View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Repositório da Universidade Nova de Lisboa

COMBINATORICS OF CYCLIC SHIFTS IN PLACTIC, HYPOPLACTIC, SYLVESTER, BAXTER, AND RELATED MONOIDS

ALAN J. CAIN AND ANTÓNIO MALHEIRO

Abstract. The cyclic shift graph of a monoid is the graph whose vertices are elements of the monoid and whose edges link elements that differ by a cyclic shift. This paper examines the cyclic shift graphs of 'plactic-like' monoids, whose elements can be viewed as combinatorial objects of some type: aside from the plactic monoid itself (the monoid of Young tableaux), examples include the hypoplactic monoid (quasi-ribbon tableaux), the sylvester monoid (binary search trees), the stalactic monoid (stalactic tableaux), the taiga monoid (binary search trees with multiplicities), and the Baxter monoid (pairs of twin binary search trees). It was already known that for many of these monoids, connected components of the cyclic shift graph consist of elements that have the same evaluation (that is, contain the same number of each generating symbol). This paper focusses on the maximum diameter of a connected component of the cyclic shift graph of these monoids in the rank-n case. For the hypoplactic monoid, this is $n-1$; for the sylvester and taiga monoids, at least $n-1$ and at most n; for the stalactic monoid, 3 (except for ranks 1 and 2, when it is respectively 0 and 1); for the plactic monoid, at least $n-1$ and at most $2n-3$. The current state of knowledge, including new and previously-known results, is summarized in a table.

Contents

The first author was supported by an Investigador FCT fellowship (IF/01622/2013/CP1161/CT0001).

For both authors, this work was partially supported by by the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through the project UID/MAT/00297/2019 (Centro de Matemática e Aplicações) and the projects PTDC/MHC-FIL/2583/2014 and PTDC/MAT-PUR/31174/2017.

1. Introduction

In a monoid M, two elements s and t are related by a cyclic shift, denoted $s \sim t$, if and only if there exist $x, y \in M$ such that $s = xy$ and $t = yx$. In the plactic monoid (the monoid of Young tableaux, here denoted plac), elements that have the same evaluation (that is, elements that contain the same number of each generating symbol) can be obtained from each other by iterated application of cyclic shifts [1, § 4]. Furthermore, in the plactic monoid of rank n (denoted plac_n), it is known that $2n - 2$ applications of cyclic shifts are sufficient [2, Theorem 17].

To restate these results in a new form, define the *cyclic shift graph* $K(M)$ of a monoid M to be the undirected graph with vertex set M and, for all $s, t \in M$, an edge between s and t if and only if $s \sim t$. Connected components of $K(M)$ are $\sim^*$-classes (where $\sim^*$ is the reflexive and transitive closure of $\sim$), since they consist of elements that are related by iterated cyclic shifts. Thus the results discussed above say that each connected component of $K(\mathsf{plac})$ consists of precisely the elements with a given evaluation, and that the diameter of a connected component of $K(\mathsf{plac}_n)$ is at most $2n - 2$. Note that there is no bound on the number of elements in a connected component, despite there being a bound on diameters that is dependent only on the rank.

This paper studies the cyclic shift graph for analogues of the plactic monoid in which other combinatorial objects have the role that Young tableaux play for the plactic monoid. For each monoid there are two central questions, motivated by the results for the plactic monoid: (i) whether connected components consist of precisely the elements with a given evaluation; (ii) what the maximum diameter of a connected component is in the rank n case.

The monoids considered are the plactic monoid, which is celebrated for its ubiquity, arising in such diverse contexts as symmetric functions [3], representation theory and algebraic combinatorics [4, 5], and musical theory [6]; the hypoplactic monoid, whose elements are quasi-ribbon tableaux and which arises in the theory of quasi-symmetric functions [7, 8, 9]; the sylvester monoid, whose elements are binary search trees [10]; the taiga monoid, whose elements are binary search trees with multiplicities [11]; the stalactic monoid, whose elements are stalactic tableaux [12, 11]; and the Baxter monoid, whose elements are pairs of twin binary search trees [13, 14], and which is linked

TABLE 1. Monoids and corresponding combinatorial objects.

Monoid	*Sym.*	*Combinatorial object*	*See*
Plactic	plac	Young tableau	§ 4
Hypoplactic	hypo	Quasi-ribbon tableau	§ 5
Sylvester	sylv	Binary search tree	§ 6
Stalactic	stal	Stalactic tableau	§ 7
Taiga	taig	Binary search tree with multiplicities	§ 8
Baxter	baxt	Pair of twin binary search trees	§ 9

to the theory of Baxter permutations. (See Table 1.) Each of these monoids arises by factoring the free monoid $\mathcal{A}^*$ over the ordered alphabet $\mathcal{A} = \{1 < 2 < \ldots\}$ by a congruence $\equiv$ that can be defined in two equivalent ways:

- C1 *Insertion.* $\equiv$ relates those words that yield the same combinatorial object as the result of some insertion algorithm.
- C2 *Defining relations.* $\equiv$ is defined to be the congruence generated by some set of defining relations $\mathcal{R}$.

Each of these monoids also has a rank-n version (where $n \in \mathbb{N}$), which arises by factoring the free monoid $\mathcal{A}_n^*$ over the finite ordered alphabet $\mathcal{A}_n = \{1 < 2 < \ldots < n\}$ by the natural restriction of $\equiv$. Each of these monoids is discussed in its own section, and the equivalent definitions will be recalled at the start of the relevant section. For the present, note that these monoids are *multihomogeneous*: if two words over $\mathcal{A}^*$ represent the same element of the monoid (that is, are related by $\equiv$) then they have the same evaluation (that is, they contain the same number of each symbol in $\mathcal{A}$ and, in particular, have the same length). Thus it is sensible to consider the evaluation of an *element* of the monoid to be the evaluation of any word that represents it. The relation $\equiv_{\mathrm{ev}}$ holds between elements that have the same evaluation; clearly $\equiv_{\mathrm{ev}}$ is an equivalence relation and $\equiv_{\mathrm{ev}}$-classes are finite.

This paper shows that in the cyclic shift graph of the hypoplactic, sylvester, and taiga monoids, each connected component does consist of precisely those elements with a given evaluation. In the case of the stalactic monoid, an alternative characterization is given of when two elements lie in the same connected component. (The result for the sylvester monoid was previously proved in a different way by the present authors [15, Theorem 3.4].)

Furthermore, just as for the rank-n plactic monoid, there is a bound on the maximum diameters of connected components in the cyclic shift graphs of the rank-n hypoplactic, sylvester, and taiga monoids, and this bound is only dependent on n. It is worth emphasizing how remarkable this is: although there is no global bound on the number of elements in a component (or, equivalently, which have the same evaluation), any two elements in the same component are related by a number of cyclic shifts that is dependent only on n. Table 2 shows the current state of knowledge for all the monoids considered in this paper. All the exact values and bounds shown in this table are new results (although the upper bound of $2n-3$ in the case of the plactic monoid follows by a minor modification of the reasoning that yields the Choffrut–Mercaş bound of $2n-2$).

TABLE 2. Maximum diameter of a connected component of cyclic shift graph for rank-n monoids.

		Maximum diameter			
				Known bounds	
Monoid	$\sim^* = \equiv_{\mathrm{ev}}$	*Known value*	*Conjecture*	*Lower*	*Upper*
plac_n	Y	?	$n-1$	$n-1$	$2n-3$
hypo_n	Y	$n-1$	—	—	—
sylv_n	Y	?	$n-1$	$n-1$	n
stal_n	N	$\begin{cases} n-1 & \text{if } n<3 \\ 3 & \text{if } n\geq 3 \end{cases}$	—	—	—
taig_n	Y	?	$n-1$	$n-1$	n
baxt_n	N	?	?	?	?

Experimentation using the computer algebra software Sage [16] strongly suggests that in the cases of the rank-n plactic, sylvester, and taiga monoids, the maximum diameter of a connected component is $n-1$.

Also, although the monoids considered are all multihomogeneous, Section 3 exhibits a rank 4 multihomogeneous monoid for which there is no bound on the diameter of connected components in its cyclic shift graph. Thus, the bound on diameters is *not* a general property of multihomogeneous monoids: rather, it seems to dependent on the underlying combinatorial objects. This also is of interest because cyclic shifts are a possible generalization of conjugacy from groups to monoids; thus the combinatorial objects are here linked closely to the algebraic structure of the monoid.

[The results in this article have already been announced in the conference paper [17].]

2. PRELIMINARIES

2.1. **Alphabets and words.** This subsection recalls some terminology and fixes notation for presentations. For background on the free monoid, see [18]; for semigroup presentations, see [19, 20].

For any alphabet X, the free monoid (that is, the set of all words, including the empty word) on the alphabet X is denoted X^*. The empty word is denoted ε. For any $u \in X^*$, the length of u is denoted $|u|$, and, for any $x \in X$, the number of times the symbol x appears in u is denoted $|u|_x$.

The *evaluation* (also called the *content*) of a word $u \in X^*$, denoted $\mathrm{ev}(u)$, is the $|X|$-tuple of non-negative integers, indexed by X, whose x-th element is $|u|_x$; thus this tuple describes the number of each symbol in X that appears in u. If two words $u, v \in X^*$ have the same evaluation, this is denoted $u \equiv_{\mathrm{ev}} v$. Notice that $\equiv_{\mathrm{ev}}$ is an equivalence relation (and indeed a congruence). Note further that there are clearly only finitely many words with a given evaluation, and so $\equiv_{\mathrm{ev}}$-classes are finite.

When X represents a generating set for a monoid M, every element of X^* can be interpreted either as a word or as an element of M. For words $u, v \in X^*$, write $u = v$ to indicate that u and v are equal as words and

$u \equiv_M v$ to denote that u and v represent the same element of the monoid M. A presentation is a pair $\langle X \,|\, \mathcal{R} \rangle$ where $\mathcal{R}$ is a binary relation on X^*, which defines [any monoid isomorphic to] $X^*/\mathcal{R}^\#$, where $\mathcal{R}^\#$ denotes the congruence generated by $\mathcal{R}$.

The presentation $\langle X \,|\, \mathcal{R} \rangle$ is *homogeneous* (respectively, *multihomogeneous*) if for every $(u, v) \in \mathcal{R}$ we have $|u| = |v|$ (respectively, $u \equiv_{\mathrm{ev}} v$). That is, in a homogeneous presentation, defining relations preserve length of words; in a multihomogenous presentation, defining relations preserve evaluations. A monoid is *homogeneous* (respectively, *multihomogeneous*) if it admits a homogeneous (respectively, multihomogeneous) presentation. Suppose M is a multihomogeneous monoid defined by a multihomogeneous presentation $\langle X \,|\, \mathcal{R} \rangle$. Since every word in X^* that represents a given element of M has the same evaluation, it makes sense to define the evaluation of an element of M to be the evaluation of any word representing it, and to write $s \equiv_{\mathrm{ev}} t$ if $s, t \in M$ have the same evaluation.

2.2. **'Plactic-like' monoids.** Throughout the paper, $\mathcal{A}$ denotes the infinite ordered alphabet $\{1 < 2 < \ldots\}$ (that is, the set of natural numbers, viewed as an alphabet), $\mathcal{A}_n$ the finite ordered alphabet $\{1 < 2 < \ldots < n\}$ (that is, the first n natural numbers, viewed as an alphabet). A word $u \in \mathcal{A}_n^*$ is *standard* if $u \equiv_{\mathrm{ev}} 123 \cdots |u|$. That is, u is standard if it contains each symbol in $\{1, \ldots, |u|\}$ exactly once.

This paper is mainly concerned with 'plactic-like' monoids, whose elements can be identified with some kind of combinatorial object. Each such monoid M has an associated insertion algorithm, which takes a combinatorial object of the relevant type and a letter of the alphabet $\mathcal{A}$ and computes a new combinatorial object. Thus one can compute from a word $u \in \mathcal{A}^*$ a combinatorial object $\mathrm{P}_{\mathsf{M}}(u)$ of the type associated to M by starting with the empty combinatorial object and inserting the symbols of u one-by-one using the appropriate insertion algorithm and proceeding through the word u either left-to-right or right-to-left. (The procedure is slightly different for the Baxter monoid; this will be discussed in Section 9.) One then defines a relation $\equiv_{\mathsf{M}}$ as the kernel of the map $u \mapsto \mathrm{P}_{\mathsf{M}}(u)$. In each case, the relation $\equiv_{\mathsf{M}}$ is a congruence, and M is the factor monoid $\mathcal{A}^*/\equiv_{\mathsf{M}}$; the rank-$n$ analogue is the factor monoid $\mathcal{A}_n^*/\equiv_{\mathsf{M}}$, where $\equiv_{\mathsf{M}}$ is naturally restricted to $\mathcal{A}_n^* \times \mathcal{A}_n^*$. Since each element of M is an equivalence class of words that give the same combinatorial object, elements of M can be identified with the corresponding combinatorial objects.

Each of the combinatorial objects and insertion algorithms considered in this paper is such that the number of each symbol from $\mathcal{A}$ in the word u is the same as the number of symbols $\mathcal{A}$ in the object $\mathrm{P}_{\mathsf{M}}(u)$. It follows that each of the corresponding monoids is multihomogeneous, and it makes sense to define an element of a rank-n monoid to be *standard* if it is represented by a standard word (and thus *only* by standard words).

2.3. **Cyclic shifts.** Recall that two elements $s, t \in M$ are related by a cyclic shift, denoted $s \sim t$, if and only if there exist $x, y \in M$ such that $s = xy$ and $t = yx$. If X represents a generating set for M, then $s \sim t$ if and only if there exist $u, v \in X^*$ such that uv represents s and vu represents t. Notice that

the relation $\sim$ is reflexive (because it is possible that x or y in the definition of $\sim$ can be the identity) and symmetric. For $k \in \mathbb{N}$, let $\sim^k$ be the k-fold composition of the relation $\sim$: that is,

$$\sim^k = \underbrace{\sim \circ \sim \circ \ldots \circ \sim}_{k \text{ times}}.$$

Note that $\sim^*$, the transitive closure of $\sim$, is $\bigcup_{k=1}^{\infty} \sim^k$. Note further that $\sim^k \subseteq \sim^{k+1}$ since $\sim$ is reflexive. Thus $\sim^k$ relates elements of M that differ by at most k cyclic shifts.

The following result is immediate from the definition of $\sim$:

Lemma 2.1. *In any multihomogeneous monoid,* $\sim^* \subseteq \equiv_{\mathrm{ev}}$.

For any monoid M, define the *cyclic shift graph* $K(M)$ to be the undirected graph with vertex set M and, for all $s, t \in M$, an edge between s and t if and only if $s \sim t$. (See [21] for graph-theoretical definitions and terminology.) Two elements $s, t \in M$ are a distance at most k apart in $K(M)$ if and only if $s \sim^k t$. Connected components of $K(M)$ are $\sim^*$-classes, since they consist of elements that are related by iterated cyclic shifts. The connected component of $K(M)$ containing an element $s \in M$ is denoted $K(M, s)$. If M is multihomogenous, $\sim^* \subseteq \equiv_{\mathrm{ev}}$, and thus connected components of $K(M)$ are finite.

Since $\sim$ is reflexive, there is a loop at every vertex of $K(M)$. Throughout the paper, illustrations of graphs $K(M)$ will, for clarity, omit these loops.

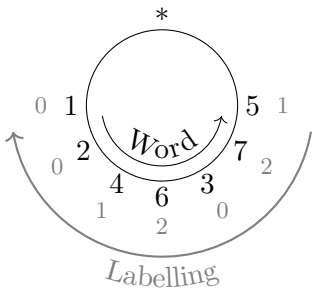
2.4. **Cocharge sequences.** This subsection introduces 'cocharge sequences', which will be used in several places to establish lower bounds on the maximum diameters of connected components of the cyclic shift graphs of finite-rank monoids.

Let $u \in \mathcal{A}^*$ be a standard word. The *cocharge sequence* of u, denoted $\mathrm{cochseq}(u)$, is a sequence (of length $|u|$) calculated from u as follows:

(1) Draw a circle, place a point $*$ somewhere on its circumference, and, starting from $*$, write u anticlockwise around the circle.
(2) Label the symbol 1 with 0.
(3) Iteratively, after labelling some i with k, proceed clockwise from i to the symbol $i + 1$:
    - if the symbol $i + 1$ is reached *before* $*$, label $i + 1$ by $k + 1$;
    - if the symbol $i + 1$ is reached *after* $*$, label $i + 1$ by k.
(4) The sequence $\mathrm{cochseq}(u)$ is the sequence whose i-th term is the label of i.

Note that at steps 2 and 3, the symbols 1 and $i + 1$ are known to be in u because u is a standard word.

For example, for the word 1246375, the labelling process gives:



and it follows that cochseq$(u) = (0, 0, 0, 1, 1, 2, 2)$. Notice that the first term of a cocharge sequence is always 0, and that each term in the sequence is either the same as its predecessor or greater by 1. Thus the i-th term in the sequence always lies in the set $\{0, 1, \ldots, i-1\}$.

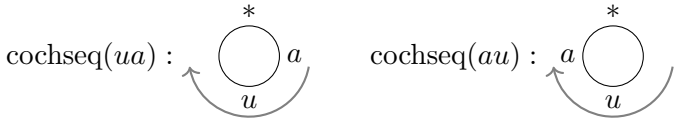
The usual notion of cocharge is obtained by summing the cocharge sequence (see [5, § 5.6]). Note, however, that cocharge is defined for all words, whereas this section defines the cocharge sequence only for standard words.

Lemma 2.2. (1) *Let $u \in \mathcal{A}^*$ and $a \in \mathcal{A}_n \setminus \{1\}$ be such that ua is a standard word. Then* cochseq(ua) *is obtained from* cochseq(au) *by adding* 1 *to the a-th component.*

(2) *Let $xy \in \mathcal{A}^*$ be a standard word such that x does not contain the symbol* 1*. Then* cochseq(yx) *is obtained from* cochseq(xy) *by adding* 1 *to the a-th component for each symbol a that appears in x.*

(3) *Let $xy \in \mathcal{A}^*$ be a standard word such that y does not contain the symbol* 1*. Then* cochseq(yx) *is obtained from* cochseq(xy) *by subtracting* 1 *from the a-th component for each symbol a that appears in y.*

Proof. Consider how a is labelled during the calculation of cochseq(ua) and cochseq(au):



In the calculation of cochseq(ua), the symbol $a-1$ receives a label k, and then a is reached *after* $*$ is passed; hence a also receives the label k. (If a symbol $a+1$ is present, it receives the label $a+1$.) In the calculation of cochseq(au), the symbols $1, \ldots, a-1$ receive the same labels as they do in the calculation of cochseq(ua), but after labelling $a-1$ by k the symbol a is reached *before* $*$ is passed; hence a receives the label $k+1$ (and if a symbol $a+1$ is present, it also receives the label $k+1$ since it is reached after $*$ is passed); after this point, labelling proceeds in the same way. Parts 2) and 3) are now immediate consequences of part 1). □

3. General multihomogeneous monoids

In order to set in context the results below on the diameters of connected components of cyclic shift graphs, this section gives an example of a multihomogeneous monoid for which the connected components of the cyclic shift graph have unbounded diameter. This shows that the results for the 'plactic-like' monoids discussed in the rest of the paper are not simply consequences of some more general result that holds for all multihomogeneous monoids.

Example 3.1. Let M be the monoid defined by the presentation

$$\langle a, b, x, y \,|\, (bxy, xyb), (byx, yxb), (axyb, byxa)\rangle.$$

Notice that M is multihomogeneous. Let $\alpha \in \mathbb{N}$. Then

$$\begin{aligned} a(xy)^\alpha b &\equiv_M axyb(xy)^{\alpha-1} \\ &\equiv_M byxa(xy)^{\alpha-1} \\ &\sim yxa(xy)^{\alpha-1}b \\ &\equiv_M yxaxyb(xy)^{\alpha-2} \\ &\equiv_M yxbyxa(xy)^{\alpha-2} \\ &\equiv_M b(yx)^2a(xy)^{\alpha-2} \\ &\sim (yx)^2a(xy)^{\alpha-2}b \\ &\vdots \\ &\sim b(yx)^\alpha a \end{aligned}$$

Thus the elements $a(xy)^\alpha b$ and $b(yx)^\alpha a$ lie in the same connected component of $K(M)$. The aim is now to prove that the distance between $a(xy)^\alpha b$ and $b(yx)^\alpha a$ in $K(M)$ is at least α. This necessitates defining an invariant, reminiscent of cocharge, but tailored specifically to the monoid M.

Let $L = \left\{ u \in \{a, b, xy, yx\}^* : |u|_a = |u|_b = 1 \right\}$. Define a map $\mu : L \to \mathbb{N} \cup \{0\}$, where $\mu(u)$ is calculated as follows:

(1) Draw a circle, place a point $*$ somewhere on itse circumference, and write u anticlockwise aroung the circle.
(2) Temporarily ignoring the symbol b, let $k(u)$ be the number of consecutive cyclic factors xy following a. (Equivalently, let $k(u)$ be the number of consecutive words xy (ignoring b) following the symbol a, proceeding anticlockwise around the circle.)
(3) Let $\mu(u)$ be $k(u)$ if starting from a and proceeding anticlockwise, one encounters b before $*$, and otherwise let $\mu(u)$ be $k(u) + 1$.

It is now necessary to show that $\mu(\cdot)$ is invariant under applications of defining relations. Clearly, the set L is closed under applying defining relations. Further, applying a defining relation (bxy, xyb) or (byx, yxb) does not alter $k(\cdot)$, and does not alter the relative positions of a, b, and $*$. So applying these defining relations does not alter $\mu(\cdot)$. So suppose that $u, v \in L$ differ by a single application of a defining relation $(axyb, byxa)$. Interchanging u and v if neceessary, suppose $u = paxybq$ and $v = pbyxaq$. Let $m \in \mathbb{N} \cup \{0\}$ be maximal such that $(xy)^m$ is a prefix of qp; thus either $qp = (xy)^m$ or

$qp = (xy)^m yxw$ for some $w \in \{xy, yx\}^*$. Applying the procedure above to u and v, one sees that $k(u) = m + 1 = k(v) + 1$, but b is encountered before $*$ for u but not for v; hence $\mu(u) = k(u) = k(v) + 1 = \mu(v)$.

Thus if two words in L represent the same element of M, they have the same image under μ. If $u, v \in L$ are such that $u \sim v$, then $k(u) = k(v)$, since applying $\sim$ does not alter the number of cyclic factors xy following a. Thus $\mu(u)$ and $\mu(v)$ differ by at most one. Hence, since $\mu(a(xy)^\alpha b) = \alpha$ and $\mu(b(yx)^\alpha a) = 1$, it follows that $a(xy)^\alpha b$ and $b(yx)^\alpha a$ are a distance at least $\alpha - 1$ apart in $K(M)$.

Since α was arbitrary, this shows that there is no bound on the diameters of connected components in $K(M)$.

4. Plactic monoid

This section recalls the essential facts about the plactic monoid; for proofs and further reading, see [5, Ch. 5].

A *Young tableau* is an array with rows left aligned and of non-increasing length from top to bottom, filled with symbols from $\mathcal{A}$ so that the entries in each row are non-decreasing from left to right, and the entries in each column are [strictly] increasing from top to bottom. An example of a Young tableau is

(4.1)

1	2	2	2	4
2	3	5		
4	4			
5	6			

.

The associated insertion algorithm is as follows:

Algorithm 4.1 (Schensted's algorithm)**.**
Input: A Young tableau T and a symbol $a \in \mathcal{A}$.
Output: A Young tableau $T \leftarrow a$.
Method:

(1) If a is greater than or equal to every entry in the topmost row of T, add a as an entry at the rightmost end of T and output the resulting tableau.
(2) Otherwise, let z be the leftmost entry in the top row of T that is strictly greater than a. Replace z by a in the topmost row and recursively insert z into the tableau formed by the rows of T below the topmost. (Note that the recursion may end with an insertion into an 'empty row' below the existing rows of T.)

Thus one can compute, for any word $u = u_1 \cdots u_k \in \mathcal{A}^*$, a Young tableau $\mathrm{P}_{\mathsf{plac}}(u)$ by starting with an empty tableau and successively inserting the symbols of u, proceeding left-to-right through the word. Define the relation $\equiv_{\mathsf{plac}}$ by

$$u \equiv_{\mathsf{plac}} v \iff \mathrm{P}_{\mathsf{plac}}(u) = \mathrm{P}_{\mathsf{plac}}(v)$$

for all $u, v \in \mathcal{A}^*$. The relation $\equiv_{\mathsf{plac}}$ is a congruence, and the *plactic monoid*, denoted plac, is the factor monoid $\mathcal{A}^*/\equiv_{\mathsf{plac}}$; the *plactic monoid of rank* n, denoted plac_n, is the factor monoid $\mathcal{A}_n^*/\equiv_{\mathsf{plac}}$ (with the natural restriction

of $\equiv_{\mathsf{plac}}$). Each element $[u]_{\equiv_{\mathsf{plac}}}$ (where $u \in \mathcal{A}^*$) can be identified with the Young tableau $\mathrm{P}_{\mathsf{plac}}(u)$.

The monoid plac is presented by $\langle \mathcal{A} \,|\, \mathcal{R}_{\mathsf{plac}} \rangle$, where

$$\begin{aligned}\mathcal{R}_{\mathsf{plac}} = {} & \{ (acb, cab) : a, b, c \in \mathcal{A}, a \leq b < c \} \\ & \cup \{ (bac, bca) : a, b, c \in \mathcal{A}, a < b \leq c \};\end{aligned}$$

the defining relations in $\mathcal{R}_{\mathsf{plac}}$ are often called the *Knuth relations* [5, § 5.2]. The monoid plac_n is presented by $\langle \mathcal{A}_n \,|\, \mathcal{R}_{\mathsf{plac}} \rangle$, where the set of defining relations $\mathcal{R}_{\mathsf{plac}}$ is naturally restricted to $\mathcal{A}_n^* \times \mathcal{A}_n^*$. Notice in particular that plac and plac_n are multihomogeneous.

Lascoux & Schützenberger proved that if two elements of plac have the same evaluation, then it is possible to transform one to the other using cyclic shifts [1, § 4]. In the terms of this paper, they proved that $\equiv_{\mathrm{ev}} \subseteq \sim^*$ in plac. Since the opposite inclusion holds in general, it follows that $\equiv_{\mathrm{ev}} = \sim^*$ in plac. Thus connected components of the cyclic shift graph $K(\mathsf{plac})$ are $\equiv_{\mathrm{ev}}$-classes.

