Abstract

Determining whether two graphs are isomorphic is a very important and difficult problem in graph theory, with applications in image recognition. One way to make progress towards this problem is by finding graph invariants that distinguish large classes of graphs. In 1995, Richard Stanley conjectured that his chromatic symmetric function distinguishes all trees, which has remained unresolved. In 2017, Takahiro Hasebe and Shuhei Tsujie proved a version of Stanley’s conjecture for posets: their strict order quasisymmetric function distinguishes all rooted trees. However, the strict order quasisymmetric function has an infinite number of terms, and it’s not clear which terms need to be computed to distinguish two rooted trees. To resolve this, we use an original combinatorial framework to devise a procedure that explicitly reconstructs a rooted tree from its strict order quasisymmetric function. This procedure takes us one step closer to a concrete implementation of the strict order quasisymmetric function. In addition, it provides an alternate, more concrete proof that the strict order quasisymmetric function distinguishes rooted trees.
1 Introduction

Determining whether two graphs are isomorphic is a very important and difficult problem in graph theory. For instance, in the field of computer vision, graphs can be used to encode visual information, and knowing whether two graphs are isomorphic is crucial for recognizing patterns [TD17]. To better understand when two graphs could be isomorphic, graph invariants are a useful tool. They are defined as follows:

A graph polynomial \( p_G \) assigns every graph \( G \) a polynomial \( p_G \). It is called a graph invariant if it is true that for any two graphs \( G \) and \( H \), \( G \cong H \implies p_G \equiv p_H \). It is useful to compute graph invariants because if two graphs have different values for a graph invariant, then they cannot be isomorphic. Additionally, a graph invariant is perfect if the converse is also true: \( p_G \equiv p_H \implies G \cong H \). Computing perfect graph invariants is even more useful: if two graphs have the same value for a perfect graph invariant, then they must be isomorphic. If a graph invariant is perfect, we say that it distinguishes all graphs.

Perhaps the most well-known graph invariant is the chromatic polynomial \( \chi_G(n) \), which is the unique polynomial such that \( \chi_G(n) \) is the number of ways to properly color \( G \) with \( n \) colors. The chromatic polynomial is not that useful for distinguishing two graphs, however, because there are many examples of pairs of graphs with the same chromatic polynomial. In particular, all trees with a fixed number of vertices have the same chromatic polynomial.

Richard P. Stanley defined a generalization of the chromatic polynomial, called the chromatic symmetric function \( X_G(x) \) [Sta95]. The chromatic symmetric function, rather than being a function of one variable, is actually a function of infinitely many variables (\( x \) represents \( (x_1, x_2, \ldots) \)). It is defined as follows:

\[
X_G(x) = \sum_{f:V(G)\to \mathbb{N}} x_f,
\]

where \( x_f = \prod_{v \in V(G)} x_{f(v)} \).

Because the chromatic symmetric function has infinitely many variables, it’s no surprise that the chromatic symmetric function is in general much better than the chromatic polynomial at telling apart graphs. However, two graphs have already been found that have the same chromatic symmetric function. Stanley posed a pivotal question about a more restricted class of graphs: does the chromatic symmetric function distinguish all trees? This problem remains unsolved to this day and is actively being researched [Hur20; HJ18].

Hasebe and Tsuji define an analogue of Stanley’s chromatic symmetric function for a poset \( P \), which they call the strict order quasisymmetric function \( \Gamma^<(P; x) \) [HT17]. It is defined as follows:

\[
\Gamma^<(P; x) = \sum_{f:V(P)\to \mathbb{N}} x_f.
\]

They then prove with algebraic methods that this function distinguishes all rooted trees, considered as posets. Furthermore, Tsuji uses a similar method to prove more results about the chromatic symmetric function [Tsu18]. Thus, [HT17] is a significant step in the direction of Stanley’s question.

However, in the interest of application, we must consider that the strict order quasisymmetric function has an infinite number of terms. We might wish to computationally check whether two rooted trees \( T_1 \) and \( T_2 \) are isomorphic by checking their strict order quasisymmetric polynomials \( \Gamma^<(T_1; x) \) and \( \Gamma^<(T_2; x) \) against each other, but a computer cannot directly check an infinite number of terms. Our best bet is to sample specific terms from each polynomial and compare them (details in Definition 15). In order for our method to be practical, we want a way to sample terms that guarantees an answer in a finite number of queries. Because Hasebe and Tsuji’s method relies on unique factorization, it does not provide such a way to sample terms.

Our approach is to find a procedure that, by sampling terms from \( \Gamma^<(T; x) \), can reconstruct the rooted tree \( T \). Then, we run this procedure on both \( \Gamma^<(T_1; x) \) and \( \Gamma^<(T_2; x) \) in parallel. We stop the reconstruction as soon as the rooted trees differ, or if the reconstruction is completed with both rooted trees identical, we are guaranteed that the rooted trees are isomorphic. As long as the reconstruction always completes in a finite number of queries, running two reconstructions in parallel will also complete in a finite number of queries.

Cai, Slettnes, and the author take a step towards constructing such an procedure by analyzing
the strict order quasisymmetric function combinatorially [CSZ20]. They use a construction that they term “introducing gaps,” which they apply to a base coloring. By querying the strict order quasisymmetric function for the colorings that result from the construction, they are able to obtain partial information about the tree: specifically, what they call the tree’s \textit{coheight profile}.

In our paper, we set up a completely new framework for their “introducing gaps” idea, which allows us to recursively extend their idea in a precise manner and to an arbitrary degree. The details of this extension are quite involved, but through a careful combinatorial argument, we are able to obtain complete information about the rooted tree in a finite number of queries.

\textbf{Theorem 14.} From knowledge of $\Gamma(T; x)$ (as defined in Definition 15), we can reconstruct $T$.

Thus, our extended procedure indeed reconstructs a rooted tree $T$ from its strict order quasisymmetric function $\Gamma(T; x)$. With knowledge of two strict order quasisymmetric functions $\Gamma(T_1; x)$ and $\Gamma(T_2; x)$, we can now determine whether the rooted trees $T_1$ and $T_2$ are isomorphic. This brings us one step closer to a concrete implementation of the strict order quasisymmetric function.

Our procedure also provides an alternate, more concrete proof that the strict order quasisymmetric function distinguishes rooted trees. Analysis of the strict order quasisymmetric function (also known as the $P$-partition generating function in other papers) has been done in terms of its expansion in the monomial basis $M_\alpha$ [HT17], the fundamental basis $L_\alpha$ [IW20], and the power sum basis $\phi_\alpha$ [IW19], but it has not been done using the terms themselves.

In Section 2 of this paper, we go over definitions and notations. Then, in Section 3, we provide an example of our procedure in action, which we believe is useful for understanding the rest of the paper in its full formality. In Section 4, we set up the framework for our extended procedure, and in Section 5, we formally prove our main result. Finally, in Section 8, we state some future directions for this project.

\section{Background and notation}

We begin by going over definitions and notations. Some are taken from [HT17] and [CSZ20], though importantly, we change the profile notation from [CSZ20] to make it easier to work with. Note that throughout this paper, $0 \in \mathbb{N}$, and we denote multisets with double curly braces: $\{\}$. In addition, we define for $S \subseteq S'$ the indicator function $\mathbbm{1}_S : S' \rightarrow \{0, 1\}$ such that $\mathbbm{1}_S(v) = 1$ if $v \in S$ and 0 otherwise.

\subsection{Tree-statistics}

\textbf{Definition 1.} A \textbf{rooted tree} $T$ is a directed tree so that every vertex has indegree 1 except for one vertex $v_T$, called the \textbf{root}.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{tree.png}
\caption{Here a tree is depicted.}
\end{figure}

\textbf{Definition 2.} For a vertex $v \in V(T)$, the \textbf{subtree induced by} $v$, denoted $S_v$, is the subtree of $T$ whose set of vertices consists of all descendants of $v$ (including $v$ itself) and whose set of arcs consists of all arcs between descendants of $v$.

\textbf{Definition 3.} A \textbf{tree-statistic} is a function $a : V(T) \rightarrow A$ for some set $A$. We write $a_v$ for the application of $a$ to $v$. 
**Definition 4.** In this paper, the main tree-statistic that we consider is the coheight \( h : V(T) \to \mathbb{N} \), where \( h_v \) is defined as the length of the unique path from \( v \) to \( v_T \).

**Definition 5.** We say that a tree \( T \) has \( n \) layers if the maximum coheight of any vertex in \( T \) is \( n - 1 \).

### 2.2 Profiles

To better understand profiles, we need two preliminary definitions:

**Definition 6.** For a set of indeterminates \( \{x_i\}_{i \in A} \) indexed by a set \( A \), we let \( \langle x_i \rangle_{i \in A} \) be the multiplicative group generated by \( \{x_i\}_{i \in A} \).

**Definition 7.** Let \( a : V(T) \to A \) be a tree-statistic. For an element \( n \in A \), let \( a^{-1}(n) \) (read: \( a \) inverse of \( n \)) be the set of vertices \( v \in V(T) \) such that \( a_v = a \).

