THE COMB POSET AND THE PARSEWORDS FUNCTION

RIK SENGUPTA AND WARUT SUKSOMPONG

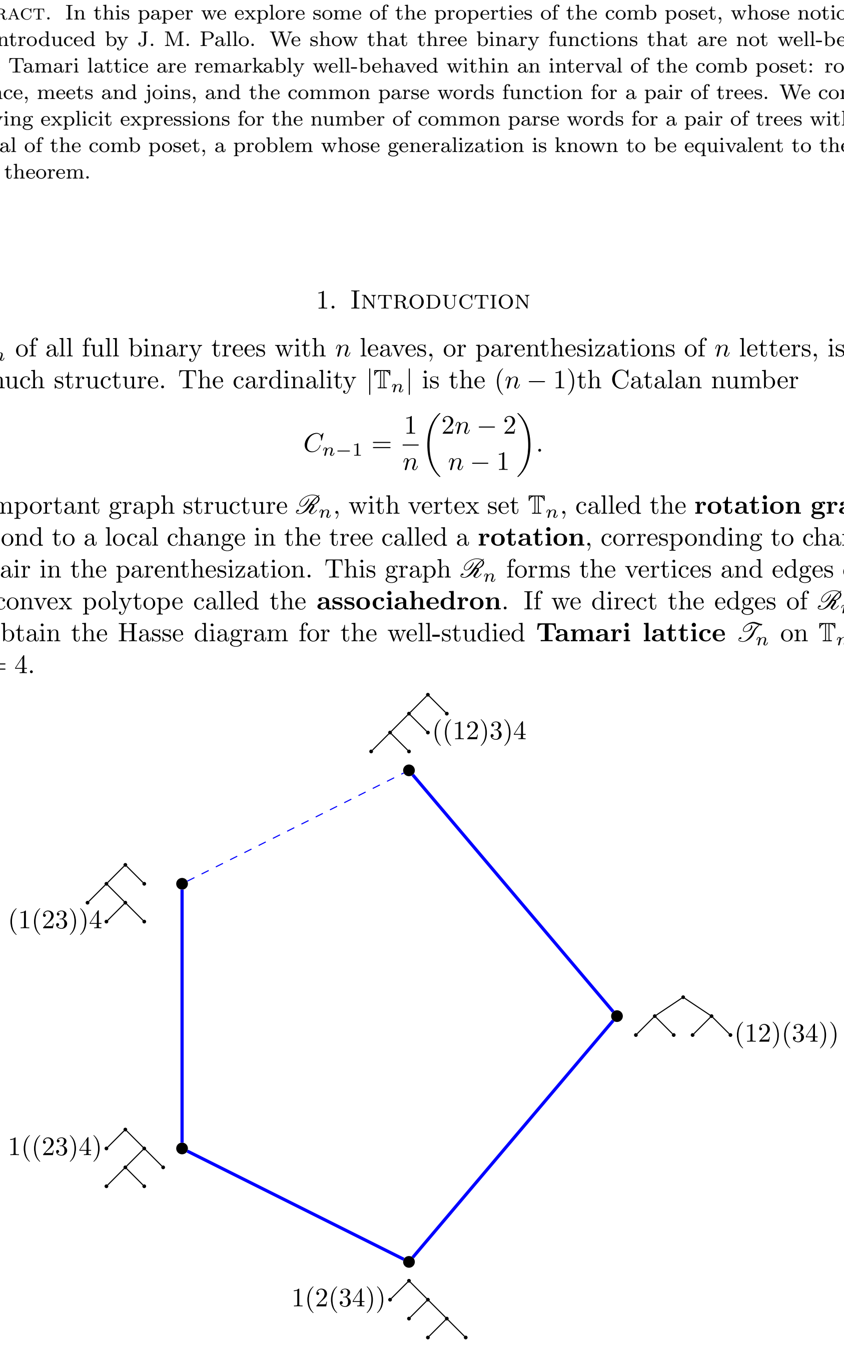
Abstract. In this paper we explore some of the properties of the comb poset, whose notion was first introduced by J. M. Pallo. We show that three binary functions that are not well-behaved in the Tamari lattice are remarkably well-behaved within an interval of the comb poset: rotation distance, meets and joins, and the common parse words function for a pair of trees. We conclude by giving explicit expressions for the number of common parse words for a pair of trees within an interval of the comb poset, a problem whose generalization is known to be equivalent to the Four Color theorem.

1. Introduction

The set $\mathbb{T}_n$ of all full binary trees with n leaves, or parenthesizations of n letters, is well-studied, and carries much structure. The cardinality $|\mathbb{T}_n|$ is the $(n-1)$th Catalan number

$$C_{n-1} = \frac{1}{n}\binom{2n-2}{n-1}.$$

There is an important graph structure $\mathscr{R}_n$, with vertex set $\mathbb{T}_n$, called the **rotation graph**, in which edges correspond to a local change in the tree called a **rotation**, corresponding to changing a single parenthesis pair in the parenthesization. This graph $\mathscr{R}_n$ forms the vertices and edges of an $(n-3)$-dimensional convex polytope called the **associahedron**. If we direct the edges of $\mathscr{R}_n$ in a certain fashion, we obtain the Hasse diagram for the well-studied **Tamari lattice** $\mathscr{T}_n$ on $\mathbb{T}_n$, as depicted below for $n = 4$.



The Tamari lattice $\mathscr{T}_n$ has many intriguing properties, but is diappointing in several ways. For instance, it is not ranked. Although one can encode the order $\mathscr{T}_n$ by componentwise comparison of their **bracketing vectors** $\langle T\rangle \in \{0, 1, \ldots, n-2\}^{n-1}$, introduced by Huang and Tamari, only the meet in $\mathscr{T}_n$ is given by the componentwise minimum of these bracketing vectors; the join is more subtle. Furthermore, computing the **rotation distance** $d_{\mathscr{R}_n}(T_1, T_2)$ between two trees T_1, T_2 in the graph $\mathscr{R}_n$ does not seem to follow easily from knowing their join $T_1 \vee_{\mathscr{T}_n} T_2$ and meet $T_1 \wedge_{\mathscr{T}_n} T_2$.

There is another subtle binary function on $\mathbb{T}_n$ that one would like to compute, motivated by an approach to the Four Color theorem suggested by Kauffman and explored more recently by Cooper, Rowland and Zeilberger: the size $|\text{ParseWords}(T_1, T_2)|$ of the set $\text{ParseWords}(T_1, T_2)$ of words $w = (w_1, w_2, \ldots, w_n) \in \{0, 1, 2\}^n$ which are **parsed** by both T_1 and T_2. Here, a word w is parsed by T is the labelling of the leaves of T by $w_1, w_2, \ldots, w_n$ from left to right extends to a proper 3-coloring with colors $\{0, 1, 2\}$ of *all* $2n - 1$ vertices in T. Kauffman showed that the Four Color theorem is equivalent to the statement that for all n and all $T_1, T_2 \in \mathbb{T}_n$, one has $|\text{ParseWords}(T_1, T_2)| \geq 1$.

This last application to the Four Color theorem motivated us to introduce a poset $\mathscr{C}_n$ on the set $\mathbb{T}_n$, which we call the **(right) comb order**, as a weakening of the Tamari order. $\mathscr{C}_n$ was first defined by J. M. Pallo in [8], where he proved that it is a meet-semilattice having the same bottom element as $\mathscr{T}_n$, which we denote $\text{RightCombTree}(n)$. In fact, (see the remark following Corollary 3.3 below), one way to think about the poset $\mathscr{C}_n$ is that $T_1 <_{\mathscr{C}_n} T_2$ exactly when T_1 lies on a geodesic path in the rotation graph $\mathscr{R}_n$ from T_2 to this bottom element $\text{RightCombTree}(n)$. These paths form the dark edges in the picture above of $\mathscr{T}_4$, and this darkened subgraph is the Hasse diagram of $\mathscr{C}_4$.

Although the comb order $\mathscr{C}_n$ is only a meet-semilattice whose meet $\wedge_{\mathscr{C}_n}$ does not in general coincide with the Tamari meet $\wedge_{\mathscr{T}_n}$, it fixes several deficiencies of $\mathscr{T}_n$ noted above:

- It is ranked, with exactly $\binom{n+r-2}{r} - \binom{n+r-2}{r-1}$ elements of rank r (see Theorem 3.8).
- It is **locally distributive**; each interval forms a distributive lattice (see Corollary 2.9).
- Locally, that is, within each interval of $\mathscr{C}_n$, the meet $\wedge_{\mathscr{C}_n}$ and join $\vee_{\mathscr{C}_n}$ are easily described, either in terms of intersection or union of parenthesizations (see Corollary 2.9), or by componentwise minimum or maximum of bracketing vectors (see Theorem 4.3). They *also* coincide with the Tamari meet $\wedge_{\mathscr{T}_n}$ and Tamari join $\vee_{\mathscr{T}_n}$ locally (see Corollary 4.4).
- When trees T_1, T_2 have an upper bound in $\mathscr{C}_n$, we have (see Theorem 3.2)

$$\begin{aligned} d_{\mathscr{R}_n}(T_1, T_2) &= \text{rank}(T_1) + \text{rank}(T_2) - 2 \cdot \text{rank}(T_1 \wedge_{\mathscr{C}_n} T_2) \\ &= 2 \cdot \text{rank}(T_1 \vee_{\mathscr{C}_n} T_2) - (\text{rank}(T_1) + \text{rank}(T_2)). \end{aligned}$$

- Furthermore, for T_1, T_2 with an upper bound in $\mathscr{C}_n$, we have (see Theorem 6.7)

$$\text{ParseWords}(T_1, T_2) = \text{ParseWords}(T_1 \wedge_{\mathscr{C}_n} T_2, T_1 \vee_{\mathscr{C}_n} T_2),$$

  with cardinality 2^{n-2-r}, where $r = \text{rank}(T_1 \vee_{\mathscr{C}_n} T_2) - \text{rank}(T_1 \wedge_{\mathscr{C}_n} T_2)$ (see Theorem 6.5).

Lastly, we also show in Section 5 that the well-known order-preserving surjection from the (right) weak order on the symmetric group $\mathfrak{S}_n$ to the Tamari poset $\mathscr{T}_{n+1}$ restricts to an order-preserving surjection from $\mathscr{E}_n$ to $\mathscr{C}_n$, where $\mathscr{E}_n$ is a subposet of the weak order considered by Edelman in [4].

Because we will be mainly confining our attention for the rest of this paper to the poset $\mathscr{C}_n$, we will drop the subscripts from $\wedge$, $\vee$, $>$ and $<$ when we mean meet, join, higher than, and lower than in $\mathscr{C}_n$ respectively.

2. The Comb Poset and Distributivity

Definition 2.1. For each binary tree $T \in \mathbb{T}_n$, consider the usual parenthesization of its leaves $12\ldots n$ taken in order. Then, delete all pairs of parentheses that enclose the leaf n. Call the resulting parenthesization the **reduced parenthesization of** T, denoted RP_T. Define an **element of** RP_T as either an unparenthesized leaf in RP_T, or any pair of parentheses J in RP_T (together with all the leaves and internal parenthesization enclosed by J) which is *not* enclosed by some other pair of parentheses in RP_T. Moreover, we say that RP_{T_1} is a **subset** of RP_{T_2}, or $RP_{T_1} \subset RP_{T_2}$ iff every pair of parentheses in RP_{T_1} appears in RP_{T_2} as well. Moreover, if RP_{T_1} is a subset of RP_{T_2}, then we say RP_{T_1} is a **proper subset** of RP_{T_2} or $RP_{T_1} \subsetneqq RP_{T_2}$ if, in addition to all the parenthesis pairs from RP_{T_1}, RP_{T_2} contains at least one other pair of parentheses as well.

Proposition 2.2. *Given the elements $1, 2, \ldots, n$ in that order, a parenthesization of those elements is the reduced parenthesization for some tree $T \in \mathbb{T}_n$ iff the two following conditions hold: n is not enclosed by any parenthesis pair, and each pair of parentheses encloses precisely two* ***factors*** *within it (which we shall call the* ***left factor*** *and the* ***right factor****).*

Proof. Suppose we take some tree T and its reduced parenthesization RP_T. Then, by definition, n is not enclosed by any parenthesis pair. Furthermore, we know that in the *full* parenthesization of the leaves of the tree T, each parenthesis pair encloses two factors (this is a well-known fact about these parenthesizations). We leave it to the readers to verify that even after deleting parenthesis pairs in order to get RP_T, every remaining pair of parentheses still encloses precisely two factors.

Conversely, take some parenthesization of the form given. Then, the full parenthesization can be recovered by pairing the two rightmost elements of RP_T successively. The expression that we will get is a full parenthesization of $1, 2, \ldots, n$ such that every pair of parentheses encloses two factors, and it is well known that such an expression represents the usual parenthesization of a binary tree. $\square$

Remark. If two pairs of parentheses J_1 and J_2 enclose the same two factors in the same order, then the internal parenthesization of J_1 and J_2 have to be the same. Definition 2.1 then implies that, if $RP_{T_1} \subset RP_{T_2}$, then each of the parenthesis pairs common to the two reduced parenthesizations also has the *same* factors in both RP_{T_1} and RP_{T_2}.

Example. The reduced parenthesization of the following tree is 1((23)4)(56)7.

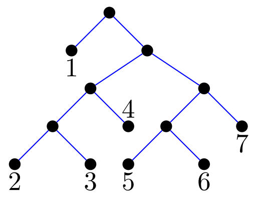


FIGURE 1.

