Combinatorial Aspects of the Lascoux-Schützenberger Tree

David P. Little

November 19, 2001

Abstract

In 1982, Richard Stanley introduced the formal series $F_\sigma(X)$ in order to enumerate reduced decompositions of a given permutation $\sigma$. In [5], he not only showed $F_\sigma(X)$ to be symmetric, but in certain cases, $F_\sigma(X)$ was a Schur function. Stanley conjectured that for arbitrary $\sigma$, $F_\sigma(X)$ was always Schur positive. Edelman and Greene subsequently proved this fact [1],[2]. Using the techniques of Lascoux and Schützenberger [4] for computing Littlewood-Richardson coefficients, we will exhibit a new bijective proof of the Schur positivity of $F_\sigma(X)$.

1 Introduction

In the following, we will think of $S_n$, the symmetric group on $n$ letters, as being generated by the simple transpositions $s_1, s_2, \ldots, s_{n-1}$ where $s_i = (i, i+1)$. These generators satisfy the Coxeter relations:

\begin{align}
\text{a) } & s_i^2 = id \text{ for } 1 \leq i \leq n - 1, \\
\text{b) } & s_i s_j = s_j s_i \text{ for } |i - j| \geq 2, \\
\text{c) } & s_i s_{i+1} s_i = s_{i+1} s_i s_{i+1} \text{ for } 1 \leq i < n - 1. 
\end{align}

The length of $\sigma \in S_n$, denoted $l(\sigma)$ is the number of inversions of $\sigma$. Symbolically we have

\[ l(\sigma) = \left| \{i < j : \sigma_i > \sigma_j \} \right|. \]

A word, $w = a_1 a_2 \cdots a_k$, in the alphabet $\{1, 2, \ldots, n-1\}$ corresponds to the permutation $\sigma$ if

\[ \sigma = s_{a_1} s_{a_2} \cdots s_{a_k}. \]

The permutation corresponding to $w$ is denoted $\sigma_w$. The descent set of a word, denoted $Des(w)$, is given by

\[ Des(w) = \{i \mid a_i > a_{i+1} \}. \]

A word, $w$, of length $k$ is said to be reduced if $k = l(\sigma_w)$. The set of all reduced words corresponding to $\sigma$ is denoted by $Red(\sigma)$. A fundamental problem is to determine the
cardinality of \( \text{Red}(\sigma) \) for a fixed permutation \( \sigma \). To this end, Stanley [5] introduced the functions \( F_{\sigma}(X) \) defined as

\[
F_{\sigma}(X) = \sum_{w \in \text{Red}(\sigma)} \sum_{b_1 \leq b_2 \leq \cdots \leq b_l} x_{b_1} x_{b_2} \cdots x_{b_l}.
\]

Because the sequence \( 1 < 2 < \cdots < l \) is accounted for exactly once for each reduced word, the number of reduced words corresponding to \( \sigma \) is simply given by the coefficient of \( x_1 x_2 \cdots x_l \) in \( F_{\sigma}(X) \). Stanley showed in [5] that for \( \sigma = (n, n-1, \ldots, 1) \), the element of longest length, the number of reduced words is equal to the number of standard Young tableaux of staircase shape \( (n-1, n-2, \ldots, 1) \). Stanley also showed that \( F_{\sigma}(X) \) is symmetric and conjectured that it was Schur positive [1],[2],[3].

Since we will be dealing extensively with reduced words for a given permutation, it will greatly help our understanding to have a graphical representation of these words. To this end, we associate a line diagram to each word \( w \), denoted \( LD(w) \), which illustrates the trajectories of the numbers \( 1, 2, \ldots, n \) as they are rearranged into the permutation \( \sigma_w \) by successive simple transpositions. For example, the line diagram in Figure 1 corresponds to the word \( 3,4,1,5,4,6,2 \). Notice that \( w \) need not be reduced in order to construct \( LD(w) \). In fact, it can be easily shown using the relations in (1) that \( w \) is reduced if and only if no two lines cross more than once in \( LD(w) \).

Our main contribution in this paper is the construction of a correspondence \( \Theta_{\sigma} \) which sends \( w \in \text{Red}(\sigma) \) to \( \Theta_{\sigma}(w) = (\alpha(w), T(w)) \), where \( \alpha(w) \) is a Grassmanian permutation\(^1\) and \( T(w) \) is a standard tableau of shape \( \lambda^t(\alpha(w)) \). It follows from our correspondence that we have

\[
F_{\sigma}(X) = \sum_{\alpha(w) \in \mathcal{A}(\sigma)} S_{\lambda^t(\alpha(w))}(X)
\]

where the sum is over the collection of Grassmanian permutations

\[
\mathcal{A}(\sigma) = \{ \alpha \mid \alpha = \alpha(w) \text{ for some } w \in \text{Red}(\sigma) \}.
\]

\(^1\)A permutation with only one descent.
The identity in 2 derives from the following basic property of our correspondence:

**Property 1** For each $\sigma \in S_n$ the map $\Theta_\sigma$ is a descent preserving bijection of $\text{Red}(\sigma)$ onto $\bigcup_{\alpha \in A(\sigma)} \text{Red}(\alpha)$.

