Identities in plactic, hypoplactic, sylvester, Baxter, and related monoids

Alan J. Cain*†
Centro de Matemática e Aplicações
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
2829–516 Caparica, Portugal
a.cain@fct.unl.pt

António Malheiro†
Departamento de Matemática & Centro de Matemática e Aplicações
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
2829–516 Caparica, Portugal
ajm@fct.unl.pt

Submitted: Apr 6, 2017; Accepted: Jul 28, 2018; Published: Aug 24, 2018

Abstract

This paper considers whether non-trivial identities are satisfied by certain 'plactic-like' monoids that, like the plactic monoid, are closely connected to combinatorics. New results show that the hypoplactic, sylvester, Baxter, stalactic, and taiga monoids satisfy identities, and indeed give shortest identities satisfied by these monoids. The existing state of knowledge is discussed for the plactic monoid and left and right patience sorting monoids.

Mathematics Subject Classifications: 05E99, 20M05

1 Introduction

The ubiquitous plactic monoid, whose elements can be viewed as semistandard Young tableaux, and which appears in such diverse contexts as symmetric functions [Mac08],

*Supported by an Investigador FCT fellowship (IF/01622/2013/CP1161/CT0001).
†This work was partially supported by the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through the project UID/MAT/00297/2013 (Centro de Matemática e Aplicações), the project PTDC/MHC-FIL/2583/2014, and the project PTDC/MAT-PUR/31174/2017.

Table 1: Examples of non-trivial identities satisfied by 'plactic-like' monoids. The stated identities are always shortest non-trivial identities satisfied by the corresponding monoid, but there may be other identities of the same length.

Monoid	*Symbol*	*Identity satisfied*	*Discussed in*	*Original result*
Plactic	plac	None	Subsec. 3.7	[CKK+17]
Hypoplactic	hypo	$xyxy = yxyx$	Subsec. 3.2	Present paper
Sylvester	sylv	$xyxy = yxxy$	Subsec. 3.4	Present paper
#-sylvester	sylv$^{\#}$	$yxyx = yxxy$	Subsec. 3.4	Present paper
Baxter	baxt	$yxxyxy = yxyxxy$	Subsec. 3.5	Present paper
Stalactic	stal	$xyx = yxx$	Subsec. 3.3	Present paper
Taiga	taig	$xyx = yxx$	Subsec. 3.3	Present paper
Left patience sorting	ℓPS	None	Subsec. 3.6	[CMS]
Right patience sorting	rPS	None	Subsec. 3.6	[CMS]

representation theory [Ful97], algebraic combinatorics [Lot02], Kostka–Foulkes polynomials [LS81, LS78], Schubert polynomials [LS85, LS90], and musical theory [Jed11], is one of a family of 'plactic-like' monoids that are closely connected with combinatorics. These monoids include the hypoplactic monoid [KT97, Nov00], the sylvester monoid [HNT05], the taiga monoid [Pri13], the stalactic monoid [HNT07, Pri13], the Baxter monoid [Gir12], and the left and right patience sorting monoids [Rey07, CMS]. Each of these monoids is obtained by factoring the free monoid $\mathcal{A}^*$ over the infinite ordered alphabet $\mathcal{A} = \{1 < 2 < 3 < \cdots\}$ by a congruence that arises from a so-called insertion algorithm that computes a combinatorial object from a word. For instance, for the plactic monoid, the corresponding combinatorial objects are (semistandard) Young tableaux; for the sylvester monoid, they are binary search trees.

An *identity* is a formal equality between two words in the free monoid, and is *non-trivial* if the two words are distinct. A monoid M *satisfies* such an identity if the equality in M holds under every substitution of letters in the words by elements of M. For example, any commutative monoid satisfies the non-trivial identity $xy = yx$. A finitely generated group has polynomial growth if and only if it is virtually nilpotent [Gro81], and a virtually nilpotent group satisfies a non-trivial identity [Mal53, NT63]. Thus it is natural to ask whether every finitely generated semigroup or monoid with polynomial growth satisfies a non-trivial identity. Schneerson [Shn93] provided the first counterexample, but it remains to be seen whether there is a 'natural' finitely generated semigroup with polynomial growth that does not satisfy a non-trivial identity. It is easy to see that the finite-rank analogues of the plactic monoid and the other related monoids discussed above have polynomial growth. This naturally leads to the question of whether these monoids satisfy non-trivial identities, for if any of them failed to do so, it would certainly be a very natural example of a polynomial-growth monoid that does not satisfy a non-trivial identity.

Further motivation for this question comes from a result of Jaszuńska & Okniński, who proved that the Chinese monoid [CEK+01], which has the same growth type as the

plactic monoid [DK94] but which does not arise from such a natural combinatorial object, satisfies Adian's identity $xyyxxyxyyx = xyyxyxxyyx$ [JO11, Corollary 3.3.4]. (This is the shortest non-trivial identity satisfied by the bicyclic monoid [Adi66, Chapter IV, Theorem 2]).

The goal of this paper is to present new results showing that some of these monoids satisfy non-trivial identities, and to survey the state of knowledge for other monoids in this family. New results show that the hypoplactic, sylvester, baxter, stalactic, and taiga monoids satisfy non-trivial identities. A discussion of the situation for the left and right patience sorting monoids and plactic monoids completes the paper. Table 1 summarizes the results.

2 'Plactic-like' monoids

In this section, we recall only the definition and essential facts about the various monoids; for further background, see [Lot02, Ch. 5] on the plactic monoid, [Nov00] on the hypoplactic monoid, [HNT05] on the sylvester monoid; [Pri13, § 5] on the taiga monoid; [Pri13] on the stalactic monoid; [Gir12] on the Baxter monoid; and [CMS] on the patience sorting monoids.

2.1 Alphabets and words

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 $|u|_x$.

Throughout the paper, $\mathcal{A} = \{1 < 2 < 3 < \cdots\}$ is the set of natural numbers viewed as an infinite ordered alphabet, and $\mathcal{A}_n = \{1 < 2 < \cdots < n\}$ is set of the first n natural numbers viewed as a finite ordered alphabet.

2.2 Combinatorial objects and insertion algorithms

A *Young 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, with shorter rows below longer ones, and with rows left-justified. An example of a Young tableau is

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

The following algorithm takes a Young tableau and a symbol from $\mathcal{A}$ and yields a new Young tableau:

Algorithm 1 (Schensted's algorithm)**.** *Input:* A Young tableau T and a symbol $a \in \mathcal{A}$.

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.)

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:

$$\begin{array}{|c|c|c|c|c|c|c|c|}\hline 1 & 1 & 1 & 2 & & & & \\ \hline & & & 3 & 3 & 5 & 5 & \\ \hline & & & & & & 6 & 6 \\ \hline \end{array} \quad . \tag{2}$$

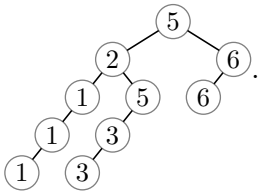
Notice that the same symbol cannot appear in two different rows of a quasi-ribbon tableau. There is also an insertion algorithm for quasi-ribon tableau:

Algorithm 2 ([Nov00, Algorithm 4.4])**.** *Input:* A quasi-ribbon tableau T and a symbol $a \in \mathcal{A}$.

If there is no entry in T that is less than or equal to a, output the tableau obtained by putting a and gluing T by its top-leftmost entry to the bottom of a.

