

A Sundaram type Bijection for $SO(3)$: Vacillating Tableaux and Pairs of Standard Young Tableaux and Orthogonal Littlewood-Richardson Tableaux

Judith Jagenteufel*

Institut für Diskrete Mathematik und Geometrie
Fakultät für Mathematik und Geoinformation
TU Wien, Austria

`judith.jagenteufel@tuwien.ac.at`

Submitted: Mar 8, 2018; Accepted: Aug 22, 2018; Published: Sep 21, 2018

© The author. Released under the CC BY license (International 4.0).

Abstract

Motivated by the direct-sum-decomposition of the r^{th} tensor power of the defining representation of the special orthogonal group $SO(2k+1)$, we present a bijection between vacillating tableaux and pairs consisting of a standard Young tableau and an orthogonal Littlewood-Richardson tableau for $SO(3)$.

Our bijection preserves a suitably defined descent set. Using it we determine the quasi-symmetric expansion of the Frobenius characters of the isotypic components.

On the combinatorial side we obtain a bijection between Riordan paths and standard Young tableaux with at most 3 rows, all of even length or all of odd length.

Mathematics Subject Classifications: 05E10

1 Introduction

We solve a problem posed by Sundaram in her 1986 thesis [12] concerning the direct-sum-decomposition into irreducibles of tensor powers of the defining representation (also known as vector representation) of the special orthogonal group $SO(3)$. In particular, our main result is a bijection for $SO(3)$ between so called vacillating tableaux and pairs consisting of an orthogonal Littlewood-Richardson tableau (recently introduced by Kwon [5]) and a standard Young tableau.

*Supported by the European Research Council (ERC): 306445 and the Austrian science fund (FWF): P29275

The question of finding such a bijection arises when we consider the decomposition of a tensor power $V^{\otimes r}$ of the defining representation V of $\mathrm{SO}(2k+1)$ as an $\mathrm{SO}(2k+1) \times \mathfrak{S}_r$ representation,

$$V^{\otimes r} = \bigoplus_{\mu} V(\mu) \otimes U(r, \mu) = \bigoplus_{\mu} V(\mu) \otimes \bigoplus_{\lambda} c_{\lambda}^{\mu} S(\lambda),$$

where $V(\mu)$ is an irreducible representation of $\mathrm{SO}(2k+1)$ and $S(\lambda)$ is a Specht module. Considering the decomposition, we concentrate on $U(r, \mu)$. A basis of $U(r, \mu)$ can be indexed by vacillating tableaux. In dimension 3 these can be regarded as Riordan paths: lattice paths with north-east, east and south-east steps, no steps below the x -axis and no east steps on the x -axis. The multiplicities c_{λ}^{μ} can be obtained by counting orthogonal Littlewood-Richardson tableaux, while a basis of $S(\lambda)$ is indexed by standard Young tableaux.

Our bijection is composed of two independent bijections. The first one, described by Algorithm 1 and its inverse Algorithm 2, considers orthogonal Littlewood-Richardson tableaux and standard Young tableaux and reduces the problem to the case of vacillating tableaux of shape \emptyset . To formulate it we give an explicit combinatorial description of Kwon's orthogonal Littlewood-Richardson tableaux, which are defined in a very general way, although, too abstract for our purposes. The second bijection, described by Algorithm 3 and its inverse Algorithm 4, is a bijection between vacillating tableaux of dimension 3 of shape \emptyset and standard Young tableaux with 3 rows, all of even length or all of odd length.

The problem of finding a bijection explaining the decomposition of $U(r, \mu)$ has been attacked several times since Sundaram's thesis; in particular by Sundaram [13], Proctor [7] and recently by Okada [6]. Okada implicitly obtains the decomposition of $U(r, \mu)$ for the multiplicity free cases $\mu = \emptyset$ and $\mu = (1)$ using representation theoretic computations in [6, Theorem 5.3(3)] and proves that the associated combinatorial objects have the same cardinality in [6, Corollary 5.5(3)]. Our bijection provides a proof for these results as a special case, which are on their part special cases of Okada's work. In fact, Okada asks for bijective proofs of his results. He speculates that Fomin's machinery of growth diagrams could be employed to find such bijections. Indeed, for the symplectic group, special cases of his results were obtained by Roby [8] and Krattenthaler [3] using this method. However, for the orthogonal group the situation appears to be quite different. In particular, at least a naive application of Fomin's ideas does not even yield the desired bijection between vacillating tableaux and the set of standard Young tableaux in question.

An advantage of our combinatorial, bijective proof is that we obtain several additional properties and consequences such as the following.

Our bijection preserves a suitably defined notion of descents: for vacillating tableaux the definition is new, whereas for standard Young tableaux we use the classical descent set introduced by Schützenberger. Thus, we obtain the quasi-symmetric expansion of the Frobenius character of the isotypic space $U(r, \mu)$,

$$\mathrm{ch} U(r, \mu) = \sum_{w \text{ a vacillating tableau}} F_{\mathrm{Des}(w)},$$

where F_D denotes a fundamental quasi-symmetric function.

Among others, this property justifies our bijection to be called “Sundaram-like”, as she described a similar bijection for the defining representation of the symplectic group in her thesis [12]. There exists a similar (but less complicated) definition for descents in oscillating tableaux, which are used in the symplectic case instead of vacillating tableaux, and which Sundaram’s bijection preserves. Thus there also exists a similar quasi-symmetric expansion of the Frobenius character. Rubey, Sagan and Westbury obtained this result in [9].

An essentially different description of our second bijection proves further properties.

Vacillating tableaux of shape \emptyset can be regarded as paths that end at the x -axis, and can thus be concatenated naively. We define concatenation on standard Young tableaux by writing them side by side and adjusting entries to obtain the following: concatenation of vacillating tableaux corresponds to concatenation of standard Young tableaux (see Theorem 41). A similar property holds for Sundaram’s bijection.

Ignoring the orthogonal Littlewood-Richardson tableaux, our bijection describes an insertion algorithm of vacillating tableaux to standard Young tableaux describing the decomposition of $U(r, \mu)$ (see Corollary 56). So does Sundaram’s bijection for the symplectic group, mapping oscillating tableaux to standard Young tableaux. Due to the exceptional isomorphism $\mathfrak{sl}_2 \cong \mathfrak{so}_3$, our algorithm can also be regarded as an insertion algorithm for the decomposition of the isotypic components in tensor powers of the adjoint representation of SL_2 . Thus, our algorithm is a first step towards a resolution of a closely related problem of Stembridge. He shows in [11, Theorem 6.2] that such an algorithm must also exist for the adjoint representation of GL_n , but so far, no explicit description has been found.

2 Background

Although our bijection considers only $k = 1$ (thus $n = 2k + 1 = 3$) we provide the setup for general k , as this ensures a better understanding of the problem.

2.1 Schur-Weyl duality

Starting with the general linear group we have “classical Schur-Weyl duality”

$$V^{\otimes r} \cong \bigoplus_{\substack{\lambda \vdash r \\ \ell(\lambda) \leq n}} V^{\text{GL}}(\lambda) \otimes S(\lambda)$$

as $GL(V) \times \mathfrak{S}_r$ representations where V is a complex vector space of dimension n and

- $GL(V)$ acts diagonally (and on each position by matrix multiplication) and \mathfrak{S}_r permutes tensor positions
- $V^{\text{GL}}(\lambda)$ is an irreducible representation of $GL(V)$ and $S(\lambda)$ is a Specht module.

Consider now the restriction from $GL(V)$ to $SO(V)$ for a vector space V of odd dimension $n = 2k + 1$

$$V(\lambda) \downarrow_{SO(V)}^{GL(V)} \cong \bigoplus_{\substack{\mu \text{ a partition} \\ \ell(\mu) \leq k}} c_{\lambda}^{\mu}(\mathfrak{d}) V^{SO}(\mu)$$

where $c_{\lambda}^{\mu}(\mathfrak{d})$ is the multiplicity of the irreducible representation $V^{SO}(\mu)$ of $SO(V)$ in $V^{GL}(\lambda)$. For $\ell(\lambda) \leq k$ this simplifies to the classical branching rule due to Littlewood. We present Kwon's definition of orthogonal Littlewood-Richardson tableaux, a set that is counted by $c_{\lambda}^{\mu}(\mathfrak{d})$, in Section 2.5.

Combining Schur-Weyl duality and the branching rule we obtain an isomorphism of $SO(V) \times \mathfrak{S}_r$ representations

$$V^{\otimes r} \cong \bigoplus_{\substack{\lambda \vdash r \\ \ell(\lambda) \leq n}} \left(\bigoplus_{\substack{\mu \text{ a partition} \\ \ell(\mu) \leq k}} c_{\lambda}^{\mu}(\mathfrak{d}) V^{SO}(\mu) \right) \otimes S(\lambda) = \bigoplus_{\substack{\mu \text{ a partition} \\ \ell(\mu) \leq k}} V^{SO}(\mu) \otimes U(r, \mu)$$

with isotypic components of weight μ

$$U(r, \mu) = \bigoplus_{\substack{\lambda \vdash r \\ \ell(\lambda) \leq n}} c_{\lambda}^{\mu}(\mathfrak{d}) S(\lambda). \quad (1)$$

The isomorphism of $SO(V)$ representations (e.g. Okada [6, Cor. 3.6]),

$$V^{SO}(\mu) \otimes V \cong \bigoplus_{\substack{\ell(\lambda) \leq k \\ \lambda = \mu \pm \square \\ \text{or } \lambda = \mu \text{ and } \ell(\mu) = k}} V^{SO}(\lambda)$$

implies that a basis of $U(r, \mu)$ can be indexed by so called vacillating tableaux of shape μ , defined in Section 2.2.

Therefore, to explain the decomposition (1) combinatorially, we are interested in a bijection between vacillating tableaux and pairs that consist of a standard Young tableau and an orthogonal Littlewood-Richardson tableau, as a basis of $S(\lambda)$ can be indexed with standard Young tableaux.

Moreover, we go further and follow the approach taken by Rubey, Sagan and Westbury [9] for the symplectic group, by introducing descent sets for vacillating tableaux; see Section 2.3. This enables us to describe the quasi-symmetric expansion of the Frobenius character. For more information on the Frobenius character see the text book by Stanley [10].

Recall that the Frobenius character of a representation ρ of the symmetric group is

$$\text{ch } \rho = \frac{1}{r!} \sum_{\pi \in \mathfrak{S}_r} \text{Tr } \rho(\pi) p_{\lambda(\pi)}$$

where p_λ denotes the power sum symmetric function associated with λ .

For $D \subseteq \{1, 2, \dots, r\}$ the *fundamental quasi-symmetric function* F_D is defined as

$$F_D = \sum_{\substack{i_1 \leq i_2 \leq \dots \leq i_r \\ j \in D \Rightarrow i_j < i_{j+1}}} x_{i_1} x_{i_2} \dots x_{i_r}.$$

Recall that

$$\text{ch } S(\lambda) = s_\lambda = \sum_{Q \in \text{SYT}(\lambda)} F_{\text{Des}(Q)}$$

where s_λ denotes a Schur function and $\text{Des}(Q)$ is the descent set of a standard Young tableau, see Section 2.3. The Frobenius character can be also defined by this equation and the requirement that it be an isometry.

For $\text{SO}(3)$ we define a descent-preserving bijection. Therefore we obtain the following theorem.

Theorem 1.

$$\text{ch } U(r, \mu) = \sum F_{\text{Des}(w)},$$

where the sum runs over all vacillating tableaux w of length r and shape μ and $\text{Des}(w)$ is the descent set of w .

The extension to the general case, $\text{SO}(2k+1)$, is work in progress and will be presented in another paper.

2.2 Vacillating Tableaux

We define two equivalent families of objects: vacillating tableaux (as defined by Sundaram in [13, Def. 4.1]) and highest weight words.

Definition 2.

1. A $((2k+1)$ -orthogonal) *vacillating tableau* of length r is a sequence of partitions $\emptyset = \mu^0, \mu^1, \dots, \mu^r = \mu$ each of at most k parts, such that:
 - μ^i and μ^{i+1} differ in at most one position,
 - $\mu^i = \mu^{i+1}$ only occurs if the k^{th} part of both partitions is non-empty.

We call the final partition μ the *shape* of the tableau.

2. A $((2k+1)$ -orthogonal) *highest weight word* of length r is a word $w = (w_1, w_2, \dots, w_r)$ with letters $w_\ell \in \{\pm 1, \pm 2, \dots, \pm k, 0\}$ such that for every initial segment s of w the following holds (we write $\#i$ for the number of i 's in s):
 - $\#i - \#(-i) \geq 0$,
 - $\#i - \#(-i) \geq \#(i+1) - \#(-i-1)$,

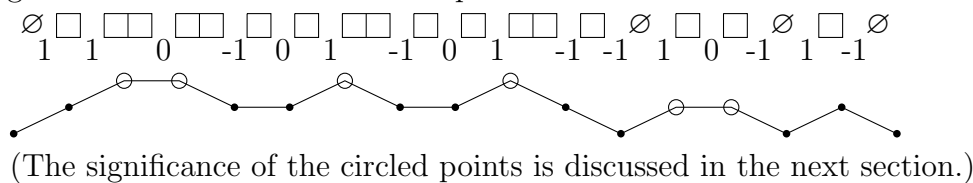
- if the last letter is 0 then $\#k - \#(-k) > 0$.

We call the partition $(\#1 - \#(-1), \#2 - \#(-2), \dots, \#k - \#(-k))$ the *weight* of a highest weight word. The vacillating tableau corresponding to a word w is the sequence of weights of the initial segments of w .

By abuse of terminology we refer to both objects as *vacillating tableaux*.

For $k = 1$ we can regard vacillating tableaux as certain lattice paths (Riordan paths: Motzkin paths without horizontal steps on the x -axis). We therefore take a north-east step for every 1, a south-east step for every -1 and an east step for every 0. In this setting we want the path to be in the first quadrant and do not allow east steps on the x -axis.

Example 3. The same object once written as a vacillating tableau, once as a highest weight word and once as a Riordan path.



2.3 Descents

Following Rubey, Sagan and Westbury [9] we define the descent set for standard Young tableaux and vacillating tableaux.