**Definition 8.** Let \( a : V(T) \to A \) be a tree-statistic, and let \( \{x_i\}_{i \in A} \) be a set of indeterminates indexed by \( A \). The \( a \) profile, denoted \( x_a \), is the element of \( \langle x_i \rangle_{i \in A} \) defined by (the two items on the right hand side are equal by definition):

\[
x_a = \prod_{v \in V(T)} x_{a_v} = \prod_{n \in A} x_n^{\left| a^{-1}(n) \right|}.
\]

**Definition 9.** Let \( a \) and \( \{x_i\} \) be as above, and let \( v \in V(T) \). The \( a \) profile of \( v \), denoted \( x_a|_v \), is the element of \( \langle x_i \rangle_{i \in A} \) defined by (the two items on the right hand side are equal by definition):

\[
x_a|_v = \prod_{u \in V(S_v)} x_{a_u} = \prod_{n \in A} x_n^{\left| a^{-1}(n) \cap V(S_v) \right|}.
\]

For example, we could talk about the **coheight profile** \( x_h \) or, given a vertex \( v \), the **coheight profile of** \( v \) \( x_h|_v \).

Profiles can also be considered tree-statistics: given a tree-statistic \( a : V(T) \to A \), then we can let \( x_a : V(T) \to \langle x_i \rangle_{i \in A} \) be the tree-statistic such that the application of \( x_a \) to \( v \) is \( x_a|_v \). Thus, we can nest profiles. For instance, we could consider the **coheight profile profile** \( x_{x_h} \).

**Example 1.** The coheight profile profile \( x_{x_h} \) of the tree depicted in Figure 1 is

\[
x_{x_{x_0}} x_{x_1} x_{x_2} x_{x_3} x_{x_4} x_{x_5} + x_{x_1} x_{x_2} x_{x_3} x_{x_4} x_{x_5} x_{x_6} + x_{x_1} x_{x_2} x_{x_3} x_{x_4} x_{x_5} x_{x_6} x_{x_7} + \ldots
\]

Note that in [CSZ20], profiles were defined as multisets and denoted \( P_a^v \) and \( P_a^v \), which we have replaced with \( x_a \) and \( x_a|_v \), respectively. The definitions contain the same information.

### 2.3 Working with profiles

**Definition 10.** Let \( A \) be a well-ordered set, and let \( \{x_i\}_{i \in A} \) be a sequence of indeterminates indexed by \( A \). We can impose a well-order on the set \( \langle x_i \rangle_{i \in A} \) by considering each term \( \prod_{i \in A} x_i^{e_i} \) (\( e_i \in \mathbb{N} \)) as the tuple \( (e_i)_{i \in A} \) and ordering them lexicographically.

To review, the tuple \( (a_i)_{i \in A} \) is **lexicographically less than** the tuple \( (b_i)_{i \in A} \) if there exists some \( i \) for which all \( j < i \), \( a_j = b_j \), and \( a_i < b_i \). For example, \((1, 1, 1) < (1, 1, 2) < (1, 2, 1)\).

**Definition 11.** Let \( A \) and \( \{x_i\} \) be as above. For some polynomial \( p \in \mathbb{Z}[x_i]_{i \in A} \), we let \( \text{max}_n^m(p) \) be the \( m \)th greatest term of \( p \), including duplicate terms. For instance, \( \text{max}_2^2(2x_1 + 2x_2) = x_1 \).

Similarly, we let \( \text{min}_n^m(p) \) be the \( m \)th least term of \( p \).

In this paper, we often use polynomials that collect together a set of co-height profiles. For instance, let’s fix an \( n \in \mathbb{N} \) and construct the polynomial

\[
p = \sum_{v \in V(T), h_v = n} x_{h}|_v.
\]

Since co-height profiles are terms of the form \( \prod_{i \in \mathbb{N}} x_i^{e_i} \), we can talk about the term \( \text{min}(p) \). In words, this term is the least co-height profile out of all the co-height profiles of the vertices with co-height \( n \).
Definition 12. Let $A, (x_i), p$ be as above. For some $n \in A$, we let $\prod_{i \leq n} x_i^{e_i}$ be the polynomial consisting of the terms $\prod_{i \in A} x_i^{e_i}$ in $p$ such that $e_i = e_i'$ for all $i \leq n$. For example, if $p = x_2 + 2x_1x_2 + 3x_1^2x_2$, then $[x_1]p = 2x_1x_2$ and $[x_2]p = x_2$ (since we require that the exponent of $x_1$ is 0).

2.4 The strict order quasisymmetric function

Definition 13. A coloring of a rooted tree $T$ is a function $f : V(T) \to \mathbb{N}$, where $V(T)$ denotes the vertex set of $T$. $f$ is increasing if for every arc $(u, v)$ in $T$, we have $f(u) < f(v)$.

Notice that a coloring $f$ can also be considered a tree-statistic with a slight abuse of notation: $f_u = f(v)$. Thus, we can consider the $f$ profile $x_f$: see Figure 2 for an example.

Definition 14. The strict order quasisymmetric function of a rooted tree $T$ is the polynomial

$$\Gamma^< (T; x) = \sum_{f: V(T) \to \mathbb{N}} x_f.$$ 

Figure 2: Here a coloring $f$ is depicted. Note that $x_f = x_1^1x_2^3x_3^4x_4^9x_5^2$.

2.5 Having knowledge of $\Gamma^< (T; x)$

Because the strict order quasisymmetric function has infinitely many terms, in order to work with it practically, we need a way of sampling and working with only a finite number of terms. Thus, we formally introduce the notion of “having knowledge of $\Gamma^< (T; x)$.”

Definition 15. To have knowledge of $\Gamma^< (T; x)$ means to have knowledge of a certain sampling function $F : S \to \Gamma^< (T; x) \cup \{\emptyset\}$, where $S$ is some set and we consider the polynomial $\Gamma^< (T; x)$ as a set of terms. Essentially, by selecting a certain input, we can obtain either a certain term of the strict order quasisymmetric function or the empty set. This allows us to work with a finite number of terms at a time while still having access to them all.

For this paper, we pick the sampling function $F : \langle(x_i)\rangle \in \mathbb{N} \to \Gamma^< (T; x) \cup \{\emptyset\}$ that sends the element $\prod_{i \in \mathbb{N}} x_i^{e_i} \in \langle(x_i)\rangle$ to the term

$$\max \left( \prod_{i \in \mathbb{N}} x_i^{e_i} \right) \Gamma^< (T; x)$$

(which can also be $\emptyset$). The reason for this will become clear in the following sections.

3 Example

Before proceeding with the formal framework of this paper, we give an example of the procedure in action. We believe that viewing this paper as a generalization of this example is beneficial to its comprehension.
We begin with knowledge of the strict order quasisymmetric function $\Gamma^<(T; x)$ of the tree $T$ depicted in Figure 3.

Some of the terms of $\Gamma^<(T; x)$ are the following:

$$\Gamma^<(T; x) = x_1^1 x_2^4 x_3^6 + 6x_1^1 x_2^4 x_3^5 x_4^1 + 6x_1^1 x_2^4 x_3^5 x_4^0 x_5^1 + \cdots + 15x_1^1 x_2^4 x_3^4 x_4^2 + \ldots$$

We wish to reconstruct the tree $T$.

### 3.1 Step 1

The first step of the reconstruction is to determine the term with the lexicographically greatest tuple of exponents. This is equivalent to finding the terms with the greatest exponent of $x_1$, and out of those the ones with the greatest exponent of $x_2$, and so on until one term has been singled out.

We write this formally as $\max(\Gamma^<(T; x))$. From our knowledge of $\Gamma^<(T; x)$, we can determine that $\max(\Gamma^<(T; x)) = x_1^1 x_2^4 x_3^6$. Figure 4 depicts the coloring of $T$ to which this term corresponds.

$$\Gamma^<(T; x) = x_1^1 x_2^4 x_3^6$$

We show in Theorem 4 that from the term $x_1^1 x_2^4 x_3^6$, we can now obtain the following information: the root of $T$ has 4 children and 6 grandchildren. Figure 5 summarizes what we now know about $T$.

### 3.2 Step 2

The second step of the reconstruction begins by determining the term with the lexicographically greatest tuple of exponents under a certain condition on the exponents. Specifically, considering that the term that we obtained from the first step is $x_1^1 x_2^4 x_3^6$, we impose the condition that the exponent of $x_1$ is 1 and the exponent of $x_2$ is $4 - 1 = 3$.