The reduced parenthesization of this tree has four elements, given by 1, ((23)4), (56), and 7. Note that n by itself is always an element of the reduced parenthesization of any tree in $\mathbb{T}_n$.

Remark. All n-leaf binary trees have a unique reduced parenthesization. This follows because there is a bijection between the full parenthesization of a tree T, and its reduced parenthesization. The full parenthesization is recovered by pairing the two rightmost elements of RP_T successively.

Definition 2.3. For $n \geq 2$, the **right comb tree of order** n, denoted by RightCombTree(n) $\in \mathbb{T}_n$, is the n-leaf binary tree corresponding to the "empty" reduced parenthesization $12 \ldots n$. Similarly, the **left comb tree** of order n is defined as the n-leaf binary tree corresponding to the reduced parenthesization $(((\ldots((a_1 a_2)a_3)\ldots)a_{n-2})a_{n-1})a_n$.

Example. RightCombTree(5), the right comb tree of order 5, is given by the following tree. The nodes labeled $1, \ldots, 5$ represent the leaves of the tree, and $6, \ldots, 9$ represent the internal vertices. Note that the structure of the left comb tree of order 5 is given by the reflection of the right comb tree about the y axis; the leaves have to be renumbered, of course.

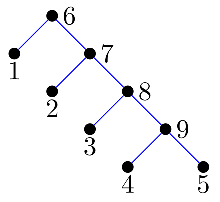


FIGURE 2. RightCombTree(5)

Definition 2.4. We define the **(right) comb poset of order** n as the poset whose elements are given by elements from $\mathbb{T}_n$. If T_1 and T_2 are two trees in the poset, then $T_2 > T_1$ iff $RP_{T_1} \subsetneq RP_{T_2}$. We denote the right comb poset of order n by $\mathscr{C}_n$.

Remark. It is evident from the order relation that the poset $\mathscr{C}_n$ has a unique minimal element, which is the tree corresponding to the "empty" parenthesization $12\ldots n$, which is RightCombTree(n) from Definition 2.3. This empty reduced parenthesization is clearly a proper subset of any other reduced parenthesization.

Example. The Hasse diagram of the right comb poset of order 5 is shown below, in Figure 3. The trees themselves corresponding to the parenthesizations are given in Appendix 1.

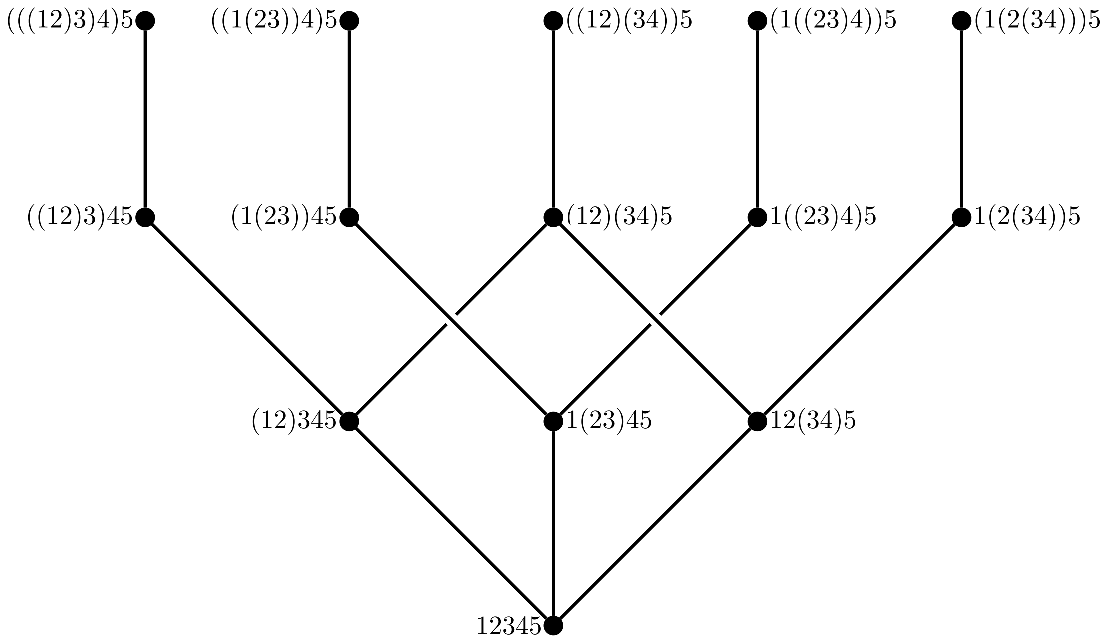


FIGURE 3. The Hasse diagram of $\mathscr{C}_5$

Definition 2.5. For any tree $T \in \mathbb{T}_n$, define the **reduced pruned poset of** T, denoted by P_T, as the set of join irreducibles corresponding to parenthesis pairs in RP_T, ordered by inclusion. So, for instance, if $RP_T = ((12)3)4(56)7$, and we number the parenthesis pair enclosing leaves 1 through 3 with the number 1, the pair enclosing leaves 1 and 2 with the number 2, and the pair enclosing 5 and 6 with the number 3, then P_T corresponds to the set

Proposition 2.6. *For any $T \in \mathbb{T}_n$, the interval [RightCombTree(n), T] in $\mathscr{C}_n$ is isomorphic to the lattice of order ideals in the reduced pruned poset of T, ordered by inclusion. In other words, for any tree T,*

$$[\mathit{RightCombTree}(n), T] \cong J(P_T).$$

Proof. The fact that there is a bijection between trees in the interval [RightCombTree(n), T] and elements in $J(P_T)$ is trivial, and follows from the definition of P_T and Proposition 2.2. That this bijection is order-preserving follows from the fact that both are ordered by inclusion. We omit the completely trivial details of this proof. □

Proposition 2.7. *In $\mathscr{C}_n$, T_1 covers T_2 iff RP_{T_1} can be obtained from RP_{T_2} by adding precisely one parenthesis pair that encloses two adjacent elements of RP_{T_2}.*

Corollary 2.8. *$\mathscr{C}_n$ is a ranked poset, with the rank of any tree T in $\mathscr{C}_n$ given by the number of parenthesis pairs in RP_T.*

Corollary 2.9. *Any interval in $\mathscr{C}_n$ is a distributive lattice, with the reduced parenthesizations of the join and meet of trees T_1 and T_2 in an interval given by the ordinary union and intersection of parenthesis pairs from RP_{T_1} and RP_{T_2}.*

Corollary 2.10. *For any two trees T_1 and T_2 that are in the same interval of $\mathscr{C}_n$, then $rank(T_1) + rank(T_2) = rank(T_1 \wedge T_2) + rank(T_J)$.*

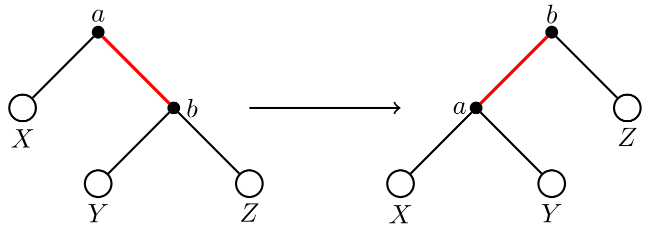
Proof. This is a well-known fact about distributive lattices. □

Note. In the rest of this paper, we shall consider only the right comb poset of order n. All the results for the right comb tree hold for LeftCombTree(n) as well, by symmetry. Hence, whenever we state any result about the right comb poset, it holds for the *left* comb poset as well, which can be defined analogously.

Definition 2.11. For $n \geq 2$, the **right arm** of any arbitrary tree $T \in \mathbb{T}_n$ is the connected graph obtained by taking the edge connecting the root of T to its right child, and taking each subsequent edge connecting each *subsequent* vertex to the right child below that. We stop including the edges as soon as we come to a vertex which has no right child, which clearly corresponds to leaf n. We shall call the length of the right arm the **dexterity** of the tree T. The tree in Figure 2 has dexterity 4, and its right arm is given by the connected graph 67895. We leave it to the reader to verify that the dexterity of any tree $T \in \mathbb{T}_n$ must be a number between 1 and $n-1$, both inclusive.

Proposition 2.12. *For any tree $T \in \mathbb{T}_n$, the elements other than leaf n in RP_T represent precisely the dangling left subtrees of the vertices on the right arm of T, taken in order, with the leftmost element corresponding to the left subtree of the uppermost vertex on the right arm of T.*

Proposition 2.13. *For any two trees T_1 and T_2 in $\mathbb{T}_n$, T_1 covers T_2 in $\mathscr{C}_n$ iff T_1 can be obtained from T_2 by a right rotation of the form*



such that the center of rotation (the vertex a in the figure) lies on the right arm of T_2. In particular, T_1 is obtained from T_2 in this way iff RP_{T_1} is obtained from RP_{T_2} by adding an additional pair of parentheses, from Proposition 2.7. Note that in the figure given, X, Y and Z represent arbitrary subtrees, while a and b represent vertices. In particular, from Proposition 2.7, any tree $T \in \mathbb{T}_n$ can be obtained from the right comb tree by a sequence of tree rotations of this form.

Corollary 2.14. *For any tree $T \in \mathbb{T}_n$, if $dex(T)$ denotes the dexterity of T and k denotes the rank of T in $\mathscr{C}_n$, then we have $dex(T) = n - 1 - k$.*

Remark. It is evident that the comb poset that we are considering is the same as the poset that J. M. Pallo defined in [8]. It is evident, furthermore, from Lemma 3 of his paper that the comb poset is a meet semilattice. Armed with a knowledge of the structure of the comb poset, we can now endeavor to find some of the nice properties that it exhibits.

3. Some Properties of the Comb Poset

3.1. **Distance Properties.** Here we will prove some properties of the comb poset which relate to the distance between pairs of trees in the rotation graph $\mathscr{R}_n$.

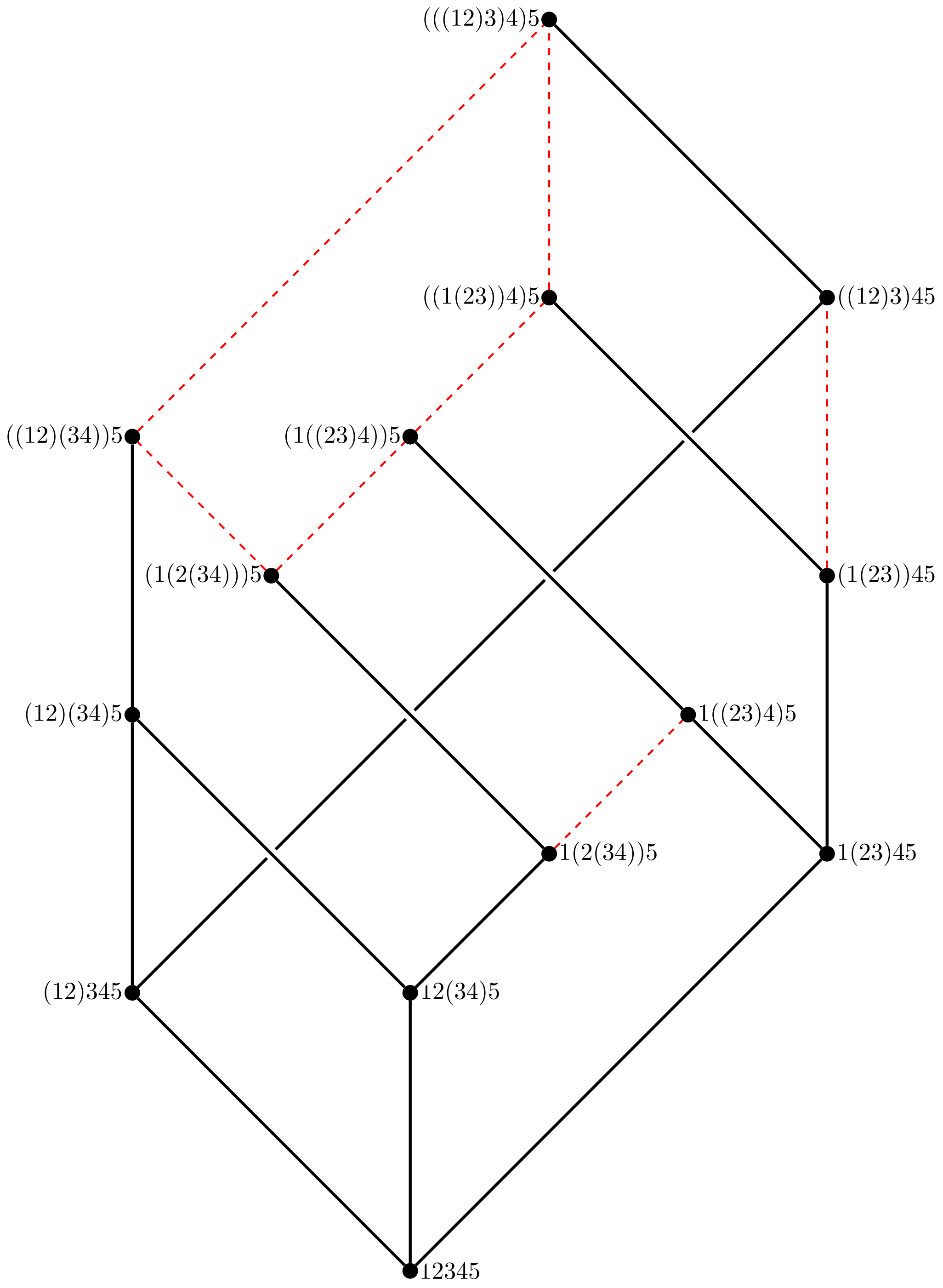


FIGURE 4. $\mathscr{C}_5$ obtained by deleting the red edges of $\mathscr{T}_5$

Proposition 3.1. *Suppose T_1 and T_2 are two trees in some interval in $\mathscr{C}_n$. Furthermore, suppose there is a pair of parentheses J in RP_{T_1} and a pair of parentheses J_2 in RP_{T_2} such that J_1 and J_2 enclose a common factor (which in the hypothesis may be a left factor for one and a right one for the other). Then, $J_1 = J_2$.*