Definition 4. Let $\text{SYT}(\lambda)$ be the set of standard Young tableaux of shape λ . For a standard Young tableau $Q \in \text{SYT}(\lambda)$ we call an entry j a *descent* if $j + 1$ is in a row below j . We define the *descent set* of Q as:

$$\text{Des}(Q) = \{j : j \text{ is a descent of } Q\}.$$

Example 5. The standard Young tableau

1	2	6	9
3	5	8	
4	7		

has descent set $\{2, 3, 6\}$.

Definition 6 (Rubey, personal conversation). We define descents for vacillating tableaux using highest weight words. We call a letter j of w a *descent* if there exists a directed path from w_j to w_{j+1} in the crystal graph for the defining representation of $\text{SO}(2k + 1)$

$$1 \rightarrow 2 \rightarrow \dots \rightarrow k \rightarrow 0 \rightarrow -k \rightarrow \dots \rightarrow -1$$

and $w_j w_{j+1} \neq j(-j)$ if for the initial segment w_1, \dots, w_{j-1} holds $\#j - \#(-j) = 0$.

For $k = 1$ this means that j is a descent if the j^{th} position is a 1 followed by a 0, or a 0 followed by a -1 , or a 1 followed by a -1 if the numbers of 1's and -1 's before the j^{th} position are different.

We define the *descent set* of w as

$$\text{Des}(w) = \{j : j \text{ is a descent of } w\}.$$

The vacillating tableau in Example 3 has descent set $\{2, 3, 6, 9, 12, 13\}$. The corresponding positions are circled.

2.4 Crystal Graphs

In this section we summarize some properties of crystal graphs. In particular, we describe a certain crystal graph, that we need for defining orthogonal Littlewood-Richardson tableaux. For more information on crystals see the text book by Hong and Kang [1].

Crystal graphs are certain acyclic directed graphs where vertices have finite in and out degree and each edge is labeled by a natural number. We only use crystal graphs whose vertices are labeled with certain tableaux.

For each vertex C there is at most one outgoing edge labeled with i . If such an edge exists we denote its target by $f_i(C)$. Otherwise $f_i(C)$ is defined to be the distinguished symbol \emptyset . Analogously there is at most one incoming edge labeled with i and we define $e_i(C)$ as the tableau obtained by following an incoming edge labeled with i . We denote by $\varphi_i(C)$ (respectively $\varepsilon_i(C)$) the number of times one can apply f_i (respectively e_i) to C .

We consider infinite crystal graphs. However, for the crystal graphs we consider, it holds that if we fix a natural number ℓ and delete all edges labeled with ℓ or larger, as well as all vertices that have incoming edges labeled with ℓ or larger, we obtain a finite crystal graph. Thus a lot of properties proven for finite ones hold also for our infinite crystal graphs.

Definition 7. The crystal graph of one-column tableaux is defined as follows:

1. The vertices are column strict tableaux with a single column and positive integers as entries.
2. Suppose that $i \in \mathbb{N}, i > 0$ is an entry in a tableau C but $i + 1$ is not. Then $f_i(C)$ is the tableau one obtains by replacing i by $i + 1$. Otherwise $f_i(C) = \emptyset$.
3. Suppose that neither 1 nor 2 is an entry in a tableau C . Then $f_0(C)$ is the tableau one obtains by adding a domino $\begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline \end{array}$ on top of C . Otherwise $f_0(C) = \emptyset$.

See Figure 1 for an example.

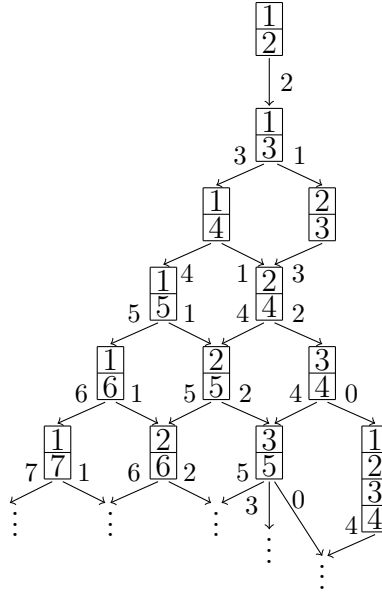


Figure 1: The component with even-length tableaux of a crystal consisting of one-column tableaux.

Next we introduce the tensor product of crystal graphs, using the left-to-right opposite of most of the literature. The tensor product is associative.

Definition 8. The tensor product $B_1 \otimes B_2$ of two crystal graphs B_1 and B_2 is a crystal graph with vertex set $B_1 \times B_2$ and edges satisfying:

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

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

and

$$\varphi_i(b \otimes b') = \varphi_i(b) + \max(0, \varphi_i(b') - \varepsilon_i(b))$$

$$\varepsilon_i(b \otimes b') = \varepsilon_i(b') + \max(0, \varepsilon_i(b) - \varphi_i(b')).$$

Lemma 9. Consider a tensor $b = b_1 \otimes b_2, \otimes \dots \otimes b_n$ and an index i .

We form a sequence of plus and minus signs according to φ_i and ε_i in the following way:

$$\underbrace{(+, +, \dots, +)}_{\varphi_i(b_1)} \underbrace{(-, -, \dots, -)}_{\varepsilon_i(b_1)} \underbrace{(+, +, \dots, +)}_{\varphi_i(b_2)} \underbrace{(-, -, \dots, -)}_{\varepsilon_i(b_2)} \dots \underbrace{(+, +, \dots, +)}_{\varphi_i(b_n)} \underbrace{(-, -, \dots, -)}_{\varepsilon_i(b_n)}$$

Now we cancel out adjacent pairs of “−, +” until we obtain $(+, +, \dots, +, -, -, \dots, -)$ as a new sequence.

Then f_i acts on the b_j corresponding to the rightmost “+” in the new sequence and e_i on the leftmost “−”. Moreover, the number of “+”s in the new sequence is $\varphi_i(b)$ and the number of “−”s is $\varepsilon_i(b)$.

Definition 10. The crystal graph $B^\circ(\mu)$ where μ is a partition with at most $k = 1$ part, is defined as follows. The vertices are pairs of tableaux (T, C) where T is a two-column skew semistandard Young tableau and C is a column strict single column such that:

- The *tail* of a two-column skew semistandard tableau is the part of the tableau where only the first column exists. We require that the tail of T have length μ_1 .
- The *residuum* of a two-column skew semistandard tableau is the number of positions we can shift the second column down such that the result is still a skew semistandard tableau. We require that the residuum of T be less than or equal to 1.

To obtain the edges, we consider this as a subgraph of the tensor product of $3 = 2k + 1$ one-column crystal graphs. Note that taking this point of view $\varphi_i(b_\ell) \in \{1, 0\}$ for $\ell = 1, 2, 3$.

Proposition 11 (Defining Proposition). *If the edges of a crystal graph are considered as a partial order, every connected component of the crystal graph has a unique maximum (see Kashiwara [2]). We call those maxima the highest weight elements.*

2.5 Orthogonal Littlewood-Richardson Tableaux

In order to define orthogonal Littlewood-Richardson tableaux we define a crystal graph, of which they are a part. A proof of the following lemma can be found in Kwon [5, Sec. 3, pages 8, 9 and 13].

Lemma 12 (Defining Lemma, Kwon). *For $n = 3$ the crystal graph $T^\circ(\mu)$ consists of those elements of $B^\circ(\mu)$ described in Subsection 2.4 that are in the same component as one of the following highest weight elements:*

- The two-column skew tableau consists only of one column filled with $1, 2, \dots, \mu_1$. The one-column tableau is empty. (This belongs to Case 1 in the next section.)
- The two-column skew tableau consists only of one column filled with $1, 2, \dots, \mu_1 > 0$. The one-column tableau contains exactly one cell filled with 1. (This belongs to Case 2 or 3 in the next section.)
- The two-column skew tableau consists of two cells in a row, both containing 1. The one-column tableau contains exactly one cell containing 1. (This belongs to Case 4 in the next section.)

See Figure 2 for images of the highest weights elements.

$$\begin{array}{ccc}
 \begin{array}{c} \emptyset \quad \emptyset \\ \boxed{1} \\ \vdots \\ \boxed{\mu_1} \end{array} & \mu_1 \geq 0 & \begin{array}{c} \emptyset \quad \boxed{1} \\ \boxed{1} \\ \vdots \\ \boxed{\mu_1} \end{array} & \mu_1 > 0 & \begin{array}{c} \boxed{11} \quad \boxed{1} \\ \mu_1 = 0 \end{array}
 \end{array}$$

Figure 2: Highest weight elements of $T^\circ(\mu)$

The crystal graph $T^0(\mu)$ is defined in Kwon [5, Eq. (4.7)] and [4, Def. 3.7] as the set of (for $k = 1$) pairs of a two-column skew semistandard Young tableaux where the tail has length μ_1 and a column strict single column with certain dependencies characterized by a list of inequalities. This list was originally described by means of crystal graphs in the way it is stated in Lemma 12.

Definition 13. The set of *orthogonal Littlewood-Richardson tableaux* is

$$\text{LR}_\lambda^\mu(\mathfrak{d}) = \{T \in T^0(\mu) : i \text{ occurs in } T \text{ exactly } \lambda'_i \text{ times and } \varepsilon_i(T) = 0 \text{ for } i \neq 0\}$$

with $\ell(\lambda) \leq n = 2k + 1$ and $\ell(\mu) \leq k$.

For an example see Figure 3.

We give another equivalent definition in the next section that we use for our algorithms.

As announced before, the set of orthogonal Littlewood-Richardson tableaux is counted by $c_\lambda^\mu(\mathfrak{d})$. See [5, Theorem 5.3]. This is one of the main results of [5].

Theorem 14 (Kwon). $c_\lambda^\mu(\mathfrak{d}) = |\text{LR}_\lambda^\mu(\mathfrak{d})|$

Although Kwon considers $\text{O}(n)$, for odd n ($n = 2k + 1$) we get $\text{SO}(n)$ as a special case. In this case $V(\lambda) \downarrow_{\text{SO}(2k+1)}^{\text{O}(2k+1)}$ is an irreducible $\text{SO}(2k + 1)$ representation and every such representation is isomorphic to such a restriction (see for example Okada [6, Sect. 2.4]).

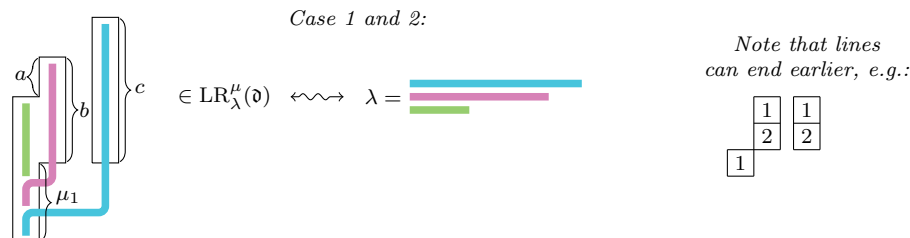
3 The Bijection for $\text{SO}(3)$

3.1 Orthogonal Littlewood-Richardson Tableaux reformulated

To formulate our bijection we first need an explicit description of Kwon's orthogonal Littlewood-Richardson tableaux for $\text{SO}(3)$, provided by the following Theorem.

Theorem 15. *Let λ and μ be partitions such that $\lambda \vdash r$, $\ell(\lambda) \leq 3$ and $\ell(\mu) \leq 1$. Then the orthogonal Littlewood-Richardson tableaux in $\text{LR}_\lambda^\mu(\mathfrak{d})$ for $\text{SO}(3)$ are exactly the following objects. (Along the i^{th} line from right to left, diagrams are filled with $\{1, 2, \dots, \lambda_i\}$. In particular, the number of entries i occurring in this pair of tableaux is λ'_i .)*

There are 4 cases:

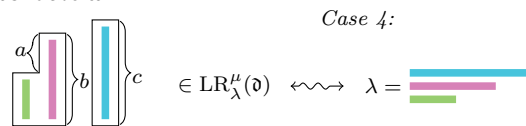


- Case 1: a , b and c are even, $b \leq c$, the residuum is at most 1.

- Case 2: a and b are even, c is odd, $b < c$, $\mu \neq \emptyset$, $\lambda_1 < c + \mu_1$, the residuum is at most 1.



a and b are even, c is odd, the residuum is 1, $\lambda_2 = b - 1 \leq c$, $\mu \neq \emptyset$, $a > 0$. The rightmost line touches the middle column in exactly one cell and starts in the leftmost column two rows below.



a is even, b and c are odd, $\mu = \emptyset$, the residuum is 0.

We can regard orthogonal Littlewood-Richardson tableaux as skew semistandard Young tableaux, whose shape is as indicated.

Corollary 16. The residuum is 1 if and only if the leftmost line has one position in the tail and the middle column is not empty. In this case $a > 0$. The leftmost line cannot have more than one position in the tail as this would give a residuum higher than 1.

Example 17. For $\lambda = (8, 5, 3)$ and thus $r = 16$ we have the following orthogonal Littlewood-Richardson tableaux:

$\mu = (1)$	$\mu = (2)$	$\mu = (3)$	$\mu = (3)$	$\mu = (4)$	$\mu = (5)$
Case 3	Case 1	Case 2	Case 3	Case 1	Case 2

In Table 3 in the appendix we provide a list of all pairs of λ and μ and their associated orthogonal Littlewood-Richardson tableaux for $r \leq 3$.

