DIAGRAM CALCULUS FOR A TYPE AFFINE C TEMPERLEY–LIEB ALGEBRA, II

DANA C. ERNST AND TARYN M. LAIRD

Abstract. Abstract Stuff Here

1. Introduction

TO DO LAST

2. Preliminaries

2.1. **Coxeter groups.** A *Coxeter system* is pair (W, S) consisting of a distinguished (finite) set S of generating involutions and a group W, called a *Coxeter group*, with presentation

$$W = \langle S \mid (st)^{m(s,t)} = e \text{ for } m(s,t) < \infty \rangle,$$

where e is the identity, $m(s,t) = 1$ if and only if $s = t$, and $m(s,t) = m(t,s)$. It turns out that the elements of S are distinct as group elements, and that $m(s,t)$ is the order of st. Since the elements of S have order two, the relation $(st)^{m(s,t)} = e$ can be written as

$$\underbrace{sts\cdots}_{m(s,t)} = \underbrace{tst\cdots}_{m(s,t)}$$

with $m(s,t) \geq 2$ factors.

Given a Coxeter system (W, S), a word $s_{x_1} s_{x_2} \cdots s_{x_m}$ in the free monoid S^* is called an *expression* for $w \in W$ if it is equal to w when considered as a group element. If m is minimal among all expressions for w, the corresponding word is called a *reduced expression* for w. In this case, we define the *length* of w to be $\ell(w) = m$. Each element $w \in W$ can have several different reduced expressions that represent it. If we wish to emphasize a fixed, possibly reduced, expression for $w \in W$, we represent it in sans serif font, say $\mathsf{w} = s_{x_1} s_{x_2} \cdots s_{x_m}$, where each $s_{x_i} \in S$. A product $w_1 w_2 \cdots w_r$ with $w_i \in W$ is called *reduced* if $\ell(w_1 w_2 \cdots w_r) = \sum \ell(w_i)$. We may not need reduced product.

Matsumoto's Theorem [4, Theorem 1.2.2] states that if $w \in W$, then every reduced expression for w can be obtained from any other by applying a sequence of *braid moves* of the form

$$\underbrace{sts\cdots}_{m(s,t)} \mapsto \underbrace{tst\cdots}_{m(s,t)}$$

where $s, t \in S$ and each factor in the move has $m(s,t)$ letters.

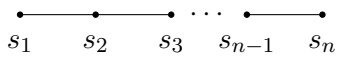
The sets $\mathcal{L}(w) = \{s \in S \mid \ell(sw) < \ell(w)\}$ and $\mathcal{R}(w) = \{s \in S \mid \ell(ws) < \ell(w)\}$ are called the *left* and *right descent sets* of w, respectively. It turns out that $s \in \mathcal{L}(w)$ (respectively, $\mathcal{R}(w)$) if and only if w has a reduced expression beginning (respectively, ending) with s.

Given a Coxeter system (W, S), the associated *Coxeter graph* is the graph Γ with vertex set S and edges $\{s, t\}$ labeled with $m(s,t)$ for all $m(s,t) \geq 3$. If $m(s,t) = 3$, it is customary to leave the corresponding edge unlabeled. Given a Coxeter graph Γ, we can uniquely reconstruct the

Date: May 7, 2015.
2000 *Mathematics Subject Classification.* 20F55, 20C08, 57M15.
Key words and phrases. diagram algebra, Temperley–Lieb algebra, Coxeter groups, heaps.

corresponding Coxeter system (W, S). In this case, we say that (W, S), or just W, is of type Γ. If (W, S) is of type Γ, for emphasis, we may write (W, S) as $(W(\Gamma), S(\Gamma))$. Note that generators s and t are connected by an edge in the Coxeter graph Γ if and only if s and s do not commute [6].

The main focus of this paper will be the Coxeter systems of types insert the types we discuss A_n which are defined by the Coxeter graphs in Figures 1 and **??**, respectively, where $n \geq 2$.



(a) Type A_n

FIGURE 1. Coxeter graphs corresponding to Coxeter systems of type A_n.

2.2. **Fully commutative elements.** Let (W, S) be a Coxeter system of type Γ and let $w \in W$. Following Stembridge [8], we define a relation $\sim$ on the set of reduced expressions for w. Let w and w' be two reduced expressions for w. We define $\mathsf{w} \sim \mathsf{w}'$ if we can obtain w' from w by applying a single commutation move of the form $st \mapsto ts$, where $m(s, t) = 2$. Now, define the equivalence relation $\approx$ by taking the reflexive transitive closure of $\sim$. Each equivalence class under $\approx$ is called a *commutation class*. If w has a single commutation class, then we say that w is *fully commutative* (FC). According to a result in [8], an element w is FC if and only if no reduced expression for w contains a subword of the form $sts\cdots$ of length $m(s, t) \geq 3$. The set of FC elements of the Coxeter system (W, S) of type Γ is denoted by $\mathrm{FC}(\Gamma)$, or possibly $\mathrm{FC}(W)$.

