

Hence (2.7.11) cannot happen and we have $p_{\ell-1}^{(i)}(\nu) > p_\ell^{(i)}(\nu)$. Now using this and (2.7.7) in (2.7.10) we get

$$0 \geq (p_\ell^{(i)}(\nu) + 1 - p_{\ell-1}^{(i)}(\nu)) + (p_\ell^{(i)}(\nu) + 1 - p_{\ell+1}^{(i)}(\nu)) \geq m_\ell^{(i-1)}(\nu) + m_\ell^{(i+1)}(\nu) \geq 0,$$

which implies $p_\ell^{(i)}(\nu) = p_{\ell-1}^{(i)}(\nu) - 1$, $p_{\ell+1}^{(i)}(\nu) = p_\ell^{(i)}(\nu) + 1$, $m_\ell^{(i-1)}(\nu) = 0$ and $m_\ell^{(i+1)}(\nu) = 0$. This proves (i) and (iii). Also $p_\ell^{(i)}(\nu) = p_{\ell-1}^{(i)}(\nu) - 1$ implies $p_{\ell-1}^{(i)}(\overline{\nu}) = p_{\ell-1}^{(i)}(\nu) - 1 = p_\ell^{(i)}(\nu) = x_\ell$. This proves (ii).

Suppose $\ell^{(i-1)} = \ell = \ell^{(i)}$. This means $m_\ell^{(i-1)}(\nu) \geq 1$ and as before if $m_{\ell-1}^{(i)}(\nu) > 0$ we have $p_\ell^{(i)}(\nu) = x_\ell \leq x_{\ell-1} \leq p_{\ell-1}^{(i)}(\nu)$. If $m_{\ell-1}^{(i)}(\nu) = 0$ again as in the previous case we have $p_{\ell-1}^{(i)}(\nu) \geq \min(p_j^{(i)}(\nu), p_\ell^{(i)}(\nu)) = p_\ell^{(i)}(\nu)$. Using this and (2.7.7) in (2.7.10) we get $p_\ell^{(i)}(\nu) = p_{\ell-1}^{(i)}(\nu)$, $p_{\ell+1}^{(i)}(\nu) = p_\ell^{(i)}(\nu) + 1$, $m_\ell^{(i-1)}(\nu) = 1$ and $m_\ell^{(i+1)}(\nu) = 0$. Note that since $\ell^{(i-1)} = \ell$, $p_{\ell-1}^{(i)}(\overline{\nu}) = p_{\ell-1}^{(i)}(\nu) = p_\ell^{(i)}(\nu) = x_\ell$. So we proved (i), (ii) and (iii). $\square$

Lemma 2.7.3. *Let $B = B^{r,1} \otimes B'$, $r \geq 2$ and let L be the multiplicity array of B. For $1 \leq i < n$ the following diagrams commute:*

$$\begin{array}{ccccccc} \mathrm{RC}(L) & \xrightarrow{\mathrm{lb}_{rc}} & \mathrm{RC}(\mathrm{lb}(L)) & \qquad & \mathrm{RC}(L) & \xrightarrow{\mathrm{lb}_{rc}} & \mathrm{RC}(\mathrm{lb}(L)) \\ {\scriptstyle \tilde{f}_i}\downarrow & & \downarrow{\scriptstyle \tilde{f}_i} & & {\scriptstyle \tilde{e}_i}\downarrow & & \downarrow{\scriptstyle \tilde{e}_i} \\ \mathrm{RC}(L) & \xrightarrow[\mathrm{lb}_{rc}]{} & \mathrm{RC}(\mathrm{lb}(L)) & & \mathrm{RC}(L) & \xrightarrow[\mathrm{lb}_{rc}]{} & \mathrm{RC}(\mathrm{lb}(L)) \end{array} \tag{2.7.12}$$

Proof. Note that if $i > r-1$ then the proof of (2.7.12) is trivial. Suppose $1 \leq i \leq r-1$. The proof for $\tilde{e}_i$ is very similar to the proof for $\tilde{f}_i$, so here we only prove (2.7.12) for $\tilde{f}_i$. Let $(\nu, J) \in \mathrm{RC}(L)$. Let (ℓ, x_ℓ) be the string selected by $\tilde{f}_i$ in $(\nu, J)^{(i)}$. Let $(\overline{\nu}, \overline{J}) = \mathrm{lb}_{rc}(\nu, J)$. By definition of lb_{rc} we get $(\overline{\nu}, \overline{J})^{(k)}$ by adding a singular string of length one to $(\nu, J)^{(k)}$ for $1 \leq k \leq r-1$. Hence to show that the diagram (2.7.12) commutes it suffices to show that the label for the new singular string of length one in $(\overline{\nu}, \overline{J})^{(i)}$ satisfies $p_1^{(i)}(\overline{\nu}) \geq x_\ell$.

Note that $p_1^{(i)}(\overline{\nu}) = p_1^{(i)}(\nu)$ for all $1 \leq i \leq r-1$.

If $m_1^{(i)}(\nu) > 0$ then $x_1^{(i)} \geq x_\ell$ by (2.7.2). So, $p_1^{(i)}(\overline{\nu}) = p_1^{(i)}(\nu) \geq x_1^{(i)} \geq x_\ell$. If $m_1^{(i)}(\nu) = 0$ let j be smallest such that $m_j^{(i)}(\nu) > 0$ and (j, x_j) be a string in $(\nu, J)^{(i)}$. By Lemma 2.7.1 (2) we get $p_1^{(i)}(\nu) \geq \min(p_0^{(i)}(\nu), p_j^{(i)}(\nu))$. Recall that $p_0^{(i)}(\nu) = 0$ and $x_\ell \leq 0$ by the definition of $\tilde{f}_i$. So, if $p_j^{(i)}(\nu) \geq 0$ then $p_j^{(i)}(\overline{\nu}) = p_1^{(i)}(\nu) \geq 0 \geq x_\ell$. If $p_j^{(i)}(\nu) < 0$ then $p_1^{(i)}(\nu) \geq p_j^{(i)}(\nu)$. But $p_j^{(i)}(\nu) \geq x_j \geq x_\ell$. Hence $p_1^{(i)}(\overline{\nu}) = p_1^{(i)}(\nu) \geq p_j^{(i)}(\nu) \geq x_\ell$ and we are done. $\square$

Lemma 2.7.4. *Let $B = B^{r,s} \otimes B'$, $r \geq 1, s \geq 2$ and let L be the multiplicity array of B. For $1 \leq i < n$ the following diagrams commute:*

$$\begin{tikzcd} \mathrm{RC}(L) \arrow[r, "\mathrm{ls}_{rc}"] \arrow[d, "\tilde{f}_i"'] & \mathrm{RC}(\mathrm{ls}(L)) \arrow[d, "\tilde{f}_i"] \\ \mathrm{RC}(L) \arrow[r, "\mathrm{ls}_{rc}"'] & \mathrm{RC}(\mathrm{ls}(L)) \end{tikzcd} \qquad \begin{tikzcd} \mathrm{RC}(L) \arrow[r, "\mathrm{ls}_{rc}"] \arrow[d, "\tilde{e}_i"'] & \mathrm{RC}(\mathrm{ls}(L)) \arrow[d, "\tilde{e}_i"] \\ \mathrm{RC}(L) \arrow[r, "\mathrm{ls}_{rc}"'] & \mathrm{RC}(\mathrm{ls}(L)) \end{tikzcd} \tag{2.7.13}$$

Proof. Let $(\nu, J) \in \mathrm{RC}(L)$. By definition ls_{rc} only changes the vacancy numbers in $(\nu, J)^{(r)}$. Hence the proof of this lemma is trivial. $\square$

Now we will prove Theorem 2.4.14.

Proof of Theorem 2.4.14. To prove this theorem we will use a diagram of the form

$$\begin{tikzcd} \bullet \arrow[rrr, "F"] \arrow[ddd, "G"'] \arrow[dr] & & & \bullet \arrow[ddd, "H"] \arrow[dl] \\ & \bullet \arrow[r] \arrow[d] & \bullet \arrow[d] & \\ & \bullet \arrow[r] & \bullet & \\ \bullet \arrow[rrr, "K"'] \arrow[ur, "g"] & & & \bullet \arrow[ul] \end{tikzcd}$$

We view this diagram as a cube with front face given by the large square. By [48, Lemma 5.3] if the squares given by all the faces of the cube except the front commute and the map g is injective then the front face also commutes.

We will prove Theorem 2.4.14 by using induction on B as we did in the proof of the bijection of Proposition 2.4.8. First let $B = B^{1,1} \otimes B'$. We prove Theorem 2.4.14 for $\tilde{f}_i$ by using Lemma 2.7.2 and the following diagram when f_i and $\tilde{f}_i$ are defined:

$$\begin{tikzcd}
\mathcal{P}(B) \arrow[rrr, "\Phi"] \arrow[dr, "\mathrm{lh}"] \arrow[ddd, "f_i"'] & & & \mathrm{RC}(L) \arrow[dl, "\delta"'] \arrow[ddd, "\tilde{f}_i"] \\
& \mathcal{P}(B') \arrow[r, "\Phi"] \arrow[d, "f_i"'] & \mathrm{RC}(L') \arrow[d, "\tilde{f}_i"] & \\
& \mathcal{P}(B') \arrow[r, "\Phi"] & \mathrm{RC}(L') & \\
\mathcal{P}(B) \arrow[rrr, "\Phi"'] \arrow[ur, "\mathrm{lh}"] & & & \mathrm{RC}(L) \arrow[ul, "\delta"']
\end{tikzcd}$$

Note the top and the bottom faces commute by Definition 2.4.7 (1). The right face commutes by Lemma 2.7.2. The left face commutes by definition of f_i on the paths and we know lh is injective. By induction hypothesis the back face commutes. Hence the front face must commute.

Let us now prove Theorem 2.4.14 when not all f_i (resp. $\tilde{f}_i$) in the above diagram are defined. Let $(\nu, J) \in \mathrm{RC}(L)$, $(\overline{\nu}, \overline{J}) = \delta(\nu, J)$, $b = \Phi^{-1}(\nu, J)$ and $b' = \Phi^{-1}(\overline{\nu}, \overline{J})$. We need to show the following cases:

1. $f_i(b)$ is defined and $f_i(b')$ is undefined if and only if $\tilde{f}_i(\nu, J)$ is defined and $\tilde{f}_i(\overline{\nu}, \overline{J})$ is undefined. In addition $\Phi(f_i(b)) = \tilde{f}_i(\nu, J)$.

2. $f_i(b)$ is undefined and $f_i(b')$ is defined if and only if $\tilde{f}_i(\nu, J)$ is undefined and $\tilde{f}_i(\overline{\nu}, \overline{J})$

is defined.

3. $f_i(b)$ and $f_i(b')$ are both undefined if and only if $\tilde{f}_i(\nu, J)$ and $\tilde{f}_i(\overline{\nu}, \overline{J})$ are both undefined.

For Case (1) suppose that $\tilde{f}_i(\nu, J) = (\tilde{\nu}, \tilde{J})$ is defined, but $\tilde{f}_i(\overline{\nu}, \overline{J})$ is undefined. Then we are in the situation described in Case (a) of Lemma 2.7.2. That is $\ell^{(i-1)} < \infty$, $\ell^{(i)} = \infty$, $\ell + 1 \geq \ell^{(i-1)}$ and the new string of length $\ell + 1$ is singular in $(\tilde{\nu}, \tilde{J})^{(i)}$. In this situation note that $m^{(i)}_{\ell+1}(\overline{\nu}) = 0$, else $p^{(i)}_{\ell+1}(\overline{\nu}) \geq x_{\ell+1} > x_\ell$ by (2.7.2), which is a contradiction to $p^{(i)}_{\ell+1}(\overline{\nu}) = x_\ell$ as discussed in Case (a) of Lemma 2.7.2. Suppose $j > \ell$ be smallest such that $m^{(i)}_j(\overline{\nu}) > 0$. Then

$$p^{(i)}_j(\overline{\nu}) \geq x_j > x_\ell = p^{(i)}_{\ell+1}(\overline{\nu}). \tag{2.7.14}$$

By Lemma 2.7.1 (2), $p^{(i)}_{\ell+1}(\overline{\nu}) \geq \min(p^{(i)}_\ell(\overline{\nu}), p^{(i)}_j(\overline{\nu}))$. By (2.7.14) this implies $p^{(i)}_{\ell+1}(\overline{\nu}) \geq p^{(i)}_\ell(\overline{\nu})$. But $x_\ell = p^{(i)}_{\ell+1}(\overline{\nu}) \geq p^{(i)}_\ell(\overline{\nu}) \geq x_\ell$, hence we get $p^{(i)}_{\ell+1}(\overline{\nu}) = p^{(i)}_\ell(\overline{\nu})$. Again by Lemma 2.7.1 (3) since $m^{(i)}_k(\overline{\nu}) = 0$ for $\ell < k < j$ we get $p^{(i)}_{\ell+1}(\overline{\nu}) = p^{(i)}_j(\overline{\nu})$ which contradicts (2.7.14). Hence $m^{(i)}_j(\overline{\nu}) = 0$ for $j > \ell$. Also by Lemma 2.7.1 (1) $p^{(i)}_{\ell+1}(\overline{\nu}) = p^{(i)}_\ell(\overline{\nu})$ with $m^{(i)}_j(\overline{\nu}) = 0$ for $j > \ell$ implies that $m^{(i+1)}_j(\overline{\nu}) = 0$ for $j > \ell$. Since $\overline{\nu}^{(i+1)}$ and $\tilde{\nu}^{(i+1)}$ have the same shape we get $m^{(i+1)}_j(\tilde{\nu}) = 0$ for $j > \ell$. Hence $\tilde{\ell}^{(a)} = \ell^{(a)}$ for $1 \leq a \leq i-1$, $\tilde{\ell}^{(i)} = \ell + 1$ and $\tilde{\ell}^{(i+1)} = \infty$. Therefore we proved that if $\Phi^{-1}(\overline{\nu}, \overline{J}) = b' \in B'$ then $\Phi^{-1}(\nu, J) = i \otimes b'$ and $\Phi^{-1}(\tilde{\nu}, \tilde{J}) = i + 1 \otimes b'$. But $\tilde{f}_i(\overline{\nu}, \overline{J}) = 0$ implies $f_i(\Phi^{-1}(\overline{\nu}, \overline{J})) = 0$ since by induction we have that $\Phi^{-1} \circ \tilde{f}_i = f_i \circ \Phi^{-1}$ for B'. Hence $f_i(\Phi^{-1}(\nu, J)) = \Phi^{-1}(\tilde{\nu}, \tilde{J}) = \Phi^{-1}(\tilde{f}_i(\nu, J))$, so that indeed $f_i(b)$ is defined, $f_i(b')$ and $\Phi(f_i(b)) = \tilde{f}_i(\nu, J)$.

Now suppose that $f_i(b)$ is defined and $f_i(b')$ is undefined. This implies that $b = i \otimes b'$. By induction $\tilde{f}_i(\overline{\nu}, \overline{J})$ is undefined so that by Lemma 2.4.13 we have $\overline{p} = \overline{s}$ where $\overline{p} = p^{(i)}_j(\overline{\nu})$ for large j and $\overline{s}$ is the smallest label occurring in $(\overline{\nu}, \overline{J})^{(i)}$. Since b is obtained from

b' by adding i it follows that the vacancy numbers change as $p := p_j^{(i)}(\nu) = \overline{p}+1$ for large j under δ^{-1} and the new smallest label occurring in $(\nu, J)^{(i)}$ is $s = \overline{s}$. Hence $\widetilde{\varphi}_i(\nu, J) = p - s = 1$, so that $\tilde{f}_i(\nu, J)$ is defined. It remains to prove that $\Phi(f_i(b)) = \tilde{f}_i(\nu, J)$. Note that $f_i(b) = i+1 \otimes b'$. Let ℓ be the length of the largest part in $(\overline{\nu}, \overline{J})^{(i)}$. Suppose that $\overline{\nu}^{(i-1)}$ or $\overline{\nu}^{(i+1)}$ has a part strictly bigger than ℓ. In this case $p_\ell^{(i)}(\overline{\nu}) < \overline{p} = \overline{s}$ contradicting the fact that $\overline{s} \le p_\ell^{(i)}(\overline{\nu})$ is the smallest label occurring in $(\overline{\nu}, \overline{J})^{(i)}$. Hence both $\overline{\nu}^{(i-1)}$ and $\overline{\nu}^{(i+1)}$ have only parts of length less or equal to ℓ. Also by Lemma 2.3.6 we have $p_\ell^{(i)}(\overline{\nu}) = \overline{s} = s$ which shows that both δ^{-1} adding $i+1$ and $\tilde{f}_i$ pick the string of length ℓ in $(\overline{\nu}, \overline{J})^{(i)}$. Hence $\Phi(f_i(b)) = \tilde{f}_i(\nu, J)$.

Let us now consider Case (2). Suppose that $\tilde{f}_i(\nu, J)$ is undefined and $\tilde{f}_i(\overline{\nu}, \overline{J})$ is defined. Again by Lemma 2.4.13 we have that $p = s$ where $p = p_j^{(i)}(\nu)$ for large j and s is the smallest label in $(\nu, J)^{(i)}$. If $\mathrm{rk}(\nu, J) < i+1$, then s is still the smallest label in $(\overline{\nu}, \overline{J})$ and by the change in vacancy numbers $\overline{p} \le p$. Hence by Lemma 2.4.13 $\widetilde{\varphi}_i(\overline{\nu}, \overline{J}) = \overline{p} - s \le 0$ contradicting that $\tilde{f}_i(\overline{\nu}, \overline{J})$ is defined. Hence we must have $\mathrm{rk}(\nu, J) \ge i+1$. In fact we want to show that $\mathrm{rk}(\nu, J) = i+1$. Suppose $\mathrm{rk}(\nu, J) > i+1$. Then by the change in vacancy numbers by δ we have $\overline{p} = p = s$, so that $\widetilde{\varphi}_i(\overline{\nu}, \overline{J}) = s - \overline{s}$. So to achieve $\widetilde{\varphi}_i(\overline{\nu}, \overline{J}) > 0$ we need $\overline{s} < s$. This can only happen if $p_{\ell^{(i)}-1}^{(i)}(\nu) = s$ and $\ell^{(i-1)} < \ell^{(i)}$. If $m_{\ell^{(i)}-1}^{(i)}(\nu) > 0$, then the string of length $\ell^{(i)} - 1$ is singular. Since $\ell^{(i-1)} < \ell^{(i)}$ this contradicts the fact that δ picks the string of length $\ell^{(i)}$ in $(\nu, J)^{(i)}$. If $m_{\ell^{(i)}-1}^{(i)}(\nu) = 0$, by convexity Lemma 2.7.1, we get a similar contradiction. Hence we have that $b = i+1 \otimes b$. Note that the above arguments also shows that $\widetilde{\varphi}_i(\overline{\nu}, \overline{J}) = 1$ since $\overline{s} \ge s$ and $\overline{p} = p - 1$ if $\mathrm{rk}(\nu, J) = i+1$. Hence $f_i(b)$ is undefined since $\varphi_i(b') = \widetilde{\varphi}_i(\overline{\nu}, \overline{J}) = 1$.

Consider Case (2) where $f_i(b)$ is undefined and $f_i(b')$ is defined. This implies that $b = i+1 \otimes b'$. By induction $\widetilde{\varphi}_i(\overline{\nu}, \overline{J}) = \varphi_i(b') = 1$ so that by Lemma 2.4.13 we have $\overline{p} = \overline{s}+1$. Hence $\widetilde{\varphi}_i(\nu, J) = p - s = \overline{p} - 1 - s = \overline{s} - s$ by the change of vacancy numbers.

Therefore $\widetilde{\varphi}_i(\nu, J) = 0$ if $\overline{s} = s$. It remains to show that $p_{\ell+1}^{(i)}(\nu) \geq \overline{s}$ where $\ell := s^{(i)}$ is the length of the string in $(\overline{\nu}, \overline{J})^{(i)}$ selected by δ^{-1}. Hence the only problem occurs if $p_{\ell+1}^{(i)}(\overline{\nu}) = \overline{s}$ and $s^{(i-1)} < \ell$. If $m_{\ell+1}^{(i)}(\overline{\nu}) > 0$, this means that there is a singular string of length $\ell + 1 > s^{(i)}$ in $(\overline{\nu}, \overline{J})^{(i)}$ contradicting the maximality of $s^{(i)}$. If $m_{\ell+1}^{(i)}(\overline{\nu}) = 0$ one can again use convexity to arrive at similar contradiction.

By exclusion Case (3) follows from all the previous cases where at least one f_i or $\tilde{f}_i$ is defined.

Now let $B = B^{r,1} \otimes B'$ where $r \geq 2$. Consider the following diagram:

$$\begin{tikzcd}
\mathcal{P}(B) \arrow[rrr, "\Phi"] \arrow[dr, "\mathrm{lb}"] \arrow[ddd, "f_i"'] & & & \mathrm{RC}(L) \arrow[dl, "\mathrm{lb}_{rc}"'] \arrow[ddd, "\tilde{f}_i"] \\
& \mathcal{P}(\mathrm{lb}(B)) \arrow[r, "\Phi"] \arrow[d, "f_i"'] & \mathrm{RC}(\mathrm{lb}(L)) \arrow[d, "\tilde{f}_i"] & \\
& \mathcal{P}(\mathrm{lb}(B)) \arrow[r, "\Phi"] & \mathrm{RC}(\mathrm{lb}(L)) & \\
\mathcal{P}(B) \arrow[rrr, "\Phi"'] \arrow[ur, "\mathrm{lb}"] & & & \mathrm{RC}(L) \arrow[ul, "\mathrm{lb}_{rc}"']
\end{tikzcd}$$

Again the top and the bottom faces commute because of Definition 2.4.7 (3). The right face commutes by Lemma 2.7.3. The left face commutes by definition of f_i on the paths and we know lb is injective. By induction hypothesis the back face commutes too. Hence the front face commutes.

Finally let $B = B^{r,s} \otimes B'$ where $s \geq 2$. Consider the following diagram:

$$\begin{tikzcd}
\mathcal{P}(B) \arrow[rrr, "\Phi"] \arrow[dr, "\mathrm{ls}"] \arrow[ddd, "f_i"'] & & & \mathrm{RC}(L) \arrow[dl, "\mathrm{ls}_{rc}"'] \arrow[ddd, "\tilde{f}_i"] \\
& \mathcal{P}(\mathrm{ls}(B)) \arrow[r, "\Phi"] \arrow[d, "f_i"'] & \mathrm{RC}(\mathrm{ls}(L)) \arrow[d, "\tilde{f}_i"] & \\
& \mathcal{P}(\mathrm{ls}(B)) \arrow[r, "\Phi"] & \mathrm{RC}(\mathrm{ls}(L)) & \\
\mathcal{P}(B) \arrow[rrr, "\Phi"'] \arrow[ur, "\mathrm{ls}"] & & & \mathrm{RC}(L) \arrow[ul, "\mathrm{ls}_{rc}"']
\end{tikzcd}$$

As in the previous cases by Definition 2.4.7 (2), Lemma 2.7.4 and induction hypothesis all the faces commute except the front. Since the map ls is injective the front face of the above diagram commutes. This completes the proof of Theorem 2.4.14. $\square$

Chapter 3

Fermionic formulas for the characters of $N = 1$ and $N = 2$ superconformal algebras

3.1 Introduction

Bailey's lemma is a powerful method to prove q-series identities of the Rogers–Ramanujan-type [7]. One of the key features of Bailey's lemma is its iterative structure which was first observed by Andrews [4] (see also [63]). This iterative structure called the Bailey chain makes it possible to start with one seed identity and derive an infinite family of identities from it. The Bailey chain has been generalized to the Bailey lattice [1] which yields a whole tree of identities from a single seed.

The relevance of the Andrews–Bailey construction to physics was first revealed in the papers by Foda and Quano [23, 24] in which they derived identities for the Virasoro characters using Bailey's lemma. By the application of Bailey's lemma to polynomial versions of the character identity of one conformal field theory, one obtains character identities of another conformal field theory. This relation between the two conformal field theories is called Bailey flow. In [10] it was demonstrated that there is a Bailey flow from the minimal models $M(p-1,p)$ to $N=1$ and $N=2$ superconformal models. More precisely, it was shown that there is a Bailey flow from $M(p-1,p)$ to $M(p,p+1)$, and from $M(p-1,p)$ to the $N=1$ superconformal model $SM(p,p+2)$ and the unitary $N=2$ superconformal model with central charge $c=3(1-\frac{2}{p})$. In the conclusions of [10] it was conjecture that this construction can also be carried out for the nonunitary minimal models $M(p,p')$ where p and p' are relatively prime. In this chapter of the thesis we consider the nonunitary case. We show that starting with character identities for the nonunitary minimal model $M(p,p')$ of [11, 84], characters of the $N=1$ superconformal models $SM(p',2p+p')$, $SM(p',3p'-2p)$ and of the $N=2$ superconformal model with central element $c=3(1-\frac{2p}{p'})$ can be obtained via the Bailey flow. We also give a new Ramond sector character formula for a representation of the $N=2$ superconformal model with central element $c=3(1-\frac{2p}{p'})$ and calculate the corresponding fermionic formula.