As a byproduct we also obtain

**Property 2** $\Theta_\sigma$ maps $\text{Red}(\sigma)$ bijectively onto the collection of pairs

$$\{(\alpha, T) \mid \alpha \in A(\sigma) \land \lambda(T) = \lambda'(\alpha)\}.$$  

The map $\Theta_\sigma$ is obtained by a sequence of very simple transformations acting on line diagrams. For a given $w \in \text{Red}(\sigma)$ we start with the line diagram corresponding to $w$ and end up with the line diagram of a word $w' \in \text{Red}(\alpha)$ for some $\alpha \in A(\sigma)$. A reader that wishes to experiment with our bijections may find a description of its construction and a Java Applet implementing it at the following address:

http://math.ucsd.edu/~dlittle/linediagrams

Our proof that $\Theta_\sigma$ is a bijection is quite simple and its properties can be established in a straightforward manner. Thus another byproduct of our correspondence is an elementary proof of the Schur positivity of the Stanley symmetric functions. In particular we obtain from (2) a very natural combinatorial interpretation for the multiplicities of the Schur functions occurring in the expansion of any given $F_w(X)$. Since the Stanley symmetric functions are natural generalizations of skew Schur functions, we also obtain a very simple and purely combinatorial proof of the Lascoux-Schützenberger [4] version of the Littlewood-Richardson rule.

Experimentation with the applet quickly reveals that when $\sigma = (2, 1, 4, 3, \ldots, 2n, 2n-1)$, $\Theta_\sigma$ is essentially the Robinson-Schensted correspondence and when $\sigma$ is the top permutation $\sigma = (n, \ldots, 3, 2, 1)$, $\Theta_\sigma$ reduces to the Edelman-Greene correspondence [2]. At the moment, this is purely conjectural. The author will deal with these findings in a forthcoming publication.

## 2 Circle Diagrams

The *circle diagram* of $\sigma$, denoted $CD(\sigma)$, is an $n \times n$ array where the rows are labeled from 1 to $n$ (top to bottom) and the $i^{th}$ column is labeled by $\sigma_i$. In each column, a single “$\times$” is placed in the row indicated by the column index. A “$\bullet$” is placed in each cell that occurs directly below or to the right of an “$\times$”. The remaining cells are filled with a “$\bigcirc$”. The circle diagram for the permutation $(2, 4, 1, 6, 5, 7, 3)$ is given in Figure 2.

Notice that each “$\bigcirc$” that occurs in $CD(\sigma)$ corresponds to an inversion of $\sigma$. This is because if a “$\bigcirc$” occurs in the row labeled $i$ and the column labeled $j$, then the “$\times$” in row $i$ appears to its right, or in other words $i$ appears to the right of $j$ in $\sigma$. Additionally, the “$\times$” in the column labeled $j$ must appear below the “$\bigcirc$”, which means that $j$ is bigger than $i$. 

---

3
This simple observation allows us to specify a reduced word, $w$, of $\sigma$ by numbering each “$\bigcirc$” in $CD(\sigma)$ with the time at which the inversion first appears in $LD(w)$. In other words, for each reduced word, $w$, we may associate a labeled circle diagram, denoted $LCD(w)$. This is accomplished by placing a $t$ in the row labeled $i$ and the column labeled $j$ of $CD(w)$ if lines $i$ and $j$ cross at time $t$ in $LD(w)$, as illustrated in Figure 3. The labeled circle diagram corresponding to the word $3,4,1,5,4,6,2$ is given in Figure 4.

We will also make a limited use of labeled circle diagrams corresponding to words that are not reduced. In this case the labeled circle diagram is constructed in the exact same manner, however, some labels will inevitably overlap a “•”, as is the case in Figure 5.
opens the door to the possibility that more than two numbers label the same cell. In general this would be a concern, but in the following, we will carefully avoid this situation.