We write this formally as $\max([x_1^1 x_2^3] \Gamma^<(T; x))$. From our knowledge of $\Gamma^<(T; x)$, we can determine that $\max([x_1^1 x_2^3] \Gamma^<(T; x)) = 2x_1^1 x_2^3 x_3^6 x_4^1$. Figure 6 depicts one of the colorings of $T$ to which this term corresponds.
One can think of this coloring as being similar to the coloring in Figure 4, except that the \( x_2^3 \) condition forces there to be a "gap" in the lime color that is instead filled with a blue color. Let us call the vertex at which this "gap" occurs \( g \). We show in Theorem 6 that by comparing the term \( x_1^1 x_2^3 x_3^6 x_4^1 \) to the term \( x_1^1 x_2^4 x_3^6 \) from before, we can determine that \( g \) has 1 child. This is obvious in this case, but information about \( g \)’s children, grandchildren, etc. can also be deduced in larger cases. Figure 7 summarizes what we now know about \( T \).

\[\text{Figure 7: The second stage of the reconstruction.}\]

### 3.3 Step 2, part 2

We continue the second step of the reconstruction by determining \( \max([x_{1}^{1} x_{2}^{2}] \Gamma^< (T; x)) \), using our notation from before, which turns out to be \( x_{1}^{1} x_{2}^{2} x_{3}^{6} x_{4}^{2} \). Figure 8 depicts the coloring of \( T \) to which this term corresponds.

\[\text{Figure 8: The coloring of } T \text{ corresponding to the term } x_{1}^{1} x_{2}^{2} x_{3}^{6} x_{4}^{2}.\]

Let us call the second gap \( g' \). By Theorem 6 again, we can compare \( x_{1}^{1} x_{2}^{2} x_{3}^{6} x_{4}^{2} \) to \( x_{1}^{1} x_{2}^{4} x_{3}^{6} \) to determine that \( g \) and \( g' \) have a total of 2 children, and thus that \( g' \) has one child. Figure 9 summarizes what we now know about \( T \).

\[\text{Figure 9: The second stage of the reconstruction, continued.}\]

We can continue this process, determining \( \max([x_{1}^{1} x_{2}^{1}] \Gamma^< (T; x)) \) and \( \max([x_{1}^{1} x_{2}^{0}] \Gamma^< (T; x)) \) in order to obtain the number of children of the other lime vertices. After this, we have reconstructed the entire tree.

If \( T \) is a tree with four or more layers, stages three and up of the reconstruction are analogous. For example, suppose that one of the terms we obtain in stage two is \( x_{1}^{1} x_{2}^{3} x_{3}^{5} x_{4}^{2} \). Considering this, in stage three, we might impose the condition \([x_{1}^{1} x_{2}^{3} x_{3}^{5}] \). Complications do arise, but the details have been left out for the purposes of this example.
4 Framework

What follows is the formal framework for our main result, which significantly extends the method in [CSZ20]. We provide notes explaining how our notation connects to the example in Section 3 (e.g. in Section 3, we see the term $\max(\Gamma^<(T; x)) = x_1^1 x_2^1 x_3^6$; our formal notation for this term is $x_{f_3}$).

Cai, Slettnes, and the author [CSZ20] introduced the special colorings $f_{\emptyset}$, $f_n$, and $f_{m,n}$. They showed that all three of the terms $x_{f_{\emptyset}}, x_{f_n}, x_{f_{m,n}}$ can be obtained from knowledge of $\Gamma^<(T; x)$. Then, they showed that from these three terms, one can obtain the tree’s coheight profile $x_{S'}$. We begin by stating their definitions of $f_{\emptyset}$, $f_n$, and $f_{m,n}$, and then we demonstrate our generalized construction.

**Definition 16.** Let $f_{\emptyset}$ be the coloring such that $f_{\emptyset}(v) = 1 + h_v$. See Figure 10 for an example.

**Definition 17.** Let $f_n$ be the coloring such that $f_n(v) = 1 + h_v + \mathbb{1}_{V(S_1)}(v)$, where $g_1$ is the vertex with coheight $n - 1$ with the least coheight profile.

**Definition 18.** Let $f_{m,n}$ be the coloring such that $f_{m,n}(v) = 1 + h_v + \sum_{1 \leq i \leq m} \mathbb{1}_{V(S_i)}(v)$, where $g_i$ is the vertex with coheight $n - 1$ with the $i$th least coheight profile.

In Section 3, the colorings $f_{\emptyset}, f_2,$ and $f_{2,2}$ are represented in Figures 4, 6, and 8, respectively, corresponding to the terms $x_{f_{\emptyset}} = x_1^1 x_2^1 x_3^6$, $x_{f_2} = x_1^1 x_2^3 x_3^6 x_4^1$, and $x_{f_{2,2}} = x_1^1 x_2^2 x_3^6 x_4^2$.

![Figure 10: Here the coloring $f_{\emptyset}$ is depicted. Note that $x_{f_{\emptyset}} = x_1^1 x_2^4 x_3^6 x_4^8$.](image)

We generalize this construction by defining a coloring $f_S$ from any multiset of vertices $S$.

**Definition 19.** For a multiset of vertices $S$, let $f_S$ be the coloring such that $f_S(v) = 1 + h_v + \sum_{g \in S} \mathbb{1}_{V(S_g)}(v)$.

This coloring coincides with:

- $f_{\emptyset}$ when $S = \emptyset$
- $f_n$ when $S = \{g_1\}$ as defined in Definition 17
- $f_{m,n}$ when $S = \{g_i | 1 \leq i \leq m\}$ as defined in Definition 18.

We claim that this construction is general enough to encompass every increasing coloring.

**Theorem 1.** For every increasing coloring $f : V(T) \to \mathbb{N}$, there exists a unique multiset of vertices $S$ such that $f = f_S$. See Figure 11 for an example.

**Proof.** We can consider $S$ as how much $f$ “deviates” from $f_{\emptyset}$. In $f_{\emptyset}$, moving along any edge increases the color by one. If in $f$ moving along an edge increases the color by more than one, then putting a vertex in $S$ accounts for the difference.

Formally, we find the unique $S$ by using recursion on coheight $h_v$. Begin by considering $h_v = 0$, which includes only the root $v_T$. Since we have

$$f(v_T) = f_S(v_T) = 1 + h_{v_T} + \sum_{g \in S} \mathbb{1}_{V(S_g)}(v_T)$$

$$= 1 + \sum_{g \in S} \mathbb{1}_{v_T}$$
we must have that the root \( v_T \) appears \( f(v_T) - 1 \) times in \( S \). This works since \( f(v_T) - 1 \geq 0 \).

Suppose that for \( h_v < n \), if the parent of \( v \) is \( p \), then we have that \( v \) must appear \( f(v) - f(p) - 1 \) times in \( S \). This works since \( f(v) - f(p) - 1 \geq 0 \), which is true because \( f \) is increasing. (If \( v = v_T \), then we set \( f(p) = 0 \).)

Now, let’s find for \( h_v = n \) how many times \( v \) must appear in \( S \). Let the vertices on the path from \( v_T \) to \( v \) be \( p_0 = v_T, p_1, \ldots, p_{h_v - 1}, p_{h_v} = v \), where \( p_i \) is a vertex at coheight \( i \). (For convenience, let \( f(p_{-1}) = 0 \).) Then

\[
f(v) = f_S(v) = 1 + h_v + \sum_{g \in S} \mathbb{1}_{V(S_g)}(v)
\]

\[
= 1 + h_v + \sum_{g \in S} \sum_{v \in S_g} 1
\]

\[
= 1 + h_v + \sum_{0 \leq i \leq h_v - 1} (f(p_i) - f(p_{i-1}) - 1) + \sum_{g \in S} 1
\]

\[
= 1 + \sum_{0 \leq i \leq h_v - 1} (f(p_i) - f(p_{i-1})) + \sum_{g \in S} 1
\]

\[
= 1 + f(p_{h_v}) + \sum_{g \in S} 1.
\]

Thus \( v \) must appear \( f(v) - f(p) - 1 \) times in \( S \).

The next step in [CSZ20] is to show that \( x_{fs}, x_{fn}, x_{fm} \) can be obtained from \( \Gamma^<(T; x) \). A logical extension would be to show that general \( x_{fs} \) can be obtained from \( \Gamma^<(T; x) \). However, this is actually not true; resolving this is complicated, so we defer this to Section 5.

The final step in [CSZ20] is to show that from \( x_{fs}, x_{fn}, x_{fm} \), one can obtain \( x_{k_h} \). The method to do this involves looking at the differences between the three terms. We wish to generalize this to being able to obtain \( x_{(x)h} \) for any \( x \in \mathbb{N} \), which is our notation for \( x_{k_h} \) nested \( x \) times. For example, \( x_{(2)h} = x_{k_h} \).

To work towards this goal, we want to look at the difference between \( x_{fs} \) and \( x_{g} \) for similar \( S \) and \( S' \). We can do this by expressing \( x_{fs} \) in terms of \( x_{g} \) and \( x_{k_h} \) for \( g \in S \) (Theorem 3). This sets up a structure for the rest of our work, which we defer to Section 5.

The following theorem establishes the recursive step for Theorem 3 by expressing \( x_{f_{g'}} \), where \( S' = S \cup \{g\} \), in terms of \( x_{fs} \) and \( x_{k_h} \).

**Theorem 2.** Consider a multiset of vertices \( S' \). Pick a \( g' \in S' \) such that \( g' \) has no descendants in \( S' \), and let \( S = S' \setminus \{g'\} \). We define \( h' \) to be the number of vertices in \( S \) “above” \( g' \): formally \( h' = \sum_{g \in S} \mathbb{1}_{V(S_g)}(g') \). We claim that

\[
x_{f_{g'}} = x_{fs} \prod_{v \in V(S_{g'})} \frac{\mathbb{1}_{V(S_g)}(v) + 2 + h'}{\mathbb{1}_{V(S_g)}(v) + 1 + h'}.
\]

**Proof.** The idea is that upon adding \( g' \), the color of every vertex below \( g' \) is shifted up by one. \( 1 + h' + h_v \) is the color of \( v \) before the addition, and \( 2 + h' + h_v \) is the color of \( v \) after the addition.