Proof. Suppose without loss of generality that E is a left factor in J_1 but a right factor in J_2. Then, because $T_1 \vee T_2$ exists by our hypothesis, we know by Proposition 2.7 that J_1 and J_2 have to be in $RP_{T_1 \vee T_2}$ as well, which then clearly implies that E is enclosed as a single factor by a pair of parentheses in $RP_{T_1 \vee T_2}$, which is a contradiction to Proposition 2.2. So, E has to be either the left or the right factor for both J_1 and J_2. Now assume without loss of generality that E is the *left* factor for J_1 and J_2. If we denote the right factor of J_1 and J_2 by E_1 and E_2 respectively, then these have to be the right factor of the corresponding parenthesis pair in $RP_{T_1 \vee T_2}$, which forces $E_1 = E_2$. Then it follows that $J_1 = J_2$, because a pair of parentheses is characterized wholly by its factors. □

Theorem 3.2. *If T_1 and T_2 are two trees in some interval in $\mathscr{C}_n$, then the shortest distance between them* along the edges of the rotation graph $\mathscr{R}_n$ *is given by*

$$d(T_1, T_2) = rank(T_1) + rank(T_2) - 2 \cdot rank(T_1 \wedge T_2).$$

Equivalently, from Corollary 2.10, this shortest distance is also given by

$$d(T_1, T_2) = 2 \cdot rank(T_1 \vee T_2) - rank(T_1) - rank(T_2).$$

Proof. We know from Corollary 2.9 that $RP_{T_1 \wedge T_2}$ contains all the common parenthesis pairs of RP_{T_1} and RP_{T_2}. Hence, RP_{T_1} and RP_{T_2} are formed by adding respectively some r and s *extra* pairs of parentheses to $RP_{T_1 \wedge T_2}$, from Proposition 2.7, where r and s are nonnegative integers.

We claim that the r extra parenthesis pairs $J_1, J_2, \ldots, J_r$ in RP_{T_1} and the s parenthesis pairs $J'_1, J'_2, \ldots, J'_s$ in RP_{T_2} are disjoint, i.e. there is no J_m and J'_n such that they both enclose the same leaf from $1, 2, \ldots, n$. Suppose, on the contrary, that some leaf i is common to some pair from each of the sets $\{J\}$ and $\{J'\}$. Take the smallest parenthesis pairs J_m and J'_n both enclosing the leaf i. Now, it is not hard to see that this forces $J_m = J'_n$ by Proposition 3.1. But then, this parenthesis pair must belong to $RP_{T_1 \wedge T_2}$ as well, a contradiction.

So, $J_1, J_2, \ldots, J_r$ and $J'_1, J'_2, \ldots, J'_s$ are disjoint parenthesis pairs. Now, let's ignore $\mathscr{C}_n$ for a moment, and consider the rotation graph $\mathscr{R}_n$. Suppose we want to get from T_1 to T_2 along the edges of $\mathscr{R}_n$. What is the length of the shortest path? Note that any such sequence of rotations will have the overall effect of removing the r parenthesis pairs $J_1, \ldots, J_r$ from RP_{T_1} and adding the s parenthesis pairs $J'_1, \ldots J'_s$ needed to get RP_{T_2}. Furthermore, any rotation shifts precisely one parenthesis pair in the full parenthesization, and corresponds to shifting a parenthesis pair, *or* adding a parenthesis pair, *or* subtracting a pair in the reduced parenthesization. In particular, a single rotation cannot correspond to two or more of those operations. This implies that we need r distinct rotations to remove the r parenthesis pairs J, and s distinct rotations to add the s parenthesis pairs J'.

We claim now that the r rotations needed to remove the Js are all distinct from the s rotations needed to add the J's. If this is not the case, then there is some rotation which simultaneously removes some J_m and adds some J'_n to the reduced parenthesization. In particular, this can only happen when the parenthesis pair J_m "moves" right or left by a single rotation and becomes J'_n.

But any rotation that moves a pair of parentheses J either maps an expression of the form $(ab)c$ to $a(bc)$, or vice versa. In particular, the parenthesis pair J has a factor (b in the example shown) that still remains within it after a rotation in any direction. This contradicts the fact that the J_ms and the J'_ns are disjoint, proving that the r distinct rotations to remove the Js and the s distinct rotations to add the J's are disjoint. Therefore, the length of the shortest path between T_1 and T_2 along the edges of the rotation graph $\mathscr{R}_n$ must be at least $r + s$, and the rest of the result follows immediately. □

Corollary 3.3. *Given any two trees T_1 and T_2 such that $T_1 > T_2$ in $\mathscr{C}_n$, the union of paths connecting T_1 and T_2 in $\mathscr{C}_n$ are precisely* all *the shortest paths between T_1 and T_2 along the edges of $\mathscr{R}_n$.*

Proof. This corresponds to the case when $r = 0$ or $s = 0$ in the proof of Theorem 3.2 above. The theorem proves that any path in $\mathscr{C}_n$ is a shortest path in $\mathscr{R}_n$. The other direction is also trivial to see, because if $r = 0$, then any shortest path between T_1 and T_2 in $\mathscr{R}_n$ will necessarily have to precisely *add* all the s extra parenthesis pairs, and from Proposition 2.7, this sequence will correspond to an upward path in $\mathscr{C}_n$. □

Remark. It is now trivial to see that an alternative way of viewing the comb poset is as follows: $\mathscr{C}_n$ is obtained by taking the rotation graph $\mathscr{R}_n$, fixing the right comb tree, and then, for every tree T different from the right comb itself, taking the union of all shortest paths along the rotation graph from the right comb tree to T. The union of all the paths we included yields the edges of $\mathscr{C}_n$, and then we arrange this union as a ranked poset in the obvious way.

Conjecture 3.4. *For trees T_1 and T_2 with a common upper bound in $\mathscr{C}_n$, any shortest path between them in $\mathscr{R}_n$ also lies in $\mathscr{C}_n$. In fact, such a shortest path lies within the interval $[T_1 \wedge T_2, T_1 \vee T_2]$ in $\mathscr{C}_n$.*

Corollary 3.5. *The rank of any tree $T \in \mathbb{T}_n$ in $\mathscr{C}_n$ is its distance from the right comb tree along the edges of the rotation graph $\mathscr{R}_n$. Furthermore, from Corollary 2.8, the distance of T from the right comb tree in $\mathscr{R}_n$ is given by the number of parenthesis pairs in RP_T.*

Remark. It can be easily shown from the results above that the **diameter** of the rotation graph $\mathscr{R}_n$, given by the maximum distance taken over all pairs of trees in $\mathscr{R}_n$, is at most $2n-4$ for any $n \in \mathbb{N}$. Sleator, Tarjan and Thurston established an upper bound of $2n-6$ on the diameter of the rotation graph in [10] for $n \geq 11$.

3.2. **Enumerative Properties.** We now prove a few enumerative properties of $\mathscr{C}_n$. To simplify notation slightly, we will denote by Q_i the set of trees in the ith rank of $\mathscr{C}_n$.

Proposition 3.6. *For $n \geq 3$, $\mathscr{C}_n$ is a ranked poset with many maximal elements. Furthermore, a tree is a maximal element iff its rank is $n-2$ in $\mathscr{C}_n$ (or equivalently, from Corollary 2.14, iff its dexterity is 1). In particular, the left comb tree is in the maximal rank in $\mathscr{C}_n$.*

Proposition 3.7. *For $0 \leq i \leq n-2$, every tree in Q_i is covered by precisely $n-2-i$ trees.*

Proof. A tree in rank Q_i has, by Corollary 2.14, dexterity $n-1-i$, i.e. its right arm is of length $n-1-i$. Equivalently, its right arm has $n-i$ vertices. But notice that by the way rotation is defined, the center of rotation cannot be either of the two lowest vertices on the right arm, because the lowest vertex by definition is a leaf. Hence, the only admissible rotations are on the upper $n-i-2$ vertices on the right arm, and they all yield distinct trees (because of distinct reduced parenthesizations). □

Theorem 3.8. *For $0 \leq r \leq n-2$, the number of elements in rank r of $\mathscr{C}_n$ is precisely*

$$|Q_r| = \binom{n+r-2}{r} - \binom{n+r-2}{r-1}.$$

We state a famous lemma first. The proof of the lemma is well known, and can be found, for instance, in [5].

Lemma 3.9 (Lagrange Inversion)**.** *For any function of the form $y = x\varphi(y)$, where φ is a power series that does not vanish at 0, the coefficient of x^n in any power series function $\psi(y)$ (which we will denote by $[x^n]\psi(y)$ to simplify notation) is given by*

$$[x^n]\psi(y) = \frac{1}{n}[u^{n-1}]\varphi(u)^n\psi'(u).$$

In particular, for just a single power, we have

$$[x^m]y^k = \frac{k}{n}[u^{n-k}](1-u)^{-n}.$$

Proof of Theorem 3.8. First we will show that $|Q_r|$ is given by the coefficient of x^r in the expression $c(x)^{n-r-1}$, where $c(x)$ is the generating function for the Catalan numbers. After that, we will find this precise number.

Fix a tree T in rank r of $\mathscr{C}_n$. Note that from Corollary 2.14, we know that the dexterity of T is $n-r-1$, and so there are precisely $n-r-1$ vertices on the right arm of T which have (possibly single-leaf) left subtrees. The total number of leaves on these left subtrees is $n-1$, because the nth leaf is on the right arm. So, we are looking for the number of ways of distributing $n-1$ leaves among $n-r-1$ "blank" subtrees, such that the subtrees are also binary trees in their own right. Furthermore, each of the $n-r-1$ subtrees must have at least one leaf. Therefore, we can simplify the problem slightly by putting in a leaf in each subtree to begin with. It is now obvious that we are looking for the number of ways of putting $n-1-(n-r-1)=r$ indistinguishable objects in $n-r-1$ numbered boxes, with each box weighted C_i (if i objects go into it).

The number of ways of doing this is given by the coefficient of x^r in the expansion $(1+C_1x+C_2x^2+C_3x^3+\ldots)^{n-r-1}$. But the expression which is being raised to the $(n-r-1)$th power is nothing but the generating function $c(x)$ for the Catalan numbers. This concludes the first part.

To see the second part, we will use Lagrange Inversion as follows: if $c(x)$ is defined as above, then we set $y(x)=xc(x)$. Then, y satisfies $y=x(1-y)^{-1}$. Now, using Lagrange Inversion from Lemma 3.9 above with $k=n-r-1$ and $m=n-1$, we get, after a few simplifications, the much less intimidating formula

$$|Q_r| = \frac{n-r-1}{n-1}\binom{n+r-2}{r}.$$

We leave it to the reader to verify that the expression above reduces to the form desired. □

Corollary 3.10. *The sizes of each rank in $\mathscr{C}_n$ weakly increase as we go up the poset to the rank, and in fact they* strongly *increase up to the penultimate rank.*

Proof. From Theorem 3.8, it is not hard to see that $|Q_i| = \frac{(n+i-2)!}{i!(n-1)!}\cdot(n-i-1)$, and so, for arbitrary adjacent ranks r and $r+1$, we have

$$\frac{|Q_{r+1}|}{|Q_r|} = \frac{(n+r-1)(n-2-r)}{(r+1)(n-1-r)}.$$

We know that the rank size increases weakly whenever the numerator is at least as large as the denominator, and hence the condition for weakly increasing rank size is simply $(n+r-1)(n-r-2)\geq(r+1)(n-r-1)$. But this condition reduces after a few simple manipulations to the condition $n^2-4n+3-r(n-1)\geq 0$. But since $r\leq n-3$ from Proposition 3.6 (recall that the rank $r+1$ is defined), we have $r(n-1)\leq n^2-4n+3$, and hence our condition above is trivially true. Furthermore, for $r\leq n-4$, we have the strict inequality $r(n-1)<n^2-4n+3$, implying a strong increase. □

Corollary 3.11. *The maximal rank (which is Q_{n-2} by Proposition 3.6) as well as the rank Q_{n-3} just below that, both have precisely C_{n-2} elements.*

Proof. These two results follow by direct calculation, using Theorem 3.8. □

3.3. **Other Properties.**

Proposition 3.12. *Any upward-going path in $\mathscr{C}_n$ is a directed path in the Tamari lattice $\mathscr{T}_n$, if we define the right comb tree to be the minimal element of the Tamari lattice instead of the usual left comb tree (and therefore reverse the usual Tamari order).*

Proof. From Proposition 2.7, we know that T_2 is a cover of T_1 in $\mathscr{C}_n$ iff RP_{T_2} can be obtained from RP_{T_1} by adding precisely one more parenthesis pair. We leave it to the reader to verify that adding any parenthesis pair to RP_T is the same as shifting a pair of parentheses to the *left* in the corresponding full parenthesization of the leaves of T. □

Proposition 3.13. *There is an order-preserving involution on $\mathscr{C}_n$.*

Proof. For any tree T, take RP_T, and construct a new parenthesization $RP_{T'}$ by enclosing leave $n-j$ through $n-i$ in $RP_{T'}$ for every parenthesis pair in RP_T which encloses leaves i through j. After all parenthesis pairs in RP_T have been reconstructed as above in $RP_{T'}$, it is not hard to see (using Proposition 2.2) that $RP_{T'}$ corresponds to a tree T'. Define π to be the map that takes T to T' as described above. Then, π is an order preserving involution on $\mathscr{C}_n$. □

4. Tamari Meets and Joins for two Trees in Some Interval

From Corollary 2.9, we know the precise meaning of the meet and join of a pair of trees in some interval of our lattice. It is natural to ask how these joins and meets relate to joins and meets in the Tamari lattice. The first observation is that, while two arbitrary trees in $\mathscr{C}_n$ *do* have a well-defined meet in $\mathscr{C}_n$, this meet does not necessarily correspond to the Tamari meet. To see this, consider the pair of trees represented by $T_1 = (((12)3)4)5$ and $T_2 = ((1(23))4)5$. This pair has Tamari meet T_2, while their meet in $\mathscr{C}_n$ is just the right comb tree. Furthermore, any two trees have a join in the Tamari lattice, while this is not necessarily true of any arbitrary pair in the right comb poset.