Remark 2.2.1. The elements of $\mathrm{FC}(\widetilde{C}_n)$ are precisely those whose reduced expressions avoid consecutive subwords of the following types:

(1) $s_i s_j s_i$ for $|i - j| = 1$ and $1 < i, j < n + 1$;
(2) $s_i s_j s_i s_j$ for $\{i, j\} = \{1, 2\}$ or $\{n, n + 1\}$.

Note that the FC elements of $W(B_n)$ and $W(B'_n)$ avoid the respective subwords above.

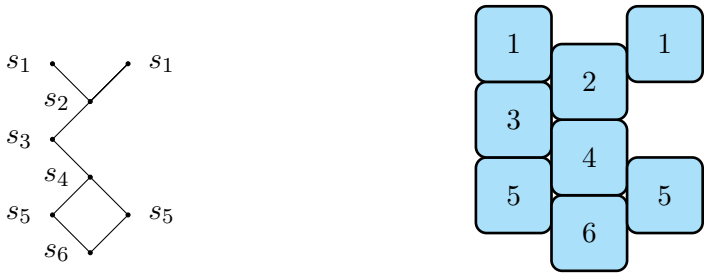
In [8], Stembridge classified the Coxeter groups that contain a finite number of FC elements. According to [8, Theorem 5.1], $W(\widetilde{C}_n)$ contains an infinite number of FC elements, while $W(B_n)$ (and hence $W(B'_n)$) contains finitely many. There are examples of infinite Coxeter groups that contain a finite number of FC elements (e.g., $W(E_n)$ is infinite for $n \geq 9$, but contains only finitely many FC elements [8, Theorem 5.1]).

2.3. **Heaps.** Every reduced expression can be associated with a partially ordered set called a heap that will allow us to visualize a reduced expression while preserving the essential information about the relations among the generators. The theory of heaps was introduced in [10] by Viennot and visually capture the combinatorial structure of the Cartier–Foata monoid of [2]. In [8] and [9], Stembridge studied heaps in the context of fully commutative elements, which is our motivation here. In this section, we mimic the development found in [1], [3], and [8].

Let (W, S) be a Coxeter system. Suppose $\mathsf{w} = s_{x_1} \cdots s_{x_r}$ is a fixed reduced expression for $w \in W$. As in [8], we define a partial ordering on the indices $\{1, \ldots, r\}$ by the transitive closure of the relation $\lessdot$ defined via $j \lessdot i$ if $i < j$ and s_{x_i} and s_{x_j} do not commute. In particular, since w is reduced, $j \lessdot i$ if $i < j$ and $s_{x_i} = s_{x_j}$ by transitivity. This partial order is referred to as the *heap* of w, where i is labeled by s_{x_i} Note that for simplicity, we are omitting the labels of the underlying poset but retaining the labels of the corresponding generators.

It follows from [8, Proposition 2.2] that heaps are well-defined up to commutativity class. That is, if w and w' are two reduced expressions for $w \in W$ that are in the same commutativity class, then the heaps of w and w' are equal. In particular, if w is FC, then it has a single commutativity class, and so there is a unique heap associated to w.

Example 2.3.1. Let $\mathsf{w} = s_3s_5s_2s_4s_6s_1s_5s_2$ be a reduced expression for $w \in (\mathcal{A}_6)$. We see that w is indexed by $\{1, 2, 3, 4, 5\}$. As an example, $3 \lessdot 2$ since $2 < 3$ and the second and third generators do not commute. The labeled Hasse diagram for the heap poset of w is shown in Figure 2(a).



(a) (b)

FIGURE 2. The labeled Hasse diagram and a possible lattice point representation for the heap of an element from $(\widetilde{C}_5)$.

Let w be a fixed reduced expression for $w \in W(\mathcal{A}_n)$. As in [1] and [3], we will represent a heap for w as a set of lattice points embedded in $\{1, 2, \ldots, n+1\} \times \mathbb{N}$. To do so, we assign coordinates (not unique) $(x, y) \in \{1, 2, \ldots, n+1\} \times \mathbb{N}$ to each entry of the labeled Hasse diagram for the heap of w in such a way that:

(1) an entry with coordinates (x, y) is labeled s_i in the heap if and only if $x = i$;
(2) an entry with coordinates (x, y) is greater than an entry with coordinates (x', y') in the heap if and only if $y > y'$.