Otherwise, let x be the right-most and bottom-most entry of T that is less than or equal to a. Put a new entry a to the right of x and glue the remaining part of T (below and to the right of x) onto the bottom of the new entry a. Output the new tableau.

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



. (3)

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

Algorithm 3 ([HNT05, § 3.3])**.** *Input:* A right strict binary search tree T and a symbol $a \in \mathcal{A}$.

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 right subtree of the root node; otherwise recursively insert a into the left subtree of the root note. Output the resulting tree.

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 (4) below for an example.

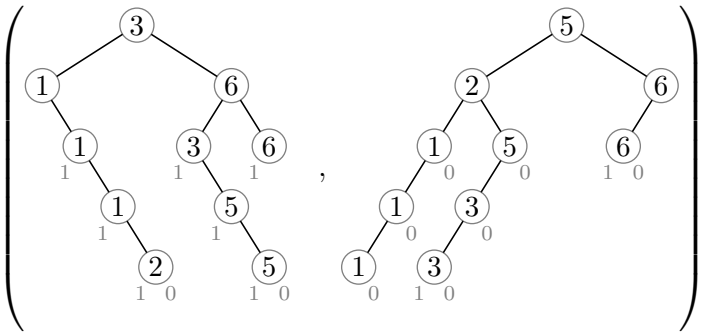
The insertion algorithm for left strict binary search trees is dual to Algorithm 3 above:

Algorithm 4. *Input:* A left strict binary search tree T and a symbol $a \in \mathcal{A}$.

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 note. Output the resulting tree.

The *canopy* of a (labelled or unlabelled) 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 (4) 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 contain 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_R 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:



(4)

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

3	1	2	6	5
3	1		6	5
	1			

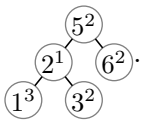
(5)

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

Algorithm 5 ([HNT07, § 3.7])**.** *Input:* A stalactic tableau T and a symbol $a \in \mathcal{A}$.

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.

A *binary search tree with multiplicities* is a labelled binary search tree in which each label appears at most once, and where a non-negative integer called the *multiplicity* is assigned to each node label. An example of a binary search tree is:



(6)

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

Algorithm 6. ([Pri13, Algorithm 3]) *Input:* A binary search tree with multiplicities T and a symbol $a \in \mathcal{A}$.

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 note; if $a = x$, increment by 1 the multiplicity of the node label x.

An *ℓPS tableau* is a finite array of symbols of $\mathcal{A}$ in which columns are top-aligned, the entries in the top row are non-decreasing from left to right, and the entries in each column are strictly increasing from top to bottom. For example,

$$\begin{array}{|c|c|c|c|c|}\hline 1 & 1 & 1 & 2 & 5 \\ \hline 3 & 3 & 5 & & 6 \\ \cline{1-3}\cline{5-5} & 6 & \\ \cline{2-2}\end{array} \tag{7}$$

is an ℓPS tableau. The insertion algorithm is very straightforward:

Algorithm 7 ([TY11, § 3.2])**.** *Input:* An ℓPS tableau T and a symbol $a \in \mathcal{A}$.

If a is greater than or equal to every symbol that appears in the top row of T, add a to the right of the top row of T. Otherwise, let C be the leftmost column whose topmost symbol is strictly greater than a. Slide column C down by one space and add a as a new entry on top of C. Output the new tableau.

An *rPS tableau* is a finite array of symbols of $\mathcal{A}$ in which columns are top-aligned, the entries in the top row are increasing from left to right, and the entries in each column are non-decreasing from top to bottom. For example,

$$\begin{array}{|c|c|c|c|}\hline 1 & 2 & 5 & 6 \\ \hline 1 & 3 & 5 \\ \cline{1-3} 1 & 6 \\ \cline{1-2} 3 \\ \cline{1-1}\end{array} \tag{8}$$

is an rPS tableau. Again, the insertion algorithm is very straightforward:

Algorithm 8 ([TY11, § 3.2])**.** *Input:* An rPS tableau T and a symbol $a \in \mathcal{A}$.

If a is greater than every symbol that appears in the top row of T, add a to the right of the top row of T. Otherwise, let C be the leftmost column whose topmost symbol is greater than or equal to a. Slide column C down by one space and add a as a new entry on top of C. Output the new tableau.

2.3 Monoids from insertion

Let $\mathsf{M} \in \{\mathsf{plac}, \mathsf{hypo}, \mathsf{sylv}, \mathsf{sylv}^{\#}, \mathsf{baxt}, \mathsf{stal}, \mathsf{taig}, \ell\mathsf{PS}, r\mathsf{PS}\}$ and $u \in \mathcal{A}^*$. Using the insertion algorithms described above, one can compute from u a combinatorial object $\mathrm{P}_{\mathsf{M}}(u)$ of the type associated to M in Table 2. If $\mathsf{M} \neq \mathsf{baxt}$, one computes $\mathrm{P}_{\mathsf{M}}(u)$ by starting

Table 2: Insertion algorithms used to compute combinatorial objects, and the corresponding monoids. 'L-to-R' and 'R-to-L' abbreviate 'left-to-right' and 'right-to-left', respectively. The algorithm used to compute $\mathrm{P}_{\mathsf{baxt}}(\cdot)$ is a special case and is discussed in the main text. The example for each monoid M is $\mathrm{P}_{\mathsf{M}}(3613151265)$.

Monoid	*Symbol*	*Combinatorial object*	*Algorithm*	*Direction*	*Ex.*
Plactic	plac	Young tableau	1	L-to-R	(1)
Hypoplactic	hypo	Quasi-ribbon tableau	2	L-to-R	(2)
Sylvester	sylv	Right strict binary search tree	3	R-to-L	(3)
#-sylvester	$\mathsf{sylv}^{\#}$	Left strict binary search tree	4	L-to-R	(3)
Baxter	baxt	Pair of twin BSTs	—	—	(4)
Stalactic	stal	Stalactic tableau	5	R-to-L	(5)
Taiga	taig	BST with multiplicities	6	R-to-L	(6)
Left patience sorting	ℓPS	ℓPS tableau	7	L-to-R	(7)
Right patience sorting	rPS	rPS tableau	8	L-to-R	(8)

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 as shown in the table. For $\mathsf{M} = \mathsf{baxt}$, one uses a slightly different procedure: $\mathrm{P}_{\mathsf{baxt}}(u)$ is the pair of twin binary search trees $\big(\mathrm{P}_{\mathsf{sylv}^{\#}}(u), \mathrm{P}_{\mathsf{sylv}}(u)\big)$.

For each $\mathsf{M} \in \{\mathsf{plac}, \mathsf{hypo}, \mathsf{sylv}, \mathsf{sylv}^{\#}, \mathsf{baxt}, \mathsf{stal}, \mathsf{taig}, \ell\mathsf{PS}, r\mathsf{PS}\}$, define the relation $\equiv_{\mathsf{M}}$ by

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

In each case, the relation $\equiv_{\mathsf{M}}$ is a congruence on $\mathcal{A}^*$, and so the factor monoid $\mathsf{M} = \mathcal{A}^*/\equiv_{\mathsf{M}}$ can be formed, and is named as in Table 2. The rank-n analogue is the factor monoid $\mathsf{M}_n = \mathcal{A}_n^*/\equiv_{\mathsf{M}}$, where the relation $\equiv_{\mathsf{M}}$ is naturally restricted to $\mathcal{A}_n^* \times \mathcal{A}_n^*$.