However, something much stronger can be said if both the trees under consideration are in some interval in the comb poset; it turns out that their meet and join in $\mathscr{C}_n$ correspond to their Tamari meet and Tamari join respectively.

To prove this, we shall delve into the notion of **bracketing vectors**, as analyzed by J. M. Pallo. In [9], Pallo proved an equivalence between the Tamari meet of two trees and the coordinatewise minimum of the bracketing vectors corresponding to these two trees. We shall use this equivalence and prove a potent result.

First, we will define what bracketing vectors mean in the context of this paper. In [6], Huang and Tamari discuss bracketing vectors in a much more formal treatment, but we will not need the details for our purposes. We will use the same concept of bracketing vectors used in [9].

We will make use of the notation introduced by Stanley in [11]; Stanley considered the set of "pruned" trees on $n-1$ vertices, which are obtained from ordinary n-leaf binary trees by simply deleting all the leaves. The fact that this operation of "pruning" is a bijection (between n-leaf binary trees and pruned trees on $n-1$ vertices) is well-known, proven, for instance, in [12]. Furthermore, there is an natural numbering of the vertices of the pruned tree using $1, 2, \dots, n-1$, in which a vertex receives a higher number than any vertex in its left subtree, but a lower one than any vertex in its right subtree. This labeling is unique, and well known. It is called the **in order** labeling of a pruned tree on $n-1$ vertices, discussed by Stanley in p. 24 of [11].

Example. The following pruned tree on 8 vertices, corresponding to the 9-leaf binary tree whose reduced parenthesization is $((12)(34))5(6(78))9$, is labeled in the in order labeling.

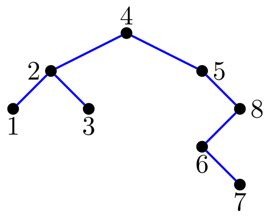


Figure 5.

Definition 4.1. Consider the "pruned" binary tree representation of some tree $T \in \mathbb{T}_n$, and number the $n-1$ vertices by the in order labeling. Then, the bracketing vector for T corresponds to the ordered $(n-1)$-tuple of integers, whose jth coordinate is the number of vertices in the left subtree of the vertex j in the pruned tree. In particular, the first coordinate of a bracketing vector is always 0.

Proposition 4.2. *For any $T \in \mathbb{T}_n$, take its corresponding RP_T, and then enclose every leaf by an additional parenthesis pair (which therefore encloses just one factor, the leaf itself). Then, for $i \in \{1, \dots, n-1\}$, take the largest parenthesis pair such that the leaf i is the largest leaf enclosed by that parenthesis pair in the representation above. Then, if this parenthesis pair encloses k_i leaves, then the ith component of the bracketing vector for T is precisely $k_i - 1$.*

Proof. From Proposition 2.12, we know that the unbracketed elements in RP_T correspond to vertices on the right arm that do not have left subtrees (in other words, their coordinate in the bracketing vector is 0). It is trivial to see that this implies the desired result for $k_i = 1$. For $k_i > 1$, we can follow a recursive argument by considering the subtrees represented by the bracketed elements as trees in their own right. The details of this proof are left as an exercise for the reader. □

Theorem 4.3. *Let $\langle T\rangle$ denote the bracketing vector for T. Let T_1 and T_2 be arbitrary trees in the same interval of $\mathscr{C}_n$. Then, their meet and join in $\mathscr{C}_n$ are given by the trees corresponding respectively to the componentwise minimum and the componentwise maximum of $\langle T_1\rangle$ and $\langle T_2\rangle$.*

Proof. For any coordinate i of $\langle T_1 \vee T_2\rangle$, the corresponding leaf is the largest leaf in one of more parenthesis pairs in the representation described in the hypothesis of Proposition 4.2. It is easy to see, from Corollary 2.9, that because these parenthesis pairs represent the union of pairs from RP_{T_1} and RP_{T_2}, the same parenthesis pairs must be present in at least one of RP_{T_1} and RP_{T_2}. Furthermore, any *bigger* pair enclosing the corresponding leaf would also have to appear in $RP_{T_1 \vee T_2}$, a contradiction. From Proposition 4.2 above, this proves the theorem for joins, and the proof for meets is analogous. □

Corollary 4.4. *For T_1 and T_2 in some interval in $\mathscr{C}_n$, their meet and join in $\mathscr{C}_n$ correspond respectively to their meet and join in the Tamari lattice $\mathscr{T}_n$.*

Proof. The statement follows directly from the equivalence proven in Theorem 2 of [9], and Theorem 4.3 above. □

5. Relation with Edelman's Poset

In [4], Paul Edelman introduced a poset obtained by imposing an additional condition on the right weak order on the symmetric group $\mathfrak{S}_n$, which we summarize now.

Definition 5.1. The **right weak order on** $\mathfrak{S}_n$ is a partial ordering of the elements of $\mathfrak{S}_n$ defined as the transitive closure of the following covering relation: a permutation A is a cover of another permutation $B = (a_1, \dots, a_n)$ if $A = (a_1, \dots, a_{j-1}, a_{j+1}, a_j, a_{j+2}, \dots, a_n)$ is obtained from B by a single transposition that is also an inversion. In other words, A is obtained from B by taking a pair (a, b) of adjacent elements in B such that $a < b$, and then transposing them. It is a consequence of this ordering that the identity permutation $(1, 2, \dots, n)$ is the unique minimal element of the poset.

Edelman imposed an additional constraint on this ordering, where he only considered A to be a covering element of B, if, after the transposition of a_j and a_{j+1} as above, nothing to the left of a_{j+1} in A is greater than a_{j+1}. Edelman's poset is a subposet of the right weak ordering on $\mathfrak{S}_n$. We shall denote this poset by $\mathscr{E}_n$.

Example. The following is the representation of $\mathscr{E}_3$. The dashed red line represents the only extra edge needed to specify the right weak order on $\mathfrak{S}_3$ without Edelman's further imposition.

Although this poset is clearly not a lattice, because it has are numerous maximal elements (specifically, $(n-1)!$ maximal elements, as Edelman proved), it was proven that $\mathscr{E}_n$ has the beautiful property that every interval of it is a distributive lattice, much like the comb poset $\mathscr{C}_n$. In this section of the paper, we will prove a further analogy between the comb poset and $\mathscr{E}_n$: we will show that there is a natural order-preserving surjective map from $\mathscr{E}_n$ to $\mathscr{C}_{n+1}$.

First, we will define this natural map from $\mathscr{E}_n$ to $\mathscr{C}_{n+1}$. In order to do this, we will make use of the notation introduced by Stanley in [11] and the in order labeling once more, whence we will obtain the surjection from $\mathfrak{S}_n$ to the set of pruned trees on n vertices.

Definition 5.2 (Inverse Stanley map)**.** We define the **inverse Stanley map** to be the map from $\mathscr{E}_n$ to the set of pruned trees on n vertices, obtained as follows: given any permutation $(a_1, a_2, \dots, a_n) \in \mathfrak{S}_n$, we construct a pruned tree by taking a_1 to be the root. Then, we take *in order* the elements from a_2 through a_n that are less than a_1 (call this set A), and write them in that order as the left child of a_1. We do the same with the elements from a_2 through a_n that are greater than a_1 (denoted by B), and write them (again, in order) as the right child of a_1. Note that if A or B is empty, we do *not* draw the branch to the corresponding child at all. We are allowed to do that, because a vertex can have 0, 1 or 2 children in a pruned tree. Once we have finished this step, we keep repeating this operation recursively on each of the lower elements until each vertex has received only one number

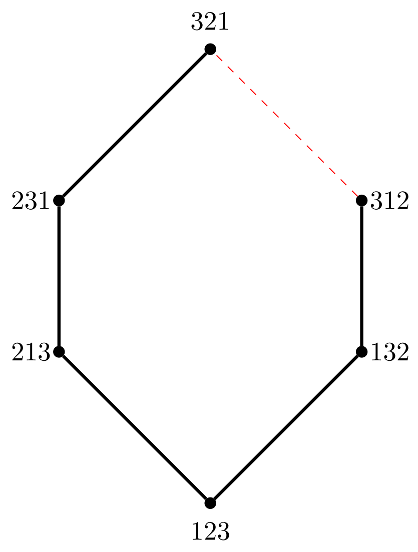


FIGURE 6. Edelman's Poset $\mathscr{E}_3$.

as a label. We have at this point obtained the final pruned tree. It is clear by the nature of the map that we get the in order labeling of the pruned binary tree on n vertices, which we discussed before.

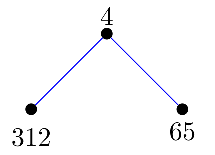
Remark. It is well known that the inverse Stanley map is a surjection: given any pruned tree on n vertices, we can always find a permutation in $\mathfrak{S}_n$ that maps to that tree under the inverse Stanley map. It is left as an exercise for the reader to prove that it is not an injection.

Theorem 5.3. *The inverse Stanley map of Definition 5.2 is an order-preserving surjection from $\mathscr{E}_n$ to $\mathscr{C}_{n+1}$.*

Proof. We will prove that the inverse Stanley map is order-preserving. We shall prove an equivalent statement: suppose B covers A in $\mathscr{E}_n$. Then, if T_2 is the image of B and T_1 is the image of A under the inverse Stanley map, then either T_2 is the same as T_1, or T_2 covers T_1 in $\mathscr{C}_{n+1}$.

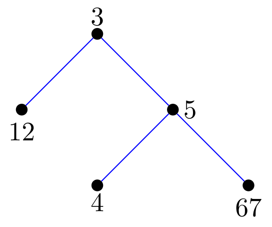
To see this, suppose A and B are as above. Let A be $(a_1, a_2, \ldots, a_j, a_{j+1}, \ldots, a_n) \in \mathfrak{S}_n$, and B be $(a_1, a_2, \ldots, a_{j+1}, a_j, a_{j+2}, \ldots, a_n) \in \mathfrak{S}_n$, with B covering A in $\mathscr{E}_n$. From the definition of a cover in the right weak order, we have $a_j < a_{j+1}$. Now, if $j = 1$, then this transposition represents a right rotation centered on the root, and therefore T_2 covers T_1 in $\mathscr{C}_{n+1}$, from Proposition 2.13. This fact is not hard to see, and we will leave it as an exercise. So assume $j \neq 1$; in other words, a_j is not the root a_1 of the tree.

Now, the first thing to note is that if $a_j < a_1 < a_{j+1}$, then a_j and a_{j+1} are in different subtrees of the root. Furthermore, they are adjacent elements, and hence, the trasposition of a_j and a_{j+1} will not be reflected when we "separate" out the different subtrees. This is illustrated in the example above. For instance, if $A = (4, 3, 1, 6, 2, 5)$ and $B = (4, 3, 6, 1, 2, 5)$, then they both yield the tree

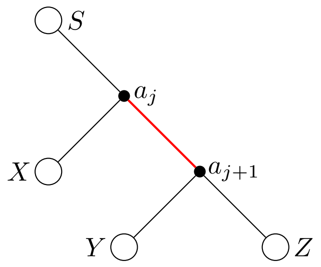


after the first step of the inverse Stanley map, and hence, they are identical trees. So, in order for T_1 and T_2 to be different, a_j and a_{j+1} must be on the same side of the root. But because of Edelman's extra imposition, we have $a_{j+1} > a_i$ for $i \in \{1, \ldots, j\}$, and in particular, $a_{j+1} > a_1$, and so the condition of the two trees T_1 and T_2 being different forces $a_{j+1} > a_j > a_1$, i.e. a_j and a_{j+1} are both in the right subtree of the root.