The chapter is organized as follows. In section 3.2 we provide the necessary background about Bailey pairs and discuss how Bailey lemma can be used to prove RR type identities. In section 3.3 we derive new Bailey pairs using the Bose-Fermi identity for the minimal model $M(p,p')$. In section 3.4 we state the fermionic formulas of the $M(p,p')$ models following [10, 11, 12]. In section 3.5 necessary background for $N=1$ superconformal algebras is stated and the characters of the $N=1$ supersymmetric models $SM(2p+p',p')$ and $SM(3p'-2p,p')$ are derived using the Bailey flow. Explicit fermionic expressions for these characters are given. In section 3.6 the background regarding $N=2$ superconformal models is stated and a new character for the Ramond sector is derived. Then it is demonstrated how to obtain the characters of the $N=2$ superconformal model with central element $c = 3(1 - \frac{2p}{p'})$ via the Bailey flow along with the explicit fermionic expressions for these characters. In section 3.7 we conclude with some remarks.

Acknowledgment

Special thanks to Professor Gaberdiel and Hanno Klemm for their help through the jungle of literature regarding $N = 2$ character formulas and their help regarding the spectral flow of $N = 2$ superconformal algebras. We are grateful to both of them for their e-mail correspondences.

3.2 Bailey's lemma

A pair (α_n, β_n) of sequences $\{\alpha_n\}_{n\geq 0}$ and $\{\beta_n\}_{n\geq 0}$ is called a **Bailey pair** with respect to a if

$$\beta_n = \sum_{j=0}^{n} \frac{\alpha_j}{(q)_{n-j}(aq)_{n+j}} \tag{3.2.1}$$

where

$$(a)_n := (a;q)_n = \prod_{k=0}^{n-1}(1 - aq^k),$$

Theorem 3.2.1. ***(Bailey lemma)*** *If (α_n, β_n) is a Bailey pair with respect to a then for two parameters ρ_1, ρ_2,*

$$\begin{aligned}&\sum_{n=0}^{\infty}(\rho_1)_n(\rho_2)_n(aq/\rho_1\rho_2)^n\beta_n \\ &\qquad = \frac{(aq/\rho_1)_\infty(aq/\rho_2)_\infty}{(aq)_\infty(aq/\rho_1\rho_2)_\infty}\sum_{n=0}^{\infty}\frac{(\rho_1)_n(\rho_2)_n(aq/\rho_1\rho_2)^n\alpha_n}{(aq/\rho_1)_n(aq/\rho_2)_n}.\end{aligned} \tag{3.2.2}$$

Putting different Bailey pairs in this lemma many RR type identities were proved by Rogers [64, 65], Bailey [7] and Slater [81, 82] by considering the following two specializations of the parameters:

$$\rho_1 \longrightarrow \infty, \quad \rho_2 \longrightarrow \infty \tag{3.2.3}$$

$$\rho_1 \longrightarrow \infty, \quad \rho_2 = \text{finite}. \tag{3.2.4}$$

Let us discuss here briefly how RR identities (1.2.14) and (1.2.15) were proved using Bailey pairs and specialization (3.2.3). Let $\rho_1 \longrightarrow \infty, \quad \rho_2 \longrightarrow \infty$ in the Bailey lemma. We obtain

$$\sum_{n=0}^{\infty} a^n q^{n^2} \beta_n = \frac{1}{(aq)_\infty} \sum_{n=0}^{\infty} a^n q^{n^2} \alpha_n \tag{3.2.5}$$

The Bailey pair used to prove (1.2.14) is given by

$$\begin{aligned}
&\alpha_0 = 1 \\
&\alpha_n = (-1)^n q^{n(3n-1)/2}(1+q^n), \quad n \geq 1 \\
&\beta_n = \frac{1}{(q)_n} \quad n \geq 0
\end{aligned}$$

Inputing this Bailey pair in (3.2.5) with $a = 1$ and simplifying we find,

$$\sum_{n=0}^{\infty} \frac{q^{n^2}}{(q)_n} = \frac{1}{(q)_\infty} \sum_{n=-\infty}^{\infty} (-1)^n q^{n(5n-1)/2} \tag{3.2.6}$$

Note the left-hand side is exactly the left-hand side of (1.2.14). To prove that the right hand side of (3.2.6) equals the right hand side of (1.2.14) we use Jacobi's triple product identity,

$$\sum_{n=-\infty}^{n=\infty} q^{k^2} z^n = (q^2, -qz, -q/z; q^2)_\infty, \tag{3.2.7}$$

where $(a_1, a_2, a_3; q^2)_\infty = (a_1; q^2)_\infty (a_2; q^2)_\infty (a_3; q^2)_\infty$.

Using (3.2.7) we can rewite right-hand side of (3.2.6) as

$$\begin{aligned}
\frac{1}{(q)_\infty} \sum_{n=-\infty}^{\infty} (-1)^n q^{n(5n-1)/2} &= \frac{1}{(q)_\infty} \sum_{n=-\infty}^{\infty} (-1)^n (-q^{1/2})^n (q^{5/2})^{n^2} \\
&= \frac{1}{(q)_\infty} \times (q^5; q^5)_\infty \times (q^2; q^5)_\infty \times (q^3; q^5)_\infty \\
&= \prod_{n=1}^{\infty} \frac{1}{(1-q^{5n-1})(1-q^{5n-4})}
\end{aligned}$$

which is the right hand side of (1.2.14). Similarly, the second RR identity (1.2.15) can be proved using the Bailey pair

$$\begin{aligned}
\alpha_n &= (-1)^n \frac{q^{n(3n+1)/2}(1-q^{2n+1})}{1-q}, \quad n \geq 0, \\
\beta_n &= \frac{1}{(q)_n} \quad n \geq 0.
\end{aligned}$$

Upon the discovery of CFTs [13, 14] from Slater's famous list of 130 identities [81] it was observed that the specialization (3.2.3) leads to characters of minimal model $M(p, p')$ and the second specialization (3.2.4) leads to the characters of $N = 1$ superconformal model. Hence by putting suitable Bailey pairs and then using some appropiate specialization of the parameters one can derive Bose-Fermi type character identities for CFTs. Therefore, the main question is: how do we find suitable Bailey pairs? The sources for the list of bailey pairs used by Rogers, Bailey and Slater were some well known hypergeometric series identities. In physics Foda and Quano [24] observed the remarkable fact that the finitized Bose-Fermi identities of the configuration sum of the ABF model are of the form (3.2.1). Hence one can read off Bailey pairs from this. This fact has been used in [5, 10, 12, 24, 85] to derive Bose Fermi identities for some subset of the minimal models, superconformal models and higher level coset models. Berkovich, McCoy and Schilling explored this in [10] for the Minimal model $M(p-1,p)$. They calculated the Bailey pairs using the Bose Fermi identity for the unitary minimal model $M(p-1,p)$ and used the specialization (3.2.4) and the additional specialization:

$$\rho_1 = \text{finite}, \quad \rho_2 = \text{finite}. \tag{3.2.8}$$

to obtain the characters of $N = 1, 2$ superconformal models, hence demonstrated a Bailey flow between these CFTs.

Following [10], we are going to use an extended definition of Bailey pair called the bilateral Bailey pair. A pair (α_n, β_n) of sequences $\{\alpha_n\}_{n\in\mathbb{Z}}$ and $\{\beta_n\}_{n\in\mathbb{Z}}$ is said to be a **bilateral Bailey pair** with respect to a if

$$\beta_n = \sum_{j=-\infty}^{n} \frac{\alpha_j}{(q)_{n-j}(aq)_{n+j}}. \tag{3.2.9}$$

Theorem 3.2.2 (Bilateral Bailey lemma [4, 7, 10]). *If (α_n, β_n) is a bilateral Bailey pair then*

$$\begin{aligned}\sum_{n=-\infty}^{\infty} &(\rho_1)_n(\rho_2)_n(aq/\rho_1\rho_2)^n\beta_n \\ &= \frac{(aq/\rho_1)_\infty(aq/\rho_2)_\infty}{(aq)_\infty(aq/\rho_1\rho_2)_\infty} \sum_{n=-\infty}^{\infty} \frac{(\rho_1)_n(\rho_2)_n(aq/\rho_1\rho_2)^n\alpha_n}{(aq/\rho_1)_n(aq/\rho_2)_n}.\end{aligned} \tag{3.2.10}$$

This lemma has been used with various Bailey pairs and different specializations of the parameters ρ_1 and ρ_2 to prove many q-series identities (see for example [1, 10, 24, 81]). In this paper the bilateral Bailey lemma is used to derive character identities for $N = 1, 2$ superconformal algebras from nonunitary minimal models $M(p, p')$.

A useful way to obtain new Bailey pairs from old ones is the construction of dual Bailey pairs. If (α_n, β_n) is a bilateral Bailey pair with respect to a, the **dual Bailey pair** (A_n, B_n) is defined as

$$\begin{aligned} A_n(a,q) &= a^n q^{n^2} \alpha_n(a^{-1}, q^{-1}), \\ B_n(a,q) &= a^{-n} q^{-n^2-n} \beta_n(a^{-1}, q^{-1}). \end{aligned} \tag{3.2.11}$$

Then (A_n, B_n) satisfies (3.2.9) with respect to a.

3.3 Bailey pairs from the minimal models $M(p,p')$

In this section we derive new Bailey pairs using the Bose Fermi identity for the minimal model $M(p,p')$.

As shown by Foda and Quano [24], the Bose-Fermi character identities [9, 11, 25, 84] for the minimal models $M(p,p')$ are of the form

$$B^{(p,p')}_{r(b),s}(L,b;q) = q^{-\mathcal{N}_{r(b),s}} F^{(p,p')}_{r(b),s}(L,b;q), \tag{3.3.1}$$

where the bosonic side is given by

$$\begin{aligned} B^{(p,p')}_{r(b),s}(L,b;q) = \sum_{j=-\infty}^{\infty} \Bigg(& q^{j(jpp'+r(b)p'-sp)} \begin{bmatrix} L \\ \frac{1}{2}(L+s-b)-jp' \end{bmatrix}_q \\ & - q^{(jp-r)(jp'-s)} \begin{bmatrix} L \\ \frac{1}{2}(L-s-b)+jp' \end{bmatrix}_q \Bigg). \end{aligned} \tag{3.3.2}$$

The function fermionic formula $F^{(p,p')}_{r(b),s}(L,b;q)$ will be discussed in the next section. The normalization constant $\mathcal{N}_{r(b),s}$ is explicitly calculated in [11]. Since the explicit expression is not used any where in our calculations we will exclude the details.

We will now construct new Bailey pairs using (3.3.1). For simplicity we are going to write r for $r(b)$. Let us set $L = 2n + b - s + 2x$ to rewrite the q-binomial coefficients in (3.3.2),

$$\begin{aligned}
\begin{bmatrix} 2n+b-s+2x \\ n+x-p'j \end{bmatrix}_q &= \frac{\left(q^{b-s+2x+1}\right)_{2n}}{(q)_{n-(p'j-x)}\left(q^{b-s+2x+1}\right)_{n+(p'j-x)}}, \\
\begin{bmatrix} 2n+b-s+2x \\ n+x-s+p'j \end{bmatrix}_q &= \frac{\left(q^{b-s+2x+1}\right)_{2n}}{(q)_{n-(p'j-b-x)}\left(q^{b-s+2x+1}\right)_{n+(p'j-b-x)}}.
\end{aligned}$$

Following [10, 24] we rewite (3.3.1) as

$$\begin{aligned}
q^{-\mathcal{N}_{r(b),s}} F^{(p,p')}_{r(b),s}(L,b;q) = \sum_{j=-\infty}^{\infty} \Bigg(& q^{j(jpp'+rp'-sp)} \frac{\left(q^{b-s+2x+1}\right)_{2n}}{(q)_{n-(p'j-x)}\left(q^{b-s+2x+1}\right)_{n+(p'j-x)}} \\
& - q^{(jp-r)(jp'-s)} \frac{\left(q^{b-s+2x+1}\right)_{2n}}{(q)_{n-(p'j-b-x)}\left(q^{b-s+2x+1}\right)_{n+(p'j-b-x)}} \Bigg).
\end{aligned}$$

where $L = 2n + b - s + 2x$. This is in the form of (3.2.9), hence we can read off the bilateral Bailey pair relative to $a = q^{b-s+2x}$

$$\begin{aligned}
\alpha_n &= \begin{cases} q^{j(jpp'+rp'-sp)} & \text{if } n = jp' - x \\ -q^{(jp-r)(jp'-s)} & \text{if } n = jp' - b - x \\ 0 & \text{otherwise} \end{cases} \\
\beta_n &= \frac{q^{-\mathcal{N}_{r,s}}}{(aq)_{2n}} F^{(p,p')}_{r,s}(2n+b-s+2x, b; q).
\end{aligned} \tag{3.3.3}$$

where $x = \frac{L-2n-b+s}{2}$.

Using the definition (3.2.11) we calculate the Bailey pair dual to (3.3.3) relative to $a = q^{b-s+2x}$ and denote it by $(\hat{\alpha}_n, \hat{\beta}_n)$ where

$$\hat{\alpha}_n = \begin{cases} q^{j^2p'(p'-p)-jp'(r-b)-js(p'-p)-x(b+x-s)} & \text{if } n = jp' - x \\ -q^{(jp'-s)(j(p'-p)+r-b)-x(b+x-s)} & \text{if } n = jp' - b - x \\ 0 & \text{otherwise} \end{cases} \tag{3.3.4}$$
$$\hat{\beta}_n = \frac{q^{\mathcal{N}_{r,s}}}{(aq)_{2n}} a^n q^{n^2} F_{r,s}^{(p,p')}(2n + b - s + 2x, b; q^{-1}).$$

Inserting (3.3.3) and (3.3.4) into the bilateral Bailey lemma yields

$$\sum_{n=0}^{\infty} (\rho_1)_n (\rho_2)_n (aq/\rho_1\rho_2)^n \frac{q^{-\mathcal{N}_{r(b),s}}}{(aq)_{2n}} F_{r,s}^{(p,p')}(2n + b - s + 2x, b; q)$$
$$= \frac{(aq/\rho_1)_\infty (aq/\rho_2)_\infty}{(aq)_\infty (aq/\rho_1\rho_2)_\infty} \sum_{j=-\infty}^{\infty} \left(\frac{(\rho_1)_{jp'-x}(\rho_2)_{jp'-x}}{(aq/\rho_1)_{jp'-x}(aq/\rho_2)_{jp'-x}} (aq/\rho_1\rho_2)^{jp'-x} \right. \tag{3.3.5}$$
$$\times q^{j(jpp'+rp'-sp)} - \frac{(\rho_1)_{jp'-b-x}(\rho_2)_{jp'-b-x}}{(aq/\rho_1)_{jp'-b-x}(aq/\rho_2)_{jp'-b-x}}$$
$$\left. \times (aq/\rho_1\rho_2)^{jp'-b-x} q^{(jp-r)(jp'-s)} \right)$$

and

$$\begin{aligned}
&\sum_{n=0}^{\infty}(\rho_1)_n(\rho_2)_n(aq/\rho_1\rho_2)^n\frac{q^{\mathcal{N}_{r(b),s}}}{(aq)_{2n}}a^n q^{n^2}F_{r,s}^{(p,p')}(2n+b-s+2x,b;q^{-1})\\
&=\frac{(aq/\rho_1)_\infty(aq/\rho_2)_\infty}{(aq)_\infty(aq/\rho_1\rho_2)_\infty}\sum_{j=-\infty}^{\infty}\left(\frac{(\rho_1)_{jp'-x}(\rho_2)_{jp'-x}}{(aq/\rho_1)_{jp'-x}(aq/\rho_2)_{jp'-x}}(aq/\rho_1\rho_2)^{jp'-x}\right.\\
&\times q^{j^2p'(p'-p)-jp'(r-b)-js(p'-p)-x(b+x-s)}-\frac{(\rho_1)_{jp'-b-x}(\rho_2)_{jp'-b-x}}{(aq/\rho_1)_{jp'-b-x}(aq/\rho_2)_{jp'-b-x}}\\
&\left.\times\,(aq/\rho_1\rho_2)^{jp'-b-x}q^{(jp'-s)(j(p'-p)+r-b)-x(b+x-s)}\right).
\end{aligned}\tag{3.3.6}$$

As in [10], we are going to consider different specializations of the parameters ρ_1 and ρ_2 in (3.3.5) and (3.3.6) to get character identities for $N=1,2$ superconformal algebras.

3.4 Fermionic formulas for $M(p,p')$

So far we have only considered the bosonic side of (3.3.1) explicitly. For the fermionic side we will consider two cases $p<p'<2p$ and $p'>2p$ separately with p and p' relatively prime and r, s being pure Takahashi length.

3.4.1 Fermionic formula for $M(p,p')$ with $p<p'<2p$

We need to introduce a lot of notations to give the explicit fermionic formula and we follow [12, Section 4] here. The fermionic formula depends on the continued fraction decomposition

$$\frac{p'}{p'-p} = 1 + \nu_0 + \cfrac{1}{\nu_1 + \cfrac{1}{\nu_2 + \cdots \cfrac{1}{\nu_{n_0}+2}}}.$$

Define $t_i = \sum_{j=0}^{i-1} \nu_j$ for $1 \le i \le n_0+1$ and the fractional level incidence matrix $\mathcal{I}_B$ and corresponding Cartan matrix B as

$$(\mathcal{I}_B)_{j,k} = \begin{cases} \delta_{j,k+1} + \delta_{j,k-1} & \text{for } 1 \le j < t_{n_0+1}, j \ne t_i \\ \delta_{j,k+1} + \delta_{j,k} - \delta_{j,k-1} & \text{for } j = t_i, 1 \le i \le n_0 - \delta_{\nu_{n_0},0} \\ \delta_{j,k+1} + \delta_{\nu_{n_0},0}\delta_{j,k} & \text{for } j = t_{n_0+1} \end{cases}$$

$$B = 2I_{t_{n_0+1}} - \mathcal{I}_B,$$

where I_n is the identity matrix of dimension n. Recursively define

$$y_{m+1} = y_{m-1} + (\nu_m + \delta_{m,0} + 2\delta_{m,n_0})y_m, \qquad y_{-1} = 0, \qquad y_0 = 1,$$
$$\overline{y}_{m+1} = \overline{y}_{m-1} + (\nu_m + \delta_{m,0} + 2\delta_{m,n_0})\overline{y}_m, \qquad \overline{y}_{-1} = -1, \qquad \overline{y}_0 = 1.$$

Then the Takahashi length and truncated Takahashi length are given by

$$\begin{aligned} \ell_{j+1} &= y_{m-1} + (j - t_m)y_m \\ \overline{\ell}_{j+1} &= \overline{y}_{m-1} + (j - t_m)\overline{y}_m \end{aligned} \qquad \text{for } t_m < j \le t_{m+1} + \delta_{m,n_0} \text{ with } 0 \le m \le n_0.$$

Let us define the t_{n_0+1}-dimensional vectors $Q^{(j)}$ $(j = 1, 2, \cdots, t_{n_0+1}+1)$ which we will need to specify the parity of the summation variables in the fermionic formula. For $1 \le i \le t_{n_0+1}$ and $0 \le m \le n_0$ such that $t_m < j \le t_{m+1} + \delta_{m,n_0}$ the components of $Q^{(j)}$

are defined recursively as

$$Q_i^{(j)} = \begin{cases} 0 & \text{for } j \le i \le t_{n_0+1}, \\ j - i & \text{for } t_m \le i < j, \\ Q_{i+1}^{(j)} + Q_{t_{m'}+1}^{(j)} & \text{for } t_{m'-1} \le i < t_{m'}, 1 \le m' \le m. \end{cases} \tag{3.4.1}$$

When $\nu_{n_0} = 0$, so that $t_{n_0+1} = t_{n_0}$, we need to set the initial condition $Q_{t_{n_0}+1}^{(t_{n_0}+1)} = 0$. Also define

$$Q_{\mathbf{u}} = \sum_{j=1}^{t_{n_0+1}+1} u_j Q^{(j)},$$

and for $t_i < j \le t_{i+1}$,

$$(A_{\mathbf{u},\mathbf{v}})_j = \begin{cases} u_j & \text{for } i \text{ odd}, \\ v_j & \text{for } i \text{ even}. \end{cases} \tag{3.4.2}$$

For $b = \ell_{\beta+1}$, $r(b) = \overline{\ell}_{\beta+1}$ with $t_\xi < \beta \le t_{\xi+1} + \delta_{\xi,n_0}$ and $s = \ell_{\sigma+1}$ with $t_\zeta < \sigma \le t_{\zeta+1} + \delta_{\zeta,n_0}$ the fermionic formula is given by

$$F_{r,s}^{(p,p')}(L, b; q) = q^{k_{b,s}} \sum_{\mathbf{m} \equiv Q_{\mathbf{u}+\mathbf{v}} \pmod 2} q^{\frac{1}{4}\mathbf{m}^t B \mathbf{m} - \frac{1}{2} A_{\mathbf{u},\mathbf{v}} \mathbf{m}} \prod_{j=1}^{t_{n_0+1}} \begin{bmatrix} n_j + m_j \\ m_j \end{bmatrix}_q' \tag{3.4.3}$$

where $k_{b,s}$ is a normalization constant and $\mathbf{n}, \mathbf{m} \in \mathbb{Z}^{t_{n_0+1}}$ such that

$$\mathbf{n} + \mathbf{m} = \frac{1}{2}\Big(\mathcal{I}_B \mathbf{m} + \mathbf{u} + \mathbf{v} + L\mathbf{e}_1\Big) \tag{3.4.4}$$

with $\mathbf{e}_i$ the standard i-th basis element of $\mathbb{Z}^{t_{n_0+1}}$, $\mathbf{u} = \mathbf{e}_\beta - \sum_{k=\xi+1}^{n_0} \mathbf{e}_{t_k}$ and $\mathbf{v} = \mathbf{e}_\sigma - \sum_{k=\zeta+1}^{n_0} \mathbf{e}_{t_k}$. The notation $\mathbf{m} \equiv Q_{\mathbf{u}+\mathbf{v}} \pmod 2$ stands for m_j even when $(Q_{\mathbf{u}+\mathbf{v}})_j$ is even and m_j is odd when $(Q_{\mathbf{u}+\mathbf{v}})_j$ is odd.