Figure 11: The coloring \( f \) from Figure 2 is equal to \( f_S \) for \( S \) that consists of the circled vertices.
Here is the formal proof. We know that \( f_{S'}(v) = f_S(v) + \mathbb{1}_{V(S'_g)}(v) \), so:

\[
x_{f_{S'}} = x_{f_S} \prod_{v \in V(S'_g)} \frac{x_{f_{S'}(v)}}{x_{f_S(v)}} = x_{f_S} \prod_{v \in V(S'_g)} \frac{x_{f_{S'}(v)}+1}{x_{f_S(v)}}.
\]

We claim that the numerator and denominator of this expression are equal to the numerator and denominator of the desired expression. It is sufficient to prove that for every vertex \( v \in V(S'_g) \), it is true that \( f_S(v) = 1 + h' + h_v \).

We know that \( g' \) has no descendants in \( S \). Thus, for all \( g \in S, S'_g \) is either completely inside or completely outside \( S'_g \). This means that for all \( v \in V(S'_g) \), it is true that \( \mathbb{1}_{V(S'_g)}(v) = \mathbb{1}_{V(S'_g)}(g') \), allowing us to write

\[
f_S(v) = 1 + h_u + \sum_{g \in S} \mathbb{1}_{V(S'_g)}(v) \\
= 1 + h_u + \sum_{g \in S} \mathbb{1}_{V(S'_g)}(g') \\
= 1 + h_u + h'.
\]

\[\square\]

To clean up the notation, we define the \textit{shift function} \( \sigma \) and the \textit{shift difference function} \( \tau \):

**Definition 20.** Let the \textit{shift function} \( \sigma : \mathbb{Z}[x_i]_{i \in \mathbb{N}} \rightarrow \mathbb{Z}[x_i]_{i \in \mathbb{N}} \) be the function that takes \( x_i \) to \( x_{i+1} \) for all \( i \in \mathbb{N} \). More explicitly, we set

\[
\sigma \left( k \prod_{i \in \mathbb{N}} x_i^{x_i} \right) = k \prod_{i \in \mathbb{N}} x_i^{x_i+1}.
\]

We denote \( s \in \mathbb{N} \) repeated applications of \( \sigma \) by \( \sigma^s \).

**Definition 21.** Let the \textit{shift difference function} \( \tau : \langle x_i \rangle_{i \in \mathbb{N}} \rightarrow \langle x_i \rangle_{i \in \mathbb{N}} \) be the function that takes \( x_i \) to \( \frac{x_i-1}{x_i} \) for all \( i \in \mathbb{N} \). Said another way, \( \tau(x) = \frac{x}{x-1} \).

Now, we can rewrite Theorem 2 as follows:

\[
x_{f_{S'}} = x_{f_S} \prod_{v \in V(S'_g)} \frac{x_{2+h'+h_v}}{x_{1+h'+h_v}} = x_{f_S} \cdot \sigma(\tau(\sigma^{h'(x_h|g')})).
\]

In order to turn this inductive step into a full expression for \( x_{f_S} \), we need to encode the dependence of \( h' \) on \( S \) and \( g' \) into its notation. We do this by defining the \textit{elevation function} of \( S \):

**Definition 22.** Given a multiset of vertices \( S \), the \textit{elevation function} of \( S \) is the function \( h_S : S \rightarrow \mathbb{N} \) that maps \( g \in S \) to the number of vertices in \( S \setminus \{g\} \) “above” \( g \): formally,

\[
h_S(g) = \sum_{g' \in S \setminus \{g\}} \mathbb{1}_{V(S'_g)}(g).
\]

If duplicates of the same vertex appear in \( S \), give them an arbitrary linear order so that they take consecutive values under \( h_S \). For example, if three duplicates of the root appear in \( S \), then they take the values 0, 1, 2 under \( h_S \), and a child of the root would take the value 3.

Since \( h' \) is defined as the number of vertices in \( S'_g \setminus \{g\} \) “above” \( g \), Theorem 2’s final form is as follows:

\[
x_{f_{S'}} = x_{f_S} \cdot \sigma(\tau(\sigma^{h'(x_h|g')})) = x_{f_S} \cdot \sigma(\tau(\sigma^{h'_2}(x_h|g'))).
\]

Finally, we have the theorem that expresses \( x_{f_S} \) in terms of \( x_h \) and \( x_h|g \) for \( g \in S \).

**Theorem 3.** For \( S \) a multiset of vertices, we have:

\[
x_{f_S} = \sigma \left( x_h \tau \left( \prod_{g \in S} \sigma^{h_S(g)}(x_h|g) \right) \right).
\]
Proof. We present a recursive proof.

- For \( S = \varnothing \), we have by definition that \( x_{f_S} = \sigma(x_h) \).

- Suppose that we already know that the desired is true for all \( |S| = n \). Now, consider a multiset of vertices \( S' \) such that \( |S'| = n + 1 \). Pick a \( g' \in S' \) such that \( g' \) has no descendants in \( S' \), and let \( S = S' \setminus g' \). By Theorem 2, we have the following. (We use here the fact that \( \sigma \) and \( \tau \) are multiplicative.)

\[
x_{f_{S'}} = x_{f_S} \cdot \sigma(\tau(\sigma^{h_{S'}}(g')(x_{h|g'})))
\]

\[
= \sigma \left( x_h \tau \left( \prod_{g \in S} \sigma^{h_S(g)}(x_{h|g}) \right) \right) \cdot \sigma(\tau(\sigma^{h_{S'}}(g')(x_{h|g'})))
\]

\[
= \sigma \left( x_h \tau \left( \prod_{g \in S'} \sigma^{h_S(g)}(x_{h|g}) \right) \right).
\]

\( \square \)

Example 2. Consider the coloring from Figure 11. First of all, note that no vertices of \( S \) are “above” any other, so \( h_S(g) \) is always zero. Then Theorem 3 says the following. (It’s instructive to split up \( \tau \) to emphasize the shift that every individual gap produces. Also note that the exponents of \( x_{h|g} \) don’t have to all be 1; they just happen to be so in this example.)

\[
x_{f_S} = \sigma \left( x_h \prod_{g \in S} \tau(x_{h|g}) \right)
\]

\[
= \sigma \left( x_0^1 x_1 x_2^6 x_3^8 \cdot \tau(x_1 x_2 x_3) \cdot \tau(x_2) \cdot \tau(x_3) \right)
\]

\[
= \sigma \left( x_0^1 x_1 x_2^6 x_3^8 \cdot \frac{x_2 x_3 x_4}{x_1 x_2 x_3} \cdot \frac{x_2}{x_2 x_3} \right)
\]

\[
= \sigma \left( x_0^1 x_1^3 x_2^4 x_3^9 x_4^2 \right)
\]

\[
= x_1^1 x_2^3 x_3^4 x_4^9 x_5^2,
\]

as expected.

5 Main result

Our ultimate goal (Theorem 14) is to obtain \( T \) from knowledge of \( \Gamma^<(T; x) \). Recall (Definition 15) that knowledge of \( \Gamma^<(T; x) \) is knowledge of the sampling function that sends \( \prod_{i \in N} x_i^{e_i} \) to

\[
\max \left( \prod_{i \in N} x_i^{e_i} \right) \Gamma^<(T; x).
\]

Our stepping stones to Theorem 14 involve obtaining \( x_{(x)^h} \) (\( x \_ x \_ x \_ x \) nested \( x \) times). This requires recursive action: first obtaining \( x_h \) (Theorem 4), then \( x_{x_h} \) (Theorem 5), \( x_{(3)x} \) (Theorem 7), and so on (Theorem 13). Theorems 4 and 6 are equivalent to the results in [CSZ20], but our notation simplifies the proofs considerably. In addition, understanding our proofs of Theorems 4 and 6 is instructive for our proofs of Theorems 7 and 13.

Theorem 4. From knowledge of \( \Gamma^<(T; x) \), we can obtain \( x_h \).

Proof. Note that the below is a generalization of Section 3.1.

We obtain \( x_h \) through evaluating \( \max(\Gamma^<(T; x)) \) (the sampling function at \( \prod_{i \in N} x_i^{0_i} = 1 \)).

In short, the maximum \( x_{f_S} \) is achieved with \( S = \varnothing \). See Figure 10 for a depiction of \( f_S \). We know that \( f_S(v) = 1 + h_v \), so to find the number of vertices with coheight \( n \), we need only check how many vertices are colored \( 1 + n \) in \( f_S \).
To show this more formally, we apply Theorem 3 and move $\max$ into the expression. Notice that $\sigma$ preserves the ordering of the elements of $(x_i)_{i \in \mathbb{N}}$, while $\tau$ flips it; hence, $\max$ becomes $\min$.

$$\max(\Gamma^<(T; x)) = \max_S (x_{f_S})$$

$$= \max_S \left( \sigma \left( x_h \tau \left( \prod_{g \in S} \sigma^{h_S(g)}(x_h|_g) \right) \right) \right)$$

$$= \sigma \left( x_h \tau \left( \min_S \left( \prod_{g \in S} \sigma^{h_S(g)}(x_h|_g) \right) \right) \right)$$

$$= \sigma(x_h \tau(1))$$

$$= \sigma(x_h).$$

$\sigma$ is invertible, so we have successfully obtained $x_h$. \hfill \Box

To get more information out of the strict order quasisymmetric function, we need to exploit the sampling function more generally. Our method primarily involves repeated applications of the following expression:

**Theorem 5.** From knowledge of $\Gamma^<(T; x)$, we can obtain any expression of the form

$$\min_S \left( \prod_{i \leq n} x_i^{e_i} \prod_{g \in S} \sigma^{h_S(g)}(x_h|_g) \right).$$

(We treat the expression inside the brackets as a condition: the minimum is taken over all $S$ that produce a nonempty expression inside $\min_S$.)