Now, we can disregard all the elements immediately to the right of a_1 in A which are less than a_1, because they are in the left subtree of the root, and hence cannot play any role in determining the labels of a_j and a_{j+1} at all. So consider the *first* element, say a_r, to the right of a_1 in A, which is greater than a_1. Suppose $r \neq j$ (the case $r = j$ is what the condition $T_1 \neq T_2$ reduces to eventually, and this is discussed slightly later on in the proof). Then, we can argue that the tree will remain the same if $a_j < a_r < a_{j+1}$, by the same argument as above, and so they must be both on the right subtree of a_r if the trees T_1 and T_2 are to be different. This is illustrated by the example in $\mathscr{E}_7$, if we take the two elements $A = (3, 1, 5, 2, 4, 6, 7)$ and $B = (3, 1, 5, 2, 6, 4, 7)$. Note that even though B covers A in $\mathscr{E}_7$, the trees become identical under the inverse Stanley map, as seen from

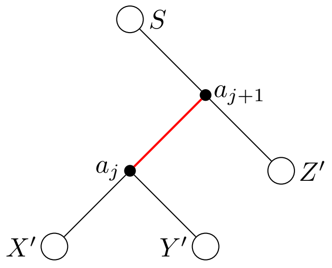


Hence, a repeated application of the previous argument forces the conclusion that, if T_1 and T_2 are to be different, then a_j and a_{j+1} must continue being on the same (in particular, the *right*) subtree at each step, and hence, we conclude that they must be consecutive elements on the right arm of T_1. Hence, in order for T_1 and T_2 to be *different* trees, T_1 must be of the form



Here the white circle S denotes the arbitrary parent tree of the entire subtree shown, with the condition that a_j and a_{j+1} lie on the right arm. The white circles X, Y and Z denote arbitrary subtrees, whose interpretations in terms of the elements in A are as follows: X is the ordered sequence of elements appearing *after* a_j which are less than a_j, Z is the ordered sequence of elements appearing *after* a_{j+1} which are greater than a_{j+1}, and Y is the ordered sequence of elements appearing after a_j that lie between a_j and a_{j+1}.

Now, consider what happens to T_2, when we switch a_j and a_{j+1}. We draw T_2 now, as shown.



Here, S is clearly going to be unchanged, and a_j and a_{j+1} must go to the spaces shown. In addition, there will be subtrees X', Y', and Z' as drawn above. However, notice that, if we interpret what these subtrees must be with respect to the permutation B, the fact that a_j and a_{j+1} are adjacent forces the conclusion that the subtrees are unchanged from their interpretation in A, or in other words that $X = X'$, $Y = Y'$ and $Z = Z'$. So then, T_2 is obtained by a rotation on T_1 of the form given in Proposition 2.13, i.e. a right rotation centered on a vertex on the right arm of T_1. Therefore, T_2 covers T_2 in $\mathscr{C}_n$, and hence we are done. $\square$

Remark. It turns out that we can get an alternative proof of Corollary 2.9 by using a method analogous to the one that Edelman used to obtain the same result for $\mathscr{E}_n$. We omit this alternative proof in this paper, but it is not particularly hard to derive it based on Theorems 2.7 and 2.8 in [4], and proving the analogous results for $\mathscr{C}_n$.

6. The ParseWords Function for the Comb Poset

What motivated us to define and explore the structure of the right comb poset in great detail in Sections 2 and 3 was the problem of finding common parse words for any two n-leaf trees, for an arbitrary positive integer n. The problem, discussed in great detail in [3], is of great importance because of its implications: it is equivalent to the Four Color theorem.

In its shortest form, the Four Color theorem states that every bridgeless plane map can be colored with at most four colors such that no two regions sharing a boundary are colored by the same color. The problem remained unsolved for a long time until 1977, when Appel and Haken came up with a controversial proof (see [1], [2]) that reduced the entire problem to 1,936 cases, and a computer checked all of them. Thirteen years later, Kauffman proved in [7] that the Four Color theorem was equivalent to the problem of finding a common parse word for any two arbitrary n-leaf binary trees, which he stated in a slightly different form.

Kauffman had proven his problem by showing the equivalence. Recently, in [3], Cooper, Rowland and Zeilberger recognized that the Four Color theorem could be solved by making use of the same equivalence, by finding a combinatorial proof of the existence of a common parse word for an arbitrary pair of trees. They made some progress, but not enough to solve the problem in its entirety.

The problem is a language theoretic one. Let G be a context-free grammar with three elements 0, 1 and 2, and the mapping rules $0 \mapsto 12$, $0 \mapsto 21$, $1 \mapsto 02$, $1 \mapsto 20$, $2 \mapsto 01$ or $2 \mapsto 10$. A binary tree $T \in \mathbb{T}_n$ can, then, be labeled according to this scheme, and the final "word" can be read off the bottom leaves in order. This word is said to be **parsed** by the tree T.

Example. An example of a tree parsing the word 2202 is as follows.

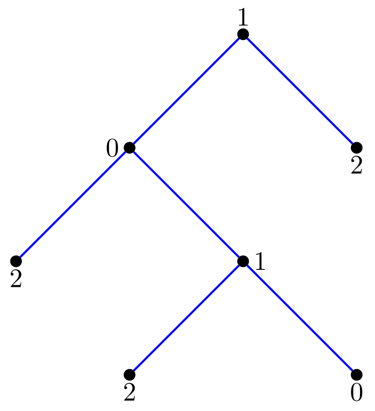


Figure 7.

Now, it is clear that the grammar G is **ambiguous**: a tree can parse two different words, and what is more, two different trees can parse the same word. What is unclear is that G is **totally ambiguous**, i.e. for any two arbitrary binary trees with the same number of leaves, there is a word that they both parse. Kauffman proved this in [7] by showing that the statement is equivalent to the Four Color theorem, while Cooper, Rowland and Zeilberger tried to prove the same statement by other means. It may be formally stated at this point as the main conjecture in this paper, as it was in [3].

Conjecture 6.1. *For any $n \in \mathbb{N}$, let T_1 and T_2 be arbitrary trees in $\mathbb{T}_n$. Then, there exists a word that they both parse, following the rules of the context-free grammar G described above.*

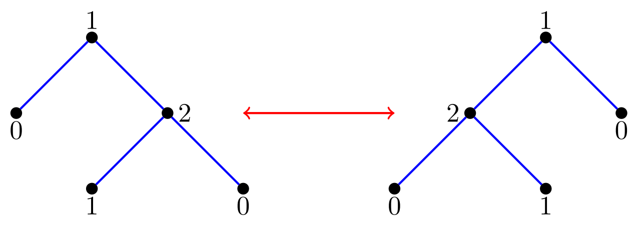


FIGURE 8.

Example. An example of two trees parsing the same word 010 is the following.

In this section we shall compute the precise number of common parse words for any two trees lying in any interval of the right comb poset $\mathscr{C}_n$. In particular, we will prove a special case of Conjecture 6.1.

Note. Whenever we talk about the number of common parse words, we will do so *up to permutations of the alphabet*. This is because whenever we label a tree using the rules of G, we can easily switch two or more of the letters whenever they occur throughout, by permuting the letters of the alphabet. There are six possible such switches between 0, 1 and 2, given by the identity permutation, and the five permutations (12), (23), (13), (123) and (132). We shall consider only distinct labelings that do not depend on permutations; we can think of this as imposing the condition that the root of all the trees under consideration get the label 0, and its left and right children get the labels 1 and 2 respectively. Since there are actually five *other* ways to label these (specifically, the root, the left child and the right child get one of the orders 021, 102, 120, 210 and 201), hence all our results should be multiplied by 6 in order to get the actual number of parse words.

To begin, we need two basic tools.

Proposition 6.2 (Common root property)**.** *If two trees $T_1, T_2 \in \mathbb{T}_n$ parse the same word, then their roots receive the same label, no matter what their internal structure is. Hence, if $T_1, T_2 \in \mathbb{T}_n$ satisfy the property that there is a vertex i in T_1 and a vertex j in T_2 such that the dangling subtrees from these vertices have precisely the same leaves (i.e. both the dangling subtrees contain precisely the leaves m_1 through m_2, for some natural numbers $m_1 < m_2 \leq n$), then, the vertices i and j receive the same label if we label the trees with the same parse word.*

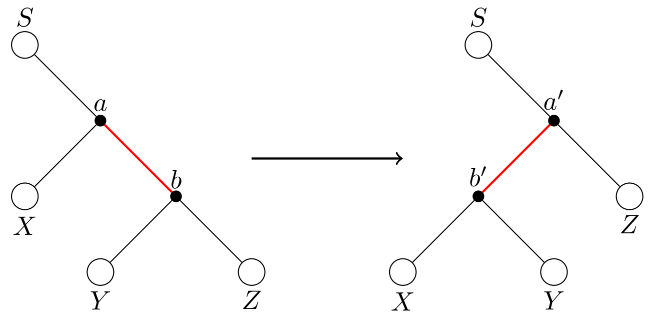
Proof. The first assertion, that two trees parsing the same word receive the same letter for the root, is a well known result, proved in Proposition 2 of [3]. The second assertion can be seen by considering the dangling subtrees as trees in themselves. Since T_1 and T_2 parse the same word in this labeling, in particular, the dangling subtrees parse the same word, because they have precisely the same corresponding leaves, and hence they have a common root by the first assertion. This root is precisely given by the vertex i in the case of T_1, and j in the case of T_2. Hence, i and j receive the same label. □

Proposition 6.3. *For $n \geq 2$, if $T \in \mathbb{T}_n$, then $\left|ParseWords(T)\right| = 2^{n-2}$.*

Proof. This can be seen in the way we construct a tree from its root. At each step, when we pick a vertex with no children, and we add two children to it, we can label the new children in exactly two different ways (following the mapping rules of the grammar G), depending on the label on their parent. We leave the details of the proof as an easy exercise for the reader. □

Theorem 6.4. *For any $n \geq 2$, if T_2 covers T_1 in $\mathscr{C}_n$, then $|ParseWords(T_1, T_2)| = 2^{n-3}$.*

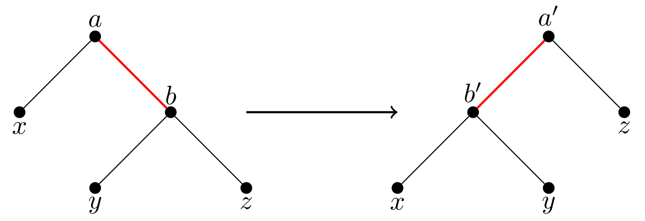
Proof. Consider arbitrary T_1 and T_2 defined as above, with the trees given by:



Here, in T_1, a and b are on the right arm, S is the arbitrary parent tree, and X, Y and Z are the corresponding subtrees. Because of the way tree rotation is defined, note that X, Y and Z will assume the precise positions that they are shown in the figure for T_2. Suppose we label both trees so that the word they parse is the same.

Now, the first observation is that, since X, Y and Z will have the positions shown, any labeling of the two trees with a common parse word will result in a and a' receiving the same label by Proposition 6.2, and therefore, we need not worry about what happens in S at all; the only variation that can possibly arise is entirely within the dangling subtree of the vertex a in T_1 and that of the vertex a' in T_2, and so S may be disregarded entirely. We will show that T_2 will share exactly half the parse words of T_1, thereby proving our result using Proposition 6.3.

Now, because the labeled parse word is the same for both trees, that means in particular that each of the subtrees represented by X, Y and Z receives the same parse word in both the trees (because their order is not changed). Hence, by Proposition 6.2, they each receive the same root in T_1 and T_2. Let's say these roots are labeled respectively x, y and z. Now, the key is to realize that we can forget about what is happening within those subtrees, and only encode them by their roots. This is because the subtrees X, Y and Z preserve their internal structure during the rotation from T_1 to T_2, and so, as long as these subtrees receive the same label for their roots, they are guaranteed of parsing the same word. So, we may forget the X, Y and Z, and simply concentrate on x, y and z now. The problem now reduces to the following figure:



By Proposition 6.2, a and a' must be the same, so we can label them both 0 without loss of generality. Now, in T_1, this means that b is nonzero, say 1, and so, $(y, z) = (0, 2)$ or $(2, 0)$ with equal probability. If b were 2, it would be same for $(0, 1)$ or $(1, 0)$. What this means is that for all labelings of T_1, *exactly* half of them have the same label for a and y, and the other half has different labelings. We will now show that the half with a different labeling for a and y never yields a common parse word, while the half with the same label for a and y *always* does.

Say a and y are different, with $a = 0$, $b = 1$ and $y = 2$. Then, $z = 0$ and $x = 2$, and the parse word in the figure above is 220. This cannot be a parse word for T_2, because x and y must have different labels, as its structure necessitates.

So alternatively, suppose $a = 0$, $b = 1$ and $y = 0$. Then, $z = 2$, $x = 2$ and the parse word is 202. If we label T_2 correspondingly with that, we see that it is indeed a valid labeling that yields $a = 0$ as well, in accordance with what is desired. Therefore, this case yields a common parse word.

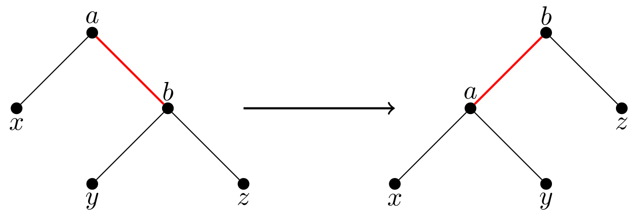
Hence, we see that ParseWords(T_1, T_2) has exactly half as many elements as ParseWords(T_1), which, coupled with Proposition 6.3, yields our desired result. □

Theorem 6.5. *Suppose T_1 and T_2 are comparable in $\mathscr{C}_n$, with the difference in their ranks equal to k. Then, the number of common parse words between T_1 and T_2 (up to permutation) is given by*

2^{n-2-k}. *In fact, since $k \leq n-2$, from Proposition 3.6, any pair T_1 and T_2 of trees comparable in $\mathscr{C}_n$ always has a common parse word.*

Proof. The cases $k = 0$ and $k = 1$ are precisely Proposition 6.3 and Theorem 6.4 respectively. So let's assume $k > 1$.

