LASCOUX EXPANSION OF THE PRODUCT OF A LASCOUX AND A STABLE GROTHENDIECK

GIDON ORELOWITZ AND TIANYI YU

ABSTRACT. This paper gives a tableau formula for expanding the product of a Lascoux polynomial and a stable Grothendieck polynomial into Lascoux polynomials. Lascoux and stable Grothendieck polynomials are inhomogeneous analogues of key polynomials and Stanley symmetric functions, respectively. Our formula refines the K-theoretic Littlewood-Richardson rule of Buch and extends the key expansion of key times Schur established by Haglund, Luoto, Mason, and van Willigenburg. Our proof is combinatorial, relying heavily on a novel row insertion algorithm of Huang, Shimozono and Yu.

1. INTRODUCTION

Fix $n \in \mathbb{Z}_{>0}$ throughout this paper. We consider two families of inhomogeneous polynomials. Both families have positive integer coefficients and involve variables $\beta, x_1, \cdots, x_n$.

- Introduced by Lascoux [Las03], the *Lascoux polynomial* $\mathfrak{L}_\alpha^{(\beta)}$ is indexed by a *weak composition* α, a sequence of n non-negative integers.
- Introduced by Fomin and Kirillov [FK96], the *stable Grothendieck polynomial* $G_w^{(\beta)}$ is indexed by a permutation $w \in S_+$, the set of permutations of $\mathbb{Z}_{>0}$ where only finitely many numbers are permuted. The polynomial $G_w^{(\beta)}$ involves β and $x_1, x_2, \cdots$. We let $G_w^{(\beta)}(x_1, \cdots, x_n)$ be the polynomial obtained by setting $x_{n+1} = \cdots = 0$ in $G_w^{(\beta)}$.

We give a tableau formula that expands $\mathfrak{L}_\alpha^{(\beta)} \times G_w^{(\beta)}(x_1, \cdots, x_n)$ into Lascoux polynomials, where the coefficients are positive integers multiplied by a power of β. Our expansion simultaneously extends the following two results. Both results are generalizations of the famous Littlewood-Richardson rule that gives the Schur expansion of the product of two Schur polynomials s_λ.

(1) Introduced by Demazure [Dem74], the key polynomials κ_α, are characters of the Borel subgroup B of upper triangular matrices in GL_n. They can be viewed as non-symmetric generalization of Schur polynomials s_λ: when $\alpha = (\alpha_1, \cdots, \alpha_n)$ is weakly increasing, κ_α agrees with $s_{(\alpha_n, \cdots, \alpha_1)}(x_1, \cdots, x_n)$ [Dem74]. Haglund, Luoto, Mason, and van Willigenburg [HLMvW11] established a non-symmetric refinement of the Littlewood-Richardson rule: They expanded $\kappa_\alpha \times s_\lambda(x_1, \cdots, x_n)$ into key polynomials using skyline fillings.

(2) In enumerative geometry, Schur polynomials represent Schubert classes in the cohomology ring of the Grassmannian. When w is a Grassmannian permutation, the stable Grothendieck polynomial $G_w^{(\beta)}$ is the connective K-theoretic analogue of Schur polynomials [LS82, Hud14]. Combinatorially, this means if we set β to be 0 in $G_w^{(\beta)}$ when w is Grassmannian, we get a Schur polynomial. Buch [Buc02] established the connective K-theoretic Littlewood-Richardson rule: a tableau formula that computes the coefficient of $G_w^{(\beta)}$ in $G_u^{(\beta)} \times G_v^{(\beta)}$ for Grassmannian permutations u, v, w.

Lascoux polynomials generalize both key polynomials and $G_w^{(\beta)}$ for Grassmannian w. If we set β to be 0 in $\mathfrak{L}_\alpha^{(\beta)}$, we get the key polynomial κ_α [Las03]. If α is weakly increasing, $\mathfrak{L}_\alpha^{(\beta)}$ agrees with $G_w^{(\beta)}(x_1, \cdots, x_n)$ for some Grassmannian w [BSW20]. Therefore, the expansion we are studying is an inhomogeneous analogue of (1) and a non-symmetric extension of (2).

When w is Grassmannian, Monical [Mon16] described a conjectural rule for this expansion involving genomic semistandard skyline fillings. Monical's conjecture is still open. In this paper, we establish a different combinatorial rule for arbitrary $w \in S_+$. Our rule involves *increasing tableaux*, fillings of Young diagrams with positive integers such that each row and column is strictly increasing. Our rule applies a sequence of three operators to increasing tableaux: $K_-(\cdot), \mathrm{cap}_n(\cdot)$ and $\mathrm{wt}(\cdot)$.

For a tableau T, we use T_i to denote the set of numbers that appear in column i of T. We say a tableau T is a *key* if each of its column strictly increasing and $T_1 \supseteq T_2 \supseteq \cdots$. Each increasing tableau P is associated with a key called its *left key*, denoted as $K_-(P)$. In Section 5.1, we give the usual definition of $K_-(P)$ using the K-theoretic jeu-de-taquin of Thomas and Yong [TY09]. In Section 5.2, we derive a method to compute $K_-(P)$ using the $\lhd$ operator.

Take finite sets $S, T \subseteq \mathbb{Z}_{>0}$. Define $T \lhd S$ via the following algorithm. Go through elements in S from the largest to the smallest. For s in S, it picks the largest number in T that is less than s and has not been picked. If such a number exists, we put it in $T \lhd S$. For instance, $\{1, 3, 4, 6, 7, 9\} \lhd \{2, 3, 7, 8\} = \{1, 6, 7\}$. In Corollary 5.10, we show that $K_-(P)_j$ consists of $P_1 \lhd P_2 \lhd \cdots \lhd P_j$ where the evaluation of $\lhd$ is from right to left.

The operator $\mathrm{cap}_n(\cdot)$ is defined on keys with at most n rows. It replaces numbers larger than n in each column by largest integers in $[n]$ that are missing in that column. Then it sorts each column to make it increasing. For instance, a column consisting of $1, 4, 6, 8, 9$ will consist of $1, 3, 4, 5, 6$ after cap_6.

Finally, let $\mathrm{wt}(\cdot)$ be the operator that sends a tableau with entries in $[n]$ to a weak composition. The ith entry is the number of i's in the tableau.

Example 1.1. Suppose $n = 3$. Consider the following three increasing tableaux:

1	4	6	7
3			
7			

1	4	6	7
3	7		
7			

1	4	6	7
3	6		
6	7		

After applying $K_-(\cdot)$, they become the following three keys:

1	3	3	3
3			
7			

1	1	3	3
3	3		
7			

1	1	3	3
3	3		
6	6		

After applying $\mathrm{cap}_3(\cdot)$, they become

1	3	3	3
2			
3			

1	1	3	3
2	3		
3			

1	1	3	3
2	2		
3	3		

Finally, we apply $\mathrm{wt}(\cdot)$ and get $(1, 1, 4), (2, 1, 4)$ and $(2, 2, 4)$.

Each increasing tableau P is associated with a word denoted as $\mathrm{word}(P)$ known as the *reading word*. It is obtained by reading the entries of P from left to right, and bottom to top in each column. The three increasing tableaux in Example 1.1 have reading words 731467, 7317467, and 63176467 respectively. Each word a is associated with a permutation $[a]_H \in S_+$ defined as follows. The *0-Hecke monoid* is the quotient of the free monoid of words on the alphabet $[n-1]$ by the relations

$$ii \equiv_H i$$
$$i(i+1)i \equiv_H (i+1)i(i+1)$$
$$ij \equiv_H ji \qquad \text{for } |i - j| \geq 2.$$

Let $s_i \in S_+$ be the transposition that swaps i and $i + 1$. A *reduced word* of $w \in S_+$ is a word $a_1 a_2 \cdots a_l$ with minimal length such that $w = s_{a_1} \cdots s_{a_l}$. We denote $\ell(w) = l$. We say a is a *Hecke word* of w, denoted as $[a]_H = w$, if $a \equiv_H b$ and b is a reduced word of w.

Example 1.2. We have $421433 \equiv_H 2143$, which is a reduced word of $w = s_2 s_1 s_4 s_3$. Thus $[421433]_H = [2143]_H = w$.

For a weak composition $\alpha = (\alpha_1, \cdots, \alpha_n)$, we define $|\alpha| := \alpha_1 + \cdots + \alpha_n$. For a tableau P, we define $|P|$ as the number of cells in P. Now we can describe our main result.

Theorem 1.3. *Let α be a weak composition. Take an increasing tableau P_1 with $\mathrm{wt}(K_-(P_1)) = \alpha$. Let N be a number such that $N > n$ and $N > \max(P_1)$. Suppose $w \in S_+$ fixes $1, 2, \cdots, N$. Then*

$$(1.1) \qquad \mathfrak{L}_\alpha^{(\beta)} \times G_w^{(\beta)}(x_1, \cdots, x_n) = \sum_P \beta^{|P| - \ell(w) - |\alpha|} \mathfrak{L}_{\mathrm{wt}(\mathrm{cap}_n(K_-(P)))}^{(\beta)},$$

where the sum is over all increasing tableau P which has at most n rows and satisfies the following.

- *If we ignore entries of P that are larger than N, we get P_1.*
- *If we ignore all entries smaller than N in $\mathrm{word}(P)$, we get a Hecke word for w.*

We explain why Theorem 1.3 gives the Lascoux expansion of $\mathfrak{L}_\alpha^{(\beta)} \times G_w(x_1, \cdots, x_n)$ for arbitrary weak composition α and $w \in S_+$. First, we can always find increasing tableau P_1 with $\mathrm{wt}(K_-(P_1)) = \alpha$ for any α: If $\alpha = (\alpha_1, \cdots, \alpha_n)$, we may let P_1 be the increasing tableau whose column c consists of $\{\alpha_i + c - 1 : \alpha_i \geq c\}$. Next,

a fact is $G_w^{(\beta)}(x_1, \cdots, x_n) = G_{1^N \times w}^{(\beta)}(x_1, \cdots, x_n)$, where $1^N \times w$ sends i to $w(i-N) + N$ for all $i > N$ and fixes $1, 2, \cdots, N$. This fact can be deduced from (2.3). Thus, we may replace w by $1^N \times w$, so w satisfies the conditions in Theorem 1.3.

Example 1.4. Say we would like to expand $\mathfrak{L}_\alpha^{(\beta)} \times G_w^{(\beta)}(x_1, \cdots, x_n)$ into Lascoux polynomials where $\alpha = (1, 0, 2)$, $n = 3$ and w has one-line notation 321. We may let P_1 be

$$P_1 = \begin{array}{|c|c|} \hline 1 & 4 \\ \hline 3 \\ \cline{1-1} \end{array}$$

so that $K_-(P_1) = (1, 0, 2)$. Then we may pick $N = 5$, so that $N > \max(P_1)$ and $N > n$. We may redefine w as the permutation with one-line notation 12345876. This replacement does not change $G_w^{(\beta)}$. Then the three increasing tableaux in Example 1.1 satisfy the conditions of Theorem 1.3. They contribute $\mathfrak{L}_{(1,1,4)}^{(\beta)}$, $\beta \mathfrak{L}_{(2,1,4)}^{(\beta)}$ and $\beta^2 \mathfrak{L}_{(2,2,4)}^{(\beta)}$ to the expansion. In Appendix A, we enumerate all increasing tableaux that satisfy the conditions for this example.

Remark 1.5. After setting $\beta = 0$, $\mathfrak{L}_\alpha^{(\beta)}$ becomes κ_α and $G_w^{(\beta)}(x_1, \cdots, x_n)$ becomes $F_w(x_1, \cdots, x_n)$, the Stanley symmetric function [Sta84] in variables $x_1, \cdots, x_n$. By setting $\beta = 0$ in (1.1), we have:

$$(1.2) \qquad \kappa_\alpha \times F_w(x_1, \cdots, x_n) = \sum_P \kappa_{\mathrm{wt}(\mathrm{cap}_n(K_-(P)))},$$

where the sum is over all increasing tableau P such that

- If we ignore entries of P that are larger than N, we get P_1.
- If we ignore all entries smaller than N in word(P), we get a *reduced word* for w.

When w is Grassmannian, $F_w(x_1, \cdots, x_n)$ is a Schur polynomial in $x_1, \cdots, x_n$. Thus, our result restricts to a tableau formula for the key expansion of $\kappa_\alpha \times s_\lambda(x_1, \cdots, x_n)$. We do not have a bijection between our tableaux and skyline fillings in [HLMvW11].

Our approach relies on combinatorial formulas of $G_w^{(\beta)}$ and $\mathfrak{L}_\alpha^{(\beta)}$. Fomin and Kirillov established a combinatorial formula of $G_w^{(\beta)}$ via compatible pairs, certain pairs of words. Buciumas, Scrimshaw, and Weber [BSW20] established a formula for Lascoux polynomials involving reverse set-valued tableaux (RSVT). Our main tool is an insertion algorithm developed by Huang, Shimozono and Yu [HSY22]. This algorithm is a row insertion analogue of Hecke column insertion [BKS+08]. It gives a bijection between compatible pairs and the set of (P, Q) where P is an increasing tableau and Q is a RSVT of the same shape. In Theorem 3.1, we show this row insertion satisfies an analogous property of Hecke column insertion established in [SY23, Theorem 4.2]. This property allows us to turn the tableau formula of $\mathfrak{L}_\alpha^{(\beta)}$ into a compatible pair formula. Then both sides of Theorem 1.3 can be viewed as a sum of certain compatible pairs. Finally, we establish bijections between the compatible pairs representing the two sides.

Remark 1.6. In [SY23], the authors established Theorem 4.2 for Hecke column insertion under the convention that the insertion tableau is a decreasing tableau. The analogous property would not hold for Hecke column insertion of increasing tableaux. However, we do not have a way to state Theorem 1.3 using decreasing tableaux. Moreover, our main argument would not work for decreasing tableaux (see Remark 3.4). Thus, we chose to use the row insertion of Huang, Shimozono and Yu [HSY22] on increasing tableaux.

Lastly, a by-product of Theorem 3.1 is the Lascoux expansion of Grothendieck polynomials. This expansion was first conjectured by Reiner and Yong [RY21] and proved by Shimozono and Yu [SY23]. The proof in [SY23] uses Hecke column insertion on decreasing tableaux while our proof uses the row insertion on increasing tableaux.

The rest of the paper is organized as follows. In §2, we cover some necessary background. In §3, we state Theorem 3.1 and use it to prove Theorem 1.3. The rest of the paper aims to prove Theorem 3.1. In §4, we introduce and investigate the operator $\lhd$. In §5, we define the left key of increasing tableaux and give a simple way to compute it using $\lhd$. In §6, we describe and study the insertion algorithm of Huang, Shimozono and Yu. Then we prove Theorem 3.1.