Choffrut & Mercaş showed that if two elements of plac_n have the same evaluation, then it is possible to transform one to the other using at most $2n - 2$ cyclic shifts [2, Theorem 17]. Thus the maximum diameter of a connected component of $K(\mathsf{plac}_n)$ is at most $2n - 2$.

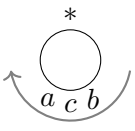
This section gives a lower bound on the maximum diameter of a connected component of $K(\mathsf{plac}_n)$, makes a slight improvement on the upper bound of Choffrut & Mercaş, and conjecture the exact value on the basis of experimentation using computer algebra software.

Establish the lower bound requires the use of cocharge sequences (defined in Subsection 2.4 above). It is first of all necessary to show that cocharge sequences are well-defined on standard elements of plac:

Proposition 4.2. *Let $w, w' \in \mathcal{A}^*$ be standard words such that $w \equiv_{\mathsf{plac}} w'$. Then* $\mathrm{cochseq}(w) = \mathrm{cochseq}(w')$.

Proof. It suffices to prove the result when w and w' differ by a single application of a defining relation. Assume that w and w' differ by an application of a defining relation $(acb, cab) \in \mathcal{R}_{\mathsf{plac}}$ where $a \leq b < c$. So $w = pacbq$ and $w' = pcabq$, where $p, q \in \mathcal{A}_n^*$ and $a, b, c \in \mathcal{A}_n$ with $a \leq b < c$. Since w and w' are standard words, $a < b$.

Consider how labels are assigned to the symbols a, b, and c when calculating $\mathrm{cochseq}(w)$ as described in Subsection 2.4:



Among these three symbols, a will receive a label first, then b, then c. Thus, after a, the labelling process will pass $*$ at least once to visit b and only then visit c. Thus interchanging a and c does not alter the resulting labelling. Hence $\mathrm{cochseq}(w) = \mathrm{cochseq}(w')$. Similar reasoning shows that if w and w' differ by an application of a defining relation $(bac, bca) \in \mathcal{R}_{\mathsf{plac}}$, then $\mathrm{cochseq}(w) = \mathrm{cochseq}(w')$. □

For any standard tableau T in plac, define $\mathrm{cochseq}(T)$ to be $\mathrm{cochseq}(u)$ for any standard word $u \in \mathcal{A}^*$ such that $T = \mathrm{P}(u)$. By Proposition 4.2, $\mathrm{cochseq}(T)$ is well-defined.

Proposition 4.3. (1) *Connected components of* $K(\mathsf{plac})$ *coincide with* $\equiv_{\mathrm{ev}}$*-classes of* plac.

(2) *Let* k_n *be the maximum diameter of a connected component of* $K(\mathsf{plac}_n)$. *Then* $n-1 \leq k_n \leq 2n-3$.

Proof. The first part is the result of Lascoux & Schützenberger [1, § 4], restated in the terms used of this paper. It remains to prove the second part.

To establish the lower bound on k_n, it is necessary to exhibit a pair of elements of plac_n that lie in the same connected component of $K(\mathsf{plac}_n)$ but are a distance at least $n-1$ apart.

Let $t = 12\cdots(n-1)n$ and $u = n(n-1)\cdots 21$, and let

$$T = \mathrm{P}_{\mathsf{plac}}(t) = \begin{array}{|c|c|c|c|}\hline 1 & 2 & \cdots & n \\ \hline \end{array} \quad \text{and } U = \mathrm{P}_{\mathsf{plac}}(u) = \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline \vdots \\ \hline n \\ \hline \end{array}$$

Since $T \equiv_{\mathrm{ev}} U$, the elements T and U are in the same connected component of $K(\mathsf{plac}_n)$. Let $T = T_0, T_1, \ldots, T_{m-1}, T_m = U$ be a path in $K(\mathsf{plac}_n)$ from T to U. Then

$$T = T_0 \sim T_1 \sim \ldots \sim T_{m-1} \sim T_m = U.$$

Thus for $i = 0, \ldots, m-1$, there are words $u_i, v_i \in \mathcal{A}_n^*$ such that $T_i = \mathrm{P}_{\mathsf{plac}}(u_i v_i)$ and $T_{i+1} = \mathrm{P}_{\mathsf{plac}}(v_i u_i)$. For each i, at least one of u_i and v_i does not contain the symbol 1, and by parts 2) and 3) of Lemma 2.2, $\mathrm{cochseq}(T_i)$ and $\mathrm{cochseq}(T_{i+1})$ differ by adding 1 to certain components or subtracting 1 from certain components. Hence corresponding components of $\mathrm{cochseq}(T)$ and $\mathrm{cochseq}(U)$ differ by at most m. Since $\mathrm{cochseq}(T) = (0, 0, \ldots, 0, 0)$ and $\mathrm{cochseq}(U) = (0, 1, \ldots, n-2, n-1)$, it follows that $m \geq n-1$. Hence T and U are a distance at least $n-1$ apart in $K(\mathsf{plac}_n)$.

To establish the upper bound for k_n, we proceed as follows. Let $T, U \in \mathsf{plac}_n$ be such that $T \equiv_{\mathrm{ev}} U$. Let R be the unique tableau of row shape such that $T \equiv_{\mathrm{ev}} R \equiv_{\mathrm{ev}} U$. By [2, Proof of Theorem 17], $T \sim^{n-1} R$ and $U \sim^{n-1} R$. Thus there exist $T', U' \in \mathsf{plac}_n$ such that $T \sim^{n-2} T'$ and $U \sim^{n-2} U'$, and $T' \sim R \sim U'$. Since there is only one word $w \in \mathcal{A}_n^*$ that represents the row R, it follows that there are words t and u that are cyclic shift of w with $T' = \mathrm{P}_{\mathsf{plac}}(t)$ and $U' = \mathrm{P}_{\mathsf{plac}}(u)$. Since t and u are both cyclic shifts of w, they are cyclic shifts of each other and so $T' \sim U'$. Hence $T \sim^{2n-3} U$. □

Experimentation using the computer algebra software Sage [16] suggests the following conjecture on maximum diameters of connected components of $K(\mathsf{plac}_n)$; see Figure 1:

Conjecture 4.4. The maximum diameter of a connected component of $K(\mathsf{plac}_n)$ is $n-1$.

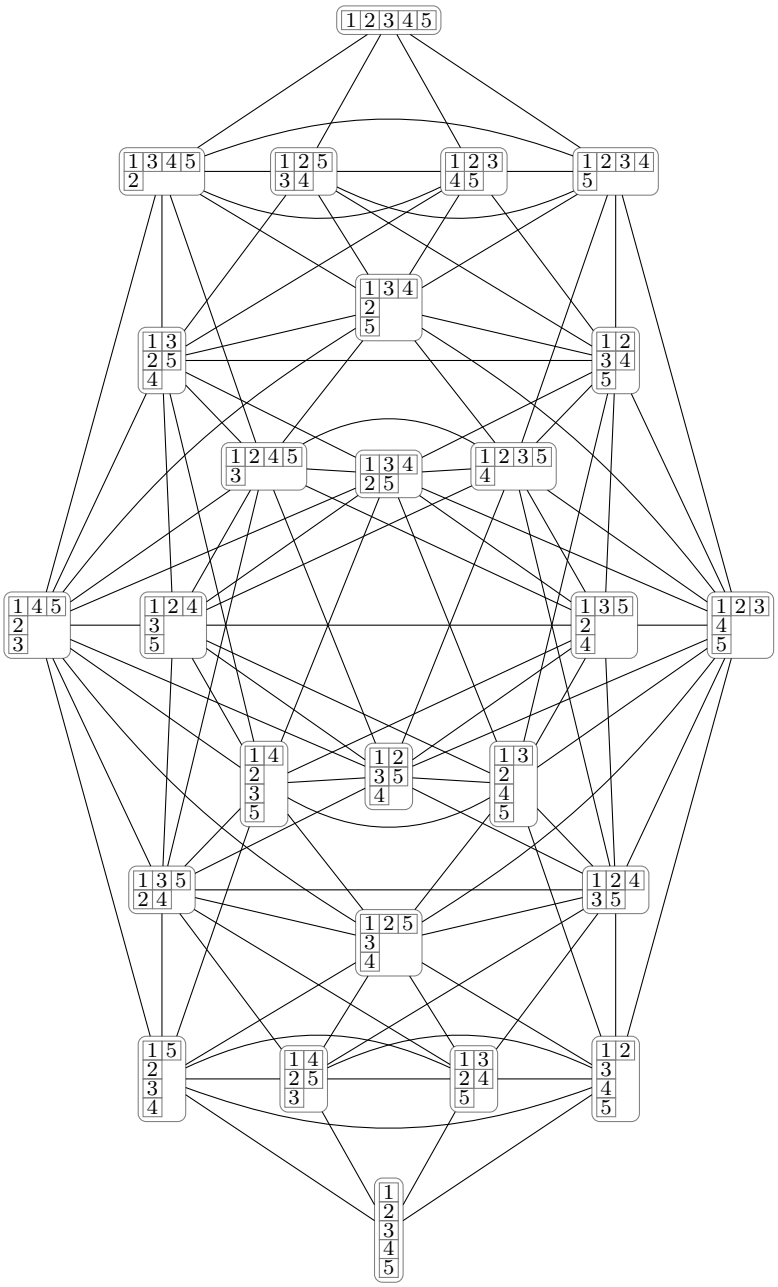


FIGURE 1. The connected component $K(\mathsf{plac}_5, \mathrm{P}_{\mathsf{plac}}(12345))$ of the cyclic shift graph. Note that its diameter is 4. (As discussed following the definition of the cyclic shift graph, the loops at each vertex are not shown.)

5. Hypoplactic monoid

Following the course of the previous section, only the essential facts about the hypoplactic monoid are recalled here; for proofs and further reading, see [9].

A *quasi-ribbon tableau* is a finite array of symbols from $\mathcal{A}$, with rows non-decreasing from left to right and columns strictly increasing from top to bottom, that does not contain any 2×2 subarray (that is, of the form $\boxplus$). An example of a quasi-ribbon tableau is:

(5.1)
$$\begin{array}{|c|c|c|c|c|}\hline 1 & 1 & 2 & & \\ \hline & & 3 & 4 & 4 \\ \cline{3-5} & & & & 5 \\ \cline{4-5} & & & 6 & 6 \\ \cline{4-5}\end{array} .$$

Notice that the same symbol cannot appear in two different rows of a quasi-ribbon tableau.

The insertion algorithm is as follows:

Algorithm 5.1 ([7, § 7.2])**.**

Input: A quasi-ribbon tableau T and a symbol $a \in \mathcal{A}$.

Output: A quasi-ribbon tableau $T \leftarrow a$.

Method: If there is no entry in T that is less than or equal to a, output the quasi-ribbon tableau obtained by creating a new entry a and attaching (by its top-left-most entry) the quasi-ribbon tableau T to the bottom of a.

If there is no entry in T that is greater than a, output the quasi-ribbon tableau obtained by creating a new entry a and attaching (by its bottom-right-most entry) the quasi-ribbon tableau T to the left of a.

Otherwise, let x and z be the adjacent entries of the quasi-ribbon tableau T such that $x \leq a < z$. (Equivalently, let x be the right-most and bottom-most entry of T that is less than or equal to a, and let z be the left-most and top-most entry that is greater than a. Note that x and z could be either horizontally or vertically adjacent.) Take the part of T from the top left down to and including x, put a new entry a to the right of x and attach the remaining part of T (from z onwards to the bottom right) to the bottom of the new entry a, as illustrated here:

$$\begin{array}{|c|c|}\hline & x \\ \hline & z \\ \cline{2-2}\end{array} \leftarrow a = \begin{array}{|c|c|c|}\hline & x & a \\ \hline & z & \\ \cline{2-3}\end{array}$$

[where x and z are vertically adjacent]

$$\begin{array}{|c|c|c|}\hline & & \\ \hline x & z & \\ \cline{1-3}\end{array} \leftarrow a = \begin{array}{|c|c|}\hline & \\ \hline x & a \\ \hline & z \\ \cline{2-2}\end{array}$$

[where x and z are horizontally adjacent]

Output the resulting quasi-ribbon tableau.

Thus one can compute, for any word $u \in \mathcal{A}^*$, a quasi-ribbon tableau $\mathrm{P}_{\mathsf{hypo}}(u)$ by starting with an empty quasi-ribbon tableau and successively inserting the symbols of u, proceeding left-to-right through the word. Define the relation $\equiv_{\mathsf{hypo}}$ by

$$u \equiv_{\mathsf{hypo}} v \iff \mathrm{P}_{\mathsf{hypo}}(u) = \mathrm{P}_{\mathsf{hypo}}(v)$$

for all $u, v \in \mathcal{A}^*$. The relation $\equiv_{\mathsf{hypo}}$ is a congruence, and the *hypoplactic monoid*, denoted hypo, is the factor monoid $\mathcal{A}^*/\equiv_{\mathsf{hypo}}$; the *hypoplactic monoid of rank* n, denoted hypo_n, is the factor monoid $\mathcal{A}_n^*/\equiv_{\mathsf{hypo}}$ (with the

natural restriction of $\equiv_{\mathsf{hypo}}$). Each element $[u]_{\equiv_{\mathsf{hypo}}}$ (where $u \in \mathcal{A}^*$) can be identified with the quasi-ribbon tableau $\mathrm{P}_{\mathsf{hypo}}(u)$. For two quasi-ribbon tableaux T and U, denote the product of T and U in hypo by $T \circ U$.

The monoid hypo is presented by $\langle A \,|\, \mathcal{R}_{\mathsf{hypo}} \rangle$, where

$$\begin{aligned}\mathcal{R}_{\mathsf{hypo}} = \mathcal{R}_{\mathsf{plac}} &\\ \cup \big\{ (cadb, acbd) : a \leq b < c \leq d \big\} &\\ \cup \big\{ (bdac, dbca) : a < b \leq c < d \big\};&\end{aligned}$$

see [9, § 4.1] or [7, § 4.8]. The monoid hypo_n is presented by $\langle \mathcal{A}_n \,|\, \mathcal{R}_{\mathsf{hypo}} \rangle$, where the set of defining relations $\mathcal{R}_{\mathsf{hypo}}$ is naturally restricted to $\mathcal{A}_n^* \times \mathcal{A}_n^*$. Notice that hypo and hypo_n are multihomogeneous.

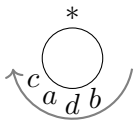
It seems that cyclic shifts of elements of hypo have not been explicitly discussed in the existing literature. However, it is clear from the defining relations that hypo is a quotient of plac under the natural homomorphism. If two elements are $\sim$-related in plac, they are $\sim$-related in hypo. Furthermore, since an element of plac and its image in hypo have the same evaluation, and since connected components of $K(\mathsf{plac})$ coincide with $\equiv_{\mathrm{ev}}$-classes, it follows that connected components of $K(\mathsf{hypo})$ also coincide with $\equiv_{\mathrm{ev}}$-classes. That is, $\sim^* = \equiv_{\mathrm{ev}}$ in hypo.

In contrast to $K(\mathsf{plac}_n)$, it is possible to give an exact value for the maximum diameter of a connected component in $K(\mathsf{hypo}_n)$: the aim is to prove that it is $n-1$. The proof that it cannot be smaller than $n-1$ is similar to the proof of the lower bound in Proposition 4.3. Again, cocharge sequences are the key:

Proposition 5.2. *Let $u, v \in \mathcal{A}^*$ be standard words such that $u \equiv_{\mathsf{hypo}} v$. Then* $\mathrm{cochseq}(u) = \mathrm{cochseq}(v)$.

Proof. It suffices to prove the result when u and v differ by a single application of a defining relation. Assume that u and v differ by an application of a defining relation $(cadb, acbd) \in \mathcal{R}_{\mathsf{hypo}} \setminus \mathcal{R}_{\mathsf{plac}}$ where $a \leq b < c \leq d$. The reasoning for defining relations of the form $(bdac, dbca)$ is similar, and for defining relations in $\mathcal{R}_{\mathsf{plac}}$, one can proceed in the same way as in the proof of Proposition 4.2. So $u = pcadbq$ and $v = pacbdq$, where $p, q \in \mathcal{A}_n^*$ and $a, b, c, d \in \mathcal{A}_n$ with $a \leq b < c \leq d$. Since u and v are standard words, $a < b$ and $c < d$.

Consider how labels are assigned to the symbols a, b, c, and d when calculating $\mathrm{cochseq}(w)$ as described in Subsection 2.4:



Among these four symbols, a will receive a label first, then b, then c, then d. Thus, after a, the labelling process will pass $*$ at least once to visit b and (perhaps after passing $*$ more times) only then visit c. After visiting b, it must visit c first and thus must pass $*$ at least once before visiting d. Thus interchanging a and c and interchanging b and d does not alter the resulting labelling. Hence $\mathrm{cochseq}(u) = \mathrm{cochseq}(v)$. □

For any standard quasi-ribbon tableau T in hypo, define cochseq(T) to be cochseq(u) for any standard word $u \in \mathcal{A}^*$ such that $T = \mathrm{P}_{\mathsf{hypo}}(u)$. By Proposition 5.2, cochseq(T) is well-defined.

Lemma 5.3. *There is a connected component in $K(\mathsf{hypo}_n)$ with diameter at least $n-1$.*

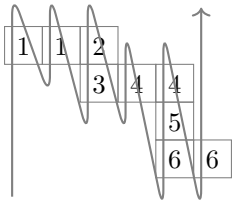
Proof. To prove this result, it suffices to exhibit two elements that lie in the same connected component of $K(\mathsf{hypo}_n)$), but that are a distance strictly at least $n-1$ apart. Let $t = 12\cdots(n-1)n$ and $u = n(n-1)\cdots 21$, and let

$$T = \mathrm{P}_{\mathsf{hypo}}(t) = \begin{array}{|c|c|c|}\hline 1 & 2 & \\ \hline \end{array}\!\!\cdots\!\!\begin{array}{|c|}\hline n \\ \hline \end{array} \quad \text{and } U = \mathrm{P}_{\mathsf{hypo}}(u) = \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline \end{array}\;\vdots\;\begin{array}{|c|}\hline n \\ \hline \end{array}$$

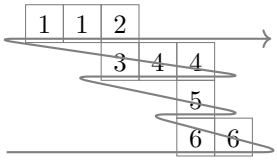
Then $T \equiv_{\mathrm{ev}} U$ and so T and U are in the same connected component of $K(\mathsf{hypo}_n)$. Reasoning similar to the proof of Proposition 4.3 shows that a path from T to U in $K(\mathsf{hypo}_n)$ must have length at least $n-1$. □

Having shown that there is a connected component of $K(\mathsf{hypo}_n)$ with diameter at least $n-1$, the next step is to prove that connected components of $K(\mathsf{hypo}_n)$ have diameter at most $n-1$. To do this, some more definitions are necessary:

- The *column reading* of a quasi-ribbon tableau T, denoted $\mathrm{C}(T)$, is the word in $\mathcal{A}^*$ obtained by reading each column from bottom to top, proceeding left to right through the columns. So the column reading of (5.1) is 1 1 32 4 654 6 (where the spaces are simply for clarity, to show readings of individual columns):



- The *row reading* of a quasi-ribbon tableau T, denoted $\mathrm{R}(T)$, is the word in $\mathcal{A}^*$ obtained by reading the each row from left to right, proceeding through the rows from bottom to top. So the row reading of (5.1) is 66 5 344 112 (where the spaces are simply to show readings of individual rows):



Let $w \in \mathcal{A}^*$ and let $a_1 < a_2 < \ldots < a_k$ be the symbols in $\mathcal{A}_n$ that appear in w. The word w contains an (a_i, a_{i+1})-*inversion* if it contains a symbol a_{i+1} somewhere to the left of a symbol a_i. The following lemma is immediate from the statement of Algorithm 5.1:

Lemma 5.4. *Let $w \in \mathcal{A}^*$ and let $a_1 < a_2 < \ldots < a_k$ be the symbols in $\mathcal{A}_n$ that appear in w. Then w contains an (a_i, a_{i+1})-inversion if and only if a_i and a_{i+1} are on different rows of $\mathrm{P}_{\mathsf{hypo}}(w)$.*

Proposition 5.5. *Let T be a quasi-ribbon tableau. Then $\mathrm{P}_{\mathsf{hypo}}(\mathrm{C}(T)) = \mathrm{P}_{\mathsf{hypo}}(\mathrm{R}(T)) = T$.*

[The fact that $\mathrm{P}_{\mathsf{hypo}}(\mathrm{C}(T)) = T$ follows from [9, Note 4.5], but the following proof treats $\mathrm{P}_{\mathsf{hypo}}(\mathrm{C}(T))$ in parallel.]

Proof. Let $a_1 < a_2 < \ldots < a_k$ be the symbols in $\mathcal{A}_n$ that appear in T. By definition, $\mathrm{C}(T)$ and $\mathrm{R}(T)$ will contain an (a_i, a_{i+1})-inversion if and only if a_i and a_{i+1} are on different rows of T. Hence, by Lemma 5.4, symbols a_i and a_{i+1} are on different rows of $\mathrm{P}_{\mathsf{hypo}}(\mathrm{C}(T))$ and $\mathrm{P}_{\mathsf{hypo}}(\mathrm{R}(T))$ if and only if they are on different rows of T. Furthermore, $\mathrm{P}_{\mathsf{hypo}}(\mathrm{C}(T))$ and $\mathrm{P}_{\mathsf{hypo}}(\mathrm{R}(T))$ both have the same evaluation as T. Since quasi-ribbon tableau is determined by its evaluation and whether adjacent symbols are on different rows, it follows that $\mathrm{P}_{\mathsf{hypo}}(\mathrm{C}(T)) = \mathrm{P}_{\mathsf{hypo}}(\mathrm{R}(T)) = T$. □

Lemma 5.6. *Every connected component of $K(\mathsf{hypo}_n)$ has diameter at most $n - 1$.*

(While reading this proof, the reader may wish to look ahead to Example 5.8, which illustrates the strategy.)

Proof. Let T and U be elements of the same connected component of $K(\mathsf{hypo}_n)$. Then $T \sim_{\mathrm{ev}} U$. Let $a_1 < a_2 < \ldots < a_k$ be the symbols in $\mathcal{A}_n$ that appear in T and U. Since the defining relations in $\mathcal{R}_{\mathsf{hypo}}$ depend only on the relative order of symbols, it is clear that there is an isomorphism from $\langle a_1, \ldots, a_k \rangle$ to hypo_k extending $a_i \mapsto i$. Since $k \leq n$, it suffices to prove that the distance between T and U is at most $n - 1$ when T and U contain every symbol in $\mathcal{A}_n$.

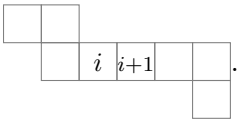
The aim is to construct a path in $K(\mathsf{hypo}_n)$ from T to U of length at most $n - 1$. For simplicity, the aim is to find a sequence $T_0, T_1, \ldots, T_{n-1}$ such that $T_0 = T$ and $T_{n-1} = U$, and $T_i \sim T_{i+1}$ for $i = 0, \ldots, n - 2$. The construction is inductive, starting from $T_0 = T$ and building each T_i so that the part of T_i that contains only symbols from $\{1, \ldots, i + 1\}$ is equal to the corresponding part of U.

Set $T_0 = T$. Notice that since $T \sim_{\mathrm{ev}} U$, both T and U contain the same number of symbols 1, which must all lie at the left of the top rows of the two tableaux. Thus for $i = 0$, it holds that the part of T_i that contains only symbols from $\{1, \ldots, i + 1\}$ is equal to the corresponding part of U.

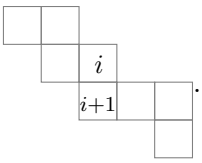
Now suppose that for some i with $1 \leq i < n$, it holds that the part of T_{i-1} that contains only symbols $\{1, \ldots, i\}$ is equal to the corresponding part of U.

Consider the positions of symbols i and $i + 1$ in a quasi-ribbon tableau. One of two cases holds:

(S) All symbols i and $i+1$ are on the same row, with the rightmost symbol i immediately to the left of the leftmost symbol $i+1$:



(D) The symbols i and the symbols $i+1$ are on different rows, with the rightmost symbol i immediately above the leftmost symbol $i+1$:



In each of the quasi-ribbon tableau T_{i-1} and U, the symbols i and $i+1$ may be in the same row or in different rows. There are thus four possible combinations of cases:

(1) Suppose case (S) holds in both T_{i-1} and U. Set $T_i = T_{i-1}$; thus $T_{i-1} \sim T_i$ by the reflexivity of $\sim$. Since T_i and U contain the same number of symbols $i+1$, all of which must lie in the same row, the part of T_i that contains only symbols $\{1, \ldots, i+1\}$ is equal to the corresponding part of U.

(2) Suppose case (D) holds in both T_{i-1} and U. Set $T_i = T_{i-1}$. By the same reasoning as above, $T_{i-1} \sim T_i$ and the part of T_i that contains only symbols $\{1, \ldots, i+1\}$ is equal to the corresponding part of U.

(3) Suppose case (S) holds in T_{i-1} but case (D) holds in U. Let $\mathrm{C}(T_{i-1}) = st$, where s is the column reading of T_{i-1} up to and including the whole of the column containing the rightmost symbol i, and t is the column reading of T_{i-1} from (the whole of) the column containing the leftmost symbol $i+1$. Let $T_i = \mathrm{P}_{\mathsf{hypo}}(ts)$; then $T_{i-1} \sim T_i$.

Notice that s contains all the symbols $\{1, \ldots, i\}$ in the word st. Applying Lemma 5.4 twice, one therefore sees that for any $j < i$, the symbols j and $j+1$ are on different rows of T_{i-1} if and only if s contains a $(j+1, j)$-inversion, which is true if and only if j and $j+1$ are on different rows of T_i. Thus the parts of T_{i-1} and T_i (and thus, by the induction hypothesis, U) that contain only symbols from $\{1, \ldots, i\}$ are equal. Furthermore, ts also contains an $(i+1, i)$ inversion, since t contains at least one symbol $i+1$ and s contains at least one symbol i, and so by Lemma 5.4 the symbols i and $i+1$ are on different rows of T_i. Hence the parts of T_i and U that contain symbols from $\{1, \ldots, i+1\}$ are equal.