This proof relies heavily on the proof of Theorem 6.4. The key is to notice that, if T_1 and T_2 are comparable, with T_2 higher in the poset $\mathscr{C}_n$ by k ranks, then there exist trees $T_{1'}, T_{2'}, \dots, T_{(k-1)'}$, all in $\mathscr{C}_n$, such that $T_1 \lessdot T_{1'} \lessdot \dots \lessdot T_{(k-1)'} \lessdot T_2$, where at each step, $T_i \lessdot T_1 \vee T_2$ means that $T_1 \vee T_2$ covers T_i. Then we can apply the same reasoning as in the proof of Theorem 6.4 repeatedly at each covering.



In each case, only half of the labelings in the lower tree will have a common parse word with its cover, as above, and because the condition for that to happen (i.e. for a and y to have the same labeling in the proof of Theorem 6.4) only affects two vertices which *both* pass into a left subtree at each step (and are encoded not by any of the subtree elements but only by their common root a in the figure above, which acts effectively as a leaf henceforth), thereby not affecting the two vertices in the next step. For instance, in Figure 6, the relevant vertices a and y both pass into a left subtree after the rotation, and therefore, from now on, *no* rotation can affect these labels. Hence, because of the highly localized nature of the condition for a common parse word (which arises from the highly localized nature of the potent Proposition 6.2), the relevant vertices will never interact in order to give anything but a nice halving of the number of common parse words. The crucial point to note here is that after each rotation, the remaining vertices on the right arm will be labeled with the same letters as before.

So, Theorem 6.4 will be applicable at each intermediate step, so that we have a nice halving at each step up the poset, and the result follows immediately. □

Corollary 6.6. *If T_1 and T_2 are defined as in Theorem 6.5, then every tree in the interval between T_1 and T_2 in $\mathscr{C}_n$ will parse all the common parse words of T_1 and T_2.*

Proof. This follows from the same localized property of the common roots that was exploited in proving Theorem 6.5. We leave the details as an exercise. □

Example. Corollary 6.6 is very easy to check for any specific case, as illustrated by this example. Consider the example shown below. Then Corollary 6.6 can be checked as follows: take any parse word of the first and the last one tree, say 01202. Now, by numbering the leaves in this order in *all* the intermediate trees, we can label the internal vertices (by "backward" reasoning) until we come to the root (which will be 1 for all the trees in this example). In every case, we will be able to obtain a labeling of the internal vertices following the rules of G.

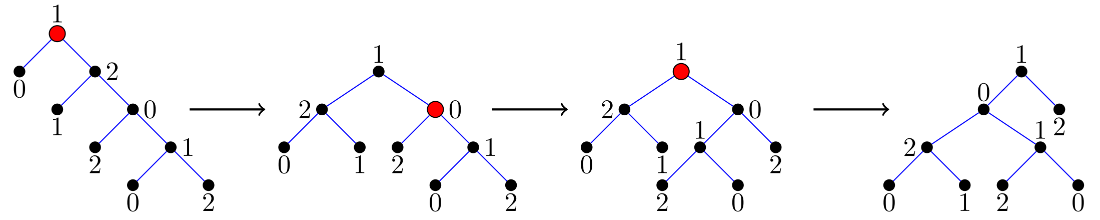


FIGURE 9.

Theorem 6.7. *Fix any two trees T_1 and T_2 in $\mathscr{C}_n$ which lie in some interval (or in other words, which have a well-defined join). Then, the set of common parse words between T_1 and T_2 is precisely the set of common parse words of their join and meet (which are both well-defined from Corollary 2.9).*

Proof. If T_1 and T_2 are comparable in $\mathscr{C}_n$, then the result is trivially true, because the join and meet are the trees themselves, in some order. So consider only the case where they are not comparable, but in fact lie in the same interval; in other words, they have a join that does not equal one of them.

Now they are guaranteed from Corollary 2.9 to have a join $T_1 \vee T_2$ and a meet $T_1 \wedge T_2$. One direction of the proof is immediate, because along with $T_1 \vee T_2$ and $T_1 \wedge T_2$ (which are comparable by definition), T_1 and T_2 separately satisfy the hypotheses of Corollary 6.6, and therefore, any common parse word of $T_1 \vee T_2$ and $T_1 \wedge T_2$ is also parsed by T_1 as well as T_2. We will now show that any word that is parsed by T_1 and T_2 is also, in fact, parsed by $T_1 \wedge T_2$. The same result will hold for $T_1 \vee T_2$ by a symmetric argument, and this will be enough for our desired proof.

We will make extensive use of Corollary 2.9. To do so, consider $RP_{T_1 \wedge T_2}$, and suppose it has k elements (including the leaf n, which is one of the elements immediately from the definition).

Now, each of the internal parenthesizations of these k elements *must* also be present in the reduced parenthesizations of T_1, T_2 and $T_1 \vee T_2$ from Corollary 2.9. Hence, the subtrees represented by these parenthesizations will be present in all four trees, and therefore, we may choose to ignore the internal parenthesizations of these k elements. They are all encoded by their roots, and if we can obtain the same label for these roots, then we can label each vertex of each such subtree in the same way (because their structure is the same in all four of T_1, T_2, $T_1 \wedge T_2$ and $T_1 \vee T_2$). Hence, we only consider the roots of these k elements, which encode all internal parenthesizations within them; these roots act as new "leaves" in some sense. We will, therefore, refer to these roots a_1 through a_k as "leaves" henceforth, and using Proposition 2.12, assume $T_1 \wedge T_2$ to be RightCombTree(k).

Now, RP_{T_1} and RP_{T_2} must have completely disjoint parenthesis pairs; if not, this means that there exists some parenthesis pair J in RP_{T_1} and some pair J' in RP_{T_2} such that they enclose the same factor (which may be the right factor in one and the left one in the other). But then, by Proposition 3.1 forces $J = J'$, and so, this common parenthesis pair J is also present in $RP_{T_1 \wedge T_2}$ by definition, which implies that the number of elements of $RP_{T_1 \wedge T_2}$ is strictly less than k, contradicting the assumption that $RP_{T_1 \wedge T_2}$ has k elements. We will use this fact (that the parenthesis pairs of RP_{T_1} and RP_{T_2} are disjoint) as the basis of our proof.

Take any common parse word $w = w_1 w_2 \dots w_k$ of T_1 and T_2 (recall that from our simplification, all trees are now effectively k-leaf trees), and label both T_1 and T_2 completely with the word w, including all internal vertices.

We will now construct the same parse word for $T_1 \wedge T_2$, starting with the rightmost letter w_k. Now, w_k labels the leaf n, which is unbracketed in RP_{T_1} as well as RP_{T_2}. So, we may label this leaf with the letter it is labeled with in T_1 and T_2. Now, take a look at the next leaf, w_{k-1}. Either this w_{k-1} will be unbracketed in RP_{T_1} as well as RP_{T_2}, in which case, we may continue labeling these leaves as before. Otherwise, w_{k-1} will be bracketed in exactly one of RP_{T_1} and RP_{T_2}, say RP_{T_1}, and in particular, will correspond to some element containing some bracketing of the letters $w_j w_{j+1} \dots w_{k-1}$, for some $j \leq k-2$. The crucial observation here is that *all* these letters $w_j, \dots, w_{k-1}$, will be unbracketed in RP_{T_2}, from Corollary 2.9 and the discussion above. Hence, the subtree containing the leaves $a_j, \dots, a_k$ in RP_{T_2} will have the *exact same internal structure* in T_2 and $T_1 \wedge T_2$. So, we can label this entire subtree of $T_1 \wedge T_2$ in the same way as T_2. Once we do that, we have already labeled the last $k-j+1$ leaves in $T_1 \wedge T_2$. Now consider all three trees T_1, T_2 and $T_1 \wedge T_2$ with this entire subtree (the subtree containing leaves w_j through w_k) encoded by its common root as a dangling "new" leaf. By the potent Proposition 6.2, this "new" leaf will receive the same label for all three trees. Now we can keep doing the exact same thing by considering this new leaf as the new rightmost element (corresponding to a_j), and apply the process above recursively. At each step of this process, we can either choose both trees (if none of them bracket the next leaf), or exactly one of the trees. Either way, we can continue recursively and label $T_1 \wedge T_2$ with the same parse word w.

We proved above that RP_{T_1} and RP_{T_2}, under the assumptions above, have disjoint parenthesis pairs. This fact, coupled with Corollary 2.9, guarantees that $RP_{T_1 \vee T_2}$ is the *disjoint* union of RP_{T_1}

and RP_{T_2}. We can now label $T_1 \vee T_2$ with w, by the same process described above, thereby concluding the proof. This proof is best understood by an example, which we give below. $\square$

Example. This example will demonstrate the construction described in the proof of Theorem 6.7. Suppose we have the situation of Figure 10 below, with $RP_{T_1} = (1(23))4567$, $RP_{T_2} = 1234(56)7$, $RP_{T_1 \wedge T_2} = 1234567$, and $RP_{T_1 \vee T_2} = (1(23))4(56)7$. We will illustrate the proof of Theorem 6.7 above by labeling the join $T_1 \vee T_2$ with a common parse word. We take any parse word common to T_1 and T_2 in this example, and label $T_1 \vee T_2$ with that same parse word. The construction for $T_1 \wedge T_2$ will be analogous.

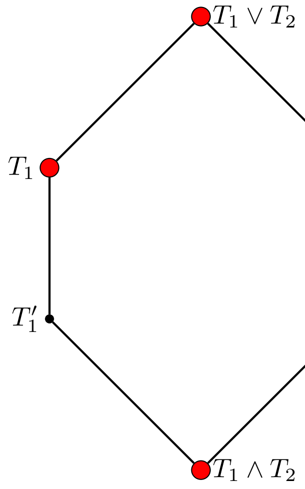
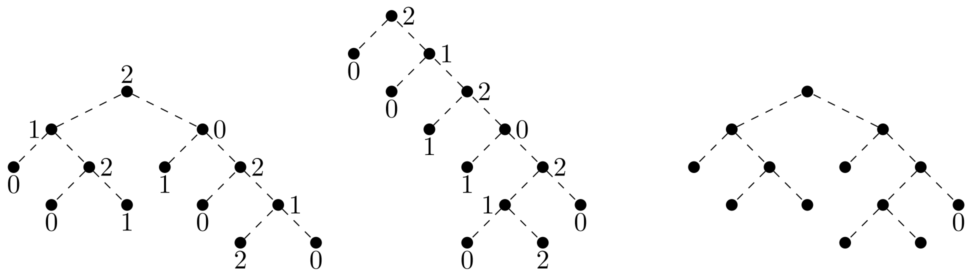


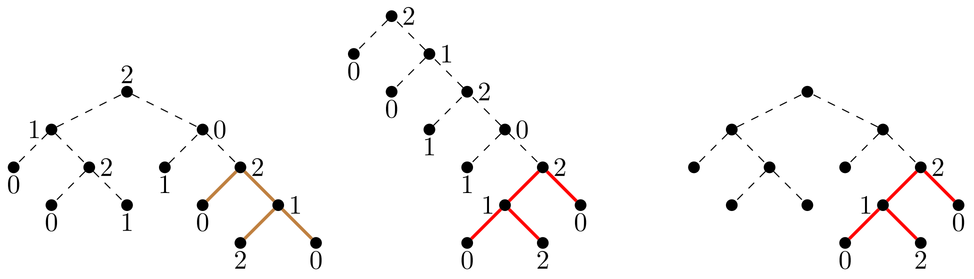
FIGURE 10.

Consider T_1 and T_2 in $\mathscr{C}_n$, with their reduced parenthesizations as given above. Furthermore, Corollary 2.8 tells us the ranks of T_1 and T_2 in terms of parenthesis pairs, meaning that there are trees T_1' and T_2' in the places shown. $RP_{T_1 \wedge T_2}$, RP_{T_1}, RP_{T_2} and $RP_{T_1 \vee T_2}$ have 7, 5, 6 and 4 elements respectively (we will call each of the 7 elements of $T_1 \wedge T_2$ "leaves" in this example just as in the proof of Theorem 6.7, as the internal labelings of the subtrees they represent will be the same throughout; this corresponds to $k = 7$ in the proof of Theorem 6.7). For ease of understanding of the proof, draw all trees with black dotted lines for now.

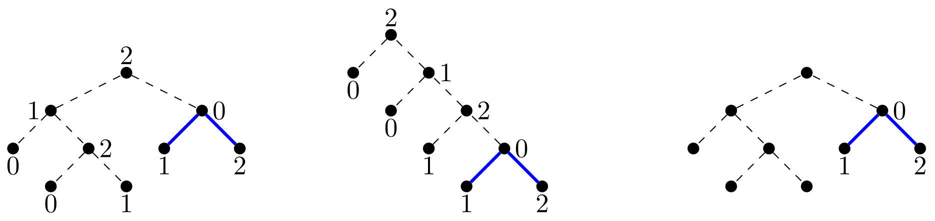
We start with an arbitrary parse word for T_1 and T_2, say 0011020, and then label leaf 7 in $T_1 \vee T_2$ with 0.



