Séminaire Lotharingien de Combinatoire **93B** (2025)
Article #142, 12 pp.

Proceedings of the 37th Conference on Formal Power Series and Algebraic Combinatorics (Sapporo)

Plactic-like monoids arising from meets and joins of taiga congruences

Thomas Aird*[1] and Duarte Ribeiro†[2]

[1]*Department of Mathematics, The University of Manchester, Alan Turing Building, Oxford Rd, Manchester, M13 9PL, UK*
[2]*Center for Mathematics and Applications (NOVA Math), FCT NOVA, 2829-516 Caparica, Portugal*

Abstract. We study algebraic and combinatorial properties of plactic-like monoids arising from the meets and joins of taiga congruences. We construct combinatorial objects associated with the meet-taiga monoid, establish a Robinson–Schensted-like correspondence and give extraction and iterative insertion algorithms for said objects. Finally, we obtain 'hook-length'-like formulas for the various taiga monoids.

Keywords: Taiga monoid, binary trees with multiplicities, Robinson–Schensted correspondence, extraction algorithm, iterative insertion algorithm, hook-length formula

1 Introduction

Plactic-like monoids are an informal class of monoids that share the interesting property of their elements being uniquely identified with combinatorial objects, thus giving these monoids a fundamental role in algebraic combinatorics. The namesake of this family is the plactic monoid, also known as the monoid of Young tableaux [8]. It has been widely applied in several areas of mathematics such as representation theory, symmetric functions and crystal bases. Other plactic-like monoids have since arisen in the context of combinatorial Hopf algebras whose bases are indexed by combinatorial objects.

The Baxter monoid was introduced by Giraudo [4] as the analogue of the plactic monoid for Hopf algebras indexed by Baxter permutations. Its defining congruence was shown to be the meet of the sylvester congruence and its dual, in the lattice of congruences on the free monoid. Thus, its combinatorial objects are pairs of twin binary

*thomas.aird@manchester.ac.uk. Thomas Aird's work was supported by the London Mathematical Society, the Heilbronn Institute for Mathematical Research, and NOVA University Lisbon.

†duarte.ribeiro.math@gmail.com. Duarte Ribeiro's work was supported by National Funds through the FCT – Fundação para a Ciência e a Tecnologia, I.P., under the scope of the projects PTDC/MAT-PUR/31174/2017, UIDB/04621/2020 and UIDP/04621/2020 (Center for Computational and Stochastic Mathematics), and UIDB/00297/2020 (https://doi.org/10.54499/UIDB/00297/2020) and UIDP/00297/2020 (https://doi.org/10.54499/UIDP/00297/2020) (Center for Mathematics and Applications).

search trees, that is, pairs of combinatorial objects of the sylvester monoid and its dual, that satisfy certain restrictions. On the other hand, it is also known that the join of the sylvester congruence and its dual is the hypoplactic congruence [9].

The (right-) taiga monoid was defined by Priez [10] as the quotient of the free monoid by the join of the sylvester and (right-) stalactic congruences. Its associated combinatorial objects are binary search trees with multiplicities. Due to their similarity to the sylvester monoids, it is natural to ask what are the combinatorial properties of the 'Baxter' and 'hypoplactic' analogues of the taiga monoids.

In this extended abstract, we study such analogues, which we call the meet- and join-taiga monoids. These were introduced by the authors in [1], where one can find detailed proofs, as well as analogous results on the meet- and join-stalactic monoids.

The necessary background is given in Section 2. We introduce the meet- and join-taiga congruences in Section 3, for which we give several equivalent characterisations.

Section 4 focuses on the combinatorial objects associated with the meet-taiga monoid. We define P-symbols by taking pairs of binary search trees with multiplicities satisfying certain restrictions. We similarly define Q-symbols, and use them to establish Robinson–Schensted-like correspondences. We then give algorithms to extract words from P-symbols, as well as iterative versions of the insertion algorithms, allowing us to easily compute the concatenation of words under the meet-taiga congruence.

In Section 5, we provide explicit 'hook-length'-like formulas for the sizes of the left-, right- and join-taiga congruence classes. The meet-taiga case is equivalent to counting the number of linear extensions in specific posets, for which we state its time complexity. Finally, we give bounds for the number of binary search trees with multiplicities which form a pair of twin binary search trees with multiplicities with a given one.

2 Background

For background on monoids, congruences and presentations, see [7, Chapter 1]. Let $\mathbb{N} = \{1, 2, \ldots\}$ denote the set of natural numbers, without zero. For $a, b \in \mathbb{N}$, we denote by $[a]$ the set $\{1 < \cdots < a\}$, and by $[a, b]$ the set $\{k \in \mathbb{N} \mid a \leq k \leq b\}$, when $a < b$.