2. Background

In this section, we first introduce the two main players of this paper: stable Grothendieck polynomials and Lascoux polynomials. Instead of providing their usual definitions in the literature, we describe combinatorial formulas to compute them. Finally, we briefly describe the reverse row insertion and leave the details to §6.

2.1. **Stable Grothendieck polynomials and compatible pairs.** The stable Grothendieck polynomial can be computed using compatible pairs.

Definition 2.1. [BJS93] Let $a = a_1 \cdots a_m$ and $i = i_1 \cdots i_m$ be two words of the same length. We say (a, i) is a *compatible pair* if

- i is weakly increasing, and
- $i_j = i_{j+1}$ implies $a_j > a_{j+1}$.

We say the compatible pair (a, i) is *bounded* if $i_j \leq a_j$ for all $j \in [m]$.

Let $\mathcal{C}$ be the set of all compatible pairs. Let $\mathcal{C}^b$ be the set of bounded compatible pairs. Let $\mathcal{C}_w$ (resp. $\mathcal{C}_w^b$) be the set of $(a, i) \in C$ (resp. C^b) such that a is a Hecke word of w.

Example 2.2. The pair $(421433, 111224)$ is compatible and thus in $\mathcal{C}$. This pair is not bounded since the last number in the first word is smaller than the last number in the second word. On the other hand, $(421433, 111223)$ is bounded.

For a word i, let $\ell(i)$ be its length and let $\mathrm{wt}(i)$ be a sequence where the jth entry is the number of times j appears in i. For a sequence of numbers $(c_1, c_2, \cdots)$ with only finitely many non-zero entries, we use $x^{(c_1, c_2, \cdots)}$ to denote the monomial where the power of x_i is c_i,

Definition 2.3. [FK96] The *Grothendieck polynomial* $\mathfrak{G}_w^{(\beta)}$ and the *stable Grothendieck function* $G_w^{(\beta)}$ can be defined as:

$$(2.1) \qquad \mathfrak{G}_w^{(\beta)} = \sum_{(a,i) \in \mathcal{C}_{w^{-1}}^b} \beta^{\ell(a) - \ell(w)} x^{\mathrm{wt}(i)}$$

$$(2.2) \qquad G_w^{(\beta)} = \sum_{(a,i) \in \mathcal{C}_{w^{-1}}} \beta^{\ell(a) - \ell(w)} x^{\mathrm{wt}(i)}.$$

Example 2.4. Consider w with one-line notation 31524. Then w^{-1} has one-line notation 24153. We know $[421433]_H = w^{-1}$ and $(421433, 111223)$ is bounded, so this pair is in $C_{w^{-1}}^b$. It would contribute $\beta^2 x_1^3 x_2^2 x_3$ to $\mathfrak{G}_w^{(\beta)}$ and $G_w^{(\beta)}$.

Notice that the stable Grothendieck function generally involves an infinite set of variables $x_1, x_2, \cdots$ and has infinitely many terms. One main player of this paper is $G_w^{(\beta)}(x_1, \cdots, x_n)$, which is obtained from $G_w^{(\beta)}$ by setting $x_{n+1} = \cdots = 0$. Clearly,

$$(2.3) \qquad G_w^{(\beta)}(x_1, \cdots, x_n) = \sum_{(a,i) \in \mathcal{C}_{w^{-1}}^{\leq n}} \beta^{\ell(a) - \ell(w)} x^{\mathrm{wt}(i)},$$

where $\mathcal{C}_w^{\leq n} := \{(a, i) \in \mathcal{C}_w : \text{entries of } i \text{ are at most } n\}$.

2.2. **Lascoux polynomials and tableaux.** A *weak composition* is a sequence of n non-negative integers. For a weak composition α, we use α_i to denote its ith entry. We also define x^α as the monomial $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ and define $|\alpha| := \sum_{i=1}^n \alpha_i$.

For a weak composition α, Lascoux [Las03] defined the *Lascoux polynomial* $\mathfrak{L}_\alpha^{(\beta)} \in \mathbb{Z}_{\geq 0}[\beta][x_1, \cdots, x_n]$. We define $\mathfrak{L}_\alpha^{(\beta)}$ using a tableaux formula of Buciumas, Scrimshaw and Weber [BSW20]. It generalizes a classical tableau formula of key polynomials found by Lascoux and Schützenberger [LS88].

A partition is a weakly decreasing sequence of positive numbers. The *Young diagram* of a partition $\lambda = (\lambda_1, \ldots, \lambda_m)$ is the set $\{(r, c) : c \leq \lambda_r\}$. We represent a Young diagram by drawing a cell in row r column c for each (r, c) in the set under the English convention: Row 1 is the topmost row and column 1 is the leftmost column. A *tableau* is a filling of a Young diagram. For a tableau T, we use $T(r, c)$ to denote its filling in the cell (r, c). The *shape* of T, denoted as shape(T), is the underlying Young diagram of T. We say (r, c) is a cell in T if (r, c) is in shape(T).

In this paper, usually, we fill each cell by $\mathbb{Z}_{>0}$ or subsets of $\mathbb{Z}_{>0}$. When T is a tableau filled by $[n]$, we let $\mathrm{wt}(T)$ be the weak composition whose i^{th} entry is the number of (r, c) in T such that $T(r, c) = i$. When T is a tableau filled by subsets of $[n]$, we let $\mathrm{wt}(T)$ be the weak composition whose i^{th} entry is the number of (r, c) in T such that $i \in T(r, c)$.

We define Lascoux polynomials using tableaux. A *reverse semi-standard Young tableau (RSSYT)* is a filling of the Young diagram with $[n]$. We require every row (resp. column) to be weakly (resp. strictly) decreasing from left to right (resp. top to bottom). A *reverse set-valued tableau* (RSVT) is a filling of the Young diagram with non-empty subsets of $[n]$. Moreover, for two horizontally adjacent cells (r, c) and $(r, c+1)$, we require $\min(T(r, c)) \geq \max(T(r, c+1))$. For two vertically adjacent cells (r, c) and $(r+1, c)$, we require $\min(T(r, c)) > \max(T(r+1, c))$. For an RSVT T, let $L(T)$ be the tableau obtained by keeping only the largest number in each entry of T. Clearly, $L(\cdot)$ is a map from RSVTs to RSSYTs.

Example 2.5. For the following RSVT T, we compute $L(T)$.

$$T = \begin{array}{|c|c|c|c|} \hline 6 & 53 & 3 & 321 \\ \hline 4 & 21 & 1 \\ \cline{1-3} 32 \\ \cline{1-1} \end{array} \quad, \qquad L(T) = \begin{array}{|c|c|c|c|} \hline 6 & 5 & 3 & 3 \\ \hline 4 & 2 & 1 \\ \cline{1-3} 3 \\ \cline{1-1} \end{array} \quad.$$

When writing an entry in a RSVT, we simply list its elements in decreasing order. For instance, the "6" in cell $(1,1)$ represents $T(1,1) = \{6\}$ and "321" in cell $(1,4)$ represents $T(1,4) = \{3,2,1\}$.

Lascoux polynomials can be written as sums over RSVTs. To describe which RSVTs can appear in a sum, we need a few more definitions. Recall a *key* is a tableau where each column is increasing and numbers in one column also appear in the column to the left. There is a natural bijection, denoted as $\mathrm{key}(\cdot)$, that sends a weak composition to a key with entries in $[n]$. For a weak composition α, $\mathrm{key}(\alpha)$ is the unique key whose column c consists of $\{i : \alpha_i \geq c\}$. Its inverse is just $\mathrm{wt}(\cdot)$.

Example 2.6. Let $\alpha = (2,1,4,0,2)$. Then

$$\mathrm{key}(\alpha) = \begin{array}{|c|c|c|c|} \hline 1 & 1 & 3 & 3 \\ \hline 2 & 3 \\ \cline{1-2} 3 & 5 \\ \cline{1-2} 5 \\ \cline{1-1} \end{array} \quad.$$

Each RSSYT T is associated with a key called its *left key*, denoted as $K_-(T)$. The left key is originally defined via moves on RSSYT known as jeu-de-taquin. We present one simple method to compute the left key, which is a reformulation of Willis' algorithm [Wil13], using an operation on sets.

Definition 2.7 ([SY23, Definition 3.11]). Take finite sets $S, T \subseteq \mathbb{Z}_{>0}$. Define $T \trianglerighteq S$ be the set computed as follows. Iterate through elements of S from the smallest to the largest. For each $s \in S$, it picks the smallest $t \in T$ such that $t \geq s$ and t has not been picked. Then $T \trianglerighteq S$ is the set of all picked $t \in T$.

For instance, $\{1,4,5,6,7\} \trianglerighteq \{2,3,7\} = \{4,5,7\}$, since 2 picks 4, 3 picks 5 and 7 picks 7.

Definition 2.8. For a RSSYT T, its left key $K_-(T)$ is the key whose column i consists of

$$T_1 \trianglerighteq (T_2 \trianglerighteq (\cdots (T_{i-1} \trianglerighteq T_i) \cdots)).$$

Example 2.9. Let T be the following RSSYT:

$$T = \begin{array}{|c|c|c|c|} \hline 6 & 5 & 3 & 3 \\ \hline 4 & 2 & 1 \\ \cline{1-3} 3 \\ \cline{1-1} \end{array} \quad.$$

Then column 1 of $K_-(T)$ consists of $T_1 = \{3,4,6\}$. Column 2 of $K_-(T)$ consists of $T_1 \trianglerighteq T_2 = \{3,6\}$. Column 3 of $K_-(T)$ consists of $T_1 \trianglerighteq (T_2 \trianglerighteq T_3) = \{3,6\}$. Column 4 of $K_-(T)$ consists of $T_1 \trianglerighteq (T_2 \trianglerighteq (T_3 \trianglerighteq T_4)) = \{6\}$. Thus,

$$K_-(T) = \begin{array}{|c|c|c|c|} \hline 3 & 3 & 3 & 6 \\ \hline 4 & 6 & 6 \\ \cline{1-3} 6 \\ \cline{1-1} \end{array} \quad.$$

Definition 2.10. We define the Lascoux polynomial $\mathfrak{L}_\alpha^{(\beta)}$ using a tableau formula [BSW20]:

$$(2.4) \qquad \mathfrak{L}_\alpha^{(\beta)} = \sum_{\substack{\text{RSVT } T \\ K_-(L(T)) \leq \mathrm{key}(\alpha)}} \beta^{|\mathrm{wt}(T)| - |\alpha|} x^{\mathrm{wt}(T)}.$$

Example 2.11. When $\alpha = (0,2,1)$, the following are all the RSVT that contribute to $\mathfrak{L}_\alpha^{(\beta)}$:

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

$$\begin{array}{|c|c|} \hline 3 & 2 \\ \hline 21 \\ \cline{1-1} \end{array} \, , \quad \begin{array}{|c|c|} \hline 3 & 21 \\ \hline 2 \\ \cline{1-1} \end{array} \, , \quad \begin{array}{|c|c|} \hline 2 & 21 \\ \hline 1 \\ \cline{1-1} \end{array} \, , \quad \begin{array}{|c|c|} \hline 32 & 1 \\ \hline 1 \\ \cline{1-1} \end{array} \, , \quad \begin{array}{|c|c|} \hline 3 & 1 \\ \hline 21 \\ \cline{1-1} \end{array} \, , \quad \begin{array}{|c|c|} \hline 3 & 21 \\ \hline 21 \\ \cline{1-1} \end{array} \, .$$

Thus, $\mathfrak{L}_\alpha^{(\beta)} = x_2^2 x_3 + x_1 x_2^2 + x_1 x_2 x_3 + x_1^2 x_2 + x_1^2 x_3 + \beta(2x_1 x_2^2 x_3 + x_1^2 x_2^2 + 2x_1^2 x_2 x_3) + \beta^2 x_1^2 x_2^2 x_3$.

2.3. **Row insertion algorithm.** The main tool of this paper is an insertion algorithm of Huang, Shimozono and Yu [HSY22]. We adopt a slightly different convention: They described the insertion algorithm on decreasing tableaux but we describe it on increasing tableaux.

Let $\mathcal{T}$ be the set of pairs of tableaux (P, Q), where P is increasing, Q is RSVT and $\text{shape}(P) = \text{shape}(Q)$. For $w \in S_+$, let $\mathcal{T}_w$ be the the set $\{(P, Q) \in \mathcal{T} : [\text{rev}(\text{word}(T))]_H = w\}$ where $\text{rev}(\cdot)$ reverses a word.

In [HSY22], Huang, Shimozono and Yu described a map $\Psi : \mathcal{T} \to \mathcal{C}$ which relies on a row insertion algorithm. We postpone the technical definition of this map to §6. For now, we just need the following result.

Theorem 2.12. *[HSY22, Corollary 5.9] The map Ψ is a bijection from $\mathcal{T}_w$ to $\mathcal{C}_w$ that preserves the weight of the second entry. In other words, Ψ is a bijection from $\mathcal{T}$ to $\mathcal{C}$ such that if $\Psi(P, Q) = (a, i)$, then $[\text{rev}(\text{word}(P))]_H \equiv_H [a]_H$ and $\text{wt}(Q) = \text{wt}(i)$.*

We also need the following statement which we will prove in Section 6.1.

Lemma 2.13. *Take $(A, I) \in \mathcal{C}$ and let N be an integer that does not appear in A. Let a be the word obtained by removing all numbers larger than N in A. Suppose $\Psi^{-1}(A, I) = (P, Q)$ and $\Psi^{-1}(a, i) = (p, q)$ for some i such that $(a, i) \in \mathcal{C}$. Then after removing all numbers larger than N in P, we obtain p.*

3. Proof of Theorem 1.3

Our proof approach requires a technical lemma which describes the preimage of $\mathcal{C}_w^b$ under Ψ. Let $\mathcal{T}^b$ be the set of $(P, Q) \in \mathcal{T}$ such that $K_-(P) \geq K_-(Q)$ where the comparison is entry-wise. Define $\mathcal{T}_w^b$ as the intersection of $\mathcal{T}^b$ and $\mathcal{T}_w$.

Theorem 3.1. *The map Ψ is a bijection from $\mathcal{T}^b$ to $\mathcal{C}^b$, so it is a bijection from $\mathcal{T}_w^b$ to $\mathcal{C}_w^b$ for any $w \in S_+$.*

In this section, we use Theorem 3.1 to prove Theorem 1.3. The proof of Theorem 3.1 will be the goal of all remaining sections.