The code of $\sigma$, denoted $c(\sigma)$, is the sequence

$$
c(\sigma) = (c_1(\sigma), c_2(\sigma), \ldots, c_n(\sigma))
$$

where

$$
c_i = \#\{j > i \mid \sigma_j < \sigma_i\}.
$$

In other words, $c_i(\sigma)$ is the number of circles that appear in the $i^{th}$ column of $CD(\sigma)$. The shape of $\sigma$, denoted $\lambda(\sigma)$, is the partition of $l(\sigma)$ corresponding to the decreasing rearrangement of $c(\sigma)$. We use $\lambda'(\sigma)$ to denote the conjugate shape of $\lambda(\sigma)$. Using Figure 2, we see that the code for $\sigma = (2, 4, 1, 6, 3, 5, 7)$ is $(1, 2, 0, 2, 1, 1, 0)$, $\lambda(\sigma) = (2, 2, 1, 1, 1)$ and $\lambda'(\sigma) = (5, 2)$.

For certain permutations, the circle diagram holds special significance. For example, we say that $\sigma$ is 321-avoiding if there does not exist indices $i < j < k$ such that $\sigma_i > \sigma_j > \sigma_k$. In this case, there is a French skew shape that can be associated with $\sigma$. That is to say, if one were to remove the rows and columns that do not contain a "$\bigcirc$" in $CD(\sigma)$, the remaining cells would form a French skew shape. Additionally, if $w$ is a word corresponding to a 321-avoiding permutation, then $LCD(w)$ can be thought of as a standard tableau of French skew shape, which we will denote by $T(w)$.

Note that Grassmanian permutations are a special case of 321-avoiding permutations. In particular, let $w \in Red(\sigma)$ where $\sigma$ is Grassmanian. If $T(w)$ is reflected about a vertical
line and each entry $i$ is replaced by $l(\sigma) + 1 - i$, then the resulting tableau, $\tilde{T}(w)$, would be a standard tableau of shape $\lambda'(\sigma)$. Therefore when $\sigma$ is Grassmanian, we simply set $\Theta_\sigma(w) = (\sigma, \tilde{T}(w))$.

To see how $\Theta_\sigma$ acts for an arbitrary $\sigma$, we require the Lascoux-Schützenberger version of the Littlewood-Richardson rule.

### 3 Lascoux-Schützenberger Tree

In [4], Lascoux and Schützenberger announced a new algorithm for computing the Littlewood-Richardson coefficients. To describe this algorithm they introduced a tree associated to each permutation. Defining this tree requires the following definitions. For a fixed $\sigma \in S_n$, let

$$
\begin{align*}
r &= \max(i \mid \sigma_i > \sigma_{i+1}), \\
s &= \max(i > r \mid \sigma_i < \sigma_r), \\
I &= \{i < r \mid \sigma_i < \sigma_r \text{ and } \forall j \in (i, r) \sigma_j \notin (\sigma_i, \sigma_s)\}.
\end{align*}
$$

(3)

In other words, $r$ is the index of the last descent in $\sigma$, $s$ is the index of the largest number following $\sigma_r$ that is less than $\sigma_r$, and $I$ is the set of indices, $i < r$, such that

$$
l(\sigma_{t_r}, t_{i,r}) = l(\sigma),
$$

(4)

where $t_{i,j}$ is the transposition $(i,j)$. For a given $\sigma$, let

$$
\Phi(\sigma) = \{\sigma_{t_r}, t_{i,r} \mid i \in I\}
$$

If $\sigma' \in \Phi(\sigma)$, we say that $\sigma'$ is a child of $\sigma$, denoted $\sigma \rightarrow \sigma'$. For example, if $\sigma = (2, 4, 1, 6, 5, 7, 3)$, then $r = 6$, $s = 7$, $I = \{1, 3\}$ and

$$
\Phi(\sigma) = \{(3, 4, 1, 6, 5, 2, 7), (2, 4, 3, 6, 5, 1, 7)\}.
$$

In the event that $I = \emptyset$, then $\Phi(\sigma)$ is defined as

$$
\Phi(\sigma) = \Phi(1 \otimes \sigma).
$$
where

$$1 \otimes \sigma = \begin{pmatrix} 1 & 2 & 3 & \ldots & n+1 \\ 1 & \sigma_1+1 & \sigma_2+1 & \ldots & \sigma_n+1 \end{pmatrix}. \quad (5)$$

Note that $I$ is empty if for all $i < r$, $\sigma_i > \sigma_r$. In other words, the largest $i < r$ that satisfies $\sigma_i < \sigma_r$ is automatically in $I$. While this operation has the property that

$$F_\sigma(X) = F_{1 \otimes \sigma}(X),$$

it is somewhat lacking in motivation. In the next section, we will put this construction in a combinatorial setting where it turns out this is the natural thing to do.