(4) Suppose case (D) holds in T_{i-1} but case (S) holds in U. Let $\mathrm{R}(T_{i-1}) = st$, where s is the row reading of T_{i-1} up to and including the whole of the row containing the symbols $i+1$, and t is the row reading of T_{i-1} from (the whole of) the row containing the symbols i. Let $T_i = \mathrm{P}_{\mathsf{hypo}}(ts)$; then $T_{i-1} \sim T_i$.

Notice that t contains all the symbols $\{1, \ldots, i\}$ in the word st. Applying Lemma 5.4 twice, one therefore sees that for any $j < i$, the symbols j and $j+1$ are on different rows of T_{i-1} if and only if

t contains a $(j+1, j)$-inversion, which is true if and only if j and $j+1$ are on different rows of T_i. Thus the parts of T_{i-1} and T_i (and thus, by the induction hypothesis, U) that contain only symbols from $\{1, \dots, i\}$ are equal. Furthermore, ts does not contains an $(i+1, i)$ inversion, since t contains all the symbols i and s contains all the symbols $i+1$, and so by Lemma 5.4 the symbols i and $i+1$ are on the same row of T_i. Hence the parts of T_i and U that contain symbols from $\{1, \dots, i+1\}$ are equal.

By induction, the parts of the quasi-ribbon tableaux T_{n-1} and U that contain symbols from $\{1, \dots, n\}$ are equal. That is, $T_{n-1} = U$. Therefore T and U are a distance at most $n-1$ apart. □

Combining Lemmata 5.6 and 5.3 gives the following result:

Theorem 5.7. (1) *Connected components of* $K(\mathsf{hypo})$ *coincide with* $\equiv_{\mathrm{ev}}$*-classes of* hypo.

(2) *The maximum diameter of a connected component of* $K(\mathsf{hypo}_n)$ *is* $n-1$.

The following example illustrates how the construction in the proof of Lemma 5.6:

Example 5.8. Consider the following elements of hypo_5:

$$T = \begin{array}{|c|c|c|c|c|}\hline 1 & & & & \\ \hline 2 & 3 & & & \\ \hline & 4 & 4 & 5 & \\ \hline \end{array} \qquad \text{and } U = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & & & \\ \hline & 3 & 4 & 4 & \\ \hline & & & 5 & \\ \hline \end{array}.$$

Then T and U have the same evaluation, and so lie in the same connected component of $K(\mathsf{hypo})$ by Theorem 5.7(1), and the distance between them is at most 4 by Theorem 5.7(2). The connected component $K(\mathsf{hypo}_5, T)$ is shown in Figure 2, and the construction in the proof of Lemma 5.6 yields the following path between T and U:

$$T = T_0 = \begin{array}{|c|c|c|c|c|}\hline 1 & & & & \\ \hline 2 & 3 & & & \\ \hline & 4 & 4 & 5 & \\ \hline \end{array}$$

$$\equiv_{\mathsf{hypo}} \begin{array}{|c|c|c|}\hline 2 & 3 & \\ \hline & 4 & 4 & 5 \\ \hline \end{array} \circ \boxed{1} \sim \boxed{1} \circ \begin{array}{|c|c|c|}\hline 2 & 3 & \\ \hline & 4 & 4 & 5 \\ \hline \end{array} \qquad \text{(case 4)}$$

$$\equiv_{\mathsf{hypo}} T_1 = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 3 & & \\ \hline & & 4 & 4 & 5 \\ \hline \end{array}$$

$$\equiv_{\mathsf{hypo}} \boxed{1}\boxed{2} \circ \begin{array}{|c|c|c|}\hline 3 & & \\ \hline 4 & 4 & 5 \\ \hline \end{array} \sim \begin{array}{|c|c|c|}\hline 3 & & \\ \hline 4 & 4 & 5 \\ \hline \end{array} \circ \boxed{1}\boxed{2} \qquad \text{(case 3)}$$

$$\equiv_{\mathsf{hypo}} T_2 = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & & & \\ \hline & 3 & & & \\ \hline & 4 & 4 & 5 & \\ \hline \end{array}$$

$$\equiv_{\mathsf{hypo}} \boxed{4}\boxed{4}\boxed{5} \circ \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & \\ \hline \end{array} \sim \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & \\ \hline \end{array} \circ \boxed{4}\boxed{4}\boxed{5} \qquad \text{(case 4)}$$

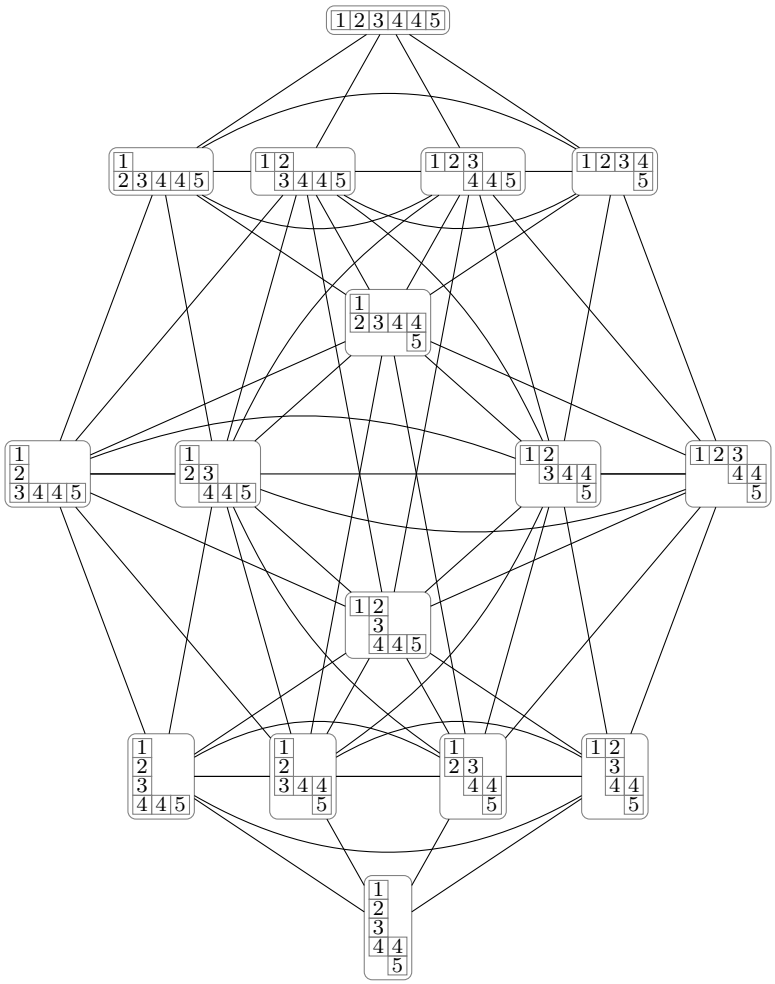


FIGURE 2. The connected component $K(\mathsf{hypo}_5, \mathrm{P}_{\mathsf{hypo}}(123445))$. Note that its diameter is 4.

$$\equiv_{\mathsf{hypo}} T_3 = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & & & \\ \hline & 3 & 4 & 4 & 5 \\ \cline{2-5}\end{array}$$

$$\equiv_{\mathsf{hypo}} \begin{array}{|c|c|c|c|}\hline 1 & 2 & & \\ \hline & 3 & 4 & 4 \\ \cline{2-4}\end{array} \circ \boxed{5} \sim \boxed{5} \circ \begin{array}{|c|c|c|c|}\hline 1 & 2 & & \\ \hline & 3 & 4 & 4 \\ \cline{2-4}\end{array} \qquad \text{(case 3)}$$

$$\equiv_{\mathsf{hypo}} T_4 = \begin{array}{|c|c|c|c|}\hline 1 & 2 & & \\ \hline & 3 & 4 & 4 \\ \cline{2-4} & & & 5 \\ \cline{3-4}\end{array} = U.$$

This gives a path of length 4 between T and U in $K(\mathsf{hypo}_5)$.

Notice, however, that T and U are actually a distance 1 apart in $K(\mathsf{hypo}_5)$:

$$T = \begin{array}{|c|c|c|c|}\hline 1 & & & \\\hline 2 & 3 & & \\\hline & 4 & 4 & 5 \\\hline\end{array} \equiv_{\mathsf{hypo}} \begin{array}{|c|c|c|}\hline 2 & 4 & 4 \\\hline\end{array} \circ \begin{array}{|c|}\hline 1 \\\hline 3 \\\hline 5 \\\hline\end{array} \sim \begin{array}{|c|}\hline 1 \\\hline 3 \\\hline 5 \\\hline\end{array} \circ \begin{array}{|c|c|c|}\hline 2 & 4 & 4 \\\hline\end{array}$$

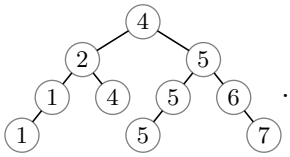
$$\equiv_{\mathsf{hypo}} \begin{array}{|c|c|c|c|}\hline 1 & 2 & & \\\hline & 3 & 4 & 4 \\\hline & & & 5 \\\hline\end{array} = U.$$

6. Sylvester monoid

6.1. **Preliminaries.** Only the necessary facts about the sylvester monoid are recalled here; see [10] for further background.

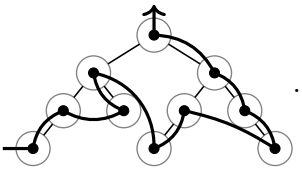
A (*right strict*) *binary search tree* is a labelled rooted binary tree where the label of each node is greater than or equal to the label of every node in its left subtree, and strictly less than every node in its right subtree. An example of a binary search tree is:

(6.1)



The *left-to-right postfix traversal*, or simply the *postfix traversal*, of a rooted binary tree T is the sequence that 'visits' every node in the tree as follows: it recursively perform the postfix traversal of the left subtree of the root of T, then recursively perform the postfix traversal of the right subtree of the root of T, then visits the root of T. Thus the postfix traversal of any binary tree with the same shape as the (6.1) visits nodes as follows:

(6.2)



The insertion algorithm for binary search trees adds the new symbol as a leaf node in the unique place that maintains the property of being a binary search tree:

Algorithm 6.1 (Right strict leaf insertion)**.**

Input: A binary search tree T and a symbol $a \in \mathcal{A}$.

Output: A binary search tree $a \to T$.

Method: If T is empty, create a node and label it a. If T is non-empty, examine the label x of the root node; if $a \leq x$, recursively insert a into the left subtree of the root node; otherwise recursively insert a into the right subtree of the root note. Output the resulting tree.

Thus one can compute, for any word $u \in \mathcal{A}^*$, a binary search tree $\mathrm{P}_{\mathsf{sylv}}(u)$ by starting with an empty binary search tree and successively inserting the symbols of u, proceeding right-to-left through the word. For example $\mathrm{P}_{\mathsf{sylv}}(5451761524)$ is (6.1).

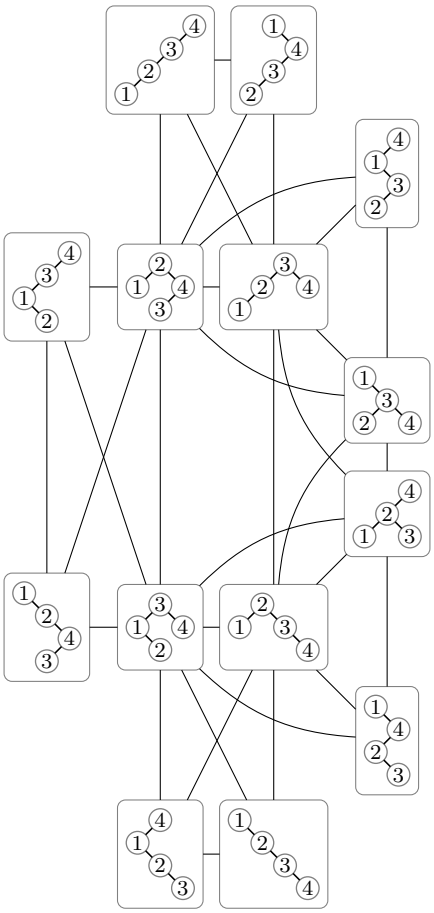


FIGURE 3. The connected component $K(\mathsf{sylv}_4, \mathrm{P}_{\mathsf{sylv}}(1234))$; note that its diameter is 3.

A *reading* of a binary search tree T is a word u such that $\mathrm{P}_{\mathsf{sylv}}(u) = T$. It is easy to see that a reading of T is a word formed from the symbols that appear in the nodes of T, arranged so that every symbol from a parent node appears to the right of those from its children. For example, 1571456254 is a reading of (6.1).

Define the relation $\equiv_{\mathsf{sylv}}$ by

$$u \equiv_{\mathsf{sylv}} v \iff \mathrm{P}_{\mathsf{sylv}}(u) = \mathrm{P}_{\mathsf{sylv}}(v).$$

for all $u, v \in \mathcal{A}^*$. The relation $\equiv_{\mathsf{sylv}}$ is a congruence, and the *sylvester monoid*, denoted sylv, is the factor monoid $\mathcal{A}^*/\equiv_{\mathsf{sylv}}$; the *sylvester monoid of rank* n, denoted sylv_n, is the factor monoid $\mathcal{A}_n^*/\equiv_{\mathsf{sylv}}$ (with the natural restriction of $\equiv_{\mathsf{sylv}}$). Each element $[u]_{\equiv_{\mathsf{sylv}}}$ (where $u \in \mathcal{A}^*$) can be identified with the binary search tree $\mathrm{P}_{\mathsf{sylv}}(u)$. The words in $[u]_{\equiv_{\mathsf{sylv}}}$ are precisely the readings of $\mathrm{P}_{\mathsf{sylv}}(u)$.

The monoid sylv is presented by $\langle \mathcal{A} \,|\, \mathcal{R}_{\mathsf{sylv}} \rangle$, where

$$\mathcal{R}_{\mathsf{sylv}} = \big\{ (cavb, acvb) : a \leq b < c,\ v \in \mathcal{A}^* \big\};$$

the monoid sylv_n is presented by $\langle \mathcal{A}_n \,|\, \mathcal{R}_{\mathsf{sylv}} \rangle$, where the set of defining relations $\mathcal{R}_{\mathsf{sylv}}$ is naturally restricted to $\mathcal{A}_n^* \times \mathcal{A}_n^*$. Notice that sylv and sylv_n are multihomogeneous.

The present authors proved that the relations $\equiv_{\mathrm{ev}}$ and $\sim^*$ coincide in sylv [15, Theorem 3.4]. In the terms of this paper, this proves that connected components of $K(\mathsf{sylv})$ are $\equiv_{\mathrm{ev}}$-classes. The aim in this section is to prove that the maximum diameter of a connected component of $K(\mathsf{sylv}_n)$ is either $n-1$ or n. Subsection 6.2 shows that $K(\mathsf{sylv}_n)$ has a connected component with diameter at least $n-1$. Subsections 6.3 to 6.5 show that connected components of $K(\mathsf{sylv}_n)$ have diameter at most n.

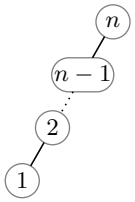
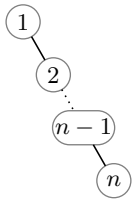
6.2. **Lower bound for diameters.** As in the cases of the plactic and hypoplactic monoids, cocharge sequences are the key to proving that there is a connected component of $K(\mathsf{sylv}_n)$ with diameter at least $n-1$. Reasoning similar to the proofs of Propositions 4.2 and 5.2 establishes the following result:

Proposition 6.2. *Let $u, v \in \mathcal{A}^*$ be standard words such that $u \equiv_{\mathsf{sylv}} v$. Then* $\mathrm{cochseq}(u) = \mathrm{cochseq}(v)$.

For any standard binary tree T in sylv, define $\mathrm{cochseq}(T)$ to be $\mathrm{cochseq}(u)$ for any standard word $u \in \mathcal{A}^*$ such that $T = \mathrm{P}_{\mathsf{sylv}}(u)$. By Proposition 6.2, $\mathrm{cochseq}(T)$ is well-defined.

Lemma 6.3. *There is a connected component in $K(\mathsf{sylv}_n)$ of diameter at least $n-1$.*

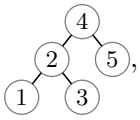
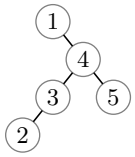
Proof. The strategy is the same as in the plactic and hypoplactic monoids: let $t = 12\cdots(n-1)n$ and $u = n(n-1)\cdots 21$, and let

$T = \mathrm{P}_{\mathsf{sylv}}(t) =$  and $U = \mathrm{P}_{\mathsf{sylv}}(u) =$ 

The same reasoning as in the proof of Proposition 4.3 shows that T and U are not related by $\sim^{\leq n-2}$ in sylv_n. □

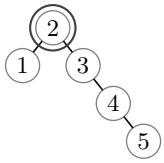
6.3. **Upper bound for diameters I: Overview.** The proof that every connected component of $K(\mathsf{sylv}_n)$ has diameter at most n is long and complicated. To illustrate the strategy, Example 6.4 explicity constructs a path of length 5 between two elements of $K(\mathsf{sylv}_5)$ that have the same evaluation. Note, however, that these elements are standard, and much of the complexity of the general proof is due to having to consider multiple appearances of each symbol.

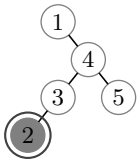
Example 6.4. Let T and U be the following elements of sylv_5:

$T =$  , $U =$  $\in \mathsf{sylv}_5$

Notice that $T \equiv_{\mathrm{ev}} U$. The aim is to build a sequence $T = T_0 \sim T_1 \sim T_2 \sim T_3 \sim T_4 \sim T_5 = U$. The strategy is build the path based upon a left-to-right postfix traversal of U. At any point in such a traversal, the set of vertices already encountered is a union of subtrees of U. By applying an appropriate cyclic shift to obtain T_{i+1} from T_i (for each i), one gradually builds copies of these subtrees within the T_i, arranged down the path of left child nodes descending from the root, with the 'just completed' subtree at the root itself. In the example construction below, the left-hand column shows the i-th tree built so far, and the next column shows the i-th step of the left-to-right postfix traversal. The subtrees of U containing the vertices encountered up to the i-th step are outlined, as are the corresponding vertices in the tree T_i. Note that cyclic shifts never break up the subwords (outlined) that represent the already-built subtrees.

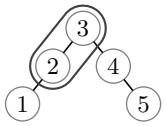
$T = T_0 = \mathrm{P}_{\mathsf{sylv}}(13254) \sim \mathrm{P}_{\mathsf{sylv}}(54132)$

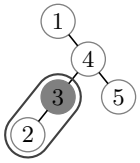
$= T_1 =$ 



One cyclic shift moves the first node visited by the traversal to the root. This will be the base of the induction in the proof.

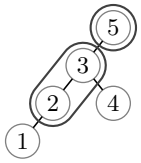
$= \mathrm{P}_{\mathsf{sylv}}(5431\boxed{2}) \sim \mathrm{P}_{\mathsf{sylv}}(1\boxed{2}543)$

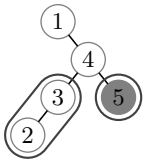
$= T_2 =$ 



The postfix traversal moves from a left child directly to its parent (which has empty right subtree). This will be the induction step, case 3.

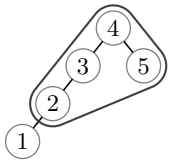
$= \mathrm{P}_{\mathsf{sylv}}(541\boxed{23}) \sim \mathrm{P}_{\mathsf{sylv}}(41\boxed{23}5)$

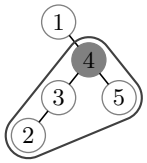
$= T_3 =$ 



The postfix traversal moves from a left child to a node in the right subtree of its parent. This will be the induction step, case 1.

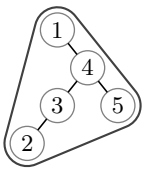
$= \mathrm{P}_{\mathsf{sylv}}(41\boxed{23}\boxed{5}) \sim \mathrm{P}_{\mathsf{sylv}}(1\boxed{23}\boxed{5}4)$

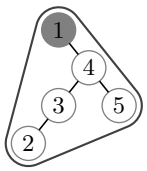
$= T_4 =$ 



The postfix traversal moves from a right child to a parent whose left subtree is non-empty. This well be the induction step, case 2.

$= \mathrm{P}_{\mathsf{sylv}}(1\boxed{2354}) \sim \mathrm{P}_{\mathsf{sylv}}(\boxed{2354}1)$

$= T_5 =$ 

$= U$


The postfix traversal moves from a right child to a parent whose left subtree is empty. This well be the induction step, case 4.

Before beginning the proof, it is necessary to set up some terminology and conventions for diagrams of binary search trees. For brevity, write 'the node x' instead of 'the node labelled by x' and 'the (sub)tree α' instead of 'the (sub)tree with reading α'. However, equalities and inequalties always refer to comparisons of labels: for example, $x = y$ means that the nodes x and y have equal labels, not that they are the same node.

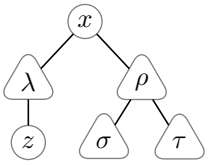
Let x and y be nodes of a binary tree. If x is a descendent of y, then x is *below* y and y is *above* x. (Note that the terms 'above' and 'below' do not refer to levels of the tree: the right child of the root is not above any node in the left subtree.) Let v be the lowest common ancestor of x and y. If x is in the left subtree of v or is v itself, and y is in the right subtree of v or is v itself, and x and y are not both v, then x is *to the left* of y, and y is *to the right* of x. Note that if x is to the left of y, then x is less than or equal to y.

Note that it important to distinguish directional terms like 'above' and 'below' from order terms like 'less than' and 'greater than'. The former refer to the position of nodes within the tree, whereas the latter refer to the order of symbols in the alphabet $\mathcal{A}$.

As used in this section, a *subtree* of a binary search tree will always be a rooted subtree. The *complete subtree* at a node x is the entire tree below and including x. Given a subtree T' of a tree T, the *left-minimal* subtree of T' in T is the complete subtree at the left child of the left-most node in T'; the *right-maximal* subtree of T' in T is the complete subtree at the right child of the right-most node in T'. A node x is *topmost* if it is above all other nodes labelled x. The *path of left child nodes* (respectively, *path of right child nodes*) from a node x is the path obtained by starting at the x and entering left (respectively, right) child nodes until a node with no left (respectively, right) child is encountered.

In diagrams, individual nodes are shown as round, while subtrees as shown as triangles. An edge emerging from the top of a triangle is the edge running from the root of that subtree to its parent. A vertical edge joining a node to its parent indicates that the node may be either a left or right child. An edge emerging from the bottom-left of a triangle is the edge to its left-minimal subtree; an edge emerging from the bottom-right of a triangle is the edge to its right-maximal subtree. For example, consider the following diagram:

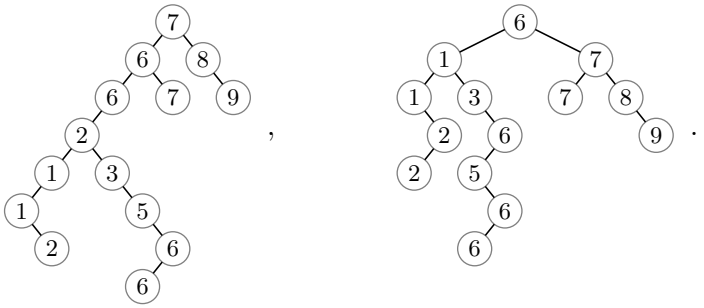
(6.3)



;

this shows a tree with root node x. Its left subtree consists of the subtree λ and a single node z (which may be a left or right child) whose parent is some node in the subtree λ. The right subtree of x consists of subtrees with readings ρ, σ, and τ, with σ being the left-minimal subtree of ρ and τ being the right-maximal subtree of ρ. Note that the tree ρ may be deeper than σ or τ, in the sense that the paths leading to its lowest nodes may longer than the paths to the lowest nodes in σ or τ.

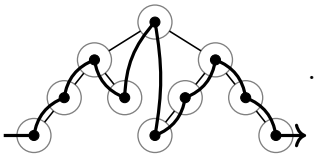
The strategy in the proof of Proposition 6.12 below is to pick $T, U \in \mathsf{sylv}_n$ such that $T \equiv_{\mathrm{ev}} U$ and construct a path of length n from T to U by using a left-to-right postfix traversal of U. Specifically, one considers only the n steps in the left to right postfix traversal that visit the topmost node labelled by each symbol. (There is a unique topmost node labelled by each symbol by Lemma 6.6 below.) Just as in Example 6.4, if the node of U visited at the h-th step of the traversal has labelel x, then the h-th cyclic shift in this path moves a node x in the $h-1$-th tree in the path to become the root of the tree h-th tree. However, the situation is more complicated because there are other nodes with the same label, and these may be distributed very differently in T and U. For example, the following two binary search trees have the same evaluation, but nodes with the same labels are distributed differently:



6.4. **Upper bound for diameters II: Properties of trees.** This section gathers some properties of trees, and in particular properties of how nodes with the same label are arranged in binary search trees. These properties are mostly technical but simple to prove.

The *left-to-right infix traversal* (or simply the *infix traversal*) of a rooted binary tree T is the sequence that 'visits' every node in the tree as follows: it recursively performs the infix traversal of the left subtree of the root of T, then visits the root of T, then recursively performs the infix traversal of the right subtree of the root of T. Thus the infix traversal of any binary tree with the same shape as the right-hand tree in (6.1) visits nodes as follows:

(6.4)



The following result is immediate from the definition of a binary search tree, but it is used frequently:

Proposition 6.5. *For any binary search tree T, if a node x is encountered before a node y in an infix traversal, then $x \leq y$.*

Let U be a binary search tree and let $a \in \mathcal{A}$ be a symbol that appears in U.