First, we study the $\text{cap}_n(\cdot)$ operator defined in §1. Let T be a key with at most n rows. Notice that a number i is in $\text{cap}_n(T)_c$ if and only if $i \in T_c$ or $|T_c \cap (i, \infty)| > n - i$. Then we have the following property:

Lemma 3.2. *Let T be a key with at most n rows. Then $\text{cap}_n(T)$ is also a key. Moreover, if T' is another key, $\text{cap}_n(T) \geq T'$ if and only if $T \geq T'$ and $\max(T') \leq n$.*

Proof. Assume $i \in \text{cap}_n(T)_c$ for some $c > 1$. Then $i \in T_c$ or $|T_c \cap (i, \infty)| > n - i$, so $i \in T_{c-1}$ or $|T_{c-1} \cap (i, \infty)| > n - i$. We have $i \in \text{cap}_n(T)_{c-1}$, so $\text{cap}_n(T)$ is a key.

Assume $\text{cap}_n(T) \geq T'$. We know entries of $\text{cap}_n(T)$ are at most n, so $\max(T') \leq n$. Also, $T \geq \text{cap}_n(T) \geq T'$.

Now assume $T \geq T'$ and $\max(T') \leq n$. Consider an arbitrary column T_c with m entries. If $T_c(i) = \text{cap}_n(T)_c(i)$, we clearly have $\text{cap}_n(T)_c(i) \geq T'_c(i)$. If $T_c(i) > \text{cap}_n(T)_c(i)$, we know $\text{cap}_n(T)_c(i), \cdots, \text{cap}_n(T)_c(m)$ are exactly $n - m + i, \cdots, n$. Since $\max(T') \leq n$, we have $T'_c(m) \leq n$ so $T'_c(i) \leq n - m + i = \text{cap}_n(T)_c(i)$. □

Corollary 3.3. *For an key T with at most n rows, we have*

$$\sum_{\text{RSVT } Q : K_-(L(Q)) \leq T, \max(Q) \leq n} \beta^{|\text{wt}(Q)| - |\text{wt}(T)|} x^{\text{wt}(Q)} = \mathfrak{L}_{\text{wt}(\text{cap}_n(T))}^{(\beta)}.$$

Proof. Just need to check the Q we sum over are precisely all Q such that $K_-(L(Q)) \leq \text{cap}_n(T)$. Notice that $\max(Q) \leq n$ is equivalent to $\max(L(Q)) \leq n$, which is equivalent to $\max(K_-(L(Q))) \leq n$. Then the proof is finished by Lemma 3.2. □

Now we prove our main result.

Proof of Theorem 1.3. Observe that the second condition of P is the same as saying $[\text{word}(P)]_H$ is a permutation that agrees with w after N.

Define

$$\mathcal{T}_1 = \{(P, Q) \in \mathcal{T}^b : P = P_1\} \text{ and } C_1 := \Psi(\mathcal{T}_1).$$

By Theorem 3.1, $C_1 \subset \mathcal{C}^b$. Let C_2 denote $\mathcal{C}_w^{\leq n}$. Since we assume w fixes $1, \cdots, N$ and $N > n$, we know $C_2 \subseteq \mathcal{C}^b$. Now by (2.4) and (2.3), we have

$$\mathfrak{L}_\alpha^{(\beta)} = \sum_{(P,Q) \in \mathcal{T}_1} \beta^{|\text{wt}(Q)| - |\alpha|} x^{\text{wt}(Q)} = \sum_{(a,i) \in \mathcal{C}_1} \beta^{|\text{wt}(i)| - |\alpha|} x^{\text{wt}(i)}, \text{ and}$$

$$G_w^{(\beta)}(x_1, \cdots, x_n) = \sum_{(b,j) \in \mathcal{C}_2} \beta^{|\text{wt}(j)| - \ell(w)} x^{\text{wt}(j)}.$$

Let $\mathcal{T}_3$ be the set of $(P, Q) \in \mathcal{T}^b$ such that P satisfies the condition in the Theorem and $\max(Q) \leq n$. Let $\mathcal{C}_3 = \Psi(\mathcal{T}_3)$, so $\mathcal{C}_3 \subset \mathcal{C}^b$. By Corollary 3.3, the right hand side of (1.1) becomes

$$\sum_{(c,k)\in\mathcal{C}_3} \beta^{|\mathrm{wt}(k)|-\ell(w)-|\alpha|} x^{\mathrm{wt}(k)}.$$

It remains to build a bijection between $\mathcal{C}_1 \times \mathcal{C}_2$ and $\mathcal{C}_3$ such that if $(a, i), (b, j) \mapsto (c, k)$, then $\mathrm{wt}(i) + \mathrm{wt}(j) = \mathrm{wt}(k)$. Given $(a, i) \in C_1$ and $(b, j) \in C_2$, numbers in a are smaller than N while numbers in b are larger than N. There is a unique way to "shuffle" them and obtain an element of $\mathcal{C}$. More explicitly, there exists a unique $(c, k) = (c_1 \ldots c_m, k_1 \cdots k_m) \in \mathcal{C}$ such that if we let $S = \{s : c_s < N\}$ and $T = \{t : c_t > N\}$ then

$$(a, i) = (c_{S(1)} \cdots c_{S(|S|)}, k_{S(1)} \cdots k_{S(|S|)}) \quad \text{and} \quad (b, j) = (c_{T(1)} \cdots c_{T(|T|)}, k_{T(1)} \cdots k_{T(|T|)}).$$

Now we check such (c, k) is in $\mathcal{C}_3$. Since $(a, i), (b, j) \in \mathcal{C}^b$, so is (c, k). Let $(P, Q) = \Psi^{-1}(c, j)$. By Theorem 3.1, $\Psi^{-1}(c, j) \in \mathcal{T}^b$. Clearly, $\max(Q) = \max(k) \leq n$, which also implies Q and P have at most n rows. It remains to check P satisfies the two conditions in the Theorem:

- If we only look at numbers in c that are smaller than N, we get a. By Lemma 2.13, if we only look at entries in P that are smaller than N, we get P_1.
- If we only look at numbers in c that are smaller than N, we get b, a Hecke word of w. Thus, $[\mathrm{word}(P_1)]_H = [c]_H$ is a permutation that agrees with w after N.

Now we describe the inverse of the bijection above. Take $(c, k) \in \mathcal{C}_3$. Let $S = \{s : c_s < N\}$ and $T = \{t : c_t > N\}$ and define

$$(a, i) = (c_{S(1)} \cdots c_{S(|S|)}, k_{S(1)} \cdots k_{S(|S|)}) \quad \text{and} \quad (b, j) = (c_{T(1)} \cdots c_{T(|T|)}, k_{T(1)} \cdots k_{T(|T|)}).$$

It remains to check $(a, i) \in \mathcal{C}_1$ and $(b, j) \in \mathcal{C}_2$. By $(c, k) \in \mathcal{C}^b$, we know so are (a, i) and (b, j). By Lemma 2.13, we know $\Psi(a, i) = (P_1, Q)$ for some Q, so $(a, i) \in \mathcal{C}_1$. Since b is obtained from c by looking at numbers larger than N, $[b]_H$ agree with $[c]_H$ after N and fixes $1, 2, \cdots, N$. The condition of $\mathcal{C}_3$ guarantees that $[c]_H$ agrees with w after N. Thus, $[b]_H = w$, so $(b, j) \in \mathcal{C}_2$. □

Remark 3.4. We explain why our argument works on increasing tableaux but not decreasing tableaux. In a decreasing tableau, numbers will be decreasing in each column and row. If we ignore numbers larger than a given number, we might not get a tableau of partition shape.

4. THE ◁ OPERATOR

In this section, we define and study an operator ◁. In the next section, we will use this operator to give a simple algorithm that computes $K_-(P)$ of an increasing tableau P.

In the rest of this paper, we use S, T, U to denote finite subsets of $\mathbb{Z}_{>0}$. Given a set S, let $S(i)$ be the ith smallest element of S for $i \in [|S|]$. Clearly, if $S \subseteq T$, then $S(i) \geq T(i)$ for all $i \in [S]$. Also, given $x \in S$, we use $S - x$ to denote the set $S \setminus \{x\}$. Similarly for $x \notin S$, we use $S \sqcup x$ to denote the set $S \sqcup \{x\}$. We evaluate these expressions from left to right. For instance, $S \sqcup x - y = (S \sqcup x) - y$.

Now we define the ◁ operator in two different, but clearly equivalent, ways.

Definition 4.1. Define $T \triangleleft S$ as follows:

- Recursive definition: If $s \leq t$ for all $s \in S$ and $t \in T$ (including when this is vacuously true, if $S = \emptyset$ or $T = \emptyset$), $T \triangleleft S = \emptyset$. Otherwise, $\max(S) > \min(T)$. Let $m = \max(T_{<\max(S)})$ and define

$$T \triangleleft S := m \sqcup (T_{<m} \triangleleft (S - \max(S))).$$

- Non-recursive definition: We compute $T \triangleleft S$ via the following process. We initialize $T \triangleleft S$ as the empty set and go through elements in S from the largest to the smallest. For s in S, it picks the largest number in T that is less than s and has not been picked. If such a number exists, we put it in $T \triangleleft S$.

We use the recursive definition in proofs by induction and the non-recursive algorithm in other proofs.

Example 4.2. We have

$$\{1, 3, 4, 6, 7, 9\} \triangleleft \{2, 3, 7, 8\} = \{1, 6, 7\}, \qquad \{1, 3, 4, 6, 7, 9\} \triangleleft \{2, 4, 7, 8\} = \{1, 3, 6, 7\}.$$

Observe as well that the ◁ operation is not associative, as

$$(\{1, 2\} \triangleleft \{2, 3\}) \triangleleft \{3\} = \{1, 2\} \triangleleft \{3\} = \{2\} \quad \text{but} \quad \{1, 2\} \triangleleft (\{2, 3\} \triangleleft \{3\}) = \{1, 2\} \triangleleft \{2\} = \{1\}.$$

Because of this, whenever there is ambiguity of the order of evaluation we assume that the expression is evaluated from right-to-left, so $\{1, 2\} \triangleleft \{2, 3\} \triangleleft \{3\} = \{1\}$.

We now study the $\lhd$ operator from various aspects and develop some technical lemmas that will be used in future sections.

4.1. **Understanding** $(T - x) \lhd S$. For $x \in T$, we would like to understand how $(T - x) \lhd S$ differs from $T \lhd S$. We start with an easy special case:

Lemma 4.3. *If* $x \in T \setminus (T \lhd S)$, *then* $(T - x) \lhd S = T \lhd S$.

Proof. Prove by induction on $|S|$. If $|S| = 0$, we are done. Now assume that the statement is proved whenever $|S| = k - 1$, and let $|S| = k$. If $\max(S) \leq \min(T)$, then $\max(S) \leq \min(T - x)$ as well, and so $(T - x) \lhd S = T \lhd S = \emptyset$ and we are done. Otherwise, let $m = \max(T_{<\max(S)})$. By definition, $m \in T \lhd S$, so $m \neq x$ and $m = \max((T - x)_{<\max(S)})$ as well. Notice that $(T - x)_{<m}$ is either $T_{<m}$ or $T_{<m} - x$. In either case, we have $(T - x)_{<m} \lhd (S - \max(S)) = T_{<m} \lhd (S - \max(S))$ by the inductive hypothesis. Thus,

$$(T - x) \lhd S = m \sqcup ((T - x)_{<m} \lhd (S - \max(S))) = m \sqcup (T_{<m} \lhd (S - \max(S))) = (T - x) \lhd S. \quad \square$$

This Lemma allows us to remove certain elements from T without changing T.

Corollary 4.4. *Let* $T' \subseteq T$ *such that* $T \lhd S \subseteq T'$. *Then* $T \lhd S = T' \lhd S$.

Proof. This follows from repeated application of Lemma 4.3. $\square$

Now we analyze the harder case: $x \in T \lhd S$.

Lemma 4.5. *Take* $x \in T \lhd S$. *If* $T_{<x} \setminus (T \lhd S)$ *is empty, then* $(T - x) \lhd S = (T \lhd S) - x$. *Otherwise,* $(T - x) \lhd S = (T \lhd S) - x \sqcup x'$, *where* $x' = \max(T_{<x} \setminus (T \lhd S))$.

Proof. Let i be such that $T(i) = x$. Find smallest i' such that for any $i' \leq j \leq i$, $T(j) \in T \lhd S$. Notice that $i' > 1$ if and only if $T_{<x} \setminus (T \lhd S)$ is not empty. In this case, $T(i' - 1) = x'$.

Consider the non-recursive way to compute $T \lhd S$ and $(T - x) \lhd S$. If $x = T(i)$ is picked by some $s \in S$ when computing $T \lhd S$, then s would pick $T(i-1)$ instead when computing $(T-x) \lhd S$. Inductively, for $j = i, i-1, \cdots, i'+1$, if $s \in S$ picks $T(j)$ when computing $T \lhd S$, s would pick $T(j - 1)$ instead when computing $(T - x) \lhd S$. If $i' = 1$, we have $(T - x) \lhd S = T \lhd S - x$ and we are done. Otherwise, when $s \in S$ picks $T(i')$ when computing $T \lhd S$, s would pick $T(i' - 1) = x'$ instead when computing $(T - x) \lhd S$. After that, the computations of $T \lhd S$ and $(T - x) \lhd S$ behave the same. $\square$

4.2. **Domination.** Clearly, $|T \lhd S| \leq |S|$. In Example 4.2, we see sometimes $|T \lhd S| < |S|$. We would like to understand when $T \lhd S$ has the same size as S. This motivation brings us to the following notion:

Definition 4.6. If $|T| \geq |S|$ and $T(i) < S(i)$ for all $i \in [|S|]$, we say that S *dominates* T and write $T \preceq S$.

An alternate characterization for dominance is that $S_1 \preceq S_2 \preceq \cdots \preceq S_+$ if and only if there exists an n-column increasing tableau of normal shape with column i consisting of the numbers in S_i. From this perspective, it is clear that if $T \preceq S$ and $S' \subseteq S$, then $T \preceq S'$. Notice that $\preceq$ is transitive, but it is not irreflexive since $\emptyset \preceq \emptyset$.

This notion characterizes when $T \lhd S$ has the same size as S:

Lemma 4.7. *We have* $|T \lhd S| = |S|$ *if and only if* $T \preceq S$.

Proof. Prove by induction on $|S|$. If $|S| = 0$, then $S = \emptyset$ and $T \lhd \emptyset = \emptyset$. Correspondingly, we have $T \preceq \emptyset$.