It follows from the definition of $\equiv_{\mathsf{M}}$ for any $(\mathsf{M} \in \big\{\mathsf{plac}, \mathsf{hypo}, \mathsf{sylv}, \mathsf{sylv}^{\#}, \mathsf{baxt}, \mathsf{stal}, \mathsf{taig}, \ell\mathsf{PS}, r\mathsf{PS}\big\})$ that each element $[u]_{\equiv_{\mathsf{M}}}$ of the factor monoid M can be identified with the combinatorial object $\mathrm{P}_{\mathrm{M}}(u)$.

The *evaluation* (also called the *content*) of a word $u \in \mathcal{A}^*$, denoted $\mathrm{ev}(u)$, is the infinite tuple of non-negative integers, indexed by $\mathcal{A}$, whose a-th element is $|u|_a$; thus this tuple describes the number of each symbol in $\mathcal{A}$ that appears in u. It is immediate from the definition of the monoids above that if $u \equiv_{\mathsf{M}} v$, then $\mathrm{ev}(u) = \mathrm{ev}(v)$, and hence it makes sense to define the evaluation of an element p of one of these monoids to be the evaluation of any word representing it.

Note that M_n is a submonoid of M for all n, and that M_m is a submonoid of M_n for all $m \leqslant n$. In each case, M_1 is a free monogenic monoid and thus commutative, but that M_n is non-commutative for $n \geqslant 2$ (and thus M is also non-commutative).

3 Identities

Subsections 3.2 to 3.5 find shortest non-trivial identities satisfied by hypo, stal, taig, sylv,

sylv$^{\#}$, and baxt, in that order. Subsection 3.6 discusses the situation for ℓPS and rPS, and Subsection 3.7 discusses the current state of knowledge for plac.

3.1 Preliminaries

An *identity* over an alphabet X is a formal equality $u = v$, where u and v are words in the free monoid X^*, and is non-trivial if u and v are not equal. The elements of X are called *variables*. A monoid M *satisfies* the identity $u = v$ if, for every homomorphism $\phi : X^* \to M$, the equality $\phi(u) = \phi(v)$ holds in M. It is often useful to think of this in the informal way mentioned in the introduction: M satisfies the identity $u = v$ if equality in M holds under every substitution of variables in the words u and v by elements of M. Note that if a monoid satisfies an identity $u = v$, all of its submonoids and all of its homomorphic images satisfy $u = v$.

Lemma 9. *Let M be a monoid satisfying a non-trivial identity. Then the shortest (in terms of the sums of the lengths of the two sides) non-trivial identity satisfied by M is an identity over an alphabet with at most two variables.*

Proof. It is sufficient to show that, for any non-trivial identity over an alphabet with at most three variables, there is a strictly shorter identity with at most two variables. So suppose that $u = v$ is a non-trivial identity over X satisfied by M, where $|X| \geqslant 3$. Interchanging u and v if necessary, assume $|u| \leqslant |v|$. Let e_M be the identity (that is, the multiplicative neutral element) of M.

First consider the case where u is a prefix of v. By non-triviality, u is strictly shorter than v, so that $v = uw$ for some non-empty word w. Let $x \in X$ be a variable that appears in w. So $x^{|u|_x} = x^{|v|_x}$ is a non-trivial identity. To see that it is satisfied by M, proceed as follows: Let $\phi' : \{x\}^* \to M$ be a homomorphism. Extend ϕ' to a homomorphism $\phi : X \to M$ by defining $\phi(z) = e_M$ for all $z \in X \setminus \{x\}$. Note that $\phi'(x^{|u|_x}) = \phi(u)$ and $\phi'(x^{|v|_x}) = \phi(v)$ by the definition of ϕ', and that $\phi'(u) = \phi'(v)$ since M satisfies $u = v$. Combining these equaltities shows that $\phi(x^{|u|_x}) = \phi(x^{|v|_x})$. Hence M satisfies the non-trivial identity $x^{|u|_x} = x^{|v|_x}$ over the alphabet $\{x\}$.

Now suppose u is not a prefix of v. Suppose $u = u_1 \cdots u_{|u|}$ and $v = v_1 \cdots v_{|v|}$. Let i be minimal such that the variables u_i and v_i are different; suppose these variables are x and y. Let u' and v' be obtained from u and v respectively by deleting all variables other than x and y. By the choice of i, $u' = v'$ is a non-trivial identity. To see that it is satisfied by M, proceed as follows: Let $\phi' : \{x, y\}^* \to M$ be a homomorphism. Extend ϕ' to a homomorphism $\phi : X^* \to M$ by defining $\phi(z) = e_M$ for all $z \in X \setminus \{x, y\}$. Note that $\phi(u) = \phi'(u')$ and $\phi(v) = \phi'(v')$ by the definitions of the words u' and v' and the homomorphism ϕ; note also that $\phi(u) = \phi(v)$ since M satisfies the identity $u = v$. Combining these equalities shows that $\phi'(u') = \phi'(v')$. So M satisfies the non-trivial identity $u' = v'$ over the alphabet $\{x, y\}$.

Note that in both cases the resulting identity is strictly shorter than the original one. □

Lemma 10. *Let M be a monoid that contains a free monogenic submonoid and suppose that M satisfies an identity $u = v$ over an alphabet X. Then $|u| = |v|$ and $|u|_x = |v|_x$ for all $x \in X$. Consequently, if $u = v$ is non-trivial, then $|X| \geqslant 2$.*

Proof. Let $a \in M$ generate a free monogenic submonoid of M and let e_M be the identity of M. Let $x \in X$. Define

$$\phi_x : X^* \to M, \qquad x \mapsto a, \quad z \mapsto e_M \text{ for all } z \in X \setminus \{x\}.$$

Then $a^{|u|_x} = \phi_x(u) = \phi_x(v) = a^{|v|_x}$ since M satisfies $u = v$. Since a generates a free monogenic submonoid, $|u|_x = |v|_x$. Since this holds for all $x \in X$, it follows that $|u| = |v|$. □

The *length* of an identity $u = v$ with $|u| = |v|$ is defined to be $|u| = |v|$; note that this applies to any identity satisfied by a monoid that contains a free monogenic submonoid by Lemma 10.

Two identities $u = v$ and $u' = v'$ are *equivalent* if one can be obtained from the other by possibly renaming variables and swapping the two sides of the equality. For example, $xyxy = yxxy$ and $xyyx = yxyx$ are equivalent, since the second can be obtained from the first by interchanging the variables x and y and swapping the two sides.

Lemma 11. *1. Up to equivalence, the only length-2 non-trivial identity that can be satisfied by a monoid that contains a free monogenic submonoid is $xy = yx$; thus any homogeneous monoid satisfying a length-2 identity is commutative.*

2. Up to equivalence, the only length-3 non-trivial identities over a two-letter alphabet that can be satisfied by a monoid that contains a free monogenic submonoid are

$$xxy = xyx, \qquad xxy = yxx, \qquad xyx = yxx.$$

Proof. 1. Let $u_1u_2 = v_1v_2$ be a non-trivial identity satisfied by some monoid that contains a free monogenic submonoid. Suppose u_1 and v_1 are the same variable. Then Lemma 10, implies that u_2 and v_2 are also the same variable, contradicting non-triviality. Thus u_1 and v_1 are different variables. Then Lemma 10, implies that u_2 is the same variable as v_1 and v_2 is the same variable as u_1. Thus, up to equivalence, the identity is $xy = yx$.