2.1 Words

For any non-empty set $\mathcal{X}$, we denote by $\mathcal{X}^*$ the free monoid generated by $\mathcal{X}$, that is, the set of all words over $\mathcal{X}$ under concatenation. We denote the empty word by ε, and refer to $\mathcal{X}$ as an *alphabet*, and to its elements as *letters*. For a word $\mathbf{w} \in \mathcal{X}^*$, its *restriction* to a subset $\mathcal{S}$ of $\mathcal{X}$, denoted by $\mathbf{w}[S]$, is the word obtained from $\mathbf{w}$ by removing any letter not in $\mathcal{S}$. We denote the length of $\mathbf{w}$ by $|\mathbf{w}|$ and, for each $x \in \mathcal{X}$, we denote the number of occurrences of x in $\mathbf{w}$ by $|\mathbf{w}|_x$. If $|\mathbf{w}|_x = 1$, we say x is a *simple letter* of $\mathbf{w}$, and we denote

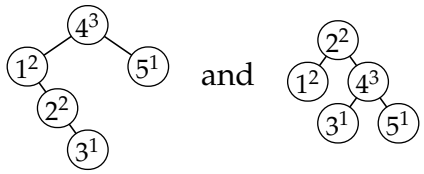
the restriction of $\mathbf{w}$ to its simple letters by $\overline{\mathbf{w}}$.

The subset of $\mathcal{X}$ of letters that occur in $\mathbf{w}$ is called the *support* of $\mathbf{w}$, denoted by $\mathrm{supp}(\mathbf{w})$, and the function from $\mathcal{X}$ to $\mathbb{N}_0$ given by $x \mapsto |\mathbf{w}|_x$ is called the *content* of $\mathbf{w}$, denoted by $\mathrm{cont}(\mathbf{w})$. Clearly, two words that share the same content also share the same support.

2.2 Binary trees with multiplicities and binary search trees with multiplicities

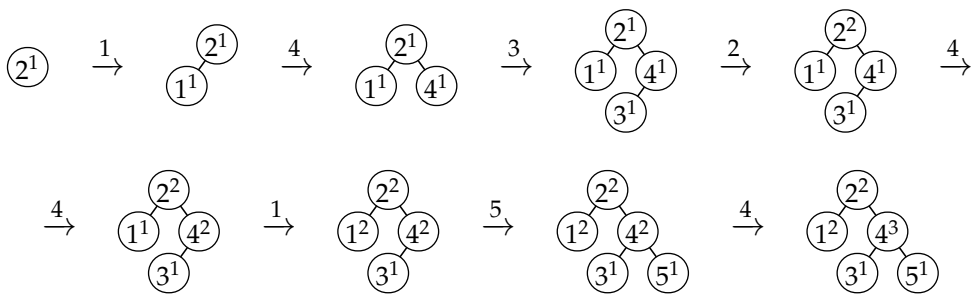
For background on right strict, left strict and pairs of twin binary search trees, see [4]. Empty combinatorial objects are denoted by $\perp$. We refer to the underlying unlabelled objects of labelled objects as their *shapes*.

A *binary tree with multiplicities* (BTM) is a rooted planar binary tree where each node is labelled by a positive integer, called a *multiplicity*. A *binary search tree with multiplicities* (BSTM) is a BTM where each node is further labelled by a letter of $\mathbb{N}$, such that the letter of the left (resp. right) child of any node is strictly less than (resp. strictly greater than) the letter of said node. We consider the *label* of a node of a BSTM to be the pair consisting of the letter and multiplicity that labels the node. For ease of notation, we write multiplicities as superscripts. Examples of BSTMs are the following:



Algorithm TgLI allows one to insert a letter into a BSTM, either by a new leaf labelled by it or by increasing the multiplicity of a node labelled by it, and still obtain a BSTM. Using this algorithm, one can compute a unique BSTM from a word $\mathbf{w}$: starting from the empty tree, read $\mathbf{w}$ from right to left (resp. left to right) and insert its letters one-by-one into the tree. The resulting tree is denoted by $\mathrm{P}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{w})$ (resp. $\mathrm{P}^{\rightarrow}_{\mathsf{lTg}}(\mathbf{w})$).

Example 2.1. Computing $\mathrm{P}^{\leftarrow}_{\mathsf{rTg}}(451423412)$:



Algorithm 1: *Taiga Leaf Insertion* (TgLI).

Input: A BSTM T, a letter $a \in \mathbb{N}$.
Output: A BSTM $T \uparrow a$.

if T *is empty* **then** add a node labelled a with multiplicity 1.
else
    let b be the letter that labels the root node of T;
    if $a < b$ **then** recursively insert a into the left subtree of the root node;
    else if $a > b$ **then** recursively insert a into the right subtree of the root node;
    else increment by 1 the multiplicity of the root node;
return the resulting tree $T \uparrow a$.

An *increasing* (resp. *decreasing*) *binary tree* is an $\mathbb{N}$-labelled rooted planar binary tree, such that the label of each node is less than (resp. greater than) the label of its children, and no two nodes have the same label.