Now assume that the statement is true when $|S| = k-1$, and say that $|S| = k$ for some $k \geq 1$. If $\max(S) \leq \min(T)$, then $T \lhd S = \emptyset$ and we cannot have $T \preceq S$, completing the proof in this case. Otherwise, $\max(S) > \min(T)$. Let $m = \max(T_{<\max(S)})$, so $T \lhd S = m \sqcup (T_{<\max(S)} \lhd (S - \max(S)))$. By the inductive hypothesis, we deduce:

$$\begin{aligned}
|T \lhd S| = k &\Leftrightarrow |T_{<\max(S)} \lhd (S - \max(S))| = k - 1 \\
&\Leftrightarrow T_{<\max(S)} \preceq (S - \max(S)) \\
&\Leftrightarrow |T_{<\max(S)}| \geq k - 1, \text{ and for each } i \in [k-1], T_{<\max(S)}(i) \leq S(i) \\
&\Leftrightarrow |T| \geq k, \text{ and for each } i \in [k], T(i) \leq S(i) \\
&\Leftrightarrow T \preceq S. \quad \square
\end{aligned}$$

We are mainly interested in $T \lhd S$ when $T \preceq S$. In this case, $|T \lhd S| = |S|$. If we consider the non-recursive way to compute $T \lhd S$, when $S(i)$ picks a number in T, we know it is $(T \lhd S)(i)$. Thus, we can use the non-recursive definition to study $T \lhd S$. We first show that $T \lhd S$ has an entry-wise upper bound.

Corollary 4.8. *If* $T \preceq S$, *then* $T \lhd S \preceq S$.

Proof. Since $T \preceq S$, $|S| = |T \triangleleft S|$, so $|S| \leq |T \triangleleft S|$. Using the non-recursive definition of $\triangleleft$, the number $S(i)$ picks someone that is smaller than $S(i)$, so $(T \triangleleft S)(i) < S(i)$, completing the proof. $\square$

Using Corollary 4.8, we deduce a simple result that allows us to understand the small numbers in $T \triangleleft S$ in certain cases.

Lemma 4.9. *Assume $T \preceq S$. Take x such that $|T_{<x}| = |S_{\leq x}|$. Then for any $j \leq |T_{<x}|$, we have $(T \triangleleft S)(j) = T(j)$.*

Proof. Let $i = |T_{<x}|$. If $i = 0$, our statement is trivial. Now assume $i > 0$. By $T \preceq S$ and Corollary 4.8, $(T \triangleleft S)(i) < S(i) \leq x$, so there are at least i numbers in $T \triangleleft S$ that are less than x. There are exactly i such numbers in T, so they are all in $T \triangleleft S$. In other words, for any $j \in [i]$, $(T \triangleleft S)(j) = T(j)$. $\square$

Corollary 4.10. *Assume $T \preceq S$ and there exists $x \notin S$ such that $|S_{<x}| = |T_{<x}|$. Take $y \in S$ such that $x > y$ and let $S' = (x \sqcup S) - y$. Then $T \preceq S'$ and for all i such that $S(i) < x$, $(T \triangleleft S)(i) = (T \triangleleft S')(i) = T(i)$.*

Proof. For each $i \in [|S|]$, we have $T(i) < S(i) \leq S'(i)$. Thus, $T \preceq S'$. Notice that $|S_{\leq x}| = |S'_{\leq x}| = |T_{<x}|$. Then the lemma is finished by Lemma 4.9. $\square$

Using results from the previous section, we can get another lower bound of $T \triangleleft S$ involving $(T - x) \triangleleft S$.

Lemma 4.11. *Assume $T \preceq S$. Take any $x \in T$ such that $T - x \preceq S$. Then $(T \triangleleft S)(i) \geq ((T - x) \triangleleft S)(i)$ for $i \in [|S|]$.*

Proof. If $x \notin T \triangleleft S$, then by Lemma 4.3 $T \triangleleft S = (T - x) \triangleleft S$, and the statement is true. Thus, we assume $x \in T \triangleleft S$.
    Let $x' \in T$ be the largest such that $x' < x$ and $x' \notin T \triangleleft S$. It x' does not exist, by Lemma 4.5,

$$|(T - x) \triangleleft S| = |(T \triangleleft S) - x| < |S|.$$

By Lemma 4.7 and $(T - x) \preceq S$, we cannot have $|(T - x) \preceq S| < |S|$. Thus, x' must exist. By Lemma 4.5 $(T - x) \triangleleft S = (T \triangleleft S) - x \sqcup x'$. Our statement is proved since $x' < x$. $\square$

We end this section with a proposition that allows us to "split" the computation of $T \triangleleft S$.

Proposition 4.12. *Find x such that $T_{\geq x} \preceq S_{>x}$. Then*

$$T \triangleleft S = (T_{<x} \triangleleft S_{\leq x}) \sqcup (T_{\geq x} \triangleleft S_{>x})$$

Proof. Consider the non-recursive way of computing $T \triangleleft S$ and $(T_{\geq x} \triangleleft S_{>x})$. Since $T_{\geq x} \preceq S_{>x}$, $s \in S_{>x}$ will pick the same $t \in T_{\geq x}$ in both computations.
    Now consider the computation of $T \triangleleft S$. After every $s \in S_{>x}$ picks, the non-picked numbers in T are $T_{<x}$ together with some numbers at least x. When the next $s \in S$ picks, we know $s \leq x$, so it will only pick numbers from $T_{<x}$. Starting from here, the computation behaves the same as if computing $T_{<x} \triangleleft S_{\leq x}$. $\square$

Corollary 4.13. *Find $x \in T$ such that $T_{\geq x+1} \preceq S_{>x+1}$ and $x + 1 \notin S$. Then $x \notin T \triangleleft S$.*

Proof. By Proposition 4.12, $T \triangleleft S = (T_{<x+1} \triangleleft S_{\leq x+1}) \sqcup (T_{\geq x+1} \triangleleft S_{>x+1})$. Clearly x is not in the second term. By $x + 1 \notin S$, x is not in the first term either. $\square$

4.3. Comparing $T \triangleleft S'$ and $T \triangleleft S$ when $S' \subseteq S$. We start with a simple property.

Lemma 4.14. *For $S' \subseteq S$, we have $T \triangleleft S' \subseteq T \triangleleft S$.*

Proof. Prove by induction on $|S|$. If $|S| = 0$, then $S = S' = \emptyset$, so $T \triangleleft S' = \emptyset = T \triangleleft S$.
    Now assume that the statement is true whenever $|S| = k - 1$, and consider an arbitrary S such that $|S| = k$. If $\max(S) \leq \min(T)$, then $T \triangleleft S' = T \triangleleft S = \emptyset$ and we are done. Otherwise, let $m = \max(T_{<\max(S)})$. If $m \in S'$, by the inductive hypothesis,

$$T \triangleleft S' = m \sqcup (T_{<m} \triangleleft (S' - m)) \subseteq m \sqcup (T_{<m} \triangleleft (S - m)) = T \triangleleft S.$$

Otherwise, $m \notin S'$. We have $S' \subseteq (S - m)$. Notice that $T \triangleleft S \subseteq T_{<m}$. By Corollary 4.4 and the inductive hypothesis,

$$T \triangleleft S' = T_{<m} \triangleleft S' \subseteq (T_{<m} \triangleleft (S - m)) = T \triangleleft S.$$

$\square$

By Lemma 4.14, when $T \preceq S$, $T \triangleleft (S - a)$ will be $T \triangleleft S$ with one element removed. For certain a, we can figure out which element of $T \triangleleft S$ is removed.

Lemma 4.15. *Assume $T \preceq S$ and take $S' \subset S$. If $a = \min(S \setminus S')$ and $b = \min((T \triangleleft S) \setminus (T \triangleleft S'))$, then $T \triangleleft (S - a) = (T \triangleleft S) - b$*

Proof. Since $S-a$ and S' are subsets of S, by Lemma 4.14, $T \lhd (S-a)$ and $T \lhd S'$ are subsets of $T \lhd S$. By Corollary 4.4, we may replace T by $T \lhd S$. This change does not affect $T \lhd S$, $T \lhd (S-a)$ and $T \lhd S'$.

By Lemma 4.7, $|T \lhd S| = |S| = 1 + |S - a| = 1 + |T \lhd (S-a)|$. By Lemma 4.14 we have that $T \lhd (S-a) \subseteq T \lhd S$. Thus, $(T \lhd S) \setminus (T \lhd (S-a))$ contains only one element, which we denote as x. It suffices to show that $x = b$. Let $x = T(i)$. It remains to check: $x \notin T \lhd S'$ and $T(j) \in T \lhd S'$ for any $j \in [i-1]$.

- Since $x \notin T \lhd (S-a)$ and $S' \subseteq S - a$, by Lemma 4.14, $x \notin T \lhd S'$.
- On one hand, we have $S(i) > T(i) = x$. On the other hand, consider the non-recursive way to compute $T \lhd (S-a)$. When $(S-a)(i-1)$ picks $T(i-1)$, x has not been picked. Thus, $(S-a)(i-1) \le x$. In particular, $S(i) > (S-a)(i-1)$, so $S(i) \le a$ and $(S-a)(i-1) = S(i-1)$.

  Now we have $|S_{\le x}| = i - 1$. By the definition of a, $S(i) \le a$ implies that $S(1), \cdots, S(i-1) \in S'$, so $|S'_{\le x}| = i - 1 = |T_{<x}|$. By Lemma 4.9, $(T \lhd S')(j) = T(j)$ for $j \in [i-1]$. $\square$

4.4. **Understanding** $U \lhd T \lhd S$.

The main goal of this section is to prove the following technical lemma:

Lemma 4.16. *Suppose $U \preceq T \preceq S$. Assume that there exists $x \notin T$ such that $|U_{<x}| = |T_{<x}|$. Assume there exists $y \in T_{<x}$ such that $T \sqcup x - y \preceq S$. Take the smallest such y, then*

$$U \lhd T \lhd S' = U \lhd (T \sqcup x - y) \lhd S'$$

where S' is any subset of S. Consequently,

$$U \lhd T = U \lhd (T \sqcup x - y).$$

To prove this Lemma, we consider three sets:

$$(4.1) \qquad U \lhd T \lhd S', \quad U \lhd (T \sqcup x) \lhd S', \quad U \lhd (T \sqcup x - y) \lhd S'.$$

The goal is to show the first set equals the last set in (4.1. We first derive a lemma that implies the last two agree.

Lemma 4.17. *Assume $T \preceq S$ and $|T| > |S|$. Let $y \in T$ be the smallest such that $(T - y) \preceq S$. Then*

- *For all $T(j) < y$, $(T \lhd S)(j) = T(j)$.*
- *We have $y \notin T \lhd S$.*
- *For any $S' \subseteq S$, $(T - y) \lhd S' = T \lhd S'$.*

Proof. Let i be such that $y = T(i)$.

- If $i = 1$, then $y = \min(T)$, and the first statement is vacuously true. Now assume that $i > 1$. By the definition of y, $T - T(i) \preceq S$, but $T - T(i-1) \npreceq S$. Thus, $T(i) \ge S(i-1)$. In other words, $|S_{\le y}| = i - 1 = |T_{<y}|$. Then the first statement follows from Lemma 4.9.
- By the first statement, for all $j < i$, $(T \lhd S)(j) = T(j) < T(i) = y$. For all $j \ge i$, Lemma 4.11 says that $(T \lhd S)(j) \ge ((T-y) \lhd S)(j) \ge (T-y)(j) = T(j+1) > T(i) = y$. Therefore, for all j, $(T \lhd S)(j) \ne y$, so $y \notin T \lhd S$.
- By Lemma 4.14, $T \lhd S' \subseteq T \lhd S$. By the second statement, $y \notin T \lhd S$, so $y \notin T \lhd S'$. By Lemma 4.3, $(T-y) \lhd S' = T \lhd S'$. $\square$

To show the first two sets in (4.1) agree, we need the following two lemmas:

Lemma 4.18. *Suppose $|U| \le |T|$. Let $\delta = |U| - |T|$. Suppose for each $i \in [|U|]$, $U(i) \le T(i + \delta)$. Take x such that $x > \max(T)$.*

- *We have $U \lhd (T \sqcup x) = U \lhd T = T$.*
- *For any $T' \subsetneq T$, let $x' = \max(T \setminus T')$. Then $U \lhd (T' \sqcup x) = U \lhd (T' \sqcup x')$.*

Proof. The first statement is immediate. For the second statement, assume $x' = T(i)$. We know $T(|T|), \cdots, T(i+1) \in T'$. Consider the non-recursive way to compute $U \lhd (T' \sqcup x)$ and $U \lhd (T' \sqcup x')$. In the first process, clearly, $x, T(|T|), \cdots, T(i+1)$ would pick $U(|U|), U(|U|-1) \cdots, U(i-\delta)$. In the second process, $T(|T|), \cdots, T(i)$ would also pick $U(|U|), \cdots, U(i-\delta)$. After that, the two processes have the same behavior. $\square$

Lemma 4.19. *Assume we can find $x \notin T$ such that $U_{\ge x} \preceq T_{>x}$ and $|U_{<x}| \le |T_{<x}|$. Let $\delta = |T_{<x}| - |U_{<x}|$. Assume for any $i \in [|U_{<x}|]$, $U(i) \le T(i+\delta)$. Then $U \lhd T \lhd S = U \lhd (T \sqcup x) \lhd S$ for any set S.*

Proof. Suppose $x \notin (T \sqcup x) \lhd S$. By Lemma 4.3, $(T \sqcup x) \lhd S = T \lhd S$ so our claim is trivial.

Otherwise, let $T' = (T \sqcup x) \lhd S$. Since $U_{\geq x} \preceq T_{>x}$, we have $U_{\geq x} \preceq T'_{>x}$. By Proposition 4.12,

$$(4.2) \qquad U \lhd T' = U_{<x} \lhd T'_{\leq x} \sqcup U_{\geq x} \preceq T'_{>x}.$$

By Lemma 4.5, $(T \lhd S)_{>x} = T'_{>x}$. Thus, (4.2) still holds if we replace T' by $T \lhd S$. To show $U \lhd T' = U \lhd (T \lhd S)$, it remains to check:

$$(4.3) \qquad U_{<x} \lhd T'_{\leq x} = U_{<x} \lhd (T \lhd S)_{\leq x}.$$

Let $x' = \max(T_{<x} \setminus T'_{<x})$. If x' does not exist, by Lemma 4.5, $T'_{<x} = (T \lhd S)_{<x} \sqcup x$. Then (4.3) follows from the first statement of Lemma 4.18. Otherwise, by Lemma 4.5, $T'_{<x} = (T \lhd S)_{<x} \sqcup x - x'$. Then (4.3) follows from the second statement of Lemma 4.18. □

Finally, we can prove Lemma 4.16.