2. Let $u_1u_2u_3 = v_1v_2v_3$ (where $u_i, v_i \in \{x, y\}$) be a non-trivial identity satisfied by some homogeneous monoid. Let I be the set of indices $i \in \{1, 2, 3\}$ where the variables u_i and v_i are the same. Clearly, non-triviality shows that $|I| < 3$. If $|I| = 2$, then, interchanging x and y if necessary, there is a unique $i \in \{1, 2, 3\}$ where u_i is x and v_i is y. Then $|u|_x = 1 + |v|_x$, since the words u and v differ only at the i-th symbol; this contradicts Lemma 10. If $|I| = 0$, then every symbol of u differs from every symbol of v. Since $|u| = 3$, at least one of $|u|_x$ and $|u|_y$ is at least 2. Interchanging x and y if necessary, assume $|u|_x \geqslant 2$. Then $|v|_x = |u|_y \leqslant 1$, which is a contadiction.

The only remaining possibility is $|I| = 1$. Interchanging x and y if necessary, assume that u_i and v_i are both x for exactly one $i \in 1, 2, 3$. For each of the three possibilities, the conditions $|u|_x = |v|_x$ and $|u|_y = |v|_y$ require the other symbols in u must be x and y, with the corresponding symbols in v being the opposite. Each possibility gives one of the given identities. □

Let $\mathsf{M} \in \{\mathsf{plac}, \mathsf{hypo}, \mathsf{sylv}, \mathsf{taig}, \mathsf{stal}, \mathsf{baxt}, \ell\mathsf{PS}, r\mathsf{PS}\}$. Then M contains the free monogenic submonoid M_1. (Note that M_1 thus satisfies the non-trivial identity $xy = yx$.) By Lemma 10, if M satisfies a non-trivial identity, it must be over an alphabet with at least two variables. The aim is to find shortest non-trivial identities satisfied by each monoid M: Lemmata 9 and 10 show that it will suffice to consider identities $u = v$ over $\{x, y\}$ with $|u| = |v|$. Note that since none of the monoids M is commutative, any identity must have length at least 3.

3.2 Hypoplactic monoid

The authors proved the following result using a quasi-crystal structure for the hypoplactic monoid [CM17, Theorem 9.3]; this subsection presents a direct proof.

Proposition 12. *The hypoplactic monoid satisfies the following non-trivial identities*

$$xyxy = xyyx = yxxy = yxyx;$$
$$xxyx = xyxx.$$

Furthermore, up to equivalence, these are the shortest non-trivial identities satisfied by the hypoplactic monoid.

Proof. The first goal is to show that these identities are satisfied by hypo. Let $s, t \in \mathsf{hypo}$, and let $p, q \in \mathcal{A}^*$ be words representing s, t, respectively. Let $B = \{a_1 < \cdots < a_k\}$ be the set of symbols in $\mathcal{A}$ that appear in at least one of p and q.

By Algorithm 2, symbols a_{i+1} and a_i are on the same row of $\mathrm{P}_{\mathsf{hypo}}(w)$ (for any $w \in B^*$) if and only if the word w does not contain a symbol a_i somewhere to the right of a symbol a_{i+1} (since inserting this symbol a_i results in the part of the quasi-ribbon tableau that contains a_{i+1} being glued *below* the entry a_i).

Suppose $pqpq$ contains a symbol a_i somewhere to the right of a symbol a_{i+1}. Then, regardless of whether these symbols both lie in u, both lie in v, or one lies in u and the other lies in v, the word $pqqp$ also contains a symbol a_i to the right of a symbol a_{i+1}. Similar reasoning establishes that if any of the words $pqpq$, $qppq$, $qpqp$, $pqqp$, $ppqp$, or $pqpp$, contains a symbol a_i somewhere to the right of a symbol a_{i+1}, so do all the others. Hence either the symbols a_i and a_{i+1} are on the same row in all of $\mathrm{P}_{\mathsf{hypo}}(pqpq)$, $\mathrm{P}_{\mathsf{hypo}}(pqqp)$, $\mathrm{P}_{\mathsf{hypo}}(qppq)$, $\mathrm{P}_{\mathsf{hypo}}(qpqp)$, $\mathrm{P}_{\mathsf{hypo}}(ppqp)$, and $\mathrm{P}_{\mathsf{hypo}}(pqpp)$, or on different rows in all of them.

A quasi-ribbon tableau is clearly determined by its evaluation and by knowledge of whether adjacent different entries are on the same row. Thus, noting that $\mathrm{ev}(pqpq) = \mathrm{ev}(qppq) = \mathrm{ev}(qpqp) = \mathrm{ev}(pqqp)$, it follows from the previous paragraph that

$$ststst = \mathrm{P}_{\mathsf{hypo}}(pqpq) = tsst = \mathrm{P}_{\mathsf{hypo}}(qppq) = tsts = \mathrm{P}_{\mathsf{hypo}}(qpqp) = stts = \mathrm{P}_{\mathsf{hypo}}(pqqp).$$

Similarly, noting that $\mathrm{ev}(ppqp) = \mathrm{ev}(pqpp)$, it follows that

$$ssts = \mathrm{P}_{\mathsf{hypo}}(ppqp) = \mathrm{P}_{\mathsf{hypo}}(pqpp) = stss.$$

The next goal is to show that, up to equivalence, these are the only length-4 non-trivial identities satisfied by hypo. Let $u = v$ be a length-4 non-trivial identity over $\{x, y\}$ satisfied by hypo. By Lemma 10, $|u|_x = |v|_x$ and $|u|_y = |v|_y$.

By Algorithm 2, symbols 1 and 2 are on different rows of $\mathrm{P}_{\mathsf{hypo}}(w)$ (for any $w \in \{1,2\}^*$) if and only if the word w contains a symbol 1 somewhere to the right of a symbol 2. Suppose u contains a factor xy. Let $\phi : \{x, y\}^* \to \mathsf{hypo}$ map x to 2 and y to 1; then $\mathrm{P}_{\mathsf{hypo}}(\phi(u))$ has 1 and 2 on different rows. The same must hold for $\mathrm{P}_{\mathsf{hypo}}(\phi(v))$, and thus v must contain a factor xy. The converse is similar, and the reasoning is parallel for factors yx. Thus u contains a factor xy (respectively, yx) if and only if v contains a factor xy (respectively, yx).

If both u and v contain only factors xy, then $u = x^{|u|_x}y^{|u|_y} = x^{|v|_x}y^{|v|_y} = v$, which contradicts non-triviality. Similarly, it is impossible for u and v to contain only factors yx. So both u and v contain factors xy and yx.

Interchanging x and y if necessary, assume $|u|_x = |v|_x \geqslant |u|_y = |v|_y$. First consider the case where $|u|_x = |v|_x = 3$ and $|u|_y = |v|_y = 1$. Since u and v both contain xy and yx, this implies that $u, v \in \{xxyx, xyxx\}$. Now consider the case $|u|_x = |v|_x = 2$ and $|u|_y = |v|_y = 2$. Again, since u and v both contain xy and yx, this implies that $u, v \in \{xyxy, xyyx, yxxy, yxyx\}$. Thus $u = v$ is equivalent to one of the identities in the statement.