The Lascoux-Schützenberger (L-S) Tree corresponding to $\sigma$ is obtained by recursively applying $\Phi$ until every child is Grassmanian. The L-S Tree corresponding to $(3,5,1,4,2,7,6)$ is shown in Figure 8. For each permutation, the numbers $\sigma_r$, $\sigma_s$, and $\sigma_i$ for $i \in I$, are boxed, circled, and underlined, respectively. For each permutation $\sigma$, if $I$ is empty, then $\sigma$ is replaced by $1 \otimes \sigma$. Attached to each leaf of the L-S Tree is the Ferrers diagram corresponding to its conjugate shape. For further information regarding the L-S tree, the reader is referred to [4].

As it turns out, $F_\sigma(X)$ can be written as $\sum F_\mu(X)$, where the sum is over permutations $\mu$ that appear as leaves of the L-S tree corresponding to $\sigma$. This result can be derived from the following identity.
Theorem 1 For any permutation $\sigma$, we have

$$F_\sigma(X) = \sum_{\sigma' \sim \sigma} F_{\sigma'}(X).$$

As shown in [3], this is a simple consequence of Monk’s rule for Schubert polynomials. However, the remainder of the paper is dedicated to proving this fact combinatorially.

Before we move on, we should point out how this result yields the Schur positivity of $F_\sigma(X)$. To this end, we say that a permutation is vexillary if it does not contain the pattern $2143$, i.e., there does not exist indices $i < j < k < l$ such that $\sigma_j < \sigma_i < \sigma_l < \sigma_k$. If a permutation does contain such a pattern, it will necessarily have at least 2 descents. Since a Grassmanian permutation has only one descent, it is also vexillary. Schur positivity then follows from the following theorem of Stanley.

Theorem 2 (Stanley) If $\sigma$ is vexillary then $F_\sigma(X) = S_{\lambda(\sigma)}(X)$.

Applying Theorems 1 and 2 to $\sigma = (2, 4, 1, 6, 5, 7, 3)$ and using Figure 8, we have

$$F_\sigma = S_{4,2,1} + S_{4,3} + S_{5,1,1} + S_{5,2}.$$

Our proof of Theorem 1 will also provide us with a mechanism for defining $\Theta_\sigma$ for an arbitrary permutation $\sigma$. For any $w \in Red(\sigma)$, we will repeatedly apply this mechanism to $w$, traversing the L-S tree until we come to $w' \in Red(\sigma')$, where $\sigma'$ is a leaf. We would then simply set $\Theta_\sigma(w) = (\sigma', T(w'))$. So without further delay, let us describe this machinery.

4 A Bumping Process

To prove Theorem 1, we will demonstrate a bijection, $\theta$, between the sets

$$Red(\sigma) \text{ and } \bigcup_{\sigma' \sim \sigma} Red(\sigma')$$

such that for all $w \in Red(\sigma)$, $Des(w) = Des(\theta(w))$. The easiest way to describe the bijection is through an example, after which we will show exactly how and why it works. We begin our example with the reduced word

$$w = 3, 4, 1, 5, 4, 6, 2$$

which corresponds to the permutation $\sigma = (2, 4, 1, 6, 5, 7, 3)$ with $r = 6$, $s = 7$ and $I = \{1, 3\}$. For a given word $w = a_1 a_2 \cdots a_k$, we define

$$w(t) = a_1 a_2 \cdots a_{t-1} a_{t+1} \cdots a_k$$

and

$$w|_t = \begin{cases} a_1 a_2 \cdots a_{t-1} (a_t - 1) a_{t+1} \cdots a_k & \text{if } a_t > 1 \\ (a_1 + 1)(a_2 + 1) \cdots (a_{t-1} + 1) a_t (a_{t+1} + 1) \cdots (a_k + 1) & \text{if } a_t = 1 \end{cases}$$
Figure 9: Line diagram corresponding to the word $w_6$

where we will refer to $w^t_1$ as the word obtained from $w$ by bumping at time $t$. We also define the word obtained after a sequence of bumps by

$$w^t_{t_1,t_2,...,t_k} = (((w^t_{t_1})^t_{t_2})\cdots)^t_{t_k}$$

and we will refer to $(t_1,t_2,...,t_k)$ as a bumping sequence.

We begin the bumping process by locating the letter of $w$ which interchanges $\sigma_s$ and $\sigma_r$. There is at least one letter that does so since $r < s$ and $\sigma_s < \sigma_r$. However, there cannot be anymore since $w$ is reduced. From Figure 1, we see that the $6^{th}$ letter of $w$ interchanges $\sigma_s = 3$ and $\sigma_r = 7$. Using Figure 4, we can also identify 6 by looking at the bottom-most “□” in the right-most column with a “□”. This cell will invariably be labeled with the position in which $\sigma_s$ and $\sigma_r$ are interchanged in $LD(w)$.