The q-binomial is also defined for negative entries

$$\begin{bmatrix} n+m \\ m \end{bmatrix}_q' = \frac{(q^{n+1})_m}{(q)_m}.$$

Note that

$$\begin{bmatrix} n+m \\ m \end{bmatrix}_{q^{-1}}' = q^{-nm} \begin{bmatrix} n+m \\ m \end{bmatrix}_q' . \tag{3.4.5}$$

In fact using (3.4.5) we get the following dual form of the fermionic formula that will be useful later on

$$F_{r,s}^{(p,p')}(L,b;q^{-1}) =$$
$$q^{-k_{b,s}} \sum_{\mathbf{m}\equiv Q_{\mathbf{u}+\mathbf{v}}} q^{\frac{1}{4}\mathbf{m}^t B\mathbf{m} - \frac{1}{2}Lm_1 + \frac{1}{2}A_{\mathbf{u},\mathbf{v}}\mathbf{m} - \frac{1}{2}\mathbf{m}^t(\mathbf{u}+\mathbf{v})} \prod_{j=1}^{t_{n_0+1}} \begin{bmatrix} n_j+m_j \\ m_j \end{bmatrix}_q' . \tag{3.4.6}$$

3.4.2 Fermionic formula for $M(p,p')$ with $p' > 2p$:

We use the fermionic formula $F_{r(b),s}^{(p,p')}(L,b;q)$ derived in [11, section 10] with notations defined in [11, section 2, section 3] . In this case the fermionic formula depends on the continued fraction decomposition of

$$\frac{p'}{p} = \nu_0 + 1 + \cfrac{1}{\nu_1 + \cfrac{1}{\nu_2 + \cdots \cfrac{1}{\nu_{n_0}+2}}}$$

Let t_i for $1 \leq i \leq n_0 + 1$ be the same as in the previous case along with $t_0 = -1$. Recursively define

$$y_{m+1} = y_{m-1} + (\nu_m + \delta_{m,0} + 2\delta_{m,n_0})y_m, \quad y_{-1} = 0, \quad y_0 = 1, \quad 0 \leq m \leq n_0$$

$$z_{m+1} = z_{m-1} + (\nu_{m+1} + 2\delta_{m+1,n_0})z_m, \quad z_{-1} = 0, \quad z_0 = 1, \quad 0 \leq m \leq n_0 - 1.$$

Then the Takahashi length and truncated Takahashi length are given by

$$\ell_{j+1}^{(m)} = \begin{cases} j+1 & \text{for} \quad m = 0 \text{and } 0 \leq j \leq t_1 \\ y_{m-1} + (j - t_m)y_m & \text{for } 1 \leq \mu \leq n_0 \text{ and } 1 + t_m \leq j \leq t_{1+\mu} + \delta_{n,\mu}. \end{cases}$$

$$\tilde{\ell}_{j+1}^{(m)} = \begin{cases} z_{m-2} + (j - t_m)z_{m-1} & \text{for } 1 + t_m < j \leq t_{m+1} + \delta_{m,n_0} \text{ with } 1 \leq m \leq n_0 \\ 0 & \text{for } m = 0. \end{cases}$$

Let us define the corresponding Cartan matrix B in this situation. The nonzero elements of the matrix B are given by the $\nu_0 \times \nu_0$ matrix

$$C_T^{-1} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 2 & \cdots & 2 \\ 1 & 2 & 3 & \cdots & 3 \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ 1 & 2 & 3 & \cdots & \nu_0 \end{pmatrix} \tag{3.4.7}$$

as

$$\begin{aligned}
&B_{i,j} = 2\left(C_T^{-1}\right)_{i,j} \qquad \text{for } 1 \le i,j \le \nu_0 \\
&B_{\nu_0+1,j} = B_{j,\nu_0+1} \quad \text{for } 1 \le j \le \nu_0 \\
&B_{j,j} = \frac{\nu_0}{2}\delta_{j,\nu_0+1} + (1 - \frac{1}{2}\sum_{i=2}^{t_{n_0}} \delta_{j,t_i}) \quad \text{for } \nu_0 + 1 \le j \le t_{n_0+1} - 1 \\
&B_{t_{n_0+1},t_{n_0+1}} = 1 \\
&B_{j,j+1} = -\frac{1}{2} + \sum_{i=2}^{n_0} \delta_{j,t_i} \quad \text{for } j > \nu_0 \\
&B_{j+1,j} = -\frac{1}{2} \quad \text{for } j > \nu_0
\end{aligned} \tag{3.4.8}$$

Define the t_{n_0+1}-dimensional vector $\overline{\mathbf{e}}_k$ and $1 + t_{n_0+1}$-dimensional vector $\mathbf{e}_k$ by

$$(\overline{\mathbf{e}}_k)_j = \begin{cases} \delta_{j,k} & \text{for} \quad 1 \le k \le t_{n_0+1} \\ 0 & \text{for} \quad k = 0, 1 + t_{n_0+1} \end{cases} \tag{3.4.9}$$

$$(\mathbf{e}_k)_j = \begin{cases} \delta_{j,k} & \text{for} \quad 1 \le k \le t_{n_0+1} + 1 \\ 0 & \text{for} \quad k = 0. \end{cases} \tag{3.4.10}$$

The t_{n_0+1}-dimensional vectors $\overline{\mathbf{u}}, \overline{\mathbf{u}}_+, \overline{\mathbf{u}}_-, V$ and $\overline{E}_{a,b}$ are defined by

$$
\begin{aligned}
\overline{\mathbf{u}} &= \overline{\mathbf{u}}_+ + \overline{\mathbf{u}}_- \\
\overline{\mathbf{u}}_+ &= \sum_{i=1}^{\nu_0} \overline{u}_i \overline{\mathbf{e}}_i \\
\overline{\mathbf{u}}_- &= \sum_{i=\nu_0+1}^{t_{n_0+1}} \overline{u}_i \overline{\mathbf{e}}_i \\
V &= \sum_{i=1}^{\nu_0} i \overline{\mathbf{e}}_i \\
\overline{E}_{a,b} &= \sum_{i=a}^{b} \overline{\mathbf{e}}_i
\end{aligned}
\tag{3.4.11}
$$

We will write $A = A^{(b)} + A^{(s)}$ where the t_{n_0+1}-dimensional vectors $A^{(b)}$ and $A^{(s)}$ are defined as

$$
A_k^{(b)} = \begin{cases} -\frac{1}{2} u_k & \text{for } k \text{ in an even, nonzero zone,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.4.12}
$$

where $\mathbf{u}$ is a $t_{n_0+1} + 1$-dimensional vector.

$$
A_k^{(s)} = \begin{cases} -\frac{1}{2}\overline{\mathbf{u}}(s)_k - \frac{1}{2} V^t \cdot \overline{\mathbf{u}}(s) \delta_{k,\nu_0+1} & \text{for } k \text{ in an odd zone,} \\ -\frac{1}{2} \overline{\mathbf{e}}^t B \overline{\mathbf{u}}(s)_+ & \text{for } k \text{ in an even zone.} \end{cases} \tag{3.4.13}
$$

where t_{n_0+1}-dimensinal vector $\overline{\mathbf{u}}(s)$ is given by: for $1 \le k \le t_{n_0+1}$

$$
(\overline{\mathbf{u}}(s))_k = \begin{cases} \delta_{k,j_s} - \sum_{i=\mu_s+1}^{n_0} & \text{for } 1 + t_{\mu_s} \le j_s \le t_{\mu_s+1} \text{ and } \mu_s \le n-1, \\ \delta_{k,j_s} & \text{for } t_\mu < j_s, \mu_s = n \end{cases} \tag{3.4.14}
$$

For $\mathbf{n}, \mathbf{m} \in \mathbf{Z}^{\mathbf{t_{n_0+1}}}$ define

$$\tilde{\mathbf{m}} = (n_1, n_2, \cdots, n_{\nu_0}, m_{\nu_0+1}, m_{\nu_0+2}, \cdots, m_{t_{n_0+1}})$$
$$\tilde{\mathbf{n}} = (m_1, m_2, \cdots, m_{\nu_0}, n_{\nu_0+1}, n_{\nu_0+2}, \cdots, n_{t_{n_0+1}})$$

such that

$$\tilde{\mathbf{n}} + \tilde{\mathbf{m}} = (I_{t_{n_0}+1} - B)\tilde{\mathbf{m}} + L\overline{E}_{1,\nu_0} + \frac{L}{2}\overline{\mathbf{e}}_{\nu_0+1} + \frac{B}{2}\overline{\mathbf{u}}_+ + \frac{1}{2}\overline{\mathbf{e}}_{\nu_0+1}(\overline{\mathbf{u}}_+^t.V) + \frac{1}{2}\overline{\mathbf{u}}_- \quad (3.4.15)$$

For $b = \ell^{(\mu)}_{j_\mu+1}, r(b) = \delta_{\mu,0} + \tilde{\ell}^{(\mu)}_{j_\mu+1}$ with $1 + t_\mu < j_\mu \leq t_{\mu+1} + \delta_{\mu,n_0}$ and $s = \ell^{(\beta)}_{j_\beta+1}$ with $1 + t_\beta < j_\beta \leq t_{\beta+1} + \delta_{\beta,n_0}$ the fermoinic formula can be written as:

$$F^{(p,p')}_{r(b),s}(L,b;q) = q^{C_{b,s}} \sum_{\tilde{\mathbf{m}} \equiv \overline{Q}_{\mathbf{u}} \pmod 2} q^{\frac{1}{2}\tilde{\mathbf{m}}^t B \tilde{\mathbf{m}} + A^t \tilde{\mathbf{m}}} \prod_{j=1}^{t_{n_0+1}} \begin{bmatrix} \tilde{n}_j + \tilde{m}_j \\ \tilde{m}_j \end{bmatrix}'_q \quad (3.4.16)$$

where the normalization constant $C_{b,s} = C(j_\mu)$ is defined in [11, (8.33)] as

$$\begin{aligned} &C(j_0) = 0 \quad \text{for } 1 \leq j_0 \leq t_1 \\ &C(j_\mu) = \frac{1}{2}(-1)^\mu + (j_\mu - t_\mu)\{-\frac{\nu_0+1}{4}\theta(\mu \quad \text{odd}) + c(t_\mu) + c(t_{\mu-1})\} \\ &\qquad \text{for} \quad t_\mu + 1 \leq j_\mu \leq t_{\mu+1} + 2\delta_{\mu,n_0} \quad \text{for} \quad 1 \leq \mu \leq n_0 \end{aligned} \quad (3.4.17)$$

where $\theta(S) = 1$ if S is true and $\theta(S) = 0$ if S is false.

Also in our case $\mathbf{u} = \mathbf{e}_{j_\mu} - \sum_{i=\mu+1}^{n_0} \mathbf{e}_i$. To explain the sum $\tilde{\mathbf{m}} \equiv \overline{Q}_{\mathbf{u}} \pmod 2$ let us introduce the following notation. For $1 \leq k \leq t_{n_0+1}, t_{\mu_0} + \delta_{\mu_0,0} + 1 \leq j \leq t_{\mu_0+1} + \delta_{\mu_0,n_0}$, for some $0 \le \mu_0 \le n_0$,

$$w_k^{(j)} = \begin{cases} 0 & \text{for } k \ge j, \\ j-k & \text{for } t_{\mu_0} \le k < j, \\ w_{k+1}^{(j)} + w_{t_{\mu+1}+1}^{(j)} & \text{for } t_\mu \le k < t_{\mu+1}, 1 \le \mu < \mu_0. \end{cases}$$

then define

$$\mathbf{w}(u_{1+t_{n_0+1}}, \overline{\mathbf{u}}) = \sum_{k=1}^{t_{n_0+1}} \overline{\mathbf{e}}_k \left(u_{1+t_{n_0+1}} w_k^{(1+t_{n_0+1}} + \sum_{j=1}^{t_{n_0+1}} w_k^{(j)} \overline{u}_j \right).$$

With the notations above $\tilde{\mathbf{m}} \equiv \overline{Q}_{\mathbf{u}} \pmod 2$ means $\tilde{\mathbf{m}}_- \in 2\mathbb{Z}^{t_{n_0+1}-\nu_0} + \mathbf{w}_-(u_{1+t_{n_0+1}}, \overline{\mathbf{u}})$,and $m_+ \in \mathbb{Z}^{\nu_0}$. This restriction makes sure that the entries of all q-binomials in (3.4.16) are integers as long as $\mathbf{u} \in \mathbb{Z}^{1+t_{n_0+1}}$.

Again using (3.4.5) we get the following useful fermionic formula:

$$F_{r(b),s}^{(p,p')}(L;q^{-1}) = q^{-C_{b,s}} \sum_{\tilde{\mathbf{m}} \equiv \overline{Q}_{\mathbf{u}}} q^{\frac{1}{2}\tilde{\mathbf{m}}^t B \tilde{\mathbf{m}} - L\tilde{\mathbf{m}}^t \overline{E}_{1,\nu_0} - \frac{L}{2}\tilde{\mathbf{m}}^t \mathbf{e}_{\nu_0+1}}$$
$$\times\, q^{-A^t\tilde{\mathbf{m}} - \frac{1}{2}\tilde{\mathbf{m}}^t B \mathbf{u}_+ - \frac{1}{2}\tilde{\mathbf{m}}^t \mathbf{e}_{\nu_0+1}(\mathbf{u}_+^t.V) - \frac{1}{2}\tilde{\mathbf{m}}^t \mathbf{u}_-} \times \prod_{j=1}^{t_{n_0+1}} \begin{bmatrix} \tilde{n}_j + \tilde{m}_j \\ \tilde{m}_j \end{bmatrix}_q' \qquad (3.4.18)$$

We will use this in the later sections.

3.5 $N = 1$ Superconformal character from $M(p, p')$

The $N = 1$ superconformal algebra is the infinite dimensional Lie super algebra [59, 66, 39] with basis $L_n, G_r, \tilde{C}$ and (anti)-commutation relation given by

$$[L_m, L_n] = (m-n)L_{m+n} + \frac{\tilde{C}}{8}(m^3 - m)\delta_{m+n,0}$$
$$[L_m, G_r] = (\frac{m}{2} - r)G_{m+r}$$
$$\{G_r, G_s\} = 2L_{r+s} + \frac{\tilde{C}}{2}(r^2 - \frac{1}{4})\delta_{r+s,0}$$

where m, n are integers $\tilde{C}$ is the central charge and its eigen value is parametrized by $\tilde{c} = \frac{3}{2} - \frac{3(p-p')^2}{pp'}$. If r, s are integers the algebra is called Neveu-Schwarz (NS) algebra and if r, s are half integers then the algrebra is called Ramond (R) algebra. Let us denote these algebras by $SM(p, p')$.

The character formula of these algebras are calculated in [28] and are given by,

$$\tilde{\chi}_{r,s}^{(p,p')}(q) = \tilde{\chi}_{p-r,p'-s}^{(p,p')}(q) = \frac{(-q^{\epsilon_{r-s}})_\infty}{(q)_\infty} \sum_{j=-\infty}^{\infty} \Big(q^{\frac{j(jpp'+rp'-sp)}{2}} - q^{\frac{(jp-r)(jp'-s)}{2}} \Big), \qquad (3.5.1)$$

where $1 \le r \le p-1, 1 \le s \le p'-1$, p and $(p'-p)/2$ are relatively prime and

$$\epsilon_i = \begin{cases} \frac{1}{2} & \text{if } i \text{ is even (NS-sector)}, \\ 1 & \text{if } i \text{ is odd (R-sector)}. \end{cases} \qquad (3.5.2)$$

In this section we are going to consider the specialization of the form of (3.2.4) in (3.3.5) and (3.3.6). We will see that these give Bailey flows from the minimal model $M(p, p')$ to the superconformal models $SM(p', 2p + p')$ and $SM(p', 3p' - 2p)$.

3.5.1 The model $SM(p',2p+p')$

Specializing $\rho_1 \longrightarrow \infty$ and $\rho_2 = -q^{\frac{b-s+1}{2}}$ with $x=0$ in (3.3.5) and comparing with (3.5.1) we find for $b-s$ even (NS sector)

$$\tilde{\chi}^{(p',2p+p')}_{s,2r+b}(q) = \sum_{n\geq 0} \frac{q^{\frac{1}{2}(n^2+nb-ns)}(-q^{\frac{1}{2}})_{n+(b-s)/2}}{(q)_{2n+b-s}} q^{-\mathcal{N}_{r,s}} F^{(p,p')}_{r,s}(2n+b-s,b;q) \tag{3.5.3}$$

and for $b-s$ odd (R-sector)

$$\tilde{\chi}^{(p',2p+p')}_{s,2r+b}(q) = \sum_{n\geq 0} \frac{q^{\frac{1}{2}(n^2+nb-ns)}(-q)_{n+(b-s-1)/2}}{(q)_{2n+b-s}} q^{-\mathcal{N}_{r,s}} F^{(p,p')}_{r,s}(2n+b-s,b;q). \tag{3.5.4}$$

Hence there is a Bailey flow from $M(p,p')$ to the superconformal model $SM(p',2p+p')$. Let us calculate the fermionic formula using section 3.4.

Fermionic formula for $SM(2p+p',p')$ with $p<p'<2p$: To obtain an explicit fermionic formula set $m_0 = L = 2n+b-s$ and insert (3.4.3) into (3.5.3). Then using

$$(-q^{\frac{1}{2}})_{\frac{m_0}{2}} = \sum_{k=0}^{\frac{m_0}{2}} q^{\frac{1}{2}(\frac{m_0}{2}-k)^2} \begin{bmatrix} \frac{m_0}{2} \\ k \end{bmatrix}_q \tag{3.5.5}$$

we find

$$\begin{aligned}\tilde{\chi}^{(p',2p+p')}_{s,2r+b}(q) = q^{-\frac{1}{8}(b-s)^2-\mathcal{N}_{r,s}+k_{b,s}} &\sum_{\substack{m_0=0\\ m_0 \text{ even}}}^{\infty} \sum_{k=0}^{\frac{m_0}{2}} \sum_{\mathbf{m}\equiv Q_{\mathbf{u}+\mathbf{v}}} q^{\frac{1}{8}m_0^2+\frac{1}{2}(\frac{m_0}{2}-k)^2} \\ &\times q^{\frac{1}{4}\mathbf{m}^t B\mathbf{m}-\frac{1}{2}A_{\mathbf{u},\mathbf{v}}\mathbf{m}} \times \frac{1}{(q)_{m_0}} \begin{bmatrix} \frac{m_0}{2} \\ k \end{bmatrix}_q \prod_{j=1}^{t_{n_0+1}} \begin{bmatrix} n_j+m_j \\ m_j \end{bmatrix}_q' .\end{aligned} \tag{3.5.6}$$

Setting $\mathbf{p} = (k, m_0, \mathbf{m}) \in \mathbb{Z}^{t_{n_0+1}+2}$ we can write

$$\frac{1}{8}m_0^2 + \frac{1}{2}(\frac{m_0}{2} - k)^2 + \frac{1}{4}\mathbf{m}^t B \mathbf{m} = \frac{1}{4}\mathbf{p}^t \tilde{B}\mathbf{p} \tag{3.5.7}$$

where

$$\tilde{B} = \left(\begin{array}{cc|c} 2 & -1 & 0 \\ -1 & 1 & 1 \\ \hline 0 & -1 & B \end{array}\right) \tag{3.5.8}$$

Using this the NS-sector character (3.5.6) can be rewritten as

$$\begin{aligned} \tilde{\chi}^{(p',2p+p')}_{s,2r+b}(q) = q^{-\frac{1}{8}(b-s)^2 - \mathcal{N}_{r,s} + k_{b,s}} & \sum_{\substack{\mathbf{p}\in\mathbb{Z}^{t_{n_0+1}+2} \\ p_i \equiv (\tilde{Q}_{\mathbf{u},\mathbf{v}})_i, i\geq 2}} q^{\frac{1}{4}\mathbf{p}^t\tilde{B}\mathbf{p} - \frac{1}{2}\tilde{A}_{\mathbf{u},\mathbf{v}}\mathbf{p}} \\ & \times \frac{1}{(q)_{p_2}} \prod_{j=1, j\neq 2}^{t_{n_0+1}+2} \begin{bmatrix} \frac{1}{2}(\mathcal{I}_{\tilde{B}}\mathbf{p} + \tilde{\mathbf{u}} + \tilde{\mathbf{v}})_j \\ p_j \end{bmatrix}_q' \end{aligned} \tag{3.5.9}$$

where $\mathcal{I}_{\tilde{B}} = 2I_{t_{n_0+1}+2} - \tilde{B}$,

$$\begin{aligned} \tilde{A}_{\mathbf{u},\mathbf{v}} &= (0, 0, A_{\mathbf{u},\mathbf{v}}), \\ \tilde{\mathbf{u}}^t &= (0, 0, \mathbf{u}^t), \\ \tilde{\mathbf{v}}^t &= (0, 0, \mathbf{v}^t), \\ \tilde{Q}^t_{\mathbf{u}+\mathbf{v}} &= (0, 0, Q^t_{\mathbf{u}+\mathbf{v}}). \end{aligned} \tag{3.5.10}$$

Similarly setting $m_0 = 2n + b - s$ in (3.5.4) and using

$$(-q)_{\frac{m_0-1}{2}} = \frac{1}{2}\sum_{k=0}^{\frac{m_0+1}{2}} q^{\frac{1}{2}(\frac{m_0+1}{2}-k)(\frac{m_0-1}{2}-k)} \begin{bmatrix} \frac{m_0+1}{2} \\ k \end{bmatrix}_q \tag{3.5.11}$$

we get the fermionic formula in the R-sector,

$$\tilde{\chi}^{(p',2p+p')}_{s,2r+b}(q) = \frac{1}{2} q^{-\frac{1}{8}((b-s)^2+1)-\mathcal{N}_{r,s}+k_{b,s}} \sum_{\substack{\mathbf{p}\in\mathbb{Z}^{t_{n_0+1}+2} \\ p_i \equiv (\tilde{Q}_{\mathbf{u},\mathbf{v}})_i, i\geq 2}} q^{\frac{1}{4}\mathbf{p}^t \tilde{B}\mathbf{p} - \frac{1}{2}\tilde{A}_{\mathbf{u},\mathbf{v}}\mathbf{p}}$$
$$\times \frac{1}{(q)_{p_2}} \prod_{j=1, j\neq 2}^{t_{n_0+1}+2} \begin{bmatrix} \frac{1}{2}(\mathcal{I}_{\tilde{B}}\mathbf{p} + \tilde{\mathbf{u}} + \tilde{\mathbf{v}})_j \\ p_j \end{bmatrix}'_q \tag{3.5.12}$$

where $\tilde{B}$, $\tilde{A}$, $\tilde{\mathbf{v}}$ are as in (3.5.10) and $\tilde{\mathbf{u}}^t = (1, 0, \mathbf{u}^t)$, $\tilde{Q}^t_{\mathbf{u}+\mathbf{v}} = (0, 1, Q^t_{\mathbf{u}+\mathbf{v}})$.

Fermionic formula for $SM(2p + p', p')$ with $p' > 2p$: In this section we will just state the formulas without showing the calculations. The calculation is very similar to the previous case.

Using (3.4.16) in NS sector we get,

$$\tilde{\chi}^{(2p+p',p')}_{2r+b,s}(q) = q^{-\frac{1}{8}(b-s)^2-\mathcal{N}_{r,s}+C_{b,s}} \sum_{\mathbf{p}\equiv\hat{\tilde{Q}}_{\mathbf{u}}} q^{\frac{1}{4}\mathbf{p}^t T\mathbf{p} - \frac{1}{2}\hat{A}^t\mathbf{p}} \times \frac{1}{(q)_{p_2}} \begin{bmatrix} \frac{p_2}{2} \\ p_1 \end{bmatrix}_q$$
$$\times \prod_{j=3}^{t_{n_0+1}+2} \begin{bmatrix} \frac{1}{2}(\mathcal{I}_T\mathbf{p} + \hat{B}\hat{\mathbf{u}}_+ + \hat{\mathbf{e}}_{\nu_0+1}(\hat{\mathbf{u}}^t_+.\hat{V}) + \hat{\mathbf{u}}_-)_j \\ p_j \end{bmatrix}'_q \tag{3.5.13}$$

where

$$
\begin{aligned}
\mathbf{p} &= (k, m_0, \tilde{\mathbf{m}}) \in Z^{t_{n_o+1}+2} \\
T &= \left(\begin{array}{cc|ccccccccc}
2 & -1 & 0 & . & . & . & . & . & . & . & 0 \\
-1 & 1 & 2 & . & . & 2 & 1 & 0 & . & . & 0 \\
\hline
0 & -2 & & & & & & & & & \\
. & . & & & & & & & & & \\
. & . & & & & & & & & & \\
. & -2 & & & & & 2B & & & & \\
. & -1 & & & & & & & & & \\
. & 0 & & & & & & & & & \\
. & . & & & & & & & & & \\
. & . & & & & & & & & & \\
0 & 0 & & & & & & & & &
\end{array}\right) \\
\mathcal{I}_T &= 2I_{t_{n_o+1}+2} - T \\
\hat{X}^t &= (0, 0, X^t) \text{ for } X \in Z^{t_{n_o+1}} \\
\hat{B} &= \left(\begin{array}{cc|c}
0 & 0 & 0 \\
0 & 0 & 0 \\
\hline
0 & 0 & B
\end{array}\right) \\
\mathbf{p} \equiv \hat{\overline{Q}}_{\mathbf{u}} &:= \begin{cases}
0 \le k \le \frac{m_0}{2} \\
m_0 \ge 0, m_0 \text{ is even} \\
\tilde{\mathbf{m}} \equiv \overline{Q}_{\mathbf{u}}
\end{cases}
\end{aligned}
$$

Note T is a $(t_{n_o+1} + 2 \times t_{n_o+1} + 2)$ matrix, number of 2 in second row is ν_0 and number of

-2 in second column is ν_0.