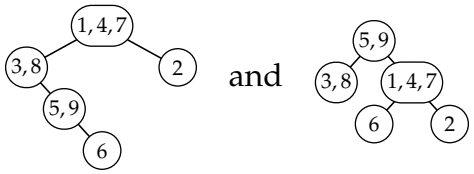
For a word $\mathbf{w}$ with no repeated letters, algorithm RiBTA allows one to compute an increasing binary tree $\mathrm{incr}(\mathbf{w})$ obtained from $\mathbf{w}$. We define a *Recursive Decreasing Binary Tree Algorithm* (RdBTA) algorithm by replacing "incr" with "decr" and "least" with "greatest" in RiBTA.

Algorithm 2: *Recursive Increasing Binary Tree Algorithm* (RiBTA).

Input: A word $\mathbf{w} \in \mathbb{N}^*$ with no repeated letters.
Output: An increasing binary tree $\mathrm{incr}(\mathbf{w})$.

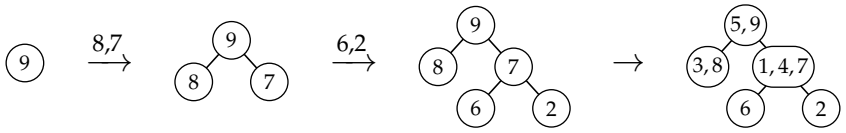
let $\mathrm{incr}(\mathbf{w}) := \perp$;
if $|\mathbf{w}| \geq 1$ **then** write $\mathbf{w} = \mathbf{u}a\mathbf{v}$, where a is the least letter of $\mathbf{w}$, label the root node of $\mathrm{incr}(\mathbf{w})$ by a and recursively compute its left subtree $\mathrm{incr}(\mathbf{u})$ and right subtree $\mathrm{incr}(\mathbf{v})$;
return the resulting tree $\mathrm{incr}(\mathbf{w})$.

We say that a binary tree whose labels are non-intersecting subsets of $\mathbb{N}$ is an *increasing* (resp. *decreasing*) *binary tree over sets* (BTS) if the binary tree obtained by replacing each set with its minimum (resp. maximum) element is an increasing (resp. decreasing) binary tree. Examples of increasing and decreasing binary trees over sets are the following:



Using Algorithm RiBTA (resp. RdBTA), one can compute a unique increasing (resp. decreasing) BTS from a word $\mathbf{w}$: Let $\mathrm{supp}(\mathbf{w}) = \{x_1 < \cdots < x_k\}$. For each $i \in [k]$, consider the set W_i of all $j \in [|\mathbf{w}|]$ such that x_i is the j-th letter of $\mathbf{w}$, when reading $\mathbf{w}$ from left to right. Let $\mathbf{v} = v_1 \cdots v_k$ be the word where $v_i = \min(W_i)$ (resp. $v_i = \max(W_i)$). Now, apply RiBTA (resp. RdBTA), to $\mathbf{v}$ to obtain an increasing (resp. decreasing) binary tree. Finally, for each $i \in [k]$ replace the label v_i in the tree with the set W_i. The resulting tree is denoted $\mathrm{Q}^{\rightarrow}_{\mathsf{lTg}}(\mathbf{w})$ (resp. $\mathrm{Q}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{w})$).

Example 2.2. Computing $\mathrm{Q}^{\leftarrow}_{\mathsf{rTg}}(451423412)$: Let $\mathbf{w} = 451423412$. Then, $W_1 = \{3, 8\}$, $W_2 = \{5, 9\}$, $W_3 = \{6\}$, $W_4 = \{1, 4, 7\}$ and $W_5 = \{2\}$ and, as such, $\mathbf{v} = 89672$.



Ley $\mathbf{w}$ be a word with support $\{x_1 < \cdots < x_k\}$. Then, $\mathrm{P}^{\rightarrow}_{\mathsf{lTg}}(\mathbf{w})$ (resp. $\mathrm{P}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{w})$) and $\mathrm{Q}^{\rightarrow}_{\mathsf{lTg}}(\mathbf{w})$ (resp. $\mathrm{Q}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{w})$) have the same underlying binary tree shape. Furthermore, for each $i \in [k]$, the letters x_i have their positions in $\mathbf{w}$ given by the label of the i-th node of $\mathrm{Q}^{\rightarrow}_{\mathsf{lTg}}(\mathbf{w})$ (resp. $\mathrm{Q}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{w})$). From this, we have an analogue of the Robinson–Schensted correspondence for pairs of BSTMs and BTSs, first stated in [10]. We give the result for the right-insertion algorithm case:

Theorem 2.3. *The map* $\mathbf{w} \mapsto (\mathrm{P}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{w}), \mathrm{Q}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{w}))$ *is a bijection between the elements of* $\mathbb{N}^*$ *and the set formed by the pairs* (T, S) *where*

(i) *T is a BSTM;*

(ii) *S is a decreasing BTS such that the union of the sets labelling S is the interval $[m]$, where m is the sum of the multiplicities of T;*

(iii) *T and S have the same underlying binary tree shape;*

(iv) *the multiplicity of the i-th node of T is the cardinality of the set labelling the i-th node of S.*

2.3 The right- and left-taiga monoids

For background on the hypoplactic, sylvester and stalactic monoids, see [9], [5] and [6], respectively.