The next step is to temporarily set

$$v = w^t_6$$

Notice that in $LD(v)$ as shown in Figure 9, lines 3 and 5 cross in positions 2 and 6. We can also indentify these numbers using the $LD(v)$, which is shown in Figure 5. This clearly indicates that at time 6, lines 3 and 5 cross. But we can easily see that these lines also cross at time 2. Therefore $v$ is not reduced. Since we have already bumped up at time 6, we continue the process by resetting $v$ to be

$$v = v^t_2$$

Again, $v$ is not reduced because lines 3 and 4 cross in positions 1 and 2 as seen in Figure 10. Since we just bumped up position 2, we reset $v$ to be

$$v = v^t_1$$

$LD(v)$ is now shown in Figure 11. We stop the bumping process here since $v$ is reduced and $\sigma_v \in \Phi(\sigma_w)$.

In summary, the bumping process starts by identifying the unique time, $t_0$, at which lines $\sigma_r$ and $\sigma_s$ cross in $LD(w)$ and letting $v = w^t_{t_0}$. The cross at time $t_0$ switches two lines
in \(LD(v)\). If these two lines cross again in \(LD(v)\), we will show that they cross at exactly one other time, \(t_1 \neq t_0\). This being the case, let \(v = v|_{t_1}\) and repeat until \(v\) is reduced. The bumping algorithm is formally defined in Algorithm 1.

We should point out here that in the description of Algorithm 1, we refer to the \((\sigma_s, r)\) entry of a labeled circle diagram. By this we mean the cell at the intersection of the \(\sigma_s^{th}\) row and the \(r^{th}\) column. We are not referring to the column with label \(r\). Later, we will comment on why this particular entry always contains the next number in the bumping sequence.

Before moving on, we should also point out that in the event \(I\) is empty, the bumping process will inevitably bump across in the first row into an empty row above it. If we create a new row on top and relabel the rows from top to bottom, it would have the same effect as applying the bumping process to \(1 \otimes \sigma\).

### 5 A More General Bumping Algorithm

It turns out that the bumping process described above can be performed for a variety of initial values of \(r\) and \(s\). To describe this more general algorithm, we will need a few more
Input: \( w \in \text{Red}(\sigma) \)

Algorithm body:
\[
\begin{align*}
    r &:= \max \{ i | \sigma_i > \sigma_{i+1} \} \\
    s &:= \max \{ i | r \sigma_i < \sigma_r \} \\
    t &:= (\sigma_s, r) \text{ entry of } \text{LCD}(w). \\
    v &:= w|_i \\
\end{align*}
\]

\textbf{while} (v not reduced)
\[
\begin{align*}
    t &:= (\sigma_s, r) \text{ entry of } \text{LCD}(v). \\
    v &:= v|_t \\
\end{align*}
\]
\textbf{end while}

Output: \( \theta(w) = v. \)

Algorithm 1: Bumping Algorithm

definitions. For a given permutation \( u \in S_n \) and \( 1 \leq r \leq n \), define the following
\[
\begin{align*}
    I(u, r) &= \{ i < r \mid l(u_i, r) = l(u) + 1 \} \\
    S(u, r) &= \{ s > r \mid l(u_{r, s}) = l(u) + 1 \}
\end{align*}
\]
and
\[
\begin{align*}
    \Phi(u, r) &= \begin{cases}
        \{ ut_{i, r} \mid i \in I(u, r) \} & \text{if } I(u, r) \neq \emptyset \\
        \Phi(1 \otimes u, r + 1) & \text{otherwise}
    \end{cases} \\
    \Psi(u, r) &= \begin{cases}
        \{ ut_{r, s} \mid s \in S(u, r) \} & \text{if } S(u, r) \neq \emptyset \\
        \Psi(u \otimes 1, r) & \text{otherwise}
    \end{cases}
\end{align*}
\]
where
\[
u \otimes 1 = \begin{pmatrix}
1 & 2 & \ldots & n & n + 1 \\
\mu_1 & \mu_2 & \ldots & \mu_n & n + 1
\end{pmatrix}.
\]
In other words, \( u \otimes 1 \) is the natural embedding of \( u \in S_n \) into \( S_{n+1} \). With these definitions in hand, we can state the following generalization of Theorem 1.