Lemma 6.6. *Every node a appears on a single path descending from the root to a leaf; thus there is a unique topmost node a.*

Proof. Suppose the node a appeared on two different paths. Let v be the least common ancestor of two such appearances. Then a is in both the left and right subtrees of v, and so $a \leq v < a$ by the definition of a binary search tree, which is a contradiction. □

Lemma 6.7. *If a node a has a non-empty right subtree, it is the topmost node a.*

Proof. Suppose a particular node a has a non-empty right subtree; let b be a symbol in this subtree. Then $a < b$, since b is in the right subtree of a. If this node a is not the topmost node a, then b is also in the left subtree of the topmost node a (since the distinguished node a must be in the left subtree of the topmost a); hence $b \leq a$, which is a contradiction. Thus this node a must be topmost. □

Lemma 6.8. *Let x be a left child of a parent node z. Then the symbol z is the least symbol in the tree U greater than or equal to every node in the complete subtree at x. Furthermore, if x is not topmost, then $x = z$.*

Proof. In the infix traversal of U, the node z is the first node visited after visiting all the nodes in the complete subtree at x. Since the infix traversal visit nodes in weakly increasing order, z is certainly the least symbol greater than or equal to every node in the complete subtree at x. Suppose further x is not topmost. Then its right subtree is empty and so z is visited immediately after x. Since the topmost node x is visited at some later point, and since nodes are visited in weakly increasing order, $x \leq z \leq x$ and so $x = z$. □

Lemma 6.9. *Suppose z is not a topmost node. Then it is a left child if and only if its parent is another node z.*

Proof. If z is a left child, then its parent is another node z by Lemma 6.8. If the parent of z is another node z, then the child node must be a left child by the definition of a binary search tree. □

Lemma 6.10. *Suppose x is a node and let y be a symbol such that $y \leq x$ and y labels some node in the complete subtree at x. Then the topmost node y is also in the complete subtree at x.*

Proof. The result holds trivially if $y = x$, so suppose $y < x$. Then the node y is in the left subtree of x and the infix traversal visits the node y in the complete subtree at x before it visits x itself. Since the infix traversal visits nodes in weakly increasing order, it must visit the topmost node y before visiting any node x. Hence the topmost node y must also be in the left subtree of x. □

Lemma 6.11. *Suppose x is a topmost node and its right child z is not a topmost node. Then z is the least symbol that is greater than every topmost node in the complete subtree at x.*

Proof. Since the node z is not topmost, its right subtree is empty. Thus it is the maximum symbol in the complete subtree rooted at x. For any node y in the left subtree of x, the topmost node y is also in the left subtree of x by Lemma 6.10. For any node y' in the left subtree of z, the topmost node

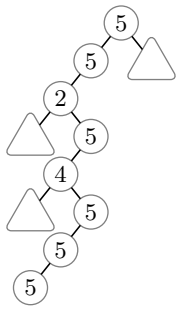
y' is also in the left subtree of z by Lemma 6.10. So z is greater than every topmost node in the complete subtree at x. Since the symbols labelling these topmost nodes are the ones visited by the infix traversal immediately before it visits nodes labelled by z, it follows that z is the least symbol that is greater than every topmost node in the complete subtree at x. □

These results give information about how repeated symbols can appear in a binary search tree. Consider a symbol z that appears more than once in U. If one chooses a node z, then one of the following holds:

- the node z is topmost;
- the node z is not a topmost node, and is the left child of another node z (by Lemma 6.9);
- the node z is not a topmost node, and the right child of a topmost node x, and z is the least symbol greater than every topmost node in the complete subtree at x (by Lemma 6.11).

For example, consider the following binary search tree and the repeated symbol 5:

(6.5)



The *primary* nodes a are those nodes labelled by a that are consecutive with the topmost node a, including the topmost node itself; in (6.5) there are two primary nodes 5. Any node a that has no children, and any node a consecutive with it, provided they are not primary, are the *tertiary* nodes a; in (6.5) there are three tertiary nodes 5. All other nodes a are *secondary*. Note that in each group of consecutive secondary nodes a, the uppermost node is a right child of some non-a node, and the lowermost node has as its left child another non-a node. (Secondary and tertiary nodes always have empty right subtrees, since they are never topmost.) In (6.5), there is one secondary node 5.

In constructing a path of length n from T to U, the h-th cyclic shift, corresponding to visiting a topmost node y of U, will move a node y of T_{h-1} to form the root of T_h, and must simultaneously deal with any secondary nodes z that are attached at the right child of y in U. Tertiary nodes z either fall into place naturally or are dealt with during the cyclic shift that moves z to the root. The difficulty is in proving that there always exists a cyclic shift that performs these tasks. This is the reason for the slightly complicated conditions that form the inductive statement in the proof of Proposition 6.12 below.

6.5. **Upper bound for diameters III: Result.** This subsection is dedicated entirely to Proposition 6.12 and its proof. Extensive use will be made of the concepts and results in Subsections 6.3 and 6.4.

Proposition 6.12. *The diameter of any connected component of $K(\mathsf{sylv}_n)$ is at most n.*

Proof. Let T and U be in the same connected component of $K(\mathsf{sylv}_n)$. Then $T \equiv_{\mathrm{ev}} U$ and so T and U contain the same number of nodes labelled by each symbol. Without loss of generality, assume that every symbol in $\mathcal{A}_n$ appears in T and U.

Preliminaries. Consider the left-to-right postfix traversal of U. Modify this traversal so that it only visits topmost nodes; for the purposes of this proof, call this the *topmost traversal*. Since U contains every symbol in $\mathcal{A}_n$, there are exactly n steps in this traversal. Let u_i be the i-th node visited in this modified traversal.

For $h = 1, \ldots, n$, define $U_h = \{u_1, \ldots, u_h\}$ and let $U_h^{\uparrow}$ be the set of nodes in U_h that do not lie below any other node in U_h. Since a later step in a left-to-right postfix traversal is never below an earlier step, $u_h \in U_h^{\uparrow}$ for all h. (The set $U_h^{\uparrow}$ will turn out to be the roots of the complete subtrees that have been 'built' in T_h.)

Let B_h be the complete subtree of U at u_h; see Figure 4 for an example of this and later definitions.

Define m_h to be the minimum symbol in B_h; thus m_h is the minimum symbol below and including the node u_h in U. Note that the topmost node m_h is in the subtree B_h by Lemma 6.10, and must (by minimality) be on the path of left child nodes from u_h.

Define q_h to be the minimum symbol that is greater than every topmost node in B_h, with q_h undefined if there is no such symbol. Note that q_h can be found by following the path from u_h to the root: q_h will be symbol labelling the first node entered from a left child, because this node is the node visited by the infix traversal of U immediately after the nodes in the subtree B_h have been visited. In particular, q_h is defined precisely when u_h does not lie on the path of right child nodes from the root of U.

Define p_h to be the maximum symbol that is less than every symbol in B_h, with p_h undefined if there is no such symbol. Note that p_h can be found by following the path from u_h to the root: p_h will be the symbol labelling the first node entered from a right child, because this node is the node visited by the infix traversal of U immediately before visiting nodes in the subtree B_h. (This process always locates the *topmost* node p_h, since it has a non-empty right subtree.) In particular, p_h is defined precisely when u_h does not lie on the path of left child nodes from the root of U.

Note that $p_h < x \leq q_h$ for all symbols x in B_h, ignoring the inequalities involving p_h or q_h when these are undefined. Note that the symbol q_h may appear in B_h, labelling a secondary or tertiary node. In particular, $p_h < m_h \leq u_h < q_h$, since u_h is a topmost symbol.

Define C_h to the the tree obtained from B_h by deleting all nodes m_h except the topmost. (Note that this leaves no 'orphaned' subtrees, since only the topmost m_h can have a right subtree, and the bottommost m_h has empty left subtree by the minimality of m_h.)

Define D_h to be the tree obtained from C_h by deleting any tertiary nodes q_h from C_h.

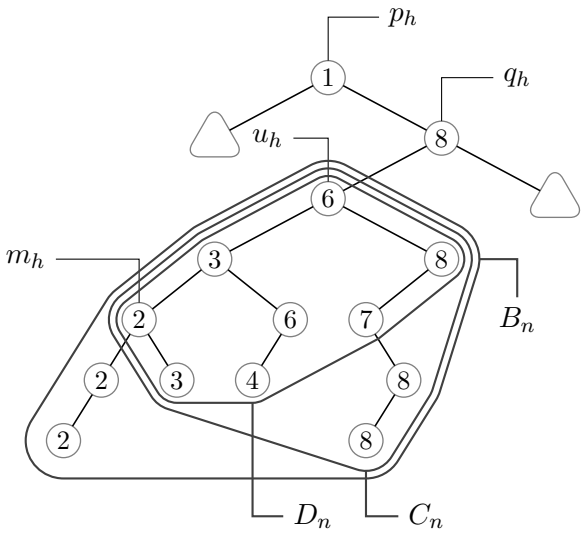


FIGURE 4. Example tree U illustrating definitions of notation. Here, u_h is the topmost node 5, and thus $p_h = 1$ and $q_h = 7$. The subtree B_h consists of the complete subtree at u_h, and the minimum symbol in this subtree is $m_h = 2$. The subtree C_h is obtained from B_h by deleting all nodes 2 except the topmost, and D_h is obtained from C_h by deleting the tertiary nodes 8. Since there is a [secondary] node 8 in D_h, the tree E_h is obtained from D_h by inserting three symbols 8 using Algorithm 6.1, since there are three nodes 8 in the tree U outside D_h.

If D_h contains no node q_h, define $E_h = D_h$. If D_h contains a node q_h, define E_h to be the tree obtained from D_h by inserting s symbols q_h into D_h using Algorithm 6.1, where s is the difference between the number of nodes q_h in U and the the number of nodes q_h in D_h.

Notice that the set of symbols in E_h is the same as the set of symbols in D_h (in either case), and is contained in the set of symbols in C_h (this containment will be strict if in C_h there are tertiary nodes q_h but no secondary nodes q_h), which in turn is equal to the set of symbols in B_h.

Suppose B_h contains a node x, but that B_h does not contain the topmost node x. Then by Lemma 6.6, B_h is below the topmost node x and must be in its left subtree since it contains a node x. In following the path from u_h to the root, q_h labels the first node entered from a left child. Hence $q_h \leq x$. However, x appears in B_h, so $x \leq q_h$. Hence $x = q_h$. Thus q_h is the only symbol that can label a node in B_h but whose corresponding topmost node lies outside B_h. By their definitions, the same applies to C_h, D_h, and E_h.

Statement of induction. The aim is to construct inductively a sequence $T = T_0, T_1, \ldots, T_n = U$ with $T_i \sim T_{i+1}$ for $i \in \{1, \ldots, n-1\}$. Let $h = 1, \ldots, n$ and suppose $U_h^{\uparrow} = \{u_{i_1}, \ldots, u_{i_k}\}$ (where $i_1 < \ldots < i_k = h$). Then the tree T_h will satisfy the following four conditions P1–P4:

P1 The subtree E_{i_k} appears at the root of T_h.

P2 The subtrees $E_{i_k}, \ldots, E_{i_1}$ appear, in that order, on the path of left child nodes from the root of T_h. (These subtrees may be separated by other nodes on the path of left child nodes.)

P3 For $j = 1, \ldots, k$, every node below E_{i_j} is in its left-minimal or right-maximal subtrees in T_h.

P4 For $j = 1, \ldots, k$, no node m_{i_j} is below a node p_{i_j} in T_h.

(Note that conditions P1–P4 do not apply to $T_0 = T$, which is an arbitrary element of sylv_n.)

Base of induction. The base of the induction is to apply a cyclic shift to T_0 and obtain a tree T_1 that satisfies conditions P1–P4.

Note first that $U_1^{\uparrow} = U_1 = \{u_1\}$. By the definition of the topmost traversal, there can be no topmost nodes below u_1 in U. Since $m_1 \leq u_1$, the topmost node m_1 is in B_1 by Lemma 6.10, and thus $m_1 = u_1$. Thus B_1 consists only of symbols u_1 and possibly tertiary nodes q_1. Thus C_1 consists only of the topmost node $m_1 = u_1$ and possibly tertiary nodes q_1, and so D_1 consists only of the single node u_1. Thus $E_1 = D_1$ since D_1 does not contain a symbol q_1.

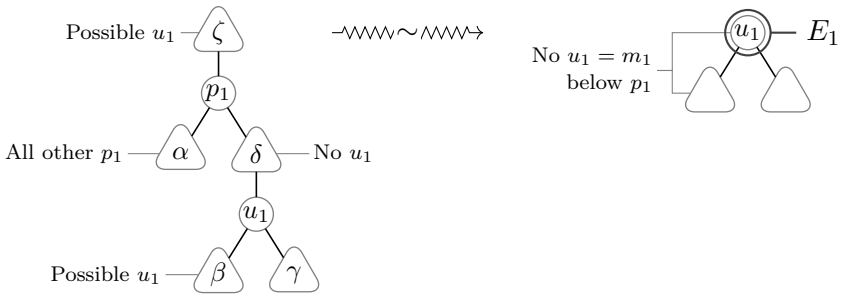
There are two cases to consider:

Case 1. Suppose that there is some node with label u_1 below some node with label p_1 in T_0. (Note that this case can only hold when p_1 is defined, or, equivalently, if u_1 is not on the path of left child nodes from the root of U.)

In T_0, distinguish the uppermost node with label u_1 that lies below some node with label p_1. (Note that although u_1 and p_1 are defined in terms of the tree U, here they are used to pick out nodes with the same labels in the tree T_0.) Since $p_1 < u_1$, this node u_1 must be in the right subtree of the node p_1. Thus this node p_1 must be a topmost node since it has non-empty right subtree. As shown in Figure 5, let ζ be a reading of the part of T_0 outside the complete subtree at the topmost node p_1, let α be a reading of the left subtree of the topmost node p_1, let δ be a reading of the right subtree of the topmost node p_1 outside the complete subtree at the distinguished node u_1, and let β and γ be readings of the left and right subtrees of this node u_1. All nodes p_1 other than the distinguished topmost node p_1 must be in α. There are no nodes u_1 in δ, by the choice of the distinguished node u_1. It is possible that there may be nodes u_1 in β or in ζ. (If there is a node u_1 in ζ, then the distinguished node u_1 is not topmost and so γ is empty.)

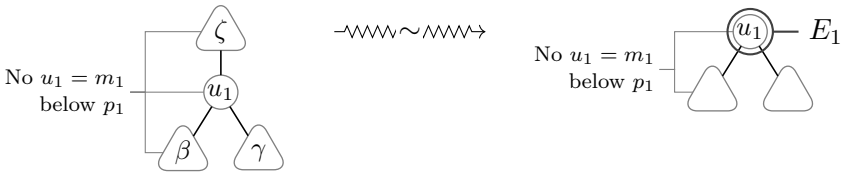
Thus $T_0 = \mathrm{P}_{\mathsf{sylv}}(\beta\gamma u_1 \delta\alpha p_1 \zeta)$. Let $T_1 = \mathrm{P}_{\mathsf{sylv}}(\delta\alpha p_1 \zeta\beta\gamma u_1)$; note that $T_0 \sim T_1$.

In computing T_1, the symbol u_1 is inserted first and becomes the root node. Since other symbols u_1 can only appear in β and ζ, and other symbols p_1 can only appear in α, all symbols $m_1 = u_1$ are inserted before symbols p_1. Thus there is no node m_1 below a node p_1; thus T_1 satisfies P4. Since E_1 consists only of a node u_1, the tree E_1 appears at the root of T_1 and so T_1 satisfies P1 and P2. Furthermore, every node below E_1 is either in the left-minimal subtree of E_1 (that is, the left subtree of the root node u_1) or the right-maximal subtree of E_1 (that is, right subtree of u_1), and so T_1 satisfies P3.



$T_0 = \mathrm{P}_{\mathsf{sylv}}(\beta\gamma u_1 \delta\alpha p_1 \zeta)$ $\qquad$ $T_1 = \mathrm{P}_{\mathsf{sylv}}(\delta\alpha p_1 \zeta\beta\gamma u_1)$

FIGURE 5. Base of induction, case 1: some node u_1 lies below some node p_1 in T_0.



$T_0 = \mathrm{P}_{\mathsf{sylv}}(\beta\gamma u_1 \zeta)$ $\qquad$ $T_1 = \mathrm{P}_{\mathsf{sylv}}(\zeta\beta\gamma u_1)$

FIGURE 6. Base of induction, case 2: no node u_1 lies below a node p_1 in T_0.

Case 2. Suppose that no node labelled u_1 lies below a node with label p_1 in T_0. (This case always holds when p_1 is undefined.)

Distinguish the topmost node u_1 in T_0. As shown in Figure 6, let ζ be a reading of the part of T_0 outside the complete subtree at the topmost node u_1. Since no node u_1 lies below a node p_1, there exists a reading β of the left subtree of the topmost node u_1 in which all symbols p_1 appear before all symbols u_1. Let γ be a reading of the right subtree of the topmost node u_1.

Thus $T_0 = \mathrm{P}_{\mathsf{sylv}}(\beta\gamma u_1 \zeta)$. Let $T_1 = \mathrm{P}_{\mathsf{sylv}}(\zeta\beta\gamma u_1)$; note that $T_0 \sim T_1$.

In computing T_1, the symbol u_1 is inserted first and becomes the root node. Since other symbols u_1 must appear after symbols p_1 in β, all symbols $m_1 = u_1$ are inserted before symbols p_1. Thus there is no node m_1 below a node p_1; thus T_1 satisfies P4. Since E_1 consists only of a node u_1, the tree T_1 satisfies P1–P3 by the same reasoning as in Case 1.

This completes the base of the induction: the tree T_1 satisfies P1–P4 and $T_0 \sim T_1$.

Induction step. Let $h \in \{1, \ldots, n-1\}$ and suppose that $U_h^{\uparrow} = \{u_{i_1}, \ldots, u_{i_k}\}$ (where $i_1 < \ldots < i_k$). Recall that $h = i_k$; for brevity, let $g = i_{k-1}$. Suppose that the tree T_h satisfies conditions P1–P4. The aim is to apply a cyclic shift to T_h and obtain a tree T_{h+1} that satisfies conditions P1–P4.

There are four cases, depending on the relatives positions of u_h and u_{h+1} in U:

(1) u_h is in the left subtree of v and u_{h+1} is in the right subtree of v, where v is the lowest common ancestor of u_h and u_{h+1} in U;
(2) u_h is in the left subtree of u_{h+1};
(3) u_h is in the right subtree of u_{h+1}, and there is no node u_i in the left subtree of u_{h+1};
(4) u_h is in the right subtree of u_{h+1}, and there is some node u_i in the left subtree of u_{h+1}.

Case 1. Suppose that, in U, the node u_h is in the left subtree of v and u_{h+1} is in the right subtree of v, where v is the lowest common ancestor of u_h and u_{h+1} in U.

In this case, $U_{h+1}^{\uparrow} = U_h^{\uparrow} \cup \{u_{h+1}\}$. By the definition of the topmost traversal, there are no topmost nodes below u_{h+1} in U. Since $m_{h+1} \leq u_{h+1}$, the topmost node m_{h+1} is in B_{h+1} by Lemma 6.10, and thus $m_{h+1} = u_{h+1}$. Thus B_{h+1} consists only of symbols u_{h+1} and possibly tertiary nodes q_{h+1}. Thus C_{h+1} consists only of the topmost node $m_{h+1} = u_{h+1}$ and possibly tertiary nodes q_{h+1}, and so D_{h+1} consists only of the single node u_{h+1}. Thus $E_{h+1} = D_{h+1}$ since D_{h+1} does not contain a symbol q_{h+1}.

There are two sub-cases:

Sub-case 1(a). Suppose that there is some node labelled u_{h+1} below some node with label p_{h+1} in T_h and that $q_h \neq p_{h+1}$.

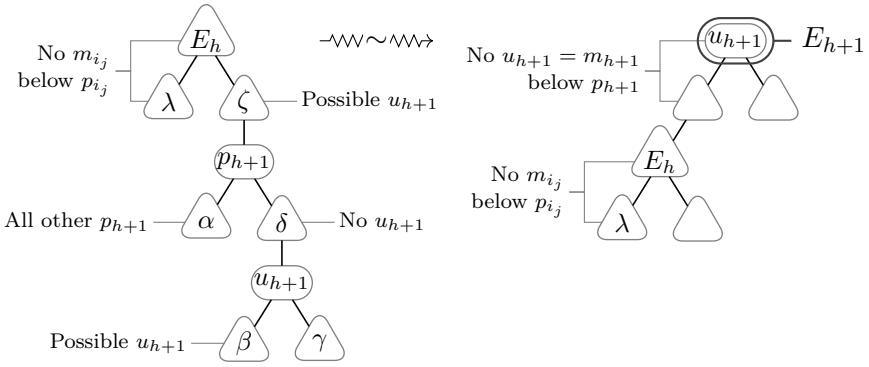
Since u_{h+1} is in the right subtree of v, the symbol p_{h+1} is greater than or equal to v. The symbol v is greater than or equal to every symbol in B_h. It is impossible that p_{h+1} is equal to some symbol in B_h, since this would require $p_{h+1} = v = q_h$, which is excluded from this sub-case. Hence p_{h+1} is strictly greater than every symbol in B_h and thus in E_h.

The tree E_h appears at the root of T_h by P1. Since p_{h+1} is strictly greater than every symbol in E_h, every symbol p_{h+1} must be in the right-maximal subtree of E_h in T_h by P3. Distinguish the uppermost node u_{h+1} that lies below some node p_{h+1}; since $p_{h+1} < u_{h+1}$, this node u_{h+1} must be in the right subtree of the node p_{h+1}. Thus this node p_{h+1} must be a topmost node since it has non-empty right subtree.

As shown in Figure 7, let λ be a reading of the left-minimal subtree of E_h; note that λ contains all the subtrees E_{i_j} by P2. Let ζ be a reading of the right-maximal subtree of E_h outside the complete subtree at the topmost node p_{h+1}, let α be a reading of the left subtree of the topmost node p_{h+1}, let δ be a reading of the right subtree of the topmost node p_{h+1} outside the complete subtree at the distinguished node u_{h+1}, and let β and γ be readings of the left and right subtrees of this node u_{h+1}. All other nodes p_{h+1} must be in α. There are no nodes u_{h+1} in δ, by the choice of the distinguished node u_{h+1}. It is possible that there may be nodes u_{h+1} in β or in ζ. (If there is a node u_{h+1} in ζ, then the distinguished node u_{h+1} is not topmost and so γ is empty.)

Thus

$$T_h = \mathrm{P}_{\mathsf{sylv}}(\beta\gamma u_{h+1}\delta\alpha p_{h+1}\zeta\lambda E_h).$$



$T_h = \mathrm{P}_{\mathsf{sylv}}(\beta\gamma u_{h+1}\delta\alpha p_{h+1}\zeta\lambda E_h)$ $\qquad$ $T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(\delta\alpha p_{h+1}\zeta\lambda E_h\beta\gamma u_{h+1})$

FIGURE 7. Induction step, sub-case 1(a): $E_h = D_h$ and some node u_{h+1} lies below some node p_{h+1} in T_h.

Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(\delta\alpha p_{h+1}\zeta\lambda E_h\beta\gamma u_{h+1});$$

notice that $T_h \sim T_{h+1}$.

In computing T_{h+1}, the symbol u_{h+1} is inserted first and becomes the root node. Since other symbols u_{h+1} can only appear in β and ζ, all symbols $m_{h+1} = u_{h+1}$ are inserted before symbols p_{h+1}. Thus there is no node m_{h+1} below a node p_{h+1}. Since E_{h+1} consists only of a node u_{h+1}, the tree T_{h+1} satisfies P1. Since every symbol in the trees λ and E_h are strictly less than every other symbol, these trees reinserted on the path of left child nodes in the same way; hence T_{h+1} satisfies P2 since T_h does, and satisfies P4 since T_h does and since there is no node m_{h+1} below a node p_{h+1}. Finally, T_{h+1} satisfies P3 because all the E_{i_j} in λ satisfy the condition in P3 (since T_h satisfies P3), and trivially E_{h+1} satisfies the condition in P3.

Sub-case 1(b). Suppose that that no node u_{h+1} lies below a node p_{h+1} in T_h, or that $p_{h+1} = q_h$.

The tree E_h appears at the root of T_h by P1. The symbol u_{h+1} is strictly greater than every symbol in E_h and so every node u_{h+1} must be in the right-maximal subtree of E_h in T_h by P3. Distinguish the topmost node u_{h+1} in T_h. As shown in Figure 8, let λ be a reading of the left-minimal subtree of E_h; note that λ contains all the subtrees E_{i_j} by P2. Let ζ be a reading of the right-maximal subtree of E_h outside the complete subtree at the topmost node u_{h+1}. Note that no symbol u_{h+1} appears in ζ. If no node u_{h+1} lies below a node p_{h+1} in T_h, choose a reading β of the left subtree of the topmost node u_{h+1} in which all symbols p_{h+1} appear before all symbols u_{h+1}. On the other hand, if $p_{h+1} = z_h$, so that every node with this label in T_h is in the subtree E_h, then fix any reading β of the left subtree of the topmost node u_{h+1}; then β vacuously has the same property (since it contains no symbols p_{h+1} at all). Let γ be a reading of the right subtree of the topmost node u_{h+1}.

Thus

$$T_h = \mathrm{P}_{\mathsf{sylv}}(\beta\gamma u_{h+1}\zeta\lambda E_h).$$

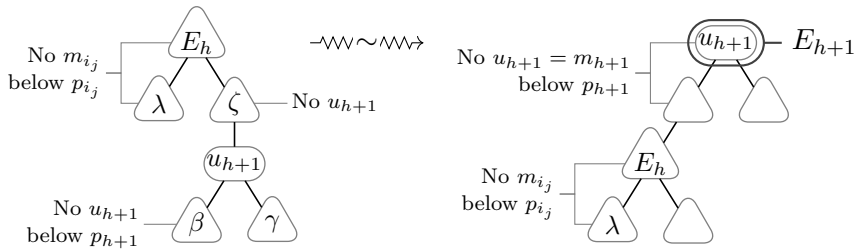


FIGURE 8. Induction step, sub-case (b): $E_h = D_h$ and no node u_{h+1} lies below a node p_{h+1} in T_h.

Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(\zeta\lambda E_h\beta\gamma u_{h+1});$$

note that $T_h \sim T_{h+1}$.

In computing T_{h+1}, the symbol u_{h+1} is inserted first and becomes the root node. Since other symbols u_{h+1} can only appear in β, and any symbols u_{h+1} in β must appear after symbols p_{h+1}, all symbols $m_{h+1} = u_{h+1}$ are inserted before symbols p_{h+1}. Thus there is no node m_{h+1} below a node p_{h+1}. Since E_{h+1} consists only of a node u_{h+1}, the tree T_{h+1} satisfies P1. Since every symbol in the trees λ and E_h are strictly less than every other symbol, these trees are re-inserted on the path of left child nodes in the same way; hence T_{h+1} satisfies P2 since T_h does, and satisfies P4 since T_h does and since there is no node m_{h+1} below a node p_{h+1}. Finally, T_{h+1} satisfies P3 because all the E_{i_j} in λ satisfy the condition in P3 because T_h satisfies P3, and trivially E_{h+1} satisfies the condition in P3.