Proof of Lemma 4.16. By the third statement of Lemma 4.17, the last two sets in (4.1) agree. We just need to check the first two sets in (4.1) agree. Since $U \preceq T$ and $|U_{<x}| = |T_{<x}|$, we have $U_{\geq x} \preceq T_{\geq x}$. Then U and T satisfies the condition in Lemma 4.19, so we have the first equation in Lemma 4.16. For the second equation, we simply let S be a set consisting of $|T|$ very large numbers, so $T \lhd S = T$ and $(T \sqcup x - y) \lhd S = (T \sqcup x - y)$. □

5. THE LEFT KEY OF AN INCREASING TABLEAU

For an increasing tableau P, its left key is denoted as $K_-(P)$. In Section 5.1, we give the usual definition of $K_-(P)$ as in [RY21] using K-theoretic jeu-de-taquin of Thomas and Yong [TY09]. In Section5.2, we derive a simple way to compute $K_-(P)$ using the $\lhd$ operator.

5.1. **Defining $K_-(P)$ using K-theoretic jeu-de-taquin.** In this section, we consider a more general family of tableaux. Let λ and μ be two partitions such that the Young diagram of μ is contained in the Young diagram of λ. The *skew Young diagram* of λ/μ is the set theoretic difference between the Young diagram of λ and the Young diagram of μ. A *skew tableau* of shape λ/μ is a filling of the skew Young diagram of λ/μ with positive integers. Skew tableaux whose shapes are left-justified and top-justified are called *normal*. Skew tableaux whose cells are right-justified and bottom-justified are called *anti-normal*. We say a skew tableau is increasing if each of its row (resp. column) increases from left to right (resp. top to bottom). Let sInc be the set of all increasing tableaux with skew shape.
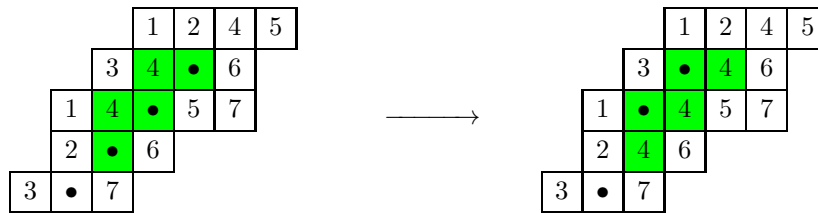
Thomas and Yong [TY09] introduce an operation called *anti-rectification* which shifts cells in $T \in$ sInc to make it anti-normal. This map consists of small moves called *reverse K-jeu-de-taquin* (revKjdt) which are defined on a more general family of fillings.

Definition 5.1. A *dotted skew tableau* is a filling of a skew Young diagram with positive integers and the symbol $\bullet$. Let m be a positive integer. Let $\mathrm{sInc}_{m<\bullet}$ be the set of dotted skew tableaux such that each row and column is strictly increasing with respect to the order $\cdots < m < \bullet < m+1 < \cdots$.

The shape of a dotted skew tableau is defined similarly. Therefore, we can also use normal and anti-normal to describe dotted skew tableaux.

Definition 5.2. [TY09] Define the *reverse K-jeu-de-taquin* (revKjdt) as a shape-preserving map from $\mathrm{sInc}_{m<\bullet}$ to $\mathrm{sInc}_{m-1<\bullet}$ when $m \geq 1$. Take $T \in \mathrm{sInc}_{m<\bullet}$. For each $\bullet$ (resp. m) that is adjacent to an m (resp. $\bullet$) in T, we replace it by m (resp. $\bullet$). The resulting dotted skew tableau is in $\mathrm{sInc}_{m-1<\bullet}$.

Example 5.3. We perform revKjdt to send an element of $\mathrm{sInc}_{4<\bullet}$ to $\mathrm{sInc}_{3<\bullet}$

We provide another way to understand revKjdt. Let us start with following definitions.

Definition 5.4. A skew shape is called a **short ribbon** if it satisfies the following.
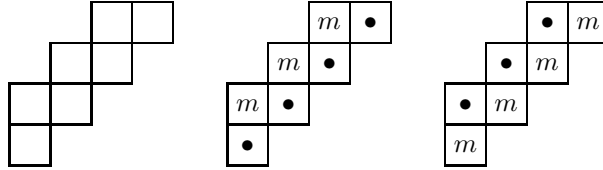
- All cells are connected.
- Does not have a 2×2 subshape.
- Each column or row has at most 2 cells.

A dotted skew tableau is called an **alternating ribbon** of m if it satisfies the following.

- Its shape is a short ribbon.
- All cells are filled by $\bullet$ and m. Adjacent cells are filled differently.

It is clear that for any short ribbon, there are two alternating ribbons of m with that shape. We say an alternating ribbon has type 1 if its topmost cell in the leftmost column is m. Otherwise, it has type 2.
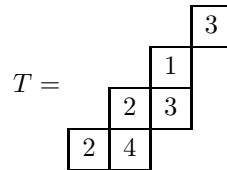
Example 5.5. In this example, we present a short ribbon and the two alternating ribbons of m with that shape. The former has type 1 and the latter has type 2.
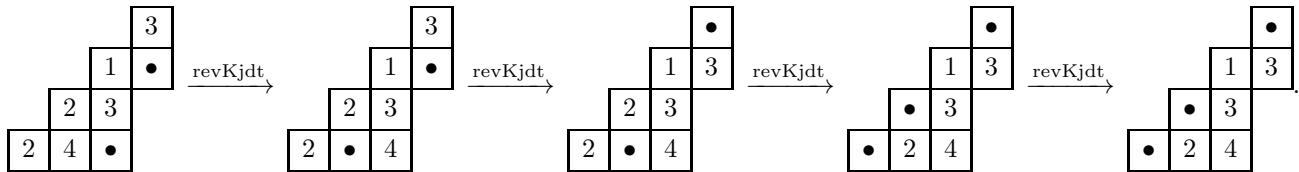
Now take $T \in \mathrm{sInc}_{m<\bullet}$. We may describe revKjdt as follows. If we focus on its m and $\bullet$, ignoring all other numbers, we see a skew-shape tableau where each connected component is an alternating ribbon. Each alternating ribbon with more than one cell must have type 1. A revKjdt move would change these alternating ribbons with more than one cell into type 2.

Next, we use revKjdt to describe a process called *anti-rectification*. Let T be an increasing tableau of skew shape λ/μ. Say all numbers in T are at most m for some m. We may add some cells containing $\bullet$ to T and obtain an element of $\mathrm{sInc}_{m<\bullet}$. By recursively applying revKjdt, we obtain an element of $\mathrm{sInc}_{0<\bullet}$. Then we remove all $\bullet$. This process yields a skew increasing tableau with a different shape. By repeating this process, we will end with an increasing tableau of anti-normal shape.
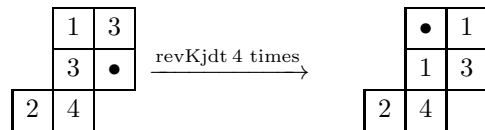
Example 5.6. Let T be the following element of sInc.

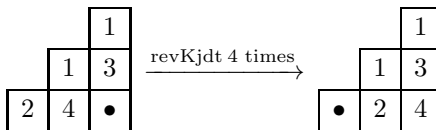We may put a $\bullet$ under each "3" and perform revKjdt:

Now we may ignore the $\bullet$, obtaining an element of sInc. Then we may place a $\bullet$ at row 3 and repeat this process:
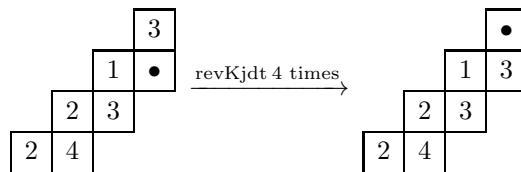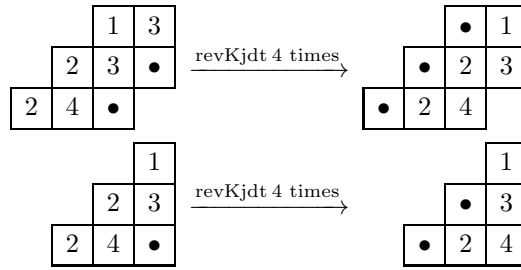
Finally, we remove the current $\bullet$ and put a $\bullet$ under 3. We perfrom revKjdt again, obtaining

which is the result of our anti-rectification if we ignore its $\bullet$

Remark 5.7. During the anti-rectification process, we need to choose where to place the $\bullet$. Making different choices will actually affect the final result. Let us anti-rectify the T in Example 5.6 with different choices of the positions to place the $\bullet$.

$$
\begin{array}{cc}
\begin{array}{|c|c|c|}
\hline
 & 1 & 3 \\
\hline
\end{array} & \\
\end{array}
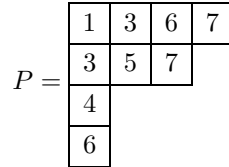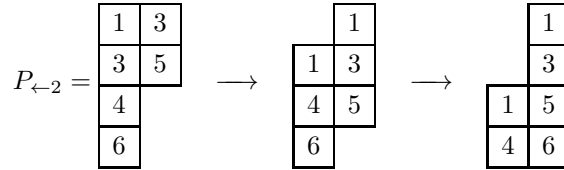\quad\xrightarrow{\text{revKjdt 4 times}}\quad
$$

If we ignore the bullet in the last dotted skew tableau, we obtain the result of our anti-rectification. This time, we end up with something different from the output in Example 5.6.

Finally, for an increasing tableau P of normal shape. We use anti-rectification to define $K_-(P)$, the left key of P. Let $P_{\leftarrow j}$ to be the tableau obtained by keeping only the first j columns of P. Column j of $K_-(P)$ only depends on $P_{\leftarrow j}$. Say P has r rows. We may embed $P_{\leftarrow j}$ in an j by r rectangle. Then we anti-rectify $P_{\leftarrow j}$. When we choose where to place the $\bullet$, we always place one $\bullet$ at the leftmost available position inside the rectangle. After the anti-rectification, numbers in the leftmost column will form column j of $K_-(P)$.
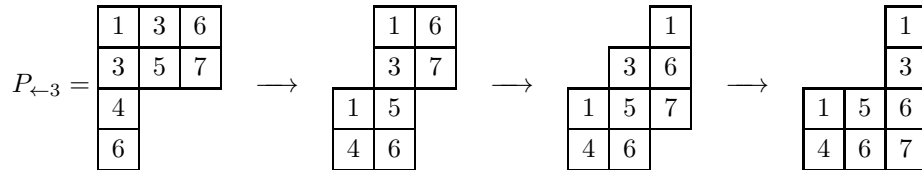
Example 5.8. Let P be the following increasing tableau of normal shape.

$$
P =
\begin{array}{|c|c|c|c|}
\hline
1 & 3 & 6 & 7 \\
\hline
3 & 5 & 7 \\
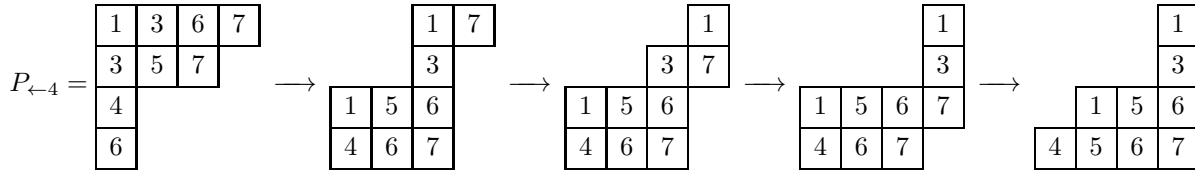\cline{1-3}
4 \\
\cline{1-1}
6 \\
\cline{1-1}
\end{array}
$$

We compute $K_-(P)$ as follows. First, consider $P_{\leftarrow 1}$, which is already anti-normal. Column 1 of $K_-(P)$ consists of $\{1, 3, 4, 6\}$. Next, consider $P_{\leftarrow 2}$. We may anti-rectify it as follows.

$$
P_{\leftarrow 2} =
\begin{array}{|c|c|}
\hline
1 & 3 \\
\hline
3 & 5 \\
\hline
4 \\
\cline{1-1}
6 \\
\cline{1-1}
\end{array}
\longrightarrow
\cdots
\longrightarrow
\cdots
.
$$

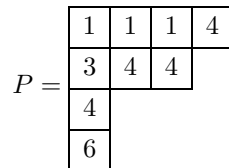Column 2 of $K_-(P)$ consists of $\{1, 4\}$. Next, we anti-rectify $P_{\leftarrow 3}$. Notice that the first two iterations of anti-rectifying $P_{\leftarrow 3}$ will be the same as anti-rectifying $P_{\leftarrow 2}$.

$$
P_{\leftarrow 3} =
\begin{array}{|c|c|c|}
\hline
1 & 3 & 6 \\
\hline
3 & 5 & 7 \\
\hline
4 \\
\cline{1-1}
6 \\
\cline{1-1}
\end{array}
\longrightarrow \cdots \longrightarrow \cdots \longrightarrow \cdots
.
$$

Column 3 of $K_-(P)$ also consists of $\{1, 4\}$. Finally, we anti-rectify $P_{\leftarrow 4}$.

$$
P_{\leftarrow 4} =
\begin{array}{|c|c|c|c|}
\hline
1 & 3 & 6 & 7 \\
\hline
3 & 5 & 7 \\
\cline{1-3}
4 \\
\cline{1-1}
6 \\
\cline{1-1}
\end{array}
\longrightarrow \cdots \longrightarrow \cdots \longrightarrow \cdots \longrightarrow \cdots
.
$$

Column 4 of $K_-(P)$ consists of $\{4\}$. Thus, $K_-(P)$ is

$$
P =
\begin{array}{|c|c|c|c|}
\hline
1 & 1 & 1 & 4 \\
\hline
3 & 4 & 4 \\
\cline{1-3}
4 \\
\cline{1-1}
6 \\
\cline{1-1}
\end{array}
$$

From this definition, it is not clear that $K_-(P)$ must be a key. A proof of this can be found in [RY21]. This argument is originally formed by the first author of this paper.

It is hard to compute $K_-(P)$ via this definition since the computation consists of many revKjdt moves. Thus, in the next section, we provide a simple way to compute $K_-(P)$ using $\lhd$.

5.2. **Computing $K_-(P)$ using $\lhd$.** We now present a simple method that computes $K_-(P)$ for $P \in \mathrm{Inc}_n$. Recall that P_i is the set of numbers that appear in column i of P. Define

$$P_{i,j} := P_i \lhd P_{i+1} \lhd \cdots \lhd P_j.$$

Then we can state the main result of this section.