Being able to obtain this expression is the basis of the rest of this paper. The idea is that with carefully chosen values of $\prod_{i \leq n} x_i^{e_i}$, we can methodically obtain the information that we want. For example, in Theorem 6, we set $\prod_{i \leq n} x_i^{e_i} = x_n$ in order to force $S$ to include one vertex with coheight $n$. Then, taking the minimum helps us get rid of anything extra, so we’re left with $x_h|_g$.

We need a quick definition for the proof.

**Definition 23.** For $n \in \mathbb{N}$, let the **truncate function** $\phi_n : \mathbb{Z}[x_i]_{i \in \mathbb{N}} \to \mathbb{Z}[x_i]_{i \in \mathbb{N}}$ be the function that gets rid of all $x_i$ for $i > n$. Explicitly, $\phi_n$ is defined by

$$\phi_n \left( k \prod_{i \in \mathbb{N}} x_i^{e_i} \right) = k \prod_{i \leq n} x_i^{e_i}.$$

**Proof of Theorem 5.** Since we know $x_h$ by Theorem 4, we can obtain the following from the sampling function. This expression might look scary, but it will become the expression that we want.

$$\max \left( \sigma \left( \phi_n(x_h) \prod_{i \leq n} x_i^{e_i - e_i} \right) \right) \Gamma^<(T; x).$$

To gain some intuition for what this expression means, let’s consider the specific case of $\prod_{i \leq n} x_i^{e_i} = x_n$. Note that the below is a generalization of Section 3.2.

In this case, the above expression is

$$\max \left( \left[ \sigma \left( \phi_n(x_h)x_n^{-1} \right) \right] \Gamma^<(T; x) \right),$$

and we claim that we can obtain

$$\min_S \left( [x_n] \prod_{g \in S} \sigma^{h_S(g)}(x_h|_g) \right),$$

which equals $x_h|_{g_1}$ for the $g_1$ in layer $n$ with the smallest coheight profile.
Let $f$ be the coloring such that (1) is $x_f$. The condition means that $x_f$ must match $\sigma(x_h) = x_{f\omega}^n$ up to the exponent of $x_{n-1}$, but $x_f$’s exponent of $x_n$ is 1 smaller. Thus, $f$ up to layer $n$ is like $f_{\omega}$ but with one less use of color $n$, leaving a “gap” in layer $n$. With the maximality condition, $f$ after layer $n$ is also like $f_{\omega}$ except that the colors of the gap’s descendants are shifted up by one. Given this, it’s possible to see that the maximality condition forces the gap to be $g_1$ (smallest coheight profile). See Figure 12 for a depiction of $f$. It’s then possible to show that by comparing $x_f$ with $x_{f\omega}$, we can obtain $x_{h|g_1}$.

Figure 12: Here the coloring $f$ as defined in Theorem 6 is depicted. $g_1$ is circled.

The idea behind the general case is to leave more gaps by imposing more stringent conditions in (1). For example, imposing the condition $[\sigma(\phi_n(x_h)x_{n-2}^n)]$ in (1) leaves 2 gaps in layer $n$, equivalent to imposing the condition $[x_{n}^2]$ in (2).

Here is the formal treatment of the general case. The strategy is to apply Theorem 3 and move the condition into the expression, step by step. Like in Theorem 4, notice that $\sigma$ preserves the ordering of the elements of $(x_i)_{i \in N}$, while $\tau$ flips it; hence, max becomes min.

\[
\begin{align*}
&= \max_S \left( \sigma \left( \phi_n(x_h) \prod_{i \leq n} x_i^{e_{i}-e_i} \right) \right) x_{f\omega} \\
&= \max_S \left( \sigma \left( \phi_n(x_h) \prod_{i \leq n} x_i^{e_{i}-e_i} \right) \right) \sigma \left( x_h \tau \left( \prod_{g \in S} \sigma^h_s(g)(x_h|g) \right) \right) \\
&= \sigma \left( \max_S \left( \phi_n(x_h) \prod_{i \leq n} x_i^{e_{i}-e_i} \right) \right) \sigma \left( x_h \tau \left( \prod_{g \in S} \sigma^h_s(g)(x_h|g) \right) \right) \\
&= \sigma \left( x_h \max_S \left( \prod_{i \leq n} x_i^{e_{i}-e_i} \right) \tau \left( \prod_{g \in S} \sigma^h_s(g)(x_h|g) \right) \right) \\
&= \sigma \left( x_h^\tau \left( \min_S \left( \prod_{i \leq n} x_i^{e_{i}} \prod_{g \in S} \sigma^h_s(g)(x_h|g) \right) \right) \right).
\end{align*}
\]

In the last step, we use that $\tau(\prod_{i \leq n} x_i^{e_{i}}) = \prod_{i \leq n} x_i^{e_{i}-e_i}$. Since $\sigma$ and $\tau$ are reversible, we can now obtain

\[
\min_S \left( \prod_{i \leq n} x_i^{e_{i}} \prod_{g \in S} \sigma^h_s(g)(x_h|g) \right).
\]

**Remark 1.** Note that one can freely add to or remove from $S$ any vertex $v$ satisfying $h_S(v) + h_v > n$, since this will not change whether the exponent of $x_i$ in the product is $e_i$. Removing vertices from $S$ is guaranteed to decrease the product; thus, we know that the minimum $S$ has no removable vertices.

**Theorem 6.** From knowledge of $\Gamma^<(T; x)$, we can obtain $x_{h|\omega}$.
Proof. By Theorem 5, we can obtain the following for all \( n, m \):

\[
\min_S \left( \left[ x_n^m \right] \prod_{g \in S} \sigma^{h_S(g)}(x_{h \mid g}) \right).
\]

Why choose this expression? The idea is to force \( S \) to include no vertices with coheight \(< n \) and \( m \) vertices with coheight \( n \). The \([x_n^m]\) condition achieves this. Then, by taking the minimum, we get rid of anything extra: we know from Remark 1 that \( S \) need not contain vertices with coheight \( > n \), nor any repeats (else \( h_S(g) \neq 0 \)). Then, the minimum will happen when \( S = \{ g_i \mid 1 \leq i \leq m \} \), where \( g_i \) is the vertex with coheight \( n \) with the \( i \)th least coheight profile. Thus, we have

\[
\min_S \left( \left[ x_n^m \right] \prod_{g \in S} \sigma^{h_S(g)}(x_{h \mid g}) \right) = \prod_{1 \leq i \leq m} x_{h \mid g_i}.
\]

If we know this expression for every value of \( m \), then we can obtain \( x_{h \mid g_m} \).

Splitting \( x_{h \mid g} \) by coheight, we can write:

\[
x_{h \mid g} = \prod_{v \in V(T)} x_{h \mid g \mid v} = \prod_{n} \prod_{v \in V(T) \mid h_n = n} x_{h \mid g \mid v},
\]

and then, for each \( n \), the last product we can obtain from the \( x_{h \mid g_m} \) that we’ve obtained. \( \square \)

**Theorem 7.** From knowledge of \( \Gamma^<(T; x) \), we can obtain \( x_{(3)_h} \).

**Proof.** For a certain coheight \( n_0 \), let \( g_i \) be as defined in Theorem 6. By Theorem 6, we can obtain \( x_{h \mid g_i} \) for every \( i \). Thus, by Theorem 5, we can obtain the following for all \( m_0, n > n_0, m \):

\[
\min_S \left( \phi_n \left( \prod_{1 \leq i \leq m_0} x_{h \mid g_i} \right) x_n^m \prod_{g \in S} \sigma^{h_S(g)}(x_{h \mid g}) \right).
\]

(3)

The idea here is to force \( S \) to include \( g_i \mid 1 \leq i \leq m_0 \), and then in addition include \( m \) vertices that satisfy \( h_S(v) + h_n = n \). When taking the minimum, we run into the issue that the second requirement interferes with the first, which requires a careful combinatorial argument to resolve. Once the interference is resolved, we can use a technique similar to that used in the proof of Theorem 6 to obtain the coheight profile of each vertex in \( V(S_{g_i}) \), and compiling all of these together gives us \( x_{(3)_h} \).