\textbf{Theorem 3} \ For any permutation \( u \in S_n \) and \( 1 \leq r \leq n \), we have
\[
\sum_{\sigma \in \Psi(u, r)} F_\sigma(X) = \sum_{\sigma \in \Phi(u, r)} F_\sigma(X).
\]

To verify that Theorem 3 does indeed generalize Theorem 1, it suffices to show that for any \( \sigma \), we must have
\[
\Phi(\sigma t_{r, s}, r) = \Phi(\sigma),
\]
and
\[
\Psi(\sigma t_{r, s}, r) = \{ \sigma \}
\]
for the values of \( r \) and \( s \) as defined in (3). We leave this as an exercise to the reader.
Input: \( w \in \text{Red}(\sigma) \) and \( r < s \) such that \( l(\sigma_{t,s}) = l(\sigma) - 1 \)

Algorithm body:
\[
t := (\sigma_s, r) \text{ entry of } \text{LD}(w).
\]
\[
v := w|_t
\]
\[
\text{while } (v \text{ not reduced})
\]
\[
t := (\sigma_s, r) \text{ entry of } \text{LD}(v).
\]
\[
v := v|_t
\]
end while

Output: \( \theta_{r}(w) = v. \)

Algorithm 2: General Bumping Algorithm

To prove Theorem 3, we define a more general bijection, \( \theta_r \), between
\[
\bigcup_{\sigma \in \Psi(u,v)} \text{Red}(\sigma) \quad \text{and} \quad \bigcup_{\sigma \in \Psi(u,v)} \text{Red}(\sigma)
\]
such that \( \text{Des}(w) = \text{Des}(\theta_r(w)) \). This generalized bumping process is formally defined in Algorithm 2.

We begin the process of showing that \( \theta_r \) is a bijection with the following lemma.

Lemma 4 Let \( w = a_1 a_2 \cdots a_k \) be a word corresponding to the permutation \( \sigma \) with \( k \geq l(\sigma) \). Pick \( 1 \leq t \leq k \) such that \( w^{(t)} \) is reduced and let \( v = w|_t \). Then \( v \) is reduced or there exists \( l \neq t \) such that \( v^{(l)} \) is reduced.

Before proving the above lemma, we point out that it applies to each step of Algorithm 2. Using the initial values of \( r \) and \( s \), it is clear that switching \( \sigma_r \) and \( \sigma_s \) in \( \sigma \) decreases the number of inversions by exactly one. Thus if lines \( \sigma_r \) and \( \sigma_s \) cross at time \( t \), \( w^{(t)} \) is reduced, justifying the first two lines of the algorithm. Letting \( v = w|_t \), the proof will show how to identify a particular value \( l \) such that \( v^{(l)} \) is reduced if \( v \) is not, thereby allowing us to apply the lemma to \( v \). The next step of the algorithm would be to reset \( v \) to be \( v|_l \) and repeat until \( v \) is reduced.

Proof. As in the statement of the lemma, let \( w = a_1 a_2 \cdots a_k \), pick \( t \) such that \( w^{(t)} \) is reduced and let \( v = w|_t \). Assume that lines \( \sigma_a \) and \( \sigma_b \) cross at time \( t \) in \( \text{LD}(w) \). Furthermore, define \( c \) such that lines \( \sigma_a \) and \( \sigma_c \) cross at time \( t \) in \( \text{LD}(v) \). We will prove the above lemma by considering four cases.

We begin with the event that \( c < b \) and \( \sigma_c < \sigma_a \). In this case, we must have that for all \( \sigma_i < \sigma_j < \sigma_a \), either \( i < c \) or \( i > a \). If not, there would be two lines that cross twice in \( \text{LD}(w^{(t)}) \) and thus \( w^{(t)} \) would not be reduced. Therefore switching \( \sigma_c \) and \( \sigma_a \) in \( \sigma_{t,a,b} \) would increase the number of inversions by exactly one, the new inversion being the pair \( \{\sigma_a, \sigma_c\} \). Since \( w^{(t)} \) is reduced, \( \sigma_{t,a,b} \) has exactly \( k-1 \) inversions and thus \( \sigma_c \) has exactly \( k \) inversions. Therefore \( v \) is reduced. To help visualize this case, the line diagrams corresponding to \( w \) and \( v \) are shown in Figure 12.

If \( c < b \) and \( \sigma_c > \sigma_a \) then \( v \) is not reduced. This is simply because lines \( \sigma_c \) and \( \sigma_a \) cross exactly once in \( \text{LD}(w) \), say at time \( l < t \). The bumping process causes these two lines to
Figure 12: Line diagram of $w$ (left) and $v$ (right) when $c < b$ and $\sigma_c < \sigma_a$

Figure 13: Line diagram of $w$ (left) and $v$ (right) when $c < b$ and $\sigma_c > \sigma_a$

cross a second time in $LD(v)$. But since $w^{(l)}$ and $v^{(l)}$ correspond to the same permutation, $v^{(l)}$ is reduced. The line diagrams corresponding to $w$ and $v$ in this case are shown in Figure 13.