The *right-taiga congruence* $\equiv_{\mathsf{rTg}}$ on $\mathbb{N}^*$ is defined in the following way: for $\mathbf{u}, \mathbf{v} \in \mathbb{N}^*$,

$$\mathbf{u} \equiv_{\mathsf{rTg}} \mathbf{v} \Leftrightarrow \mathrm{P}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{u}) = \mathrm{P}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{v}).$$

The factor monoid $\mathbb{N}^*/\equiv_{\mathsf{rTg}}$ is the (infinite-rank) *right-taiga monoid*, denoted by rTg. It follows from the definition of rTg that any element $[\mathbf{w}]_{\mathsf{rTg}}$ can be uniquely identified with the BSTM $\mathrm{P}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{w})$. All words in each $\equiv_{\mathsf{rTg}}$-class share the same content (and therefore the same support), hence, we extend these definitions to $\equiv_{\mathsf{rTg}}$-classes and BSTMs in a natural way. The left-taiga congruence $\equiv_{\mathsf{lTg}}$ and monoid lTg are defined analogously.

Recall that the sylvester and #-sylvester congruences are generated, respectively, by the relations

$$\mathcal{R}_{\mathsf{sylv}} = \{(ca\mathbf{u}b, ac\mathbf{u}b) \mid a \leq b < c, \mathbf{u} \in \mathbb{N}^*\} \text{ and}$$
$$\mathcal{R}_{\mathsf{sylvh}} = \{(b\mathbf{u}ac, b\mathbf{u}ca) \mid a < b \leq c, \mathbf{u} \in \mathbb{N}^*\},$$

and the right- and left-stalactic congruences are generated, respectively, by the relations

$$\mathcal{R}_{\mathsf{rSt}} = \{(ba\mathbf{u}b, ab\mathbf{u}b) \mid a, b \in \mathbb{N}, \mathbf{u} \in \mathbb{N}^*\} \text{ and}$$
$$\mathcal{R}_{\mathsf{lSt}} = \{(b\mathbf{u}ab, b\mathbf{u}ba) \mid a, b \in \mathbb{N}, \mathbf{u} \in \mathbb{N}^*\}.$$

Proposition 2.4 ([10, Proposition 4])**.** *The right- and left-taiga congruences are generated, respectively, by the relations*

$$\mathcal{R}_{\mathsf{rTg}} = \mathcal{R}_{\mathsf{sylv}} \cup \mathcal{R}_{\mathsf{rSt}} \quad \textit{and} \quad \mathcal{R}_{\mathsf{lTg}} = \mathcal{R}_{\mathsf{sylvh}} \cup \mathcal{R}_{\mathsf{lSt}}.$$

The proof of [2, Lemma 17] can be easily adapted to show that rTg is left-cancellative and lTg is right-cancellative.

3 Meets and joins of taiga congruences

We now consider, in the lattice of congruences on $\mathbb{N}^*$, the meet and join of the taiga congruences given in the previous subsection. We respectively call them the *meet-taiga* and the *join-taiga* congruences, and denote them by $\equiv_{\mathsf{mTg}}$ and $\equiv_{\mathsf{jTg}}$. Thus, we define the *meet-taiga* monoid mTg and *join-taiga* monoid jTg as quotients of $\mathbb{N}^*$ by the corresponding congruences.

We start by giving generating relations for these congruences. Clearly, the join-taiga congruence is generated by the relations $\mathcal{R}_{\mathsf{jTg}} := \mathcal{R}_{\mathsf{rTg}} \cup \mathcal{R}_{\mathsf{lTg}}$. On the other hand, we have that:

Proposition 3.1. *The meet-taiga congruence is generated by the relations*

$$\mathcal{R}_{\mathsf{mTg}} := \{(b\mathbf{u}ad\mathbf{v}c, b\mathbf{u}da\mathbf{v}c) \mid a \leq d,\ b, c \in [a, d],\ \mathbf{u}, \mathbf{v} \in \mathbb{N}^*\}.$$

We can characterise the join-taiga class of a word by looking at the hypoplactic classes of specific subsequences of it:

Proposition 3.2. *Let* $\mathbf{u}, \mathbf{v} \in \mathbb{N}^*$*. Let* $A_1, \ldots, A_k$ *be all the intervals of* $\mathrm{supp}(\mathbf{u})$ *such that* A_i *only contains simple letters and* $A_i \cup \{a\}$ *is not an interval, for any simple letter* $a \notin A_i$*. Then,* $\mathbf{u} \equiv_{\mathsf{jTg}} \mathbf{v}$ *if and only if* $\mathrm{cont}(\mathbf{u}) = \mathrm{cont}(\mathbf{v})$ *and* $\mathbf{u}[A_i] \equiv_{\mathsf{hypo}} \mathbf{v}[A_i]$ *for all* $1 \leq i \leq k$*.*

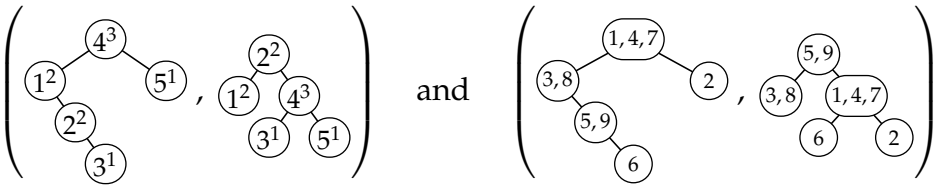
4 Robinson–Schensted-like correspondence

In this section, we first introduce an analogue of the Robinson–Schensted correspondence for the meet-taiga monoid. We then show how to extract representatives of classes from their P-symbols, and give an iterative version of the insertion algorithm, allowing us to compute a combinatorial object while reading a word in one direction only.