Considering the \( \left[ \phi_n \left( \prod_{1 \leq i \leq m_0} x_{h \mid g_i} \right) x_n^m \right] \) condition, the first nonzero exponent is \( x_n^{m_0} \), which means that \( S \) must include \( m_0 \) vertices with coheight \( n_0 \). These \( m_0 \) vertices (let’s call the set \( S_0 \)) must satisfy the following, where the left inequality comes from \( S_0 \subseteq S \) and the right equality comes from the condition:

\[
\phi_n \left( \prod_{g \in S_0} x_{h \mid g} \right) \leq \phi_n \left( \prod_{g \in S} \sigma^{h_S(g)}(x_{h \mid g}) \right) = \phi_n \left( \prod_{1 \leq i \leq m_0} x_{h \mid g_i} \right) x_n^m.
\]

(4)

Recall that \( \{ g_i \mid 1 \leq i \leq m_0 \} \) is the \( S_0 \) with the least possible product of coheight profiles, or in other words

\[
\min_{S_0} \left( \prod_{g \in S_0} x_{h \mid g} \right) = \prod_{1 \leq i \leq m_0} x_{h \mid g_i},
\]

which necessarily means that

\[
\phi_n \left( \prod_{g \in S_0} x_{h \mid g} \right) \geq \phi_n \left( \prod_{1 \leq i \leq m_0} x_{h \mid g_i} \right).
\]

(5)
Putting (4) and (5) together, we have that

$$\phi_n \left( \prod_{1 \leq i \leq m_0} x_{h | g_i} \right) \leq \phi_n \left( \prod_{g \in S_0} x_{h | g} \right) \leq \phi_n \left( \prod_{1 \leq i \leq m_0} x_{h | g_i} \right) x_n^{m'}.$$  \hspace{1cm} (6)

This forces \( \phi_n \left( \prod_{g \in S_0} x_{h | g} \right) \) to be of the form \( \phi_n \left( \prod_{1 \leq i \leq m_0} x_{h | g_i} \right) x_n^{m'} \) for some \( 0 \leq m' \leq m \).

Now, since by Theorem 6 we know \( x_{h | g_i} \), for all \( i \), we know all the choices we have for \( S_0 \).

Let \( S' \setminus S_0 = S_1 \). Consider the equality in (4). We can split the left hand side into terms for \( S_0 \) and \( S_1 \) as follows:

$$\phi_n \left( \prod_{1 \leq i \leq m_0} x_{h | g_i} \right) x_n^{m'} \cdot \phi_n \left( \prod_{g \in S_1} \sigma^{h_S(g)}(x_{h | g}) \right) = \phi_n \left( \prod_{1 \leq i \leq m_0} x_{h | g_i} \right) x_n^{m},$$

which gives us

$$\phi_n \left( \prod_{g \in S_1} \sigma^{h_S(g)}(x_{h | g}) \right) = x_n^{m-m'}.$$

This forces that \( S_1 \) contains no vertices with \( h_S(v) + h_v < n \) and \( m-m' \) vertices with \( h_S(v) + h_v = n \). In addition, since we are taking a minimum, we need not consider any vertices with \( h_S(v) + h_v > n \) (Remark 1).

Now, we know that (3) is equal to

$$\min_{S = S_0 \cup S_1} \left( \prod_{g \in S} \sigma^{h_S(g)}(x_{h | g}) \right) = \min_{S_0, S_1} \left( \left( \prod_{g \in S_0} x_{h | g} \right) \cdot \left( \prod_{g \in S_1} \sigma^{h_S(g)}(x_{h | g}) \right) \right),$$  \hspace{1cm} (8)

where the minimum is taken over \( S_0 \) satisfying (6) and \( S_1 \) satisfying (7).

One might hope that this choice of \( S_0 \) would be the same as the choice that produces a minimum value for \( \prod_{g \in S_0} x_{h | g} \). If this were the case, then one would be guaranteed \( S_0 = \{ g_i \mid 1 \leq i \leq m_0 \} \) by Theorem 6. However, this is not true. Notice that the possibilities for \( S_1 \) depend on \( S_0 \) due to the \( h_S(v) \) term. Thus, it might be the case that the minimal \( S_1 \) of a non-minimal \( S_0 \) produces a smaller value than the minimal \( S_1 \) of the minimal \( S_0 \). (We better be thankful for the \( h_S(v) \) term, though, because otherwise we’d have no way of telling whether a vertex \( v \) is “under” a vertex of \( S_0 \), which would give us no additional information.) We call this non-minimality issue the “swapping problem.”

Rest assured that we can still determine the information that we want, which is the coheight profile of each vertex in \( V(S_g) \). In Section 6, we set up another framework to describe the combinatorial procedure that cleans up our information, and we defer the rest of the proof to Section 7.

To be continued.

6 Second framework

With this framework, we aim to describe the combinatorial procedure that cleans up the information obtained in the proof of Theorem 7.

Recall that the information is as follows: for every choice of \( n_0, m_0, n, m \), we know expression (8). We wish to determine, for each \( g_i \) with coheight \( n_0 \), the coheight profile \( x_{h | v} \) of each \( v \in V(S_g) \).

Let us first set up some notation.

- Let \( L_k \) be the set of vertices with coheight \( k \).
- Let \( L_k(g_i) \) be the set of descendants of \( g_i \) with coheight \( k \).
- Let \( L_k(S_0) \) be the set of descendants of vertices in \( S_0 \) with coheight \( k \).
• Let \( g_i \) be the element of \( L_{m_0} \) with the \( i \)th least coheight profile (equivalent to the definition above).

• Let \( v_i \) be the element of \( L_n \) with the \( i \)th least coheight profile.

For a given \( S_0 \), the candidates for vertices in \( S_1 \) are those that satisfy \( h_{S_0}(v) + h_v = n \). These include \( L_{n-1}(S_0) \) as well as \( L_n \setminus L_n(S_0) \). We invoke induction on \( n \) so that our inductive hypothesis is the following: for any \( g_i \), we know the coheight profiles of \( L_{n-1}(g_i) \). By Theorem 6, we also know the coheight profiles of \( L_n \). Thus, our task is to determine for any \( g_i \) which of the vertices in \( L_n \) are in \( L_n(g_i) \).

This goal can be more easily discussed with the following definition.

**Definition 24.** For each \( v_i \), the **position** of \( v_i \) is the vertex \( g_j \) such that \( v_i \in L_n(g_j) \).

With this definition, our goal is to determine the position of each vertex \( v_i \in L_n \).

We determine the positions of \( v_1, \ldots, v_{|L_n|} \) inductively. To determine the position of \( v_i \), we make certain choices of \( n, m, n_0, m_0 \) such that we can predict (8) with our current knowledge, assuming \( v_i \not\in L_n(S_0) \) for the predicted \( S_0 \). Thus, we know that \( v_i \not\in L_n(S_0) \) if and only if the prediction is right. In addition, we will show that the predicted \( S_{0|S} \) is sufficient to narrow down \( v_i \) to one possible position.

We set up a few definitions to allow (8) to be more easily compared. Note that these definitions are not wholly rigorous; they are meant to be a guide and will be modified throughout the section.

(8) is a product of two terms. The first term is encapsulated in the following definition.

**Definition 25.** The **padding** on \( S_0 \), denoted \( P(S_0) \), is \( \prod_{g \in S_0} x_h|_g \).

The second term is encapsulated in the following definition.

**Definition 26.** Fix an \( S_0 \). Recall that the set of candidates for \( S_1 \) is \( L_{n-1}(S_0) \cup L_n \setminus L_n(S_0) \). The **stack** on \( S_0 \), denoted \( S(S_0) \), is the sequence defined by the set

\[
\{a^{h_{S_0}(g)}(x_h|_v) \mid v \in L_{n-1}(S_0) \cup L_n \setminus L_n(S_0)\}
\]

arranged from least to greatest. In the case of equalities, we place elements of \( L_{n-1}(S_0) \) before elements of \( L_n \setminus L_n(S_0) \). We let the \( i \)th element of \( S(S_0) \) be \( S_i(S_0) \).

Importantly, notice that we don’t currently know all of \( S(S_0) \). We know only the elements that are less than \( x_h|_{v_1} \), because it’s uncertain whether \( v_1 \in L_n \setminus L_n(S_0) \).

**Definition 27.** The **\( k \)th partial product** of the stack on \( S_0 \), denoted \( \prod_{i=1}^{k} S_i(S_0) \), is the product of the first \( k \) elements of the stack.

Using the above definitions, we rewrite (8) as

\[
\min_{S_0, S_1} \left( P(S_0) \prod_{i=1}^{\|S_1\|} S_i(S_0) \right).
\]

We only know the value of this if all the terms \( S_i(S_0) \) are \( < x_h|_{v_1} \). Any bigger, and we have to consider whether \( x_h|_{v_1} \) is in the stack. Our strategy is to first compare partial products until we have to consider \( v_1 \), and then determine which stacks contain \( v_1 \).

**Theorem 8.** Let us take the initial expression (3) for \( m_0 = 1 \) and variable \( m \); precisely,

\[
\min_{S} \left[ \phi_n(x_h|_{g_1}) x_m \prod_{g \in S} a^{h_{S}(g)}(x_h|_g) \right].
\]

Pretend that \( x_h|_{v_1} \) is in every stack (it’s not, but pretend that it is). We claim that there exists an \( m \) such that the \( S_1 \) we predict has largest vertex \( v_1 \).
Proof. In this case, $S_0$ must be a single vertex $g$ satisfying condition (6). Hereafter, we call vertices $g$ that satisfy condition (6) “possible” $g$s.

For each $m$, our prediction involves the vertex $g$ that gives the smallest $P(g) \cdot \prod_{i=1}^{m-m'} S_i(g)$. Note that $m'$ is a function of $g$, so our partial products are not lined up. In order to line them up, we lift each stack $S(g)$ up by $m'$ elements; that is, we increase the indices by $m'$ (leaving $< m'$th partial products undefined).