Case 2. Suppose that, in U, the node u_h is in the left subtree of u_{h+1}. By the definition of the topmost traversal, the right subtree of u_{h+1} contains no node u_i.

In this case, $U_{h+1}^{\uparrow} = \left(U_h^{\uparrow} \setminus \{u_h\}\right) \cup \{u_{h+1}\}$. It is immediate from the definitions that $p_{h+1} = p_h$ and $m_{h+1} = m_h$. Finally, q_h is defined and $q_h = u_{h+1}$, since u_{h+1} will be the next topmost node that the infix traversal of U visits after visiting all nodes in B_h, and so u_{h+1} must be the topmost node q_h since the infix traversal visits nodes in weakly increasing order.

Sub-case 2(a). Suppose that $E_h \neq D_h$. Then D_h contains nodes with label $q_h = u_{h+1}$. By the definition of q_h, the symbol u_{h+1} is greater than or equal to every symbol in D_h. Thus u_{h+1} is the rightmost symbol in D_h and this is the node where the right-maximal subtree of D_h (and E_h) is attached in T_h. The tree E_h consists of D_h with s nodes u_{h+1} inserted, where s is the number of nodes u_{h+1} that appear in U outside of the complete subtree at u_h.

As shown in Figure 9, let λ be a reading of the left-minimal subtree of D_h; note that the subtree λ contains all the E_{i_j} except E_h by P2. Let β be a reading of the right-maximal subtree of D_h.

Thus

$$T_h = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^s\beta\lambda D_h).$$

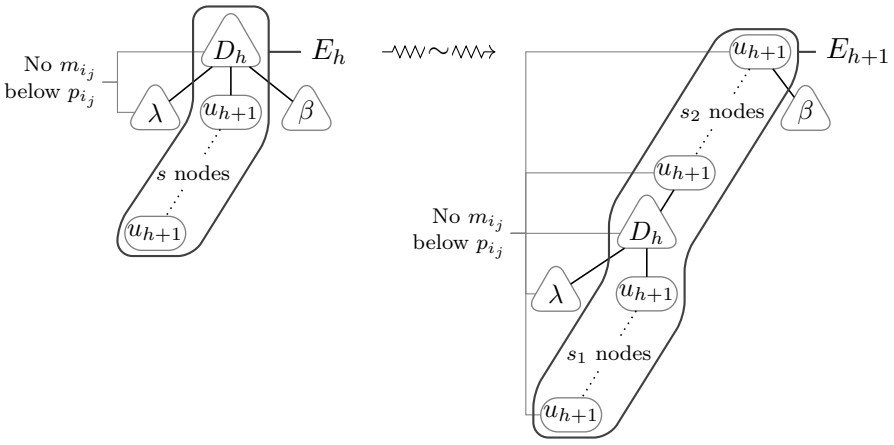


FIGURE 9. Induction step, sub-case 2(a): $E_h \neq D_h$.

Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{s_1} \beta \lambda D_h u_{h+1}^{s_2}),$$

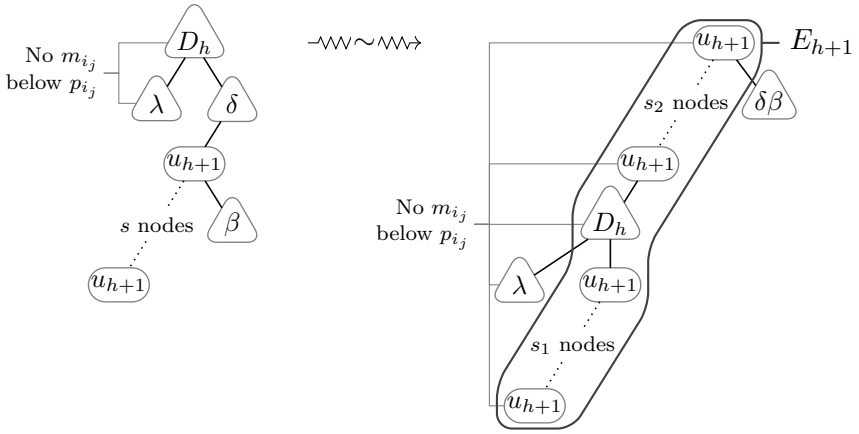
where s_2 is the number of primary nodes u_{h+1} in U, and where $s_1 = s - s_2$. Notice that $T_{h+1} \sim T_h$.

In computing T_{h+1}, the rightmost symbol u_{h+1} is inserted first and becomes the root node. Then the next $s_2 - 1$ nodes u_{h+1} are attached along the path of left child nodes. Since every symbol in D_h or λ is less than or equal to u_{h+1}, the subtrees D_h and λ are re-inserted at the left child of the bottommost node u_{h+1}. Since every symbol in β is strictly greater than u_{h+1}, the subtree β is re-inserted as the right child of the root node u_{h+1}. Finally, the remaining s_1 symbols u_{h+1} are re-inserted into D_h.

All secondary nodes u_{h+1} are inside D_h, so the s nodes u_{h+1} outside D_h are either primary or tertiary. Since there are s_2 primary nodes u_{h+1} in U, and since the evaluation of U is the same as the evaluation of T_h, there are s_1 tertiary nodes u_{h+1} in U. Hence D_h and the other nodes u_{h+1} in T_{h+1} together make up E_{h+1}; thus T_{h+1} satisfies P1. All the other E_{i_j} are contained in λ, so T_{h+1} satisfies P2. Since T_h satisfies P3, and since λ and β are the left-minimal and right-maximal subtrees of E_{h+1}, it follows that T_{h+1} satisfies P3. Finally, the relative positions of the m_{i_j} and p_{i_j} have not been altered, so T_{h+1} satisfies P4.

Sub-case 2(b). Suppose that $E_h = D_h$. Thus D_h does not contain nodes labelled $q_h = u_{h+1}$. Since u_{h+1} is greater than every node in D_h, it follows that all nodes u_{h+1} are in the right-maximal subtree of D_h (and E_h) in T_h. Furthermore, since u_{h+1} is the smallest symbol greater than every symbol in D_h, only another node u_{h+1} can be the left child of a node u_{h+1} in T_h.

As shown in Figure 10, let λ be a reading of the left-minimal subtree of D_h; note that the subtree λ contains all the E_{i_j} except E_h by P2. Let δ be a reading of the right-maximal subtree of D_h outside of the complete subtree at the topmost node u_{h+1}. (Note that δ may be empty.) Let β be a reading of the right subtree of the topmost node u_{h+1}; note that the left subtree



$T_h = \mathrm{P}_{\mathsf{sylv}}(\beta u_{h+1}^t \delta\lambda D_h)$ $\qquad$ $T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{s_1}\delta\lambda D_h \beta u_{h+1}^{s_2})$

FIGURE 10. Induction step, sub-case 2(b): $E_h = D_h$.

of this node can only contain other nodes u_{h+1}. Suppose there are s nodes u_{h+1} in U (and so in T_h).

Thus $T_h = \mathrm{P}_{\mathsf{sylv}}(\beta u_{h+1}^s \delta\lambda D_h)$. Let $T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{s_1}\delta\lambda D_h \beta u_{h+1}^{s_2})$, where s_2 is the number of primary nodes u_{h+1} in U, and where $s_1 = s - s_2$. In computing T_{h+1}, the rightmost symbol u_{h+1} is inserted first and becomes the root node. Then the next $s_2 - 1$ nodes u_{h+1} are attached along the path of left child nodes. Since every symbol in β is strictly greater than u_{h+1}, the subtree β is re-inserted as the right child of the root node u_{h+1}. Since every symbol in D_h or λ is less than or equal to u_{h+1}, the subtrees D_h and λ are re-inserted as the left child of the bottommost node u_{h+1}. Since every symbol in δ is strictly greater than u_{h+1}, the symbols in δ are also inserted into the right subtree of the root node u_{h+1}. Finally, the remaining s_1 symbols u_{h+1} are re-inserted into D_h.

By the definition of D_h and the fact that there are s_2 primary nodes u_{h+1} in U, and the fact that the evaluation of U is the same as the evaluation of T_h, there are s_1 tertiary nodes u_{h+1} in U. Hence D_h and the other nodes u_{h+1} in T_{h+1} together make up E_{h+1}; thus T_{h+1} satisfies P1. All the other E_{i_j} are contained in λ, so E_{h+1} satisfies P2. Since T_h satisfies P3, and since λ and $\delta\beta$ are the left-minimal and right-maximal subtrees of E_{h+1}, it follows that T_{h+1} satisfies P3. Finally, the relative positions of m_{i_j} and p_{i_j} have not been altered, so T_{h+1} satisfies P4.

Case 3. Suppose that, in U, the node u_h is in the right subtree of u_{h+1}, and there is no node u_i in the left subtree of u_{h+1}. Note that q_h is defined if and only if q_{h+1} is defined, in which case $q_h = q_{h+1}$.

In this case, $U_{h+1}^{\uparrow} = \left(U_h^{\uparrow} \setminus \{u_h\}\right) \cup \{u_{h+1}\}$, and $p_h = u_{h+1}$.

By definition, B_{h+1} is the complete subtree of U at u_{h+1}. So C_{h+1} is B_{h+1} with all but the topmost node m_{h+1} deleted. Since the left subtree of u_{h+1} in U contains no nodes u_i, by Lemma 6.10 it only contains other nodes u_{h+1} and so $m_{h+1} = u_{h+1}$. Thus C_{h+1} consists of u_{h+1} and its right subtree. By defintion, q_{h+1} (if defined) is the least symbol greater than every

topmost symbol in B_{h+1}; since u_{h+1} is less than every symbol in B_h, this implies that $q_{h+1} = q_h$. By definition of the topmost traversal, there is no topmost node between u_h and u_{h+1}, so only (non-topmost) nodes q_{h+1} can lie between them. Thus C_{h+1} consists of u_{h+1}, some number s_2 (possibly zero) of secondary nodes q_{h+1}, and the subtree B_h. Hence D_{h+1} consists of u_{h+1}, these nodes q_{h+1}, and the subtree B_h with its tertiary nodes $q_{h+1} = q_h$ deleted, since these tertiary nodes are the same in both B_{h+1} and B_h.

Notice that, in T_h, all nodes m_h that are not in E_h itself are in the left-minimal subtree of E_h, but may not be consecutive. Since no m_h is below $p_h = u_{h+1}$, which is the greatest symbol less than m_h, it follows that the uppermost node u_{h+1} in T_h is in the left subtree of the lowest node m_h.

Furthermore, as shown in Figures 11 and 12 below, this uppermost node u_{h+1} in T_h must be on the path of right child nodes from the left child of the lowest node m_h, for otherwise it would be in a left subtree of some node x below the lowest node m_h, which would imply $u_{h+1} = p_h < x < m_h$, which contradicts p_h being the greatest symbol less than m_h.

Sub-case 3(a). Suppose that $E_h \neq D_h$. Then q_h is defined and D_h contains nodes q_h. Since q_h is greater than or equal to every node in D_h, it follows that the uppermost node q_h is where the right-maximal subtree of D_h (and E_h) is attached in T_h. The tree E_h consists of D_h with s nodes q_h inserted, where s is the number of nodes q_h that appear in U outside of the subtree D_h. Suppose there are $r+1$ nodes m_h in U, so there are r nodes m_h outside D_h in U.)

As shown in Figure 11, let λ_r be reading of the left-minimal subtree of D_h outside of the complete subtree at the uppermost m_h below D_h. (Note that λ_r may be empty.) For $i = r-1, \ldots, 2$, let λ_i be readings of the left subtree of the i-th node m_h (counting from the lowermost to the uppermostmost) outside of the complete subtree of the $i-1$-th. (Note that λ_i will be empty if the i-th node m_h is the left child of the $i+1$-th.) Let λ_0 be a reading of the left subtree of the bottommost node m_h outside the complete subtree at the uppermost node u_{h+1}. Let τ be a reading of the left subtree of the uppermost node u_{h+1}. Note that the uppermost non-empty subtree λ_i or τ contains all the E_{i_j} except E_h by P2. Let β be a reading of the right-maximal subtree of D_h.

Thus

$$T_h = \mathrm{P}_{\mathsf{sylv}}(\tau q_h^{s_2} u_{h+1} \lambda_0 m_h \lambda_1 m_h \cdots \lambda_{r-1} m_h \lambda_r q_h^{s_1} \beta D_h);$$

where $s_1 = s - s_2$. (Recall that s_2 was defined above as the number of secondary nodes q_h between u_h and $p_h = u_{h+1}$ in U.) Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(\lambda_0 m_h \lambda_1 m_h \cdots \lambda_{r-1} m_h \lambda_r q_h^{s_1} \beta D_h \tau q_h^{s_2} u_{h+1}),$$

Notice that $T_h \sim T_{h+1}$.

In computing T_{h+1}, the symbol u_{h+1} is inserted first and becomes the root node. Then the s_2 symbols q_h are inserted into the right subtree of the root nodes, since $q_h > p_h = u_{h+1}$. Every symbol in τ is less than or equal to u_{h+1} and so is inserted into the left subtree of the root note u_{h+1}. Every symbol in D_h is greater than u_{h+1} and less than or equal to q_h, so the tree D_h is re-inserted at the left child of the bottommost q_h. Every symbol in β is greater than q_h (since in T_h the subtree β is the right child of a node

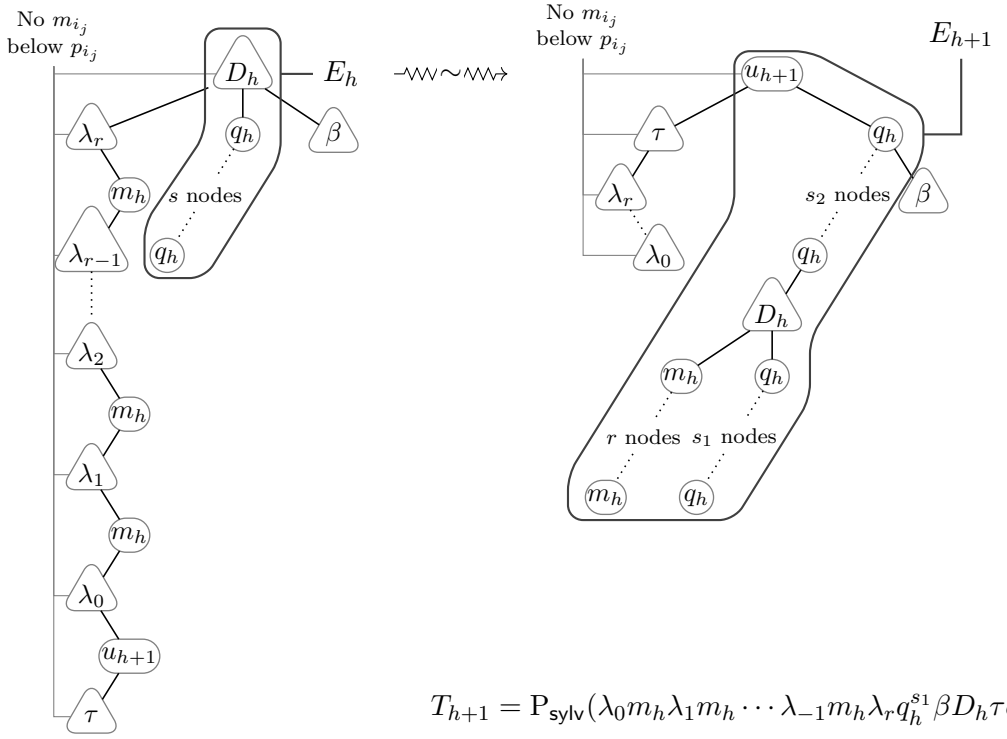


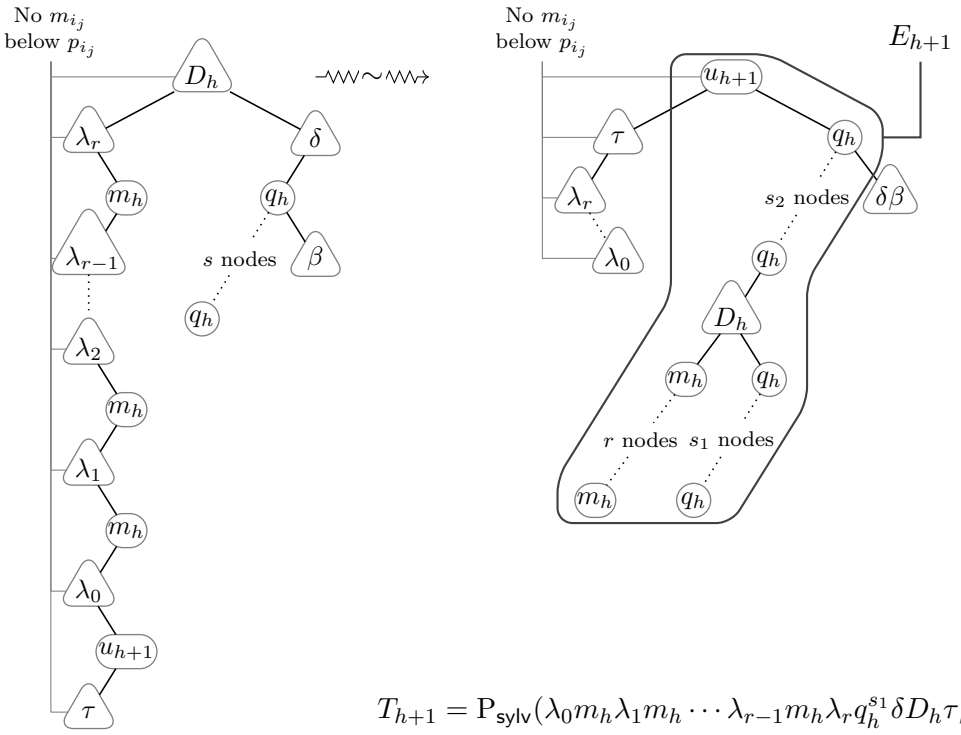
FIGURE 11. Induction step, sub-case 3(a): $E_h \neq D_h$.

q_h, as discussed above), so β is inserted as the right subtree of the topmost q_h. The remaining s_1 symbols q_h are re-inserted below D_h, attached at the same position as in T_h. The symbol m_h is the smallest symbol greater than u_{h+1} and all symbols m_h are inserted into the left-minimal subtree of D_h (which is attached at the single node m_h in D_h). Every symbol in every λ_i is less than every symbol in τ and so are inserted into the left-minimal subtree of τ. Since every symbol in λ_i is greater than every symbol in λ_{i+1}, the former is inserted into the right-maximal subtree of the latter.

As noted above, D_{h+1} consists of u_{h+1} with empty left subtree, s_2 nodes $q_{h+1} = q_h$, and the subtree B_h with its tertiary nodes $q_{h+1} = q_h$ deleted. Hence, since D_h contains secondary nodes q_h, so does D_{h+1}, and so E_{h+1} consists of D_{h+1} with s_1 nodes $q_{h+1} = q_h$ inserted, as shown in Figure 11.

Therefore E_{h+1} appears at the root of U and so T_{h+1} satisfies P1. The other trees E_{i_j} in T_h were in the uppermost non-empty λ_i or τ; this still holds and so T_{h+1} satisfies P2. Since T_h satisfies P3 and the insertions into the left-minimal and right-maximal subtrees of E_{h+1}, the tree T_{h+1} satisfies P3. Finally, T_{h+1} satisfies P4 since T_h does. (Note that m_h is below $p_h = u_{h+1}$, but this does not matter since u_h is not in $U_{h+1}^{\uparrow}$.)

Sub-case 3(b). Suppose that $E_h = D_h$. Then either q_h is undefined, or is defined but D_h does not contain nodes q_h.



$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(\lambda_0 m_h \lambda_1 m_h \cdots \lambda_{r-1} m_h \lambda_r q_h^{s_1} \delta D_h \tau \beta q_h^{s_2} u_{h+1})$$

$$T_h = \mathrm{P}_{\mathsf{sylv}}(\tau \beta q_h^{s_2} u_{h+1} \lambda_0 m_h \lambda_1 m_h \cdots \lambda_{r-1} m_h \lambda_r q_h^{s_1} \delta D_h)$$

FIGURE 12. Induction step, sub-case 3(b): $E_h = D_h$.

Suppose that q_h is defined. Let s be the number of nodes q_h that appear in U outside of the subtree D_h.

Since q_h is greater than every node in D_h, it follows that the s nodes q_h are all in the right-maximal subtree of D_h (and E_h) in T_h. Since q_h is the least symbol greater than every symbol in D_h, in T_h the left child of a node q_h can only be another node q_h.

Now return to the general situation, where q_h may or may not be defined. As shown in Figure 12, let λ_i and τ be as in sub-case 3(b). Let δ be a reading of the right-maximal subtree of D_h outside the complete subtree at the uppermost node q_h, and let β be a reading of the right subtree of the topmost node q_h. (If q_h is undefined, β is empty.)

Now,

$$T_h = \mathrm{P}_{\mathsf{sylv}}(\tau \beta q_h^{s_2} u_{h+1} \lambda_0 m_h \lambda_1 m_h \cdots \lambda_{r-1} m_h \lambda_r q_h^{s_1} \delta D_h),$$

where $s_1 = s - s_2$. (Recall that s_2 was defined above as the number of secondary nodes q_h between u_h and $p_h = u_{h+1}$ in U.) Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(\lambda_0 m_h \lambda_1 m_h \cdots \lambda_{r-1} m_h \lambda_r q_h^{s_1} \delta D_h \tau \beta q_h^{s_2} u_{h+1}).$$

Notice that $T_h \sim T_{h+1}$.

Assume for the moment that q_h is defined and $s_2 > 0$. In computing T_{h+1}, the symbol u_{h+1} is inserted first and becomes the root node. Then the s_2 symbols q_h are inserted into the right subtree of the root nodes, since

$q_h > u_{h+1}$. Every symbol in β is greater than q_h, so β is inserted as the right subtree of the topmost q_h. Every symbol in τ is less than or equal to u_{h+1} and so is inserted into the left subtree of the root node u_{h+1}. Every symbol in D_h is greater than u_{h+1} and less than q_h, so the tree D_h is re-inserted as the left child of the bottommost q_h. The remaining s_1 symbols q_h are re-inserted below D_h, attached at the right-maximal subtree of D_h. The symbol m_h is the smallest symbol greater than $p_h = u_{h+1}$ and so all symbols m_h are inserted into the the left-minimal subtree of D_h (which is attached at the single node m_h in T_h). Every symbol in every λ_i is less than every symbol in τ and so the λ_i are inserted into the left-minimal subtree of τ. Since every symbol in λ_i is greater than every symbol in λ_{i+1}, the former is inserted into the right-maximal subtree of the latter.

As noted above, D_{h+1} consists of u_{h+1} with empty left subtree, s_2 nodes $q_{h+1} = q_h$, and the subtree B_h with its tertiary nodes $q_{h+1} = q_h$ deleted. The tree D_{h+1} contains s_2 nodes $q_{h+1} = q_h$, and so E_{h+1} consists of D_{h+1} with s_1 nodes $q_{h+1} = q_h$ inserted, as shown in Figure 12. Thus E_{h+1} appears at the root of T_{h+1} and every node below it is in its left-minimal or right-maximal subtrees.

If, on the other hand, q_h is defined and $s_2 = 0$, then, by near-identical reasoning, D_h is inserted at the right child of u_{h+1} and $\delta\beta$ becomes the right-maximal subtree of D_h, and the $s = s_1$ nodes q_h are inserted into the tree $\delta\beta$.

In this case, D_{h+1} does not contain any nodes $q_{h+1} = q_h$ and so $E_{h+1} = D_{h+1}$ consists of u_{h+1} and the subtree B_h with its tertiary nodes $q_{h+1} = q_h$ deleted and so E_{h+1} appears at the root of T_{h+1} and every node below it is in its left-minimal or right-maximal subtrees.

Finally, if q_h is undefined, then again D_h is inserted at the right child of u_{h+1} and $\delta\beta$ becomes the right-maximal subtree of D_h.

In this case, q_{h+1} is also undefined and so there are no tertiary nodes in B_h. Hence $E_{h+1} = D_{h+1}$ consists precisely of u_{h+1} with right subtree B_h. So E_{h+1} appears at the root of T_{h+1} and every node below it is in its left-minimal or right-maximal subtrees.

Therefore T_{h+1} satisfies P1. The other trees E_{i_j} in T_h were in the uppermost non-empty λ_i or τ; this still holds and so T_{h+1} satisfies P2. Since T_h satisfies P3 and all nodes below E_{h+1} are in its left-minimal and right-maximal subtrees, the tree T_{h+1} satisfies P3. Finally, T_{h+1} satisfies P4 since T_h does. (Note that m_h is below p_{h+1}, but this does not matter since u_h is not in $U^{\uparrow}_{h+1}$.)

Case 4. Suppose that, in U, the node u_h is in the right subtree of u_{h+1}, and there is some node u_i in the left subtree of u_{h+1}.

Suppose there are topmost nodes u_j and $u_{j'}$ in this left subtree. Then their lowest common ancestor v is also in this left subtree; v must have both subtrees non-empty and so be a topmost node and thus lie in U_h. Thus there is a unique node in $U^{\uparrow}_h$ in the left subtree of v; clearly this is u_g, which is the rightmost node in $U^{\uparrow}_h \setminus \{u_h\}$. (Recall that, for brevity, $h = i_k$ and $g = i_{k-1}$.) Furthermore, this node is on the path of left child nodes from u_{h+1} (since only topmost nodes have right subtrees). Therefore,

$U^{\uparrow}_{h+1} = \left(U^{\uparrow}_h \setminus \{u_h, u_i\}\right) \cup \{u_{h+1}\}$, where u_i is the unique node from $U^{\uparrow}_h$ in the left subtree of u_{h+1}. As in case 3, $p_h = u_{h+1}$. The smallest symbol in the tree B_{h+1} is the smallest symbol in the left subtree of u_{h+1} (which is certainly non-empty in this case) and so $m_{h+1} = m_g$. The least symbol less than or equal to every symbol in B_{h+1} is the least symbol less than or equal to every symbol in the right subtree of u_{h+1} (which is certainly non-empty in this case) and so $q_h = q_{h+1}$ (or q_{h+1} is undefined if q_h is undefined). The only nodes between u_g and the topmost node u_{h+1} are other nodes u_{h+1}, so $q_g = u_{h+1}$ (in particular, q_g is defined). Finally, u_{h+1} is the first node entered from a right child on ascending from u_h to the root (by the definition of the topmost traversal) so and $u_{h+1} = p_h$.