Theorem 5.9. *Let $P \in \mathrm{Inc}_n$ with C columns. If we anti-rectify P and get $P^{\searrow}$ whose rightmost column is in column C, then $P_1^{\searrow} = P_{1,C}$.*

A consequence of this result is the following simple algorithm that computes $K_-(P)$.

Corollary 5.10. *Let P be an increasing tableau with normal shape. Column i of $K_-(P)$ consists of $P_{1,i}$.*

Proof of Theorem 5.9. Since $P^{\searrow}$ is anti-normal, $P_{1,C}^{\searrow} = P_1^{\searrow}$. We need to show $P_{1,C} = P_{1,C}^{\searrow}$. We obtain $P^{\searrow}$ from P by applying the three operators iteratively: Adding $\bullet$ to get a tableau in $\mathrm{sInc}_{n<\bullet}$; Performing revKjdt to get a tableau in $\mathrm{sInc}_{0<\bullet}$; Removing all $\bullet$. It remains to check the Lemma 5.11 below. $\square$

For $P \in \mathrm{sInc}_{n<\bullet}$, we let P_i be the set of numbers appearing in column i of P, ignoring the $\bullet$. Then $P_{1,C}$ is also well-defined.

Lemma 5.11. *Take $P \in \mathrm{sInc}_{n<\bullet}$. Let C be the largest such that $P_C \neq \emptyset$. Perform a revKjdt move on P and obtain $\tilde{P} \in \mathrm{Inc}_{n-1<\bullet}$. Then $P_{1,C} = \tilde{P}_{1,C}$.*

This Lemma can be proved by studying the effect of changing one alternating ribbon.

Lemma 5.12. *Take $P \in \mathrm{sInc}_{n<\bullet}$ with C columns. Assume there is one alternating ribbon of n and $\bullet$ within P. Moreover, assume the alternating ribbon has more than one cell and appears in every column of P.*
Now switch this alternating ribbon into type 2 of the same shape. Let $\tilde{P} \in \mathrm{sInc}_{n-1<\bullet}$ be the result. Then

$$P_1 \lhd \cdots \lhd P_C \lhd S = \tilde{P}_1 \lhd \cdots \lhd \tilde{P}_C \lhd S$$

for any set S.

Proof. When $C = 1$, the claim is immediate. Assume $C > 1$. We start with following observations.
- If $n \notin P_C$, then $\tilde{P}_C = P_C \sqcup n$. Otherwise, $\tilde{P}_C = P_C$.
- Similarly, if column 1 of P does not have $\bullet$, then $\tilde{P}_1 = P_1 - n$. Otherwise, $\tilde{P}_1 = P_1$.
- For $1 < c < C$, $\tilde{P}_c = P_c$.

First, we claim

$$P_{C-1} \lhd P_C \lhd S = P_{C-1} \lhd \tilde{P}_C \lhd S.$$

The claim is immediate if $P_C = \tilde{P}_C$. Assume $\tilde{P}_C = P_C \sqcup n$. Notice that setting $U = P_{C-1}$, $T = P_C$ and $x = n$ would satisfy the conditions of Lemma 4.19. Our claim is implied by Lemma 4.19.

Now, notice that we are done when column 1 of P contains a $\bullet$. If so, $\tilde{P}_1 = P_1, \ldots, \tilde{P}_{C-1} = P_{C-1}$. Then $P_1 \lhd \cdots \lhd P_C \lhd S = \tilde{P}_1 \lhd \cdots \lhd \tilde{P}_C \lhd S$ by the previous claim.

Otherwise, we assume column 1 of P does not have a $\bullet$, so $\tilde{P}_1 = P_1 - n$. Notice that setting $T = P_1$, $S = P_2$ and $x = n$ would satisfies the conditions of Corollary 4.13. Thus, $n \notin P_1 \lhd P2$. Now by our first claim,

$$P_1 \lhd \cdots \lhd P_C \lhd S = P_1 \lhd \cdots \lhd P_{C-1} \lhd \tilde{P}_C \lhd S.$$

By $n \notin P_1 \lhd P2$ and Lemma 4.14, both sides of the equation above do not contain n. Thus,

$$P_1 \lhd \cdots \lhd P_{C-1} \lhd \tilde{P}_C \lhd S = \tilde{P}_1 \lhd P_2 \lhd \cdots \lhd P_{C-1} \lhd \tilde{P}_C \lhd S.$$

Then the proof is finished after replacing all P_i on the right by $\tilde{P}_i$. $\square$

Now we can prove Lemma 5.11.

Proof. Look at n and $\bullet$ in T. Assume there are J alternating ribbons with more than one cell. Label them with $1, 2, \ldots, J$ arbitrarily. Now let $P^0 = P$ and P^j is obtained from P^{j-1} by switching the alternating ribbon j to type 2. Then we know $P^J = \tilde{P}$. It remains to show $P_{1,C}^{j-1} = P_{1,C}^j$.

Assume the alternating ribbon j spans from column a_j to column b_j. Then P^{j-1} and P only differ between these two columns. Thus, column $a_j, \ldots, b_j$ of P^{j-1} form a dotted tableau in $\mathrm{sInc}_{n<\bullet}$. We may apply Lemma 5.12 and get

$$P_{a_j}^{j-1} \lhd \ldots \lhd P_{b_j}^{j-1} \lhd S = P_{a_j}^j \lhd \ldots \lhd P_{b_j}^j \lhd S$$

for any set S.

Finally, set $S = T_{b_j+1,C}^{j-1}$ if $b_j < C$. If $b_j = C$, set $S = \{1, \ldots, N\}$ for some N large enough. We have $P_{a_j,C}^{j-1} = P_{a_j,C}^j$, which implies $P_{1,C}^{j-1} = P_{1,C}^j$ $\square$

6. The reverse row insertion

Recall the main tool we used is a bijection $\Psi : \mathcal{T} \to \mathcal{C}$ of Huang, Shimozono and Yu. This map relies on a reverse row insertion algorithm on increasing tableaux. In Section 6.1, we describe the reverse row insertion algorithm which leads to the bijection $\Psi : \mathcal{C} \to \mathcal{T}$. In Section 6.2, we describe the change of left key when we apply the reverse insertion to a tableau. In Section 6.3, we prove Theorem 3.1.

6.1. **Describing the reverse row insertion.** Huang, Shimozono and Yu [HSY22] introduced a row analogue of the Hecke column insertion [BKS$^+$08]. Notice that in [HSY22], the authors described the algorithm on decreasing tableaux. For the purpose of this paper, we reverse all the comparisons and describe the algorithm on increasing tableaux. We start with the following definitions. For a tableau P, we let $P_{r\downarrow}$ denote the tableau obtained by removing the first $r-1$ rows of P.

Definition 6.1 ([HSY22]). A value x is *ejectable* in P if x occurs in the first row of P and either $x+1$ is not in the first row of P, or $x+1$ is in the first row of P and $x+1$ is ejectable in $P_{2\downarrow}$.

Definition 6.2. Let (r,c) be a cell in an increasing tableau P. The *bumping path* of (r,c) in P is the following sequence of cells in P: $(r,c_r),(r-1,c_{r-1}),\ldots,(1,c_1)$ defined recursively. First, $c_r := c$. Then c_i is the largest such that $P(i,c_i) < P(i+1,c_{i+1})$.

Clearly, a bumping path $(r,c_r),(r-1,c_{r-1}),\ldots,(1,c_1)$ satisfies $c_r \leq \cdots \leq c_1$.

We say a cell (r,c) in a tableau P is an *outer cell* in P if neither $(r+1,c)$ nor $(r,c+1)$ is in P. The reverse row insertion algorithm takes P, (r,c), and α as input where P is an increasing tableau, (r,c) is an outer cell in P and $\alpha = 0$ or 1. The output will be an increasing tableau P' and a positive integer m. The input α indicates whether P "loses the cell (r,c)":

$$\mathrm{shape}(P') = \begin{cases} \mathrm{shape}(P) & \text{if } \alpha = 0 \\ \mathrm{shape}(P) - \{(r,c)\} & \text{if } \alpha = 1. \end{cases}$$

Let $(r,c_r),(r-1,c_{r-1}),\ldots,(1,c_1)$ be the bumping path starting at (r,c) in P and let $m_i = P(i,c_i)$. The output value m is by definition the value m_1. The output tableau P' will only differ from P along the bumping path. It is enough to describe whether each m_i on the bumping path gets replaced, and if so, by what value. This decision is determined iteratively by decreasing i based on the values m_i and m_{i+1}, the i-th row of P, the subtableau $P'_{>i}$, and a status indicator $\alpha_{i+1} \in \{0,1\}$. The i-th iteration updates the i-th row of P (which becomes the i-th row of P') and produces $\alpha_i \in \{0,1\}$.

Let P' be a working tableau which is initialized to P. In the initialization step, if $\alpha = 1$, remove from P' the removable cell in row r and its contents m_r and set $\alpha_r = 1$ and $i = r-1$. If $\alpha = 0$ set $m_{r+1} = \infty$, $\alpha_{r+1} = 0$ and $i = r$.

The algorithm enters a loop. If $i = 0$ the algorithm terminates and the current tableau P' is the output tableau. Now assume $i \geq 1$. Let R be the set consisting of numbers in row i of the current tableau P' (or equivalently P, since P and P' differ only in rows of index greater than i). By definition $m_i \in R$. There are several cases for one iteration and each case has a nickname.

- **Dummy (Case D)**: If $m_i + 1 \in R$ (which implies $m_{i+1} = m_i + 1$) do not change the i-th row and set $\alpha_i = \alpha_{i+1}$.
- **Direct Replacement (Case DR)**: Otherwise if $\alpha_{i+1} = 1$ and $m_{i+1} \notin R$, replace m_i by m_{i+1} in row i of P' and set $\alpha_i = 1$.

Suppose neither of the two above cases hold. Find the largest ejectable entry x in $P'_{i+1\downarrow}$ such that $m_i < x < m_{i+1}$.

- **Indirect Replacement (Case IR)**: Suppose x exists. Replace m_i by x in row i of P' and set $\alpha_i = 1$.
- **No Replacement (Case NR)**: Suppose x does not exist. Do not change the i-th row and set $\alpha_i = 0$.

Now decrement i and go to the top of the loop.

Example 6.3. Let P be the increasing tableau below.

$$P = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & 5 \\ \hline 2 & 5 & 6 \\ \cline{1-3} 3 & 6 \\ \cline{1-2} 6 & 7 \\ \cline{1-2} 8 \\ \cline{1-1} \end{array}$$

Let (r, c) be $(4, 2)$ and $\alpha = 0$. First, the algorithm finds the bumping path starting at $(4, 2)$, which is highlighted by yellow. Thus, the output number m is 3. To compute the output tableau, it first initializes: $i = 4, \alpha_5 = 0$ and $m_5 = \infty$. Now We trace the iterations of this algorithm.

- Iteration when $i = 4$: We have $m_4 = 7$. Since $m_4 + 1 \notin R$, we are not in Case D. Since $\alpha_5 = 0$, we are not in Case DR. Then find the largest x that is ejectable in $P'_{5\downarrow}$ and $7 < x < \infty$. We have $x = 8$, so we are in **Case IR**. Replace the 7 in row 4 by 8 and set $\alpha_4 = 1, i = 3$.
- Iteration when $i = 3$: We have $m_3 = 6$. Since $m_3 + 1 \notin R$, we are not in Case D. Since $\alpha_4 = 1$ and $m_4 \notin R$, we are in **Case DR**. Replace the 6 in row 3 by 7 and set $\alpha_3 = 1, i = 2$.
- Iteration when $i = 2$: We have $m_2 = 5$. Since $m_2 + 1 = 6$ is in R, we are in **Case D**. We do not change row 2 and set $\alpha_3 = 1, i = 1$.
- Iteration when $i = 1$: We have $m_1 = 3$. Since $m_1 + 1 \notin R$, we are not in Case D. Since $m_2 \in R$, we are not in Case DR. Then find the largest x that is ejectable in $P'_{2\downarrow}$ and $3 < x < 5$. Such x does not exist, so we are in **Case NR**. We do not change row 1.

The tableau P' after each iteration is depicted as follows:

After 1 iteration   After 2 iterations   After 3 iterations   After 4 iterations

Huang, Shimozono and Yu also defined the insertion algorithm, which takes an increasing tableau P' and a positive integer m as input, producing a triple $(P, (r, c), \alpha)$. By Theorem 5.3 of [HSY22], the reverse insertion algorithm and insertion are inverses of each other. We deduce the following property of the insertion algorithm.

Lemma 6.4. *Let P be an increasing tableau without the number N. Perform row insertion on (P, m) and get $(P', (r, c), \alpha)$. Let p (resp. p') be the increasing tableau we get after ignoring all numbers larger than N in P (resp. p'). If $m \geq N$, then $p = p'$. If $m < N$, then p' is the tableau we get after performing row insertion on (p, m)*

Proof. We know (P, m) is obtained by reverse row insertion on $(P', (r, c), \alpha)$. If $m \geq N$, then the whole bumping path does not involve numbers smaller than N. Since P is obtained from P' by changing numbers in the bumping path, we know $p = p'$.

Now suppose $m < N$ and consider the reverse insertion on $(P', (r, c), \alpha)$. Let i be the largest such that $m_i < N$. Assume this value is at (i, j) of P. The algorithm behaves the same in the first i rows of P as computing the reverse insertion on $(p', (i, j), \alpha_{i+1})$. Thus, reverse insertion on $(p', (i, j), \alpha_{i+1})$ yields (p, m), so p' is the tableau we get after performing insertion on (p, m). $\square$

Now we may describe the bijection Ψ from $\mathcal{T}$ to $\mathcal{C}$ recursively. Given $(P, Q) \in \mathcal{T}$. If P and Q are both the empty tableaux, then $\Psi(P, Q)$ is the pair of two empty words. Otherwise, find the smallest number in Q and break ties by picking the rightmost. Say it is the number q in cell (r, c). Consider two cases.

- If q is the only number in (r, c) of Q, then we remove this cell and let Q' be the resulting tableau. Perform the reverse insertion on P with input (r, c) and $\alpha = 1$. Let P', m be the output.
- If q is not the only number in (r, c), we remove q from this cell and let Q' be the resulting tableau. Perform the reverse insertion on P with input (r, c) and $\alpha = 0$. Let P', m be the output.

For $(P', Q') \in \mathcal{T}$, we may apply the map Ψ recursively and get $\Psi(P', Q') = (a', i')$. Then $\Psi(P, Q) := (m \circ a', q \circ i')$ where $\circ$ represents concatenation of words.