The case when $c > b$ and $\sigma_c < \sigma_a$ is similar to the previous one except that $l > t$. The final case, $c > b$ and $\sigma_c > \sigma_a$, does not satisfy the condition that $w^{(l)}$ is reduced since the lines $\sigma_c$ and $\sigma_a$ would cross twice in $LD(w^{(l)})$. □

It would be wise to spend a moment here explaining how the value of $l$ identified in the above proof is located in Algorithm 2. In the case when $c < b$ and $\sigma_c > \sigma_a$, we identified $l$ as being the time at which lines $\sigma_c$ and $\sigma_a$ cross in $LD(w)$. We can also identify $l$ using only $LCD(v)$. By assumption, $t$ is at the intersection of the row labeled $\sigma_a$ and the column labeled $\sigma_b$ in $LCD(w)$. In $LCD(v)$, $t$ appears at the intersection of the row labeled $\sigma_c$ and the column labeled $\sigma_a$. Since lines $\sigma_a$ and $\sigma_c$ cross twice in $LD(v)$, $l$ must be the number at the intersection of the row labeled $\sigma_a$ and the column labeled $\sigma_c$ in $LCD(v)$. But since $\sigma_c$ is now where $\sigma_b$ was, $l$ can be found in $LCD(v)$ in the exact same position where $t$ was found in $LCD(w)$. Therefore, the next entry in the bumping sequence (if there is one) will always be located in the $(\sigma_a, r)$ entry of $LCD(v)$, as described in Algorithm 1.

Our next step is to show that this process preserves descent sets. To this end, we will show that no cross is bumped more than once. This given, any descent of $w$ must necessarily be a descent of $\theta_i(w)$. In fact, the only way a descent could be destroyed is if $a_i = a_{i+1} + 1$, where $w = a_1 a_2 \cdots a_k$. But if $w$ is bumped at time $i$, then $w^i_j$ would not be reduced and in fact, the very next step in the algorithm would be to bump $w^i_j$ at time $i + 1$, which
preserves the descent at $i$. For a similar reason, no new descents can be created.

This simple fact also explains why the bumping algorithm must terminate. Since if each cross may be bumped up at most once, the worst case scenario occurs when each cross is bumped up exactly once. But as soon as this happens, the resulting word would be reduced and the algorithm would stop. Therefore it is critical we have the following lemma.

**Lemma 5** Let $w \in Red(\sigma)$ and let $(t_1, t_2, t_3 \ldots)$ be the bumping sequence that arises from applying Algorithm 2 to $w$. Then for all $i \neq j$, $t_i \neq t_j$.

**Proof.** We begin by defining the boundary. Let $v_i = w|_{t_i, \ldots, t_i}$ for $i \geq 1$. Assume that lines $A$ and $B$ are interchanged in $LD(v_{i-1})$ at time $t_i$, as shown in Figure 14. We will refer to the first $t_i$ segments of line $A$ and the last $l(\sigma) - t_i$ segments of line $B$ in $LD(v_{i-1})$ as the boundary at time $i$, denoted $B_i$.

By definition, the only crosses that could possibly be bumped up at time $t_{i+1}$ must involve $B_i$. More specifically, if we let $C$ be the line directly above the cross at time $t_i$, then $v_i$ will not be reduced if line $C$ crosses $B_i$, as shown in Figure 14.

Also notice that $B_{i+1}$ lies weakly above $B_i$. In particular, the line segments between $t_i$ and $t_{i+1}$ that are on the boundary at time $i + 1$ lie strictly above those that are on the boundary at time $i$, while all other boundary segments remain the same. Therefore, once the boundary can only move up, once a cross is strictly below $B_i$, it cannot be bumped.

We are now ready to show that for any $i$, the cross at time $t_i$ cannot be bumped more than once. Using the previous lemma, it’s clear that $t_{i+1} \neq t_i$. Without lose of generality, let us assume that $t_{i+1} < t_i$. In order to bump up $t_i$ again we must bump up a cross into a portion of line $A$ that is strictly above $B_{i+1}$. We can see from Figure 14 that the only portion of line $A$ that meets this criterion is to the right of $t_{i+1}$, and since the boundary can only move up, this will be the case for all $B_j$ for all $j > i + 1$.