In R sector we get,

$$\tilde{\chi}^{(2p+p',p')}_{2r+b,s}(q) = q^{-\frac{1}{8}((b-s)^2+1)-\mathcal{N}_{r,s}+C_{b,s}} \sum_{\mathbf{p}\equiv\hat{\overline{Q}}_{\mathbf{u}}} q^{\frac{1}{4}\mathbf{p}^tT\mathbf{p}-\frac{1}{2}\hat{A}^t\mathbf{p}} \times \frac{1}{(q)_{p_2}} \begin{bmatrix} \frac{p_2+1}{2} \\ p_1 \end{bmatrix}_q \times \prod_{j=3}^{t_{n_0+1}+2} \begin{bmatrix} \frac{1}{2}(\mathcal{I}_T\mathbf{p} + \frac{\hat{B}}{2}\hat{\mathbf{u}}_+ + \frac{1}{2}\hat{\mathbf{e}}_{\nu_0+1}(\hat{\mathbf{u}}^t_+.\hat{V}) + \frac{1}{2}\hat{\mathbf{u}}_-)_j \\ p_j \end{bmatrix}'_q \tag{3.5.14}$$

where T, $\hat{A}$, $\hat{\mathbf{u}}$, $\hat{B}$ are as in (3.5.14) and

$$\mathbf{p} \equiv \hat{\overline{Q}}_{\mathbf{u}} := \begin{cases} 0 \leq k \leq \frac{m_0+1}{2} \\ m_0 \geq 0, m_0 \text{ is odd} \\ \tilde{\mathbf{m}} \equiv \overline{Q}_{\mathbf{u}} \end{cases}$$

3.5.2 The model $SM(p', 3p'-2p)$

Similarly using the same specialization with the dual Bailey pair in (3.3.6) and comparing the bosonic side with (3.5.1) with we find for $b-s$ even in the NS-sector

$$\tilde{\chi}^{(p',3p'-2p)}_{s,3b-2r}(q) = \sum_{n\geq 0} \frac{q^{\frac{3n}{2}(n+b-s)}(-q^{\frac{1}{2}})_{n+(b-s)/2}}{(q)_{2n+b-s}} q^{\mathcal{N}_{r,s}} F^{(p,p')}_{r,s}(2n+b-s, b; q^{-1}) \tag{3.5.15}$$

and for $b-s$ odd in the R-sector

$$\tilde{\chi}^{(p',3p'-2p)}_{s,3b-2r}(q) = \sum_{n\geq 0} \frac{q^{\frac{3n}{2}(n+b-s)}(-q)_{n+(b-s-1)/2}}{(q)_{2n+b-s}} q^{\mathcal{N}_{r,s}} F^{(p,p')}_{r,s}(2n+b-s, b; q^{-1}). \tag{3.5.16}$$

Fermionic formula for $SM(p', 3p' - 2p)$ with $p < p' < 2p$: To obtain the fermionic formula, as before we are going to set $m_0 = 2n + b - s$. Inserting (3.5.11) and (3.4.6) into (3.5.16) we get in the R-sector

$$\begin{aligned}\tilde{\chi}_{s,3b-2r}^{(p',3p'-2p)}(q) = \frac{1}{2} q^{-\frac{1}{8}(3(b-s)^2+1)+\mathcal{N}_{r,s}-k_{b,s}} & \sum_{\substack{m_0=0 \\ m_0 \text{ odd}}}^{\infty} \sum_{k=0}^{\frac{m_0+1}{2}} \sum_{\mathbf{m} \equiv Q_{\mathbf{u}+\mathbf{v}}} \\ & \times q^{\frac{1}{2}(m_0^2+k^2-m_0k-m_0m_1)} q^{\frac{1}{4}\mathbf{m}^t B \mathbf{m} - \frac{1}{2}\mathbf{m}^t(\mathbf{u}+\mathbf{v}) + \frac{1}{2}A_{\mathbf{u},\mathbf{v}}\mathbf{m}} \\ & \times \frac{1}{(q)_{m_0}} \begin{bmatrix} \frac{m_0+1}{2} \\ k \end{bmatrix}_q \prod_{j=1}^{t_{n_0+1}} \begin{bmatrix} n_j + m_j \\ m_j \end{bmatrix}_q' .\end{aligned} \tag{3.5.17}$$

Define $\mathbf{p} = (k, m_0, \mathbf{m}) \in \mathbb{Z}^{t_{n_0+1}+2}$, so that (3.5.17) in the R-sector can be rewritten as

$$\begin{aligned}\tilde{\chi}_{s,3b-2r}^{(p',3p'-2p)}(q) = \frac{1}{2} q^{-\frac{1}{8}(3(b-s)^2+1)+\mathcal{N}_{r,s}-k_{b,s}} & \sum_{\substack{\mathbf{p} \in \mathbb{Z}^{t_{n_0+1}+2} \\ p_i \equiv (\tilde{Q}'_{\mathbf{u}+\mathbf{v}})_i, i \geq 2}} q^{\frac{1}{4}\mathbf{p}^t \tilde{B}' \mathbf{p} + \frac{1}{2}\tilde{A}_{\mathbf{u},\mathbf{v}}\mathbf{p}} \\ & \times \frac{1}{(q)_{p_2}} \prod_{j=1, j\neq 2}^{t_{n_0+1}+2} \begin{bmatrix} \frac{1}{2}(\mathcal{I}_{\tilde{B}'}\mathbf{p} + \tilde{\mathbf{u}} + \tilde{\mathbf{v}})_j \\ p_j \end{bmatrix}_q' \end{aligned} \tag{3.5.18}$$

where $\mathcal{I}_{\tilde{B}'}=2I_{t_{n_0+1}+2}-\tilde{B}'$, $\tilde{\mathbf{v}}$ as in (3.5.10), $\tilde{\mathbf{u}}^t=(1,0,\mathbf{u}^t)$, $(\tilde{Q}'_{\mathbf{u}+\mathbf{v}})^t=(0,1,Q^t_{\mathbf{u}+\mathbf{v}})$, and

$$\tilde{B}'=\left(\begin{array}{cc|c} 2 & -1 & 0 \\ -1 & 2 & -1 \\ \hline 0 & -1 & B \end{array}\right) \tag{3.5.19}$$
$$\tilde{A}_{\mathbf{u},\mathbf{v}}=(0,0,A_{\mathbf{u},\mathbf{v}}-\mathbf{u}^t-\mathbf{v}^t).$$

Similarly, for the NS-sector it follows from (3.5.15)

$$\tilde{\chi}^{(p',3p'-2p)}_{s,3b-2r}(q)=q^{-\frac{3}{8}(b-s)^2+\mathcal{N}_{r,s}-k_{b,s}}\sum_{\substack{\mathbf{p}\in\mathbb{Z}^{t_{n_0+1}+2}\\ p_i\equiv(\tilde{Q}'_{\mathbf{u}+\mathbf{v}})_i,i\geq 2}} q^{\frac{1}{4}\mathbf{p}^t\tilde{B}'\mathbf{p}+\frac{1}{2}\tilde{A}_{\mathbf{u},\mathbf{v}}\mathbf{p}}$$
$$\times\frac{1}{(q)_{p_2}}\prod_{j=1,j\neq 2}^{t_{n_0+1}+2}\left[\begin{array}{c}\frac{1}{2}(\mathcal{I}_{\tilde{B}'}\mathbf{p}+\tilde{\mathbf{u}}+\tilde{\mathbf{v}})_j\\ p_j\end{array}\right]'_q \tag{3.5.20}$$

with $\tilde{B}'$ and $\tilde{A}_{\mathbf{u},\mathbf{v}}$ as in (3.5.19), $(\tilde{Q}'_{\mathbf{u}+\mathbf{v}})^t=(0,0,Q^t_{\mathbf{u}+\mathbf{v}})$, $\tilde{\mathbf{u}}^t=(0,0,\mathbf{u}^t)$ and $\tilde{\mathbf{v}}^t=(0,0,\mathbf{v}^t)$.

Fermionic formula for $SM(3p'-2p,p')$ with $p'>2p$: Again we state the formulas without showing the calculations.

Using (3.4.18) in NS sector we get,

$$\tilde{\chi}^{(3p'-2p,p')}_{3b-2r,s}(q) = q^{-\frac{3}{8}(b-s)^2+\mathcal{N}_{r,s}-C_{b,s}} \sum_{\mathbf{p}\equiv\hat{\tilde{Q}}_{\mathbf{u}}} q^{\frac{1}{4}\mathbf{p}^t T'\mathbf{p}}$$
$$\times q^{-\hat{A}^t\mathbf{p}-\frac{1}{2}\mathbf{p}^t\hat{B}\hat{\mathbf{u}}_+-\frac{1}{2}\mathbf{p}^t\hat{\mathbf{e}}_{\nu_0+1}(\hat{\mathbf{u}}^t_+.\hat{V})-\frac{1}{2}\mathbf{p}^t\hat{\mathbf{u}}_-} \times \frac{1}{(q)_{p_2}} \begin{bmatrix} \frac{p_2}{2} \\ p_1 \end{bmatrix}_q \tag{3.5.21}$$
$$\times \prod_{j=3}^{t_{n_0+1}+2} \begin{bmatrix} \frac{1}{2}(\mathcal{I}_{T'}\mathbf{p}+\frac{\hat{B}}{2}\hat{\mathbf{u}}_+ + \frac{1}{2}\hat{\mathbf{e}}_{\nu_0+1}(\hat{\mathbf{u}}^t_+.\hat{V})+\frac{1}{2}\hat{\mathbf{u}}_-)_j \\ p_j \end{bmatrix}'_q$$

where

$$\mathbf{p} = (k, m_0, \tilde{\mathbf{m}}) \in Z^{t_{n_o+1}+2} \tag{3.5.22}$$

$$T' = \left(\begin{array}{cc|cccccccc} 2 & -1 & 0 & . & . & . & . & . & . & 0 \\ -1 & 1 & -2 & . & . & -2 & -1 & 0 & . & 0 \\ \hline 0 & -2 & & & & & & & & \\ . & . & & & & & & & & \\ . & . & & & & & & & & \\ . & -2 & & & & 2B & & & & \\ . & -1 & & & & & & & & \\ . & 0 & & & & & & & & \\ . & . & & & & & & & & \\ . & . & & & & & & & & \\ 0 & 0 & & & & & & & & \end{array}\right)$$

$$\mathcal{I}_{T'} = 2I_{t_{n_o+1}+2} - T'$$

In R sector we get,

$$\tilde{\chi}^{(3p'-2p,p')}_{3b-2r,s}(q) = q^{-\frac{3}{8}(b-s)^2+\mathcal{N}_{r,s}-C_{b,s}} \sum_{\mathbf{p}\equiv\hat{\overline{Q}}_{\mathbf{u}}} q^{\frac{1}{4}\mathbf{p}^t T'\mathbf{p}}$$
$$\times\, q^{-\hat{A}^t\mathbf{p}-\frac{1}{2}\mathbf{p}^t\hat{B}\hat{\mathbf{u}}_+-\frac{1}{2}\mathbf{p}^t\hat{\mathbf{e}}_{\nu_0+1}(\hat{\mathbf{u}}^t_+.\hat{V})-\frac{1}{2}\mathbf{p}^t\hat{\mathbf{u}}_-} \times \frac{1}{(q)_{p_2}} \begin{bmatrix} \frac{p_2+1}{2} \\ p_1 \end{bmatrix}_q \tag{3.5.23}$$
$$\times \prod_{j=3}^{t_{n_0+1}+2} \begin{bmatrix} \frac{1}{2}(\mathcal{I}_{T'}\mathbf{p}+\frac{\hat{B}}{2}\hat{\mathbf{u}}_+ + \frac{1}{2}\hat{\mathbf{e}}_{\nu_0+1}(\hat{\mathbf{u}}^t_+.\hat{V})+\frac{1}{2}\hat{\mathbf{u}}_-)_j \\ p_j \end{bmatrix}'_q$$

where T' is as in (3.5.22) and

$$\mathbf{p}\equiv\hat{\overline{Q}}_{\mathbf{u}} := \begin{cases} 0\le k\le \frac{m_0}{2} \\ m_0\ge 0, m_0 \text{ is even} \\ \tilde{\mathbf{m}}\equiv\overline{Q}_{\mathbf{u}} \end{cases}$$

3.6 $N = 2$ Character formulas

3.6.1 $N = 2$ superconformal algebra and Spectral flow

The $N = 2$ superconformal algebra $\mathcal{A}$ is the infinite dimensional Lie super algebra [21] with basis $L_n, T_n, G_r^{\pm}, C$ and (anti)-commutation relation given by

$$\begin{aligned}
[L_m, L_n] &= (m-n)L_{m+n} + \frac{C}{12}(m^3 - m)\delta_{m+n,0} \\
\left[L_m, G_r^{\pm}\right] &= (\frac{1}{2}m - r)G_{m+r}^{\pm} \\
[L_m, T_n] &= -nT_{m+n} \\
[T_m, T_n] &= \frac{1}{3}cm\delta_{m+n,0} \\
\left[T_m, G_r^{\pm}\right] &= \pm G_{m+r}^{\pm} \\
\{G_r^+, G_s^-\} &= 2L_{r+s} + (r-s)T_{r+s} + \frac{C}{3}(r^2 - \frac{1}{4})\delta_{r+s,0} \\
[L_m, C] = [T_n, C] &= \left[G_r^{\pm}, C\right] = 0 \\
\{G_r^+, G_s^+\} &= \{G_r^-, G_s^-\} = 0
\end{aligned}$$

where $n, m \in \mathbb{Z}$, but r, s are integers in R-sector and half-integer in NS-sector. The element C is the central element and its eigenvalue c is parametrized as $c = 3(1 - \frac{2p}{p'})$, where p, p' are relatively prime positive integers. Let us denote this algebra by $\mathcal{A}(p, p')$.

It was observed in [35, 72] that there exits a family of outer automorphisms $\alpha_\eta : \mathcal{A} \to \mathcal{A}$

which maps the $N = 2$ superconformal algebras to itself. These are explicitly given by

$$\begin{aligned}
\alpha_\eta(G_r^+) &= \hat{G}_r^+ = G_{r-\eta}^+ \\
\alpha_\eta(G_r^-) &= \hat{G}_r^- = G_{r+\eta}^- \\
\alpha_\eta(L_n) &= \hat{L}_n = L_n - \eta T_n + \frac{c}{6}\eta^2 \delta_{n,0} \\
\alpha_\eta(T_n) &= \hat{T}_n = T_n - \frac{c}{3}\eta\delta_{n,0}
\end{aligned} \tag{3.6.1}$$

This family of automorphisms is called **spectral flow** and $\eta \in \mathbb{R}$ is called the **flow parameter**. When $\eta \in \mathbb{Z}$ each sector of the algebra is mapped to itself. When $\eta \in \mathbb{Z} + \frac{1}{2}$ the Neveu-Schwarz sector is mapped to the Ramond sector and vice-versa. We are going to use the spectral flow $\eta = \pm\frac{1}{2}$ to map the NS-sector to the R-sector.

3.6.2 Spectral flow and characters

We denote the Verma module generated from a highest weight state $|h, Q, c\rangle$ with L_0 eigenvalue h, T_0 eigenvalue Q and central charge c by $V_{h,Q}$. The character $\chi_{V_{h,Q}}$ of a highest weight representation $V_{h,Q}$ is defined as

$$\chi_{V_{h,Q}}(q, z) = \mathrm{Tr}_{V_{h,Q}}(q^{L_0 - c/24} z^{T_0}).$$

Following [35] the character transforms under the spectral flow in the following way

$$\mathrm{Tr}_{V_{h,Q}}(q^{\hat{L}_0 - c/24} z^{\hat{T}_0}) = \mathrm{Tr}_{V_{h^\eta, Q^\eta}}(q^{L_0 - c/24} z^{T_0}), \tag{3.6.2}$$

where h^η and Q^η are the eigenvalues of $\hat{L}_0$ and $\hat{T}_0$, respectively, as defined in (3.6.1). This means the new character $\chi_{V_{h^\eta, Q^\eta}}(q, z)$ which is the trace of the transformed operators over the original representation equals the character of the representation defined by the eigenvalues h^η and Q^η of $\hat{L}_0$ and $\hat{T}_0$, respectively. So the new character is the character of the representation V_{h^η,Q^η}.

For $\eta = \frac{1}{2}$ the spectral flow $\alpha_{\frac{1}{2}}$ takes a NS-sector character to an R-sector character. Let $\chi^{NS}_{V_{h,Q}}(q,z)$ be a NS-sector character corresponding to the representation $V_{h,Q}$. Then by (3.6.2) and (3.6.1) the new R-sector character $\chi^{R}_{V_{h^\eta,Q^\eta}}(q,z)$ is derived using

$$\begin{aligned}\chi^{R}_{V_{h^\eta,Q^\eta}}(q,z) &= \mathrm{Tr}_{V_{h,Q}}(q^{\hat{L}_0-c/24}z^{\hat{T}_0}) = \mathrm{Tr}_{V_{h,Q}}(q^{L_0-\frac{1}{2}T_0+\frac{c}{24}-\frac{c}{24}}z^{T_0-\frac{c}{6}})\\ &= q^{\frac{c}{24}}z^{-\frac{c}{6}}\mathrm{Tr}_{V_{h,Q}}(q^{L_0-\frac{c}{24}}(zq^{-\frac{1}{2}})^{T_0}) = q^{\frac{c}{24}}z^{-\frac{c}{6}}\chi^{NS}_{V_{h,Q}}(q,zq^{-\frac{1}{2}}).\end{aligned} \tag{3.6.3}$$

3.6.3 R-sector character from NS-sector character

To simplify notation we are going to use a slightly different notation for characters. Since we are only dealing with the vacuum character in the NS-sector for which $h = 0, Q = 0$, we write $\hat{\chi}^{NS}_{p,p'}(q,z)$. The R-sector character is denoted by $\hat{\chi}^{R}_{p,p'}(q,z)$ with the corresponding (h,Q) specified separately.

Following [20, 21, 35, 36] the vacuum character for the $N = 2$ superconformal algebra with central element $c = 3(1-\frac{2p}{p'})$ in the NS-sector is given by

$$\begin{aligned}\hat{\chi}^{NS}_{p,p'}(q,z) &= q^{-c/24}\prod_{n=1}^{\infty}\frac{(1+zq^{n-\frac{1}{2}})(1+z^{-1}q^{n-\frac{1}{2}})}{(1-q^n)^2}\\ &\times\Big(1-\sum_{n=0}^{\infty}q^{p'(n+1)(p(n+1)-1)}+\frac{zq^{pn(p'n+1)+p'n+\frac{1}{2}}}{1+zq^{pn+\frac{1}{2}}}+\frac{z^{-1}q^{pn(p'n+1)+p'n+\frac{1}{2}}}{1+z^{-1}q^{pn+\frac{1}{2}}}\\ &\qquad+\sum_{n=1}^{\infty}q^{p'n(pn+1)}+\frac{zq^{pn(p'n+1)-p'n-\frac{1}{2}}}{1+zq^{pn-\frac{1}{2}}}+\frac{z^{-1}q^{pn(p'n+1)-p'n-\frac{1}{2}}}{1+z^{-1}q^{pn-\frac{1}{2}}}\Big).\end{aligned} \tag{3.6.4}$$

This formula can be verified using the embedding diagram for the vacuum character as described in [21, 35] and can be rewritten as (as will be useful later)

$$\hat{\chi}^{NS}_{p,p'}(q,z) = q^{-c/24} \prod_{n=1}^{\infty} \frac{(1+zq^{n-\frac{1}{2}})(1+z^{-1}q^{n-\frac{1}{2}})}{(1-q^n)^2} \times \sum_{j=-\infty}^{\infty} q^{pj(p'j+1)} \frac{1-q^{2p'j+1}}{(1+zq^{p'j+\frac{1}{2}})(1+z^{-1}q^{p'j+\frac{1}{2}})}. \quad (3.6.5)$$

In particular if we put $z = 1$ in (3.6.5) we obtain the following formula derived in [21]

$$\hat{\chi}^{NS}_{p,p'}(q) = q^{-c/24} \prod_{n=1}^{\infty} \frac{(1+q^{n-\frac{1}{2}})^2}{(1-q^n)^2} \sum_{j=-\infty}^{\infty} q^{pj(p'j+1)} \frac{1-q^{p'j+\frac{1}{2}}}{1+q^{p'j+\frac{1}{2}}}. \quad (3.6.6)$$

Let us apply (3.6.3) to the NS-sector vacuum character (3.6.5) to get a Ramond sector character. From (3.6.1) it follows that

$$\hat{L}_0 = L_0 - \frac{1}{2}T_0 + \frac{c}{24}$$
$$\hat{T}_0 = T_0 - \frac{c}{6}.$$

For the vacuum character in the NS-sector $(h, Q) = (0, 0)$, so the new eigenvalues are $(h^\eta, Q^\eta) = (\frac{c}{24}, -\frac{c}{6})$ in the R-sector. Hence the new character in the R-sector corresponds to (h^η, Q^η) and by (3.6.3)

$$\hat{\chi}^{R}_{p,p'}(q,z) = q^{\frac{c}{24}} z^{-\frac{c}{6}} \hat{\chi}^{NS}_{p,p'}(q, zq^{-\frac{1}{2}}) = z^{-\frac{c}{6}} \frac{(-z)_\infty(-z^{-1}q)_\infty}{(q)^2_\infty} \sum_{j=-\infty}^{\infty} q^{pj(p'j+1)} \frac{1-q^{2p'j+1}}{(1+zq^{p'j})(1+z^{-1}q^{p'j+1})}. \quad (3.6.7)$$

3.6.4 Bailey flow from the minimal model $M(p, p')$ to $N = 2$ superconformal

We will consider two set of special values for r and b to find Bailey flows from the minimal model $M(p, p')$ to $N = 2$ superconformal models.

First we use $r = 0$ and $b = 1$ in (3.3.5) and we obtain

$$\sum_{n=0}^{\infty}(\rho_1)_n(\rho_2)_n(aq/\rho_1\rho_2)^n \frac{q^{-\mathcal{N}_{0,s}}}{(aq)_{2n}} F_{0,s}^{(p,p')}(2n+1-s+2x, 1; q)$$
$$= \frac{(aq/\rho_1)_\infty(aq/\rho_2)_\infty}{(aq)_\infty(aq/\rho_1\rho_2)_\infty} \sum_{j=-\infty}^{\infty}\left(\frac{(\rho_1)_{jp'-x}(\rho_2)_{jp'-x}}{(aq/\rho_1)_{jp'-x}(aq/\rho_2)_{jp'-x}}(aq/\rho_1\rho_2)^{jp'-x} \right.$$
$$\left. - \frac{(\rho_1)_{jp'-1-x}(\rho_2)_{jp'-1-x}}{(aq/\rho_1)_{jp'-1-x}(aq/\rho_2)_{jp'-1-x}}(aq/\rho_1\rho_2)^{jp'-1-x} \right) q^{jp(jp'-s)}. \quad (3.6.8)$$

Then we are going to assume $r(b) = b = 1$ in (3.3.6). This gives us

$$\sum_{n=0}^{\infty}(\rho_1)_n(\rho_2)_n(aq/\rho_1\rho_2)^n \frac{q^{\mathcal{N}_{1,s}}}{(aq)_{2n}} q^{n^2} F_{1,s}^{(p,p')}(2n+1-s+2x, 1; q^{-1})$$
$$= \frac{(aq/\rho_1)_\infty(aq/\rho_2)_\infty}{(aq)_\infty(aq/\rho_1\rho_2)_\infty} \sum_{j=-\infty}^{\infty}\left(\frac{(\rho_1)_{jp'-x}(\rho_2)_{jp'-x}}{(aq/\rho_1)_{jp'-x}(aq/\rho_2)_{jp'-x}}(aq/\rho_1\rho_2)^{jp'-x} \right.$$
$$\left. - \frac{(\rho_1)_{jp'-1-x}(\rho_2)_{jp'-1-x}}{(aq/\rho_1)_{jp'-1-x}(aq/\rho_2)_{jp'-1-x}}(aq/\rho_1\rho_2)^{jp'-1-x} \right) q^{j^2p'(p'-p)-js(p'-p)-x(1+x-s)}$$
(3.6.9)

In (3.6.8) and (3.6.9) we consider the specialization

$$\rho_1 = \text{finite}, \quad \rho_2 = \text{finite}.$$

so that $\frac{aq}{\rho_1\rho_2}\longrightarrow 1$. Taking the limit $\frac{aq}{\rho_1\rho_2}\longrightarrow 1$ in (3.6.8), we find

$$\sum_{n=0}^{\infty}(\rho_1)_n(\rho_2)_n\frac{q^{-\mathcal{N}_{0,s}}}{(aq)_{2n}}F_{0,s}^{(p,p')}(2n+1-s+2x,1;q)$$
$$=\frac{(\rho_1)_\infty(\rho_2)_\infty}{(\rho_1\rho_2)_\infty(q)_\infty}\sum_{j=-\infty}^{\infty}q^{jp(jp'-s)}\frac{\rho_1\rho_2q^{2(jp'-x-1)}-1}{(1-\rho_1q^{jp'-x-1})(1-\rho_2q^{jp'-x-1})}. \quad (3.6.10)$$

Similarly taking the limit $\frac{aq}{\rho_1\rho_2}\longrightarrow 1$ in (3.6.9), we find

$$\sum_{n=0}^{\infty}(\rho_1)_n(\rho_2)_n\frac{q^{\mathcal{N}_{1,s}}}{(aq)_{2n}}q^{n^2}F_{1,s}^{(p,p')}(2n+1-s+2x,1;q^{-1})$$
$$=q^{x(s-x-1)}\frac{(\rho_1)_\infty(\rho_2)_\infty}{(\rho_1\rho_2)_\infty(q)_\infty}\sum_{j=-\infty}^{\infty}q^{j^2p'(p'-p)-js(p'-p)}\frac{\rho_1\rho_2q^{2(jp'-x-1)}-1}{(1-\rho_1q^{jp'-x-1})(1-\rho_2q^{jp'-x-1})} \quad (3.6.11)$$

Now we will consider appropiate finite values for ρ_1 and ρ_2.

Remark 3.6.1. We like to mention that the fermionic formula in section 3.4 for $p<p'<2p$ is not valid for $r=b=1$ and the fermionic formula for $p'>2p$ is not valid for $r=0,b=1$. Hence we can calculate the fermionic side of (3.6.10) only for $p<p'<2p$ and the fermionic side of (3.6.11) only for $p'>2p$.

Here we set

$$\rho_1=-zq^{x+\frac{1}{2}} \quad \text{and} \quad \rho_2=-z^{-1}q^{x+\frac{1}{2}}. \quad (3.6.12)$$

3.6.5 Fermionic formula for $p < p' < 2p$

Let us use the specialization (3.6.12) in (3.6.10), which implies $a = q^{2x}$ and $s = 1$. Making the variable change $j \longrightarrow -j$ in (3.6.10) and setting $x = 0$ we obtain

$$\begin{aligned}\sum_{n=0}^{\infty}(-zq^{\frac{1}{2}})_n(-z^{-1}q^{\frac{1}{2}})_n\frac{q^{-\mathcal{N}_{0,1}}}{(q)_{2n}}F_{0,1}^{(p,p')}(2n,1;q)\\ =\frac{(-zq^{\frac{1}{2}})_\infty(-z^{-1}q^{\frac{1}{2}})_\infty}{(q)_\infty^2}\sum_{j=-\infty}^{\infty}q^{jp(jp'+1)}\frac{1-q^{2jp'+1}}{(1+zq^{jp'+\frac{1}{2}})(1+z^{-1}q^{jp'+\frac{1}{2}})}.\end{aligned} \quad (3.6.13)$$

Comparing with (3.6.5), we obtain

$$\hat{\chi}_{p,p'}^{NS}(q,z)=q^{-\frac{c}{24}-\mathcal{N}_{0,1}}\sum_{n=0}^{\infty}\frac{(-zq^{\frac{1}{2}})_n(-z^{-1}q^{\frac{1}{2}})_n}{(q)_{2n}}F_{0,1}^{(p,p')}(2n,1;q). \quad (3.6.14)$$

This gives us a Bailey flow from $M(p,p')$ model to $N = 2$ superconformal model in the NS sector. Now we calculate the fermionic side for $p < p' < 2p$.