4.1 Definition and correctness of the correspondence

We define the *meet-taiga* P*-symbol* of a word $\mathbf{w}$ as the pair $(\mathrm{P}^{\rightarrow}_{\mathsf{lTg}}(\mathbf{w}), \mathrm{P}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{w}))$ of BSTMs, denoted by $\mathrm{P}_{\mathsf{mTg}}(\mathbf{w})$, and the *meet-taiga* Q*-symbol* of $\mathbf{w}$ as the pair $(\mathrm{Q}^{\rightarrow}_{\mathsf{lTg}}(\mathbf{w}), \mathrm{Q}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{w}))$ of (respectively) increasing and decreasing binary trees, denoted by $\mathrm{Q}_{\mathsf{mTg}}(\mathbf{w})$.

Example 4.1. The meet-taiga P and Q-symbols of 451423412 are, respectively, the following:



Proposition 4.2. *Let* $\mathbf{u}, \mathbf{v} \in \mathbb{N}^*$. *Then* $\mathbf{u} \equiv_{\mathsf{mTg}} \mathbf{v}$ *if and only if* $\mathrm{P}_{\mathsf{mTg}}(\mathbf{u}) = \mathrm{P}_{\mathsf{mTg}}(\mathbf{v})$.

Let T_L, T_R be BTMs, with the same number of nodes. We say (T_L, T_R) is a *pair of twin binary trees with multiplicities* (pair of twin BTMs), if for all i,

(i) the i-th node of T_L and the i-th node of T_R have the same multiplicities;

(ii) if the i-th node of T_L has multiplicity 1 and has a left (resp. right) child then the i-th node of T_R does not have a left (resp. right) child.

Notice that condition (ii) is equivalent to saying that if r_i has multiplicity 1 then if r_i has a left (resp. right) child then l_i does not have a left (resp. right) child.

Let T_L, T_R be BSTMs. We say (T_L, T_R) is a *pair of twin binary search trees with multiplicities* (pair of twin BSTMs) if $\mathrm{cont}(T_R) = \mathrm{cont}(T_L)$ and the shape of (T_L, T_R) is a pair of twin BTMs. The P-symbol given in Example 4.1 is a pair of twin BSTMs. In fact, we have the following:

Proposition 4.3. *The* $\mathrm{P}_{\mathsf{mTg}}$*-symbol* $(\mathrm{P}^{\rightarrow}_{\mathsf{lTg}}(\mathbf{w}), \mathrm{P}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{w}))$ *of* $\mathbf{w}$ *is a pair of twin BSTMs, for any* $\mathbf{w} \in \mathbb{N}^*$.

Let (S_L, S_R) be a pair of (respectively) increasing and decreasing binary trees over sets, with the same number of nodes. We say (S_L, S_R) are a *pair of twin binary trees over sets* (pair of twin BTSs) if, for all i,

(i) the i-th node of S_L has the same label as the i-th node of S_R;

(ii) if the i-th node of S_L is labelled by a set of cardinality 1 and has a left (resp. right) child, then the i-th node of S_R does not have a left (resp. right) child.

The Q-symbol given in Example 4.1 is a pair of twin BTSs. In fact, we have the following:

Proposition 4.4. *The meet-taiga* Q*-symbol* $(\mathrm{Q}^{\rightarrow}_{\mathsf{lTg}}(\mathbf{w}), \mathrm{Q}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{w}))$ *of* $\mathbf{w}$ *is a pair of twin BTSs, for any* $\mathbf{w} \in \mathbb{N}^*$.

Now, we are able to state our analogue of the Robinson–Schensted correspondence for the meet-taiga case:

Theorem 4.5. *The map* $\mathbf{w} \mapsto (\mathrm{P}_{\mathsf{mTg}}(\mathbf{w}), \mathrm{Q}_{\mathsf{mTg}}(\mathbf{w}))$ *is a bijection between the elements of* $\mathbb{N}^*$ *and the set formed by the pairs* $((T_L, T_R), (S_L, S_R))$ *where*

(i) (T_L, T_R) *is a pair of twin BSTMs;*

(ii) (S_L, S_R) *is a pair of twin BTSs such that the union of the sets labelling* S_L*, and therefore* S_R*, is the interval* $[m]$*, where* m *is the sum of the multiplicities of* T_L *(or* T_R*);*

(iii) (T_L, T_R) *and* (S_L, S_R) *have the same underlying pair of binary trees shape;*

(iv) the multiplicity of the i*-th node of* T_L *(resp.* T_R*) is the cardinality of the set labelling the* i*-th node of* S_L *(resp.* S_R*).*

4.2 Extraction algorithm

In the previous subsection, we have shown how to obtain a word from its meet-taiga P and Q-symbols. We now show how to obtain (possibly several) words from P-symbols alone, without requiring a Q-symbol.