Recall that a finite poset is determined by its covering relations. In the case of a Coxeter group of type $\mathcal{A}_n$ (and any straight line Coxeter graph), it follows from the definition that (x, y) covers (x', y') in the heap if and only if $x = x' \pm 1$, $y > y'$, and there are no entries (x'', y'') such that $x'' \in \{x, x'\}$ and $y' < y'' < y$. This implies that we can completely reconstruct the edges of the Hasse diagram and the corresponding heap poset from a lattice point representation. The lattice point representation of a heap allows us to visualize potentially cumbersome arguments. In this case our heaps correspond to a left version of [1] and several other papers. That is, in this paper entries at the left of a heap correspond to generators occurring to the rights, as opposed to the left, in the corresponding reduced expression.

Let w be a reduced expression for $w \in W(\mathcal{A}_n)$. We let $H(\mathsf{w})$ denote a lattice representation of the heap poset in $\{1, 2, \ldots, n+1\} \times \mathbb{N}$ described in the preceding paragraphs. If w is FC, then the choice of reduced expression for w is irrelevant, in which case, we will often write $H(w)$ (note the absence of sans serif font) and we will refer to $H(w)$ as the heap of w.

Given a heap, there are many possible coordinate assignments, yet the x-coordinates for each entry will be fixed for all of them. In particular, two entries labeled by the same generator may only differ by the amount of horizontal space between them while maintaining their relative vertical position to adjacent entries in the heap.

Let $\mathsf{w} = s_{x_1} \cdots s_{x_r}$ be a reduced expression for $w \in (\mathcal{A}_n)$. If s_{x_i} and s_{x_j} are adjacent generators in the Coxeter graph with $i < j$, then we must place the point labeled by s_{x_i} in the column that is *left* of the point labeled by s_{x_j}. Because generators that are not adjacent in the Coxeter graph do commute, points whose x-coordinates differ by more than one can slide past each other or land in the same column. To emphasize the covering relations of the lattice representation we will enclose each entry of the heap in a rectangle in such a way that if one entry covers another, the rectangles overlap halfway.

Example 2.3.2. Let w be as in Example 2.3.1. Then one possible representation for $H(w)$ appears in Figure 2(b).

When w is FC, we wish to make a canonical choice for the representation $H(w)$ by assembling the entries in a particular way. To do this, we give all entries corresponding to elements in $\mathcal{L}(w)$ the leftmost column and all other entries in the heap should have horizontal position as far to the left as possible. For example, the representation of $H(w)$ given in Figure 2(b) is the canonical representation. Note that our canonical representation of heaps of FC elements corresponds precisely to the unique heap factorization of [10, Lemma 2.9] and to the Cartier–Foata normal form for monomials [2, 5]. When illustrating heaps, we will adhere to this canonical choice, and when we consider the heaps of arbitrary reduced expressions, we will only allude to the relative columns of the entries, and never their absolute coordinates.

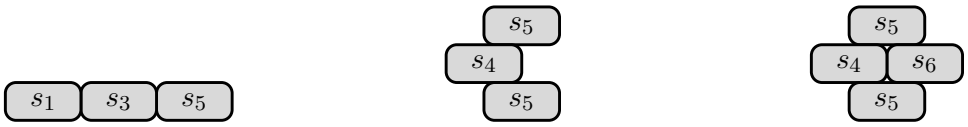
Let $w \in \mathrm{FC}(\mathcal{A}_n)$ have reduced expression $\mathsf{w} = s_{x_1} \cdots s_{x_r}$ and suppose s_{x_i} and s_{x_j} equal the same generator s_k, so that the corresponding entries have x-coordinate k in $H(w)$. We say that s_{x_i} and s_{x_j} are *consecutive* if there is no other occurrence of s_k occurring between them in w.

Let $\mathsf{w} = s_{x_1} \cdots s_{x_r}$ be a reduced expression for $w \in W(\mathcal{A}_n)$. We define a heap H' to be a *subheap* of $H(\mathsf{w})$ if $H' = H(\mathsf{w}')$, where $\mathsf{w}' = s_{y_1} s_{y_2} \cdots s_{y_k}$ is a subexpression of w. We emphasize that the subexpression need not be a subword (i.e., a consecutive subexpression).