Setting $z = 1$ and inserting the fermionic formula (3.4.3), we find

$$\begin{aligned}\hat{\chi}_{p,p'}^{NS}(q)=q^{-\frac{c}{24}-\mathcal{N}_{0,1}+k_{1,1}}\sum_{n=0}^{\infty}\Bigg(\frac{(-q^{\frac{1}{2}})_n^2}{(q)_{2n}}\sum_{\mathbf{m}\equiv Q_{\mathbf{u}+\mathbf{v}}}q^{\frac{1}{4}\mathbf{m}^tB\mathbf{m}-\frac{1}{2}A_{\mathbf{u},\mathbf{v}}\mathbf{m}}\\ \times\prod_{j=1}^{t_{n_0+1}}\begin{bmatrix}n_j+m_j\\ m_j\end{bmatrix}_q'\Bigg).\end{aligned} \quad (3.6.15)$$

Let us set $m_0 = 2n$ and use (3.5.5) to get

$$\hat{\chi}^{NS}_{p,p'}(q) = q^{-\frac{c}{24}-\mathcal{N}_{0,1}+k_{1,1}} \sum_{\substack{m_0=0\\ m_0 \text{ even}}}^{\infty} \sum_{k_1=0}^{\frac{m_0}{2}} \sum_{k_2=0}^{\frac{m_0}{2}} \sum_{\mathbf{m}\equiv Q_{\mathbf{u}+\mathbf{v}}} q^{\frac{1}{2}(\frac{m_0}{2}-k_1)^2+\frac{1}{2}(\frac{m_0}{2}-k_2)^2}$$
$$\times\, q^{\frac{1}{4}\mathbf{m}^t B\mathbf{m}-\frac{1}{2}A_{\mathbf{u},\mathbf{v}}\mathbf{m}} \frac{1}{(q)_{m_0}} \begin{bmatrix} \frac{m_0}{2} \\ k_1 \end{bmatrix}_q \begin{bmatrix} \frac{m_0}{2} \\ k_2 \end{bmatrix}_q \prod_{j=1}^{t_{n_0+1}} \begin{bmatrix} n_j+m_j \\ m_j \end{bmatrix}_q' . \quad (3.6.16)$$

Define $\mathbf{p} = (k_1, k_2, m_0, \mathbf{m}) \in \mathbb{Z}^{t_{n_0+1}+3}$, so that (3.6.16) can be rewritten as

$$\hat{\chi}^{NS}_{p,p'}(q) = q^{-\frac{c}{24}-\mathcal{N}_{0,1}+k_{1,1}} \sum_{\substack{\mathbf{p}\in\mathbb{Z}^{t_{n_0+1}+3}\\ p_i\equiv(\hat{Q}_{\mathbf{u},\mathbf{v}})_i, i\geq 3}} q^{\frac{1}{4}\mathbf{p}^t D\mathbf{p}-\frac{1}{2}\hat{A}_{\mathbf{u},\mathbf{v}}\mathbf{p}}$$
$$\times \frac{1}{(q)_{p_3}} \prod_{j=1, j\neq 3}^{t_{n_0+1}+3} \begin{bmatrix} \frac{1}{2}(\mathcal{I}_D\mathbf{p}+\hat{\mathbf{u}}+\hat{\mathbf{v}})_j \\ p_j \end{bmatrix}_q' , \quad (3.6.17)$$

where $\mathcal{I}_D = 2I_{t_{n_0+1}+3} - D$ and

$$D = \left(\begin{array}{ccc|c} 2 & 0 & -1 & 0 \\ 0 & 2 & -1 & 0 \\ -1 & -1 & 1 & 1 \\ \hline 0 & 0 & -1 & B \end{array}\right),$$
$$\hat{A}_{\mathbf{u},\mathbf{v}} = (0,0,0,A_{\mathbf{u},\mathbf{v}}),$$
$$\hat{\mathbf{u}}^t = (0,0,0,\mathbf{u}^t),$$
$$\hat{\mathbf{v}}^t = (0,0,0,\mathbf{v}^t),$$
$$\hat{Q}^t_{\mathbf{u},\mathbf{v}} = (0,0,0,Q^t_{\mathbf{u}+\mathbf{v}}). \quad (3.6.18)$$

This gives a new fermionic expression for the NS-sector character.

Ramond sector characters: Let us set $\rho_1 = -zq^x, \rho_2 = -z^{-1}q^{x+1}$ in (3.6.11), which implies $a = q^{2x}$ and $s = 1$. Setting $x = 0$ and changing $j \longrightarrow -j$ we obtain

$$\sum_{n=0}^{\infty} \frac{(-z)_n(-z^{-1}q)_n}{(q)_{2n}} q^{-\mathcal{N}_{0,1}} F_{0,1}^{(p,p')}(2n, 1; q)$$
$$= \frac{(-z)_\infty(-z^{-1}q)_\infty}{(q)^2{}_\infty} \sum_{j=-\infty}^{\infty} q^{jp(jp'+1)} \frac{1 - q^{2jp'+1}}{(1 + zq^{jp'})(1 + z^{-1}q^{jp'+1})}. \quad (3.6.19)$$

Comparing with (3.6.7) we get

$$\hat{\chi}^R_{p,p'}(q, z) = z^{-\frac{c}{6}} q^{-\mathcal{N}_{0,1}} \sum_{n=0}^{\infty} \frac{(-z)_n(-z^{-1}q)_n}{(q)_{2n}} F_{0,1}^{(p,p')}(2n, 1; q). \quad (3.6.20)$$

This shows a Bailey flow from $M(p, p')$ to $N = 2$ superconformal algebra in the R-sector.

Again using (3.4.3) in a similar way to the NS-sector and setting $z = 1$ we find

$$\hat{\chi}^R_{p,p'}(q) = 2q^{-\mathcal{N}_{0,1}+k_{1,1}} \sum_{n=0}^{\infty} \left(\frac{(-q)_{n-1}(-q)_n}{(q)_{2n}} \sum_{\mathbf{m} \equiv Q_{\mathbf{u}+\mathbf{v}}} q^{\frac{1}{4}\mathbf{m}^t B \mathbf{m} - \frac{1}{2} A_{\mathbf{u},\mathbf{v}} \mathbf{m}} \right.$$
$$\left. \times \prod_{j=1}^{t_{n_0+1}} \begin{bmatrix} n_j + m_j \\ m_j \end{bmatrix}_q' \right). \quad (3.6.21)$$

Using

$$(x)_n = \sum_{k=0}^{n} (-x)^{(n-k)} q^{\frac{1}{2}(n-k)(n-k-1)} \begin{bmatrix} n \\ k \end{bmatrix}_q$$

and setting $m_0 = 2n$, equation (3.6.21) can be rewritten as

$$\hat{\chi}^R_{p,p'}(q) = 2q^{-\mathcal{N}_{0,1}+k_{1,1}} \sum_{\substack{m_0=0 \\ m_0 \text{ even}}}^{\infty} \sum_{k_1=0}^{\frac{m_0}{2}-1} \sum_{k_2=0}^{\frac{m_0}{2}} \sum_{\mathbf{m}\equiv Q_{\mathbf{u}+\mathbf{v}}} q^{\frac{1}{4}(m_0^2+2k_1^2+2k_2^2-2m_0k_1-2m_0k_2)}$$
$$\times\, q^{\frac{1}{4}\mathbf{m}^tB\mathbf{m}-\frac{1}{2}A_{\mathbf{u},\mathbf{v}}\mathbf{m}+\frac{1}{2}(k_1-k_2)} \frac{1}{(q)_{m_0}} \begin{bmatrix} \frac{m_0}{2}-1 \\ k_1 \end{bmatrix}_q \begin{bmatrix} \frac{m_0}{2} \\ k_2 \end{bmatrix}_q \prod_{j=1}^{t_{n_0+1}} \begin{bmatrix} n_j+m_j \\ m_j \end{bmatrix}_q{}' . \quad (3.6.22)$$

Setting $\mathbf{p} = (k_1, k_2, m_0, \mathbf{m}) \in \mathbb{Z}^{t_{n_0+1}+3}$ this becomes

$$\hat{\chi}^R_{p,p'}(q) = 2q^{-\mathcal{N}_{0,1}+k_{1,1}} \sum_{\substack{\mathbf{p}\in\mathbb{Z}^{t_{n_0+1}+3} \\ p_i\equiv(\hat{Q}_{\mathbf{u},\mathbf{v}})_i, i\geq 3}} q^{\frac{1}{4}\mathbf{p}^tD\mathbf{p}-\frac{1}{2}\hat{A}_{\mathbf{u},\mathbf{v}}\mathbf{p}}$$
$$\times \frac{1}{(q)_{p_3}} \prod_{j=1, j\neq 3}^{t_{n_0+1}+3} \begin{bmatrix} \frac{1}{2}(\mathcal{I}_D\mathbf{p}+\hat{\mathbf{u}}+\hat{\mathbf{v}})_j \\ p_j \end{bmatrix}_q{}', \quad (3.6.23)$$

with the same notations as in (3.6.18) except

$$\hat{A}_{\mathbf{u},\mathbf{v}} = (1, -1, 0, A_{\mathbf{u},\mathbf{v}}), \quad \hat{\mathbf{u}}^t = (-1, 0, 0, \mathbf{u}^t), \quad \hat{\mathbf{v}}^t = (-1, 0, 0, \mathbf{v}^t).$$

This gives a new fermionic expression of the new R-sector character.

3.6.6 Fermionic formula for $p' > 2p$

Now we use the same specialization (3.6.12) in (3.6.11). All the calculations are done in a similar way as in the previous section but using the fermionic formula for $p' > 2p$. Hence we just state the result here.

The explicit fermionic formula in this case looks like

$$\hat{\chi}^{NS}_{p'-p,p'}(q) = q^{-c(p'-p,p')+\mathcal{N}_{1,1}-C_{1,1}} \sum_{\mathbf{p}\equiv\hat{\bar{Q}}} \Bigg(q^{\frac{1}{4}\mathbf{p}^t D' \mathbf{p} - \hat{A}'\mathbf{p}} \times q^{-\frac{1}{2}\mathbf{p}^t \hat{B}' \hat{\mathbf{u}_+} - \frac{1}{2}\mathbf{p}^t \hat{\mathbf{e}}_{\nu_0+1}(\hat{\mathbf{u}}^t_+.\hat{V}) - \frac{1}{2}\mathbf{p}^t \hat{\mathbf{u}_-}} \times \frac{1}{(q)_{p_3}} \begin{bmatrix} \frac{p_3}{2} \\ p_1 \end{bmatrix}_q \begin{bmatrix} \frac{p_3}{2} \\ p_2 \end{bmatrix}_q \times \prod_{j=4}^{t_{n_0+1}+3} \begin{bmatrix} \frac{1}{2}(\mathcal{I}'_D \mathbf{p} + \hat{B}'\hat{\mathbf{u}}_+ + \hat{\mathbf{e}'}_{\nu_0+1}(\hat{\mathbf{u}'}^t_+.\hat{V}') + \hat{\mathbf{u}'}_-)_j \\ p_j \end{bmatrix}'_q \Bigg) \tag{3.6.24}$$

where

$$
\begin{aligned}
D' &= \left(\begin{array}{ccc|cccccccc}
2 & 0 & -1 & 0 & . & . & . & . & . & . & 0 \\
0 & 2 & -1 & 0 & . & . & . & . & . & . & 0 \\
-1 & -1 & 2 & -2 & . & . & -2 & -1 & 0 & . & 0 \\
\hline
0 & 0 & -2 & & & & & & & & \\
. & . & . & & & & & & & & \\
. & . & . & & & & & & & & \\
. & . & -2 & & & & 2B & & & & \\
. & . & -1 & & & & & & & & \\
. & . & 0 & & & & & & & & \\
. & . & . & & & & & & & & \\
. & . & . & & & & & & & & \\
0 & 0 & 0 & & & & & & & &
\end{array}\right) \\
\mathcal{I}_{D'} &= 2I_{t_{n_0+1}+3} - D' \\
\hat{A}' &= (0,0,0,A) \\
\hat{X}^t &= (0,0,0,X^t) \\
\mathbf{p} \equiv \overline{\hat{Q}'}_{\mathbf{u}} &:= \begin{cases} 0 \leq k_1 \leq \frac{m_0}{2} \\ 0 \leq k_2 \leq \frac{m_0}{2} \\ m_0 \geq 0, m_0 \text{ is even} \\ \tilde{\mathbf{m}} \equiv \overline{Q}_{\mathbf{u}} \end{cases}
\end{aligned} \tag{3.6.25}
$$

Note that D' is a $(t_{n_0+1}+3) \times (t_{n_0+1}+3)$ matrix and number of -2 in the third row and column is ν_0.

Ramond sector: In this case we state the formula only. We get

$$\begin{aligned}\hat{\chi}^{R}_{p'-p,p'}(q) =& 2q^{\mathcal{N}_{1,1}-C_{1,1}} \sum_{\mathbf{p}\equiv\hat{\overline{Q}}} \Bigg(q^{\frac{1}{4}\mathbf{p}^t D'\mathbf{p}-\hat{A'}^t\mathbf{p}} \\ & \times q^{-\frac{1}{2}\mathbf{p}^t\hat{B'}\hat{\mathbf{u}}_+ -\frac{1}{2}\mathbf{p}^t\hat{\mathbf{e}}_{\nu_0+1}(\hat{\mathbf{u}}^t_+.\hat{V})-\frac{1}{2}\mathbf{p}^t\hat{\mathbf{u}}_-} \times \frac{1}{(q)_{p_3}} \begin{bmatrix} \frac{p_3}{2}-1 \\ p_1 \end{bmatrix}_q \begin{bmatrix} \frac{p_3}{2} \\ p_2 \end{bmatrix}_q \\ & \times \prod_{j=4}^{t_{n_0+1}+3} \begin{bmatrix} \frac{1}{2}(\mathcal{I}_{D'}\mathbf{p}+\hat{B}'\hat{\mathbf{u}}_+ + \hat{\mathbf{e}'}_{\nu_0+1}(\hat{\mathbf{u}'}^t_+.\hat{V}') + \hat{\mathbf{u}'}_-)_j \\ p_j \end{bmatrix}'_q \Bigg)\end{aligned} \tag{3.6.26}$$

with the same notations as in (3.6.25) except

$$\mathbf{p} \equiv \hat{\overline{Q'}}_{\mathbf{u},\mathbf{v}} := \begin{cases} 0 \le k_1 \le \frac{m_0}{2}-1 \\ 0 \le k_2 \le \frac{m_0}{2} \\ m_0 \ge 0, m_0 \text{ is even} \\ \mathbf{m} \equiv \overline{Q}_{\mathbf{u},\mathbf{v}} \end{cases}$$

(3.6.23)and (3.6.26) gives us new fermionic expressions of the new R sector character.

3.7 Conclusion

In this thesis we only considered the vacuum character for the $N = 2$ superconformal algebra with central charge $c = 3(1 - \frac{2p}{p'})$ with $p < p'$ in the NS-sector and the Ramond sector character derived from the vacuum character. We believe that similar Bailey flows exist for the general $N = 2$ superconformal characters, but explicit formulas are not yet available in the literature.

We would also like to mention that unlike in section 3.5 we did not carry out the Bailey flow for both the Bailey pairs (3.3.3) and (3.3.4) in section 3.6. As we mentioned in section 3.6 the fermionic formulas $F_{r,s}^{(p,p')}(L,b;q)$ when $p < p' < 2p$ and $r = b = 1$ and when $p' > 2p$ and $r = 0, b = 1$ are not given in [11, 12]. We believe that a formula for these cases does appear in [84]. One can easily calculate the fermionic side in these cases using the formula given in [84] by similar calculations.

Chapter 4

Implementation

In this chapter we describe the programs that were used to verify the conjectures for our results on unrestricted Kostka polynomials presented in Chapter 2. The bijection Φ in Chapter 2 has been implemented as a program written in C++. Several different versions of this program have been used to carry out calculations regarding the unrestricted rigged configurations. We used six different programs to verify conjectures regarding the lower bound conditions, the convexity property of the unrestricted rigged configurations, and the fact that the bijection Φ preserves the statistics. We describe three of these programs in this chapter. The programs presented here can be used by anyone studying unrestricted Kostka polynomials. The code for these programs are provided in the appendix and can be downloaded at `http://math.ucdavis.edu/˜deka`.

The progams have also been incorporated into MuPAD-Combinat as a dynamic module by Francois Descouens [58]. For example, the command

```
riggedConfigurations::RcPathsEnergy::
         fromOnePath([[[3]],[[2],[1]],[[4,5,6],[1,2,3]]])
```

calculates $\Phi(b)$ with b as in Example 2.4.9.

The programs have been compiled and tested using the gnu c++ compiler, (g++), version 3.2.3.

4.1 Program: $allpaths\_bijection.c$

The program named $allpaths\_bijection.c$ performs the bijection Φ from the set of unrestricted paths $\mathcal{P}(B,\lambda)$ to the set of unrestricted rigged configurations $\mathrm{RC}(L,\lambda)$ for a fixed value of λ and L. Let us recall that λ is the weight vector for the unrestricted paths in $\mathcal{P}(B,\lambda)$ and L is the multiplicity matrix for the shape of the tensor product $B = B^{r_k,s_k} \otimes \cdots \otimes B^{r_1,s_1}$. Let us denote the shape of B by a sequence of rectangular partitions μ. This program first calculates the set $\mathcal{P}(B,\lambda)$. Then, for each $b \in \mathcal{P}(B,\lambda)$ it calculates $\Phi(b)$, thus calculates the set $\mathrm{RC}(L,\lambda)$. The program sorts the elements of $\mathrm{RC}(L,\lambda)$ so that all the rigged configurations with the same shape appear together. It also calculates the statistics for each pair $(b,\Phi(b))$ and finally prints out the unrestricted Kostka polynomial $X(\lambda,B)$.

Input: Here we explain how to input data for the program. The input file for this program is called $input\_allpaths$. The input data for this program are:

n = The rank of the Lie algebra of type A_n and we input it in the 1st line of the input file.

λ = The fixed weight of the unrestricted paths. λ is an $n+1$ tuple of non-negative numbers. We enter this in the 2nd line of the input file with exactly $n+1$ parts including 0 if necessary.

μ = The fixed shape of the paths. μ is a sequence of rectangular partitions. We enter a rectangular partition columnwise. For example, 2 2 2 0 represents a rectangular partition with 2 boxes in the 1st column, 2 boxes in the 2nd column and 2 boxes in the 3rd column. The 0 at the end indicates the end of that partition. We input each component of μ in a new line.

We illustrate how to enter n, λ and μ to the program using a small example given below.

$$n = 5, \quad \lambda = (1, 2, 2, 2, 10), \quad \mu = ((1, 1), (3, 3)).$$

The input file is:

```
5
1 2 2 2 1 0
2 0
2 2 2 0
```

WARNING: Do not leave any extra blank space at the end of a line. Do not forget to include 0 if necessary to make λ an $n+1$ tuple. Do not forget to put 0 at the end of a part of μ. The program will read the input data incorrectly if you forget any of these and you will get a wrong answer.

Remark 4.1.1. The maximum size of the rank n of the algebra A_n is limited by 'RIGSIZE' which is defined to be 20. For a larger n, the 'RIGSIZE' needs to be increased accordingly in the beginning of the program. For a very large n, the program might take longer to compile and run.

Output: Let us consider the input data: $n = 3, \lambda = (0, 1, 1, 1)$ and $\mu = ((1), (1, 1))$. Input file is

```
3
0 1 1 1
1 0
2 0
```

The output of the program for this example is shown below.

```
n = 3
Lambda is:  0 1 1 1
mu is :
..........
 1
..........
 2
---------------------------------------
There are 3 unrestricted paths.
---------------------------------------

Path (1):
```

```
---------------------------------------
 2
---------------------------------------
 3
 4

Corresponding rigged configuration is:
---------------------------------------
(1)
 ___ ___
|   | -1| -1
 --- ---
---------------------------------------
(2)
 ___
| -1| -1
 ---
| -1| -1
 ---
---------------------------------------
(3)
 ___
|  0| 0
 ---
Statistic is = 0
```

```
************************************************

Path (2):
---------------------------------------
 3
---------------------------------------
 2
 4

Corresponding rigged configuration is:
---------------------------------------
(1)
 ___ ___
|   | -1| -1
 --- ---
---------------------------------------
(2)
 ___ ___
|   |  0| 0
 --- ---
---------------------------------------
(3)
 ___
| -1| -1
 ---
```

```
Statistic is = 0
************************************************


Path (3):
---------------------------------------
 4
---------------------------------------
 2
 3


Corresponding rigged configuration is:
---------------------------------------
(1)
 ___
| -1| -1
 ---
| -1| -1
 ---
---------------------------------------
(2)
 ___
|  0| 0
 ---
|  0| 0
 ---
```

```
---------------------------------------

(3)

 ___

|  0| 0

 ---

Statistic is = 1

************************************************


Unrestricted Kostka polynomial is:  2q^0  + 1q^1
```

The output means there are 3 unrestricted paths. Each of the path and the corresponding rigged configuration are printed together along with the statistic. Finally the unrestricted Kostka polynomial, $X(B,\lambda) = 2 + q$.

4.2 Program: $one\_path\_bij.c$

The program called $one\_path\_bij.c$ calculates the image of an unrestricted path $b \in \mathcal{P}(B,\lambda)$ under the map Φ. The output is an unrestricted rigged configuration in $\mathrm{RC}(L,\lambda)$. The program also calculates the corresponding statistic.

Input: The input file for this program is called *inputpath*. We explain how to enter a path to the program using an example. Suppose we want to input the following path:

$b = \begin{array}{|c|c|c|}\hline 2 & 3 & 4 \\ \hline \end{array} \otimes \begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 2 & 3 & 4 \\ \hline 3 & 5 & 6 \\ \hline \end{array} \otimes \begin{array}{|c|}\hline 2 \\ \hline 4 \\ \hline \end{array} \otimes \begin{array}{|c|c|c|}\hline 2 & 4 & 5 \\ \hline 3 & 5 & 6 \\ \hline \end{array}$ for the algebra of type A_5.

This path has 4 parts and the rank $n = 5$. The input for this example will be

```
5 4              [First entry is n=5, 2nd is number of parts=4]
2 3 4           [This is the 1st part of the path]
```

```
0                [0 separates the parts]
1 2 3
2 3 4           [2nd part]
3 5 6
0                [0 separates the parts]
2                [3rd part]
4
0                [0 separates the parts]
2 4 5           [4th part of the path]
3 5 6
0                [0 to indicate the end of the path]
```

WARNING: Do not leave any extra blank space at the end of a line. The program will read the input incorrectly in that case and you will get a wrong answer.

Output: Now the output for our example is

```
n=5
Given path is:
---------------------------------------
 2  3  4
---------------------------------------
 1  2  3
 2  3  4
 3  5  6
---------------------------------------
 2
```

```
 4
---------------------------------------
 2  4  5
 3  5  6
---------------------------------------

Corresponding rigged configuration is :
---------------------------------------
(1)
 ___ ___ ___ ___ ___ ___ ___
|   |   |   |   |   |   |   | -4| -4
 --- --- --- --- --- --- ---
|   |   | -1| 0
 --- ---
---------------------------------------
(2)
 ___ ___ ___ ___ ___ ___
|   |   |   |   |   |   | -1| -1
 --- --- --- --- --- ---
|   |   |   |  0| 1
 --- --- ---
|   |   |  1| 1
 --- ---
---------------------------------------
(3)
```

```
 ___ ___ ___ ___
|   |   |   | -1| -1
 --- --- --- ---
|   |   |  0| 0
 --- --- ---
|   | -1| 0
 --- ---
---------------------------------------
(4)
 ___ ___ ___
|   |   | -1| 0
 --- --- ---
|   | -1| 0
 --- ---
---------------------------------------
(5)
 ___ ___
|   |  0| 0
 --- ---
---------------------------------------
Statistic = 5
```

For the path, the dotted lines separates the parts of the path. For the rigged configuration the program gives the component number for the rigged partition and separates the different components with dotted lines. In the end, the program gives the statistics corresponding to the path and the rigged configuration. We proved in Chapter 2 that the statistics for the path and the rigged configuration are preserved under the bijection.

4.3 Program: $inverse\_bijection.c$

The program called $inverse\_bijection.c$ computes the inverse map of Φ. It takes an unrestricted rigged configuration $(\nu, J) \in \mathrm{RC}(L, \lambda)$ as an input and finds the image under the inverse bijection. The output is a path in $\mathcal{P}(B, \lambda)$.

Input: Let us explain the input file with an example. The input file is called $inputrigged$ for this program. Suppose we want to find the image of the rigged configuration

$$(\nu, J) = \begin{array}{ll} \square\square\square\square & -1 \\ \square\square\square & 0 \end{array} \quad \begin{array}{ll} \square\square\square\square & 0 \\ \square & -1 \\ \square & -1 \end{array} \quad \begin{array}{ll} \square\square\square & -1 \\ \square & 0 \end{array} \quad \begin{array}{ll} \square\square & -1 \end{array}$$

with $n = 5$ and $\mu = ((1), (1), (3, 3), (2, 2, 2))$. To enter components of mu we use the dimension of the rectangular box. For example, the third component of μ in our example is entered as 2 3 indicating a 2 by 3 rectangle.