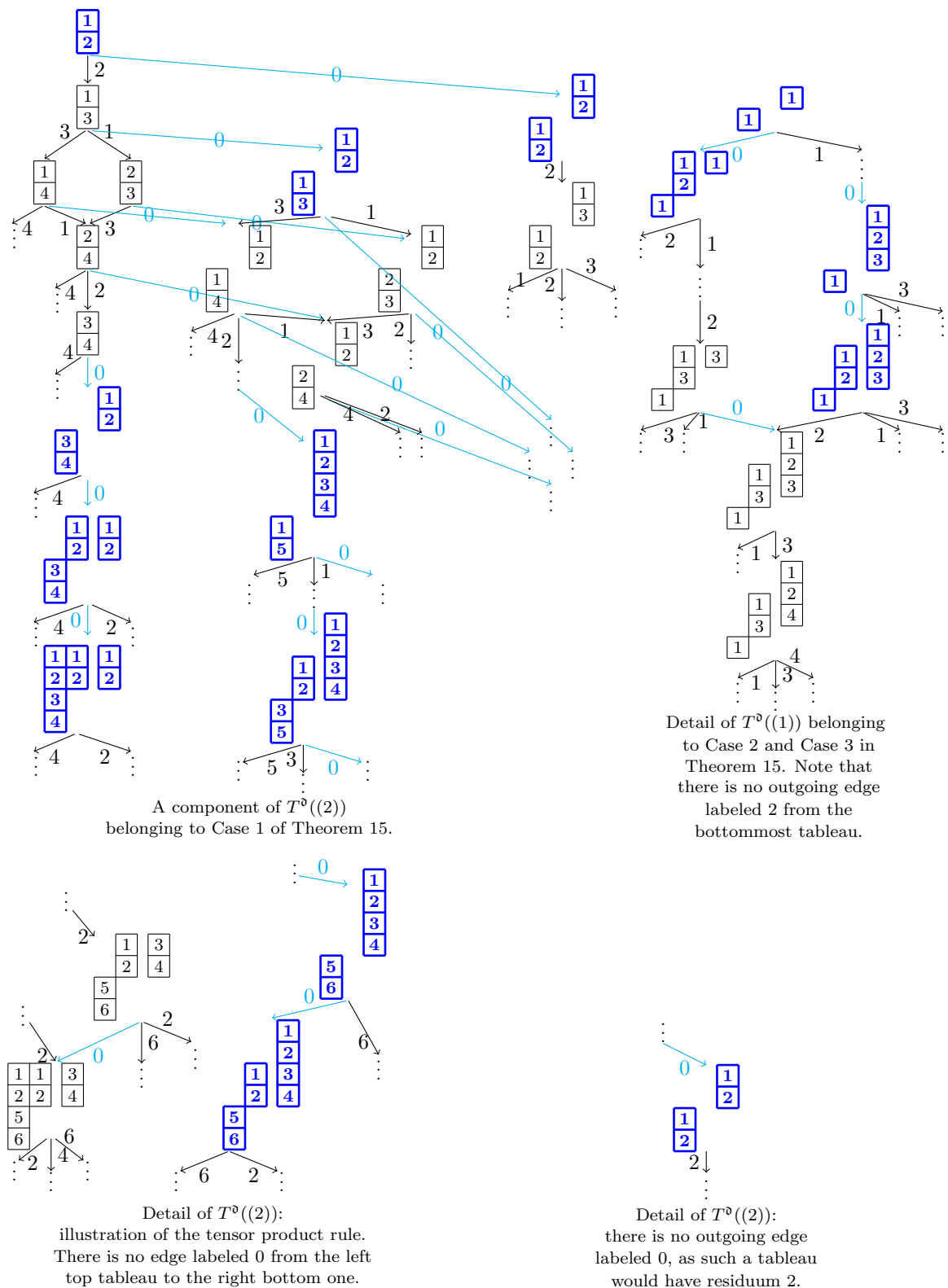


Figure 3: Parts of crystal graphs. Orthogonal Littlewood-Richardson tableaux are emphasized.

For the proof of Theorem 15 we need another observation in order to concentrate on the right paths in a crystal graph.

Proposition 18. *Every element in $T^\circ(\mu)$ can be obtained by applying a sequence of operators $f_{i_j} \circ f_{i_{j-1}} \circ \cdots \circ f_{i_2} \circ f_{i_1}$ to one of the top elements T such that $i_\ell = 0$ if and only if $f_{i_\ell} \circ \cdots \circ f_{i_1}(T)$ is an orthogonal Littlewood-Richardson tableau.*

We call the corresponding path in a crystal graph a good path. The sequence of orthogonal Littlewood-Richardson tableaux on a good path is uniquely determined.

Proof. This follows from Proposition 11: as $T^\circ(\mu)$ is a crystal graph, every component has a unique maximal element (its highest weight element). Each component of the graph obtained from $T^\circ(\mu)$ by erasing the edges labeled with 0 is also a crystal graph and therefore has also a unique maximal element. \square

The following lemma describes orthogonal Littlewood-Richardson tableaux in $T^\circ(\mu)$ and proves Theorem 15.

Lemma 19. *Let L be an orthogonal Littlewood-Richardson tableau in $T^\circ(\mu)$. Define a gap to be an entry $j > 1$ in L such that $j - 1$ is not in the same column.*

1. *L has no gap in the rightmost column.*
2. *L is a skew semistandard Young tableau of the outer shape indicated in Theorem 15. Moreover the parity constraints are as indicated.*
3. *A gap in the middle column b_2 of L can only occur in Case 3 and can only be the largest element of b_2 . Moreover in Case 3 the largest element of b_2 is $c + 1$.*
4. *In Case 3 the tail has entries $b - a + 1, c + 2, c + 3, \dots, c + \mu_1$ with $a \geq 2$. In particular, the residuum is 1.*
5. *In Cases 1 and 2 gaps can only occur in the tail. If j is a gap, then $j - 1$ is the largest entry in a column to the right.*
6. *In Case 2, the smallest entry of the tail is at most c .*

Proof. Before proving the lemma we recall that:

- $\text{LR}_\lambda^\mu(\mathfrak{d})$ is the set of tableaux L in $T^\circ(\mu)$ such that i occurs in L exactly λ'_i times and $\varepsilon_i(L) = 0$ for $i \neq 0$.
 - The length of the tail in $T^\circ(\mu)$ is always μ_1 , thus constant.
1. Suppose that the rightmost column contains a gap j . Then the sequence in Lemma 9 ends with a “ $-$ ” because $\varepsilon_{j-1}(b_3) > 0$ and thus $\varepsilon_{j-1}(b_1 \otimes b_2 \otimes b_3) = \varepsilon_{j-1}(L) > 0$.

2. We use induction on the number of Littlewood-Richardson tableaux on a good path. Let us consider the highest weight elements of $T^\flat(\mu)$

$$L_1 : \begin{array}{|c|} \hline 1 \\ \hline \vdots \\ \hline \mu_1 \\ \hline \end{array} \quad , \quad L_{23} : \begin{array}{|c|} \hline 1 \\ \hline \vdots \\ \hline \mu_1 \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline \end{array} \quad , \quad L_4 : \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline \end{array} \quad (\mu_1 = 0)$$

which belong to Case 1, Case 2 or 3 and Case 4 respectively. As illustrated for $T^\flat(1)$ in Figure 3, the tableaux belonging to Case 2 and Case 3 occur in the same component of the crystal graph. Moreover, following any good path beginning in L_{23} the next orthogonal Littlewood-Richardson tableaux encountered is one of the following four:

$$\begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 3 \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline \vdots \\ \hline \mu_1 \\ \hline \end{array} (\mu_1 \geq 1) \quad , \quad \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 3 \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline 4 \\ \hline \vdots \\ \hline \mu_1+2 \\ \hline \end{array} (\mu_1 \geq 2) \quad , \quad \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 4 \\ \hline \vdots \\ \hline \mu_1+1 \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 3 \\ \hline \end{array} (\mu_1 \geq 3) \quad \text{and} \quad \begin{array}{|c|c|} \hline 1 & 1 \\ \hline 2 & \\ \hline 3 & \\ \hline \vdots & \\ \hline \mu_1+1 & \\ \hline \end{array} (\mu_1 \geq 1)$$

The first three of these belong to Case 2, the last belongs to Case 3. Together with L_1 , L_{23} and L_4 , these tableaux constitute the base case of the induction. Clearly, they are all skew semistandard of the required shape.

In the inductive step we show additionally that the Littlewood-Richardson tableaux on a good path all belong to the same case, with the exception of L_{23} .

For $i > 0$ the generators f_i map skew semistandard Young tableaux to skew semistandard Young tableaux of the same shape (see [2]). Thus it remains to consider f_0 , which adds a domino to one of the three columns and yields an orthogonal Littlewood-Richardson tableau.

If the domino is added on top of the rightmost column there is nothing to check.

Thus, suppose now that the domino is added to the middle column. We have to show that the result is a skew semistandard Young tableau of the same outer shape as the preimage. Item 1 shows that there are no gaps in the rightmost column. If the middle column is non-empty, its smallest entry has a right neighbor by induction, and is at least 3. Otherwise, the rightmost column contains at least two entries, because if it were empty, the domino could not be added to the middle column by Lemma 9. In both cases f_0 preserves the outer shape and semistandardness. Moreover the resulting tableau belongs to the same case as the preceding orthogonal Littlewood-Richardson tableau on the good path.

If the domino is added to the leftmost column, there is nothing to check because the first two columns are skew semistandard by definition of $B^\flat(\mu)$.

Finally, in Case 2 we have $b < c$ because c is odd.

3. If j is a gap in the middle column, $\varepsilon_{j-1}(b_2) > 0$. Because of $\varepsilon_{j-1}(L) = \varepsilon_{j-1}(b_1 \otimes b_2 \otimes b_3) = 0$, we have $\varphi_{j-1}(b_3) > 0$. By item 1 there cannot be a gap in b_3 , so $j - 1$ must be b_3 's largest entry. In particular $j = c + 1$. Because of semistandardness, this implies that the cell containing the gap of b_2 has no right neighbor. The only shape compatible with this requirement is that of Case 3.

Suppose now that b_2 contains no gap. By induction, b_2 contains entries 1 and 2. Therefore, $\varepsilon_0(b_2) = 1$, and Lemma 9 implies that the domino was not added to the rightmost column to obtain L . Let L' be the orthogonal Littlewood-Richardson tableau preceding L on the good path. Then the rightmost columns of L and L' are the same. By induction, the largest entry in the middle column of L' is $c + 1$. Because the generators f_i only increase entries, the largest entry of the middle column of L , is at least $c + 1$. The claim follows, since the length of the middle column is at most $c + 1$, and it contains no gaps.

4. Let t be the smallest entry of the tail and $s = b - 1$ the second largest element of the middle column. We show $t \leq s$ for all tableaux on the good path. Suppose that $t = s$, and consider the action of f_t . If there is a gap in the middle column, $\varphi_t(b_2) > 0$ and the smallest entry in the tail is not modified. Otherwise, the rightmost column also contains $t = c$, so $\varphi_t(b_3) > 0$ and $\varepsilon_t(b_2) = 0$, so the entry in the tail is not modified either.

Because the residuum is at most 1, the second entry of the tail, if present, is at least $c + 2$. Because $c + 2 - (b - a - 1) \geq 2$ it is a gap. However, an entry $j > c + 2$ cannot be a gap, because $\varepsilon_{j-1}(L) = 0$ but $\varphi_{j-1}(b_2) = \varphi_{j-1}(b_3) = 0$ for $j > c + 2$.

5. Suppose that j is a gap in b_1 . Because $\varepsilon_{j-1}(L) = 0$, either $\varphi_{j-1}(b_2) > 0$ or $\varphi_{j-1}(b_3) > 0$. Since there are no gaps in b_2 and b_3 , in both cases $j - 1$ is the largest entry in its column. Semistandardness implies that j must be in the tail.
6. Let $t = b - a + 1$ be the smallest entry of the tail. We show that $t \leq c$ for all tableaux on the good path. Suppose that $t = c$ and consider the action of f_t . Since there is no gap in the middle column, $\varepsilon_t(b_2) = 0$, so f_t modifies the largest element of b_3 . Note that this argument does not hold for Case 1 due to the base case L_1 . \square

3.2 A descent-preserving bijection

In this section we describe our bijection for $\text{SO}(3)$. It maps pairs consisting of a standard Young tableaux with at most 3 rows and an orthogonal Littlewood-Richardson tableau for $\text{SO}(3)$ to vacillating tableaux with at most one row. Moreover, it preserves descents.

The bijection is composed of two parts. First we use the information of the orthogonal Littlewood-Richardson tableau to obtain a bigger standard Young tableau, all of whose row lengths have the same parity. In the second step we map this new standard Young tableau to a vacillating tableau of shape \emptyset . To obtain a vacillating tableau of the desired shape, we delete positions at the end of it such that the number of deleted positions corresponds to the number of added positions in the first part.

We formulate now our bijection using Algorithms 1 and 3, give a complete example as well as formulate the reversed bijection using the reversed Algorithms 2 and 4. In Table 1 we present notation used in Algorithms 3 and 4.

In Section 3.3 we examine the details of the algorithms describing the first part of the bijection. In Section 3.4 we give details for the second part.

3.2.1 Formulation of the bijection

Starting with (Q, L) where Q is a standard Young tableau in $\text{SYT}(\lambda)$ and L an orthogonal Littlewood-Richardson tableau in $\text{LR}^\mu_\lambda(\mathfrak{d})$ we use Algorithm 1 which adds μ_1 cells to Q to obtain a standard Young tableau \tilde{Q} whose row lengths all have the same parity.

Now we distinguish two cases. If our resulting tableau \tilde{Q} consists of rows of even length this is the tableau we use in Algorithm 3. Otherwise, thus when \tilde{Q} consists of three rows of odd length, we look for the largest entry e in \tilde{Q} and attach $e + 1$ to the first row, $e + 2$ to the second one and $e + 3$ to the third one. We obtain a standard Young tableau with rows of even length that we use in in Algorithm 3.

We continue using Algorithm 3 to obtain a vacillating tableau \tilde{V} of shape \emptyset from our modified standard Young tableau \tilde{Q} .

Once again we distinguish the two cases from before. If we added entries to \tilde{Q} to obtain a tableau with rows of even length from one with rows of odd length, we delete now the last three positions of \tilde{V} . Those are always $1, 0, -1$, (see Lemma 38). Otherwise we do not change \tilde{V} . Therefore we obtain once again a vacillating tableau \tilde{V} of shape \emptyset .

We finish our algorithm by deleting the last μ_1 entries of \tilde{V} to obtain a vacillating tableau V of shape μ and length $r = |\lambda|$. Those are always -1 steps (see Lemma 25 and Lemma 39).