Let s_2 be the number of secondary nodes q_h between u_h and u_{h+1} in U, and let t_2 is the number of primary nodes u_{h+1} and let $s_1 = s - s_2$.

Sub-case 4(a). Suppose that $E_h \neq D_h$ and $E_g \neq D_g$. Then q_h is defined, D_h contains nodes q_h, and D_g contains nodes q_g. Since q_h is greater than or equal to every node in D_h, it follows that the topmost node q_h is where the right-maximal subtree of D_h (and E_h) is attached in T_h. Similarly, since q_g is greater than or equal to every node in D_g, it follows that the topmost node q_g is where the right-maximal subtree of D_g (and E_g) is attached in T_h. The tree E_h consists of D_h with s nodes q_h inserted, where s is the number of nodes q_h that appear in U outside of the subtree D_h. The tree E_g consists of D_g with t nodes q_g inserted, where t is the number of nodes $q_g = u_{h+1}$ that appear in U outside of the subtree D_g.

By P2, the subtree E_g appears on the path of left child nodes and thus in the left-minimal subtree of D_h (and E_h). The only symbols that are less than or equal to m_h (the minimum symbol in E_h) and greater than or equal to $q_g = p_h = u_{h+1}$ (the maximum symbol in E_g) are the symbols m_h and $q_g = p_h = u_{h+1}$ themselves.

By P4, no node m_h appears below a node $p_h = q_g$, so nodes m_h cannot appear in the right-maximal subtree of E_g. Thus the nodes m_h (except for the single node m_h in E_h) are precisely the nodes on the path of left child nodes between E_h and E_g.

As shown in Figure 13, let λ be a reading of the left-minimal subtree of D_g and let β be a reading of the right-maximal subtree of D_h.

Then

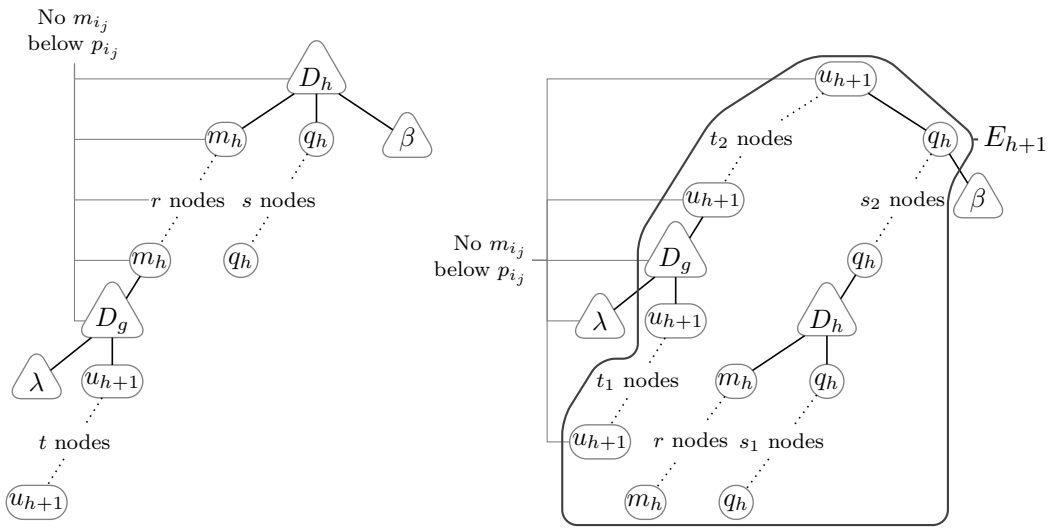
$$T_h = \mathrm{P}_{\mathsf{sylv}}(q_h^{s_2} u_{h+1}^{t} \lambda D_g \beta m_h^{r} q_h^{s_1} D_h).$$

Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{t_1} \lambda D_g \beta m_h^{r} q_h^{s_1} D_h q_h^{s_2} u_{h+1}^{t_2}).$$

Note that $T_h \sim T_{h+1}$.

In computing T_{h+1}, the rightmost symbol u_{h+1} is inserted first and becomes the root node, with the remaining $t_2 - 1$ symbols descending from it on the path of left child nodes. Since $q_h = q_{h+1} > u_{h+1}$, the s_2 symbols q_h are inserted into the right subtree of the root node u_{h+1}. Every symbol in D_h is greater than u_{h+1} and less than or equal to q_h, so the subtree D_h is re-inserted as the left child node of the bottommost node q_h. The remaining s_1 symbols q_h are inserted below D_h (attached at the same place as in T_h). The symbol m_h is the smallest symbol greater than u_{h+1} and so the r symbols m_h is re-inserted into the left-minimal subtree of D_h (attached as the



$$T_h = \mathrm{P}_{\mathsf{sylv}}(q_h^{s_2} u_{h+1}^{t} \lambda D_g \beta m_h^{r} q_h^{s_1} D_h) \qquad T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{t_1} \lambda D_g \beta m_h^{r} q_h^{s_1} D_h q_h^{s_2} u_{h+1}^{t_2})$$

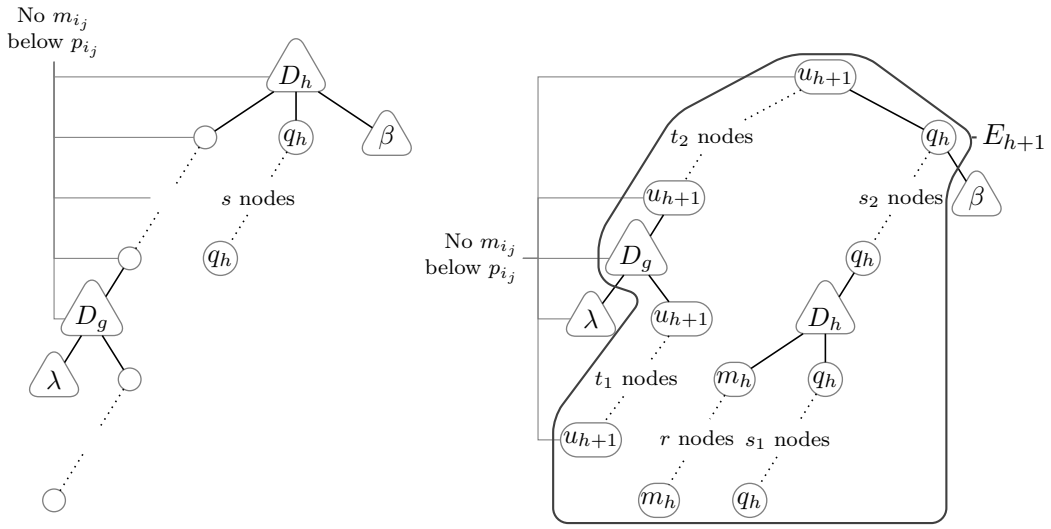
FIGURE 13. Induction step, sub-case 4(a): $E_h \neq D_h$ and $E_g \neq D_g$.

left child of the node m_h in D_h). Every symbol in β is greater than q_h, so β is re-inserted as the right child of the topmost q_h. Every symbol in D_g and λ is less than or equal to u_{h+1} and so D_g and λ are re-inserted at the left child of the bottommost node u_{h+1}. The remaining t_1 nodes u_{h+1} are into the right-maximal subtree of D_g (in the same place as they were attached in T_h).

The subtree D_h, the nodes m_h, and possibly some of the q_h below D_h make up B_h. This tree B_h, together with the nodes q_h above D_h, the nodes u_{h+1} and D_g together make up D_{h+1}. Adding the remaining nodes $q_h = q_{h+1}$ below D_h gives the tree E_{h+1}. So T_{h+1} satisfies P1. The other trees E_{i_j} in T_h were in λ; this still holds and so T_{h+1} satisfies P2. Every node not in E_{h+1} is in its left-minimal or right-maximal subtree; together with the fact that T_h satisfies P3, this shows that T_{h+1} satisfies P3. Finally, T_{h+1} satisfies P4 since T_h does. (Note that m_h is below $p_h = u_{h+1}$, but this does not matter since u_h is not in $U_{h+1}^{\uparrow}$.)

Sub-case 4(b). Suppose that $E_h \neq D_h$ and $E_g = D_g$. Then q_h is defined and D_h contains nodes q_h. Further, q_g is defined and $q_g = u_{h+1}$, but D_g does not contain nodes q_g. Since q_h is greater than or equal to every node in D_h, it follows that the topmost node q_h is where the right-maximal subtree of D_h (and E_h) is attached. The tree E_h consists of D_h with s nodes q_h inserted, where s is the number of nodes q_h that appear in U outside of the complete subtree at u_h.

By P2, in T_h the subtree E_g appears on the path of left child nodes and thus in the left-minimal subtree of E_h (and thus of D_h), which appears at the root of T_h by P1. The only symbols that are less than or equal to m_h



(1) $T_h = \mathrm{P}_{\mathsf{sylv}}(q_h^{s_2} u_{h+1}^{t} m_h^{r_1} \lambda D_g m_h^{r_2} \beta q_h^{s_1} D_h)$;
(2) $T_h = \mathrm{P}_{\mathsf{sylv}}(q_h^{s_2} u_{h+1}^{q_1} \lambda D_g u_{h+1}^{q_2} m_h^{r} \beta q_h^{s_1} D_h)$;
(3) $T_h = \mathrm{P}_{\mathsf{sylv}}(q_h^{s_2} u_{h+1}^{q_1} \lambda D_g u_{h+1}^{q_2} m_h^{r} \beta q_h^{s_1} D_h)$.

(1) $T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{t_1} m_h^{r_1} \lambda D_g m_h^{r_2} \beta q_h^{s_1} D_h q_h^{s_2} u_{h+1}^{t_2})$;
(2) $T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{q_2 - t_2} m_h^{r} \beta q_h^{s_1} D_h q_h^{s_2} u_{h+1}^{q_1} \lambda D_g u_{h+1}^{t_2})$;
(3) $T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{q_1 - t_2 + q_2} \lambda D_g u_{h+1}^{q_2} m_h^{r} \beta q_h^{s_1} D_h q_h^{s_2} u_{h+1}^{t_2 - q_2})$.

FIGURE 14. Induction step, sub-case 4(b): $E_h \neq D_h$ and $E_g = D_g$; sub-sub-cases (1)–(3) use different cyclic shifts.

(the minimum symbol in E_h) and greater than or equal to every symbol in E_g are m_h and $p_h = u_{h+1} = q_g$. Thus nodes m_h and u_{h+1} must appear either on the path of left child nodes between E_h and E_g, or in the right-maximal subtree of E_g.

As shown in Figure 14, let λ be a reading of the left-minimal subtree of D_g and let β be a reading of the right-maximal subtree of D_h. The nodes shown as empty consist of r nodes m_h and t nodes u_{h+1} (for some r and t), with the m_h above the u_{h+1} by condition P4 since $u_{h+1} = p_h$. Note, however, that the boundary between the m_h and the u_{h+1} may be either above or below D_g.

Consider three sub-sub-cases:

(1) There are no nodes u_{h+1} above D_g. Suppose there are r_1 nodes m_h below D_g and r_2 above. Then

$$T_h = \mathrm{P}_{\mathsf{sylv}}(q_h^{s_2} u_{h+1}^{t} m_h^{r_1} \lambda D_g m_h^{r_2} \beta q_h^{s_1} D_h).$$

Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{t_1} m_h^{r_1} \lambda D_g m_h^{r_2} \beta q_h^{s_1} D_h q_h^{s_2} u_{h+1}^{t_2}).$$

Note that $T_h \sim T_{h+1}$.

In computing T_{h+1}, the rightmost symbol u_{h+1} is inserted first and becomes the root node, with the remaining $t_2 - 1$ symbols descending from it on the path of left child nodes. Since $q_h = q_{h+1} > u_{h+1}$, the s_2 symbols q_h are inserted into the right subtree of the root node u_{h+1}. Every symbol in D_h is greater than u_{h+1} and less than or

equal to q_h, so if $s_2 > 0$, the subtree D_h is re-inserted as the left child node of the bottommost node q_h, while if $s_2 = 0$, the subtree D_h is inserted as the right child of u_{h+1}. The remaining s_1 symbols q_h are inserted below D_h (attached at the same place as in T_h). Every symbol in β is greater than q_h, so if $s_2 > 0$, the subtree β is re-inserted as the right child of the topmost q_h, while if $s_2 = 0$, the subtree β is inserted as the right-maximal subtree of D_h. The symbol m_h is the smallest symbol greater than u_{h+1} and so the r_2 symbols m_h are inserted into the left-minimal subtree of D_h (attached as the left child of the node m_h in D_h). Every symbol in D_g and λ is less than or equal to u_{h+1} and so D_g and λ are re-inserted at the left child of the bottommost node u_{h+1}. The remaining r_1 nodes m_h are inserted at the left child of the bottommost node m_h (so that the nodes m_h are now consecutive). The remaining t_1 nodes u_{h+1} are into the right-maximal subtree of D_g.

(2) There are o_1 nodes u_{h+1} below D_g and o_2 above, where $o_2 \geq t_2$. Then

$$T_h = \mathrm{P}_{\mathsf{sylv}}(q_h^{s_2} u_{h+1}^{o_1} \lambda D_g u_{h+1}^{o_2} m_h^r \beta q_h^{s_1} D_h).$$

Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{o_2-t_2} m_h^r \beta q_h^{s_1} D_h q_h^{s_2} u_{h+1}^{o_1} \lambda D_g u_{h+1}^{t_2}).$$

Note that $T_h \sim T_{h+1}$.

In computing T_{h+1}, the rightmost symbol u_{h+1} is inserted first and becomes the root node, with the remaining $t_2 - 1$ symbols descending from it on the path of left child nodes. Every symbol in D_g and λ is less than or equal to u_{h+1} and so D_g and λ are re-inserted at the left child of the bottommost node u_{h+1}. The next o_1 nodes u_{h+1} are inserted into the right-maximal subtree D_g. Since $q_h = q_{h+1} > u_{h+1}$, the s_2 symbols q_h are inserted into the right subtree of the root node u_{h+1}. Every symbol in D_h is greater than u_{h+1} and less than or equal to q_h, so if $s_2 > 0$, the subtree D_h is re-inserted as the left child node of the bottommost node q_h, while if $s_2 = 0$, the subtree D_h is inserted as the right child of u_{h+1}. The remaining s_1 symbols q_h are inserted below D_h (attached at the same place as in T_h). Every symbol in β is greater than q_h, so if $s_2 > 0$, the subtree β is re-inserted as the right child of the topmost q_h, while if $s_2 = 0$, the subtree β is attached at as the right-maximal subtree of D_h. The symbol m_h is the smallest symbol greater than u_{h+1} and so the r symbols m_h are inserted into the left-minimal subtree of D_h (attached as the left child of the node m_h in D_h). The remaining $o_2 - t_2$ nodes u_{h+1} are inserted at the left child of the bottommost node u_{h+1} (so that the $o_1 + o_2 - t_2 = t_1$ nodes u_{h+1} below D_g are now consecutive).

(3) There are o_1 nodes u_{h+1} below D_g and o_2 above, where $o_2 < t_2$. Then

$$T_h = \mathrm{P}_{\mathsf{sylv}}(q_h^{s_2} u_{h+1}^{o_1} \lambda D_g u_{h+1}^{o_2} m_h^r \beta q_h^{s_1} D_h).$$

Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{o_1-t_2+o_2} \lambda D_g u_{h+1}^{o_2} m_h^r \beta q_h^{s_1} D_h q_h^{s_2} u_{h+1}^{t_2-o_2}).$$

Note that $o_1 - t_2 + o_2 = t_1$ and that $T_h \sim T_{h+1}$.

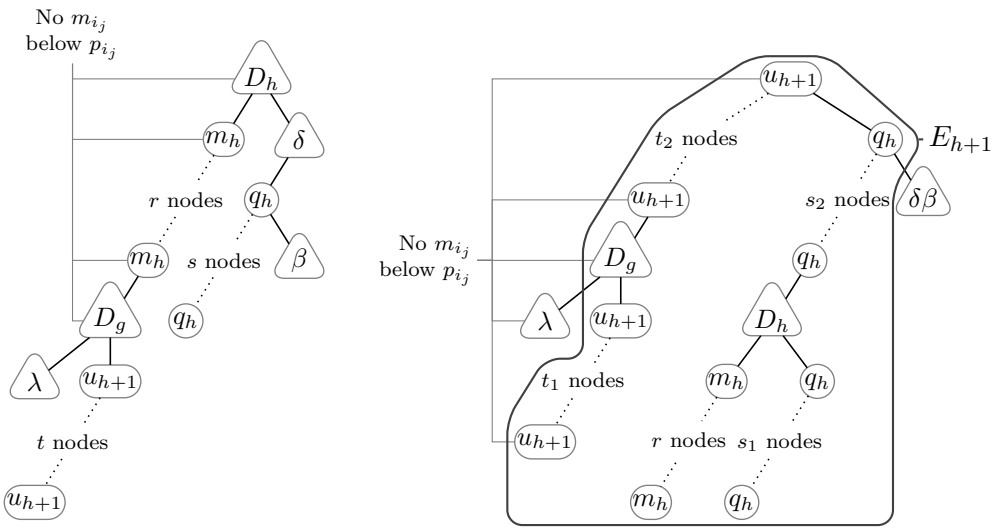
In computing T_{h+1}, the rightmost symbol u_{h+1} is inserted first and becomes the root node, with the remaining $t_2 - o_2 - 1$ symbols descending from it on the path of left child nodes. Since $q_h = q_{h+1} > u_{h+1}$, the s_2 symbols q_h are inserted into the right subtree of the root node u_{h+1}. Every symbol in D_h is greater than u_{h+1} and less than or equal to q_h, so if $s_2 > 0$, the subtree D_h is re-inserted as the left child node of the bottommost node q_h, while if $s_2 = 0$, the subtree D_h is inserted as the right child of u_{h+1}. The remaining s_1 symbols q_h are inserted below D_h (attached at the same place as in T_h). Every symbol in β is greater than q_h, so if $s_2 > 0$, the subtree β is re-inserted as the right child of the topmost q_h, while if $s_2 = 0$, the subtree β is attached at as the right-maximal subtree of D_h. The symbol m_h is the smallest symbol greater than u_{h+1} and so the r symbols m_h are inserted into the left-minimal subtree of D_h (attached as the left child of the node m_h in D_h). The next o_2 symbols u_{h+1} are inserted into the left subtree of the bottommost node u_{n+1}, so that there are $t_2 - o_2 + o_2 = t_2$ consecutive nodes u_{h+1}. Every symbol in D_g and λ is less than or equal to u_{h+1} and so D_g and λ are re-inserted at the left child of the bottommost node u_{h+1}. The remaining $o_1 - t_2 + o_2 = t_1$ nodes u_{h+1} are inserted into the right-maximal subtree of D_g.

The subtree D_h, the nodes m_h, and possibly some of the q_h below D_h make up B_h. This tree B_h, together with the nodes q_h above D_h, the nodes u_{h+1} and D_g together make up D_{h+1}. Adding the remaining nodes $q_h = q_{h+1}$ below D_h gives the tree E_{h+1}. So T_{h+1} satisfies P1. The other trees E_{i_j} in T_2 were in λ; this still holds and so T_{h+1} satisfies P2. Every node not in E_{h+1} is in its left-minimal or right-maximal subtree; together with the fact that T_h satisfies P3, this shows that T_{h+1} satisfies P3. Finally, T_{h+1} satisfies P4 since T_h does. (Note that m_h is below $p_h = u_{h+1}$, but this does not matter since u_h is not in $U^{\uparrow}_{h+1}$.)

Sub-case 4(c). Suppose that $E_h = D_h$ and $E_g \neq D_g$. Then D_g contains nodes q_g. Since q_g is greater than or equal to every node in D_g, it follows that the topmost node q_g is where the right-maximal subtree of D_g (and E_g) is attached in T_h. The tree E_g consists of D_g with t nodes q_g inserted, where t is the number of nodes $q_g = u_{h+1}$ that appear in U outside of the complete subtree at u_g.

The subtree E_g appears on the path of left child nodes and thus in the left-minimal subtree of E_h (and E_h). The only symbols that is less than or equal to m_h and greater than or equal to every symbol in q_g are m_h and $p_h = u_{h+1} = q_g$. By P4, no node m_h appears below a node p_h, so nodes m_h cannot appear in the right-maximal subtree of E_g. Thus the nodes m_h (except for the single node m_h in D_h) are precisely the nodes on the path of left child nodes between D_h and D_g.

As shown in Figure 15, let λ be a reading of the left-minimal subtree of D_g. Let δ be a reading of the right-maximal subtree of D_h outside the complete subtree at the topmost node q_h (if q_h is defined) and let β be a reading of the right-maximal subtree of the topmost q_h. (Note that β is empty if q_h is not defined.)



(1) $T_h = \mathrm{P}_{\mathsf{sylv}}(\beta q_h^{s_2} u_{h+1}^t \lambda D_g m_h^r q_h^{s_1} \delta D_h)$;
(2) $T_h = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^t \lambda D_g m_h^r \beta q_h^s \delta D_h)$.

(1) $T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{t_1} \lambda D_g m_h^r \beta q_h^s \delta D_h u_{h+1}^{t_2})$;
(2) $T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{t_1} \lambda D_g m_h^r \beta q_h^s \delta D_h u_{h+1}^{t_2})$.

FIGURE 15. Induction step, sub-case 4(c): $E_h = D_h$ and $E_g \neq D_g$; sub-sub-cases (1) and (3) use different cyclic shifts.

Let s_2 be the number of secondary nodes q_h between u_h and u_{h+1} in U, and let t_2 be the number of primary nodes u_{h+1}. Note that if q_h is defined, then $s_2 < s$ since there must be at least one primary node q_h. Consider two sub-sub-cases:

(1) Suppose $s_2 > 0$ and q_h is defined. Then

$$T_h = \mathrm{P}_{\mathsf{sylv}}(\beta q_h^{s_2} u_{h+1}^t \lambda D_g m_h^r q_h^{s_1} \delta D_h).$$

Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{t_1} \lambda D_g m_h^r q_h^{s_1} \delta D_h \beta q_h^{s_2} u_{h+1}^{t_2}).$$

Note that $T_h \sim T_{h+1}$.

In computing T_{h+1}, the rightmost symbol u_{h+1} is inserted first and becomes the root node, with the remaining $t_2 - 1$ symbols descending from it on the path of left child nodes. Since $q_h = q_{h+1} > u_{h+1}$, the s_2 symbols q_h are inserted into the right subtree of the root node u_{h+1}. Every symbol in β is greater than q_h, so β is re-inserted as the right child of the topmost q_h. Every symbol in D_h is greater than u_{h+1} and less than or equal to q_h, so the subtree D_h is re-inserted as the left child node of the bottommost node q_h. Every symbol in δ is greater than q_h, so δ is inserted into the subtree β. The remaining s_1 symbols q_h are inserted into the right-maximal subtree of D_h. The symbol m_h is the smallest symbol greater than u_{h+1} and so the r symbols m_h are re-inserted into the left-minimal subtree of D_h (attached as the left child of the node m_h in D_h). Every symbol in D_g and λ is less than or equal to u_{h+1} and so D_g and λ are re-inserted at the left child of the bottommost node u_{h+1}. The

remaining t_1 nodes u_{h+1} are into D_g (in the same place as they were attached in T_h).

(2) Suppose $s_2 = 0$ or q_h is not defined. Then

$$T_h = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^t \lambda D_g m_h^r \beta q_h^s \delta D_h).$$

(If q_h is undefined, formally treat q_h and β as empty and s as 0.) Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{t_1} \lambda D_g m_h^r \beta q_h^s \delta D_h u_{h+1}^{t_2}),$$

Note that $T_h \sim T_{h+1}$.

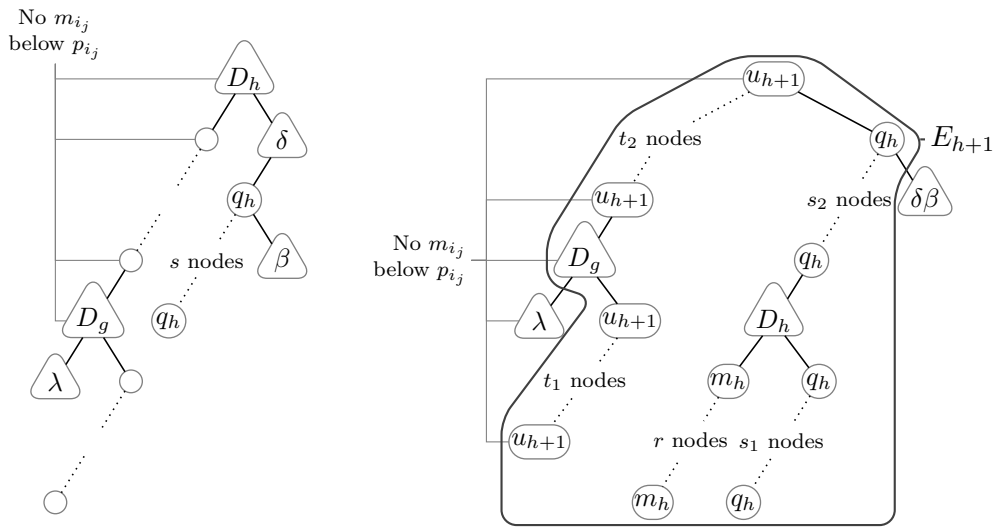
In computing T_{h+1}, the rightmost symbol u_{h+1} is inserted first and becomes the root node, with the remaining $t_2 - 1$ symbols descending from it on the path of left child nodes. Every symbol in D_h and δ is greater than u_{h+1}, so the subtrees D_h and δ are re-inserted as the right child node of the root node u_{h+1}. The s symbols q_h are inserted into te right-maximal subtree of D_h. Every symbol in β is greater than every symbol in D_h, so β is inserted into the subtree δ. The symbol m_h is the smallest symbol greater than u_{h+1} and so the r symbols m_h are re-inserted into the left-minimal subtree of D_h (attached as the left child of the node m_h in D_h). Every symbol in D_g and λ is less than or equal to u_{h+1} and so D_g and λ are re-inserted at the left child of the bottommost node u_{h+1}. The remaining t_1 nodes u_{h+1} are into D_g (in the same place as they were attached in T_h).

(The key difference between the sub-sub-case is the different placement of β in the reading of T_h. In sub-sub-case (2), the subword $\beta q_h^s \delta D_h$ makes up the entire part of T_h outside of the left-minimal subtree of D_h, and since this subword is intact in the reading of T_{h+1}, it will appear as the right subtree of the root, as shown by the reasoning below. This observation will be used to abbreviate some of the sub-sub-cases in sub-case 4(d) below.)