The input file for this example is

```
5 4         [n=5, 4 is the number of components in mu]
1 1         [first component of mu]
1 1         [2nd component of mu]
2 3         [3rd component of mu]
3 2         [4th component of mu]
4 1         [first rigged partition]
-1 0        [riggings for respective parts right below]
4 3         [2nd rigged partition]
```

```
-1 0       [riggings for respective parts]
4 1 1     [3rd rigged partition]
0 -1 -1   [riggings for respective parts]
3 1        [4th rigged partition]
-1 0       [riggings for respective parts]
2           [5th rigged partition]
-1          [riggings for respective parts]
```

WARNING: As in the previous cases, do not leave any extra blank space at the end of a line. The program will get confused and will give a wrong result.

Output: The output for our example is:

```
n = 5  L= 4
mu
1 1
1 1
2 3
3 2

Given rigged configuration is:
---------------------------------------
(1)
 ___ ___ ___ ___
|   |   |   | -1| -1
 --- --- --- ---
|  0| 0
```

```
 ---
---------------------------------------
(2)
 ___ ___ ___ ___
|   |   |   | -1| 0
 --- --- --- ---
|   |   |  0| 0
 --- --- ---
---------------------------------------
(3)
 ___ ___ ___ ___
|   |   |   |  0| 1
 --- --- --- ---
| -1| -1
 ---
| -1| -1
 ---
---------------------------------------
(4)
 ___ ___ ___
|   |   | -1| -1
 --- --- ---
|  0| 0
 ---
---------------------------------------
```

```
(5)
 ___ ___
|   | -1| -1
 --- ---
---------------------------------------
The corresponding path is:
-------------------------
 3
-------------------------
 2
-------------------------
 1 3 5
 2 4 6
-------------------------
 1 2
 3 4
 5 6
-------------------------
```

Note that the program first prints out the input data and then prints the image of the rigged configuration under the inverse map which is an unrestricted path. The different parts of the path are separated by dotted lines. The output of the above example is the path

$$\boxed{3} \otimes \boxed{2} \otimes \begin{array}{|c|c|c|}\hline 1&3&5\\\hline 2&4&6\\\hline\end{array} \otimes \begin{array}{|c|c|}\hline 1&2\\\hline 3&4\\\hline 5&6\\\hline\end{array}$$

.

.1 Code for $allpaths\_bijection.c$

This program computes the bijection for all unrestricted paths, finds all the unrestricted rigged configurations for a fixed λ and μ. It also calculates the statistics and the unrestricted Kostka polynomial corresponding to the λ and μ.

```
#include <stdio.h>
#define UNUSED 9999
#define RIGSIZE 20
int n, l, num_shapes;
int lambda[100];
int tab_shape[100];
int tableau[100][100];
int r, tab_indx, num_rc_lb_tab;
int *cum_lambda;
int *new_lambda;
int bigL [RIGSIZE][RIGSIZE];
int curL [RIGSIZE][RIGSIZE];
int path_index;
int tblu_index;
int num_paths;
int exp[1000];

FILE *fp;
class shape_class;
```

```
class tblu_row {
public:
    int *col;
    int num_col;
    tblu_row(int c);
    void print_row();
};
tblu_row::tblu_row(int c):num_col(c) {
    col = new int[c];
    for (int i=0; i < c; i++) col[i] = UNUSED;
}
void tblu_row::print_row() {
    if (col[0] == UNUSED) return;
    int i = 0;
    while (col[i] != UNUSED && i < num_col) {
       fprintf(stderr, "%2d ", col[i]);
       i++;
    }
    fprintf (stderr, "\n");
}
```

A doubly linked list of objects of type $tblu\_class$ makes up a path. Each object of type $tblu\_class$ represents a tableau which is a part of a path.

```
class tblu_class {
public:
```

```
    int tblu_id;
    tblu_row* row;
    int* tab_lambda;
    int num_row;
    tblu_class* next;
    tblu_class* prev;
    shape_class* tblu_shape; // pointer to the mu
         // from which we got the shape
    tblu_class(int r, int c);
    void print_tblu();
};
tblu_class::tblu_class(int r, int c):num_row(r) {
    row = new tblu_row [r](c);
    tab_lambda = new int[n+1];
    for (int i=0; i<=n; i++) tab_lambda[i] = 0;
    next = NULL;
    prev = NULL;
    tblu_shape = NULL;
}
```

Prints a tableau

```
void tblu_class::print_tblu(){
    fprintf (stderr,"------------------------\n");
    for (int i=0; i < num_row; i++) {
       row[i].print_row();
```

```
    }
}
tblu_class *tblu_list;
tblu_class *tblu_list_end;
typedef tblu_class* tblu_class_ptr;
tblu_class_ptr *tblu_array;
class shape_class {
public:
    int* shape;
    int num_col;
    int num_row;
    shape_class* prev;
    shape_class* next;
    tblu_class* first_tblu;
    shape_class(int ncol, int nrow);
};
shape_class::shape_class (int ncol, int nrow){
    num_col = ncol;
    num_row = nrow;
    shape = new int[ncol];
    for (int i=0; i<ncol; i++) shape[i]=nrow;
    first_tblu = NULL;
    prev = NULL;
    next = NULL;
}
```

```
shape_class* shape_list;
shape_class* shape_list_end;
```

An object of type path_class represents a path. A doubly linked list of objects of type path_class has all the unrestricted paths and the corresponding rigged configuration for each path.

```
class path_class {
public:
    int path_len;
    int index;
    int rigged[RIGSIZE][5][RIGSIZE];
              // the rigged set for this path
    tblu_class_ptr *path;
              // the array of pointers to tableaux
    path_class* next;
    path_class* prev;
    int cocharge;
    int Energy;
    path_class(int path_len);
    void print_path();
    void path_class::reset_flags(int pathi);
    void path_class::print_rigged_for_this_path();
    int path_class::
        find_largest_inside_outside_others
        (int index,int old_largest_index,int pathi);
```

```
int path_class::
    find_largest_inside_outside_first
    (int index, int pathi) ;
void path_class::add_new_col
         (int index, int pathi);
void path_class::init_unused_rigged
    (int index, int pathi);
void path_class::add_to_rigged
     (int index, int column_index, int pathi) ;
int path_class::num_box_1k_col
    (unsigned int i, int k, int pathi);
int path_class::add_box_to_rigged
    (int index, int begin,
    int old_largest_index, int pathi) ;
int path_class::second_func(int part_size,
    int rig_num);
void path_class::calc_outer_label
     (int rig_num, int pathi);
void path_class::calc_inner_label
     (int i, int pathi);
void path_class::insert_element_to_rigged
    (int num, int row_indx, int col_indx,
     int  nrow, int ncol);
void path_class::insert_tableau_to_rigged
    (tblu_class* cur_tblu);
```

```
    void path_class::build_rigged_for_path () ;
    void path_class::calculate_cocharge();
    int path_class::alpha(int k, int i);
};
path_class::path_class(int len):path_len(len) {
    path = new tblu_class_ptr [len];
    for (int i=0; i<len; i++) path[i] = NULL;
    next = NULL;
    prev = NULL;
    index=0;
    for (int i=0; i < RIGSIZE; i++) {
       for (int j = 0; j < 3; j++)
          for (int k = 0;  k < RIGSIZE; k++)
             rigged [i][j][k] = UNUSED;
       for (int j = 3; j < 5; j++)
          for (int k = 0;  k < RIGSIZE; k++)
             rigged [i][j][k] = 0;
    }
};
```

Prints a path.

```
void path_class::print_path() {
    for (int i=0; i <path_len; i++) {
        if (path[i] != NULL) path[i]->print_tblu();
    }
```

```
}
typedef path_class* path_class_ptr;
path_class_ptr tmp_path;
path_class_ptr path_list;
path_class_ptr path_list_end;
path_class_ptr *path_array;
int *print_order;

void reset_tableau() {
    for (int i=0;i<RIGSIZE; i++){
        for (int j=0; j<RIGSIZE;j++){
            tableau[i][j]=UNUSED;
        }
    }
}
void initialize_lambda() {
    int i, j, k,m;
    for (i=0; i < RIGSIZE; i++) {
        lambda[i] = -1;
       tab_shape[i]=-1;
    }
    for (i=0; i<1000; i++);
        exp[i]=0;
    reset_tableau();
}
```

Reads the input file.

```
void read_input(){
    int i, tmp;
    tmp = UNUSED;
    i = 0;
    l = 0;
    fp = fopen ("input_allpaths","rw");
    fscanf(fp,"%d\n", &n);
    while (i< n+1) {
        fscanf (fp, "%d", &tmp);
        lambda[i] = tmp;
        l = l + tmp;
        i++;
    }
}
```

Prints the input data: n, λ *and* μ.

```
void print_input() {
    int i;
    fprintf (stderr, "n = %d \n", n);
    fprintf (stderr, "Lambda is:  ");
    for (i=0; i <=n; i++) {
    if (lambda[i] == -1) break;
        fprintf (stderr, "%d ", lambda[i]);
    }
```

```
    fprintf (stderr, "\n");
}
```

Copies the tableau constructed to the tableau list.

```
void print_and_copy_tableau(int k, int nrow,
       int ncol, shape_class*  shape_obj ) {
    int i, j;
    i=0;
    tblu_class *my_tblu =
              new tblu_class(nrow,ncol);
    my_tblu->tblu_id = tblu_index;
    tblu_index += 1;
    while (tableau[i][0] != UNUSED){
       j=0;
       while ( tableau[i][j]!=UNUSED){
          my_tblu->row[i].col[j] = tableau[i][j];
          my_tblu->tab_lambda[tableau[i][j] - 1] =
          my_tblu->tab_lambda[tableau[i][j] - 1]
                                             + 1;
          j++;
       }
       i++;
    }
    if (shape_obj->first_tblu == NULL)
       shape_obj->first_tblu = my_tblu;
```

```
    my_tblu->tblu_shape = shape_obj;
    if (tblu_list == NULL){
        tblu_list = my_tblu;
        tblu_list_end = my_tblu;
    } else {
        tblu_list_end->next = my_tblu;
        my_tblu->prev = tblu_list_end;
        tblu_list_end = my_tblu;
    }
}
```

Builds a tableau recursively.

```
void build_tableau(int m,int row,int col, int nrow,
       int ncol, shape_class* shape_obj){
    int k,p,q,m1,h,h1,valid,num,a,b,c ;

    if ( (col == 0 && row == 0) ||
       ((col == 0) && (m > tableau[row - 1][col]) )
       || (row == 0 && m >= tableau[row][col-1]) ||
       (row > 0 && col > 0 && m >
          tableau[row-1][col] && m >=
          tableau[row][col-1]))
    {
        tableau[row][col]=m;
        if ( row == tab_shape[col]-1) {
```

```
          if ( tab_shape[col+1] < 1) {
             print_and_copy_tableau(tab_indx,
               nrow, ncol, shape_obj);
             tab_indx=tab_indx+1;
             return;
          }
          else{
              for ( k=1; k <= n+1; k++) {
                 build_tableau(k,0,
                  col+1,nrow,ncol, shape_obj);
              }
       }
       } else{
         for ( k=m+1; k<=n+1; k++) {
            build_tableau(k,row+1,col,nrow,
                  ncol, shape_obj);
          }
       }


   }
}
```

Finds all possible tableaux of a given shape.

```
void find_tableau(){
  int i,j,k,h,tmp;
```

```
int nrow, ncol;
tab_indx=0;num_rc_lb_tab=0;
num_shapes = 0;
for (i=0;i<RIGSIZE;i++){
  tab_shape[i]=0;
}
j = 0;
tblu_list = NULL;
tblu_list_end = NULL;
shape_list = NULL;
shape_list_end = NULL;
tblu_index = 0;
fprintf (stderr, "mu is :\n");
while (fscanf (fp, "%d", &tmp) != EOF){
   if (tmp == 0)  {
      k = 0;
      shape_class* my_shape_obj
        = new shape_class(k,tab_shape[0]);
      if (shape_list == NULL){
         shape_list = my_shape_obj;
         shape_list_end = my_shape_obj;
      } else {
         my_shape_obj->prev = shape_list_end;
         shape_list_end->next = my_shape_obj;
         shape_list_end = my_shape_obj;
```

```
        }

        fprintf (stderr, "..........\n");

        while (tab_shape[k] != 0) {
           fprintf (stderr, "%2d ", tab_shape[k]);
           k++;
        }
        num_shapes++;
        fprintf (stderr, "\n");
        nrow = tab_shape[0];
        ncol = k;
        reset_tableau();
        for (int i = 1; i <= n+1; i++){
           build_tableau(i, 0, 0,
             nrow, ncol, my_shape_obj);
        }
        for (int i = 0; i < RIGSIZE; i++)
           tab_shape[i] = 0;
        j = 0;
     } else {
        tab_shape[j] = tmp;
        j++;
     }
  }
```

```
}
```

Finds the possible parts (which are tableaux) of a path with the given μ and λ.

```
void find_path_element(int position, shape_class*
       cur_shape, tblu_class* cur_tblu) {

    if (cur_tblu == NULL) {
        return;
    }

    int *this_lambda = cur_tblu->tab_lambda;
    bool satisfy = true;
    for (int i=0; i<= n; i++) {
        new_lambda[i] =
               cum_lambda[i] + this_lambda[i];
        if (new_lambda[i] > lambda[i]) {
           satisfy = false;
           break;
        }
    }
    if (satisfy == false) {
       return;
    } else {
       for (int i=0; i<=n; i++)
          cum_lambda[i] = new_lambda[i];
```

```
       tmp_path->path[position] = cur_tblu;
       if (position == num_shapes - 1) {
          bool found= true;
          for (int i=0; i<=n; i++)
            if (cum_lambda[i] != lambda[i])
               found = false;
          if (found) {
               path_class* tmp_path_list =
                 new path_class (num_shapes);
               path_index=path_index+1;
               tmp_path_list -> index=path_index;
               for (int i=0; i<num_shapes; i++) {
                   tmp_path_list->path[i] =
                     tmp_path->path[i];
               }
               if (path_list == NULL) {
                  path_list = tmp_path_list;
                  path_list_end = tmp_path_list;
               } else {
                  path_list_end->next =
                                  tmp_path_list;
                  tmp_path_list->prev =
                                  path_list_end;
                  path_list_end = tmp_path_list;
               }
```

```
            }
        } else {
            tblu_class* this_tblu =
                 cur_shape->next->first_tblu;
            shape_class* this_shape =cur_shape->next;
            while (this_tblu != NULL &&
               this_tblu->tblu_shape == this_shape){
               find_path_element(position +1,
                  this_shape, this_tblu);
               this_tblu = this_tblu->next;
            }
        }
        for (int i=0; i<=n; i++) cum_lambda[i] =
            cum_lambda[i] - this_lambda[i];
        tmp_path->path[position] = NULL;
    }
}
```

Builds paths of shape μ and weight λ.

```
void build_paths() {
    cum_lambda = new int [n+1];
    new_lambda = new int [n+1];
    path_index = 0;
    for (int i = 0; i <=n; i++) {
        cum_lambda[i] = 0;
```

```
      new_lambda[i] = 0;
    }
    tmp_path = new path_class (num_shapes);
    path_list = NULL;
    path_list_end = NULL;
    tblu_class* this_tblu = shape_list->first_tblu;
    while (this_tblu != NULL &&
           this_tblu->tblu_shape == shape_list) {
      find_path_element (0, shape_list, this_tblu);
      this_tblu = this_tblu->next;
    }

    path_class* tmp_path_list = path_list;
    while (tmp_path_list != NULL) {

      tmp_path_list = tmp_path_list->next;
    }

}



void path_class::reset_flags (int pathi) {
   int i, k;
   for (i=0; i < RIGSIZE; i++) {
      for (k=0; k < RIGSIZE; k++) {
```

```
          rigged [i][3][k] = 0;
      }
   }
}
```

This finds the cocharge for a given rigged configuration.

```
int path_class::alpha(int k, int i){
       int num_coln,j;
       num_coln=0;
       if (k>=n) num_coln=0;
     else{
       for (j=0; j<RIGSIZE;j++){
          if (rigged[k][0][0]==UNUSED){
               num_coln=0 ;
               break;
          }
          else{
             if (rigged[k][0][j]!=UNUSED){
                  if (rigged[k][0][j]>=i+1){
                     num_coln=num_coln+1;
             }
       }
     }
        }
     }
```

```
       return num_coln;
}
void path_class::calculate_cocharge(){
  int k,j,i,sum,cosum;
  sum=0;
  for (k=0;k<=n-1;k++){
    if (rigged[k][0][0]!=UNUSED){
      for (i=0;i< rigged[k][0][0];i++){
      sum=sum+alpha(k,i)*(alpha(k,i)-alpha(k+1,i));
      }
    }
  }
  cosum=sum;
  for (k=0;k<=n-1;k++){
    for (j=0;j<RIGSIZE;j++){
      if (rigged[k][2][j]==UNUSED) break;
      if ( rigged[k][2][j]!=UNUSED){
        cosum=cosum + rigged[k][2][j];
      }
    }
  }
  cocharge=cosum;
  fprintf (stderr, "Statistic is = %d   \n", cosum );
  exp[cosum]=exp[cosum]+1;
}
```

Prints the rigged configuration corresponding to a path.

```
void path_class::print_rigged_for_this_path() {
   int i, j, k, l,a,b;

   for (i = 0; i < n ; i++) {
      if (rigged[i][0][0]==UNUSED) {
         fprintf (stderr,"-------------------\n");
         fprintf(stderr,"(%d)   Empty\n", i+1);
      } else {
         if (rigged[i][0][0] != UNUSED) {
            fprintf (stderr,"----------------\n");
            fprintf(stderr, "(%d)\n", i+1);

            j=0;
            if (rigged[i][0][j] != UNUSED)
               for (k=0; k <rigged[i][0][j]; k++)
                   fprintf (stderr, " ___");
            fprintf (stderr,"\n");
            while (rigged[i][0][j] !=UNUSED){
                k=rigged[i][0][j];
                for (l=0; l<k-1;l++){
                   fprintf (stderr, "|   ");
                }
                if (l==k-1) fprintf (stderr,
```

```
                    "| %2d",rigged[i][2][j]);

                fprintf (stderr,
                      "| %d\n",rigged[i][1][j]);
                for (l=0; l<k; l++){
                   fprintf (stderr, " ---");
                }
                fprintf (stderr,"\n");
                j++;
            }
         }
      }

   }
   calculate_cocharge();
   fprintf (stderr, "*********************\n");
   fprintf (stderr,"\n");
}
```

Finds the largest singular string in a rigged partition other than the first one which is bigger or equal to the string selected in the previous rigged partition.

```
int path_class::find_largest_inside_outside_others
         (int index, int old_largest_index,
          int pathi) {
   int i = 0;
```

```
    int largest_index = UNUSED;
    while ((rigged[index][0][i] != UNUSED) &&
            i < RIGSIZE) {
        if ((rigged[index][1][i] ==
                            rigged[index][2][i]) &&
            (rigged[index][1][i] != UNUSED) &&
            (rigged[index][0][i]
                                <= old_largest_index)) {
            largest_index = i;
            break;
        }
        i++;
    }
    return largest_index;
}
```

Finds the largest singular string in the first rigged partition

```
int path_class::find_largest_inside_outside_first
                  (int index, int pathi) {
    int i = 0;
    int largest_index = UNUSED;
    while ((rigged[index][0][i] != UNUSED) &&
                                i < RIGSIZE) {
        if ((rigged[index][1][i] ==
                    rigged[index][2][i]) &&
```

```
          (rigged[index][1][i] != UNUSED)) {
         largest_index = i;
         break;
      }
      i++;
   }
   return largest_index;
}
void path_class::add_new_col(int index, int pathi){
   int i=0;
   while (rigged[index][0][i] != UNUSED) i++;
   rigged[index][0][i] = 1;
   rigged [index][3][i] = 1;
}
void path_class::init_unused_rigged
                     (int index, int pathi) {
   rigged [index][0][0] = 1;
   rigged [index][3][0] = 1;
}
void path_class::add_to_rigged (int index,
               int column_index, int pathi) {
   rigged [index][0][column_index] += 1;
   rigged [index][3][column_index] = 1;
}
```

Calculates the number of boxes in the first k columns of a rigged partition.

```
int path_class::num_box_1k_col
          (unsigned int i, int k, int pathi){
   int j, l;
   int num_boxes = 0;
   for (l=1; l <= k; l++) {
      for (j = 0; j < RIGSIZE; j++){
         if (rigged[i][0][j] == UNUSED) break;
         if (rigged[i][0][j] >= l){
             num_boxes += 1;
         }
      }
   }
   return num_boxes;
}
```

This adds a box to a rigged partition while doing the bijection.

```
int path_class::add_box_to_rigged (int index,
     int begin, int old_largest_index, int pathi){
   int largest_index;
   if (index == begin) {
      largest_index =
         find_largest_inside_outside_first(index,
                                          pathi);
   } else {
```

```
        largest_index =
            find_largest_inside_outside_others(index,
                       old_largest_index, pathi);
    }
    if (largest_index == UNUSED) {
        if (rigged[index][0][0] == UNUSED)
            init_unused_rigged (index, pathi);
        else
            add_new_col (index, pathi);
        return 0;
    } else {
        add_to_rigged (index, largest_index, pathi);
        return (rigged [index][0][largest_index] - 1);
    }
}
```

This calculates the second function in the definition of vacancy numbers.

```
int path_class::second_func
        (int part_size, int rig_num) {
    int sum = 0;
    for (int i=1; i<RIGSIZE; i++) {
        if (curL[rig_num+1][i] != 0) {
            int minimum = part_size;
            if (i < part_size) minimum = i;
            sum =sum + minimum * curL[rig_num+1][i];
```

```
      }
   }
   return sum;
}
```

This calculates the vacancy numbers.

```
void path_class::calc_outer_label (int rig_num,
                                   int pathi){
   int part_num=0;
   int part_size, p;

   if (rig_num == 0)  {
      for (int part_num=0; part_num < RIGSIZE;
                                part_num++) {
         part_size = rigged[0][0][part_num];
         if(part_size == UNUSED) break;
         p = (-2*num_box_1k_col (0, part_size,
                                    pathi)) +
           (num_box_1k_col (1, part_size, pathi)) +
           second_func(part_size, rig_num);
         rigged [0][1][part_num] = p;
      }
   } else {
      for (part_num=0; part_num < RIGSIZE;
                                       rt_num++) {
```

```
            part_size = rigged[rig_num][0][part_num];
            if(part_size == UNUSED) break;
            p = -2*num_box_1k_col (rig_num, part_size,
                                              pathi) +
                num_box_1k_col (rig_num-1, part_size,
                                              pathi) +
                num_box_1k_col (rig_num+1, part_size,
                                              pathi) +
                second_func(part_size, rig_num);
            rigged [rig_num][1][part_num] = p;
        }

    }
}
```

This calculates the labels or the riggings.

```
void path_class::calc_inner_label
            (int i, int pathi) {
    int j,k;
    int tmp;
    for (j = 0; j < RIGSIZE; j++) {
        if (rigged[i][0][j] == UNUSED) break;
        if (rigged[i][3][j] == 1) {
            rigged [i][2][j] = rigged [i][1][j];
        }
```

```
    }
    for (j = 0; j < RIGSIZE; j++) {
        for (k = 1; k < RIGSIZE; k++) {
            if (rigged[i][0][k] == UNUSED) break;
            if (rigged[i][0][k] == rigged[i][0][k-1]
              && rigged[i][1][k] == rigged[i][1][k-1]
              && rigged[i][2][k-1] < rigged[i][2][k])
              {
               tmp = rigged[i][2][k-1];
               rigged[i][2][k-1] = rigged[i][2][k];
               rigged[i][2][k] = tmp;
            }
        }
    }
}
```

This inserts each element of a part in the path into the bijection

```
void path_class::insert_element_to_rigged
         (int num, int row_indx, int col_indx,
                         int  nrow, int ncol) {
    int old_largest_index;
    reset_flags(0);
    for (int i = (num-2); i >= row_indx; i--) {
        old_largest_index =
        add_box_to_rigged(i, num-2, old_largest_index, 0);
```

```
}
for (int i=0; i < RIGSIZE; i++)
   for (int j = 0; j < RIGSIZE; j++)
      curL[i][j] = bigL[i][j];
if (row_indx == nrow - 1) {
   curL [nrow][1] += 1;
   if (col_indx != ncol - 1) {
      curL [nrow][ncol - col_indx - 1] += 1;
   }
} else {
   curL [nrow][ncol - col_indx - 1] += 1;
   curL [row_indx + 1][1] += 1;
}
for (int i = 0; i < n; i++) {
   calc_outer_label (i, 0);
}
for (int i = 0; i < n; i++) {
   calc_inner_label (i, 0);
}
if (row_indx == nrow - 1) {
   for (int i=0; i < RIGSIZE; i++)
      for (int j = 0; j < RIGSIZE; j++)
         curL[i][j] = bigL[i][j];

   // update curL - this is different from above
```

```
      curL [nrow][ncol - col_indx] += 1;
      for (int i = 0; i < n; i++) {
         calc_outer_label (i, 0);
      }
   }
}
```