For each possible $g$, we want to consider $m$ for which $S_m(g) = x_{h|v_1}$; specifically the largest such $m$, since we want to pick out the actual $x_{h|v_1}$ from other equal elements (remember that in the case of equalities, we place elements of $L_{n-1}(S_0)$ before elements of $L_n \setminus L_n(S_0)$). Thus we make the following definitions:

Definition 28. For a positive integer $m$, the minimal gap at $m$, denoted $g_m$, is the possible $g$ that gives the smallest $P(g) \cdot \prod_{i=1}^{m} S_i(g)$ (for $g$ where this is defined).

Definition 29. The critical point of $g$, denoted $g_x$, is the largest $m$ for which $S_m(g) = x_{h|v_1}$.

To prove this theorem, we want to show that for some $m$, the predicted $S_1$ has largest vertex $v_1$ (assuming that $v_1 \notin L_n(S_0)$ for the predicted $S_0$). In terms of the above definitions, this $m$ needs to satisfy two criteria: it’s the critical point of some possible $g$ (so that the largest vertex of the predicted $S_1$ is $v_1$), and this $g$ is the minimal gap at $m$ (so that this $g$ is actually the predicted). Thus, we want to show that some $m$ satisfies $m_g = m$.

We begin from $m = 1$ and increment upward. At every step, we have three possibilities:

1. $m_{g_m} < m$
2. $m_{g_m} = m$
3. $m_{g_m} > m$

If 2) $m_{g_m} = m$ is true, then we are done, and we stop the procedure. Thus, the procedure only continues if 1) $m_{g_m} < m$ or 3) $m_{g_m} > m$ is true. We claim that 1) $m_{g_m} < m$ is never true. Since 3) $m_{g_m} > m$ cannot be true for the maximal critical point, the procedure must eventually stop.

To show that 1) $m_{g_m} < m$ is never true, we proceed by induction. This is trivially true for the first critical point. For the inductive step, suppose that 1) $m_{g_m} < m$ is true. Let’s now consider $m - 1$. Our procedure has already passed $m - 1$, so 2) $m_{g_{m-1}} = m - 1$ is not true. By our inductive hypothesis, 1) $m_{g_{m-1}} < m - 1$ is not true. Thus, we must have 3) $m_{g_{m-1}} > m - 1$.

By the definition of $g_{m-1}$, we know that

$$P(g_m) \cdot \prod_{i=1}^{m-1} S_i(g_m) \geq P(g_{m-1}) \cdot \prod_{i=1}^{m-1} S_i(g_{m-1}).$$

Now, let’s look at the definition of critical point.

- Since $m_{g_m}$ is the critical point of $g_m$, we know that $S_i(g_m) \geq x_{h|v_1}$ for $i \geq m_{g_m}$. We assumed above that $m_{g_m} < m$, so we know that $i \geq m_{g_m}$ includes $i \geq m$.
- Since $m_{g_{m-1}}$ is the critical point of $g_{m-1}$, we know that $S_i(g_{m-1}) \leq x_{h|v_1}$ for $i \leq m_{g_{m-1}}$. We determined above that $m_{g_{m-1}} > m - 1$, so we know that $i \leq m_{g_{m-1}}$ includes $i \leq m$.

The two conditions overlap at $i = m$. Thus, we know that $S_m(g_m) \geq x_{h|v_1} > S_m(g_{m-1})$. Multiplying inequalities, we get

$$P(g_m) \cdot \prod_{i=1}^{m} S_i(g_m) \geq P(g_{m-1}) \cdot \prod_{i=1}^{m} S_i(g_{m-1}).$$

This contradicts the definition that $g_m$ is the minimal gap at $m$. Thus, 1) could not have been true, and the theorem is proved.

Notice that in making our prediction, we didn’t need to know any elements $S_i(S_0)$ larger than $x_{h|v_1}$; we just needed to know that they were larger. \qed
Remark 2. Theorem 8 produces an ordering on the \( g \in L_n \) as follows. We let the smallest \( g \) be the predicted \( g \) in the situation of Theorem 8. Then, remove this \( g \). We can apply Theorem 8 again to get another predicted \( g \). Let this be the second smallest \( g \). We proceed like this until all the \( g \in L_n \) are used up, and this produces an ordering on the \( gs \). This will be important later.

Theorem 9. The prediction in Theorem 8 is right if and only if \( v_1 \) is not in \( L_n(g) \), where \( g \) is the predicted \( g \).

Proof. Let \( g_0 \) be the position of \( v_1 \).

If the predicted \( g = g_0 \), then the prediction will be wrong because \( x_{h|v_1} \) is not actually in \( S(g) \), as we had assumed.

If the predicted \( g \neq g_0 \), then the only change is that the partial products of \( S(g_0) \) after \( x_{h|v_1} \) increase. This wouldn’t change any of the predictions for \( m \) for \( m < m_{g_0} \), nothing changes, and for \( m > m_{g_0} \), we know that \( g_m \neq g_0 \) (otherwise \( m_{g_m} < m \)) and so increasing a partial product of \( S(g_0) \) wouldn’t change predictions for \( m \). Thus, the prediction would be correct. \( \Box \)

Theorem 10. Let us take the initial expression (3) for fixed \( m_0 \neq 1 \) and variable \( m \). We claim that there exists an \( m \) such that the \( S_i \) we predict has largest vertex \( v_1 \), and that in addition, that the predicted \( g_i \) are the \( m_0 \) least \( g_i \) according to the ordering described in Remark 2.

Proof. Let’s begin with new definitions of minimal gap (set) and critical point:

Definition 30. For a positive integer \( m \), the minimal gap set at \( m \) is the set \( S_0 \) of \( m_0 \) possible \( gs \) that minimizes \( \mathcal{P}(S_0) \cdot \prod_{i=1}^{m_0} S_i(S_0) \) (for \( S_0 \) where this is defined).

Definition 31. The critical point of \( S_0 \), denoted \( m_{S_0} \), is the largest \( m \) for which \( S_m(S_0) = x_{h|v_1} \).

In addition, let \( M \) be the set containing the \( m_0 \) least possible \( gs \) under the ordering described in Remark 2. We claim that \( M = g_{m_M} \).

Consider any other set \( S_0 \neq M \) of \( m_0 \) possible \( gs \). Let’s order the elements of \( M \) and \( S_0 \) via Remark 2. By definition of \( M \), we know that pairs \((g, g')\) of corresponding elements satisfy

\[
\mathcal{P}(g) \cdot \prod_{i=1}^{m_g} S_i(g) \leq \mathcal{P}(g') \cdot \prod_{i=1}^{m_g} S_i(g').
\]

Multiplying the equations together for all \( g \), we get

\[
\mathcal{P}(M) \cdot \prod_{(g, g')}=1 \prod_{i=1}^{m_g} S_i(g) \leq \mathcal{P}(S_0) \cdot \prod_{(g, g')}=1 \prod_{i=1}^{m_g} S_i(g').
\]

Since the left hand side contains exactly the terms of \( S_i(g) \) \( g \in M \) that are less than or equal to \( x_{h|v_1} \), we know that the left hand side is equal to \( \mathcal{P}(M) \cdot \prod_{i=1}^{m_M} S_i(M) \) (approximately, with some extra \( x_{h|v_8} \)). For the right hand side, if we only consider elements \( \leq x_{h|v_1} \), notice that \( S(S_0) = \bigcup_{g \in S_0} S(g) \) (reordered properly). Thus, we must have that \( \prod_{(g, g')}=1 \prod_{i=1}^{m_g} S_i(g') \leq \prod_{i=1}^{m_M} S_i(S_0) \). \( \Box \)

Theorem 11. Theorems 8 through 10 are also true when \( v_1 \) is replaced with \( v_i \), for any \( i \).

Proof. Generalized Theorems 8 and 9 work trivially for general \( v_i \); since we already know the position of \( v_j \) for \( j < i \), we know every stack \( S(g) \) up until \( x_{h|v_i} \), which is enough to predict whether the position of \( v_i \) is \( g \).

Generalized Theorem 10 is a little more tricky, because the rule that \( x_{h|v} \in S(g) \) implies \( x_{h|v} \in S(S_0) \) for \( S_0 \ni g \) doesn’t hold true for \( x_{h|v_i} \). However, we can apply a transformation to the stacks and then proceed in a similar fashion to Theorem 10. The transformation is as follows:

Because the stacks that we consider are up to \( x_{h|v_i} \), every \( x_{h|v_j} \) for \( j < i \) is guaranteed to be included. Thus, we can delete \( x_{h|v_j} \) from each stack and preserve the inequalities that we care about. For the unique stack \( S(g_0) \) that we’ve already determined doesn’t contain \( x_{h|v_j} \), we divide the padding \( \mathcal{P}(g_0) \) by \( x_{h|v_j} \). This preserves the rule that \( x_{h|v_j} \in S(S_0) \) if and only if \( g_0 \notin S_0 \). \( \Box \)

Theorem 12. The \( S_0 \)s we predict are sufficient to narrow down \( v_i \) to one possible position.
Proof. By generalized Theorem 9, the position of \( v_1 \) is one of the \( m_0 \) least possible \( g_s \) under Remark 2 if and only if the prediction for \( m_0 \) is wrong.