Algorithm EmTg takes a pair of twin BSTMs (T_L, T_R) and outputs a word in the $\equiv_{\mathsf{mTg}}$-class corresponding to (T_L, T_R). Notice that it is a non-deterministic algorithm, since there may be several choices for a in steps 5 and 14.

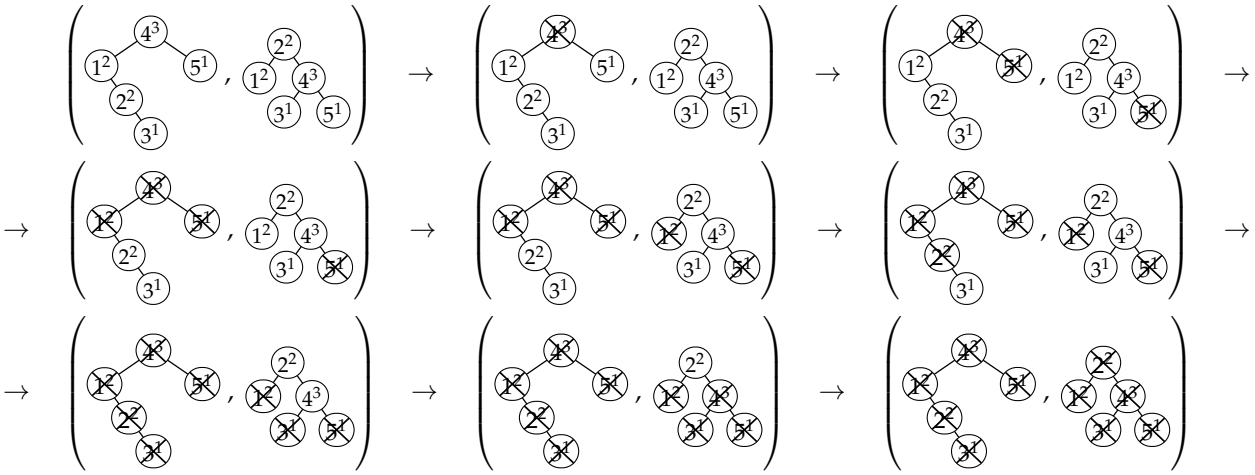
Proposition 4.6. *For any pair of twin BSTMs* (T_L, T_R) *as input, Algorithm* EmTg *computes a word belonging to the* $\equiv_{\mathsf{mTg}}$*-equivalence class encoded by* (T_L, T_R).

Algorithm 3: *Extract Meet-Taiga* (EmTg).

Input: A pair of twin BSTMs (T_L, T_R).
Output: A word in the $\equiv_{\mathsf{mTg}}$-class corresponding to (T_L, T_R).

let $\mathbf{w} := \varepsilon$ and $(U_L, U_R) := (T_L, T_R)$;
while $(U_L, U_R) \neq (\bot, \bot)$ **do**
  if *all leaves in* U_R *share letters in their labels with nodes in* U_L **then**
    choose a letter a that labels the root node of a tree in U_L, such that a has multiplicity greater than 1 or a labels a leaf in U_R, and let m_a be the multiplicity of a;
    if $m_a > 1$ **then** let $\mathbf{w} := \mathbf{w}a^{m_a - 1}$;
    else let $\mathbf{w} := \mathbf{w}a$ and remove the node labelled a from U_R;
    remove the node labelled a from U_L;
  else let a be the letter that labels a leaf in U_R such that a does not label any node in U_L, let $\mathbf{w} := \mathbf{w}a$ and remove the node labelled a from U_R;
return the word $\mathbf{w}$.

Example 4.7. Applying EmTg to the meet-taiga P-symbol of 451423412 yields the word 445112342:



The sequence of words obtained in each step of the algorithm is 44, 445, 4451, 44511, 445112, 4451123, 44511234 and lastly 445112342. In this example, we chose to remove the node labelled 5 before the node labelled 1.

As a consequence of Propositions 4.2, 4.3 and 4.6, we have the following:

Corollary 4.8. *For any $n, m \geq 0$, there is a bijection between the set of* mTg*-equivalence classes of words of length m over $[n]^*$ and pairs of twin BSTMs, labelled by letters from $[n]$ and whose sums of multiplicities is m.*

4.3 Iterative insertion algorithm

We now introduce an iterative version of our insertion algorithm which allow us to compute a pair of twin BSTMs from a word, while reading it in only one direction. Thus, we can compute the product of two pairs of twin BSTMs by applying EmTg to the second pair and inserting the letters of the resulting word, from left to right, into the first pair.

Algorithm TgRI allows one to obtain a new BSTM from another one, with the root node labelled by the letter of our choice.