A subheap H' of $H(\mathsf{w})$ is called a *saturated subheap* if whenever s_i and s_j occur in H' such that there exists a saturated chain from i to j in the underlying poset for $H(\mathsf{w})$, there also exists a saturated chain $i = i_{k_1} < i_{k_2} < \cdots < i_{k_l} = j$ in the underlying poset for H' such that the same chain is also a saturated chain in the underlying poset for $H(\mathsf{w})$.

Recall that a subposet Q of P is called convex if $y \in Q$ whenever $x < y < z$ in P and $x, z \in Q$. A *convex subheap* is a subheap in which the underlying subposet is convex.

Example 2.3.3. Let $\mathsf{w} = s_2 s_1 s_3 s_2$ as in Example 2.3.1. Also, let $\mathsf{w}' = s_1 s_3$ be the subexpression of w that results from deleting the first and last generators of w. Then $H(\mathsf{w}')$ is equal to the heap in Figure 3(a) and is a subheap of $H(\mathsf{w})$. However, $H(\mathsf{w}')$ is neither a saturated nor a convex subheap of $H(\mathsf{w})$ since there is a saturated chain in $H(\mathsf{w})$ from the lower occurrence of s_5 to the occurrence of s_3, but there is not a chain between the corresponding entries in $H(\mathsf{w}')$. Now, let $\mathsf{w}'' = s_2 s_3 s_2$ be the subexpression of w that results from deleting all but the second generator of w. Then $H(\mathsf{w}'')$ equals the heap in Figure 3(b) and is a saturated subheap of $H(\mathsf{w})$, but it is not a convex subheap since there is an entry in $H(\mathsf{w})$ labeled by s_1 occurring between the two consecutive occurrences of s_2 that does not occur in $H(\mathsf{w}'')$. However, if we do include the entry labeled by s_1, then the heap in Figure 3(c) is a convex subheap of $H(\mathsf{w})$.



(a) (b) (c)

FIGURE 3. Various subheaps of the heap in given in Figure 2(b).

From this point on, if there can be no confusion, we will not specify the exact subexpression that a subheap arises from.

The following fact is implicit in the literature (in particular, see the proof of [8, Proposition 3.3]) and follows easily from the definitions.

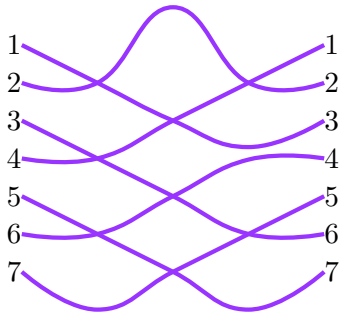
3. TYPE $\mathcal{A}$ STUFF

3.1. **Connection to Symmetric Group.** Some introductory sentence. Type $\mathcal{A}_n$ coxeter groups are isomorphic to the symmetric group on $n+1$ elements, S_{n+1}. We will not prove the isomorphism

here but will give the relation between the two groups. It is known that all elements of the symmetric group can be written as adjacent 2-cycles. Given $s_i \in \mathcal{A}_n$ the corresponding element in S_{n+1} is $(i, i+1)$ an adjacent 2-cycle. From this we see that $\mathcal{A}_n \cong S_{n+1}$ since $s_n = (n, n+1)$. Since this isomorphism exists we will use both Type A_n and S_{n+1} in the following sections for Type A_n.

3.2. **String Diagrams.** Given the relation between Type A_n and the symmetric group every reduced expression can be expressed as a string diagram that will allow us to visualize a reduced expression. It is important to note that the string diagram is not the string graph of $\mathcal{A}_n$ that is seen in Figure 1. Let (W, S) be a Coxeter system of Type A_n. Suppose $\mathsf{w} = s_{x_1} \dots s_{x_r}$ is a fixed reduced expression for $w \in W$. We see that w corresponds to the following element in S_{n+1}, $\sigma = (x_1, x_{x+1})(x_2, x_{x+2}) \dots (x_r, x_{r+1})$, where (x_i, x_{i+1}) is the adjacent transposition defined above. We construct a string diagram as follows, create two columns of $n+1$ nodes that align node 1 with node 1 as the top most nodes, node 2 with node 2 as the second nodes in each column and so on ending with node $n+1$ aligned with node $n+1$ as the bottom most nodes. Now starting with node 1 on the right utilize σ to find where 1 ends up, say j and draw a line connection node 1 on the right to node j on the left. Repeat for nodes 2 through $n+1$.
A *crossing* in a string diagram refers to the location in which two strings cross. In Type A_n two distinct strings will cross a maximum of one time.