This inserts each part of a path into the bijection

```
void path_class::insert_tableau_to_rigged
         (tblu_class* cur_tblu) {
   int nrow = cur_tblu->num_row;
   int ncol = cur_tblu->row[0].num_col;
   for (int j = ncol-1; j >=0; j--) {
     for (int i = 0; i < nrow; i++) {
        insert_element_to_rigged
         (cur_tblu->row[i].col[j], i,j,nrow, ncol);
     }
   }
   bigL [nrow][ncol] = bigL [nrow][ncol] + 1;
   for (int i = 0; i < RIGSIZE; i++)
      for (int j = 0; j < RIGSIZE; j++)
         curL[i][j] = bigL[i][j];
}
```

This insert all the parts of a path to the bijection

```
void path_class::build_rigged_for_path () {
```

```
   int j;
   for (int i=0; i < RIGSIZE; i++) {
      for (j =0; j < RIGSIZE; j++)
         bigL[i][j] = 0;
      curL[i][j] = 0;
   }
   for (int i = path_len - 1; i >= 0; i--) {
      insert_tableau_to_rigged (path[i]);
   }
}
```

Sorts the rigged configurations in a order so that the configurations with the same shape appears together.

```
void sort_rigged(){
  int i,j,k,a,b,p,T,m,h,a1,b1;
  a=0;p=0;a1=0;

  for (int a=0; a < num_paths; a++) {
     if  (path_array[a]->rigged[0][4][0] != 0)
              continue;
     print_order[p] = a;
     p = p + 1;
     for (int i=a+1; i < num_paths; i++){
        if (path_array[i]->rigged[0][4][0] != 0)
           continue;
```

```
T = 0;
for (b=0;b<=n-1;b++){
// pick the b-th rig-element of
// a-th path with i-th path
   k=0;
   while(path_array[a]->rigged[b][0][k]
                           !=UNUSED) {
     if (path_array[i]->rigged[b][0][k] !=
        path_array[a]->rigged[b][0][k]){
          T=1;break;
     }
     if (path_array[i]->rigged[b][4][k]
                                !=0) {
        T=1;break;
     }
     k++;
   }
   // Make sure if we exited while
   // loop because a[..] unused
   if (path_array[a]->rigged[b][0][k]
       != path_array[i]->rigged[b][0][k])
      T = 1;
   // if unequal quit searching
   if (T == 1) break;
```

```
        }
        // if rig-element comparison failed
        // quit path comparison
        if (T==1) {
           continue;
        }
        print_order[p]=i;
        path_array[i]->rigged[0][4][0]=1;
        p=p+1;
     }   // inner for loop - i
  }       // outer for loop - a
}
```

Calculates the rigged configurations corresponding to each of the possible paths.

```
void build_rigged() {
    path_class* tmp_path_list = path_list;
    num_paths = 0;
    while (tmp_path_list != NULL) {
       tmp_path_list->build_rigged_for_path ();
       tmp_path_list = tmp_path_list->next;
       num_paths++;
    }
    path_array = new path_class_ptr [num_paths];
    print_order = new int [num_paths];
    for (int i=0; i<num_paths; i++) {
```

```
        print_order[i] = i;
    }
    tmp_path_list = path_list;
    while (tmp_path_list != NULL) {
        path_array[tmp_path_list->index - 1] =
                                tmp_path_list;
        tmp_path_list = tmp_path_list->next;
    }
    sort_rigged();
    fprintf(stderr,"-----------------------\n");
    fprintf(stderr,"There are %d unrestricted
                  paths.\n", num_paths);
    fprintf(stderr,"-----------------------\n");
    fprintf(stderr,"\n");
    for (int i=0; i<num_paths; i++) {
        fprintf (stderr, "Path (%d): \n",i+1);
        path_array[print_order[i]]->print_path();
        fprintf (stderr, "\nCorresponding rigged
                     configuration is:\n");
        path_array[print_order[i]]->
              print_rigged_for_this_path();


    }
    fprintf(stderr,
           "Unrestricted Kostka polynomial is: ");
```

```
    int begin=1;
    for (int i=0; i<1000; i++) {
       if (exp[i]!= 0) {
          if (!begin)
             fprintf(stderr, " + ");
          begin=0;
          fprintf(stderr, "%dqˆ%d ", exp[i],i);
       }
    }
    fprintf(stderr,"\n");
}
```

Main program.

```
int main() {
   int i;
   initialize_lambda();  // initialization.
   read_input();          // this reads the input.
   print_input();       // this prints the input.
   find_tableau();// this finds all the tableaux of
                  // all the shapes from mu
   build_paths(); // this finds all the paths for
                  // given lambda and mu
   build_rigged();// this calculates all the rigged
                // configurations via the bijection
                // and sorts them in the order so
```

```
                // that all the configurations
                // with the same shape appear together.
}
```

.2 Code for the program $one\_path\_bij.tex.c$

This program does the bijection from the set of paths to the set of rigged configurations. Input data is a single path and the program calculates the corresponding rigged configuration using the bijection. It also calcutales the statistics.

```
#include <stdio.h>
#define UNUSED 9999
#define RIGSIZE 50
int n,l;
int tab_shape[100];
int tableau[100][100];
int r;
int rigged[RIGSIZE][5][RIGSIZE];
int bigL [RIGSIZE][RIGSIZE];
int curL [RIGSIZE][RIGSIZE];
int path_index;
int tblu_index;
FILE *fp;
```

A doubly linked list of objects of type tblu_class makes up a path. Each object of type tblu_class represents a tableau which is a part of a path.

```
class tblu_class {
public:
    int tblu_id;
    int** tb; // 2-dimensional array of integers
                // holding the tableau
    int* tab_lambda;
    int num_row;
    int num_col;
    tblu_class* next;
    tblu_class* prev;
    tblu_class(int r, int c);
    void print_tblu();
};

tblu_class::tblu_class(int r, int c)
              :num_row(r),num_col(c) {
   tb = new int* [r];
   for (int i=0; i < r; i++) {
      tb[i] = new int [c];
      for (int j = 0; j < c; j++)
         tb [i][j] = UNUSED;
   }
```

```
    tab_lambda = new int[n+1];
    for (int i=0; i<=n; i++) tab_lambda[i] = 0;
    next = NULL;
    prev = NULL;
}
```

Prints a tableau.

```
void tblu_class::print_tblu(){
    fprintf (stderr, "-------------------------\n");
    for (int i=0; i < num_row; i++) {
        for (int j=0; j < num_col; j++)
            fprintf (stderr, "%2d ", tb[i][j]);
        fprintf (stderr, "\n");
    }
}


typedef tblu_class* tblu_class_ptr;
tblu_class_ptr *tblu_array;
```

An object of type path_class represents a path. In this program it has only one object and the corresponding rigged configuration.

```
class path_class {
public:
```

```
int path_len;
int index;
int rigged[RIGSIZE][5][RIGSIZE];
  // this is the rigged set for this path
tblu_class_ptr *path;  // this is the array
   // of pointers to tableaux
path_class* next;
path_class* prev;
int cocharge;
path_class(int path_len);
void print_path();
void path_class::reset_flags(int pathi);
void path_class::print_rigged_for_this_path();
int path_class::find_largest_inside_outside_others
    (int index, int old_largest_index, int pathi);
int path_class::find_largest_inside_outside_first
                              (int index, int pathi) ;
void path_class::add_new_col
       (int index, int pathi);
void path_class::init_unused_rigged
       (int index, int pathi);
void path_class::add_to_rigged (int index,
                  int column_index, int pathi) ;
int path_class::num_box_1k_col (unsigned int i,
                                 int k, int pathi);
```

```
    int path_class::add_box_to_rigged (int index,
           int begin, int old_largest_index,
           int pathi) ;
    int path_class::second_func
           (int part_size, int rig_num);
    void path_class::calc_outer_label
           (int rig_num, int pathi);
    void path_class::calc_inner_label
           (int i, int pathi);
    void path_class::insert_element_to_rigged
           (int num, int row_indx, int col_indx,
            int  nrow, int ncol);
    void path_class::insert_tableau_to_rigged
           (tblu_class* cur_tblu);
    void path_class::build_rigged_for_path () ;
    void path_class::calculate_cocharge();
    int path_class::alpha(int k, int i);

};



path_class::path_class(int len):path_len(len) {
    path = new tblu_class_ptr [len];
    for (int i=0; i<len; i++) path[i] = NULL;
        next = NULL;
```

```
        prev = NULL;
        index=0;
        for (int i=0; i < RIGSIZE; i++) {
            for (int j = 0; j < 3; j++)
                for (int k = 0;  k < RIGSIZE; k++)
                    rigged [i][j][k] = UNUSED;
            for (int j = 3; j < 5; j++)
                for (int k = 0;  k < RIGSIZE; k++)
                    rigged [i][j][k] = 0;
       }
};


void path_class::print_path() {
    fprintf (stderr, "Given path is:\n");
    for (int i=0; i <path_len; i++) {
        if (path[i] != NULL) path[i]->print_tblu();
    }
}


typedef path_class* path_class_ptr;
path_class_ptr input_path;
```

```
void reset_tableau() {
    for (int i=0;i<RIGSIZE; i++){
        for (int j=0; j<RIGSIZE;j++){
            tableau[i][j]=UNUSED;
        }
    }
}
```

This reads the input file. 1st number we input is ”n” which is the number of nodes in the Dynkin diagram of type A_n. 2nd number we input is the number of parts in the path, called path length. Then we input each part of the path which is a tableau, we seperate the parts by putting a ”0”. At the end of the last part we put a ”0” to ensure the end of the path.

```
void read_input(){

   int i, j, k, tmp, c;
   int path_len, path_index;
   int col;
   tmp = UNUSED;
   path_index = 0;
   i = 0;
   j = 0;
   k = 0;
   col = 0;
   l = 0;
```

```
  fp = fopen ("inputpath","rw");
  fscanf(fp,"%d %d\n", &n, &path_len);
  input_path = new path_class (path_len);
  reset_tableau();
  while (fscanf (fp, "%d", &tmp) != EOF) {
     if (tmp == 0) {
        tblu_class *my_tblu =
           new tblu_class(i,col);
        i = 0;
        my_tblu->tblu_id = tblu_index;
        tblu_index += 1;
        while (tableau[i][0] != UNUSED){

           j=0;
   while ( tableau[i][j]!=UNUSED){
      my_tblu->tb[i][j] = tableau[i][j];
      my_tblu->tab_lambda[tableau[i][j] - 1] =
               my_tblu->tab_lambda[tableau[i][j] - 1] + 1;
              j++;
   }
   i++;
}
        input_path->path[path_index] = my_tblu;
        reset_tableau ();
        path_index += 1;
```

```
 i = 0; j = 0;
        continue;
     }
     tableau[i][j] = tmp;
     j += 1;
     c = fgetc(fp);
     if (c == '\n') {
        i += 1;
        col = j;
        j = 0;
     }
  }
}



void path_class::reset_flags (int pathi) {
   int i, k;
   for (i=0; i < RIGSIZE; i++) {
      for (k=0; k < RIGSIZE; k++) {
         rigged [i][3][k] = 0;
      }
   }
}
```

This prints the rigged configuration obtained from the bijection and prints the corresponding statistic.

```
void path_class::print_rigged_for_this_path() {
   int i, j, k, l,a,b;
   for (i = 0; i < n; i++) {
      if (rigged[i][0][0]==UNUSED) {
         fprintf (stderr,"--------------------\n");
         fprintf(stderr,"(%d)   Empty\n", i+1);
      }
      else {
        if (rigged[i][0][0] != UNUSED) {
          fprintf (stderr,"------------------\n");
          fprintf(stderr, "(%d)\n", i+1);
          j=0;
          if (rigged[i][0][j] != UNUSED)
          for (k=0; k <rigged[i][0][j]; k++)
             fprintf (stderr, " ___");
          fprintf (stderr,"\n");
          while (rigged[i][0][j] !=UNUSED){
              k=rigged[i][0][j];
              for (l=0; l<k-1;l++){
                 fprintf (stderr, "|   ");
              }
              if (l==k-1) fprintf
```

```
                  (stderr, "| %2d",rigged[i][2][j]);

               fprintf (stderr,
                  "| %d\n",rigged[i][1][j]);
               for (l=0; l<k; l++){
                  fprintf (stderr, " ---");
               }
               fprintf (stderr,"\n");
               j++;
            }
         }
      }
   }
   fprintf (stderr,"--------------------------\n");
   calculate_cocharge();
}
```

Finds the largest singular string in a rigged partition other than the first one which is bigger or equal to the string selected in the previous rigged partition by δ.

```
int path_class::find_largest_inside_outside_others
   (int index, int old_largest_index, int pathi) {
   int i = 0;
   int largest_index = UNUSED;
   while ((rigged[index][0][i] != UNUSED) &&
```

```
      i < RIGSIZE) {
      if ((rigged[index][1][i] ==
                        rigged[index][2][i])
            && (rigged[index][1][i] != UNUSED) &&
         (rigged[index][0][i] <= old_largest_index)) {
         largest_index = i;
         break;
      }
      i++;
   }
   return largest_index;
}
```

Finds the largest singular string in the first rigged partition.

```
int path_class::find_largest_inside_outside_first
      (int index, int pathi) {
    int i = 0;
    int largest_index = UNUSED;
    while ((rigged[index][0][i] != UNUSED)
             && i < RIGSIZE) {
       if ((rigged[index][1][i] ==
            rigged[index][2][i]) &&
           (rigged[index][1][i] != UNUSED)) {
          largest_index = i;
```

```
          break;
       }
       i++;
    }
    return largest_index;
}

void path_class::add_new_col (int index, int pathi)
{
    int i=0;
    while (rigged[index][0][i] != UNUSED) i++;
    rigged[index][0][i] = 1;
    rigged [index][3][i] = 1;
}

void path_class::init_unused_rigged (int index,
      int pathi) {
    rigged [index][0][0] = 1;
    rigged [index][3][0] = 1;
}

void path_class::add_to_rigged (int index,
     int column_index, int pathi) {
    rigged [index][0][column_index] += 1;
    rigged [index][3][column_index] = 1;
```

```
}
```

Calculates the number of boxes in the first k columns of a rigged partition.

```
int path_class::num_box_1k_col (unsigned int i,
     int k, int pathi){
    int j, l;
    int num_boxes = 0;
    for (l=1; l <= k; l++) {
       for (j = 0; j < RIGSIZE; j++){
          if (rigged[i][0][j] == UNUSED) break;
          if (rigged[i][0][j] >= l){
              num_boxes += 1;
          }
       }
    }
    return num_boxes;
}
```

This adds a box to a rigged partition while doing the bijection.

```
int path_class::add_box_to_rigged (int index,
       int begin, int old_largest_index,
       int pathi) {
    int largest_index;
```

```
    if (index == begin) {
       largest_index =
         find_largest_inside_outside_first
                     (index, pathi);
    } else {
       largest_index =
          find_largest_inside_outside_others
            (index, old_largest_index, pathi);
    }
    if (largest_index == UNUSED) {
       if (rigged[index][0][0] == UNUSED)
          init_unused_rigged (index, pathi);
       else
          add_new_col (index, pathi);
       return 0;
    } else {
       add_to_rigged (index, largest_index, pathi);
       return
          (rigged [index][0][largest_index] - 1);
    }
}
```

This calculates the second function in the definition of vacancy numbers.

```
int path_class::second_func(int part_size,
```

```
                              int rig_num) {
    int sum = 0;
    for (int i=1; i<RIGSIZE; i++) {
        if (curL[rig_num+1][i] != 0) {
        int minimum = part_size;
        if (i < part_size) minimum = i;
            sum =sum + minimum * curL[rig_num+1][i];
        }
    }
    return sum;
}
```

This calculates the vacancy numbers.

```
void path_class::calc_outer_label
       (int rig_num, int pathi){
   int part_num=0;
   int part_size, p;

   if (rig_num == 0)  {
      for (int part_num=0; part_num < RIGSIZE;
            part_num++) {
         part_size = rigged[0][0][part_num];
         if(part_size == UNUSED) break;
         p = (-2*num_box_1k_col (0, part_size,
```

```
                                         pathi))
             + (num_box_1k_col (1, part_size,
                                         pathi))
             + second_func(part_size, rig_num);
         rigged [0][1][part_num] = p;
      }
   } else {
      for (part_num=0; part_num < RIGSIZE;
           part_num++) {
         part_size = rigged[rig_num][0][part_num];
         if(part_size == UNUSED) break;
         p = -2*num_box_1k_col (rig_num, part_size,
             pathi) + num_box_1k_col (rig_num-1,
             part_size, pathi) + num_box_1k_col
             (rig_num+1, part_size, pathi) +
             second_func(part_size, rig_num);
         rigged [rig_num][1][part_num] = p;
      }

   }
}
```

This calculates the labels or the riggings.

```
void path_class::calc_inner_label
```

```
    (int i, int pathi) {
  int j,k;
  int tmp;
  for (j = 0; j < RIGSIZE; j++) {
    if (rigged[i][0][j] == UNUSED) break;
    if (rigged[i][3][j] == 1) {
      rigged [i][2][j] = rigged [i][1][j];
    }
  }
  for (j = 0; j < RIGSIZE; j++) {
    for (k = 1; k < RIGSIZE; k++) {
      if (rigged[i][0][k] == UNUSED) break;
      if (rigged[i][0][k] == rigged[i][0][k-1]
        && rigged[i][1][k] == rigged[i][1][k-1]
        && rigged[i][2][k-1] < rigged[i][2][k]){
        tmp = rigged[i][2][k-1];
        rigged[i][2][k-1] = rigged[i][2][k];
        rigged[i][2][k] = tmp;
      }
    }
  }
}
```

This inserts each element of a part in the path into the bijection.

```
void path_class::insert_element_to_rigged (int num,
      int row_indx, int col_indx,int nrow, int ncol)
{
   int old_largest_index;
   reset_flags(0);
   // add the new element - num - to rigged and
   // add box if necessary
   for (int i = (num-2); i >= row_indx; i--) {
      old_largest_index =
        add_box_to_rigged(i, num-2,
          old_largest_index, 0);
   }

   // initialize curL to bigL
   for (int i=0; i < RIGSIZE; i++)
      for (int j = 0; j < RIGSIZE; j++)
         curL[i][j] = bigL[i][j];

       // update curL to include the part of the
       //  tableau seen so far

       // we just finished a column
       if (row_indx == nrow - 1) {
          curL [nrow][1] += 1;
          if (col_indx != ncol - 1) {
```

```
        curL [nrow][ncol - col_indx - 1] += 1;
     }
     // we are in the middle of a column
     } else {
        curL [nrow][ncol - col_indx - 1] += 1;
        curL [row_indx + 1][1] += 1;
     }
     // calculate outer and inner labels
     // based on curL
     for (int i = 0; i < n; i++) {
        calc_outer_label (i, 0);
     }
     for (int i = 0; i < n; i++) {
        calc_inner_label (i, 0);
     }

    // only if we are at the end of a column
    // initialize curL to bigL - we'll update
    //  curL differently now in a FUSED way
    // and recompute outer labels
    if (row_indx == nrow - 1) {
       for (int i=0; i < RIGSIZE; i++)
          for (int j = 0; j < RIGSIZE; j++)
            curL[i][j] = bigL[i][j];
```

```
            // update curL -
            // this is different from above
            curL [nrow][ncol - col_indx] += 1;
            for (int i = 0; i < n; i++) {
               calc_outer_label (i, 0);
            }
         }
}
```

This inserts each part (which is a tableau) of a path to the bijection.

```
void path_class::insert_tableau_to_rigged
       (tblu_class* cur_tblu) {
    int nrow = cur_tblu->num_row;
    int ncol = cur_tblu->num_col;
    for (int j = ncol-1; j >=0; j--) {
       for (int i = 0; i < nrow; i++) {
          insert_element_to_rigged(
            cur_tblu->tb[i][j], i, j, nrow, ncol);
       }
    }
    bigL [nrow][ncol] = bigL [nrow][ncol] + 1;
    for (int i = 0; i < RIGSIZE; i++)
       for (int j = 0; j < RIGSIZE; j++)
          curL[i][j] = bigL[i][j];
```

```
}
```

This inserts all the parts of a path to the bijection.

```
void path_class::build_rigged_for_path () {
    int j;
    for (int i=0; i < RIGSIZE; i++) {
        for (j =0; j < RIGSIZE; j++)
            bigL[i][j] = 0;
        curL[i][j] = 0;
    }
    for (int i = path_len - 1; i >= 0; i--) {
        insert_tableau_to_rigged (path[i]);
    }
}
```

Calculates the α function used in the definition of cocharge.

```
int path_class::alpha(int k, int i){
   int num_coln,j;
   num_coln=0;
   if (k>=n) num_coln=0;
   else{
       for (j=0; j<RIGSIZE;j++){
         if (rigged[k][0][0]==UNUSED){
```

```
              num_coln=0 ;
              break;
           } else{
              if (rigged[k][0][j]!=UNUSED){
                 if (rigged[k][0][j]>=i+1){
                 num_coln=num_coln+1;
                 }
              }
           }
        }
   }
   return num_coln;
}
```

Calculates the cocharge for the rigged configuration corresponding.

```
void path_class::calculate_cocharge(){
  int k,j,i,sum,cosum;
  sum=0;
  for (k=0;k<=n-1;k++){
    if (rigged[k][0][0]!=UNUSED){
      for (i=0;i< rigged[k][0][0];i++){
      sum=sum+alpha(k,i)*(alpha(k,i)-alpha(k+1,i));
      }
    }
```

```
  }
  cosum=sum;

  for (k=0;k<=n-1;k++){
    for (j=0;j<RIGSIZE;j++){
      if (rigged[k][2][j]==UNUSED) break;
      if ( rigged[k][2][j]!=UNUSED){
        cosum=cosum + rigged[k][2][j];
      }
    }
  }
  cocharge=cosum;

  fprintf (stderr, "Statistic = %d \n", cosum );
}
```

This is the main program.

```
int main() {
    int i;
    read_input();
            // this reads the input file.
    fprintf (stderr, "n=%d\n", n);
    input_path->print_path();
            // this prints the input path
```

```
    input_path->build_rigged_for_path();
            //finds the corresponding rigged
           //configuration via the bijection.

    fprintf (stderr, "--------------------------\n");
    fprintf (stderr, "\n");

    fprintf (stderr,
      "Corresponding rigged configuration is : \n");

    input_path->print_rigged_for_this_path();
       // prints the resulting rigged configuration.

}
```

.3 Code for the program $inverse\_bijection.c$

This program does the inverse bijection from rigged configuration (RC) to path. Given a rigged configuration, n, path length and the shape of the path it calculates the corresponding path via the bijection.

```
#include <stdio.h>
#define UNUSED 9999
#define RIGSIZE 20
int n, l, num_shapes;
int lambda[100];
```

```
int path_shape[100][100];
int tableau_list[40000][10][10];
int tableau[100][100][100];
int pick[100];
int r, tab_indx, num_rc_lb_tab;
int rigged[RIGSIZE][5][RIGSIZE];
int bigL [RIGSIZE][RIGSIZE];
int curL [RIGSIZE][RIGSIZE];
int tblu_index , path_len;
int num_paths;
FILE *fp;

void initialize() {
   int i, j, k,m;
   for (i=0; i<RIGSIZE;i++){
      for (j=0; j<5; j++){
         for (k=0; k<RIGSIZE; k++){
            rigged[i][j][k]=UNUSED;
            curL[i][k]=0;
            bigL[i][k]=0;

         }
      }
   }
   for (i=0; i<100;i++){
```

```
      for (j=0; j<100; j++){
         for (k=0; k<100; k++){
            tableau[i][j][k]=UNUSED;
         }
      }
   }
   for (i=0;i<100; i++){
      for (j=0; j<100; j++){
         path_shape[i][j]=UNUSED;
      }
   }
}
```

Reads the input from file called "inputrigged".

```
void read_input(){
    int i,j,k,tmp,tmu ;
    char c,c1;
    fp = fopen ("inputrigged","rw");
    fscanf(fp,"%d %d\n", &n, &path_len);

    for (i=0;i<=path_len-1;i++){
        for (j=0; j<2; j++){
            fscanf(fp,"%d",&tmu);
            path_shape[i][j]=tmu;
        }
```

```
    }
    i = 0;
    j = 0;
    k = 0;
    while (fscanf (fp, "%d", &tmp)!= EOF){
      rigged[i][j][k] = tmp;
        k++;
        c=fgetc(fp);
        if (c=='\n'){
           k=0;
           if(j<1) j=j+2;
           else {j=0; i++;}
        }
    }
}
```

Prints the input.

```
void print_input() {
    int i,j;
    fprintf (stderr, "n = %d  L= %d\n", n, path_len);
    fprintf (stderr, "mu \n");
    for (i=0; i <=path_len-1; i++) {
       j=0;
       while (path_shape[i][j] != UNUSED){
           fprintf (stderr, "%d ", path_shape[i][j]);
```

```
                j++;
            }
            fprintf(stderr,  "\n");
        }

        fprintf (stderr, "\n");
}

void reset_flags () {
        int i, k;
        for (i=0; i < RIGSIZE; i++) {
            for (k=0; k < RIGSIZE; k++) {
                rigged [i][3][k] = 0;
            }
        }
}
```