The final goal is to prove that no length-3 non-trivial identity is satisfied by hypo. Consider the length-3 identities from Lemma 11(2):

$$xxy = xyx, \qquad xxy = yxx, \qquad xyx = yxx.$$

If one puts $x = 1$ and $y = 2$ into the first two identities and $x = 2$ and $y = 1$ in the third, one sees that hypo satisfies none of them:

$$\mathrm{P}_{\mathsf{hypo}}(112) = \begin{array}{|c|c|c|}\hline 1&1&2\\\hline\end{array} \neq \begin{array}{|c|c|}\hline 1&1\\\hline 2\\\cline{1-1}\end{array} = \mathrm{P}_{\mathsf{hypo}}(121);$$

$$\mathrm{P}_{\mathsf{hypo}}(112) = \begin{array}{|c|c|c|}\hline 1&1&2\\\hline\end{array} \neq \begin{array}{|c|c|}\hline 1&1\\\hline 2\\\cline{1-1}\end{array} = \mathrm{P}_{\mathsf{hypo}}(211);$$

$$\mathrm{P}_{\mathsf{hypo}}(212) = \begin{array}{|c|}\hline 1\\\hline 2\\\cline{1-1}\end{array}\!\!\begin{array}{|c|}\hline 2\\\hline\end{array} \neq \begin{array}{|c|c|c|}\hline 1&2&2\\\hline\end{array} = \mathrm{P}_{\mathsf{hypo}}(122).$$

Thus hypo satisfies no length-3 identity. □

3.3 Stalactic and taiga monoids

Lemma 13. *The stalactic monoid satisfies the non-trivial identity $xyx = yxx$.*

Proof. Let $x, y \in \mathsf{stal}$ and let $u, v \in \mathcal{A}^*$ be words representing x, y, respectively. Since the tree $\mathrm{P}_{\mathsf{stal}}(w)$ is computed by applying Algorithm 5 to each symbol in the word $w \in \mathcal{A}^*$, proceeding right to left, it is clear that the order of the rightmost appearances of each symbol in w determines order of symbols in the top row of the stalactic tableau $\mathrm{P}_{\mathsf{stal}}(w)$. Since the order of rightmost appearances of each symbol in the words uvu and vuu are the same, the order of symbols in the top row of $\mathrm{P}_{\mathsf{stal}}(uvu)$ and $\mathrm{P}_{\mathsf{stal}}(vuu)$ is also the same. Since $\mathrm{ev}(uvu) = \mathrm{ev}(vuu)$, the length of corresponding columns in $\mathrm{P}_{\mathsf{stal}}(uvu)$ and $\mathrm{P}_{\mathsf{stal}}(vuu)$ are equal. Hence $xyx = \mathrm{P}_{\mathsf{stal}}(uvu) = \mathrm{P}_{\mathsf{stal}}(vuu) = yxx$. □

Lemma 14. *The taiga monoid does not satisfy either of the identities $xxy = xyx$ or $xxy = yxx$, and does not satisfy a non-trivial identity of length 2.*

Proof. To see that taig does not satisfy either of the identities $xxy = xyx$ or $xxy = yxx$, note that the rightmost symbol of $w \in \mathcal{A}^*$ labels the root node of $\mathrm{P}_{\mathsf{taig}}(w)$, but the rightmost symbols of both these identities are different. Thus substituting 1 for x and 2 for y shows that neither identity can be satisfied by taig.

Finally, taig does not satisfy a non-trivial identity of length 2 since it is clearly non-commutative. □

Since the taiga monoid is a homomorphic image of the stalactic monoid [Pri13, § 5], any identity that is satisfied by stal is satisfied by taig. Combining this with Lemmata 11, 13, and 14 gives the following results:

Proposition 15. *The stalactic monoid satisfies the non-trivial identity $xyx = yxx$. Furthermore, this is the unique shortest non-trivial identity satisfied by the stalactic monoid.*

Proposition 16. *The taiga monoid satisfies the non-trivial identity $xyx = yxx$. Furthermore, this is the unique shortest non-trivial identity satisfied by the taiga monoid.*

3.4 Sylvester monoid

By the definition of a binary search tree, if a tree has a node with label a, then any other node with label a is either above it or in its left subtree. This shows that there is a unique path from the root to a leaf node that contains all nodes labelled a. In particular, there is at most one leaf node with label a. Furthermore, there is a unique node a that is both the leftmost node with label a and the node labelled a that is furthest from the root. Similarly, there is a unique node a that is both the rightmost node with label a and the node labelled a that is closest to the root.

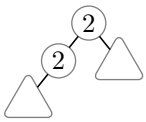
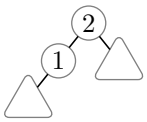
Lemma 17. *The sylvester monoid is left-cancellative.*

Proof. Let $u, v \in \mathcal{A}^*$ and $a \in \mathcal{A}$. Suppose that $\mathrm{P}_{\mathsf{sylv}}(au) = \mathrm{P}_{\mathsf{sylv}}(av)$. That is, the binary search trees $\mathrm{P}_{\mathsf{sylv}}(au)$ and $\mathrm{P}_{\mathsf{sylv}}(av)$ are equal. Note further that the last symbol a inserted into both binary search trees is a leaf node. Deleting this (unique) leaf node labelled a from the equal trees $\mathrm{P}_{\mathsf{sylv}}(au)$ and $\mathrm{P}_{\mathsf{sylv}}(av)$ leaves equal binary search trees. That is, $\mathrm{P}_{\mathsf{sylv}}(u)$ and $\mathrm{P}_{\mathsf{sylv}}(v)$ are equal.

Since u and v are arbitrary words and a represents an arbitrary generator, it follows that sylv is left-cancellative. □

Lemma 18. *The sylvester monoid does not satisfy an identity equivalent to one of the form $uxx = vyx$, for any $u, v \in \{x, y\}^*$.*

Proof. Using the Algorithm 3, one sees that

$\mathrm{P}_{\mathsf{sylv}}(\cdots 22) =$  and $\mathrm{P}_{\mathsf{sylv}}(\cdots 12) =$  ;

note that these binary search trees differ in the left child of the root node. Thus to see that the identity $uxx = vyx$ cannot be satisfied by sylv, one can substitute 2 for x and 1 for y. □

Lemma 19. *Let $p, q, r \in \mathsf{sylv}$ be such that $\mathrm{ev}(p) = \mathrm{ev}(q) = \mathrm{ev}(r)$. Then $pr = qr$.*

Proof. Let $u, v, w \in \mathcal{A}^*$ be words representing p, q, r, respectively. The tree $\mathrm{P}_{\mathsf{sylv}}(x)$ is computed by applying Algorithm 3 to each symbol in x, proceeding right to left. That is, $\mathrm{P}_{\mathsf{sylv}}(uw)$ and $\mathrm{P}_{\mathsf{sylv}}(vw)$ are obtained by inserting u and v (respectively) into $\mathrm{P}_{\mathsf{sylv}}(w)$. Since $\mathrm{ev}(u) = \mathrm{ev}(v) = \mathrm{ev}(w)$, every symbol that appears in u or v also appears in w and thus in $\mathrm{P}_{\mathsf{sylv}}(w)$.