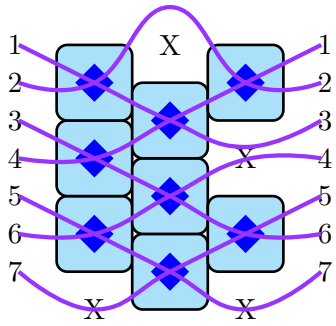
Example 3.2.1. Let $\mathsf{w} = s_3 s_5 s_2 s_4 s_6 s_1 s_5 s_2$ be a reduced expression for $w \in W(\mathcal{A}_6)$. We see that w corresponds to the following element in S_7, $\mathsf{w} = (34)(56)(23)(45)(67)(12)(56)(23)$. The labeled string diagram for w is shown in Figure **??**.



In this case our string diagrams correspond to a left version of [7] and several other papers. That is, in this paper the strings are moved from the right to the left as opposed to the top to the bottom in the corresponding reduced expression.

3.3. **Connection between String Diagrams and Heaps.** It is well known folklore that heaps can be derived from string diagrams and string diagrams can be derived from heaps. We will first show how to construct a string diagram from a heap. Fix a reduced expression $\mathsf{w} = s_i \dots s_r$ and construct the corresponding heap. For a heap block corresponding to a generator s_n and label the node corresponding to the top of the block with n and the node at the bottom of the block with $n+1$. In the center of the block create a square node which indicates a string crossing, wherever a block is missing place an x indicating. Repeat this for each block in the heap. Starting in the top most right node labeled with a 1 and trace the string to the closest square node or x, if the string hits a node it will pass through to the next node or x, if the string hits an x it will bounce in the opposite direction the string was traveling going towards the next node or x in the next column.

Example 3.3.1. Let $\mathsf{w} = s_1 s_2 s_3 s_5$ be a a reduced expression for $w \in W(A_5)$. The heap is as follows from Example 2.3.1. Here we have labeled the top and bottom of the heaps appropriately, put the square nodes in the center of the heap blocks to nodes, put an X in the center of missing heap blocks, and overlaid the strings as well.



This can be done to each heap to get the corresponding string diagram. This process can also be done to construct a heap from a string diagram. This is beyond the scope of this paper and will not be discussed here.

4. T-Avoiding in Type A

4.1. **Property T.** Given the connection to they symmetric groups of Type $(\mathcal{A}_n)$ we will consider the following definitions in terms of permutations in S_{n+1}. A permutation $\sigma \in S_{n+1}$ has *Property T* if and only if there exists an i such that either $\sigma(i) > \sigma(i+1), \sigma(i+2)$ or $\sigma(i+2) < \sigma(i), \sigma(i+1)$ Visually we see this in a string diagram as follows. Insert property T pictures for Type A here It is important to note that σ containing Property T is not exclusive itself, if σ^{-1} contains the same properties, then the element is said to have Property T.

Example 4.1.1. Consider $\sigma = (124)(35)$ in S_{n+1}. The string diagram for σ is as follows. Insert String Diagram here with Property T marked off on the sides. Note that σ has Property T twice and σ^{-1} has Property T once.

4.2. **T-Avoiding and Bad Elements.** The focus of our research has been to find elements that don't contain Property T. An element $\sigma \in S_{n+1}$ is said to be *T-avoiding* if and only if neither σ nor σ^{-1} have property T. It is clear that if σ is a product of commuting adjacent 2 cycles then σ is T-avoiding. When σ is an element of S_n of this type we say that σ is *trivially T-avoiding*. An element $\sigma \in S_{n+1}$ that is T-avoiding by not trivially T-avoiding is called *bad*.

4.3. **T-Avoiding in Type A_n.** We will show that there are no bad elements in Type A_n. To do this we will show that there are no bad elements in S_n since they are isomorphic.

Proposition 4.3.1. *Elements in S_n are T-Avoiding if and only if the elements are products of commuting adjacent 2 cycles.*

Proof. ($\Leftarrow$) Let σ be a product of commuting adjacent 2 cycles. Then σ is trivially T-avoiding.
($\Rightarrow$) To show this direction we will show the contrapositive is true. That is *if σ is not the product of commuting adjacent 2-cycles then either σ or σ^{-1} has Property T.* Assume that σ is not a product of commuting adjacent 2-cycles. This implies that any heap for σ has at least 2 columns and that the string diagram has at least one string with more than one crossing. We will proceed in two cases which we will call the easy case and the hard case.
Easy case There is at least one node in the second column (left or right hand side) is blocked by only one other node. Without loss of generality suppose the $i+1^{st}$ node is blocked in the i^{th} position but not in the $i+2^{nd}$ position. Then $\sigma(i) \geq i+1$, but $\sigma(i+1) \leq i$ and $\sigma(i+2) \leq i$. picture of the case. Thus σ has Property T.
Hard case Every node in the second column (left or right hand side) is blocked by 2 nodes in the first column, however somewhere in the heap there exists nodes in the $i-1^{st}$ column in the $\imath^{th}$ position, in the i^{th} column in the $i-1^{st}$ or in the $i+1^{st}$ position, and in the $i+1^{st}$ column in the