Prints the RC.

```
void print_rigged() {
   int i, j, k, l,a,b;
   fprintf(stderr, "Given rigged configuration is:\n");

   for (i = 0; i < n; i++) {
      if (rigged[i][0][0]==UNUSED) {
         fprintf (stderr,"-----------------------\n");
```

```
      fprintf(stderr,"(%d)   Empty\n", i+1);
   }
   else {
      if (rigged[i][0][0] != UNUSED) {
         fprintf (stderr,"---------------------\n");
         fprintf(stderr, "(%d)\n", i+1);

         j=0;
         if (rigged[i][0][j] != UNUSED)
            for (k=0; k <rigged[i][0][j]; k++)
               fprintf (stderr, " ___");
         fprintf (stderr,"\n");
         while (rigged[i][0][j] !=UNUSED){
            k=rigged[i][0][j];
            for (l=0; l<k-1;l++){
                fprintf (stderr, "|   ");
            }
            if (l==k-1)
            fprintf (stderr, "| %2d",rigged[i][2][j]);

            fprintf (stderr,
              "| %d\n",rigged[i][1][j]);
            for (l=0; l<k; l++){
               fprintf (stderr, " ---");
            }
```

```
                fprintf (stderr,"\n");
                j++;
             }
           }
         }
     }
     fprintf (stderr,"------------------------------\n");
}
```

Finds the smallest singular string in a middle RC.

```
int find_smallest_inside_outside_others
      (int index, int old_index) {
   int i = 0;
   int smallest_index = UNUSED;
   while ((rigged[index][0][i] != UNUSED)
          && i < RIGSIZE) {
      if ((rigged[index][0][i] >= old_index) &&
          (rigged[index][1][i] == rigged[index][2][i])
          && (rigged[index][1][i] != UNUSED)) {
             smallest_index = i;
      }
      i++;
   }
   return smallest_index;
}
```

Finds the smallest singular string in the starting RC.

```
int find_smallest_inside_outside_first (int index) {
   int i = 0;
   int smallest_index = UNUSED;
   while ((rigged[index][0][i] != UNUSED) &&
           i < RIGSIZE) {
      if ((rigged[index][1][i] == rigged[index][2][i])
          && (rigged[index][1][i] != UNUSED)) {
            smallest_index = i;
      }
      i++;
   }
   return smallest_index;
}
```

Calculates the new shape of the RC after removing a box.

```
void remove_box_from_this_rigged
       (int index, int column_index) {
   if (rigged [index][0][column_index]==1) {
      rigged [index][0][column_index]=UNUSED;
      rigged [index][3][column_index]=1;
   }
   else {
      rigged [index][0][column_index] -= 1;
```

```
      rigged [index][3][column_index] = 1;
   }
}


int num_box_1k_col (unsigned int i, int k){
   int j, l;
   int num_boxes = 0;
   for (l=1; l <= k; l++) {
      for (j = 0; j < RIGSIZE; j++){
        if (rigged[i][0][j] == UNUSED) break;
        if (rigged[i][0][j] >= l){
            num_boxes += 1;
        }
      }
   }
   return num_boxes;
}
```

Finds the selected singular string and remove boxes from those parts.

```
int remove_box_from_rigged
      (int index, int begin, int old_smallest_index) {
   int smallest_index;
   if (index == begin) {
      smallest_index =
        find_smallest_inside_outside_first(index);
```

```
  } else {
     smallest_index =
       find_smallest_inside_outside_others (index,
          old_smallest_index);
  }
  if (smallest_index == UNUSED)  return (-1);
  else {
     remove_box_from_this_rigged (index,
         smallest_index);
        //this removes a box from selected part
     if (rigged[index][0][smallest_index]
          ==UNUSED) return 1;
     else return (rigged [index][0][smallest_index]+1);
     // returns the length of the selected part
  }
}
```

This calculates the extra term in the calculation of the vacancy num, which is the contribution from the shape of the path.

```
int second_func(int part_size, int rig_num) {
    int sum = 0;
    for (int i=1; i<RIGSIZE; i++) {
        if (curL[rig_num+1][i] != 0) {
           int minimum = part_size;
           if (i < part_size) minimum = i;
```

```
            sum =sum + minimum * curL[rig_num+1][i];
         }
     }
     return sum;
}
```

Calculates the vacancy numbers for each part of a rigged partition.

```
void calc_outer_label(int rig_num){
   int part_num=0;
   int part_size, p;
   if (rig_num == 0)  {
      for (int part_num=0; part_num < RIGSIZE;
            part_num++) {
         part_size = rigged[0][0][part_num];
         if(part_size == UNUSED) break;
         p = (-2*num_box_1k_col (0, part_size)) +
             (num_box_1k_col (1, part_size)) +
             second_func(part_size, rig_num);
         rigged [0][1][part_num] = p;
      }
   } else {
      for (part_num=0; part_num < RIGSIZE; part_num++) {
         part_size = rigged[rig_num][0][part_num];
         if(part_size == UNUSED) break;
         p = -2*num_box_1k_col (rig_num, part_size) +
```

```
            num_box_1k_col (rig_num-1, part_size) +
            num_box_1k_col (rig_num+1, part_size) +
            second_func(part_size, rig_num);
        rigged [rig_num][1][part_num] = p;
      }
   }
}
```

Calculates the riggings for each part of a rigged partition.

```
void calc_inner_label (int i) {
   int j,k;
   int tmp;
   for (j = 0; j < RIGSIZE; j++) {
      if (rigged[i][0][j] == UNUSED) break;
      if (rigged[i][3][j] == 1) {
         rigged [i][2][j] = rigged [i][1][j];
      }
   }
   for (j = 0; j < RIGSIZE; j++) {
      for (k = 1; k < RIGSIZE; k++) {
         if (rigged[i][0][k] == UNUSED) break;
         if (rigged[i][0][k] == rigged[i][0][k-1]
            && rigged[i][1][k] == rigged[i][1][k-1]
            && rigged[i][2][k-1] > rigged[i][2][k]) {
            tmp = rigged[i][2][k-1];
```

```
            rigged[i][2][k-1] = rigged[i][2][k];
            rigged[i][2][k] = tmp;
         }
      }
   }
}
```

Calculates each element of a tableau in a path.

```
void get_element_from_rc( int row_indx,
       int col_indx, int pathi) {
   int old_smallest_index;
   reset_flags();

   // only if we are starting  a new column
   // we'll update curL by spliting and
   // recompute outer labels first
   if ((row_indx == path_shape[pathi][0]-1)
       && (col_indx < path_shape[pathi][1]-1)) {
      curL[row_indx+1][1] +=1;
      curL [row_indx+1][path_shape[pathi][1]-col_indx]
          -= 1;
      curL[row_indx+1][path_shape[pathi][1]-col_indx-1]
          +=1;
      for (int i = 0; i < n; i++) {
         calc_outer_label (i);
```

```
    }
  }

  // get a new element r from rigged and remove
  // box if necessary
  int end=0;
  for (int i = row_indx ; i < n; i++) {
    old_smallest_index =
       remove_box_from_rigged(i, row_indx,
                       old_smallest_index);
    if (old_smallest_index == -1) {
       end=1;
       tableau[pathi][row_indx][col_indx]=i+1;
       break;
    }
  }
  if (end==0) {
      tableau[pathi][row_indx][col_indx]=n+1;
  }
  // update curL to exclude the part of the
  // tableau seen so far

  // we just finished a column
  if (row_indx == 0) curL [1][1] -= 1;
```

```
    // we are in the middle of a column
    else {
       curL [ row_indx+1 ][ 1 ] -= 1;
       curL [ row_indx ][ 1 ] += 1;
    }

    // calculate outer and inner labels based
    // on updated curL
    for (int i = 0; i < n; i++) {
       calc_outer_label (i);
    }
    if (tableau[pathi][row_indx][col_indx]
        !=(row_indx+1)) {
       for (int i = 0; i < n; i++) {
          calc_inner_label (i);
       }
    }
}
```

Calculates the rigged configuration for one tableau in the path.

```
void get_tableau_from_rc(int pathi) {
   int nrow = path_shape[pathi][0];
   int ncol = path_shape[pathi][1];
   for (int j = 0; j <= ncol-1; j++) {
      for (int i = nrow-1; i >=0; i--) {
```

```
        get_element_from_rc(i,j,pathi);
      }
   }
   bigL [nrow][ncol] = bigL [nrow][ncol] - 1;
   for (int i = 0; i < RIGSIZE; i++)
   for (int j = 0; j < RIGSIZE; j++)
      curL[i][j] = bigL[i][j];
   fprintf(stderr, "-------------------------\n");
   for (int i=0; i <= nrow-1; i++) {
      for (int j=0; j<= ncol-1; j++) {
        fprintf(stderr, "%2d", tableau[pathi][i][j]);
      }
      fprintf(stderr, "\n");
   }
}
```

Calculates the rigged configuration for a given path.

```
void build_path_for_rc() {
   int i,j,k,tmp;
   for (i=0; i < RIGSIZE; i++) {
      for (j =0; j < RIGSIZE; j++) {
         if ((j==0) && (path_shape[i][j] != UNUSED)) {
            bigL[path_shape[i][j]][path_shape[i][j+1]] += 1;
            curL[path_shape[i][j]][path_shape[i][j+1]] += 1;
         }
```

```
    }
}
for (int i = 0; i < n; i++) {
    calc_outer_label (i);
}
for (i=0;i<n; i++) {
    for (j = 0; j < RIGSIZE; j++) {
        for (k = 1; k < RIGSIZE; k++) {
            if (rigged[i][0][k] == UNUSED) break;
            if (rigged[i][0][k] == rigged[i][0][k-1]
                 && rigged[i][1][k] == rigged[i][1][k-1]
                 && rigged[i][2][k-1] > rigged[i][2][k]) {
                tmp = rigged[i][2][k-1];
                rigged[i][2][k-1] = rigged[i][2][k];
                rigged[i][2][k] = tmp;
            }
        }
    }
}

print_rigged();

fprintf(stderr,"The corresponding path is:\n");
for (i = 0; i < path_len; i++) {
```

```
      get_tableau_from_rc (i);
   }
   fprintf(stderr, "-------------------------\n");
}
```

Main program.

```
int main(){
    initialize();
    read_input();
    print_input();
    build_path_for_rc();

}
```

Bibliography

[1] A.K. Agarwal, G.E. Andrews, D.M. Bressoud, *The Bailey lattice*, J. Ind. Math. Soc. **51** (1987), 57–73.

[2] G.E. Andrews, *An analytic generalization of the Rogers-Ramanujan identities for odd moduli*, Prod. Nat. Acad. Sci. USA **71** (1974), 4082-4085.

[3] G.E. Andrews, *The theory of partitions*, Encyclopedia of Mathematics, vol. **2** (Addison-Wesley, Reading, Massachusetts, 1976).

[4] G.E. Andrews, *Multiple series Rogers-Ramanujan type identities*, Pacific J. Math. **114** (1984), no. **2**, 267–283.

[5] G.E. Andrews, A. Berkovich, *A trinomial analogue of Bailey's Lemma and $N = 2$ superconformal Invariance*, Commun. Math. Phys.

[6] G.E. Andrews, A. Schilling, S.O. Warnaar, *An A_2 Bailey lemma and Rogers-Ramanujan-type identities*, J. Amer. Math. Soc. **12** (1999) 677–702.

[7] W.N. Bailey, *Identities of the Rogers–Ramanujan type*, Proc. London Math. Soc. (2) **50** (1949) 1–10.

[8] H.A. Bethe, *Zur Theorie der Metalle, I. Eigenwerte und Eigenfunktionen der linearen Atomkette*, Z. Physik **71** (1931) 205–231.

[9] A. Berkovich, B.M. McCoy, *Continued fraction and fermionic representations for characters of $M(p,p')$ minimal models*, Lett. Math. Phys. **37** (1996), 49–66.

[10] A. Berkovich, B.M. McCoy, A. Schilling, $N=2$ *Supersymmetry and Bailey pairs*, Physica A **228** (1996) 33–62.

[11] A. Berkovich, B.M. McCoy, A. Schilling, *Rogers-Schur-Ramanujan type identities for the $M(p,p')$ minimal models of conformal field theory*, Commun. Math. Phys. **191** (1998), 325–395.

[12] A. Berkovich, B.M. McCoy, A. Schilling, S.O. Warnaar, *Bailey flows and Bose-Fermi identities for coset models* $(A_1^{(1)})_N \times (A_1^{(1)})_{N'}/(A_1^{(1)})_{N+N'}$, Nucl. Phys. B **499** [PM] (1997), 621–649.

[13] A.A. Belavin, A.M. Polyyakov,A.B.Zamolodchikov, *Infinite conformal symmetry of critical fluctuations in two dimensions*, J. Stat. Phys. **34**, no.5-6 (1984), 763-774.

[14] A.A. Belavin, A.M. Polyyakov,A.B.Zamolodchikov, *Infinite conformal field symmetry in two dimensionsal quantum field theory*, Nucl.Phys. **B241** (1984), 333-380.

[15] D.M. Bressoud, *An analytic generalization of the Rogers-Ramanujan identitieswith interpretation*, Quart. J. Math. Oxford (2) **31** (1980), 385-399.

[16] A. Capelli, C. Itzykson, J.B. Zuber, *The A-D-E classifications of minimal and $A_1^{(1)}$ conformal Invariant theoies*, Commun. Math. Phys. **113** (1987), 1-26.

[17] S. Dasmahapatra, R. Kedem, B.M. McCoy, E. Melzer, *Virasoro characters from Bethe equations for the critical ferromagnetic three-state Potts model*, J. Stat. Phys. **74** (1994), 239-274.

[18] L. Deka, A. Schilling, *New Explicit expression for $A_{n-1}^{(1)}$ supernomials*, 17th International conference, FPSAC'2005, University of Messina, Italy, June 2005.

[19] V.K. Dobrev, *Characters of the irreducible highest weight modules over the Virasoro and super-Virasoro algebras*, Suppl. Rendiconti Circolo Matematici di Palermo, Serie II, Numero **14** (1987), 25-42.

[20] M. Dörrzapf, *The embedding structure of unitary $N = 2$ minimal models*, Nucl. Phys. B **529** (1998), 639–655.

[21] W. Eholzer, M.R. Gaberdiel, *Unitarity of rational $N = 2$ superconformal theories*, Commun. Math. Phys. **186** (1997) 61–85.

[22] B.L. Feigin, D.B. Fuchs, *Verma modules over the Virasoro algebra*, Topology (Leningrad, 1982), 230-245, Lecture Notes in Math. **1060** (Springer, Berlin-New York, 1984).

[23] O. Foda, Y.H. Quano, *Polynomial identities of the Rogers-Ramanujan type*, Int. J. Mod. Phys. A **10** (1995), 2291–2315.

[24] O. Foda, Y.H. Quano, *Virasoro character identities from the Andrews-Bailey construction*, Int. J. Mod. Phys. A **12** (1996), 1651–1675.

[25] O. Foda, T.A. Welsh, *On the combinatorics of Forrester-Baxter models*, Physical combinatorics (Kyoto, 1999), 49–103, Progr. Math., **191**, Birkhüser Boston, Boston, MA, 2000.

[26] H.O. Foulkes, *A survey of some combinatorial aspects of symmetric functions*, Permutations, Cauthier-Villas, Paris (1974), 79-92.

[27] W. Fulton, *Young Tableaux*, London Math. Soc., Student Texts **35**, Cambridge University Press.

[28] P. Goddard, A. Kent, D. Olive, *Unitary representations of the Virasoro and super-Virasoro algebras*, Comm. Math. Phys. **103** (1986), no. 1, 105–119.

[29] B. Gordon, *A combinatorial generalization of the Rogers-Ramanujan identities*, Amer. J. Math. **83** (1961), 393-399.

[30] G. Hatayama, A. Kuniba, M. Okado, T. Takagi, Z. Tsuboi, *Paths, crystals and fermionic formulae*, Prog. Math. Phys. **23** (2002) 205–272, Birkhäuser Boston, Boston, MA.

[31] G. Hatayama, N. Kirillov, A. Kuniba, M. Okado, T. Takagi, Y. Yamada, *Character formulae of $\hat{sl}_n$-modules and inhomogeneous paths*, Contemp. Math. **248** (1999) 243–291.

[32] G. Hatayama, A. Kuniba, M. Okado, T. Takagi, Z. Tsuboi, *Paths, crsytals and fermionic formulae*, MathPhys.odyssey, 2001, 205-272, Prog. Math. Phys. **23**, Birkhäuser Boston, Boston, MA, 2002.

[33] G. Hatayama, A. Kuniba, M. Okado, T. Takagi, Y. Yamada, *Remarks on fermionic formula*, Contemp. Math. **248** (1999) 243–291.

[34] V.G. Kac, *Infinite dimensional Lie algebras*, third edition, Cambridge University Press.

[35] H. Klemm, *Embedding diagrams of the $N = 2$ superconformal Algebra under spectral flow*, preprint hep-th/0306073.

[36] H. Klemm, private communication.

[37] M. Kashiwara, *Crystalizing the q-analogue of universal enveloping algebras*, Commun. Math. Phys. **133** (1990) 249–260.

[38] M. Kashiwara, T. Nakashima, *Crystal graphs for representations of the q-analogue of classical Lie algebras*, J. Algebra **165** (1994), no. 2, 295–345.

[39] E.B. Kiritsis, *Character formulae and the structure of of the representations of the $N = 1$ and $N = 2$ superconformal algebras*, Int. J. Mod. Phys. A **3** (1988) 1871.

[40] R. Kedem, B.M. McCoy, *Construction of modular branching functions from Bethe's equations in the 3-state Potts chain*, J. Stat. Phys. **71** (1993), 865-901.

[41] R. Kedem, T.R. Klassen, B.M. McCoy and E. Melzer, *Fermionic quasi-particle representations for character of $(G_1^{(1)} \times (G_1^{(1)}/(G_2^{(1)}$*, Phys. Lett. B **304** (1993),263-270.

[42] R.Kedem, T.R. Klassen, B.M. McCoy and E. Melzer, *Fermionic sum representations for conformal field theory characters*, Phys. Lett. B **307** (1993), 68-76.

[43] S.-J. Kang, M. Kashiwara, K.C. Misra, T. Miwa, T. Nakashima and A. Nakayashiki, *Perfect crystals of quantum affine Lie algebras*, Duke Math. J. **68** (1992) 499607.

[44] S.V. Kerov, A.N. Kirillov, N.Y. Reshetikhin, *Combinatorics, the Bethe ansatz and representations of the symmetric group* J. Soviet Math. **41** (1988), no. 2, 916–924.

[45] A.N. Kirillov, *Dilogarithm identities*, Progress of Theor. Phys. Suppl. 1995, v.118, 61-142.

[46] A.N. Kirillov, *New combinatorial formula for modified Hall- Littlewood polynomials*, in "q-series from a contemporary perspective" (South Hadley, MA, 1998), 283-333, Contemp. Math., 254, Amer. Math. Soc., Providence, RI, 2000.

[47] A.N. Kirillov, N.Y. Reshetikhin, *The Bethe ansatz and the combinatorics of Young tableaux*, (Russian) Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI) **155** (1986), Differentsialnaya Geometriya, Gruppy Li i Mekh. VIII, 65–115, 194; translation in J. Soviet Math. **41** (1988), no. 2, 925–955.

[48] A.N. Kirillov, A. Schilling, M. Shimozono, *A bijection between Littlewood-Richardson tableaux and rigged configurations*, Selecta Mathematica (N.S.) **8** (2002) 67–135.

[49] A.N. Kirillov, M. Shimozono, *A generalization of the Kostka-Foulkes polynomials*, J. Algebraic Combin. **15** (2002), no. 1, 27–69.

[50] A. Kuniba, M. Okado, R. Sakamoto, T. Takagi, Y. Yamada, private communication.

[51] A. Lascoux, B. Leclerc, J.-Y. Thibon, *Ribbon tableaux, Hall-Littlewood functions, quantum affine algebras and unipotent varieties*, J.Math. Phys. **38** (1997) 1041-1068.

[52] A. Lascoux, M.-P. Schützenberger, *Sur une conjecture de H. O. Foulkes*, C. R. Acad. Sci. Paris Sér. A-B **286** (1978), no. **7**, A323–A324.

[53] B. Leclerc, J.-Y. Thibon, *Littlewood-Richardson coefficients and Kazhdan-Lusztig polynomials*, Adv. Stud. in Pure Math. **28** (2000) 155-220.

[54] P.A. MacMohan, *Combinatory analysis* vol. 2 (Cambridge University Press, Cambridge, 1916).

[55] I.G. Macdonald, *Symmetric functions and Hall polynomials*, second edition, Clarendon Press, Oxford.

[56] S.C. Milne, G.M, Lilly, *The A_ℓ and C_ℓ Bailey transform and lemma*, Bull. Amer. Math. Soc. (N.S.) **26** (1992), 258-263.

[57] S.C. Milne, G.M, Lilly, *Consequences of the A_ℓ and C_ℓ Bailey transform and Bailey lemma*, Discrete Math. **139** (1995), 319-346.

[58] MuPAD-Combinat available at
`http://www-igm.univ-mlv.fr/~descouen/MuPAD-Combinat/MuPAD-Combinat.html`.

[59] A. Neveu, J.H. Schwarz, *Factorizable dual model of pion* Nuclear Phys. **B31** (1971), 86-112.

[60] A. Nakayashiki, Y. Yamada, *Kostka polynomials and energy functions in solvable lattice models*, Selecta Math. (N.S.) **3** (1997), no. 4, 547–599.

[61] M. Okado, A. Schilling, M. Shimozono, *Virtual crystals and fermionic formulas of type $D^{(2)}_{n+1}, A^{(2)}_{2n}$ and $C^{(1)}_n$*, Represent. Theory **7** (2003), 101-163.

[62] M. Okado, A. Schilling, M. Shimozono, *A crystal to rigged configurations bijection for nonexceptional affine algebras*, Algebraic combinatorics and quantum groups, 85-124, World Sci. Publishing, River Edge, NJ, 2003.

[63] P. Paule, *On identities of the Rogers–Ramanujan type*, J. Math. Anal. Appl. **107** (1985), 255–284.

[64] L.J. Rogers, *Second memoir on the expansion of certain infinite products*, Proc. London Math. Soc. ser. 1 **25** (1894), 313-343.

[65] L.J. Rogers, *On two theorems of combinatory analysis and some allied identities*, Proc. London Math. Soc. (2) **16** (1917), 315-336.

[66] P. Ramond, *Dual theory for free fermions*, Phys. rev. D(3) **3** (1971) 2415-2418.

[67] A. Rocha-Caridi, " *Vacuum vector representations of the Virasoro algebra*" in *Vertex operators in mathematics and physics*, eds. J. Lepowsky, S. Mandelstam and I.M. Singer, Springer-Verlag (1984).

[68] L.J. Rogers, S. Ramanujan, *Proof of certain identities in combinatory analysis*, Proc. Camb. Phil. Soc. **19** (1919), 211-216.

[69] A. Schilling, *A bijection between type $D_n^{(1)}$ crystals and rigged configurations*, J. Algebra **285** (2005) 292–334.

[70] A. Schilling, *Crystal structure on rigged configurations*, preprint math.QA/0508107.

[71] I. Schur, *Ein Beitrag zur additiven Zahlentheorie and zur Theorie der Kettenbrüche*, S.-B. Preuss. Akad. Wiss. Phys.-Math. KI. 302-321 (1917), 117-136.

[72] A. Schwimmer, N. Seiberg, *Comments on the $N = 2, N = 3, N = 4$ superconformal algebras in two dimensions*, Phys. Lett. B **184** (1987), 191–196.

[73] A. Schilling, M. Shimozono, *Fermionic formulas for level-restricted generalized Kostka polynomials and coset branching functions*, Commun. Math. Phys. **220** (2001) 105–164.

[74] A. Schilling, M. Shimozono, *$X = M$ for symmetric powers*, J. Algebra, to appear (math.QA/0412376).

[75] A. Schilling, S.O. Warnaar, *Supernomial coefficients, polynomial identities and q-series*, The Ramanujan Journal **2** (1998) 459–494.

[76] A. Schilling, S.O. Warnaar, *Inhomogeneous lattice paths, generalized Kostka polynomials and A_{n-1} supernomials*, Commun. Math. Phys. **202** (1999) 359–401.

[77] M. Shimozono, *A cyclage poset structure for Littlewood-Richardson tableaux*, European J. Combin. **22** (2001), no. 3, 365–393.

[78] M. Shimozono, *Multi-atoms and monotonicity of generalized Kostka polynomials*, European J. Combin. **22** (2001), no. 3, 395–414.

[79] M. Shimozono, J. Weyman, *Graded characters of modules supported in the closure of a nilpotent conjugacy class*, European J. Combin. **21** (2000), no. 2, 257–288.

[80] M. Shimozono, *Affine type A crystal structure on tensor products of rectangles, Demazure characters, and nilpotent varieties*, J. Algebraic Combin. **15** (2002), no. 2, 151–187.

[81] L.J. Slater, *A new proof of Rogers's transformation of infinite series*, Proc. London Math. Soc. (2) **53** (1951), 460–475.

[82] L.J. Slater, *Further identities of the Rogers-Ramanujan type*, Proc. London Math. Soc.(2) **54** (1952), 147-167.

[83] T. Takagi, *Inverse scattering method for a soliton cellular automaton*, Nuclear Phys. B **707** (2005) 577–601.

[84] T.A. Welsh, *Fermionic expressions for the minimal model Virasoro characters*, Mem. Amer. Math. Soc. **175** (2005), no.827, viii+160pp.

[85] S.O. Warnaar, *A note on the trinomial analogue of the Bailey's lemma*, J. Combin. Theory Ser. A **81** (1998), no. 1, 114–118.

[86] S.O. Warnaar, *The Bailey lemma and Kostka polynomials*, J. Algebraic Combin. **20** (2004) 131–171.