Consider how further symbols from u or v are inserted into $\mathrm{P}_{\mathsf{sylv}}(w)$:

- Let a be the smallest symbol that appears in u, v, and w. Then the leftmost node of $\mathrm{P}_{\mathsf{sylv}}(w)$ must be labelled by a. Thus, during the computation of $\mathrm{P}_{\mathsf{sylv}}(uw)$ and $\mathrm{P}_{\mathsf{sylv}}(vw)$, all symbols a in u and v will certainly be inserted into the left subtree of this leftmost node in $\mathrm{P}_{\mathsf{sylv}}(w)$, and no other symbols are inserted into this left subtree by the minimality of a.

- Let c be some symbol other than a that appears in u, v, and w, and let b be the maximum symbol less than c appearing in u, v, and w. Let N_c be the leftmost node in $\mathrm{P}_{\mathsf{sylv}}(w)$ labelled by c and let N_b be the rightmost node in $\mathrm{P}_{\mathsf{sylv}}(w)$ labelled by b.

  Suppose v is the label of the lowest common ancestor N_v of the nodes N_b and N_c. If N_v is neither N_b nor N_c, then $b \leqslant v < c$, which implies $v = b$ by the choice of b, which contradicts the fact that N_b is the rightmost node labelled b. Thus the lowest common ancestor of N_b and N_c must be one of N_b or N_c. That is, one of the following must hold:

  - N_b is above N_c. Then N_c is in the right subtree of N_b, since $b < c$. Since there is no node that is to the right of N_b and to the left of N_c, it follows that N_c has an empty left subtree. Thus any symbol c in u or v will be inserted into this currently empty left subtree of N_c, and no other symbols will be inserted into this subtree by the maximality of b among the symbols less than c.
  - N_c is above N_b. Then N_b is in the left subtree of N_c, since $b < c$. Since there is no node that is to the right of N_b and to the left of N_c, it follows that N_b is the left child of N_c, and that N_b has an empty right subtree. Thus any symbol c

in u or v will be inserted into this currently empty right subtree of N_b, and no other symbols will be inserted into this subtree by the maximality of b among the symbols less than c.

Combining these cases, one sees that every symbol d from u or v is inserted into a particular previously empty subtree of $\mathrm{P}_{\mathsf{sylv}}(w)$, dependent only on the value of the symbol d (and not on its position in u or v), and that unequal symbols are inserted into different subtrees. Since $\mathrm{ev}(u) = \mathrm{ev}(v)$, the same number of symbols d are inserted, for each such symbol d. Hence $\mathrm{P}_{\mathsf{sylv}}(uw) = \mathrm{P}_{\mathsf{sylv}}(vw)$ and thus $pr = \mathrm{P}_{\mathsf{stal}}(uw) = \mathrm{P}_{\mathsf{stal}}(vw) = qr$. □

Proposition 20. *The sylvester monoid satisfies the non-trivial identity $xyxy = yxxy$. Furthermore, up to equivalence, this is the unique shortest identity satisfied by the sylvester monoid.*

Proof. Let $x, y \in \mathsf{sylv}$. Let $p = r = xy$ and $q = yx$. Then $\mathrm{ev}(p) = \mathrm{ev}(q) = \mathrm{ev}(r)$, so $pr = qr$ by Lemma 19. Thus sylv satisfies $xyxy = yxxy$.

Since hypo is a homomorphic image of sylv [Pri13, Example 6], and hypo satisfies no length-3 identity by Proposition 12, it follows that sylv cannot satisfy a length-3 identity.

So let $u = v$ be some length-4 identity satisfied by sylv. Suppose $u = u_1u_2u_3u_4$ and $v = v_1v_2v_3v_4$, where $u_i, v_i \in \{x, y\}$. Since the rightmost symbol in a word w determines the root node of $\mathrm{P}_{\mathsf{sylv}}(w)$, it follows that u_4 and v_4 are the same symbol; interchanging x and y if necessary, assume this is y. If u_1 and v_1 were the same symbol, then the left-cancellativity of sylv (Lemma 17) would imply that sylv satisfied the length-3 identity $u_2u_3u_4 = v_2v_3v_4$, which contradicts the previous paragraph. Thus u_1 and v_1 are different symbols. Interchanging u and v if necessary, assume that u_1 is x and v_1 is y. The condition $|u|_y = |v|_y$ implies that at least one of u_2 and u_3 is y, which yields the following three possibilities for u:

$$xyyy, \quad xxyy, \quad xyxy.$$

The conditions $|u|_x = |v|_x$ and $|u|_y = |v|_y$ now imply the following four possibilities for the identity $u = v$ (with the first possibility for u above giving two different identities):

$$xyyy = yxyy, \quad xyyy = yyxy, \quad xxyy = yxxy, \quad xyxy = yxxy.$$

The last of these is the identity already proven to hold in sylv. The second and third cannot hold in sylv by Lemma 18. In the first identity, putting $x = 1$ and $y = 2$ proves that it is not satisfied by sylv:

$$\mathrm{P}_{\mathsf{sylv}}(1222) = \begin{array}{ccccccc} & & & & & & \textcircled{2} \\ & & & & & \textcircled{2} & \\ & & & & \textcircled{2} & & \\ & & & \textcircled{1} & & & \end{array} \neq \begin{array}{ccccccc} & & & & & & \textcircled{2} \\ & & & & & \textcircled{2} & \\ & & & & \textcircled{1} & & \\ & & & \textcircled{2} & & & \end{array} = \mathrm{P}_{\mathsf{sylv}}(2122)$$

Thus $xyxy = yxxy$ is the unique shortest identity satisfied by sylv. □

Since the sylvester and #-sylvester monoids are anti-isomorphic, the following results follow by dual reasoning:

Lemma 21. *The #-sylvester monoid is right-cancellative.*

Lemma 22. *The #-sylvester monoid does not satisfy an identity equivalent to one of the form $xxu = xyv$, for any $u, v \in \{x, y\}^*$.*

Lemma 23. *Let $p, q, s \in \mathsf{sylv}^{\#}$ be such that $\mathrm{ev}(p) = \mathrm{ev}(q) = \mathrm{ev}(s)$. Then $sp = sq$.*

Proposition 24. *The #-sylvester monoid satisfies the non-trivial identity $yxyx = yxxy$. Furthermore, up to equivalence, this is the unique shortest identity satisfied by the #-sylvester monoid.*

3.5 Baxter monoid

Lemma 25. *Let $p, q, r, s \in \mathsf{baxt}$ be such that $\mathrm{ev}(p) = \mathrm{ev}(q) = \mathrm{ev}(r) = \mathrm{ev}(s)$. Then $spr = sqr$.*

Proof. Let $t, u, v, w \in \mathcal{A}^*$ represent p, q, r, s, respectively.