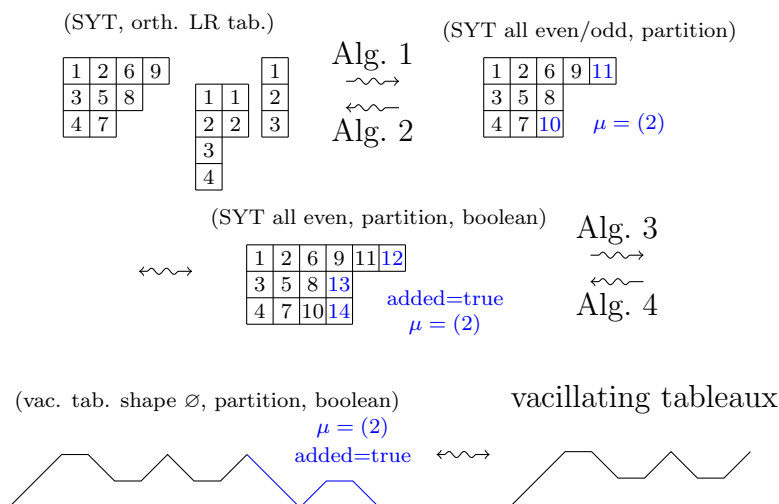


Figure 4: The strategy of our bijection outlined using our running example

In Figure 4 we illustrate the strategy of our bijection using our running example.

In Table 3 in the appendix we provide a list of all pairs (Q, L) , their associated \tilde{Q} and their associated V for $r \leq 3$.

As Algorithm 1 and 2 are inverse and Algorithm 3 and 4 are inverse, the reversed bijection is easily defined.

We start with a vacillating tableau V of shape $\mu = (\mu_1)$ and length r , add $\mu_1 (-1)$'s to obtain a vacillating tableau V of shape \emptyset . If this is of odd length we add $1, 0, -1$. Next we apply Algorithm 4 to obtain a standard Young tableau \tilde{Q} . If we added $1, 0, -1$ we delete the biggest 3 entries. Those are at the end of each row in increasing order (see Lemma 38). Then we apply Algorithm 2 to obtain an orthogonal Littlewood-Richardson tableau in $\text{LR}_\lambda^\mu(\mathfrak{d})$ and a standard Young tableau in $\text{SYT}(\lambda)$ where λ is a partition such that $\lambda \vdash r$ and $\mu \leq \lambda$.

3.3 The first Bijection

In this section we consider the first bijection consisting of Algorithm 1 and its inverse, Algorithm 2. The former maps a pair consisting of an orthogonal Littlewood-Richardson tableau in $\text{LR}_\lambda^\mu(\mathfrak{d})$ and a standard Young tableau in $\text{SYT}(\lambda)$ to a bigger standard Young tableau having μ_1 additional cells that form a horizontal strip with at most one cell in the first row, whose row lengths have all the same parity. We start by giving two examples, the former is of Case 3, the latter, our running example, is of Case 2.

Algorithm 1: (orth. LR tab. + SYT) to (SYT + suitable partition)

input : orthogonal Littlewood-Richardson tableau $L \in \text{LR}_\lambda^\mu(\mathfrak{d})$, standard Young tableau $Q \in \text{SYT}(\lambda)$

output: standard Young tableau \tilde{Q} with 3 (possibly empty) rows, all of even or all of odd length, contains the original standard Young tableau and μ_1 additional positions that form a horizontal strip with at most one element in the first row, that is filled increasingly from left to right

the partition μ , unmodified

rotate L by $\frac{\pi}{2}$ and shift cells such that it becomes the row-strict tableau \tilde{L} , row i being filled with $1, 2, \dots, \lambda_i$;

add cell labeled x to \tilde{L} in row $i + 1$ for each position in row i that was not in column i from the right;

add cells labeled x to \tilde{L} in row 3 such that the number of cells labeled x is μ_1 ;

if not all rows of \tilde{L} have the same parity **then** /* adjust parity */

if total number of cells is odd **then**

└ move last cell of the bottommost even row to the other even row;

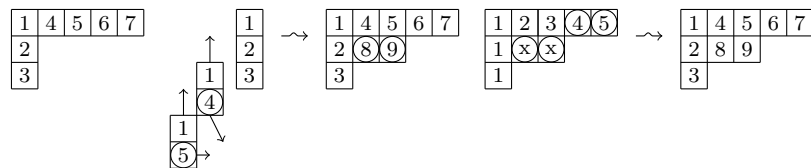
if total number of cells is even **then**

└ move last cell of the bottommost odd row to the other odd row;

add numbers $|\lambda| + 1, |\lambda| + 2, \dots$ increasingly from left to right to Q at the positions of the cells in \tilde{L} labeled x to obtain \tilde{Q} ;

return (\tilde{Q}, μ) ;

Example 20. In this example our orthogonal Littlewood-Richardson tableau is of Case 3 with $\mu = (2)$, $\lambda = (5, 1, 1)$ and $r = 7$:



Algorithm 2: (SYT + suitable partition) to (orth. LR tab. + SYT)

input : standard Young tableau $\tilde{Q} \in \text{SYT}(\tilde{\lambda})$ with 3 rows, all of even or all of odd length, a partition $\mu = (\mu_1)$, such that the skew-diagram containing the μ_1 largest entries in \tilde{Q} form a horizontal strip with at most one position in the first row, filled from left to right increasingly

output: orthogonal Littlewood-Richardson tableau $L \in \text{LR}_\lambda^\mu(\mathfrak{d})$, standard Young tableau $Q \in \text{SYT}(\lambda)$

let L be the row-strict tableau of the same shape as \tilde{Q} , with row i filled with $1, 2, \dots, \tilde{\lambda}_i$;

delete the cells in \tilde{Q} with the μ_1 largest entries to obtain Q ;

replace the entries in the corresponding cells in L with x 's;

rotate L by $-\frac{\pi}{2}$;

for each column of L **do**

/ mark elements for tail */*

if there is an odd number of cells labeled x in this column **then**

 mark the bottommost cell in this column that is not marked yet and contains a number,
 if there exists none, mark the bottommost unmarked cell in the column to the right;
 delete the bottommost cell labeled x ;

if there are still cells labeled x in this column **then**

 beginning at the bottom, mark as many cells, that are not marked yet and contain numbers, in the column to the right, as there are cells labeled x in the current column;
 delete the cells labeled x in the current column;

shift all cells that are marked to the leftmost column;

separate the rightmost column from the other ones such that we obtain two tableaux;

shift the leftmost column down such that the tail has length μ_1 ;

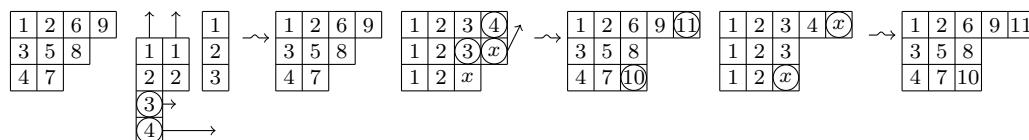
if the second column has odd length and $\mu_1 \neq 0$ **then**

/ fix parity */*

 find the smallest entry in the two-column tableau, that can change columns, such that the resulting tableau, after shifting columns to obtain a tail of length μ_1 , is an orthogonal Littlewood-Richardson tableau; move this entry to the other column in the two-column tableau and shift the columns such that the tail has length μ_1 .

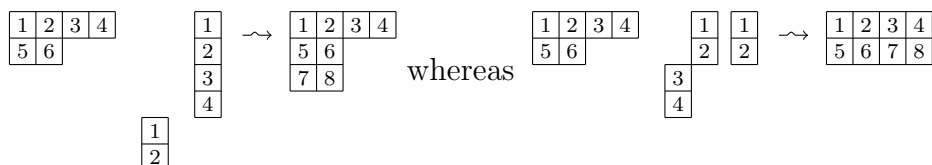
return (L, Q)

Example 21. In our running example we start with a standard Young tableau and an orthogonal Littlewood-Richardson tableau of Case 2 with $\mu = (2)$, $\lambda = (4, 3, 2)$ and $r = 9$:



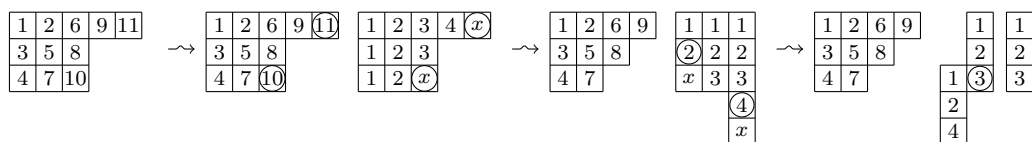
We see that in both cases the added cells form a horizontal strip with at most one position in the first row that is filled from left to right increasingly. Moreover, we point out that same standard Young tableau Q and partition μ still can lead to different standard Young tableau \tilde{Q} . Thus there is essential information in the orthogonal Littlewood-Richardson tableau:

Example 22. $\lambda = (4, 2)$, $\mu = (2)$

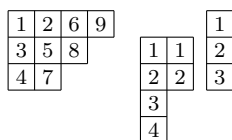


We will see that every step in Algorithm 1 is reversible, and Algorithm 2 is its inverse (see Lemma 28). To illustrate the latter we use our running example and give a full iteration Algorithm 2.

Example 23. In our running example we have $\mu = (2)$ and the following standard Young tableau \tilde{Q} :



Now we have the case that the two left columns are odd. Thus one element has to change column. In this case it is the 3 in the second column. What we obtain are our original tableaux:



We proceed by formulating and proving properties of Algorithm 1 and 2.

Since the filling is not mentioned in the algorithm, the following proposition is clear.

Proposition 24. *Algorithm 1 does not depend on the filling of the standard Young tableau. Only the shape matters.*

Lemma 25. *Algorithm 1 is well-defined and returns a standard Young tableau, all of whose row lengths have the same parity, and a partition μ with a single part. The added cells in the standard Young tableau in Algorithm 1 form a horizontal strip with at most one position in the first row. They are filled from left to right increasingly.*

Proof. Considering the classification in Theorem 15, we note that either μ_1 cells or $\mu_1 - 1$ cells labeled x are added for those cells being shifted from one row to another. An

additional cell labeled x is added to the third row if and only if L is of Case 1 or 2 and the residuum is 1. Thus Algorithm 1 is well-defined.

We now show that, when “adjusting the parity”, the third row of \tilde{L} is non-empty if the total number of cells in \tilde{L} is odd. In Case 1 \tilde{L} has an even number of cells ($|\tilde{Q}| = \mu_1 + |\lambda| = \mu_1 + \mu_1 + 2b - a + c$). In Case 2 and Case 3 the topmost entry of the tail cannot end up in the first row of \tilde{L} (compare again with the classification in Theorem 15). So it either ends up in the third row or in the second row creating a cell labeled x in the third row.

By construction all rows of \tilde{Q} have the same parity. It remains to show that \tilde{Q} is indeed a standard tableau and that the added cells behave as described. This will follow from the fact that the added cells labeled x form a horizontal strip, which is later filled with the numbers $|\lambda| + 1, |\lambda| + 2, \dots, |\lambda| + \mu_1$ from left to right.

First we observe that the cells shifted into another row (from the tail or, in Case 3, from the bottom of the middle column) form a horizontal strip essentially because they all have different entries - compare with Theorem 15.

Thus, when the residuum is 0 or L is of Case 3, cells labeled x also form a horizontal strip, because they are added one row below. This strip only spans two rows. Therefore, after moving a cell to a higher row when “adjusting the parity”, the cells labeled x still form a horizontal strip.

When the residuum is 1 (and thus $a > 0$ and as a is even $a \geq 2$) and L is of Case 1 or 2, an additional cell labeled x is added to the third row. However in this case inspection of the classification of Theorem 15 shows that the third row including all the added cells (including the additional one) is at most as long as the middle line, that is, λ_2 . Now we can argue as above.

There cannot be more than one cell shifted to the first row, as we do not shift more than one cell. \square

Lemma 26. *Algorithm 2 is well-defined.*

Proof. We have to show that there are cells to mark and that there is a cell to shift in the case that the middle and leftmost column have odd length.

As the cells labeled x form a horizontal strip before rotating, they form a vertical strip afterwards. Therefore there are always enough cells to mark in the column to the right. Note that in the rightmost column there is at most one cell labeled x . Thus, if something is marked, it is marked in the same column. Thus we have to consider the parts that mark cells in the same column, namely the columns with an odd number of cells labeled x .

If there is an odd number of cells labeled x in the leftmost column, either this marks something from the leftmost column or also from the middle one.

If there is an odd number of cells labeled x in the middle column we need to show that the middle column is longer than the number of cells labeled x in the leftmost one. Therefore we consider the original parity of the lines. As the number of cells labeled x in the middle column is odd, the number of unmarked cells has opposite parity from the original column. Thus the leftmost column contains at least one less box.

The same argument holds for the rightmost column.

It remains to show that in “fix parity” there is a cell to shift if necessary. In particular we have to show that the middle column and the leftmost one are not the same if $\mu_1 \neq 0$. Suppose therefore that they are the same.

We consider cells that might been shifted to the tail, thus to the leftmost column. Cells shifted from the rightmost column need to have bigger labels than any cells originally in the middle column as we shift the bottommost cells and there where at least as many cells labeled x in the middle column. Thus there cannot be such cells in the tail.

There cannot be cells in the tail that are shifted from the middle column to the leftmost one as those columns are the same.

Finally there can be only one cell in the tail coming from the leftmost column. This happens if there is an odd number of x in the leftmost column. However as the leftmost column and the middle one have to be the same, this is not possible due to the parity of the lengths.

Therefore the tail is \emptyset . □

Lemma 27. *Algorithm 2 produces an orthogonal Littlewood-Richardson tableau L in $\text{LR}_\lambda^\mu(\mathfrak{d})$ and a standard Young tableau Q in $\text{SYT}(\lambda)$.*

Proof. Algorithm 2 produces a standard Young tableau Q because its input \tilde{Q} is a standard Young tableau from which the largest μ_1 entries are deleted to obtain Q .