Thus, we can determine the position of \( v_1 \) as follows. We check our predictions for \( m_0 = 1, 2, \ldots \) until we get one that’s wrong: let this be \( m_f \). Then the position of \( v_1 \) must be one of the \( m_f \) least possible \( g_s \) under Remark 2, and it cannot be any of the \( m_f - 1 \) least possible \( g_s \) under Remark 2. Thus, it must be precisely the \( m_f \)th least possible \( g \) under Remark 2.  

7 Main result, continued

Proof of Theorem 7, continued. Now, we know the coheight profiles of every vertex in \( V(S_{g_1}) \) for each \( g_1 \). Thus, we know the coheight profile \( x_{S_{g_1}} \) of each \( g_1 \).

We can proceed to find \( x_{S_{g_1}} \) for the entire tree via the definition:

\[
x_{(3)h} = \prod_{v \in V(T)} x_{x_h} | v.
\]

Theorem 13. From knowledge of \( \Gamma^< (T; x) \), we can obtain \( x_{(x)h} \) for any positive integer \( x \).

Proof. We can recursively perform something analogous to Theorem 7 in order to secure general \( x_{(x)h} \). Rather than just having \( S_0 \) and \( S_1 \), we also have \( S_2, S_3, \ldots, S_{x-2} \). The expression we consider is

\[
\min_{S} \left( \phi_{h_n} \left( \prod_{0 \leq x' \leq x-2, g \in S_{g'}} \sigma_x (x_{h}|g) \right) x_{S_{g}}^{m} \prod_{g \in S} \sigma_x (g) (x_{h}|g) \right).
\]

Suppose that we’ve already determined \( x_{(x-1)h} \). So we know the possibilities for \( S_0, \ldots, S_{x-3} \). Suppose also that we’re working inductively, so that we already know the positions of some of the candidates for \( S_{x-2} \).

Out of the candidates for \( S_{x-2} \) with undetermined position, let \( v_1 \) be the one with \( i \)th smallest coheight profile. Using an argument similar to Theorems 8 through 10, we have that for fixed \( S_0, \ldots, S_{x-4} \), we can find an \( S_{x-3} \) with \( |S_{x-3}| = 1 \) and \( |S_{x-2}| \) such that the predicted \( S_{x-2} \) has largest element \( v_1 \).

Via Remark 2, we can extend this to an ordering of the candidates for \( S_{x-3} \). Now, out of the candidates for \( S_{x-3} \) with undetermined position, let \( V_i \) be the \( i \)th smallest one under the above ordering.

Then, we allow \( S_{x-4} \) to vary. Using an argument similar to Theorems 8 through 10, we have that for fixed \( S_0, \ldots, S_{x-5} \), we can find an \( S_{x-4} \) with \( |S_{x-4}| = 1 \) and \( |S_{x-3}| \) such that the predicted \( S_{x-3} \) has largest element \( V_1 \). (Note: the essential reason why this argument works is that the ordering of Remark 2 has the additive property described in Theorem 10.)

Via Remark 2, we can extend this to an ordering of the candidates for \( S_{x-4} \). Then, we allow \( S_{x-5} \) to vary, and so on.

The final collection \( S_0, \ldots, S_{x-3}, [S_{x-2}] \) we find is the one we try first, and whether our prediction is right or not tells us whether \( v_1 \in L_n (S_{x-3}) \). By the inductive hypothesis, we already know whether \( v_1 \in L_n (S_{x-3} \setminus \{V_1\}) \), so we now know whether \( v_1 \in L_n (V_1) \).

Next, we do the same procedure for \( V_2 \), and we can determine whether \( v_1 \in L_n (V_2) \). We continue like this to determine the location of \( v_1 \). This directly generalizes to general \( v_i \).

Theorem 14. From knowledge of \( \Gamma^< (T; x) \), we can reconstruct \( T \).

Proof. By Theorem 13, we can obtain \( x_{(x)h} \) for any positive integer \( x \). This will be enough to reconstruct \( T \).

We invoke recursion on the number of layers in \( T \).

- If \( T \) has 2 layers, we can reconstruct \( T \) from \( x_h \), since we just need to know the number of children of the root.
Suppose that for some \( n \geq 2 \), the following is true: if \( T \) has \( n \) layers, then we can reconstruct \( T \) from \( x_{(n-1)}h \). We claim that if \( T \) has \( n+1 \) layers, then we can reconstruct \( T \) from \( x_{(n)}h \). Note that the subtree induced by a child of the root has at most \( n \) layers. Since knowing \( x_{(n)}h \) gives us \( x_{(n-1)}h \) of each child of the root, we can reconstruct each child’s induced subtree, which just needs to be connected to the root to complete \( T \).

\[ \square \]

8 Future directions

Hasebe and Tsuji’s result in [HT17] is actually stronger than the statement we considered in Section 1, which was that \( \Gamma^<(T,x) \) distinguishes rooted trees. In fact, they showed that \( \Gamma^<(T,x) \) distinguishes \((\mathbb{N},\geq)\)-free posets, which is a class of posets that includes but is not limited to rooted trees. We are interested to see if an analogue of our formalization and/or procedure exists in this broader setting.

It would be helpful to define an \((\mathbb{N},\geq)\)-free poset, as well as state the recursive construction that Hasebe and Tsuji use in [HT17] to prove their result.

Definition 32. An \((\mathbb{N},\geq)\)-free poset is a poset \( P \) such that for any four elements \( a, b, c, d \in P \), it is not true that

- \( a < b, b > c, c < d \), and all other pairs are incomparable (this is the \( \mathbb{N} \))
- \( a < b, b > c, c < d, d > a \), and all other pairs are incomparable (this is the \( \geq \)).

Here is the recursive construction, which is Theorem 4.3 in [HT17].

Theorem 15. Borrowing Hasebe and Tsuji’s notation, let \([1]\) be the one-element poset, \( \sqcup \) be the disjoint union operation on posets, and \( \oplus \) be the ordinal sum operation on posets. Let \( \mathcal{C} \) be the set of finite posets built in the following manner:

- \([1] \in \mathcal{C} \).
- If \( P, Q \in \mathcal{C} \), then \( P \sqcup Q \in \mathcal{C} \).
- If \( P \in \mathcal{C} \), then \([1] \oplus P \in \mathcal{C}\) and \( P \oplus [1] \in \mathcal{C} \).

Then \( \mathcal{C} \) is precisely the set of \((\mathbb{N},\geq)\)-free posets.

Another direction to explore is looking for situations similar to the swapping problem and then applying the “predict and verify” solution from Section 6. The swapping problem can be stated in a more general context as the following:

Suppose we have a totally ordered abelian group \( R \) and \( A_i, B_i \) are multisets with elements from \( R \). Let \( a_i(n) \) be the sum of the \( n \) least elements of \( A_i \cup \bigcup_{j \neq i} B_j \), and let \( \delta_i \in R \) be constants. Given \( A_i, S = \bigcup B_i \) and \( s_n = \min(a_i(n) + \delta_i) \) for \( 1 \leq n \leq \max(|A_i| + |S| - |B_i|) \), can we determine each individual \( A_i \)?

Our resolution to the problem applies in this general situation as well. Thus, any problem that reduces to this general situation can be solved with our method.

9 Acknowledgements

We thank Andrew Blumberg, Yongyi Chen, and Shuhei Tsuji for useful comments and advice.

References

[CSZ20] Lucy Cai, Espen Slettnes, and Jeremy Zhou. “A Combinatorial Approach to Extracting Rooted Tree Statistics from the Order Quasisymmetric Function”. In: 2020.

[HJ18] Sam Heil and Caleb Ji. “On an Algorithm for Comparing the Chromatic Symmetric Functions of Trees”. In: arXiv e-prints (2018). arXiv: 1801.07363 [math.CO].
[HT17] Takahiro Hasebe and Shuhei Tsuji. “Order Quasisymmetric Functions Distinguish Rooted Trees”. In: *Journal of Algebraic Combinatorics* 46 (2017), pp. 499–515. doi: 10.1007/s10801-017-0761-7.

[Hur20] Jake Huryn. “A Few More Trees the Chromatic Symmetric Function Can Distinguish”. In: *Involve* 13 (2020), pp. 109–116. doi: 10.2140/involve.2020.13.109.

[IW19] Ricky Ini Liu and Michael Weselcouch. “$P$-Partitions and Quasisymmetric Power Sums”. In: *arXiv e-prints* (2019). arXiv: 1903.00551 [math.CO].

[IW20] Ricky Ini Liu and Michael Weselcouch. “$P$-Partition Generating Function Equivalence of Naturally Labeled Posets”. In: *Journal of Combinatorial Theory, Series A* 170 (2020), pp. 105–136. doi: 10.1016/j.jcta.2019.105136.

[Sta95] Richard P. Stanley. “A symmetric function generalization of the chromatic polynomial of a graph”. In: *Advances in Mathematics* 111.1 (1995), pp. 166–194. doi: 10.1006/aima.1995.1020.

[TD17] Alessio Tonioni and Luigi Di Stefano. “Product recognition in store shelves as a subgraph isomorphism problem”. In: *arXiv e-prints* (2017). arXiv: 1707.08378 [cs.CV].

[Tsu18] Shuhei Tsuji. “The Chromatic Symmetric Functions of Trivially Perfect Graphs and Cographs”. In: *Graphs and Combinatorics* 34 (2018), pp. 1037–1048. doi: 10.1007/s00373-018-1928-2.