i^{th} position. picture of the pattern we are looking for. Without loss of generality we will look at this case from the right hand side which implies that our first column referred to later is the right most column and continuing numbering in the left direction columns. First, we will show that if we are in the hard case then σ is not fully commutative. For the sake of contradiction suppose σ is fully commutative and minimal, then there is only one heap corresponding to σ which has 3 or more columns. Because we are in the hard case, every node in the second column is blocked by two nodes in the first column. Delete the first column. We should be left with an element that is FC. However, since we chose a minimal case deleting the first column creates an element in the easy case. Because of this we see that σ can not be FC. This implies that σ contains the diamond shape we are looking for. Since we now know that the diamond shape exists in σ it is easy to see that we have one of the following pictures occur within the string diagram of σ. Picture of the cases. Thus, σ has Property T.
Therefore, if σ is not the product of commuting adjacent 2-cycles, then either σ or σ^{-1} has Property T and elements in S_n are T-Avoiding if and only if the elements are products of commuting 2-cycles. □

5. T-Avoiding in Type B

5.1. **Property T.** A slightly different definition is required for Property T for Type B_n

5.2. **T-Avoiding in Type B_n.** In the following section we will show that there are no bad elements in type B_n. In order to do this we first will prove several lemmas, and then we will prove the theorem. In the following, unless otherwise stated w is presented in the diagonal factorization.

Lemma 5.2.1. *If $w_n = s_n$, $w_{n-1} = s_{n-1}s_{n-2}$ and $w_i = e$ for all $i \neq n, n-1$, then w has Property T on the left.*

Proof. Suppose $w_n = s_n$, $w_{n-1} = s_{n-1}s_{n-2}$ and $w_i = e$. Then $w = s_{n-1}s_{n-2}s_n$. Clearly w has Property T on the left. □

Lemma 5.2.2. *If everything in the second column (on the right) is blocked by two things in the rightmost column in the heap of w and all the zigs attain maximum height, then w has Property T on the left. Here maximum height is defined to be the block in the rightmost column, or s_0 if it does not reach the right most column.*

Proof. Suppose everything in the second column is blocked by two things in the rightmost column in the heap of w, and all zigs attain maximum height. Then $w = s_0s_1s_0s_2s_1s_0w_3 \dots w_n$, and clearly w has Property T on the left. □

Lemma 5.2.3. *If everything in the second column is blocked by two things in the rightmost column, and not all w_k attain maximum height as defined above,then w has Property T on the right.*

Proof. Suppose that everything in the second column of the heap of w is blocked by two things in the rightmost column of w. Choose the maximal k such that w_k does not attain maximum height. We proceed in the following cases.

- Case 1: The w_k that does not attain maximum height is one which would contribute to the rightmost column.
  Suppose that the maximal k such that w_k does not attain maximal height is one which would contribute to the rightmost column of the heap. Then we can write $w_k = s_ks_{k-1} \dots s_i$ and we can write $w = w_0 \dots w_{k-1}s_ks_{k-1} \dots s_is_{k+1}s_k \dots s_jw_{k+1} \dots w_n$. Note that s_j is in the right most column.
  If $s_i = s_j$, then applying commutations to w we see that $w = w_0 \dots w_{k-1}s_ks_{k-1} \dots s_{i-1}s_{k+1}s_k \dots s_is_{j+1}s_jw_k$
  Applying a braid to the $s_is_{j+1}s_j$ elements we get that $w = w_0 \dots w_{k-1}s_ks_{k-1} \dots s_{i-1}s_{k+1}s_k \dots s_{j+1}s_js_{j+1}w_k$

In applying the braid move to w we have created a gap in the rightmost wall of the heap. Because of this gap we can apply commutations to w and $w = w_0 \dots w_{k-1} s_k s_{k-1} \dots s_{i-1} s_{k+1} s_k \dots s_{j+1} w_{k+1} \dots$ and thus w has Property T on the right.
Now consider $s_i \neq s_j$. Since w_{k+1} attains maximum height $s_i \in \mathrm{Sup}(w_{k+1})$. Then by applying commutations to w above we see that $w = w_0 \dots w_{k-1} s_k s_{k-1} \dots s_{i-1} s_{k+1} s_k \dots s_i s_{i+1} s_i \dots s_j w_{k+1} \dots w_n$.
Using the braid moves described at the end of the Type $\mathcal{A}$ proof, which is legal since except for s_0, s_1 the same relations hold, we can using braid moves the $s_i s_{i+1} s_i$ to the rightmost wall.
In doing this we have created a gap in the rightmost column and by applying the same commutations from above we see that $w = w_0 \dots w_{k-1} s_k s_{k-1} \dots s_{i-1} s_{k+1} s_k \dots s_{j+1} w_{k+1} \dots w_n s_{i+1} s_i$,
which has Property T on the right.