Let $t' = wt$ and $u' = wu$; note that $\mathrm{ev}(t') = \mathrm{ev}(u')$. By Lemma 19 applied to the elements $\mathrm{P}_{\mathsf{sylv}}(t'), \mathrm{P}_{\mathsf{sylv}}(u'), \mathrm{P}_{\mathsf{sylv}}(v)$, it follows that $\mathrm{P}_{\mathsf{sylv}}(t'v) = \mathrm{P}_{\mathsf{sylv}}(u'v)$; thus $\mathrm{P}_{\mathsf{sylv}}(wtv) = \mathrm{P}_{\mathsf{sylv}}(wuv)$

Let $t'' = tv$ and $u'' = uv$; note that $\mathrm{ev}(t'') = \mathrm{ev}(u'')$. By Lemma 23 applied to the elements $\mathrm{P}_{\mathsf{sylv}^{\#}}(t'), \mathrm{P}_{\mathsf{sylv}^{\#}}(u'), \mathrm{P}_{\mathsf{sylv}^{\#}}(w)$, it follows that $\mathrm{P}_{\mathsf{sylv}^{\#}}(wt'') = \mathrm{P}_{\mathsf{sylv}^{\#}}(wu'')$; thus $\mathrm{P}_{\mathsf{sylv}^{\#}}(wtv) = \mathrm{P}_{\mathsf{sylv}^{\#}}(wuv)$.

Hence

$$\begin{aligned} spr = \mathrm{P}_{\mathsf{baxt}}(wuv) &= \big(\mathrm{P}_{\mathsf{sylv}^{\#}}(wtv), \mathrm{P}_{\mathsf{sylv}}(wtv)\big) \\ &= \big(\mathrm{P}_{\mathsf{sylv}^{\#}}(wuv), \mathrm{P}_{\mathsf{sylv}}(wuv)\big) = \mathrm{P}_{\mathsf{baxt}}(wuv) = sqp. \end{aligned}$$ □

Proposition 26. *The Baxter monoid satisfies the identities*

$$yxxyxy = yxyxxy \text{ and } xyxyxy = xyyxxy.$$

Furthermore, up to equivalence, these are the unique shortest non-trivial identities satisfied by the Baxter monoid.

Proof. Let $x, y \in \mathsf{baxt}$. Let $p = r = xy$ and $q = s = yx$. Then $\mathrm{ev}(p) = \mathrm{ev}(q) = \mathrm{ev}(r) = \mathrm{ev}(s)$. Thus, by Lemma 25, $yxxyxy = spr = sqr = yxyxxy$. The same reasoning with $s = xy$ shows that $xyxyxy = xyyxxy$.

The next step is to show that, up to equivalence, these are the only identities of length 6 satisfied by baxt. So suppose that $u = v$ is an identity of length 6 satisfied by baxt, with $u = u_1u_2\cdots u_6$ and $v = v_1v_2\cdots v_6$, where $u_i, v_i \in \{x, y\}$. Since the first symbol in a word w determines the root symbol of the left-hand tree in the pair $\mathrm{P}_{\mathsf{baxt}}(u)$, and since the last symbol determines the root symbol of the right-hand tree, it follows that u_1 and v_1 must be the same variable, and that u_6 and v_6 must be the same variable. By Lemmata 18 and 22, u_2 and v_2 must be the same variable, and u_5 and v_5 must be the same variable. Since sylv and $\mathsf{sylv}^{\#}$ are both homomorphic images of baxt [Gir12, Proposition 3.7], the identity $u = v$ is also satified by sylv and $\mathsf{sylv}^{\#}$. Since sylv is left-cancellative by Lemma 17, $u_3u_4u_5u_6 = v_3v_4v_5v_6$ is satisfied by sylv and is thus equivalent to $xyxy = yxxy$. The aim

is to characterize $u = v$ up to equivalence, so assume that $u_3u_4u_5u_6 = v_3v_4v_5v_6$ actually is the identity $xyxy = yxxy$.

Since $\mathsf{sylv}^\#$ is right-cancellative by Lemma 21, $u_1u_2u_3u_4 = v_1v_2v_3v_4$ is satisfied by $\mathsf{sylv}^\#$ and is thus equivalent to $yxyx = yxxy$, which is equivalent (by swapping the two sides) to $yxxy = yxyx$ and (by interchaning x and y) to $xyxy = xyyx$. Combining these with $u_3u_4u_5u_6 = v_3v_4v_5v_6$ from the previous paragraph shows that $u = v$ is (up to equivalence) either $yxxyxy = yxyxxy$ or $xyxyxy = xyyxxy$.

Finally, it is necessary to show that no length-5 identity is satisfied by baxt. Suppose $u = v$ is an identity of length 5 satisfied by the Baxter monoid. Suppose that $u = u_1u_2\cdots u_5$ and $v = v_1v_2\cdots v_5$, where $u_i, v_i \in \{x, y\}$. As before, considering the root symbols of the left-hand and right-hand trees in $\mathrm{P}_{\mathsf{baxt}}(w)$ shows that u_1 and v_1 must be the same symbol, and that u_5 and v_5 must be the same symbol.

Since sylv and $\mathsf{sylv}^\#$ are both homomorphic images of baxt [Gir12, Proposition 3.7], both of them satisfy the identity $u = v$. Furthermore, $\mathsf{sylv}^\#$ is right-cancellative and so satisfies the identity $u_1u_2u_3u_4 = v_1v_2v_3v_4$; while sylv is left-cancellative and so satisfies the identity $u_2u_3u_4u_5 = v_2v_3v_4v_5$. By Proposition 20, the unique length-4 identity satisfied by sylv is $xyxy = yxxy$, so the identity $u = v$ is either $xxyxy = xyxxy$ or $yxyxy = yyxxy$. Deleting the rightmost symbol y from each of these identities yields $xxyx = xyxx$ and $yxyx = yyxx$, one of which must be the identity $u_1u_2u_3u_4 = v_1v_2v_3v_4$ satisfied by $\mathsf{sylv}^\#$. This is a contradiction, since by Proposition 24 the only length-4 identity satisfied by $\mathsf{sylv}^\#$ is $yxyx = yxxy$. □

3.6 Left and right patience sorting monoids

The present authors and Silva [CMS, § 4.2] have shown that the elements $\mathrm{P}_{\ell\mathsf{PS}}(21)$ and $\mathrm{P}_{\ell\mathsf{PS}}(1)$ generate a free submonoid of $\ell\mathsf{PS}$. Since free monoids of rank at least 2 satisfy no non-trivial identities, it follows that $\ell\mathsf{PS}$ does not satisfy a non-trivial identity. Furthermore, $\ell\mathsf{PS}_n$ for $n \geqslant 2$ satisfies no non-trivial identities, while $\ell\mathsf{PS}_1$, as a monogenic monoid, is of course commutative and satisfies the identity $xy = yx$.

For any n, the monoid $r\mathsf{PS}_n$ satisfies the identity $(xy)^{n+1} = (xy)^n yx$ but does not satisfy any non-trivial identity of length less than or equal to n [CMS, § 4.2], which immediately implies that $r\mathsf{PS}$ does not satisfy any non-trivial identity.

3.7 Plactic monoid

Recently, the present authors together with Klein, Kubat, and Okniński used the combinatorics of Young tableaux to prove that plac_n does not satisfy any identity of length less than or equal to n [CKK+17, Proposition 3.1], which implies that plac does not satisfy a non-trivial identity.

For finite-rank plactic monoids, some more partial results are known. First, plac_1 is monogenic and thus commutative and satisfies $xy = yx$. Kubat & Okniński [KO14] have shown that plac_2 satisfies Adian's identity $xyyxxyxyyx = xyyxyxxyyx$, and that plac_3 satisfies the identity $pqqpqp = pqpqqp$, where $p(x, y)$ and $q(x, y)$ are respectively the left

and right side of Adian's identity (and so the identity $pqqpqp = pqpqqp$ has sixty variables x or y on each side). Furthermore, plac_3 does not satisfy Adian's identity [KO14, p. 111–2].