Theorem 6.5. *[HSY22, Corollary 5.9] The map Ψ is a bijection from $\mathcal{T}_w$ to $\mathcal{C}_w$ that preserves the weight of the second entry. In other words, Ψ is bijective and if $\Psi(P, Q) = (a, i)$, then $[\mathrm{rev}(\mathrm{word}(P))]_H = [a]_H$ and $\mathrm{wt}(Q) = \mathrm{wt}(i)$.*

Example 6.6. Consider the following $(P, Q) \in \mathcal{T}$:

$$P = \begin{array}{|c|c|} \hline 1 & 2 \\ \hline 3 \\ \cline{1-1} \end{array} \qquad Q = \begin{array}{|c|c|} \hline 3 & 21 \\ \hline 21 \\ \cline{1-1} \end{array}.$$

We would like to compute $\Psi(P, Q)$. First, we find the smallest number in Q and break ties by picking the rightmost. That is the 1 in cell $(1, 2)$. Thus, we perform reverse insertion on P with input $(r, c) = (1, 2)$ and $\alpha = 0$. We get the output number 2 and the resulting (P', Q') is:

$$P' = \begin{array}{|c|c|} \hline 1 & 3 \\ \hline 3 \\ \cline{1-1} \end{array} \qquad Q' = \begin{array}{|c|c|} \hline 3 & 2 \\ \hline 21 \\ \cline{1-1} \end{array}.$$

Recursively, we can compute $\Psi(P', Q') = (1313, 1223)$. Thus, $\Psi(P, Q)$ is $(21313, 11223)$. The word $\mathrm{rev}(\mathrm{word}(P)) = 213$ and 21313 are Hecke words of the same permutation. The RSVT Q and the word 11223 have the same weight.

Finally, we can prove Lemma 2.13 stated in §2.

Proof of Lemma 2.13. By induction on the length of A. The inductive step is finished by Lemma 6.4. $\qquad\square$

6.2. **Change of left key under reverse insertion.** Shimozono and Yu [SY23] studied how the left key of an RSVT changes if we remove the rightmost appearance of the smallest number.

Lemma 6.7 ([SY23, Lemma 4.18 and Lemma 4.19])**.** *Let Q be an RSVT. Let $\min(Q) = i$ and let (r, c) be the rightmost cell with $i \in Q(r, c)$. Let Q' be the RSVT obtained by removing i from Q (also remove the cell (r, c) if i is the only number in it). If i is not the only number in Q, then clearly $K_-(L(Q)) = K_-(L(Q'))$. Otherwise, $K_-(L(Q))$ and $K_-(L(Q'))$ only differ at column c: $K_-(L(Q))_c = K_-(L(Q'))_c \sqcup y$ where*

$$y = \begin{cases} i & \text{if } c = 1, \\ \min(K_-(L(Q'))_{c-1} - K_-(L(Q'))_c) & \text{if } c > 1. \end{cases}$$

We are going to derive the analogue of this lemma for increasing tableau. Take an increasing tableau P. Fix an outer cell (R, C) of P and $\alpha = 0$ or 1 throughout this section. Let (P', m) be the result of reverse insertion with input $(P, (R, C), \alpha)$. We would like to show the following:

Theorem 6.8. *If $\alpha = 0$, $K_-(P) = K_-(P')$. Otherwise, $K_-(P)$ and $K_-(P')$ only differ at column C:*
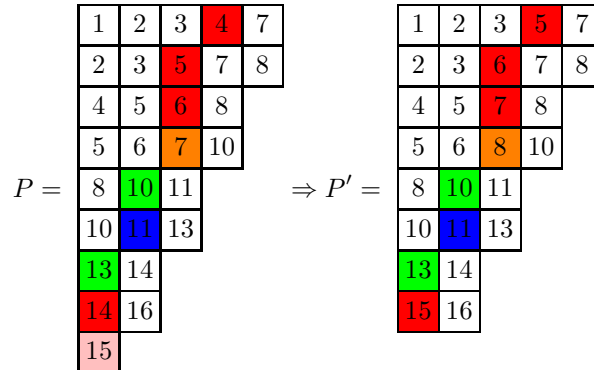
$$K_-(P')_C = K_-(P)_C - \min(K_-(P)_C \setminus K_-(P)_{C+1}).$$

The rest of this section aims to prove Theorem 6.8. We define the *block* in column c as the set of (r, c) such that $P(r, c) \neq P'(r, c)$. Recall the definition of bumping path from section §2. Let $(R, c_R), (R-1, c_{R-1}), \cdots, (1, c_1)$ be the bumping path of P that starts at (R, C). Since the algorithm only changes entries along the bumping path, we know each block consists of cells in the bumping path. Also recall that $c_R \leq c_{R-1} \leq \cdots \leq c_1$, so the block in column c, if non-empty, must be

$$\{(r_1, c), (r_1 + 1, c), \cdots, (r_2 - 1, c), (r_2, c)\}$$

for some $r_1 \leq r_2$. Moreover, we know the reverse insertion performs direct replacement in row $r_1, \ldots, r_2 - 1$. Thus, for any $r_1 \leq r < r_2$, $P'(r, c) = P(r + 1, c)$. If $\alpha = 1$ and $c = C$, $P'(r_2, c)$ is undefined. This block is also called the *deletion block*. Otherwise, the reverse insertion performs direct replacement or indirect replacement on row r_2. Such a block is also called an *insertion block.*

Example 6.9. Below is an example of the reverse-insertion algorithm applied on the cell $(9, 1)$ of P with $\alpha = 1$. We color the cells along the bumping path. The color is determined by the case of the algorithm in that row: red (DR), orange (IR), green (D), blue (NR), and pink (the initial removal).

Here, the deletion block is $\boxed{\begin{smallmatrix}14\\15\end{smallmatrix}}$ in the first column, and the insertion blocks are $\boxed{\begin{smallmatrix}5\\6\\7\end{smallmatrix}}$ in the third column, and $\boxed{4}$

in the fourth column. Notice that outside of these blocks, P and P' are the same, and each block only changes by adding a number and removing a number from the block.

We now view the effect of reverse insertion as changing the block in each column of P to obtain P'. We set $P^{(C-1)} = P$. For $c \geq C$, let $P^{(c)}$ be the tableau obtained from P^{c-1} by changing the block in column c if it exists, or $P^{(c)} = P^{(c-1)}$ if there is no block in column c. Then $P' = P^{(m)}$ for some m large enough. To prove Theorem 6.8, we just need to understand how $P^{(c)}$ differs from $P^{(c-1)}$. First, we study the effect of changing an insertion block.

Lemma 6.10. *Suppose the block in column c is an insertion block and let x be the value inserted to this block. Then $c > 1$ and $|(P_c)_{<x}| = |(P'_{c-1})_{<x}|$.*

Proof. Let (r, c) be the bottom-most cell of this insertion block, so $P(r, c) = x$ and $P'(r, c) > x$. Moreover, $P'(r, c - 1) = P(r, c - 1) < x$. We know x was inserted into (r, c) through a direct or indirect replacement, so $P(r + 1, c') = x$ for some c'. We consider two cases:

- Suppose the cell $(r + 1, c)$ does not exist in P, then trivially $c' < c$ and $|(P_c)_{<x}| = r$. If $(r + 1, c - 1)$ does not exist in P either, then $|(P_{c-1})_{<x}| = r$. Otherwise, $P'(r + 1, c - 1) \geq P(r + 1, c - 1) \geq x$, so $|(P'_{c-1})_{<x}| = r$.
- Suppose the cell $(r + 1, c)$ exists in P. By P' is increasing, $x < P'(r + 1, c) = P(r + 1, c)$, so $|(P_c)_{<x}| = r$. Then $c' < c$ and $x \leq P'(r + 1, c - 1)$, so $|(P'_{c-1})_{<x}| = r$. $\square$

Lemma 6.11. *Suppose the block in column c is an insertion block and let x be the value inserted to this block. Then the insertion block removes the number $\min\{t \in P_c : x \sqcup P_c - t \preceq P_{c+1}\}$.*

Proof. Let z be the number removed by the insertion block in column c. We have $P'_c = x \sqcup P_c - z$ and $P'_c \preceq P_{c+1}$. Assume $P(r, c) = z$. By definition of the insertion block, the algorithm cannot perform another direct replacement step in column c in row $r - 1$. Thus, either $r = 1$ or $z \geq P_{c+1}(r - 1)$. In either case, for any $i \in [r - 1]$, we cannot have $x \sqcup P_c - P_c(i) \preceq P_{c+1}$. Our claim is therefore proved. $\square$

We now understand the effect of changing an insertion block.

Lemma 6.12. *Suppose the block in column c is an insertion block, then $K_-(P^{(c)}) = K_-(P^{(c-1)})$.*

Proof. Say the insertion block gains x and loses y, so $P_c^{(c)} = x \sqcup P_c^{(c-1)} - y$. By Corollary 5.10, $K_-(P^{(c-1)})_i = P_{1,i}^{(c-1)}$ and $K_-(P^{(c)})_i = P_{1,i}^{(c)}$. Since $P^{(c-1)}$ and $P^{(c)}$ only differ in column c, we know $K_-(P^{(c-1)})_i = K_-(P^{(c)})_i$ if $i < c$.

By Lemma 6.10, $c > 1$. Notice that $P_{c-1}^{(c-1)} = P_{c-1}^{(c)} = P'_{c-1}$, $P_{c+1}^{(c-1)} = P_{c+1}^{(c)} = P_{c+1}$, $P_c^{(c-1)} = P_c$, and $P_c^{(c)} = P'_c$. We set $U = P'_{c-1}$, $T = P_c$ and $S = P_{c+1}$. By Lemma 6.10 and Lemma 6.11, U, T, S and x, y satisfy the conditions of Lemma 4.16. Thus, $P'_{c-1} \lhd P_c = P'_{c-1} \lhd P'_c$ and

$$P'_{c-1} \lhd P_c \lhd S' = P'_{c-1} \lhd P'_c \lhd S'.$$

for any $S' \subseteq P_{c+1}$. The first equation implies $P'_{1,c} = P_{1,c}$. The second equation implies $P'_{1,i} = P_{1,i}$ if $i > c$. $\square$

Next, we study the deletion block.

Lemma 6.13. *Suppose the block in column c is a deletion block. Then this block loses $\min\{t \in P_c : P_c - t \preceq P_{c+1}\}$.*

Proof. By a similar proof of Lemma 6.11. $\square$

Lemma 6.14. *If a deletion block occurs in column c, then $K_-(P)$ and $K_-(P^{(c)})$ agree except in column c, where*

$$(6.1) \qquad K_-(P^{(c)})_c = K_-(P)_c - \min(K_-(P)_c \setminus K_-(P)_{c+1})$$

Proof. Recall that $P^{(c)}$ is obtained from P by changing the deletion block. Assume the deletion block loses y, so $P_c^{(c)} = P_c - y$. By Lemma 6.13, $y = \min\{t \in P_c : P_c - t \preceq P_{c+1}\}$.

By Corollary 5.10, column i of $K_-(P)$ and $K_-(P^{(c)})$ are $P_{1,i}$ and $P_{1,i}^{(c)}$ respectively. They clearly agree if $i < c$. Suppose $i > c$. To show $P_{1,i} = P_{1,i}^{(c)}$, it is enough to check $P_{c,i} = P_{c,i}^{(c)}$, which is $P_c \lhd P_{c+1,i} = (P_c - y) \lhd P_{c+1,i}$. Since $P_{c+1,i} \subseteq P_{c+1}$, this equation is implied by the third statement in Lemma 4.17.

Finally, we show $P_{i,c}^{(c)} = P_{i,c} - \min(P_{i,c} \setminus P_{i,c+1})$ for all $i \in [c]$ by induction on i from $i = c$ to $i = 1$. Setting $i = 1$ would imply (6.1). For the base case $i = c$, we know $P_{c,c}^{(c)} = P_{c,c} - y$. It remains to check y is the smallest number in

P_c that is not in $P_{c,c+1}$. This is done by the first two statements in 4.17. For the inductive step, assume $i < c$. We have

$$P_{i,c}^{(c)} = P_i \lhd P_{i+1,c}^{(c)} = P_i \lhd (P_{i+1,c} - \min(P_{i+1,c} \setminus P_{i+1,c+1})) = P_{i,c} - \min(P_{i,c} \setminus P_{i,c+1}),$$

where the last two steps are given by the inductive hypothesis and Lemma 4.15 respectively. $\qquad\square$

Now we have understood how $K_-(P^{(c)})$ differs from $K_-(P^{(c-1)})$.

Proof of Theorem 6.8. Consider the sequence of increasing tableaux:

$$P = P^{(C-1)}, P^{(C)}, P^{(C+1)}, \cdots, P^{(m-1)}, P^{(m)} = P',$$

If $\alpha = 0$, then there is no deletion block. By Lemma 6.12, $K_-(P) = K_-(P^{(C)}) = \cdots = K_-(P')$. If $\alpha = 1$, then the block in column c is a deletion block and all other blocks are insertion blocks. By Lemma 6.12, $K_-(P^{(C)}) = K_-(P^{(C+1)}) = \cdots = K_-(P')$. Then by Lemma 6.14, $K_-(P')$ agrees with $K_-(P)$ except in column c where $K_-(P')_c = K_-(P)_c - \min(K_-(P)_c \setminus K_-(P)_{c+1})$. $\qquad\square$

6.3. **Proof of Theorem 3.1.** In this section, we prove Theorem 3.1 using similar idea as [SY23, Theorem 4.2].

Proof of Theorem 3.1. Since Ψ is a bijection from $\mathcal{T}$ to $\mathcal{C}$, it suffices to show that for any $(P,Q) \in \mathcal{T}^b$, $\Psi(P,Q) \in \mathcal{C}^b$ and for any $(a,i) \in \mathcal{C}^b$, $\Psi^{-1}(a,i) \in \mathcal{T}^b$.

Take $(P,Q) \in \mathcal{T}^b$ where Q has at least one cell. Let $i_1 = \min(Q)$. Let (r,c) be the rightmost cell in Q with $i_1 \in Q(r,c)$. Let $\alpha = 1$ if i_1 is the only number in $Q(r,c)$ and $\alpha = 0$ otherwise. Let (P', a_1) be the output of reverse insertion on $(P, (r,c), \alpha)$. Let Q' be the tableau obtained from Q by removing i_1 from $Q(r,c)$, and also remove the cell (r,c) if $\alpha = 1$. Due to the recursive definition of Ψ, $\Psi(P,Q) = (a_1 \circ a', i_1 \circ i')$ where $(a', i') = \Psi(P', Q')$. Notice that $(a,i) \in \mathcal{C}^b$ is equivalent to $a_1 \leq i_1$ and $(a', i') \in \mathcal{C}^b$. Inductively, we may assume $(a', i') \in \mathcal{C}^b$ if and only if $(P', Q') \in \mathcal{T}^b$. Thus, it suffices to show the following two statements:

- If $(P,Q) \in \mathcal{T}^b$, then $a_1 \geq i_1$ and $(P', Q') \in \mathcal{T}^b$.
- If $a_1 \geq i_1$ and $(P', Q') \in \mathcal{T}^b$, then $(P,Q) \in \mathcal{T}^b$.