- The w_k that does not attain maximum height is not one which would contribute to the right most column.
  Suppose that the maximal k such that w_k does not attain maximal height is not one which would contribute to the rightmost column. Since w_k does not attain maximal height we can write $w_k = s_k \dots s_i$ where $s_i \neq 0$. However, since w_{k+1} attains maximal height $s_i \in \mathrm{Sup}(w_{k+1})$. Then we can write $w = w_0 \dots w_{k-1} s_k s_{k-1} \dots s_i s_{k+1} s_k \dots s_i \dots s_j w_{k+1} \dots w_n$.
  By applying commutations we see that $w = w_0 \dots w_{k-1} s_k s_{k-1} \dots s_{i-1} s_{k+1} s_k \dots s_i s_{i+1} s_i \dots s_j w_{k+1} \dots w_n$.
  Then as above we can apply braid moves to $s_i s_{i+1} s_i$ to create a gap in the rightmost column and applying the commutations as above we get $w = w_0 \dots w_{k-1} s_k s_{k-1} \dots s_{i-1} s_{k+1} s_k \dots s_{j+1} w_{k+1} \dots w_n s_{i+$
  Thus w has Property T on the right.

Therefore, if everything in the second column is blocked by two things in the rightmost column, and not all w_k attain maximum height as defined above,then w has Property T on the right. □

The above lemmas when combined together show that if the elements in the second column of the heap of w are blocked by two things in the first column then w has Property T. We will use this fact in the proof the following proof.

Theorem 5.2.4. *In Type B_n, elements, $w \in B_n$, are T-Avoiding if and only if w is a product of commuting generators.*

Proof. Let $w \in B_n$ be a reduced expression with diagonal factorization.
($\Leftarrow$) Suppose w is a product of commuting generators. Then w is trivially $T - Avoiding$.
($\Rightarrow$) To prove this direction we will show the contrapositive. That is, if w is not a product of commuting generators then w has Property T on the left or right.
Suppose w is not a product of commuting generators. If s_0 is not in the support of w, then w is a Type $\mathcal{A}_{n-1}$ element and we are done. Consider $s_0 \in \mathrm{Sup}(w)$. Since w is not a product of commuting generators there exists a maximal k such that $w_k \neq \{e, s_k\}$. That is, $w_k = s_k \dots s_i$ for $i \in \{1, \dots, k-1\}$. The rest proceeds in case analysis.

(1) Them minimal $k = n$
  (a) Suppose $w_n = s_n s_{n-1} \dots s_i$ for $i \in \{0, \dots, n-1\}$ and w_n does not zig-zag. Then $w = w_0 \dots w_{k-1} s_n s_{n-1} \dots s_{i+1} s_i$, which implies that w has Property T on the right.
  (b) Suppose $w_n = s_n s_{n-1} \dots s_i$ for $i \in \{0, \dots, n\}$ and w_n does zig-zag. Then $w = w_0 \dots w_{n-1} s_n s_{n-1} \dots s_1 s_0 s_1 \dots s_i$, which implies w has Property T on the right.
(2) The minimal $k < n$.
  (a) The element $w_{k+1} = e$.
    (i) Suppose $w_k = s_k \dots s_i$ for $i \in \{1, \dots k-1\}$ and w does not zig zag. Then $w = w_0 w_1 \dots w_{k-1} s_k \dots s_i e w_{k+2} \dots w_n$ which when commutations are applied to w we see that $w = w = w_0 w_1 \dots w_{k-1} e w_{k+2} \dots w_n s_k \dots s_i$. This shows that w has Property T on the right.
    (ii) Suppose $w_k = s_k \dots s_i$ for $s_i \in \{0, k\}$. and w zig zags. then $w = w_0 \dots w_k - 1 s_k s_{k-1} \dots s_i e w_k + 2 \dots w_n$ which applying commutations, a possibility since for

$i \geq k+1$, w_i is either e or s_i, gives $w = w_0 \dots w_{k-1} s_k \dots s_{i-2} w_{k+1} \dots w_n s_{i-1} s_i$ which clearly has Property on the right.