Thus if the cross at time $t_i$ is to be bumped again, there must be a minimum number $l > i$ such that $t_{l-1} < t_i$ and $t_l > t_i$. Notice that $B_{l-1}$ involves the cross in position $t_l$ but $B_l$ will be strictly above the cross in position $t_i$. Now that the cross at position $t_i$ is below the boundary, it cannot be bumped again. 

Our next step is to show that when Algorithm 2 is applied to $w \in Red(\sigma)$, the resulting reduced word corresponds to an element of $\Phi(\sigma t_{i,e}, \tau)$.
Lemma 6  Let $w = \text{Red}(\sigma)$ and for all $1 \leq j \leq k$, $v_j = w |_{t_1, t_2, \ldots, t_j}$ where $(t_1, t_2, \ldots, t_k)$ is the bumping sequence which results from applying Algorithm 2 to $w$. Then

$$\sigma_{v_j} = \sigma_{t_r, s} t_{r, ij}$$

where $\sigma_{ij}$ and $\sigma_s$ are interchanged at time $t_j$ in $LD(v_j)$. In particular, $\sigma_{v_k} \in \Phi(\sigma_{t_r,s}, r)$.

Proof. Since the first step of the algorithm is to bump $w$ at time $t_1$, we have

$$\sigma = \sigma_{w'} t_{r, s} \quad \text{and} \quad \sigma_{v_1} = \sigma_{v'} t_{r, i_1}$$

where $w' = w(t_1)$ and $v' = v_1(t_1)$. But since $w' = v'$, we have our result for $j = 1$.

Assume that the result holds for $1 \leq j < k$. Since $v_j$ is not reduced, $\sigma_{ij}$ and $\sigma_s$ are interchanged at times $t_j$ and $t_{j+1}$ in $LD(v_j)$. Therefore

$$\sigma_{v_{j+1}} = \sigma_{v_j} t_{r, ij} t_{r, i_{j+1}} = \sigma_{t_{r, s}, t_{r, ij}} t_{r, i_{j+1}} = \sigma_{t_{r, s}, t_{r, ij+1}}.$$

Notice that since $v_k$ is reduced, from the proof of Lemma 4 we have that $i_k < r$ and therefore $\sigma_{v_k} \in \Phi(\sigma_{t_{r,s}}, r)$. \hfill $\square$

Our final task is to show that $\theta_r$ is indeed a bijection as claimed.

Lemma 7  $\theta_r$ is a bijection between the two sets given in (10).

Notice that it suffices to show that $\theta_r$ is one-to-one. This is simply because the inverse of $\theta_r$ can be described by a bumping “down” process. More formally, if we define the complement of a word $w = a_1 a_2 \cdots a_l$ to be

$$w^c = (n - a_1)(n - a_2) \cdots (n - a_l),$$

where $w$ corresponds to an element of $S_n$, then the inverse map can be defined as

$$\theta_r^{-1}(w) = (\theta_n+1 \cdots \theta_l(w^c)^e). \quad (11)$$

Proof. Let $w_1 \in \text{Red}(\sigma_1)$ and $w_2 \in \text{Red}(\sigma_2)$ where $\sigma_1, \sigma_2 \in \Psi(u, r)$. Assume that $\theta_r(w_1) = \theta_r(w_2)$ and that both words correspond to the permutation $\sigma = ut_{i,r}$ for some $i \in I(u, r)$. This implies that the last numbers of the bumping sequences obtained from applying Algorithm 2 to $w_1$ and $w_2$ are the same, namely the $(\sigma, i)$ entry of $LD(\theta_r(w_1))$. In light of (11) and the fact that the bumping sequence resulting from Algorithm 2 is unique, we conclude that $w_1 = w_2$. \hfill $\square$

One immediate application of this bijection is to construct a correspondence between balanced tableaux of a given shape and standard tableaux of the same shape. Since labeled circle diagrams of a reduced word, $w$, are balanced, see [3], repeatedly applying Algorithm 1 until the resulting word $w'$ corresponds to a Grassmanian permutation will yield the standard tableaux $T(w')$. To see this, the reader is encouraged to experiment with our bijection using the Java Applet mentioned in the introduction. The author will examine this correspondence and its connection to the Edelman-Greene bijection [2] in an upcoming paper.
6 Acknowledgements

The author would like to thank Adriano Garsia for his tireless efforts in presenting the saga of reduced factorizations during the Winter of 2001. His inspiring review of the subject pointed the author in precisely the right direction to look for such a bijection.

References


David Little
Department of Mathematics
University of California, San Diego
La Jolla, CA 92093-0112
E-mail: dlittle@math.ucsd.edu