Furthermore plac_3 satisfies the identity $pqppq = pqqpq$, where as above $p(x, y)$ and $q(x, y)$ are respectively the left and right side of Adian's identity [CKK+17, Corollary 5.4]. The proof technique was rather different, via an embedding of plac_3 into the direct product of two copies of the monoid of 3×3 upper-triangular tropical matrices, which in turn is shown to satisfy the given identity.

Conjecture 27. For each $n \geqslant 4$, there is a non-trivial identity satisfied by plac_n.

References

[Adi66] S. Adian. 'Defining relations and algorithmic problems for groups and semigroups'. *Trudy Mat. Inst. Steklov*, 85 (1966), pp. 3–123. URL: http://www.mathnet.ru/eng/tm2766.

[CEK+01] J. Cassaigne, M. Espie, D. Krob, J.-C. Novelli, & F. Hivert. 'The Chinese Monoid'. *Int. J. Alg. Comput.*, 11, no. 03 (2001), pp. 301–334. doi:10.1142/S0218196701000425.

[CKK+17] A. J. Cain, G. Klein, Ł. Kubat, A. Malheiro. & J. Okniński. 'A note on identities in plactic monoids and monoids of upper-triangular tropical matrices'. ArXiv e-prints, arXiv:1705.04596.

[CM17] A. J. Cain & A. Malheiro. 'Crystallizing the hypoplactic monoid: from quasi-Kashiwara operators to the Robinson–Schensted–Knuth-type correspondence for quasi-ribbon tableaux'. *J. Algebraic Combin.*, 45, no. 2 (2017), pp. 475–524. doi:10.1007/s10801-016-0714-6.

[CMS] A. J. Cain, A. Malheiro, & F. Silva. 'The monoids of the patience sorting algorithm'. Accepted for publication in *Int. J. Alg. Comput.*

[DK94] G. Duchamp & D. Krob. 'Plactic-growth-like monoids'. In M. Ito & H. Jürgensen, eds, *Words, languages and combinatorics, II*, p. 124–142, River Edge, NJ, 1994. World Scientific.

[Ful97] W. Fulton. *Young Tableaux: With Applications to Representation Theory and Geometry*. No. 35 in *LMS Student Texts*. Cambridge University Press, 1997.

[Gir12] S. Giraudo. 'Algebraic and combinatorial structures on pairs of twin binary trees'. *J. Algebra*, 360 (2012), pp. 115–157. doi:10.1016/j.jalgebra.2012.03.020.

[Gro81] M. Gromov. 'Groups of polynomial growth and expanding maps'. *Publ. Math. Inst. Hautes Études Sci.*, 53 (1981), pp. 53–78. URL: http://www.numdam.org/item?id=PMIHES_1981__53__53_0.

[HNT05] F. Hivert, J.-C. Novelli, & J.-Y. Thibon. 'The algebra of binary search trees'. *Theoret. Comput. Sci.*, 339, no. 1 (2005), pp. 129–165. doi:10.1016/j.tcs.2005.01.012.

[HNT07] F. Hivert, J.-C. Novelli, & J.-Y. Thibon. 'Commutative combinatorial Hopf algebras'. *J Algebr Comb*, 28, no. 1 (2007), pp. 65–95. doi:10.1007/s10801-007-0077-0.

[Jed11] 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 Comput. Sci.*, pp. 350–353. Springer, 2011. doi:10.1007/978-3-642-21590-2_31.

[JO11] J. Jaszuńska & J. Okniński. 'Structure of Chinese algebras'. *J. Algebra*, 346, no. 1 (2011), pp. 31–81. doi:10.1016/j.jalgebra.2011.08.020.

[KO14] Ł. Kubat & J. Okniński. 'Identities of the plactic monoid'. *Semigroup Forum*, 90, no. 1 (2014), pp. 100–112. doi:10.1007/s00233-014-9609-9.

[KT97] D. Krob & J.-Y. Thibon. 'Noncommutative Symmetric Functions IV: Quantum Linear Groups and Hecke Algebras at $q = 0$'. *J. Algebraic Combin.*, 6, no. 4 (1997), pp. 339–376. doi:10.1023/A:1008673127310.

[Lot02] M. Lothaire. *Algebraic Combinatorics on Words*. No. 90 in *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2002.

[LS78] A. Lascoux & M.-P. Schützenberger. 'Sur une conjecture de H. O. Foulkes'. *C. R. Acad. Sci. Paris Sér. A-B*, 286, no. 7 (1978), pp. A323–A324.

[LS81] A. Lascoux & M.-P. Schützenberger. 'Le monoïde plaxique'. In *Noncommutative structures in algebra and geometric combinatorics*, no. 109 in *Quaderni de "La Ricerca Scientifica"*, pp. 129–156, Rome, 1981. CNR. URL: http://igm.univ-mlv.fr/~berstel/Mps/Travaux/A/1981-1PlaxiqueNaples.pdf.

[LS85] A. Lascoux & M.-P. Schützenberger. 'Schubert polynomials and the Littlewood-Richardson rule'. *Lett. Math. Phys.*, 10, no. 2-3 (1985), pp. 111–124. doi:10.1007/BF00398147.

[LS90] A. Lascoux & M.-P. Schützenberger. 'Tableaux and noncommutative Schubert polynomials'. *Funct. Anal. Its. Appl.*, 23, no. 3 (1990), pp. 223–225. doi:10.1007/BF01079531.

[Mac08] I. Macdonald. *Symmetric Functions and Hall Polynomials*. Clarendon Press, Oxford University Press, 2008.

[Mal53] A. Malcev. 'Nilpotent semigroups'. *Ivanov. Gos. Ped. Inst., Uchenye Zap., Fiz.-Mat. Nauki*, 4 (1953), pp. 107–111.

[Nov00] J.-C. Novelli. 'On the hypoplactic monoid'. *Discrete Mathematics*, 217, no. 1-3 (2000), pp. 315–336. doi:10.1016/S0012-365X(99)00270-8.

[NT63] B. Neumann & T. Taylor. 'Subsemigroups of Nilpotent Groups'. *Proc. Roy. Soc. Ser. A*, 274 (1963), pp. 1–4.

[Pri13] J.-B. Priez. 'A lattice of combinatorial Hopf algebras: Binary trees with multiplicities'. In *Formal Power Series and Algebraic Combinatorics*, Nancy, 2013. The Association. Discrete Mathematics & Theoretical Computer Science. URL: https://dmtcs.episciences.org/2372/pdf.

[Rey07] M. Rey. 'Algebraic constructions on set partitions'. In *Formal Power Series and Algebraic Combinatorics*, 2007. URL: http://www-igm.univ-mlv.fr/~rey/articles/rey-fpsac07.pdf.

[Shn93] L. Shneerson. 'Identities in finitely generated semigroups of polynomial growth'. *J. Algebra*, 154, no. 1 (1993), pp. 67–85. doi:10.1006/jabr.1993.1004.

[TY11] H. Thomas & A. Yong. 'Longest increasing subsequences, Plancherel-type measure and the Hecke insertion algorithm'. *Advances in Applied Mathematics*, 46, no. 1 (2011), pp. 610–642. doi:10.1016/j.aam.2009.07.005.