(b) The element $w_{k+1} \neq e$.

(i) Suppose $w_k = s_k \dots s_i$ for $i \in \{0, \dots k\}$ and w zig-zags, and write $w = w_0 \dots w_{k-1} s_k \dots s_i w_{k+1} \dots$

(A) Suppose $i = k$. Then we can write $w = w_0 \dots w_{k-1} s_k \dots s_1 s_0 s_1 \dots s_k w_{k+1} \dots w_n$.

- Suppose $w_{k+2} = e$.Then $w = w_0 w_{k-1} s_k \dots s_1 s_0 s_1 \dots s_k s_{k+1} e w_{k+3} \dots w_n$ which after applying commutations $w = w_0 w_{k-1} s_k \dots s_1 s_0 s_1 \dots s_k s_{k+1} e w_{k+3} \dots w_n s_k s_{k+1}$ Thus w has Property T on the right.
- Suppose $w_{k+2} \neq e$. Then $w = w_0 \dots w_{k-1} s_k \dots s_1 s_0 s_1 \dots s_k s_{k+1} s_{k+2} w_{k+3} \dots w_n$. If $w_{k+3} = e$ we are done as we can commute $s_{k+1} s_{k+2}$, which implies w has Property T on the right. If $w_{k+3} \neq e$ and $w_{k+4} = e$, then we are done as we can commute $s_{k+2} s_{k+3}$ out to the right. Inductively, choose w_{k+i} and w_{k+1+i} such that neither is the identity and w_{k+i+2} is the identity, and commute them out making w have Property T on the right.

(B) Suppose $i \neq k$. Then $w = w_0 \dots w_{k-1} s_k \dots s_1 s_0 s_1 \dots s_{i+2} w_{k+1} \dots w_n$. By applying commutations we see that $w = w_0 \dots w_{k-1} s_k \dots s_1 s_0 s_1 \dots s_{i-2} w_{k+1} \dots w_n s_{i+1} s_1$. Thus w contains property Property T.

(ii) Suppose $w_k = s_k \dots s_i$ for $i \in \{0, \dots, k-1\}$ and w does not have zig zags.

- Suppose $i \neq k-1$. Then $w = w_0 \dots w_{k-1} s_k \dots s_i w_{k+1} \dots w_n$. By apply commutations we see that $w = w_0 \dots s_k \dots s_{i+2} w_{k+1} \dots w_n s_{i+1} s_i$. This implies that w has Property T on the right.
- Suppose $s_i = s_{k-1}$. Then by the combinations of the lemmas we are done.

Therefore in Type B_n elements are T-Avoiding if and only if they are a product of commuting generators. □

References

[1] S.C. Billey and B.C. Jones. Embedded factor patterns for Deodhar elements in Kazhdan–Lusztig theory. *Ann. Comb.*, 11(3-4):285–333, 2007.

[2] P. Cartier and D. Foata. Problèmes combinatoires de commutation et réarrangements. *Lect. Notes Math. Springer-Verlag, New York/Berlin*, 85, 1969.

[3] D.C. Ernst. Non-cancellable elements in type affine C Coxeter groups. *Int. Electron. J. Algebr.*, 8:191–218, 2010.

[4] M. Geck and G. Pfeiffer. *Characters of finite Coxeter groups and Iwahori–Hecke algebras*. 2000.

[5] R.M. Green. Star reducible Coxeter groups. *Glas. Math. J.*, 48:583–609, 2006.

[6] J.E. Humphreys. *Reflection Groups and Coxeter Groups*. 1990.

[7] B.C. Jones. Leading coefficients of Kazhdan—Lusztig polynomials for Deodhar elements. *J. Algebr. Comb.*, 29(2):229–260, 2009.

[8] J.R. Stembridge. On the fully commutative elements of Coxeter groups. *J. Algebr. Comb.*, 5:353–385, 1996.

[9] J.R. Stembridge. The enumeration of fully commutative elements of Coxeter groups. *J. Algebr. Comb.*, 1998.

[10] G. Viennot. Heaps of pieces, I: Basic definitions and combinatorial lemmas. In *Comb. énumérative*, Lecture Notes in Mathematics, pages 321–350. Springer Berlin Heidelberg, 1986.

Department of Mathematics and Statistics, Northern Arizona University, Flagstaff, AZ 86011
E-mail address: dana.ernst@nau.edu
URL: http://danaernst.com

Department of Mathematics and Statistics, Northern Arizona University, Flagstaff, AZ 86011
E-mail address: tml94@nau.edu