Now we consider L . We point out that L has lines filled with $1, 2, \dots, \lambda_\ell$. Moreover, there are gaps j only in the tail, or possibly in the middle column at the bottommost position. Those are always in such a way, that $j - 1$ is the last position of a column to the right. Furthermore, due to “fix parity” the parity of a and b is always even. The residuum is always less or equal 1 as there is at most one number from the original leftmost column in the tail.

Now if the rightmost column has even length, the middle one also has even length because label cells with x and mark and shift cells can only change parity if the leftmost line consists of exactly one position and this is labeled x and this does not change the parity of the rightmost line. Thus if the rightmost column has even length and if the tail has length 0, L is an orthogonal Littlewood-Richardson tableau.

If the rightmost column has odd length, there is no gap in the middle column and the middle column is shorter than the rightmost one, we show that this is Case 2, thus $c + \mu_1 < \lambda_1$. Suppose the rightmost column has odd length and $c + \mu_1 = \lambda_1$. Then there would have been no cell labeled x in the leftmost column. Thus the leftmost column with out the tail is has odd length. Thus the middle column has odd length. Thus “fix parity” is done. This has to move an element from the leftmost column to the middle one as otherwise the tail would get longer, and this has to be the topmost position of the tail. Thus we get either a gap in the middle column or a middle column that is longer than the right one.

It there is a gap in the middle column or the middle column is longer than the rightmost one, we have to show that this is Case 3, thus the residuum is 1, the middle line is only in the middle column and $a > 0$ and the rightmost column has odd length. If this

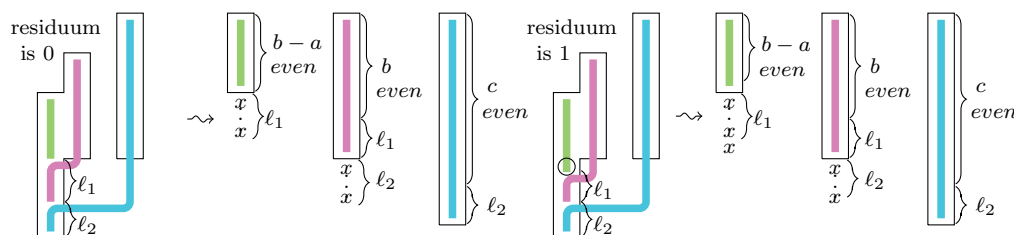
occurs, there was some change from the leftmost column to the middle one in “fix parity”. Therefore the rightmost column has odd length. Now the residuum is 1 and $a > 0$ as we shifted the leftmost column down by one. The middle line is only in the middle column as middle line elements in the leftmost column would have been smaller candidates to change column. \square

Now we show the (perhaps surprising) fact that Algorithm 1 and Algorithm 2 are inverse.

Lemma 28. *Algorithm 1 and Algorithm 2 are inverse.*

Proof. We show first, that applying Algorithm 1 on a pair consisting of an orthogonal Littlewood-Richardson tableau L and a standard Young tableau Q with $\mu \neq \emptyset$ and Algorithm 2 on the outcome (\tilde{Q}, μ) returns the original pair. This is obvious for the standard Young tableau Q as exactly the added cells are removed. For L we consider Cases 1 to 4 separately.

1. We start with Case 1. In this case the rows of \tilde{Q} have even length (because $|\tilde{Q}| = \mu_1 + |\lambda| = \mu_1 + \mu_1 + 2b - a + c$). We show that Algorithm 2 marks precisely the cells coming from the tail. The following sketch illustrates the situation just before “adjust parity” in Algorithm 1. We display the three rows of the row-strict tableau obtained from L as three separate columns and distinguish the two possible values of the residuum.



Now we have eight cases as ℓ_1 and ℓ_2 could be even or odd. We first consider the case of residuum 0 and ℓ_1 and ℓ_2 both even. In this case we mark ℓ_1 position in the middle column and ℓ_2 in the rightmost one, thus exactly those that came from the tail. If for example ℓ_1 is odd and ℓ_2 is even, we see that one cell labeled x will move to the middle column, and an even number of such positions will remain in the leftmost one. The moved position will mark one position in the middle column (there are now $\ell_2 + 1$ cells labeled x), the other ones (as there are even many left) will also mark one position each in the middle column, thus exactly those coming from the tail.

The other cases are analogous so we only consider one more example: residuum 1 and ℓ_1 and ℓ_2 both odd. Now our leftmost column has odd length $((b-a)+1+\ell_1+1)$, our middle one has even length $(b+\ell_1+\ell_2)$ and our rightmost one has odd length $(c+\ell_2)$. Thus a cell labeled x will move from the leftmost column to the rightmost one. Now there is an odd number of cells labeled x in the leftmost column (ℓ_1), an odd number of cells labeled x in the middle column (ℓ_2) and one thus an odd

number of cells labeled x in the rightmost column. Thus one cell is marked in the leftmost column, $\ell_1 - 1 + 1$ in the middle column and $\ell_2 - 1 + 1$ in the rightmost one. Those are exactly those coming from the tail.

There is no “fix parity” in Algorithm 2 in Case 1.

2. Case 2 is a little bit more complicated but similar. In this case the rows of \tilde{Q} have odd length as c is odd. To illustrate the situation we get the same sketch as above, except that c is odd.

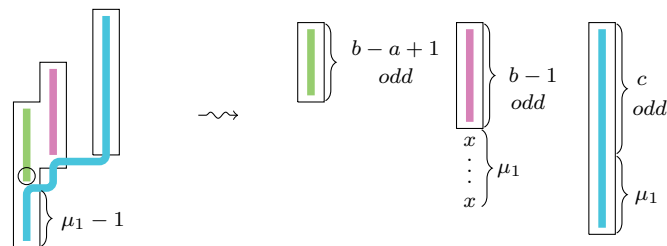
Again, we provide the details only for one of the eight different cases and leave the remaining to the reader. Let us consider the case of residuum 1, and ℓ_1 and ℓ_2 both even.

In Algorithm 1 we move a cell labeled x from the leftmost column to the middle one to obtain odd length. Now the number of cells labeled x in the middle row is $\ell_2 + 1$. Thus in Algorithm 2 there are ℓ_2 cells of the rightmost column and one cell of the middle column marked. The number of cells labeled x in the leftmost column is ℓ_1 . Thus ℓ_1 further cells are marked in the middle column.

We now have to show that after “fix parity”, exactly the ℓ_1 largest of the $\ell_1 + 1$ marked cells from the middle column end up in the tail.

Because $b - 1$, the number of remaining non-marked cells in the middle column is odd and $\mu_1 \neq 0$ we have to “fix the parity” in Algorithm 2. Thus we consider the possible candidates for such a shift. The largest entry in the leftmost column not in the tail is smaller or equal to the largest entry in the middle column because this column is longer as the residuum is 1 and thus $a > 0$. Thus this cannot change. The largest entry in the middle column cannot change either, as this would make the tail longer by 1. Thus the smallest entry that was shifted from the middle column to the leftmost one has to change to the middle column. This is exactly what we wanted to show. Note that by moving it, we also get residuum 1.

3. Case 3 is also similar but has a slightly different illustration. In this case the rows of \tilde{Q} have also odd length as c is odd and the residuum is always 1.



In this case μ_1 elements of the rightmost column are marked. We have to show, that only the largest $\mu_1 - 1$ remain in the rightmost column.

As the middle column has odd length $b - 1$ an element has to change column. The largest entry from the leftmost column not in the tail is smaller than $b - 1$ which is the largest entry in the middle column because due to the residuum $a > 0$. Thus this cannot change column. The largest entry in the middle column cannot change column either, because this would give a tail, that is longer than μ_1 . Thus the smallest element in the tail, that is the smallest one that was shifted from the middle column to the left one, changes column. This is what we wanted to show.

4. In Case 4, L is not changed at all.

Now we show that applying first Algorithm 2 on a pair (\tilde{Q}, μ) and then Algorithm 1 on the outcome returns the original pair. Therefore we have to show that we add the same positions to Q as we deleted before to obtain \tilde{Q} . As they are filled from left to right increasingly and μ is encoded in the length of the tail, this is sufficient.

Throughout this proof we arrange L like we did in the first part of the proof, namely as three columns.

The cells that are shifted to the tail in Algorithm 2 are exactly those that change column in Algorithm 1, except for the case that we have to “fix parity” in Algorithm 2.

In the former case exactly the same cells labeled x occur after Algorithm 1. One can prove this by the same case study as above. Once again we consider one case exemplarily: \tilde{Q} has rows of even length, there is one cell labeled x in the rightmost column, there is an even number e of cells labeled x in the middle column and there is an odd number o of cells labeled x in the leftmost column. Thus $e + 1$ cells are marked in the rightmost column, $o - 1$ cells are marked in the middle column and one position is marked in the leftmost column in Algorithm 2. Therefore, in Algorithm 1 there are $e + 1$ cells labeled x inserted in the middle column, and $o - 1 + 1$ inserted in the rightmost one. Thus when “adjust the parity” in Algorithm 1 one cell labeled x is shifted to the rightmost column.

In the latter case, that there is a “fix parity” in Algorithm 2 we have two cases.

If an element changes column that belonged to the leftmost column or the middle one, this does not change where the cells labeled x are added as those are added in both cases into the leftmost column.

If an element from the leftmost column is moved to the middle one that belonged to the rightmost column, this does not change anything, as this is still an element that is in the wrong column. Thus cells labeled x are put into the middle column and one of them is shifted right, if the parity demands it. \square

3.4 The second Bijection

In the second part of our bijection, we map three-rowed standard Young tableaux, all of whose row lengths have the same parity, to vacillating tableaux of shape \emptyset . Moreover, this bijection should preserve descents. This property restricts our choice, however does not define it by far. In Figure 5 in the appendix we see standard Young tableaux and the corresponding vacillating tableaux up to a size of $r = 6$.

Algorithm 3 maps standard Young tableaux with three rows of even length to vacillating tableaux of even length and shape \emptyset . As we saw in Section 3.2.1, this is sufficient

because for tableaux with rows of odd length we add three additional entries to the standard Young tableau.

We illustrate labeled words with labeled Riordan paths like we illustrate vacillating tableaux as Riordan paths. See Figure 6 in the appendix for a set of possible insertion scenarios.

Algorithm 3: standard Young tableau to vacillating tableau

```

input : standard Young tableau  $Q$  with at most 3 rows, all of them of even length
output: vacillating tableau  $V$  of shape  $\emptyset$  of even length
/* Inserting 1st row                                     */
construct word  $V = (1, -1, 1, -1, \dots, 1, -1)$  with same length as the first row of  $Q$ ;
label the letters with the numbers of the first row of  $Q$ ;
/* Inserting 2nd row                                     */
for pairs  $a, b$  in the second row of  $Q$ , starting with the rightmost pair, going left do
    insert  $-1$  at  $b$ ;                                     /* insert  $b$  */
    if position to the right of  $a$  is  $-1$  not labeled with  $b$  then           /* insert  $a$  */
        | insert  $0$  at  $a$ ; change  $-1$  to the right of  $a$  into  $0$ ;           /* case 1 */
    else
        | insert  $-1$  at  $a$ ; change next  $-1$  to the left of  $a$  into  $1$ ;       /* case 2 */
    | change pairs of  $1, -1$  between  $a$  and  $b$  into  $0, 0$ ;
/* Inserting 3rd row                                     */
for pairs  $a, b$  in the third row of  $Q$ , starting with the rightmost pair, going left do
    insert  $-1$  at  $b$ ;                                     /* insert  $b$  */
    change next  $-1$  to the left of  $b$  into  $0$ , let  $\tilde{b}$  be its label;
    start at  $b$  and let  $c$  be the current position, let  $C$  be an empty list;
    while  $\tilde{a}$  undefined do
        if  $c$  is 1 on level 0 then                                     /* connect */
            | change  $c$  and the position to the left into  $0$ , put their labels into  $C$ ;
        else if the number of positions strictly to the left is even,  $c$  and the position to the left
            are  $0$ 's on level 1 and  $a$  is not inserted then           /* separate */
            | change  $c$  into  $1$  and the position to the left into  $-1$ ;
        if  $a$  is directly to the left of  $c$  then                       /* insert  $a$  */
            | insert  $-1$  at  $a$ ;
        else if  $a$  is inserted then
            if  $c$  is  $-1$  not labeled with  $a$  then                       /* case 1 */
                | change  $c$  into  $0$ , let  $\tilde{a}$  be its label;
            else if  $c$  is  $0$ , not in  $C$ , left of  $\tilde{b}$  then           /* case 2 */
                | change  $c$  into  $1$ , let  $\tilde{a}$  be its label;
            | go one position to the left;
    forget the labeling of  $V$  and return  $V$ ;

```

Table 1: Notation for Algorithms 3 and 4

Labeled word in $\{\pm 1, 0\}$	A word in $\{\pm 1, 0\}$. Each letter of this word is labeled by an integer $1 \leq i \leq r$ strictly increasing from left to right. Thus each position consists of a label and an entry.
A position in a labeled word is “on level ℓ ”	The minimum of the sum over all entries strictly to the left and the one over all entries to the left including the one of this position is ℓ . Illustration of positions on level ℓ : $\ell \dots \swarrow \searrow \dots$
A 0 is in “3-row-position”	This 0 is either on level 2 or higher or it is the rightmost 0 of an odd sequence of 0’s on level 1.

We refer to positions by their labels and take instructions literally:

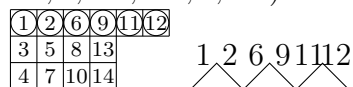
“Insert e at m ” into a given labeled word, which does not contain the label m	We insert a new position with entry e and label m such that the labels are still sorted.
“The position to the right (or left) of m ” in a labeled word, where m does not occur	The position of the right (or left) of the spot where we would insert m .

We refer to processes in the algorithms in the following way:

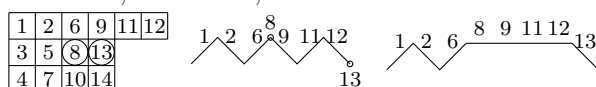
“Connect / Separate”	<p>“Connect” means that a sequence $-1, 1$ on level 0 is changed into a sequence $0, 0$ on level 1:</p> <p>$\swarrow \searrow \rightarrow \dots$</p> <p>“Separate” means that a sequence $0, 0$ on level 1 is changed into a sequence $-1, 1$ on level 0.</p> <p>This nomenclature is justified by Theorem 41.</p>
----------------------	---

We give now a full iteration of Algorithm 3 on our running example.