The subtree D_h, the nodes m_h, and possibly some of the q_h below D_h make up B_h. This tree B_h, together with any nodes q_h above D_h, the nodes u_{h+1} and D_g together make up D_{h+1}. Adding the remaining nodes $q_h = q_{h+1}$ below D_h gives the tree E_{h+1}. So T_{h+1} satisfies P1. The other trees E_{i_j} in T_2 were in λ; this still holds and so T_{h+1} satisfies P2. Every node not in E_{h+1} is in its left-minimal or right-maximal subtree; together with the fact that T_h satisfies P3, this shows that T_{h+1} satisfies P3. Finally, T_{h+1} satisfies P4 since T_h does. (Note that m_h is below $p_h = u_{h+1}$, but this does not matter since u_h is not in $U_{h+1}^{\uparrow}$.)

Sub-case 4(d). Suppose that $E_h = D_h$ and $E_g = D_g$. If q_h is defined then it is the least symbol greater than any symbol in D_h and so all nodes q_h must appear in the right-maximal subtree of E_h in T_h, and the left child of a node q_h can only be another node q_h.

It is possible that q_g is defined, in which case $q_g = u_{h+1}$, but D_g does not contain nodes q_g. The subtree tree D_g appears on the path of left child nodes and thus in the left-minimal subtree of D_h (and E_h). The only symbols that is less than or equal to every symbol in D_h and greater than or equal to every symbol in D_g are m_h and $p_h = u_{h+1}$ (which is equal to q_g, if q_g is



(1) $T_h = \mathrm{P}_{\mathsf{sylv}}(\beta q_h^{s_2} u_{h+1}^{t} m_h^{r_1} \lambda D_g m_h^{r_2} q_h^{s_1} \delta D_h)$;
(2) $T_h = \mathrm{P}_{\mathsf{sylv}}(\beta q_h^{s_2} u_{h+1}^{o_1} \lambda D_g u_{h+1}^{o_2} m_h^{r} q_h^{s_1} \delta D_h)$;
(3) $T_h = \mathrm{P}_{\mathsf{sylv}}(\beta q_h^{s_2} u_{h+1}^{o_1} \lambda D_g u_{h+1}^{o_2} m_h^{r} q_h^{s_1} \delta D_h)$;
(4) $T_h = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{t} m_h^{r_1} \lambda D_g m_h^{r_2} \beta q_h^{s} \delta D_h)$;
(5) $T_h = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{o_1} \lambda D_g u_{h+1}^{o_2} m_h^{r} \beta q_h^{s} \delta D_h)$;
(6) $T_h = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{o_1} \lambda D_g u_{h+1}^{o_2} m_h^{r} \beta q_h^{s} \delta D_h)$.

(1) $T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{t_1} m_h^{r_1} \lambda D_g m_h^{r_2} q_h^{s_1} \delta D_h \beta q_h^{s_2} u_{h+1}^{t_2})$;
(2) $T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{o_2-t_2} m_h^{r} q_h^{s_1} \delta D_h \beta q_h^{s_2} u_{h+1}^{o_1} \lambda D_g u_{h+1}^{t_2})$;
(3) $T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{o_1-t_2+o_2} \lambda D_g u_{h+1}^{o_2} m_h^{r} q_h^{s_1} \delta D_h \beta q_h^{s_2} u_{h+1}^{t_2-o_2})$;
(4) $T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{t_1} m_h^{r_1} \lambda D_g m_h^{r_2} \beta q_h^{s} \delta D_h u_{h+1}^{t_2})$;
(5) $T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{o_2-t_2} m_h^{r} \beta q_h^{s} \delta D_h u_{h+1}^{o_1} \lambda D_g u_{h+1}^{t_2})$;
(6) $T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{o_1-t_2+o_2} \lambda D_g u_{h+1}^{o_2} m_h^{r} \beta q_h^{s} \delta D_h u_{h+1}^{t_2-o_2})$.

FIGURE 16. Induction step, sub-case 4(d): $E_h = D_h$ and $E_g = D_g$; sub-sub-cases (1)–(6) use different cyclic shifts.

defined). Thus nodes m_h and u_{h+1} must appear either on the path of left child nodes between D_h and D_g, or in the right-maximal subtree of D_g.

As shown in Figure 16, let λ be a reading of the left-minimal subtree of D_g. Let δ be a reading of the right-maximal subtree of D_h outside the complete subtree at the topmost node q_h (if q_h is defined) and let β be a reading of the right-maximal subtree of the topmost q_h. (Note that β is empty if q_h is not defined.) The empty nodes are filled with r nodes m_h and t nodes u_{h+1}, with the m_h above the u_{h+1}. Note, however, that the boundary between the m_h and the u_{h+1} may be either above or below D_g.

Let s_2 be the number of secondary nodes q_h between u_h and u_{h+1} in U, and let t_2 is the number of primary nodes u_{h+1}. There are six sub-sub-cases: consider first the three sub-sub-cases where $s_2 > 0$:

(1) There are no u_{h+1} above D_g. Suppose there are r_1 nodes m_h below D_g and r_2 above. Then

$$T_h = \mathrm{P}_{\mathsf{sylv}}(\beta q_h^{s_2} u_{h+1}^{t} m_h^{r_1} \lambda D_g m_h^{r_2} q_h^{s_1} \delta D_h).$$

Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{t_1} m_h^{r_1} \lambda D_g m_h^{r_2} q_h^{s_1} \delta D_h \beta q_h^{s_2} u_{h+1}^{t_2}),$$

Note that $T_h \sim T_{h+1}$.

In computing T_{h+1}, the rightmost symbol u_{h+1} is inserted first and becomes the root node, with the remaining t_2-1 symbols descending from it on the path of left child nodes. Since $q_h = q_{h+1} > u_{h+1}$, the s_2 symbols q_h are inserted into the right subtree of the root node u_{h+1}. Every symbol in β is greater than q_h, so the subtree β is re-inserted as the right child of the topmost q_h. Every symbol in D_h is greater than u_{h+1} and less than or equal to q_h, so the subtree D_h is re-inserted as the left child node of the bottommost node q_h. Every symbol in δ is greater than q_h, so δ is inserted into the subtree β. The remaining s_1 symbols q_h are inserted into the right-maximal subtree of D_h. The symbol m_h is the smallest symbol greater than u_{h+1} and so the r_2 symbols m_h are inserted into the left-minimal subtree of D_h (attached as the left child of the node m_h in D_h). Every symbol in D_g and λ is less than or equal to u_{h+1} and so D_g and λ are re-inserted at the left child of the bottommost node u_{h+1}. The remaining r_1 nodes m_h are inserted at the left child of the bottommost node m_h (so that the nodes m_h are now consecutive). The remaining t_1 nodes u_{h+1} are into the right-maximal subtree of D_g.

(2) There are o_1 nodes u_{h+1} below D_g and o_2 above, where $o_2 \geq t_2$. Then

$$T_h = \mathrm{P}_{\mathsf{sylv}}(\beta q_h^{s_2} u_{h+1}^{o_1} \lambda D_g u_{h+1}^{o_2} m_h^r q_h^{s_1} \delta D_h).$$

Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{o_2-t_2} m_h^r q_h^{s_1} \delta D_h \beta q_h^{s_2} u_{h+1}^{o_1} \lambda D_g u_{h+1}^{t_2}).$$

Note that $T_h \sim T_{h+1}$.

In computing T_{h+1}, the rightmost symbol u_{h+1} is inserted first and becomes the root node, with the remaining t_2-1 symbols descending from it on the path of left child nodes. Every symbol in D_g and λ is less than or equal to u_{h+1} and so D_g and λ are re-inserted at the left child of the bottommost node u_{h+1}. The next o_1 nodes u_{h+1} are inserted into the right-maximal subtree of D_g. Since $q_h = q_{h+1} > u_{h+1}$, the s_2 symbols q_h are inserted into the right subtree of the root node u_{h+1}. Every symbol in β is greater than q_h, so subtree β is re-inserted as the right child of the topmost q_h. Every symbol in D_h is greater than u_{h+1} and less than or equal to q_h, so the subtree D_h is re-inserted as the left child node of the bottommost node q_h. Every symbol in δ is greater than q_h and so δ is inserted into the subtree β. The remaining s_1 symbols q_h are inserted into the right-maximal subtree of D_h. The symbol m_h is the smallest symbol greater than u_{h+1} and so the r symbols m_h are inserted into the left-minimal subtree of D_h (attached as the left child of the node m_h in D_h). The remaining $o_2 - t_2$ nodes u_{h+1} are inserted at the left child of the bottommost node u_{h+1} (so that the $o_1 + o_2 - t_2 = t_1$ nodes u_{h+1} below D_g are now consecutive).

(3) There are o_1 nodes u_{h+1} below D_g and o_2 above, where $o_2 < t_2$. Then

$$T_h = \mathrm{P}_{\mathsf{sylv}}(\beta q_h^{s_2} u_{h+1}^{o_1} \lambda D_g u_{h+1}^{o_2} m_h^r q_h^{s_1} \delta D_h).$$

Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{o_1 - t_2 + o_2} \lambda D_g u_{h+1}^{o_2} m_h^r q_h^{s_1} \delta D_h \beta q_h^{s_2} u_{h+1}^{t_2 - o_2}).$$

Note that $o_1 - t_2 + o_2 = t_1$ and that $T_h \sim T_{h+1}$.

In computing T_{h+1}, the rightmost symbol u_{h+1} is inserted first and becomes the root node, with the remaining $t_2 - o_2 - 1$ symbols descending from it on the path of left child nodes. Since $q_h = q_{h+1} > u_{h+1}$, the s_2 symbols q_h are inserted into the right subtree of the root node u_{h+1}. Every symbol in β is greater than q_h, so the subtree β is re-inserted as the right child of the topmost q_h. Every symbol in D_h is greater than u_{h+1} and less than or equal to q_h, so the subtree D_h is re-inserted as the left child node of the bottommost node q_h. Every symbol in δ is greater than q_h, so δ is inserted into the subtree β. The remaining s_1 symbols q_h are inserted into the right-maximal subtree of D_h. The symbol m_h is the smallest symbol greater than u_{h+1} and so the r symbols m_h are inserted into the left-minimal subtree of D_h (attached as the left child of the node m_h in D_h). The next o_2 symbols u_{h+1} are inserted into the left subtree of the bottommost node u_{n+1}, so that there are $t_2 - o_2 + o_2 = t_2$ consecutive nodes u_{h+1}. Every symbol in D_g and λ is less than or equal to u_{h+1} and so D_g and λ are re-inserted at the left child of the bottommost node u_{h+1}. The remaining $o_1 - t_2 + o_2 = t_1$ nodes u_{h+1} are inserted into the right-maximal subtree of D_g.

The three sub-sub-cases where $s_2 = 0$ (see below) differ from the above three sub-sub-cases in the same way that the two sub-sub-cases in subcase 4(c) differ: instead of taking a reading of T_h with β at the start, take one where β appears just before the string q_h^s. Since the reasoning is so similar, these sub-sub-cases are thus treated in an abbreviated form:

(4) There are no u_{h+1} above D_g. Suppose there are r_1 nodes m_h below D_g and r_2 above. Then

$$T_h = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^t m_h^{r_1} \lambda D_g m_h^{r_2} \beta q_h^s \delta D_h).$$

Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{t_1} m_h^{r_1} \lambda D_g m_h^{r_2} \beta q_h^s \delta D_h u_{h+1}^{t_2}),$$

Note that $T_h \sim T_{h+1}$.

(5) There are o_1 nodes u_{h+1} below D_g and o_2 above, where $o_2 \geq t_2$. Then

$$T_h = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{o_1} \lambda D_g u_{h+1}^{o_2} m_h^r \beta q_h^s \delta D_h).$$

Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{o_2 - t_2} m_h^r \beta q_h^s \delta D_h u_{h+1}^{o_1} \lambda D_g u_{h+1}^{t_2}).$$

(6) There are o_1 nodes u_{h+1} below D_g and o_2 above, where $o_2 < t_2$. Then

$$T_h = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{o_1} \lambda D_g u_{h+1}^{o_2} m_h^r \beta q_h^s \delta D_h).$$

Let

$$T_{h+1} = \mathrm{P}_{\mathsf{sylv}}(u_{h+1}^{o_1-t_2+o_2}\lambda D_g u_{h+1}^{o_2} m_h^r \beta q_h^s \delta D_h u_{h+1}^{t_2-o_2}).$$

Note that $o_1 - t_2 + o_2 = t_1$ and that $T_h \sim T_{h+1}$.

The subtree D_h, the nodes m_h, and possibly some of the q_h below D_h make up B_h. This tree B_h, together with the nodes q_h above D_h, the nodes u_{h+1} and D_g together make up D_{h+1}. Adding the remaining nodes $q_h = q_{h+1}$ below D_h gives the tree E_{h+1}. So T_{h+1} satisfies P1. The other trees E_{i_j} in T_2 were in λ; this still holds and so T_{h+1} satisfies P2. Every node not in E_{h+1} is in its left-minimal or right-maximal subtree; together with the fact that T_h satisfies P3, this shows that T_{h+1} satisfies P3. Finally, T_{h+1} satisfies P4 since T_h does. (Note that m_h is below $p_h = u_{h+1}$, but this does not matter since u_h is not in $U_{h+1}^{\uparrow}$.)

Conclusion. The tree T_n satisfies the conditions P1–P4. In particular, E_n appears at the root of T_n. Since u_n is the root of U (since it is obviously the last node visited by the topmost traversal), $B_n = U$. Thus C_n is U with all nodes m_n deleted except the topmost. Since q_n is undefined, $E_n = D_n = C_n$. Hence C_n appears at the roots of T_n and U. However, the number of nodes m_n in the trees T_n and U are equal, and since m_n is the smallest symbol appearing in T_n or U, all the nodes m_n in T_n and U must appear in the paths of left child nodes in T_n and U, in the left subtree of the single node m_n in E_n. Hence $T_n = U$.

Thus there is a sequence $T = T_0, T_1, \ldots, T_n = U$ with $T_i \sim T_{i+1}$ for $i \in \{1, \ldots, n-1\}$. Since T and U were arbitrary elements of an abritrary connected component of $K(\mathsf{sylv}_n)$, the diameter of any connected component of $K(\mathsf{sylv}_n)$ is at most n. This completes the proof. □

6.6. **Summary.** Combining Lemma 6.3 and Proposition 6.12 gives the following result:

Theorem 6.13. (1) *Connected components of* $K(\mathsf{sylv})$ *coincide with* $\equiv_{\mathrm{ev}}$*-classes of* sylv.

(2) *The maximum diameter of a connected component of* $K(\mathsf{sylv}_n)$ *is* $n-1$ *or* n.

Computational evidence strongly suggests that the maximum diameter of a connected component of $K(\mathsf{sylv}_n)$ is in fact $n-1$. When one explicitly constructs of a path of length n between two $\equiv_{\mathrm{ev}}$-related elements of $K(\mathsf{sylv}_n)$ using the strategy in the proof Proposition 6.12, there seems always to be a fairly 'obvious' way to combine two steps and so obtain a path of length $n-1$. However, these steps can apparently be located anywhere along the path and do not seem to exhibit any pattern. Thus the authors have been unable to prove that there are always two steps which it is possible to combine.

7. Stalactic monoid

The stalactic monoid is primarily used in the definition of the more interesting taiga monoid (see Section 8), but its cyclic shift graph exhibits a

particularly simple structure. As usual, this section recalls only the essentials background; for further reading, see [11].

A *stalactic tableau* is a finite array of symbols from $\mathcal{A}$ in which columns are top-aligned, and two symbols appear in the same column if and only if they are equal. For example,

$$\begin{array}{|c|c|c|c|c|}\hline 3&1&2&6&5\\\hline 3&1&&6&5\\\cline{1-2}\cline{4-5} &1& & &5\\\cline{2-2}\cline{5-5} &1\\\cline{2-2}\end{array} \tag{7.1}$$

is a stalactic tableau. The insertion algorithm is very straightforward:

Algorithm 7.1.

Input: A stalactic tableau T and a symbol $a \in \mathcal{A}$.

Output: A stalactic tableau $T \leftarrow a$.

Method: If a does not appear in T, add a to the left of the top row of T. If a does appear in T, add a to the bottom of the (by definition, unique) column in which a appears. Output the new tableau.

Thus one can compute, for any word $u \in \mathcal{A}^*$, a stalactic tableau $\mathrm{P}_{\mathsf{stal}}(u)$ by starting with an empty stalactic tableau and successively inserting the symbols of u, proceeding right-to-left through the word. For example $\mathrm{P}_{\mathsf{stal}}(361135112565)$ is (7.1). Notice that the order in which the symbols appear along the first row in $\mathrm{P}_{\mathsf{stal}}(u)$ is the same as the order of the rightmost instances of the symbols that appear in u. Define the relation $\equiv_{\mathsf{stal}}$ by

$$u \equiv_{\mathsf{stal}} v \iff \mathrm{P}_{\mathsf{stal}}(u) = \mathrm{P}_{\mathsf{stal}}(v)$$

for all $u, v \in \mathcal{A}^*$. The relation $\equiv_{\mathsf{stal}}$ is a congruence, and the *stalactic monoid*, denoted stal, is the factor monoid $\mathcal{A}^*/\equiv_{\mathsf{stal}}$; The *stalactic monoid of rank* n, denoted stal_n, is the factor monoid $\mathcal{A}_n^*/\equiv_{\mathsf{stal}}$ (with the natural restriction of $\equiv_{\mathsf{stal}}$). Each element $[u]_{\equiv_{\mathsf{stal}}}$ (where $u \in \mathcal{A}^*$) can be identified with the stalactic tableau $\mathrm{P}_{\mathsf{stal}}(u)$. Note that if T is a stalactic tableau consisting of a single row (that is, will all columns having height 1), then there is a unique word $u \in \mathcal{A}^*$, formed by reading the entries of T left-to-right, such that $\mathrm{P}_{\mathsf{stal}}(u) = T$. Thus, if $T = \mathrm{P}_{\mathsf{stal}}(a_1 \cdots a_k)$ and $U \in \mathit{stal}$ is such that $U \sim T$, then $U = \mathrm{P}_{\mathsf{stal}}(a_i \cdots a_k a_1 \cdots a_{i-1})$ for some i.

The monoid stal is presented by $\langle \mathcal{A} \,|\, \mathcal{R}_{\mathsf{stal}} \rangle$, where

$$\mathcal{R}_{\mathsf{stal}} = \big\{ (bavb, abvb) : a, b \in A,\ v \in A^* \big\}.$$

The monoid stal_n is presented by $\langle \mathcal{A}_n \,|\, \mathcal{R}_{\mathsf{stal}} \rangle$, where the set of defining relations $\mathcal{R}_{\mathsf{stal}}$ is naturally restricted to $\mathcal{A}_n^* \times \mathcal{A}_n^*$. Notice that stal and stal_n are multihomogeneous.

[The stalactic monoid was originally defined by Hivert et al. [12, § 3.7] using the defining relations $\big\{ (bvb, bbv) : b \in A, v \in A^* \big\}$; this would yield a monoid that is anti-isomorphic to stal. The definition of stal here follows Priez [11, Example 3] so as to be compatible with the construction of the taiga monoid below.]

Proposition 7.2. *Connected components of* $K(\mathsf{stal})$ *are properly contained in* $\equiv_{\mathrm{ev}}$*-classes.*

Proof. By Lemma 2.1(1), $\sim \subseteq \equiv_{\mathrm{ev}}$, so it remains to prove that equality does not hold. Since no defining relations in $\mathcal{R}_{\mathsf{stal}}$ can be applied to a word without repeated letters, a stalactic tableau consisting of a single row is represented by exactly one word over $\mathcal{A}^*$. In particular, an element of the form $\boxed{k}\cdots\boxed{n}\boxed{1}\cdots\boxed{k{-}1}$ is represented by the unique word $k\cdots n1\cdots(k{-}1)$. Thus the elements of this form are all $\sim$-related and form a $\sim^*$-class and thus a connected component of $K(\mathsf{stal})$. Thus, for $n \geq 3$, the elements $\boxed{1}\boxed{2}\boxed{3}\cdots\boxed{n}$ and $\boxed{2}\boxed{1}\boxed{3}\cdots\boxed{n}$ are $\equiv_{\mathrm{ev}}$-related but in different connected components of $K(\mathsf{stal})$. □

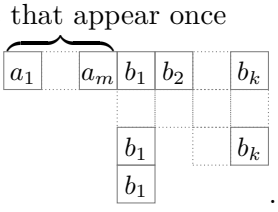
However, it is possible to characterize connected components of $K(\mathsf{stal})$. For any stalactic tableau T, let $\iota(T)$ be the word obtained by reading from left to right the symbols that appear in columns of height 1. Define $\kappa(T)$ to be the pair $\big([\iota(T)]_\sim, \mathrm{ev}(T)\big)$. (Notice that $[\iota(T)]_\sim = [\iota]_{\sim^*}$.)

Proposition 7.3. (1) *Two elements of* stal *lie in the same connected component of* $K(\mathsf{stal})$ *if and only if they have the same image under the map* κ.
(2) *The maximum diameter of a connected component of* $K(\mathsf{stal})$ *is* 3.
(3) *The maximum diameter of a connected component of* $K(\mathsf{stal}_n)$ *is* 3 *if* $n \geq 3$, *and is respectively* 1 *and* 0 *for* $n = 2$ *and* $n = 1$.

Proof. Suppose $T, U \in \mathsf{stal}$ are such that $T \sim U$. Then there exist $x, y \in \mathcal{A}^*$ such that xy represents T and yx represents U. Deleting from x and y every symbol a such that $|x|_a + |y|_a > 1$ yields two words x' and y' with $\iota(T) = x'y'$ and $\iota(U) = y'x'$; thus $\iota(T) \sim \iota(U)$. Furthermore, since $T \sim U$ it follows that $\mathrm{ev}(T) = \mathrm{ev}(U)$. Thus $\kappa(T) = \kappa(U)$. Iterating this reasoning shows that if T and U lie in the same connected component of $K(\mathsf{stal})$, then $\kappa(T) = \kappa(U)$.

Now suppose that $T, U \in \mathsf{stal}$ are such that $\kappa(T) = \kappa(U)$. Let $B = \{b_1, \ldots, b_k\}$ consist of exactly the symbols in $\mathcal{A}$ that appear more than once in U and thus in T. Choose any word t such that $\mathrm{P}_{\mathsf{stal}}(t) = T$, and delete the leftmost appearance of each symbol in B from t; call the resulting word t'. Let $t_0 = b_1 \cdots b_k t' \in \mathcal{A}^*$. Since the order of columns in a stalactic tableau corresponding to a word is determined by the rightmost appearance of each symbol in that word, and since $\mathrm{ev}(t) = \mathrm{ev}(t_0)$, it follows that $\mathrm{P}_{\mathsf{stal}}(t_0) = T$. Let $T_1 = \mathrm{P}_{\mathsf{stal}}(t' b_1 \cdots b_k)$, so that $T \sim T_1$. Then T_1 is of the form

(7.2)



Similarly, choose any word u with $\mathrm{P}_{\mathsf{stal}}(u) = U$, delete the leftmost appearance of each symbol in B to obtain a word u', and let $u_0 = b_1 \cdots b_k u'$; then $\mathrm{P}_{\mathsf{stal}}(u_0) = U$. Let $U_1 = \mathrm{P}_{\mathsf{stal}}(b_1 \cdots b_k u')$. Then $\iota(U_1) \sim \iota(U) \sim \iota(T) \sim$

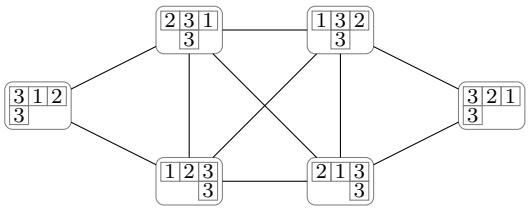
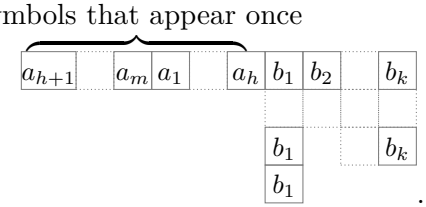


FIGURE 17. The connected component $K(\mathsf{stal}, \mathrm{P}_{\mathsf{stal}}(1233))$, which has diameter 3.

$\iota(T_1)$, and so $\iota(U_1) \sim \iota(T_1)$. Thus U_1 of the form



(7.3)

Note that (7.2) and (7.3) differ only by a cyclic permutation of the columns of height 1.

Let $s \in \mathcal{A}^*$ be such that $\mathrm{P}_{\mathsf{stal}}(a_{h+1} \cdots a_m a_1 \cdots a_h s) = U_1$. Delete the leftmost appearance of each symbol in B from the word s; call the resulting word s'. Again using the fact that the rightmost appearance of each symbol determines the order of columns, $\mathrm{P}_{\mathsf{stal}}(a_{h+1} \cdots a_m b_1 \cdots b_k a_1 \cdots a_h s') = U_1$. Similarly, $\mathrm{P}_{\mathsf{stal}}(a_1 \cdots a_h s' a_{h+1} \cdots a_m b_1 \cdots b_k) = T_1$. Hence $U_1 \sim T_1$.

Thus $T \sim T_1 \sim U_1 \sim U$, and so there is a path of length at most 3 from T to U. Hence T and U lie in the same connected component. This completes the proof of part 1).

Furthermore, this shows that connected components of $K(\mathsf{stal})$ have diameter at most 3. Direct calculation shows that the connected component $K(\mathsf{stal}, \mathrm{P}_{\mathsf{stal}}(1233))$ is as shown in Figure 17 and thus has diameter 3. This proves part 2). For part 3), note that connected components of $K(\mathsf{stal}_1)$ are singleton vertices, and the connected components of $K(\mathsf{stal}_2)$ have at most two vertices corresponding to the two possible orders of columns filled by symbols 1 and 2. □

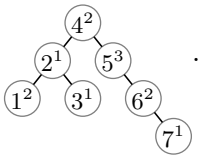
8. Taiga monoid

The taiga monoid is a quotient of the sylvester monoid that is associated with a modified notion of binary search tree. As usual, this section only recalls the essential facts; see [11, § 5] for further background.