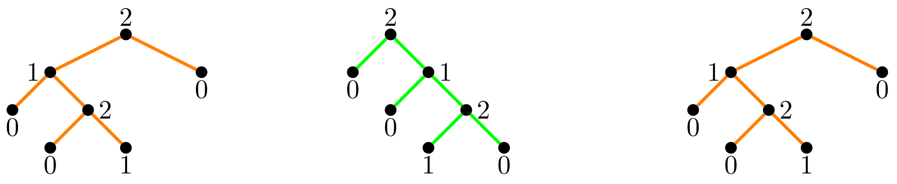
Then, the subtree in $T_1 \vee T_2$ containing the leaves 5, 6 and 7 has exactly the same structure as the corresponding subtree in T_2, so we color this corresponding subtree red in both T_2 and $T_1 \vee T_2$, and label this subtree of $T_1 \vee T_2$ throughout in the exact way as T_2, as shown. The subtree containing the corresponding leaves in T_1 is shown in brown. Then, from here on, we can disregard the subtree containing leaves 5, 6 and 7 in *all* three trees, because this subtree has already received the same parse word, and it can now be encoded by their common root (which is guaranteed to receive the same label 2 for all three trees, from Proposition 6.2). This common root will now act as the rightmost leaf in the next step. Let's call this leaf 567 from now.



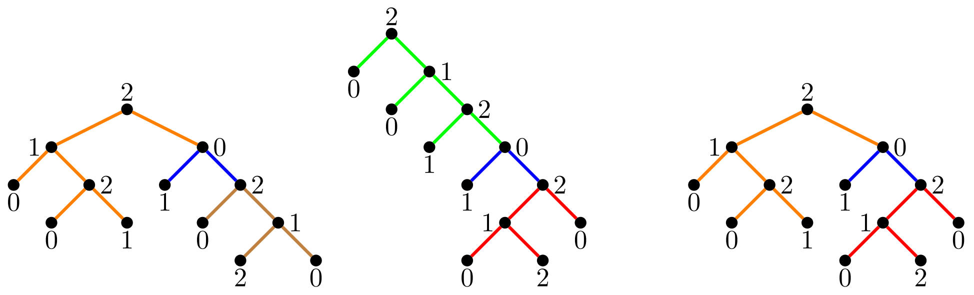
Leaf 4 (which is unparenthesized in both RP_{T_1} and RP_{T_2}), is labeled in T_1 and T_2 in the same way, so the subtree containing the leaves 4 and 567 has the same structure in all three trees. So we color all three of these subtrees blue and label it in $T_1 \vee T_2$ exactly in the same way as either of T_1 or T_2. Again, we disregard the internal structure of this subtree from here on, because Proposition 6.2 ensures that the root, which now encodes the whole subtree, receives the same label 0 for all three trees. This leaf henceforth acts as the new rightmost leaf; call this leaf 4567.



Finally, the subtree containing leaves 1, 2, 3, and 4567 is the same in T_1 and $T_1 \vee T_2$. So, we can color this subtree orange in both T_1 and $T_1 \vee T_2$, as shown, and label it in $T_1 \vee T_2$ in the same way as it is labeled in T_1, to finish off the labeling. The corresponding subtree in T_2 is shown in green, and in particular, once again, from Proposition 6.2, they receive the same root 2 (which is now effectively the root of the whole tree $T_1 \vee T_2$).



We can finally attach all the colored subtrees again to check our result. Note that attaching the subtrees back correctly recovers T_1, T_2 and $T_1 \vee T_2$, and the labels remain intact because of the potent Proposition 6.2. Given a common parse word of T_1 and T_2, this potent common root property has been used in the proof of Theorem 6.7 in order to construct the same parse word for $T_1 \vee T_2$ by following the algorithm described.



An exactly analogous proof is applicable for the meet $T_1 \wedge T_2$, which can be labeled similarly, with 7 common to T_1 and T_2, then 5, 6 and 7 common to T_1, then 4 and 567 common to both T_1 and T_2, and finally, 1, 2, 3 and 4567 common to T_2. We leave that construction as an easy exercise.

Remark. Theorem 6.7 is potent because it gives the exact number of parse words for any two trees in an interval in $\mathscr{C}_n$. In particular, if T_1 and T_2 are defined as in Theorem 6.7, and if $T_1 \wedge T_2$ and $T_1 \vee T_2$ are respectively the meet and join of T_1 and T_2, then we have

$$\left|\text{ParseWords}(T_1, T_2)\right| = \left|\text{ParseWords}(T_1 \wedge T_2, T_1 \vee T_2)\right|,$$

where both sides are nonzero, from Theorem 6.5, as $T_1 \wedge T_2$ and $T_1 \vee T_2$ are comparable in $\mathscr{C}_n$.

The discussions above reveal astonishing insights into the structure of the comb poset $\mathscr{C}_n$ and the behavior of the ParseWords function within an interval of $\mathscr{C}_n$ (in particular, Conjecture 6.1 is an immediate consequence of Theorem 6.7 if the pair of trees under consideration lies within an interval in $\mathscr{C}_n$). However, our results so far do not reveal any clear insight into the corresponding behavior for two trees that do *not* both lie in some interval in $\mathscr{C}_n$. Unfortunately, this is the generalization that is required if we are to prove Conjecture 6.1 in full generality.

Even though we did not find nice expressions relating the parse words common to an arbitrary pair of trees in different intervals, we shall salvage some hopes for complete generalization in Section 6 by a different approach.

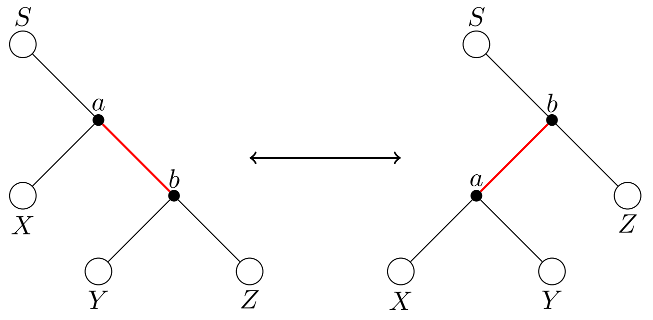
7. Towards Complete Generalization: the Rotation Graph

In this final section we shall attempt to probe a little deeper into the behavior of the ParseWords function in the neighborhood of any fixed tree in $\mathscr{R}_n$. This generalization adds back some extra edges of $\mathscr{R}_n$ which were not present in $\mathscr{C}_n$ (and we also do not lose any of the edges), which is a distinct advantage. However, we also have to give up the beautiful and rigid structure of $\mathscr{C}_n$ now, and the behavior of the ParseWords functions deviates from the completely predictable behavior within an interval.

Proposition 7.1. *For $n \geq 3$, suppose $T \in \mathbb{T}_n$, and T' is obtained from T by precisely one rotation in any direction on any vertex. Then,*

$$\left|\mathit{ParseWords}(T, T')\right| = 2^{n-3}.$$

Proof. The proof follows the same idea as the proof of Theorem 6.4. Consider an arbitrary rotation of the form shown, that takes the tree T_1 to the tree T_2. As before, S represents the arbitrary parent, X, Y and Z are arbitrary subtrees, and a and b are vertices.



We shall prove the result only for the forward direction, i.e. for a right rotation; the argument for a left rotation will be symmetric. Furthermore, it is important to note that, unlike a rotation of the form described in Proposition 2.13, the center of rotation a does not have to lie on the right arm of the tree T_1. This will not affect the proof, but it is an important distinction worth keeping in mind. Once again, we can encode everything inside X, Y and Z by their respective common roots, because the existence of common parse words depends only on the preservation of the "word" labeling the roots of X, Y and Z in order. Furthermore, similar to before, no variation can be introduced in the arbitrary parent S, and hence the only possible change will occur *below* S. With these two observations, the figure reduces precisely to the second figure in the proof of Theorem 6.4, and the exact same proof becomes applicable after this point. The proof for a left rotation is analogous. □

Proposition 7.2. *Suppose for some tree $T \in \mathbb{T}_n$, we perform two rotations, the first one on vertex i and the second one on vertex j. Suppose we reach the tree T'' by the two rotations. Then, $ParseWords(T, T'') = 2^{n-4}$ for each of the following scenarios:*

(1) *The rotation on i leaves the internal structure of the dangling subtree from j unchanged.*
(2) *The rotation on j leaves the internal structure of the dangling subtree from i unchanged (note that this dangling subtree may have changed during the first rotation on i; this assumption is that it does not change from this intermediate step to the final one).*

Proof. For the first part, denote by V_j the subtree dangling from the vertex j which is unchanged during the rotation on i. Now, it is clear that a rotation (right or left) of the form shown in Proposition 2.13 changes the structures of precisely the dangling subtrees from a and b shown in that diagram. So, the assumption implies that the vertex j must be in one of the subtrees X, Y or Z. We leave it as an exercise for the reader to show that the two "halvings" represented by two applications of Proposition 7.1 for the rotations on i and j will give us a quartering of the number of common parse words, as desired.

For the second part, note that "halving" holds for the first rotation, from Proposition 7.1. Then, by an argument similar to the one above, it follows that after this first rotation, i must lie within one of the subtrees X, Y and Z for the second rotation, on j. So, for this rotation, because the dangling subtree does not change, we can encode the entire subtree by just the root i. It is now easy to see how the quartering still follows. □

Theorem 7.3. *For any $n \geq 3$, fix any tree $T \in \mathbb{T}_n$ in the rotation graph $\mathscr{R}_n$. Consider the single-variable function*

$$f(T') = \left|ParseWords(T, T')\right|$$

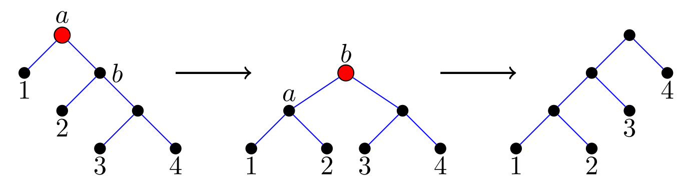
on $T' \in \mathbb{T}_n$. Then, $f(T') = 2^{n-2-i}$ for $i \in \{0, 1, 2\}$, where i represents the distance along the edges of $\mathscr{R}_n$ between T' and T.

Proof. The cases $i = 0$ and $i = 1$ are Proposition 6.3 and Theorem 7.1 respectively. To prove the case $i = 2$, we will show that the "halving" property utilized in the proofs of Theorem 6.5 and Proposition 7.1 holds for one more rotation as well.

If the dangling subtrees of the two centers of rotation are disjoint, then Proposition 7.2 applies, and we get a quartering, as desired. So consider only the case when the center for one rotation lies in the subtree dangling from the center of the other rotation. Suppose the centers are a and b, with b lying in the subtree dangling from a. Denote the subtrees dangling (before any of the rotations are performed) from a and b by V_a and V_b respectively. In particular, V_b is wholly contained in V_a.

If the vertices a and b have two or more edges of the tree T separating them, then any rotation (right or left) on a will leave the dangling subtree from b intact, and so Proposition 7.2 applies again, and we get the desired quartering of the number of common parse words again.

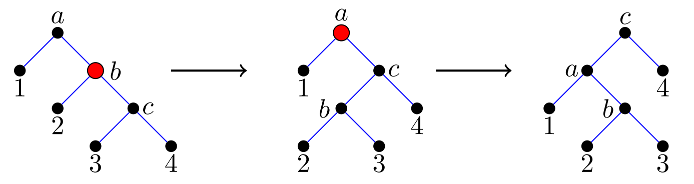
We now have the case where a is a parent of b in T. Without loss of generality take b to be a *right* child of a. Suppose we rotate on a first. Then, if the rotation on a is a *left* rotation, then Proposition 7.2 is applicable *again*. So it only remains to consider the case where the rotation on a is a right rotation. Now, a has become a *left* child of b in the process of this rotation (again, see the figure with Proposition 2.13). If we now perform a *left* rotation on b, we just get back the original tree, in contradiction of our hypothesis. So, if we rotate on a first, the only nontrivial case is when both the rotations are right rotations. This results in the figure below.

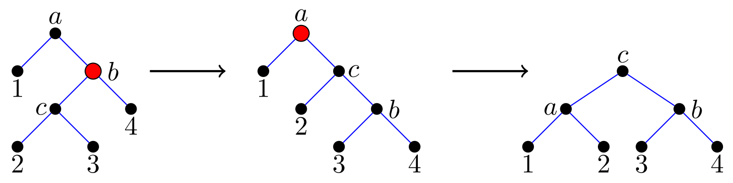


As in all proofs, we leave out the common arbitrary parent above, because that does not get affected, and encode the dangling subtrees by their roots, which are numbered as leaves here. The specified sequence of two rotations takes us from the first tree to the third tree above. So, we need to find a common parse word for the first and the last tree. But these two are effectively the right comb tree and the left comb tree of order 4. Hence, by Proposition 3.6 and Theorem 6.5, the two trees share precisely *one* common parse word, up to permutations. But the number of ways of labeling the original tree T_1 is $2^2 = 4$, from Proposition 6.3. Hence it follows that precisely a quarter of the possible labelings of the first tree above yield a common parse word for the third tree as well. This completely proves the case where we perform the rotation on the vertex a first.