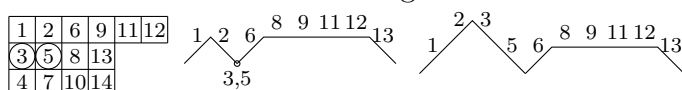
Example 29. We start with inserting the first row. There we construct the word $(1, -1, 1, -1, \dots, 1, -1)$ labeled with the elements of the first row of the tableau:



Inserting the second row we begin with case 1 at inserting a and changing the pair of $1, -1$ at $11, 12$ into $0, 0$:

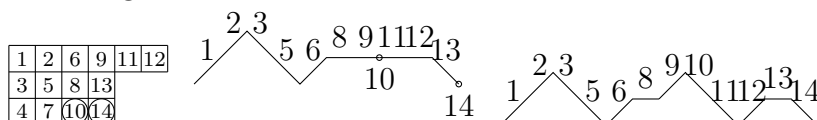


And end with case 2 at inserting a :



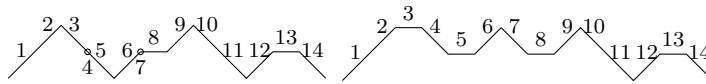
We obtained a word, where the only 0’s are in pairs on level one. We will see that this is always the case after any insertion that comes from the second row (see Lemma 35).

Inserting the third row we start with at “insert a case 2” and “separate”:



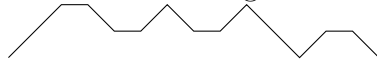
We finalize with “inserting a case 1”. We do not “connect” as \tilde{b} and b already connected the paths:

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



We see that after each insertion from the third row we obtain a new 0 in 3-row-position (that is either on level two or higher or the rightmost one in an odd sequence of 0's on level one). This is indeed always the case, see Lemma 35.

In the end we forget the labels. So what we obtain is the following path:



Algorithm 4: vacillating tableau to standard Young tableau

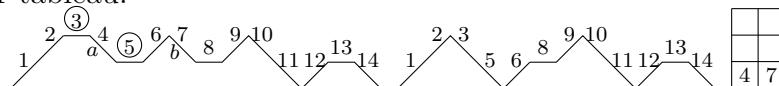
```

input : vacillating tableau  $V$  of shape  $\emptyset$  of even length  $r$ 
output: standard Young tableau  $Q$  with at most 3 rows, all of them of even length
attach labels  $\{1, 2, \dots, r\}$  to  $V$ ;
/* Extracting 3rd row */
while there are 0's in 3-row-position do
    let  $p$  be the leftmost 0 in 3-row-position and  $p'$  be the position to its right;
    if  $p$  is on level 1 and  $p'$  is a 1 then /* extract  $a$  */
        | change  $p'$  into 0, let  $\tilde{a}$  be  $p'$ ; /* case 2 */
    else
        | change  $p$  into  $-1$ , let  $\tilde{a}$  be  $p$ ; /* case 1 */
    let  $a$  be the next  $-1$  to the right be  $a$ , delete  $a$ ;
    start at  $\tilde{a}$ , let the current position be  $c$ ;
    while  $\tilde{b}$  undefined do
        if  $c$  is  $-1$  on level 0 then /* undo separate */
            | change  $c$  and the position to the right into 0, 0;
        else if the number of positions strictly to the left is odd, both  $c$  and the position to the
            right are 0 on level 1 then /* undo connect */
            | change  $c$  and the position to the right into  $-1, 1$ ;
        if  $c$  is 0 in 3-row-position then /* extract  $b$  */
            | change  $c$  into  $-1$ , let  $\tilde{b}$  be  $c$ ; let  $b$  be the next  $-1$  to the right, delete  $b$ ;
            | go one position to the right;
    insert  $a, b$  into the third row of  $Q$ ;
/* Extracting 2nd row */
while  $V \neq (1, -1, 1, -1, \dots, 1, -1)$  do
    let  $a$  be the leftmost position that is neither 1 nor on level 0; /* extract  $a$  */
    if  $a$  is  $-1$  then change the leftmost 1 on level 1 into  $-1$ ; /* case 2 */
    else change 0 to the right of  $a$  into  $-1$ ; /* case 1 */
    delete  $a$ ;
    let  $b$  be the next  $-1$  to the right of the new  $-1$ , delete  $b$ ; /* extract  $b$  */
    change 0, 0 on level 0 into 1,  $-1$ ;
    insert  $a, b$  into the second row of  $Q$ ;
/* Extracting 1st row */
insert labels still in the word into the first row of  $Q$ ;
return  $Q$ ;

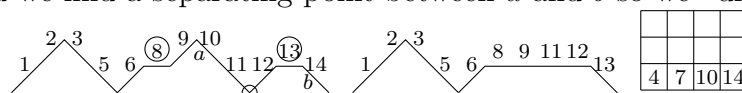
```

As we will see, everything in Algorithm 3 is reversible, and Algorithm 4 is its inverse. To illustrate the latter we use our running example and give a full iteration of it.

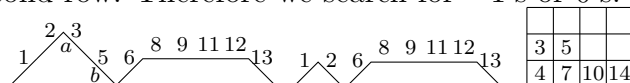
Example 30. We start searching for 0 entries in 3-row-position from left to right. The -1 entries to the right are our first a and b . This time we do not need to “undo connecting” as this is “done automatically”. We extract a and b and put them into the third row of our tableau.



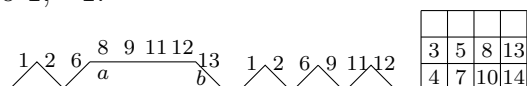
We do the same again. However, this time we change the 1 right of the first 0 into a 0 and we find a separating point between a and b so we “undo separating”:



Now we do not find 0's in 3-row-position anymore. Thus we proceed with extracting the second row. Therefore we search for -1 's or 0's:



We do it again, this time the other case occurs. Moreover, we change the 0, 0 at 11, 12 into 1, -1 :



To finish with Algorithm 4 we fill the first row with the remaining numbers:

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

We see that we obtained our original standard Young tableau.

We proceed by formulating and proving properties of Algorithm 3 and 4.

Remark 31. A (partial) standard Young tableau filled with elements of a totally ordered set produces the same vacillating tableau in Algorithm 3 as the standard Young tableau filled by $1, 2, \dots, r$ increasingly according to this totally ordered set. Thus the crucial information in the standard Young tableau is the relative order of the entries.

From this point of view, one can stop after any insertion of a pair a, b and obtains the vacillating tableau corresponding to the tableau only consisting of already inserted entries.

Lemma 32. *Algorithms 3 and 4 are well-defined.*

Proof. To show that Algorithm 3 is well-defined we need to show that a -1 exists left of a when inserting the second row in case 2 and that \tilde{b} and \tilde{a} exist when inserting the third row.

For inserting the second row consider the numbers directly above a and b in the tableau; call them c and d . We will see inductively that a and b may change the positions labeled with c and d but nothing to the left, thus nothing with smaller labels. There are two cases: $c < a < d < b$ (this corresponds to “insert a case 1”) and $c < d < a < b$ (this can happen in both cases). In both cases we know that c and d were not changed before. So they are still 1, -1 and can be used.

Analogously, for inserting the third row we consider the numbers directly above a and b . The entry above b has been inserted as -1 . The entry above a either as 0 or as -1 .

To show that Algorithm 4 is well-defined we need to show that \tilde{b} exists in extracting the third row provided that there is a 0 in 3-row-position. This holds because after deleting a and changing \tilde{a} there are an odd number of 0's left as the vacillating tableau needs to have even length. \square

Next we show that our algorithms produce the desired output.

Lemma 33. *Algorithm 3 produces a vacillating tableau of even length and shape \emptyset .*

Proof. We show this inductively. After inserting the first row we always obtain the word $(1, -1, 1, -1, \dots, 1, -1)$ which is a vacillating tableau. Moreover, after every insertion of a pair (a, b) , the sum over all entries is zero. Thus it remains to show that the algorithm neither produces any 0 at level zero or lower nor any ± 1 at level minus one or lower. We show that this is the case after every insertion of a pair (a, b) .

The only steps in the algorithm at which the level of a position decreases occur when pairs $(1, -1)$ between a and b are changed into $(0, 0)$, or when “separating”. In the first situation, the level is increased between a and b , so this does not produce a 0 on level zero. In the second situation, pairs $(0, 0)$ at level one are changed into $(-1, 1)$ at level zero. \square

Lemma 34. *Algorithm 4 produces a standard Young tableau with at most three rows, all of even length.*

Proof. The row lengths are even because elements are extracted in pairs. It remains to show that the row lengths are weakly decreasing and the tableau is standard.

Suppose we extract a pair (a, b) into the third (respectively second) row. We define two related elements $a' < b'$ with $a' < a$ and $b' < b$ which will be extracted into the second (respectively first) row and end up weakly right of a and b .

We consider the second row first. We define b' to be the -1 created just before deleting a . The element a' is the 1 directly left of b' . Thus (a', b') is a pair $(1, -1)$ on level zero, such that everything to the left is also on level zero. Therefore it is extracted into the first row.

Now we consider the third row. Suppose \tilde{a} is -1 , then we define a' to be \tilde{a} . Otherwise, if \tilde{a} is 0, we distinguish two cases. If there is no -1 extracted into the second row left of \tilde{a} or the next -1 to the left is extracted into the second row as b then a' is the leftmost 0 in the sequence of 0's of \tilde{a} . Otherwise it is the next -1 to the left. Similarly, if \tilde{b} is a -1 on level one or higher, we define b' to be \tilde{b} . Otherwise, if it is a -1 on level zero, we

distinguish two cases. If there is a -1 directly to the left of b , that is extracted as b into the third row, we define b' to be this -1 and otherwise we define b to be \tilde{b} .

Finally we show that a' and b' get extracted into the second row. Therefore we describe the elements that are extracted into the second row. -1 's on level one or higher are always such elements. -1 's on level zero are extracted to the second row if and only if there is a -1 to the left that is extracted as a or there is a 0 directly to the left. 0 's are extracted if and only if they are the leftmost in their sequence of 0 's and the -1 to the left (if it exists) is extracted as b . \square

The following lemma is one of the key observations to show that Algorithms 3 and 4 are inverse.

Lemma 35. *In Algorithm 3, after inserting a pair (a, b) from the second row, any 0 's come in pairs on level one. Any third row insertion produces 0 's in 3-row-position.*

Proof. We consider inserting the second row first. There are two possible ways to produce 0 's when inserting the second row: at “inserting a case 1”, and changing pairs $1, -1$ into $0, 0$. Both produce them in pairs. Moreover, those are always right of the current a . As the next insertion happens strictly left of the next b , that is itself left of a , they cannot be isolated later by inserting the second row.

Now we consider inserting the third row. Elements are inserted decreasingly, from right to left. Therefore left of the leftmost \tilde{a} so far there are no 0 's in 3-row-position. Moreover, “separate” and “connect” only changes or produces pairs such that an even number of positions and thus an even number of 0 's is to the left. Now we consider \tilde{a} and \tilde{b} of a pair to insert. If those are on level two or higher, they are 0 s in 3-row-position. If \tilde{b} is a 0 on level one, it needs to be between a and b . If a is the -1 to end the sequence of 0 s of b it could be the case that this sequence is even. In this case, the leftmost 0 of this sequence is thus at level two. Moreover \tilde{a} is either another 0 on level two or a 1 next to an odd number of 0 s on level one, as it deleted one. Now \tilde{a} either produced a new 0 or it deleted one. In the former case, if this is on level one, it needs to be either a single 0 or part of a sequence that was there before. Thus, that sequence was even before adding a 0 . In the latter case this deleted a 0 that was in an even sequence of 0 's, hence is now a new odd sequence of 0 's. \square

Lemma 36. *Algorithms 3 and 4 are inverse.*

Proof. Due to Lemma 35 and the fact that Algorithm 4 extracts third row elements only as long as there are 0 's in 3-row-position, it suffices to show that inserting and extracting the second (respectively third) row are inverse.

This is obvious for the second row. For the third row we point out that due to “connect” and “separate” between a and b in the third row a return to level zero between a and b can only be at odd points as we “connect” at even points and “separate” at odd ones. \square

We continue proving the following important property.

Lemma 37. *Algorithm 3 is descent-preserving.*

Proof. We show that the algorithm preserves descents after every insertion of a pair a, b in the sense that we consider the inserted numbers as a new total order.

In the first step we show that when we insert a pair (a, b) , a and b cause a descent except for the case that a and b are neighbors in the set of already inserted numbers. To see why we want this to hold, we consider the partial standard Young tableau consisting only of already inserted numbers. The number smaller than a needs to be in a row below a as numbers in the same row to the right are bigger. The same holds for b except if a and b are neighbors in the current order.

In the case that a and b are neighbors, they are both inserted as -1 's and we have to show that only a is a descent. Thus everything else is analogous to the general case.

If a or b is inserted as -1 the only way that this causes no descent is that the position to the left is a -1 or a 1 on level zero. A -1 directly to the left of a would change because it becomes an \tilde{a} when inserting the second row. A -1 directly to the left of b would change into 0 , when inserting the second row. When inserting the third row, this -1 would be \tilde{a} or \tilde{b} and thus changed into 0 . A 1 on level zero left of a or b would cause a to be inserted as 0 when inserting the second row and would be not on level zero anymore in any other case.

In the second step we show that we do not lose descents when inserting a pair (a, b) . If an entry was a descent in the partial tableau before inserting (a, b) it is still a descent in the new partial tableau, either with the same number above, or with a or b . In the former case neither a nor b are inserted inbetween those. In the latter case either a or b is inserted inbetween. This creates a descent in the vacillating tableau and removes the other descent as $(-1, x)$ can never be a descent. Choosing \tilde{a} or \tilde{b} changes a -1 into a 0 or a 0 into a -1 . The latter cannot cancel a descent as the position left of this 0 needs to be also a 0 . The former could only cancel a descent if there is a 0 directly to the left. This 0 needs to be on level one and therefore we “separate”. Thus the descent $(0, -1)$ is changed into the descent $(1, 0)$.