A *binary search tree with multiplicities* is a labelled search tree in which each label appears at most once, where the label of each node is greater than the label of every node in its left subtree, and less than the label of every node in its right subtree, and where each node label is assigned a positive

integer called its *multiplicity*. An example of a binary search tree is:

(8.1)



(The superscripts on the labels in each node denote the multiplicities.)

Algorithm 8.1.

Input: A binary search tree with multiplicities T and a symbol $a \in \mathcal{A}$.

Output: A binary search tree with multiplicities $T \leftarrow a$.

Method: If T is empty, create a node, label it by a, and assign it multiplicity 1. If T is non-empty, examine the label x of the root node; if $a < x$, recursively insert a into the left subtree of the root node; if $a > x$, recursively insert a into the right subtree of the root node; if $a = x$, increment by 1 the multiplicity of the node label x.

Thus one can compute, for any word $u \in \mathcal{A}^*$, a binary search tree with multiplicities $\mathrm{P}_{\mathsf{taig}}(u)$ by starting with an empty binary search tree with multiplicities and successively inserting the symbols of u, proceeding right-to-left through the word. For example $\mathrm{P}_{\mathsf{taig}}(65117563254)$ is (8.1).

Define the relation $\equiv_{\mathsf{taig}}$ by

$$u \equiv_{\mathsf{taig}} v \iff \mathrm{P}_{\mathsf{taig}}(u) = \mathrm{P}_{\mathsf{taig}}(v),$$

for all $u, v \in \mathcal{A}^*$. The relation $\equiv_{\mathsf{taig}}$ is a congruence, and the *taiga monoid*, denoted taig, is the factor monoid $\mathcal{A}^*/\equiv_{\mathsf{taig}}$; the *taiga monoid of rank* n, denoted taig_n, is the factor monoid $\mathcal{A}_n^*/\equiv_{\mathsf{taig}}$ (with the natural restriction of $\equiv_{\mathsf{taig}}$). Each element $[u]_{\equiv_{\mathsf{taig}}}$ can be identified with the binary search tree with multiplicities $\mathrm{P}_{\mathsf{taig}}(u)$.

As with [ordinary] binary search trees, a *reading* of a binary search tree with multiplicities T is a word u such that $\mathrm{P}_{\mathsf{taig}}(u) = T$. It is easy to see that a reading of T is a word formed from the symbols that appear in the nodes of T, with the number of times each symbol appears being its multiplicity, arranged so that the rightmost symbol from a parent node appears to the right of the rightmost symbols from its children. For example, 135671456254 is a reading of (8.1).

The monoid taig is presented by $\langle \mathcal{A} \,|\, \mathcal{R}_{\mathsf{taig}} \rangle$, where $\mathcal{R}_{\mathsf{taig}} = \mathcal{R}_{\mathsf{sylv}} \cup \mathcal{R}_{\mathsf{stal}}$; the monoid taig_n is presented by $\langle \mathcal{A}_n \,|\, \mathcal{R}_{\mathsf{taig}} \rangle$, where the set of defining relations $\mathcal{R}_{\mathsf{taig}}$ is naturally restricted to $\mathcal{A}_n^* \times \mathcal{A}_n^*$. Notice that taig and taig_n are multihomogeneous.

The taiga monoid is a quotient of the sylvester monoid under the homomorphism $\tau : \mathsf{sylv} \to \mathsf{taig}$ sending $[u]_{\mathsf{sylv}}$ to $[u]_{\mathsf{taig}}$ (or equivalently, $\mathrm{P}_{\mathsf{sylv}}(u)$ to $\mathrm{P}_{\mathsf{taig}}(u)$) for all $u \in \mathcal{A}^*$. This homomorphism naturally restricts to a surjective homomorphism $\tau : \mathsf{sylv}_n \to \mathsf{taig}_n$. This connection between sylv and taig makes it possible to use the reasoning about the diameters of connected components in $K(\mathsf{sylv}_n)$ to prove the corresponding results for $K(\mathsf{taig}_n)$.

Since defining relations in $\mathcal{R}_{\mathsf{stal}}$ involve repeated symbols, only relations in $\mathcal{R}_{\mathsf{sylv}}$ apply to standard words. That is, standard words are related by $\equiv_{\mathsf{taig}}$ if and only if they are related by $\equiv_{\mathsf{sylv}}$. Thus the proof of Lemma 6.3 applies in taig_n to establish the following result:

Lemma 8.2. *There is a connected component in $K(\mathsf{taig}_n)$ of diameter at least $n-1$.*

Lemma 8.3. *Every connected component of $K(\mathsf{taig}_n)$ has diameter at most n.*

Proof. Define a map $\psi : \mathsf{taig}_n \to \mathsf{sylv}_n$ that maps a binary search tree with multiplicities T to the standard binary search tree obtained by deleting the multiplicities of T. Note that, two elements $T, U \in \mathsf{taig}_n$ are equal if and only if $\psi(T) = \psi(U)$ and $\mathrm{ev}(T) = \mathrm{ev}(U)$.

Suppose $P, Q \in \mathsf{taig}_n$ are such that $\psi(P) \sim \psi(Q)$ (in sylv_n) and $\mathrm{ev}(P) = \mathrm{ev}(Q)$. Then there are readings xy of $\psi(P)$ and yx of $\psi(Q)$. Replacing each symbol a with a^{k_a} for each $a \in \mathcal{A}_n$, where k_a is the a-th component of $\mathrm{ev}(P) = \mathrm{ev}(Q)$, gives readings $\hat{x}\hat{y}$ of P and $\hat{y}\hat{x}$ of Q, so that $P \sim Q$ (in taig_n).

Let T and U be elements of the same connected component of $K(\mathsf{taig}_n)$. Then $T \equiv_{\mathrm{ev}} U$. By the strategy for bulding a path in $K(\mathsf{sylv}_n)$ in the proof of Proposition 6.12, there is a path $\psi(T) = T_0, \ldots, T_n = \psi(U)$ in $K(\mathsf{sylv}_n)$. By the reasoning in the previous paragraph, this path lifts to a path $T = \hat{T}_0, \ldots, \hat{T}_n = U$ in $K(\mathsf{taig}_n)$. Thus the diameter of $K(\mathsf{taig}_n, T)$ is at most n. □

Combining Lemmata 8.2 and 8.3 gives the result:

Theorem 8.4. (1) *Connected components of $K(\mathsf{taig})$ coincide with $\equiv_{\mathrm{ev}}$-classes in taig.*
(2) *The maximum diameter of a connected component of $K(\mathsf{taig}_n)$ is $n-1$ or n.*

9. Baxter monoid

The Baxter monoid is a monoid of pairs of twin binary search trees. As in previous sections, only the essential facts are given here; see [14] for further background.

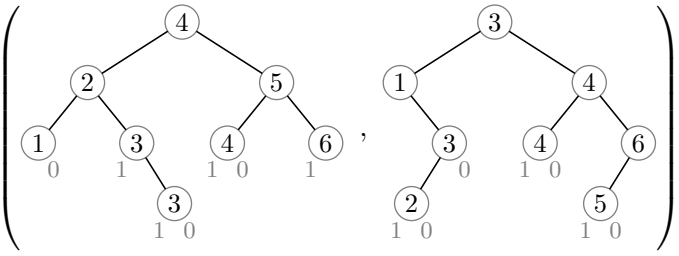
A *left strict binary search tree* is a labelled rooted binary tree where the label of each node is strictly greater than the label of every node in its left subtree, and less than or equal to every node in its right subtree; see the left tree shown in (9.1) below for an example.

The *canopy* of a binary tree T is the word over $\{0, 1\}$ obtained by traversing the empty subtrees of the nodes of T from left to right, except the first and the last, labelling an empty left subtree by 1 and an empty right subtree by 0. (See (9.1) below for examples of canopies.)

A *pair of twin binary search trees* consist of a left strict binary search tree T_L and a right strict binary search tree T_R, such that T_L and T_R contains the same symbols, and the canopies of T_L and T_R are complementary, in the sense that the i-th symbol of the canopy of T_L is 0 (respectively 1) if and only if the i-th symbol of the canopy of T_L is 1 (respectively 0). The following is an example of a pair of twin binary search trees, with the complementary

canopies 0110101 and 1001010 shown in grey:

(9.1)



The insertion algorithm for left strict binary search trees is symmetric to Algorithm 6.1:

Algorithm 9.1 (Left strict leaf insertion)**.**

Input: A left strict binary search tree T and a symbol $a \in \mathcal{A}$.

Output: A left strict binary search tree $a \to T$.

Method: If T is empty, create a node and label it a. If T is non-empty, examine the label x of the root node; if $a < x$, recursively insert a into the left subtree of the root node; otherwise recursively insert a into the right subtree of the root node. Output the resulting tree.

Thus one can compute, for any word $u \in \mathcal{A}^*$, a pair of twin binary search trees $\mathrm{P}_{\mathsf{baxt}}(u) = (T_L, T_R)$, where T_R is $\mathrm{P}_{\mathsf{sylv}}(u)$ and T_L is obtained by starting with an empty left strcit binary search tree and successively inserting the symbols of u, proceeding left-to-right through the word. For example, $\mathrm{P}_{\mathsf{baxt}}(42531643)$ is (9.1).

A *reading* of a pair of twin binary search trees (T_L, T_R) is a word u such that $\mathrm{P}_{\mathsf{baxt}}(u) = (T_L, T_R)$. It is easy to see that a reading of (T_L, T_R) is a word formed from the symbols appearing in the two binary trees T_L and T_R (which, by definition, contain the same symbols), ordered so that every symbol from a parent node in T_L appears to the left of those from its children in T_L, and every symbol from parent node in T_R appears to the right of those from its children.

Define the relation $\equiv_{\mathsf{baxt}}$ by

$$u \equiv_{\mathsf{baxt}} v \iff \mathrm{P}_{\mathsf{baxt}}(u) = \mathrm{P}_{\mathsf{baxt}}(v),$$

for all $u, v \in \mathcal{A}^*$. The relation $\equiv_{\mathsf{baxt}}$ is a congruence, and the *Baxter monoid*, denoted baxt, is the factor monoid $\mathcal{A}^*/\equiv_{\mathsf{baxt}}$; the *Baxter monoid of rank* n, denoted baxt_n, is the factor monoid $\mathcal{A}_n^*/\equiv_{\mathsf{baxt}}$ (with the natural restriction of $\equiv_{\mathsf{baxt}}$). Each element $[u]_{\equiv_{\mathsf{baxt}}}$ (where $u \in \mathcal{A}^*$) can be identified with the pair of twin binary search trees $\mathrm{P}_{\mathsf{baxt}}(u)$. The words in $[u]_{\equiv_{\mathsf{baxt}}}$ are precisely the readings of $\mathrm{P}_{\mathsf{baxt}}(u)$.

The monoid baxt is presented by $\langle \mathcal{A} \,|\, \mathcal{R}_{\mathsf{baxt}} \rangle$, where

$$\begin{aligned}\mathcal{R}_{\mathsf{baxt}} = {} & \{\, (cudavb, cuadvb) : a \le b < c \le d,\ u, v \in A^* \,\} \\ & \cup \{\, (budavc, buadvc) : a < b \le c < d,\ u, v \in A^* \,\};\end{aligned}$$

see [14, Definition 3.1]. The monoid baxt_n is presented by $\langle \mathcal{A}_n \,|\, \mathcal{R}_{\mathsf{baxt}} \rangle$, where the set of defining relations $\mathcal{R}_{\mathsf{baxt}}$ is naturally restricted to $\mathcal{A}_n^* \times \mathcal{A}_n^*$. Note that baxt and baxt_n are multihomogeneous.

There is a straightforward method for extracting every possible reading from a pair of binary search trees (T_L, T_R):

Method 9.2. *Input:* A pair of twin binary search trees (T_L, T_R).
Output: A reading of (T_L, T_R).

(1) Set (U_L, U_R) to be (T_L, T_R). (Throughout this computation, U_L is a forest of left strict binary search trees and U_R is a right strict binary search tree.)
(2) If U_L and U_R are empty, halt.
(3) Given some (U_L, U_R), choose and output some symbol a that labels a root of some tree in the forest U_L and a leaf of the tree U_R.
(4) Deleting the corresponding root vertex of U_L and the corresponding leaf vertex of U_L.

This is essentially [14, Algorithm on p.133], except that the method given here is non-deterministic in that there may be several choices for a in step 3. As these choices vary, all possible readings of (T_L, T_R) are obtained.

Proposition 9.3. *Connected components of $K(\mathsf{baxt})$ are properly contained in $\equiv_{\mathrm{ev}}$-classes.*

Proof. Connected components of $K(\mathsf{baxt})$ are $\sim^*$-classes, and by Lemma 2.1, $\sim^* \subseteq \equiv_{\mathrm{ev}}$. It thus remains to prove that equality does not hold. Since all the defining relations in $\mathcal{R}_{\mathsf{baxt}}$ have length at least 4, it follows that none of these relations can be applied to words of length 3. Thus all length-3 words represent distinct elements of baxt. Therefore the words in $\{123, 231, 312\}$ represent all the elements in one $\sim^*$-class in baxt (and thus one connected component of $K(\mathsf{baxt})$. The word 132 is not in this set, but is in the same $\equiv_{\mathrm{ev}}$-class. This completes the proof. □

A natural question at this point is whether $\sim^*$ and $\equiv_{\mathrm{ev}}$ do not coincide in baxt only for the slightly trivial reason that relations in $\mathcal{R}_{\mathsf{baxt}}$ do not apply to words of length 3, and that perhaps $\sim^*$ and $\equiv_{\mathrm{ev}}$ coincide for elements represented by words of length 4 or more. However, consider the elements of baxt represented by the words 1243, 2431, 4312, and 3124:

$$\mathrm{P}_{\mathsf{baxt}}(1243) = \left(\begin{array}{c} 1 \\ \\ 2 \\ \\ 4 \\ \\ 3 \end{array} \;,\; \begin{array}{ccc} & 3 & \\ 2 & & 4 \\ 1 & & \end{array} \right)$$

$$\mathrm{P}_{\mathsf{baxt}}(2431) = \left(\begin{array}{ccc} & 2 & \\ 1 & & 4 \\ & 3 & \end{array} \;,\; \begin{array}{ccc} 1 & & \\ & 3 & \\ 2 & & 4 \end{array} \right)$$

$$\mathrm{P}_{\mathsf{baxt}}(4312) = \left(\begin{array}{ccc} & & 4 \\ & 3 & \\ 1 & & \\ & 2 & \end{array} \;,\; \begin{array}{ccc} & 2 & \\ 1 & & 3 \\ & & 4 \end{array} \right)$$

$$\mathrm{P}_{\mathsf{baxt}}(3124) = \left(\begin{array}{ccc} & 3 & \\ 1 & & 4 \\ & 2 & \end{array} \;,\; \begin{array}{ccc} & 4 & \\ 2 & & \\ 1 & & 3 \end{array} \right)$$

It is straightforward to prove that there is exactly one reading of each of these pairs of twin binary search trees: for example, consider extracting a reading from $\mathrm{P}_{\mathsf{baxt}}(2431)$. Following Method 9.2, (U_L, U_R) is initially $\mathrm{P}_{\mathsf{baxt}}(2431)$. The first output symbol must be 2, since this is the unique root in U_L. Deleting the corresponding vertices yields

$$(U_L, U_R) = \left(\begin{array}{ccc} & & \\ 1 & & 4 \\ & 3 & \end{array} \;,\; \begin{array}{ccc} & 1 & \\ & & \\ & 3 & \\ & & 4 \end{array} \right)$$

From this point onwards, there will be exactly one leaf vertex in U_R, and so only one choice for the symbol to output. Hence the method must output 4, 3, 1. Hence the unique reading of $\mathrm{P}_{\mathsf{baxt}}(2431)$ is 2431.

Hence the element represented by the words in $\{1243, 2431, 4312, 3124\}$ form a single $\sim^*$-class and so (for example) 1234 and 1243 are not related by $\sim^*$.

Question 9.4. (1) Is there a characterization of $\sim^*$-classes in baxt?
(2) Is there a bound on the diameter of $\sim^*$-classes in baxt_n?

10. Questions

Question 10.1. For each monoid $\mathsf{M} \in \{\mathsf{plac}, \mathsf{hypo}, \mathsf{sylv}, \mathsf{stal}, \mathsf{taig}, \mathsf{baxt}\}$, is there an efficient algorithm that takes two elements $T, U \in \mathsf{M}$ such that $T \equiv_{\mathrm{ev}} U$ and computes the distance between them in $K(\mathsf{M})$?

With regard to the previous question, note that it is always possible to compute the distance via a brute-force computation: one could build the entire connected component $K(\mathsf{M}, T)$, then find the shortest path from T to U. The question is whether this can be done *efficiently*. Note that the strategies for constructing paths in the various proofs in this paper do *not* in general find shortest paths between two elements; see Example 5.8.

11. Appendix: Conjugacy

In a group, the relation $\sim$ is simply the usual notion of conjugacy. The concept of cyclic shifts can thus be viewed in an algebraic way as a generalization to monoids of the concept of conjugacy in groups. Another possible generalization, introduced by Otto [22], is o-conjugacy, defined by

$$x \sim_o y \iff (\exists g, h \in M)(xg = gy \wedge hx = yh). \tag{11.1}$$

The relation $\sim_o$ is an equivalence relation. The following result describes how $\sim_o$ is related to $\sim$ and $\equiv_{\mathrm{ev}}$:

Proposition 11.1. (1) *In any monoid,* $\sim^* \subseteq \sim_o$.
(2) *In any multihomogeneous monoid* $\sim_o \subseteq \equiv_{\mathrm{ev}}$.

Proof. For the first part, see [23, § 1]. For the second part, see [15, Lemma 3.2]. □

Thus in a multihomogeneous monoid, $\sim^* \subseteq \sim_o \subseteq \equiv_{\mathrm{ev}}$. Since $\sim^* = \equiv_{\mathrm{ev}}$ in the plactic, hypoplactic, sylvester, and taiga monoids, in these settings $\sim_o$ coincides with $\sim^*$ and $\equiv_{\mathrm{ev}}$ and thus is not of independent interest. However, it turns out that $\sim_o$ and $\equiv_{\mathrm{ev}}$ coincide in the stalactic and Baxter

monoids. (Recall that $\sim^*$ is strictly contained in $\equiv_{\mathrm{ev}}$ in both these monoids; see Propositions 7.2 and 9.3.)

Proposition 11.2. *In* stal*, the relations* $\sim_o$ *and* $\equiv_{\mathrm{ev}}$ *coincide.*

Proof. Let $u, v \in \mathcal{A}^*$ be such that $u \equiv_{\mathrm{ev}} v$. In particular, $\mathrm{P}_{\mathsf{stal}}(u)$ and $\mathrm{P}_{\mathsf{stal}}(v)$ both have m columns, for some $m \in \mathbb{N}$. For $i \in \{1, \ldots, m\}$, let $a_i \in A$ be the symbol that appears in the i-th column of $\mathrm{P}_{\mathsf{stal}}(u)$ and let $b_i \in A$ be the symbol that appears in the i-th column of $\mathrm{P}_{\mathsf{stal}}(v)$. Let $g = a_1 \cdots a_m$ and $h = b_1 \cdots b_m$. Notice that every symbol that appears in u and v appears exactly once in g and h. Hence $gu \equiv_{\mathrm{ev}} vg$ and $uh \equiv_{\mathrm{ev}} hv$. Furthermore, the order of rightmost appearances of symbols in gu and vg is identical; together with $gu \equiv_{\mathrm{ev}} vg$, this implies that $\mathrm{P}_{\mathsf{stal}}(gu) = \mathrm{P}_{\mathsf{stal}}(vg)$. Thus $gu \equiv_{\mathsf{stal}} vg$. Similarly, $uh \equiv_{\mathsf{stal}} hv$. Hence $u \sim_o v$. This proves that $\equiv_{\mathrm{ev}} \subseteq \sim_o$. The opposite inclusion follows from Proposition 11.1(2). □

Proposition 11.3. *In* baxt*, the relations* $\sim_o$ *and* $\equiv_{\mathrm{ev}}$ *coincide.*

Proof. Let $p, q \in \mathcal{A}^*$ be such that $p \equiv_{\mathrm{ev}} q$. By [24, Proposition 3.8], $ppq \equiv_{\mathsf{baxt}} pqq$ and $qpp \equiv_{\mathsf{baxt}} qqp$. Hence $pg \equiv_{\mathsf{baxt}} gq$ with $g = pq$, and $hp \equiv_{\mathsf{baxt}} qh$ with $h = qp$. Thus $p \sim_o q$. This proves that $\equiv_{\mathrm{ev}} \subseteq \sim_o$. The opposite inclusion follows from Proposition 11.1(2). □

References

[1] A. Lascoux, M.-P. Schützenberger, Le monoïde plaxique, in: Noncommutative structures in algebra and geometric combinatorics, no. 109 in Quaderni de "La Ricerca Scientifica", CNR, Rome, 1981, pp. 129–156.
URL http://igm.univ-mlv.fr/~berstel/Mps/Travaux/A/1981-1PlaxiqueNaples.pdf

[2] C. Choffrut, R. Mercaş, The lexicographic cross-section of the plactic monoid is regular, in: J. Karhumäki, A. Lepistö, L. Zamboni (Eds.), Combinatorics on Words, no. 8079 in Lecture Notes in Computer Science, Springer, 2013, pp. 83–94. doi:10.1007/978-3-642-40579-2_11.

[3] I. G. Macdonald, Symmetric Functions and Hall Polynomials, Clarendon Press, Oxford University Press, 2008.

[4] W. Fulton, Young Tableaux: With Applications to Representation Theory and Geometry, no. 35 in London Mathematical Society Student Texts, Cambridge University Press, 1997.

[5] M. Lothaire, Algebraic Combinatorics on Words, no. 90 in Encyclopedia of Mathematics and its Applications, Cambridge University Press, 2002.

[6] F. Jedrzejewski, Plactic classification of modes, in: C. Agon, M. Andreatta, G. Assayag, E. Amiot, J. Bresson, J. Mandereau (Eds.), Mathematics and Computation in Music, no. 6726 in Lecture Notes in Computer Science, Springer, 2011, pp. 350–353. doi:10.1007/978-3-642-21590-2_31.

[7] D. Krob, J.-Y. Thibon, Noncommutative Symmetric Functions IV: Quantum Linear Groups and Hecke Algebras at $q = 0$, Journal of Algebraic Combinatorics 6 (4) (1997) 339–376. doi:10.1023/A:1008673127310.

[8] D. Krob, J.-Y. Thibon, Noncommutative Symmetric Functions V: A Degenerate Version of $U_q(gl_N)$, International Journal of Algebra and Computation 9 (3–4) (1999) 405–430. doi:10.1142/S0218196799000254.

[9] J.-C. Novelli, On the hypoplactic monoid, Discrete Mathematics 217 (1–3) (2000) 315–336. doi:10.1016/S0012-365X(99)00270-8.

[10] F. Hivert, J.-C. Novelli, J.-Y. Thibon, The algebra of binary search trees, Theoretical Computer Science 339 (1) (2005) 129–165. doi:10.1016/j.tcs.2005.01.012.

[11] J.-B. Priez, A lattice of combinatorial Hopf algebras: Binary trees with multiplicities, in: Formal Power Series and Algebraic Combinatorics, The Association. Discrete Mathematics & Theoretical Computer Science, Nancy, 2013.
URL http://www.dmtcs.org/pdfpapers/dmAS0196.pdf

[12] F. Hivert, J.-C. Novelli, J.-Y. Thibon, Commutative combinatorial Hopf algebras, Journal of Algebraic Combinatorics 28 (1) (2007) 65–95. doi:10.1007/s10801-007-0077-0.

[13] S. Giraudo, Algebraic and combinatorial structures on Baxter permutations, in: 23rd International Conference on Formal Power Series and Algebraic Combinatorics, The Association. Discrete Mathematics & Theoretical Computer Science, Nancy, 2011, pp. 387–398.
URL https://www.dmtcs.org/dmtcs-ojs/index.php/proceedings/article/view/dmAO0135.1.html

[14] S. Giraudo, Algebraic and combinatorial structures on pairs of twin binary trees, Journal of Algebra 360 (2012) 115–157. doi:10.1016/j.jalgebra.2012.03.020.

[15] A. J. Cain, A. Malheiro, Deciding conjugacy in sylvester monoids and other homogeneous monoids, International Journal of Algebra and Computation 25 (5) (2015) 899–915. doi:10.1142/S0218196715500241.

[16] W. S. Stein, et al., Sage Mathematics Software (Version x.y.z), The Sage Development Team (2016).
URL http://www.sagemath.org

[17] A. J. Cain, A. Malheiro, Combinatorics of cyclic shifts in plactic, hypoplactic, sylvester, and related monoids, in: S. Brlek, F. Dolce, C. Reutenauer, É. Vandomme (Eds.), Combinatorics on Words: 11th International Conference, WORDS 2017, Montréal, Canada, September 11–15, 2017, no. 10432 in Lecture Notes in Computer Science, Springer, 2017, pp. 190–202. doi:10.1007/978-3-319-66396-8_18.

[18] J. M. Howie, Fundamentals of Semigroup Theory, no. 12 in London Mathematical Society Monographs: New Series, Clarendon Press, Oxford University Press, New York, 1995.

[19] P. M. Higgins, Techniques of Semigroup Theory, Oxford University Press, 1991.

[20] N. Ruškuc, Semigroup Presentations, Ph.d. thesis, University of St Andrews (1995).
URL http://hdl.handle.net/10023/2821

[21] F. Harary, Graph Theory, Addison–Wesley, Reading, MA, 1969.

[22] F. Otto, Conjugacy in monoids with a special Church–Rosser presentation is decidable, Semigroup Forum 29 (1) (1984) 223–240. doi:10.1007/BF02573327.

[23] J. Araújo, J. Konieczny, A. Malheiro, Conjugation in semigroups, Journal of Algebra 403 (2014) 93–134. doi:10.1016/j.jalgebra.2013.12.025.

[24] A. J. Cain, A. Malheiro, Identities in plactic, hypoplactic, sylvester, Baxter, and related monoids, Electronic Journal of Combinatorics 25 (3). arXiv:1611.04151.
URL http://www.combinatorics.org/ojs/index.php/eljc/article/view/v25i3p30

Centro de Matemática e Aplicações (CMA), Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2829–516 Caparica, Portugal
Email address: a.cain@fct.unl.pt

Centro de Matemática e Aplicações (CMA), Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2829–516 Caparica, Portugal

Departamento de Matemática, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2829–516 Caparica, Portugal
Email address: ajm@fct.unl.pt