Algorithm 4: *Taiga Root Insertion* (TgRI).

Input: A BSTM T, a letter $a \in \mathbb{N}$.
Output: A BSTM $a \downarrow T$.

1 let $T_{<a}$ (resp. $T_{>a}$) be the tree of all nodes of T labelled by letters $< a$ (resp. $> a$), such that a node x is a descendant of a node y in $T_{<a}$ (resp. $T_{>a}$) only if x is a descendant of y in T;
2 let $a \downarrow T$ be the tree with root node labelled a with multiplicity $|T|_a + 1$, with left subtree $T_{<a}$ and right subtree $T_{>a}$;
3 **return** the resulting tree $a \downarrow T$.

As before in Subsection 2.2, using Algorithm TgRI, one can compute a unique BSTM from a word $\mathbf{w}$: starting from the empty tree, read $\mathbf{w}$ from left to right (resp. right to left) and insert its letters one-by-one into the tree. The resulting BSTM is denoted by $\mathrm{P}^{\rightarrow}_{\mathsf{rTg}}(\mathbf{w})$ (resp. $\mathrm{P}^{\leftarrow}_{\mathsf{lTg}}(\mathbf{w})$). The proof of [4, Lemma 4.18] can be easily adapted to show the following:

Lemma 4.9. *Let* $\mathbf{w} \in \mathbb{N}^*$. *Then* $\mathrm{P}^{\rightarrow}_{\mathsf{rTg}}(\mathbf{w}) = \mathrm{P}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{w})$ *and* $\mathrm{P}^{\leftarrow}_{\mathsf{lTg}}(\mathbf{w}) = \mathrm{P}^{\rightarrow}_{\mathsf{lTg}}(\mathbf{w})$.

As such, we can compute the P-symbol of a meet-taiga class by reading a word in only one direction and applying Algorithms TgLI and TgRI at the same time:

Corollary 4.10. *For any* $\mathbf{w} \in \mathbb{N}^*$, *we have that*

$$\mathrm{P}_{\mathsf{mTg}}(\mathbf{w}) = \left(\mathrm{P}^{\leftarrow}_{\mathsf{lTg}}(\mathbf{w}), \mathrm{P}^{\leftarrow}_{\mathsf{rTg}}(\mathbf{w})\right) = \left(\mathrm{P}^{\rightarrow}_{\mathsf{lTg}}(\mathbf{w}), \mathrm{P}^{\rightarrow}_{\mathsf{rTg}}(\mathbf{w})\right).$$

We can now define the *left-to-right iterative meet-taiga* P*-symbol* of a word $\mathbf{w}$ as the pair $(\mathrm{P}^{\rightarrow}_{\mathsf{lTg}}(\mathbf{w}), \mathrm{P}^{\rightarrow}_{\mathsf{rTg}}(\mathbf{w}))$, obtained by reading $\mathbf{w}$ from left to right and iteratively inserting its letters into $(\bot, \bot)$, using Algorithms TgLI and TgRI for $\mathrm{P}^{\rightarrow}_{\mathsf{lTg}}(\mathbf{w})$ and $\mathrm{P}^{\rightarrow}_{\mathsf{rTg}}(\mathbf{w})$, respectively. Similarly, we define the *left-to-right iterative meet-taiga* Q*-symbol* of $\mathbf{w}$ as the pair $(\mathrm{Q}^{\rightarrow}_{\mathsf{lTg}}(\mathbf{w}), \mathrm{Q}^{\rightarrow}_{\mathsf{rTg}}(\mathbf{w}))$, of the same shape as $\mathrm{P}_{\mathsf{mTg}}(\mathbf{w})$, where each node is labelled by the positions in $\mathbf{w}$ of the letter in its corresponding node in $\mathrm{P}_{\mathsf{mTg}}(\mathbf{w})$. In other words, each

node in $\mathrm{Q}_{\mathsf{mTg}}(\mathbf{w})$ is labelled by the steps in which its corresponding node in $\mathrm{P}_{\mathsf{mTg}}(\mathbf{w})$ was created or had its multiplicity incremented, when applying the left-to-right iterative insertion algorithm.

The correctness of the iterative insertion algorithm follows from Corollary 4.10. Thus, we obtain an insertion algorithm in line with the usual Robinson–Schensted correspondence algorithms. We can also define a right-to-left version of the iterative insertion algorithm.

5 Counting in the taiga monoids

We now obtain 'hook-length'-like formulas for the sizes of classes of words under the various taiga congruences. We first give the result for the left-taiga case:

Proposition 5.1 ([10, Proposition 5])**.** *Let* $\mathbf{w} \in \mathbb{N}^*$*. Let* $\mathrm{supp}(\mathbf{w}) = \{x_1 < \cdots < x_k\}$ *and let* m_i *be the sum of the multiplicities of the node labelled* x_i *and its descendants in* $\mathrm{P}^{\rightarrow}_{\mathsf{lTg}}(\mathbf{w})$*, for all* $1 \leq i \leq k$*. Then, there are*

$$\frac{|\mathbf{w}|!}{\prod_{i=1}^{k}(|\mathbf{w}|_{x_i}-1)! \cdot m_i}$$

words over $\mathbb{N}$ *in the same* $\equiv_{\mathsf{lTg}}$*-class of* $\mathbf{w}$*.*

By Proposition 3.2, we can obtain the size of a $\equiv_{\mathsf{jTg}}$-class, by computing the size of specific $\equiv_{\mathsf{hypo}}$-classes, which are known to be efficiently computable [3, Theorem 3].