What remains to be shown is that “separate”, “connect” and “change $(1, -1)$ into $(0, 0)$ ” when inserting the second row preserves the descent set. For the latter this is clear as $(1, -1)$ on level zero is not a descent. For “separate” we consider a 0 left or right of a position that was changed in “separate”. Those need to be either \tilde{a} or \tilde{b} or they were changed in “connect”, because otherwise they would have been “separated” also. In the former case we want a descent, in the latter too, as $(1, 0)$ has changed into $(0, -1)$ or the other way around. The same holds for 1 's or -1 's of “connect”. \square

The following two lemmas about Algorithm 3 are essential for our bijection and hold as the algorithm is descent-preserving (see Lemma 37) and defines a bijection (see Lemma 36).

Lemma 38. *Let $e_1 < e_2 < e_3$ be the three biggest entries in the standard Young tableau \tilde{Q} . The last three positions of the corresponding vacillating tableau are $1, 0, -1$ if and only if \tilde{Q} has three non-empty rows and e_i is the last position of row i for $1 \leq i \leq 3$.*

For the next lemma we point out that there are $1, -1$ on level zero in the end of a vacillating tableau if and only if the two largest numbers in the corresponding standard Young tableaux are in the first row.

Lemma 39. *The skew tableau formed by the cells containing the m largest entries of a standard Young tableau are a horizontal strip with at most one box in the first row that is filled increasingly from left to right, if and only if the last m positions of the corresponding vacillating tableau are a sequence of -1 's.*

We complete this section by stating further properties of Algorithm 3. To do so we need the concept of concatenating tableaux.

Definition 40. We obtain the *concatenation* Q of two standard Young tableaux Q_1 and Q_2 in the following way. First we add the largest entry of Q_1 to each entry of Q_2 to obtain the tableau \widetilde{Q}_2 . Then we append the i^{th} row \widetilde{Q}_2 to the i^{th} row of Q_1 to obtain Q .

The *concatenation* of two vacillating tableaux is obtained by concatenating the words.




To see if a standard Young tableau Q can be written as concatenation of two standard Young tableaux Q_1 and Q_2 , one has to check every possible Q_1 . Thus there is no easy way to see this.

Algorithm 3 has a nice property concerning concatenation that we prove later, using an alternative formulation of Algorithm 4. The following theorem follows directly from Corollary 52 and the fact that Algorithm 3 defines a bijection (see Lemma 36).

Theorem 41. *Considering Algorithm 3, concatenation of vacillating tableaux of empty shape corresponds to concatenation of standard Young tableaux where all row lengths have the same parity.*

In particular, the following holds:

- *If a vacillating tableau is composed of two concatenated paths with weight 0, its corresponding standard Young tableau can be written as concatenation of two standard Young tableaux, such that each of those has row lengths of the same parity.*
- *On the other hand if a standard Young tableau can be written as concatenation of two standard Young tableaux, such that each of those has row lengths of the same parity, its corresponding vacillating tableau is also composed of two concatenated paths with weight 0.*

Example 42. Concatenating the vacillating tableaux  and  yields the vacillating tableau . Whereas concatenating the corresponding standard Young tableaux

1
2
3

 and

1	2
3	4

 yields the corresponding standard Young tableau

1	4	5
2	6	7
3		

.

Remark 43. This is another reason why we attach elements $e_1 < e_2 < e_3$ to a standard Young tableau Q where e_i is i plus the largest entry of Q to deal with tableaux with rows

3.5 Properties of the Bijection

Theorem 47. *The map defined in Section 3.2.1 is*

1. *well-defined,*
2. *has a reversed mapping and thus is a bijection between vacillating tableaux and pairs consisting of a standard Young tableau and an orthogonal Littlewood-Richardson tableau,*
3. *descent-preserving.*

Proof. All we need is stated in the Sections 3.3 and 3.4.

We point out that the second statement of Lemma 25, Lemma 38 and Lemma 39 play a crucial role when proving this for the parts of the bijection that are not described by Algorithm 1 or 3. \square

3.6 Alternative Formulation of the Second Bijection

In this section we present Algorithm 5, an alternative formulation for the inverse algorithm of the second bijection (Algorithm 4) which gives us a better understanding of Algorithms 4 and 3 and proves some further properties.

Algorithm 5: vacillating tableaux to standard Young tableaux: alternative algorithm

input : vacillating tableau V in terms of a Riordan path
output: standard Young tableau Q
 we build a graph, whose vertices are the steps of our Riordan path;
 we allow at most one incoming and at most one outgoing edge for each step; moreover, it is not allowed for an edge, that starts (respectively ends) at a step that has an incoming and an outgoing edge, to end (respectively start) at such a step also; we call those edges “double edges”;
 moreover, some (non double) edges might be marked;
for entries i in V starting with the leftmost going to right **do**
 insert (half)edges for entry i between the entries of V as described in Table 2;
 connect as many (half)edges as possible with respect to the following rules (any such connecting is admissible);
 always connect outgoing edges with incoming edges and vice versa, moreover
 outgoing edges can only be connected to incoming ones to the right;
 outgoing edges from a -1 cannot be connected with incoming edges unless there are no open double or marked edges;
 incoming edges of 0 ’s are not allowed to be connected with open double or marked edges;
 put i in the row of Q according to Table 2;
return Q ;

Table 2: Conditions for Algorithm 5

entry	conditions			kind of edge(s)	row
1	always (no 0 is active any more)			outgoing	1
0	# open double or marked edges +1 < level			incoming and outgoing	2
	# open double or marked edges +1 = level (this 0 is active, as well as 0's right from here until the next 1 / end of word)	level and # closed edges have different parity	# active 0's odd	incoming and outgoing	2
			# active 0's even	incoming	2
		level and # closed edges have same parity	# active 0's odd	marked outgoing	1
			# active 0's even	outgoing	1
	# open double or marked edges +1 > level			outgoing	1
-1	# open double or marked edges > 0			incoming	3
	# open double or marked edges = 0	level > 0		incoming	2
		level = 0	# closed edges odd	incoming	2
			# closed edges even	outgoing	1

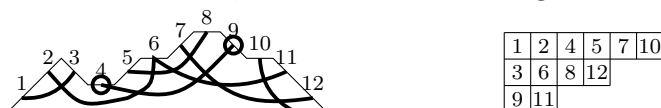
Theorem 48. *Algorithm 5 is well-defined and equivalent to Algorithm 4 for vacillating tableaux of shape \emptyset and even length.*

Before we prove this we give some examples.

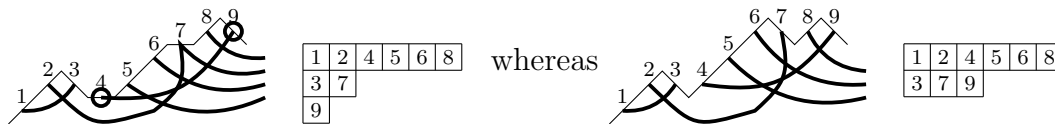
Example 49. In our running example we can obtain the standard Young tableau directly. (Even though we do not need to, we label the edges for better readability.)



Example 50. We give another example where a marked edge occurs and the vacillating tableau has weight \emptyset . We point out that while ordinary edges can be open when the path returns to level zero, double or marked edges are never open at such a point.



Example 51. This example shows the crucial point in counting open double or marked edges. Note that before inserting 9, the tableaux, the height, and the next position to consider are the same.



Proof of Theorem 48. We start this proof with providing properties of Algorithm 5, that prove among others that the algorithm is well-defined:

- Only the first active 0 of a sequence of active 0's can introduce an open double or marked edge or an incoming-only edge.

To see this we show that once an active 0 occurs the next active 0 in the same sequence will be outgoing-only, as well as once an active 0 has an outgoing-only edge, until the next 1, all active 0's have outgoing-only edges.

Once there is an outgoing double or marked edge the height condition together with the rule that such edges can only be closed at -1 's ensure that for all following active 0's the the third case for the level will be chosen.

Once there is an incoming-only edge the number of closed edges increases by one. Thus the parity changes. A height decrease also increases the number of closed edges such that we stay in the case of same parity. Moreover, because the number of active 0's is even, which is necessary for the incoming-only case, we stay in the case of outgoing-only edges.

Once there is an outgoing-only case the number of closed edges only increases at a loss of height. Now this is analogous to the former argument.

- Exactly in those steps where there is an incoming (or of course incoming and outgoing) edge introduced, there is exactly one connection of (half)edges. This is between a former outgoing edge and this new incoming one. In any other case it is not possible to connect (half)edges.

To see this we show that there are always enough outgoing open edges. Every 1 produces an outgoing edge. Every -1 (except for the last one maybe) has an incoming one. Therefore without 0's this is regulated automatically with the level. The only 0 case where the number of outgoing edges is reduced is the one where there is only an incoming edge. However in this case the number of active 0's is even and due to the former point we know that an outgoing-only edge will follow before reaching the bottom level.

- There are never open double or marked edges when the vacillating tableau reaches a bottom point, thus level zero. This holds because there cannot be such new edges when there are as many open ones as the height and those edges need to be closed before an ordinary edge can be closed by a -1 .
- The order in which the edges are closed makes no difference. Thus we can assume some first-come-first-served principle to connect (half)edges, and do not consider open (half)edges left of the last 1 to the left on level zero. Moreover, this ensures that the map defined by the algorithm is well-defined.

Next we show that the theorem holds for vacillating tableaux without 0's in 3-row-position. Those are mapped to two-rowed standard Young tableaux in Algorithm 4.

In this case we have no 0's on a level higher than one and those at level one are always together in even numbers. Therefore we never have any double or marked edges. Thus nothing is put into the third row.

If we go through the path from left to right and find a -1 not on level zero, left of the leftmost 0, if a 0 exists, we will connect this with some 1. If we find a 0 left of the leftmost -1 not on level zero this will connect with the 1 to the left. In both cases it will get to the second row and be our a , exactly as it would be in Algorithm 4.

As the number of closed edges is now odd the next closed edge will end in a -1 which will also get to the second row and therefore be our b , exactly as it would be in Algorithm 4. Now it makes no difference if we extract those or not, because the number of closed edges is even now, there is a new area of active 0's and every other statistic is the same.

Moreover, -1 's that belong to the first line are treated correctly as those are at level zero, and will not be considered as new a when the number of closed edges are even.

Next we show that the algorithm finds the 0's in 3-row-position and their associated -1 's. It thus follows that the right numbers are put into the third row. We do this inductively.

0's that are at level two or higher when executing Algorithm 4 are recognized automatically. In Algorithm 4 there would be a level decrease of 1 between \tilde{a} (respectively \tilde{b}) and a (respectively b). Therefore open double or marked edges increase the needed height. The next -1 not already used by other 0's in 3-row-position is used for a (respectively b).

0's that are on level one, part of a sequence of 0's of odd length, are also recognized. Those are the first positions of active 0's as the level condition shows: $1 = \text{level} - \#$ of open double or marked edges.

Finally we show that it makes no difference for the two-row-case of Algorithm 5 if we do not carry out changes done by extracting the third row. When extracting a there are three cases:

- \tilde{a} is a non active 0. It becomes a -1 on level one or higher, thus still is on level one or higher.
- \tilde{a} is an active 0. It becomes a -1 on level zero. Whether this is placed in the second or the first row depends in both situations on the parity of closed edges.
- \tilde{a} is a 1. Thus to the right there is an active 0 and the number of active 0's is odd. Again whether this is placed in the second or the first row depends in both situations on the parity of closed edges.

Extracting a changes the parity of closed edges. Thus in the situations where there is a dependency on the parity of closed edges, this change is considered.

When "undoing connect" this happens between two active 0's. While the right one is always put into the first row, there are two possible ways for the left ones (as well as there are for -1 's on level zero), depending on the parity of closed edges. Again everything

holds due to the change of the parity when extracting a . The same argument holds for “separate”.

When extracting b there are two cases that are analogous to the first two cases when extracting a . Note that if $\tilde{a} < \tilde{b} < a < b$ the level and the number of open double or marked edges is reduced by one.

After extracting a and b the parity of open double or marked edges is the same as before as well as the parity of closed edges. Moreover, there is no conflict with active 0's as in an area of active 0's there can be at most one \tilde{a} or \tilde{b} . \square

Algorithm 5 proves the property about concatenation of tableaux we have stated before in Theorem 41. As there cannot be an open double or marked edge when the path reaches the bottom (thus has steps on level zero) due to level conditions, concatenated parts of paths are independent of each other.

Corollary 52. *Concatenation of paths with weight 0 corresponds to concatenation of standard Young tableaux, all of whose row length have the same parity, when applying Algorithm 5. Therefore this also holds for Algorithm 3 and Algorithm 4.*

We consider now another alternative formulation.

Corollary 53. *Algorithm 6 is an alternative formulation of Algorithm 5. For an entry to insert, it only needs the current entry and four statistics we can calculate using only entries that are already inserted.*

Proof. Algorithm 6 satisfies the mentioned properties due to its formulation.

We see that Algorithm 6 is equivalent to Algorithm 5 when pointing out that:

- a_0 in Algorithm 6 is the number of active 0's in Algorithm 5.
- odm in Algorithm 6 is the number of open double or marked edges from Algorithm 5.
- h in Algorithm 6 is the height before some part of the path. Thus for a 0 or a -1 it is the level and for a 1 it is the level minus one. Although we use the level in Algorithm 5, this does not matter as the neither the height nor the level is considered for a 1 in any of the two algorithms.
- ce is the number of closed edges in Algorithm 5.

The main difference between Algorithm 5 and 6 is that in the latter the decision if an active 0 has an outgoing edge such that this is part of a double edge or a marked edge, is done at the next 1 or the next -1 where the path touches the bottom. However, this makes no difference because for active 0's it holds that for every -1 that is between them, there is another open double or marked edge that can connect with the -1 . Besides, only -1 's on level zero interrupt a sequence of active 0's. (This explains the $h = 0$ condition for $odm = odm + 1$.)