The only nontrivial case left is where the first rotation is on b. Now, the fact that we assumed without loss of generality that b is a right child of a necessitates a crucial observation: if the (second) rotation on a is a *left* rotation, then Proposition 7.2 is applicable once again. So, we need only consider the case where the rotation on a is a right rotation. Once again, we leave out the common arbitrary parent and encode the subtrees by their common roots.

Because we only consider a right rotation on a, there are two cases, both depending on the rotation on b. The two cases are respectively when the rotation on b is a right rotation, and when it is a left rotation. The two cases can be represented by the two following figures (encoding all subtrees by their roots, and without showing the arbitrary parent whose labeling is unaffected):





In both cases, note that the subtree dangling from b is unchanged during the second rotation. So then, Proposition 7.2 applies to both these cases as well, and we obtain our quartering result as before.

The only case left to consider is when a and b are, in fact, the same, or in other words, we perform two rotations on the same vertex. For this case, notice that, because we can without loss of generality suppose that the first rotation is a left rotation. The only two cases left, then, are when the second rotation is also a left rotation, or when it is a right rotation. Now, refer to the same figures above, and look at the *reverse* sequences of rotations, which take the last trees shown above to the first ones. In particular, notice that these two rotations are both on the vertex c, and correspond precisely to the two cases mentioned above. The quartering property clearly still holds, because our previous result stated that the number of parse words common to the first and last trees is exactly one fourth of the number parsed by just the first tree, which is also the number parsed by the last tree from Proposition 6.3. So, a symmetric argument proves the final case.

So, if $T' \in \mathbb{T}_n$ is obtained from some $T \in \mathbb{T}_n$ by two rotations, then $\left|\text{ParseWords}(T, T')\right| = \frac{1}{4}\left|\text{ParseWords}(T, T)\right|$, which coupled with Proposition 7.1 yields our desired result in its entirety. $\square$

Corollary 7.4. *Suppose T, T' and T'' are elements of $\mathscr{R}_n$ satisfying the hypotheses of Theorem 7.3, such that T' is at a distance 1 from T and T'' is at a distance 2 from T along the same path, i.e. T'' is at a distance 1 from T'. Then, the common parse words shared by T and T'' are, in fact, parsed by T' as well.*

Proof. If the centers of the two rotations are sufficiently far apart, then the result is readily seen to be true, from the same arguments as in the proof of Theorem 7.3. All the nontrivial cases if the centers *do* interact have been drawn as separate figures in the proof of Theorem 7.3, which all indicate the intermediate tree T'. The easiest way to get our desired result is to label the leaves of the intermediate trees in all these figures with a common parse word of its two adjacent trees (the precise form of which can be easily found), and label the internal vertices by "backward" reasoning to check that it is, in fact, a valid parse word for these intermediate trees as well. We leave this as a straightforward exercise. $\square$

Conjecture 7.5. *The property in Theorem 7.3 above is stronger, in the sense that it holds for the case $i = 3$ as well.*

Corollary 7.6. *For $n \geq 5$, a result analogous to Corollary 7.4 holds for a radius of 3. Stated precisely, if T, T', T'' and T''' are defined in the obvious way (keeping in mind that they lie on the same path in the rotation graph $\mathscr{R}_n$ starting from T and have distances 0, 1, 2 and 3 from T respectively), then*

$$\left|ParseWords(T, T', T'', T''')\right| = \frac{1}{8}\left|ParseWords(T)\right| = 2^{n-5}.$$

In particular, $|ParseWords(T, T''')| \geq 2^{n-5}$.

Proof. The proof of Corollary 7.6 is long and depends on a lot of casework similar to the proof of Theorem 7.3. We omit the long and arduous proof here. □

Remark. Interestingly, this nice and predictable behavior of the ParseWords function evident in all the results above (Proposition 7.1, Theorem 7.3, Corollaries 7.4, 7.6, and Conjecture 7.5) breaks down for distances greater than 3.

In spite of that, it seems from our observations that the value of the function always jumps *up* when it does not "halve", i.e. when the halving does not hold, we get a higher number of common parse words than half the previous value. This is the main motivation behind the next extremely important conjecture, which is actually stronger than Conjecture 6.1. However, it seems a more *tangible* conjecture with possibly an easier proof than the previously stated one, so we will state it formally below.

Conjecture 7.7. *Let T, T', T'' be elements of $\mathscr{R}_n$ such that there is an edge connecting T' and T'', and*

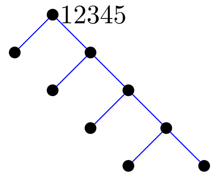
$$d_{\mathscr{R}_n}(T, T'') = d_{\mathscr{R}_n}(T, T') + 1.$$

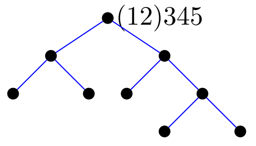
Then,

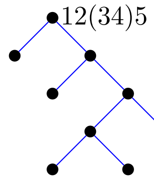
$$\left|ParseWords(T, T'')\right| \geq \frac{1}{2}\left|ParseWords(T, T')\right|.$$

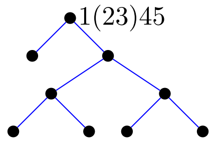
Appendix 1

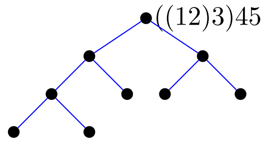
Below we indicate all the elements of $\mathscr{C}_5$, along with their reduced parenthesizations.

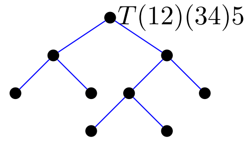


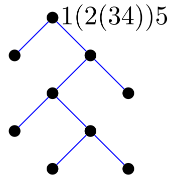


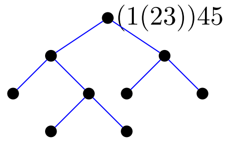


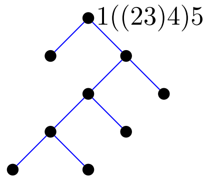


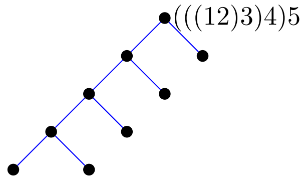


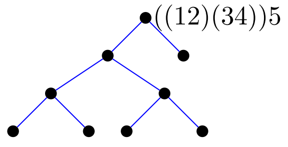


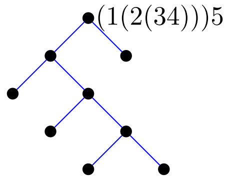


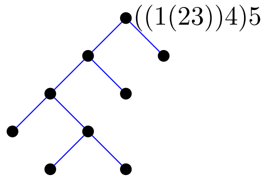


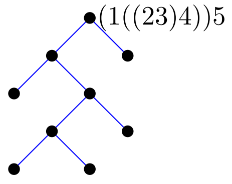












Appendix 2

Below we write out the reduced parenthesizations for all trees in $\mathscr{C}_6$ in the form of a table, and label them with the numbers 0 through 41. This labeling is for clarity in the figure that follows, which depicts the Hasse diagram of the comb poset of order 6, labeled with the appropriate number from the table to avoid unnecessary clutter.

Table of trees in Figure 11		
Zeroth Rank	0	123456
First Rank	1	1(23)456
	2	123(45)6
	3	(12)3456
	4	12(34)56
Second Rank	5	1((23)4)56
	6	(1(23))456
	7	1(23)(45)6
	8	12(3(45))6
	9	(12)3(45)6
	10	((12)3)456
	11	(12)(34)56
	12	12((34)5)6
	13	1(2(34))56
Third Rank	14	(1((23)4))56
	15	1(((23)4)5)6
	16	((1(23))4)56
	17	(1(23))(45)6
	18	1((23)(45))6
	19	1(2(3(45)))6
	20	(12)(3(45))6
	21	((12)3)(45)6
	22	(((12)3)4)56
	23	((12)(34))56
	24	(12)((34)5)6
	25	1(2((34)5))6
	26	(1(2(34)))56
	27	1((2(34))5)6
Fourth Rank	28	((1((23)4))5)6
	29	(1(((23)4)5))6
	30	(((1(23))4)5)6
	31	((1(23))(45))6
	32	(1((23)(45)))6
	33	(1(2(3(45))))6
	34	((12)(3(45)))6
	35	(((12)3)(45))6
	36	((((12)3)4)5)6
	37	(((12)(34))5)6
	38	((12)((34)5))6
	39	(1(2((34)5)))6
	40	((1(2(34)))5)6
	41	(1((2(34))5))6

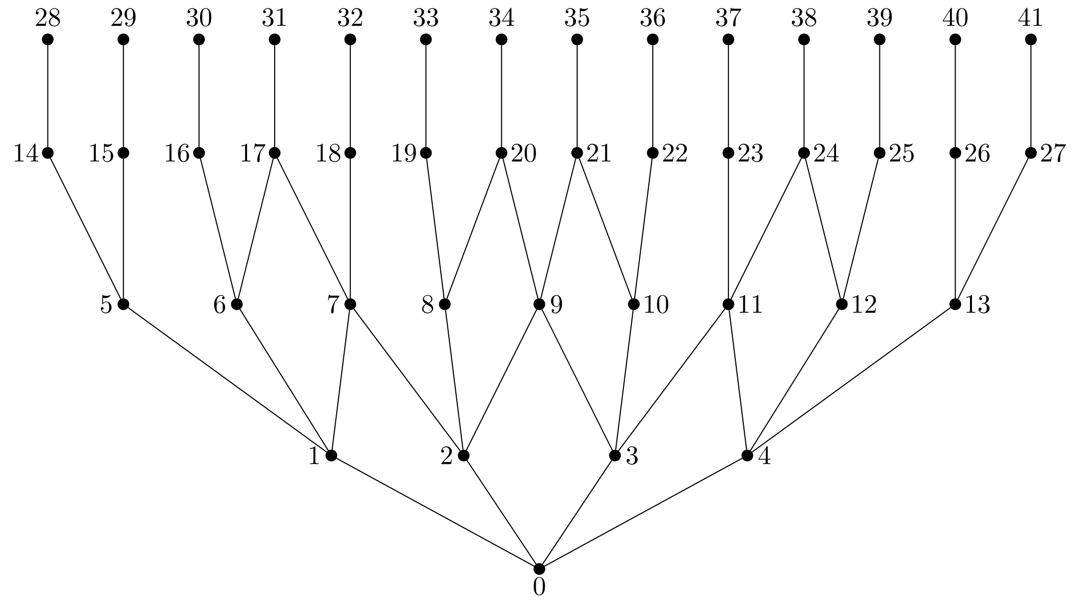


FIGURE 11. The Hasse diagram of $\mathscr{C}_6$

ACKNOWLEDGMENTS

We wish to thank Victor Reiner and Dennis Stanton for their invaluable guidance and support throughout the process of writing this paper. This paper was researched and written during the REU Program 2010 at the School of Mathematics, University of Minnesota - Twin Cities, mentored by Reiner and Stanton and funded by NSF grant DMS-1001933. We would also like to specially thank Alex Csar and Nathan Williams for their incredibly helpful words of advice and continual assessment of our progress, and Alan Guo for his conclusive input on the proof of Theorem 3.8.

REFERENCES

[1] K. Appel and W. Haken, *Every planar map is four colorable I: discharging*, Illinois Journal of Mathematics, **21** (1977), 429-490.

[2] K. Appel and W. Haken, *Every planar map is four colorable II: reducibility*, Illinois Journal of Mathematics, **21** (1977), 491-567.

[3] B. Cooper, E. S. Rowland and D. Zeilberger, *Toward a Language Theoretic Proof of the Four Color Theorem*, `arXiv:1006.1324v1`, preprint (2010).

[4] P. H. Edelman, *Tableaux and Chains in a New Partial Order of* $\mathfrak{S}_n$, Journal of Combinatorial Theory Series A, **51** (1989), 181-204.

[5] P. Flajolet and M. Noy, *Analytic Combinatorics of Non-Crossing Configurations*, Discrete Mathematics, **204**(1-3) (1999), 203-229.

[6] S. Huang and D. Tamari, *Problems of Associativity: A Simple Proof for the Lattice Property of Systems Ordered by a Semi-associative Law*, Journal of Combinatorial Theory Series A, **13** (1972), 7-13.

[7] L. Kauffman, *Map coloring and the vector cross product*, Journal of Combinatorial Theory Series A, **48** (1990), 145-154.

[8] J. M. Pallo, *Right-arm Rotation Distance between Binary Trees*, Information Processing Letters, **87** (2003), 173-177.

[9] J. M. Pallo, *Enumerating, Ranking and Unranking Binary Trees*, The Computer Journal, **29** (1986), 171-175.

[10] D. D. Sleator, R. E. Tarjan and W. P. Thurston, *Rotation Distance, Triangulations, and Hyperbolic Geometry*, Annual ACM Symposium on Theory of Computing, Berkeley, California (1986), 122-135.

[11] R. P. Stanley, *Enumerative Combinatorics volume 1*, Cambridge University Press, Cambridge (1999), 24.

[12] D. Zeilberger, *A Bijection from Ordered Trees to Binary Trees that sends the Pruning Order to the Strahler Number*, Discrete Mathematics, **82** (1990), 89-92.

Department of Mathematics, Princeton University, NJ 08543, USA.
E-mail address: rsengupt@princeton.edu

Department of Mathematics, Massachusetts Institute of Technology, MA 02139, USA.
E-mail address: warutsuk@mit.edu