Proposition 5.2. *Let* $\mathbf{w} \in \mathbb{N}^*$*. Let* $\mathrm{supp}(\mathbf{w}) = \{x_1 < \cdots < x_k\}$ *and* $A_1, \ldots, A_l$ *be all the intervals of* $\mathrm{supp}(\mathbf{w})$ *such that* A_j *only contains simple letters and* $A_j \cup \{a\}$ *is not an interval, for any simple letter* $a \notin A_j$*. Let* $h(\mathbf{w}[A_j])$ *be the size of the hypoplactic class of* $\mathbf{w}[A_j]$*. Then, there are*

$$\frac{|\mathbf{w}|!}{\prod_{i=1}^{k}|\mathbf{w}|_{x_i}!}\prod_{j=1}^{l}\frac{h(\mathbf{w}[A_j])}{|A_j|!}$$

words over $\mathbb{N}$ *in the same* $\equiv_{\mathsf{jTg}}$*-class of* $\mathbf{w}$*.*

The formula for the meet-taiga case is recursive, given by an algorithm that counts linear extensions of specific posets. Due to the extensive notation required to write it, we omit it here. With it, we can compute the size of $\equiv_{\mathsf{mTg}}$-classes in $\mathcal{O}(n^{2k-2}(k!)^3)$ operations, where n is the length of the words and k is the size of their support.

For our last result, given a BSTM, we obtain bounds for how many distinct BSTMs form a pair of twin BSTMs with it.

Proposition 5.3. *Let* T_L *be a BSTM with* n *nodes. Suppose there are* k *simple nodes in* T_L *that are not leaves. Let* $R(T_L)$ *denote the number of distinct BSTMs* T_R *such that* (T_L, T_R) *is a pair of twin BSTMs. Then,*

$$C_{n-k} \leq R(T_L) \leq C_n$$

where $C_m = \frac{(2m)!}{(m+1)!m!}$ *is the m-th Catalan number.*

Acknowledgements

The authors would like to thank Alan Cain and António Malheiro for their suggestions and helpful comments.

References

[1] T. Aird and D. Ribeiro. "Plactic-like monoids arising from meets and joins of stalactic and taiga congruences". *J. Algebra* **660** (2024), pp. 795–851. DOI.

[2] A. Cain and A. Malheiro. "Identities in plactic, hypoplactic, sylvester, Baxter, and related monoids". *Electron. J. Combin.* **25**.3 (2018), Paper No. 3.30, 19. DOI.

[3] A. J. Cain and A. Malheiro. "Crystallizing the hypoplactic monoid: from quasi-Kashiwara operators to the Robinson-Schensted-Knuth-type correspondence for quasi-ribbon tableaux". *J. Algebraic Combin.* **45**.2 (2017), pp. 475–524. DOI.

[4] S. Giraudo. "Algebraic and combinatorial structures on pairs of twin binary trees". *J. Algebra* **360** (2012), pp. 115–157. DOI.

[5] F. Hivert, J.-C. Novelli, and J.-Y. Thibon. "The algebra of binary search trees". *Theoret. Comput. Sci.* **339**.1 (2005), pp. 129–165. DOI.

[6] F. Hivert, J.-C. Novelli, and J.-Y. Thibon. "Commutative combinatorial Hopf algebras". *J. Algebr. Comb.* **28**.1 (2008), pp. 65–95. DOI.

[7] J. M. Howie. *Fundamentals of semigroup theory*. Vol. 12. London Mathematical Society Monographs. New Series. Oxford Science Publications. The Clarendon Press, Oxford University Press, New York, 1995, pp. x+351.

[8] A. Lascoux and M.-P. Schützenberger. "Le monoïde plaxique". *Noncommutative Structures in Algebra and Geometric Combinatorics: Proceedings of the Colloquium Held at Arco Felice, Naples, July 24–26, 1978.* Ed. by A. De Luca. Vol. 109. Quad. "Ricerca Sci." CNR, Rome, 1981, pp. 129–156.

[9] J.-C. Novelli. "On the hypoplactic monoid". *Discrete Math.* **217**.1-3 (2000). Formal power series and algebraic combinatorics (Vienna, 1997), pp. 315–336. DOI.

[10] J.-B. Priez. "Lattice of combinatorial Hopf algebras: binary trees with multiplicities". *25th International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC 2013)*. Ed. by A. Goupil and G. Schaeffer. Vol. AS. DMTCS Proceedings. Paris, France: Discrete Mathematics and Theoretical Computer Science, 2013, pp. 1137–1148. DOI.