Moreover, only one active 0's can have a double or marked edge. This holds because once this occurs in the original algorithm, the number of those edges increases and so the last case for 0's occurs until the next 1. \square

Algorithm 6: vacillating tableaux to standard Young tableaux: second alternative algorithm

```

input : vacillating tableau  $V$  in terms of a Riordan path
output: standard Young tableau  $Q$ 
 $a_0 = odm = h = ce = 0$ ;
for entries  $i$  in  $V$  starting with the leftmost going right do
  switch entry do
    case 1 do
      if  $a_0$  odd then  $odm = odm + 1$ ;
       $h = h + 1$ ;  $a_0 = 0$ ; put  $i$  in the first row of  $Q$ ;
    case 0 do
      switch  $odm$  do
        case  $< h - 1$  do  $odm = odm + 1$ ;  $ce = ce + 1$ ; put  $i$  in the second row of  $Q$ ;
        case  $= h - 1$  do
           $a_0 = a_0 + 1$ ;
          if  $h \not\equiv ce \pmod{2}$  then  $ce = ce + 1$ ; put  $i$  in the second row of  $Q$ ;
          else put  $i$  in the first row of  $Q$ ;
        case  $> h - 1$  do  $a_0 = a_0 + 1$ ; put  $i$  in the first row of  $Q$ ;
    case  $-1$  do
       $h = h - 1$ ;
      if  $a_0$  odd and  $h = 0$  then  $odm = odm + 1$ ;  $a_0 = 0$ ;
      switch  $odm$  do
        case  $> 0$  do  $odm = odm - 1$ ;  $ce = ce + 1$ ; put  $i$  in the third row of  $Q$ ;
        case  $= 0$  do
          switch  $h$  do
            case  $> 0$  do  $ce = ce + 1$ ; put  $i$  in the second row of  $Q$ ;
            case  $= 0$  do
              if  $ce$  odd then  $ce = ce + 1$ ; put  $i$  in the second row of  $Q$ ;
              else put  $i$  in the first row of  $Q$ ;
  return  $Q$ ;

```

Remark 54. One might think that this could be again rewritten in terms of an automaton. However as we need h and odm to be any positive natural number this could not be realized with a finite one. (As we need only the parity of the other statistics those could be implemented as boolean.)

Remark 55. Furthermore, due to Corollary 53 we get:

- For a vacillating tableau of any shape Algorithm 6 produces the right standard Young tableau. For the information we need to create the orthogonal Littlewood-Richardson tableau, we have to add -1 's until shape \emptyset is reached as we do in the reversed bijection.
- For vacillating tableaux of even length and shape \emptyset we do not need to add $1, 0, -1$ and delete the corresponding entries of the standard Young tableau afterwards.

Corollary 56. *Algorithm 6 provides an insertion algorithm in the sense of [11, Theorem 6.2 (2)]: if the vacillating tableau $(v_1, v_2, \dots, v_{m-1}, v_m)$ is mapped to the standard Young tableau Q of size m , the vacillating tableau $(v_1, v_2, \dots, v_{m-1})$ is mapped to the standard Young tableau \bar{Q} of size $m - 1$ which is obtained by deleting the biggest entry m .*

4 Outlook

4.1 Sketch of the extension to $\mathrm{SO}(2k + 1)$

For higher dimensions we consider vacillating tableaux as tuples of paths with certain dependencies. We use an inductive approach to obtain an insertion algorithm for these paths. We start with the first row as in the $\mathrm{SO}(3)$ case. Then we insert rows in pairs using a variant of the insertion algorithm for the second and third row presented here. However we need to change the “separating” and “connecting” parts of the algorithm due to the dependencies between the paths.

As this algorithm is far more complicated it will be presented in another paper.

4.2 Open problems

It would be interesting to do the same for the vector representation of other groups:

- $\mathrm{SO}(n)$, n even, vector representation - there is already a descent set conjectured by Rubey.
- G_2 , vector representation - there is already a descent set conjectured by Rubey.
- $\mathrm{Sp}(n)$, n even, vector representation using Kwon’s instead of Sundaram’s symplectic Littlewood-Richardson tableaux.

Acknowledgements

The author would like to thank Martin Rubey for valuable discussions and helpful comments on preliminary versions of this article. The author also thanks Sheila Sundaram for her ongoing interest on this work and for her helpful comments and also Cristian Lenart for his helpful comments. Moreover the author thanks an anonymous referee for a very careful reading of the paper and corrections and comments. We used the computer algebra system SageMath to implement the algorithms and test examples.

References

- [1] Jin Hong and Seok-Jin Kang. *Introduction to quantum groups and crystal bases*, volume 42 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2002.

- [2] Masaki Kashiwara. On crystal bases. In *Canadian Mathematical Society Conference Proceedings*, 1995.
- [3] C. Krattenthaler. Bijections between oscillating tableaux and (semi)standard tableaux via growth diagrams. *J. Combin. Theory Ser. A*, 144:277–291, 2016.
- [4] Jae-Hoon Kwon. Super duality and crystal bases for quantum ortho-symplectic superalgebras II. *J. Algebraic Combin.*, 43(3):553–588, 2016.
- [5] Jae-Hoon Kwon. Combinatorial extension of stable branching rules for classical groups. *Trans. Amer. Math. Soc.*, 2018.
- [6] Soichi Okada. Pieri rules for classical groups and equinumeration between generalized oscillating tableaux and semistandard tableaux. *Electron. J. Combin.*, 23(4):#P4.43, 2016.
- [7] Robert A. Proctor. A Schensted algorithm which models tensor representations of the orthogonal group. *Canad. J. Math.*, 42(1):28–49, 1990.
- [8] Thomas Walton Roby, V. *Applications and extensions of Fomin’s generalization of the Robinson-Schensted correspondence to differential posets*. ProQuest LLC, Ann Arbor, MI, 1991. Thesis (Ph.D.)–Massachusetts Institute of Technology.
- [9] Martin Rubey, Bruce E. Sagan, and Bruce W. Westbury. Descent sets for symplectic groups. *J. Algebraic Combin.*, 40(1):187–208, 2014.
- [10] Richard P. Stanley. *Enumerative combinatorics. Vol. 2*, volume 62 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 1999.
- [11] John R. Stembridge. Rational tableaux and the tensor algebra of \mathfrak{gl}_n . *J. Combin. Theory Ser. A*, 46(1):79–120, 1987.
- [12] Sheila Sundaram. *On the combinatorics of representations of the symplectic group*. ProQuest LLC, Ann Arbor, MI, 1986. Thesis (Ph.D.)–Massachusetts Institute of Technology.
- [13] Sheila Sundaram. Orthogonal tableaux and an insertion algorithm for $\mathrm{SO}(2n + 1)$. *J. Combin. Theory Ser. A*, 53(2):239–256, 1990.

A Appendix

Table 3: List of all tableaux with $n = 3$, $r \leq 3$. Note that there is not necessarily an orthogonal Littlewood-Richardson tableau for every combination of $\mu \leq \lambda$. Moreover, if we have an orthogonal Littlewood-Richardson tableau consisting of only one column, it is important if this is the tail or the single column tableau.

λ	μ	L	Q	\tilde{Q}	V
$r = 1$					
(1)	(1)	$\boxed{1}$	$\boxed{1}$	$\boxed{12}$	\diagup
$r = 2$:					
(1, 1)	(1)	$\boxed{1} \quad \boxed{1}$	$\boxed{1} \atop \boxed{2}$	$\boxed{1} \atop \boxed{2} \atop \boxed{3}$	$\diagup \quad \diagup$
(2)	\emptyset	$\boxed{1} \atop \boxed{2}$	$\boxed{12}$	$\boxed{12}$	$\diagup \quad \diagdown$
	(2)	$\boxed{1} \atop \boxed{2}$	$\boxed{12}$	$\boxed{12} \atop \boxed{34}$	$\diagup \quad \diagup$
$r = 3$:					
(1, 1, 1)	\emptyset	$\boxed{111} \quad \boxed{1}$	$\boxed{1} \atop \boxed{2} \atop \boxed{3}$	$\boxed{1} \atop \boxed{2} \atop \boxed{3}$	$\diagup \quad \diagup \quad \diagup$
(2, 1)	(1)	$\boxed{1} \quad \boxed{1} \atop \boxed{2}$	$\boxed{12} \atop \boxed{3}$	$\boxed{12} \atop \boxed{34}$	$\diagup \quad \diagup \quad \diagup$
			$\boxed{13} \atop \boxed{2}$	$\boxed{13} \atop \boxed{24}$	$\diagup \quad \diagup$
	(2)	$\boxed{1} \quad \boxed{1} \atop \boxed{2}$	$\boxed{12} \atop \boxed{3}$	$\boxed{125} \atop \boxed{3} \atop \boxed{4}$	$\diagup \quad \diagup \quad \diagup$
			$\boxed{13} \atop \boxed{2}$	$\boxed{135} \atop \boxed{2} \atop \boxed{4}$	$\diagup \quad \diagup \quad \diagup$
(3)	(1)	$\boxed{3} \quad \boxed{1} \atop \boxed{2}$	$\boxed{123}$	$\boxed{1234}$	$\diagup \quad \diagup \quad \diagup$
	(3)	$\boxed{1} \atop \boxed{2} \atop \boxed{3}$	$\boxed{123}$	$\boxed{1236} \atop \boxed{45}$	$\diagup \quad \diagup \quad \diagup$

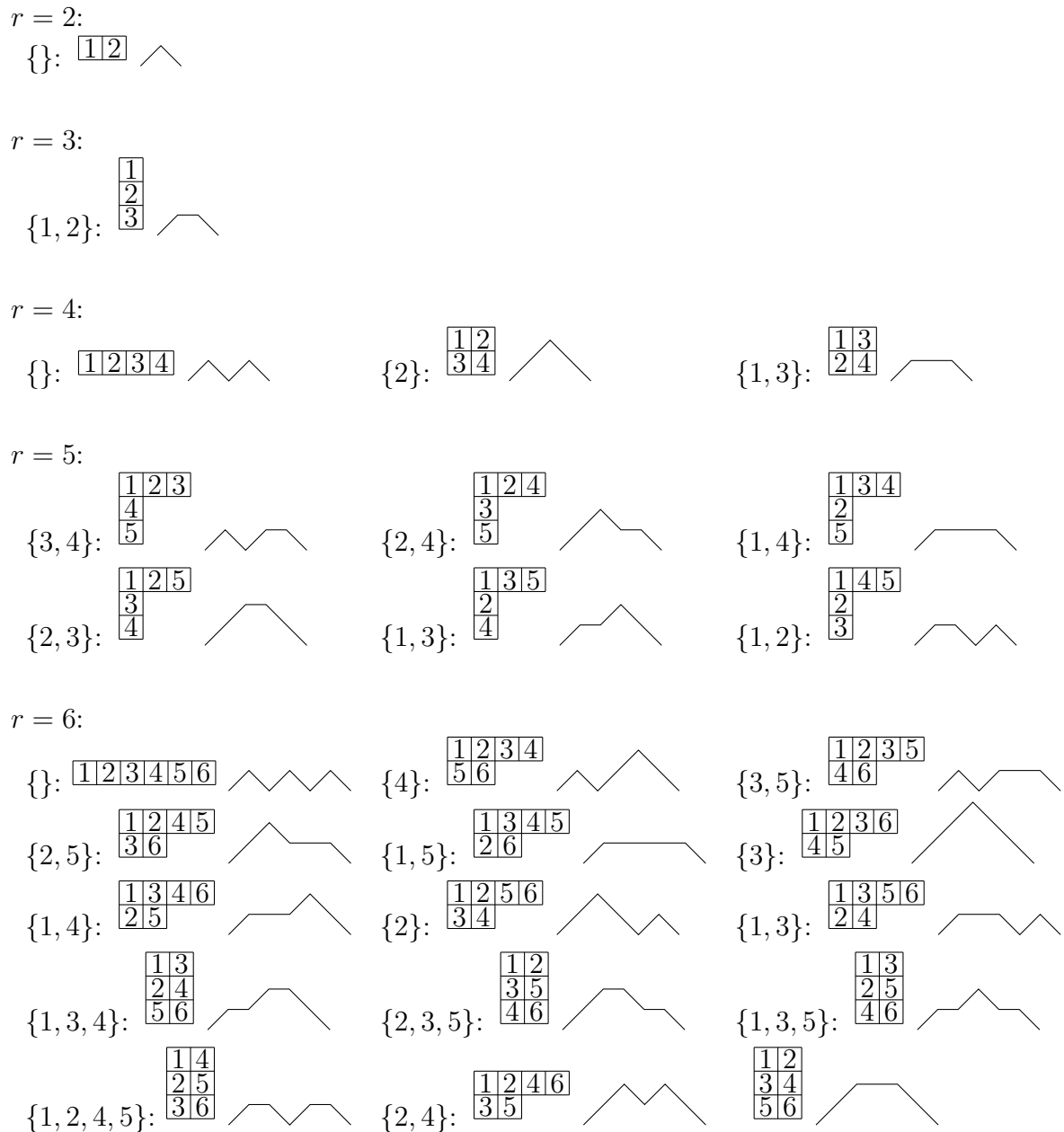


Figure 5: Standard Young tableaux with at most 3 rows, all even / odd length and vacillating tableaux up to $r = 6$ sorted by their descents. The smallest case with a non unique descent set appears at $r = 6$. The assignment we have here corresponds to our mapping.

Insert row 2:

Insert b , insert a case 1:		
Insert b , insert a case 2:		

Insert row 3:

$a < \tilde{b}$:

Insert b :		
Connect:	$\tilde{a} \quad \tilde{b}$ -- level 0	$\tilde{a} \quad \tilde{b}$ level 1
Separate:	$a \quad \tilde{b}$ even (excluding a) level 1	$a \quad \tilde{b}$ -- level 0
Insert a :	case 1 case 2 	case \tilde{a} 1 case 2
	case 1 case 2 	case \tilde{a} 1 case 2
	case 1 case 2 	case \tilde{a} 1 case 2

$a > \tilde{b}$:

Insert a and b :		
	6 cases as above in insert a	6 cases modified as above

Figure 6: Possible insertion scenarios