We start with the first statement. To see $a_1 \geq i_1$, observe that $i_1 = \min(Q)$ implies $i_1 \leq \min(L(Q))$. By the definition of $K_-(\cdot)$ on RSSYT, $\min(L(Q)) \leq \min(K_-(L(Q)))$, which in turn is at most $\min(K_-(P)) \leq \min(P) \leq a_1$. Next we show $(P', Q') \in \mathcal{T}^b$. If $\alpha = 0$, then Theorem 6.7 and Theorem 6.8 yield $K_-(L(Q')) = K_-(L(Q)) \leq K_-(P) = K_-(P')$. Now suppose $\alpha = 1$. By Theorem 6.8 the only column where $K_-(P)$ and $K_-(P')$ differ is column c and $(K_-(P'))_c = (K_-(P))_c - x$, where $x = \min((K_-(P))_c \setminus (K_-(P))_{c+1})$. By Theorem 6.7, the only column where $K_-(L(Q))$ and $K_-(L(Q'))$ differ is column c and $(K_-(Q'))_c = (K_-(Q))_c - y$ for some y. It remains to check for each $j \in [|K_-(P')_c|]$, $K_-(P')_c(j) \geq K_-(Q')_c(j)$.

For j such that $(K_-(P))_c(j) \geq x$,

$$(K_-(P'))_c(j) = (K_-(P))_c(j+1) \geq (K_-(L(Q)))_c(j+1) \geq (K_-(L(Q')))_c(j).$$

For j such that $(K_-(P))(j) < x$,

$$(K_-(P'))_c(j) = (K_-(P'))_{c+1}(j) \geq (K_-(L(Q')))_{c+1}(j) \geq (K_-(L(Q')))_c(j).$$

Therefore, for all j, $(K_-(P'))_c(j) \geq (K_-(L(Q')))_c(j)$, which shows that $K_-(P') \geq K_-(L(Q'))$.

We now show the second statement. If $\alpha = 0$, then Theorem 6.7 and Theorem 6.8 yield $K_-(L(Q')) = K_-(L(Q)) \leq K_-(P) = K_-(P')$. Otherwise, $\alpha = 1$. By Theorem 6.8, the only column where $K_-(P)$ and $K_-(P')$ differ is column c and $(K_-(P))_c = (K_-(P'))_c \sqcup x$ for some x. By Theorem 6.7, the only column where $K_-(L(Q))$ and $K_-(L(Q'))$ differ is column c, and $K_-(L(Q))_c = K_-(L(Q'))_c \sqcup y$, where

$$y = \begin{cases} i_1 & \text{if } c = 1, \\ \min(K_-(L(Q'))_{c-1} - K_-(L(Q'))_c) & \text{if } c > 1. \end{cases}$$

It remains to check for each $j \in [|K_-(P)_c|]$, $K_-(P)_c(j) \geq K_-(Q)_c(j)$. For j such that $(K_-(L(Q')))_c(j) > y$,

$$(K_-(P))_c(j) \geq (K_-(P'))_c(j-1) \geq (K_-(L(Q')))_c(j-1) = (K_-(L(Q)))_c(j).$$

Now assume $(K_-(L(Q')))_c(j) \leq y$. If $c = 1$, $y = i_1$ is the smallest number in i, so it is the smallest number in Q. We know $(K_-(L(Q')))_c(j) = y$ and $y \leq \min(a) = \min(P) \leq (K_-(P))_c(j)$. If $c > 1$, we have

$$K_-(P)_c(j) \geq K_-(P)_{c-1}(j) \geq K_-(L(Q))_{c-1}(j) = K_-(L(Q))_c(j)$$

This completes the proof. $\qquad\square$

Finally, we mention a by-product of Theorem 3.1: the Grothendieck-to-Lascoux expansion. It was conjectured by [RY21] and proved in [SY23].

Corollary 6.15. *For a permutation* $w \in S_n$,
$$\mathfrak{S}_w^{(\beta)} = \sum_{\substack{P \in \mathrm{Inc} \\ \mathsf{word}(P) = w^{-1}}} \beta^{|\mathrm{wt}(K_-(P))| - \ell(w)} \mathfrak{L}_{\mathrm{wt}(K_-(P))}^{(\beta)}.$$

Proof. This follows from Theorem 3.1, Equation 2.4, and Equation 2.1. □

7. Acknowledgements

Appendix A. Example of Theorem 1.3

Suppose $n = 3$. Say we would like to expand $\mathfrak{L}_\alpha^{(\beta)} \times G_w^{(\beta)}(x_1, \cdots, x_n)$ into Lascoux polynomials where $\alpha = (1, 0, 2)$ and w has one-line notation 12345876. We let

$$P_1 = \begin{array}{|c|c|} \hline 1 & 4 \\ \hline 3 \\ \cline{1-1} \end{array},$$

so $K_-(P_1) = (1, 0, 2)$. We pick $N = 5$. Now we are looking for increasing tableau with at most 3 rows such that
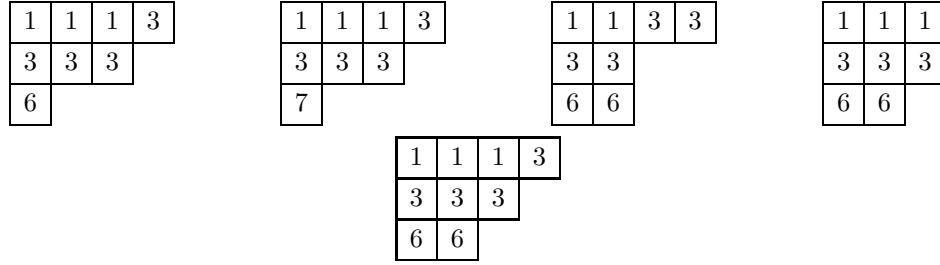
- Entries smaller than 5 form the P_1 we picked, and
- Entries of word(P) that are larger than 5 form a Hecke word of w.

The following are all increasing tableaux satisfying both conditions:

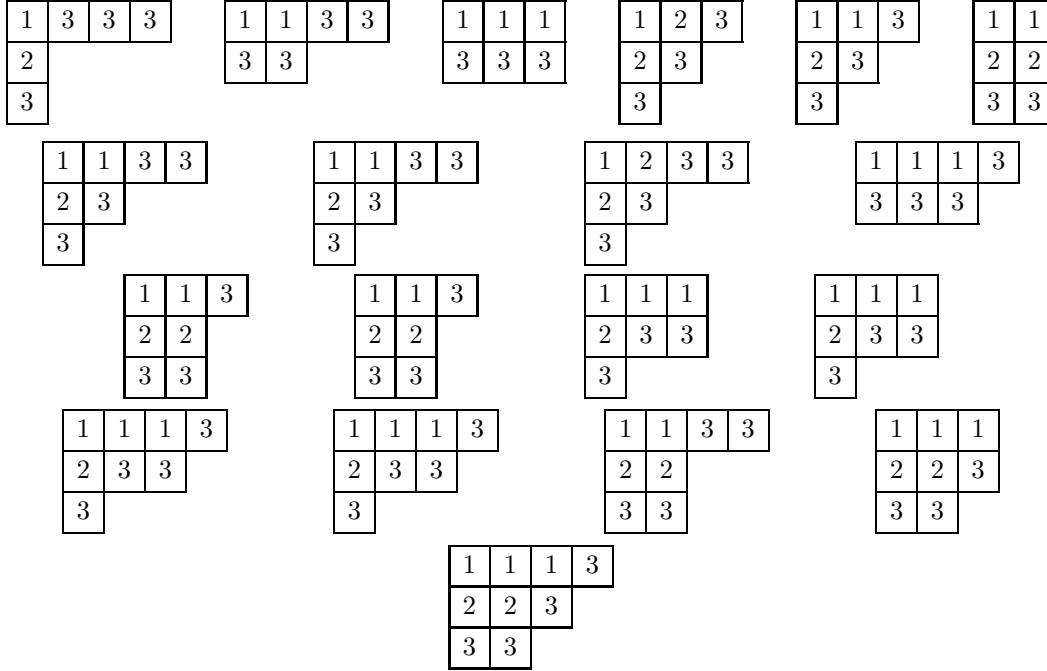After applying $K_-(\cdot)$, we obtain the keys:

```
1 1 1 3      1 1 1 3      1 1 3 3      1 1 1
3 3 3        3 3 3        3 3          3 3 3
6            7            6 6          6 6
```

```
1 1 1 3
3 3 3
6 6
```

We apply cap_3 to each key and get:

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

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

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

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

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

Finally, we apply $wt(\cdot)$ and get:

$$(1,1,4), \qquad (2,0,4), \qquad (3,0,3), \qquad (1,2,3), \qquad (2,1,3), \qquad (2,2,2)$$
$$(2,1,4), \qquad (2,1,4), \qquad (1,2,4), \qquad (3,0,4)$$
$$(2,2,3), \qquad (2,2,3), \qquad (3,1,3), \qquad (3,1,3)$$
$$(3,1,4), \qquad (3,1,4), \qquad (2,2,4), \qquad (3,2,3)$$
$$(3,2,4)$$

As a conclusion, we expand $\mathfrak{L}_\alpha^{(\beta)} \times G_w^{(\beta)}(x_1, x_2, x_3)$ into:

$$\mathfrak{L}_{(1,1,4)}^{(\beta)} + \mathfrak{L}_{(2,0,4)}^{(\beta)} + \mathfrak{L}_{(3,0,3)}^{(\beta)} + \mathfrak{L}_{(1,2,3)}^{(\beta)} + \mathfrak{L}_{(2,1,3)}^{(\beta)} + \mathfrak{L}_{(2,2,2)}^{(\beta)}$$
$$+2\beta\mathfrak{L}_{(2,1,4)}^{(\beta)} + \beta\mathfrak{L}_{(1,2,4)}^{(\beta)} + 2\beta\mathfrak{L}_{(2,2,3)}^{(\beta)} + 2\beta\mathfrak{L}_{(3,1,3)}^{(\beta)}$$
$$+2\beta^2\mathfrak{L}_{(3,1,4)}^{(\beta)} + \beta^2\mathfrak{L}_{(2,2,4)}^{(\beta)} + \beta^2\mathfrak{L}_{(3,2,3)}^{(\beta)} + \beta^3\mathfrak{L}_{(3,2,4)}^{(\beta)}$$

REFERENCES

[BJS93] Sara C Billey, William Jockusch, and Richard P Stanley. Some combinatorial properties of Schubert polynomials. *Journal of Algebraic Combinatorics*, 2(4):345–374, 1993.

[BKS⁺08] Anders Skovsted Buch, Andrew Kresch, Mark Shimozono, Harry Tamvakis, and Alexander Yong. Stable Grothendieck polynomials and K-theoretic factor sequences. *Mathematische annalen*, 340:359–382, 2008.

[BSW20] Valentin Buciumas, Travis Scrimshaw, and Katherine Weber. Colored five-vertex models and Lascoux polynomials and atoms. *Journal of the London Mathematical Society*, 102(3):1047–1066, 2020.

[Buc02] Anders Skovsted Buch. A Littlewood-Richardson rule for the K-theory of Grassmannians. *Acta Mathematica*, 189:37–78, 2002.

[Dem74] Michel Demazure. Une nouvelle formule des caracteres. *Bull. Sci. Math*, 2(98):163–172, 1974.

[FK96]   Sergey Fomin and Anatol N Kirillov. The Yang-Baxter equation, symmetric functions, and Schubert
           polynomials. *Discrete Mathematics*, 153(1-3):123–143, 1996.

[HLMvW11] James Haglund, Kurt Luoto, Sarah Mason, and Stephanie van Willigenburg. Refinements of the
           Littlewood-Richardson rule. *Transactions of the American Mathematical Society*, 363(3):1665–1686,
           2011.

[HSY22]   Daoji Huang, Mark Shimozono, and Tianyi Yu. A row analogue of Hecke column insertion. *arXiv
           preprint arXiv:2211.02993*, 2022.

[Hud14]   Thomas Hudson. A Thom-Porteous formula for connective K-theory using algebraic cobordism. *Journal
           of K-theory*, 14(2):343–369, 2014.

[Las03]   Alain Lascoux. Schubert and Grothendieck: a bidecennial balance.(Schubert et Grothendieck: un bilan
           bidécennal.). *Séminaire Lotharingien de Combinatoire [electronic only]*, 50:B50i–32, 2003.

[LS82]   Alain Lascoux and Marcel-Paul Schützenberger. Structure de Hopf de l'anneau de cohomologie et de
           l'anneau de Grothendieck d'une variété de drapeaux. *CR Acad. Sci. Paris Sér. I Math*, 295(11):629–633,
           1982.

[LS88]   A Lascoux and MP Schützenberger. Keys and standard bases, Invariant Theory and Tableaux, The
           IMA volumes in Mathematics and its Applications, 19, 1988.

[Mon16]   Cara Monical. Set-valued skyline fillings. *arXiv preprint arXiv:1611.08777*, 2016.

[RY21]   Victor Reiner and Alexander Yong. The "Grothendieck to Lascoux" conjecture. *arXiv preprint
           arXiv:2102.12399*, 2021.

[Sta84]   Richard P Stanley. On the number of reduced decompositions of elements of Coxeter groups. *European
           Journal of Combinatorics*, 5(4):359–372, 1984.

[SY23]   Mark Shimozono and Tianyi Yu. Grothendieck-to-Lascoux expansions. *Transactions of the American
           Mathematical Society*, 2023.

[TY09]   Hugh Thomas and Alexander Yong. A jeu de taquin theory for increasing tableaux, with applications
           to K-theoretic Schubert calculus. *Algebra & Number Theory*, 3(2):121–148, 2009.

[Wil13]   Matthew J Willis. A direct way to find the right key of a semistandard Young tableau. *Annals of
           Combinatorics*, 17(2):393–400, 2013.

(G. Orelowitz)
Email address: `gidon.orelowitz@gmail.com`

(T. Yu) DEPARTMENT OF MATHEMATICS, UC SAN DIEGO, LA JOLLA, CA 92093, U.S.A.
Email address: `tiy059@ucsd.edu`