DECIDING CONJUGACY IN SYLVESTER MONOIDS AND OTHER HOMOGENEOUS MONOIDS

ALAN J. CAIN AND ANTÓNIO MALHEIRO

Abstract. We give a combinatorial characterization of conjugacy in the sylvester monoid (the monoid of binary search trees), showing that conjugacy is decidable for this monoid. We then prove that conjugacy is undecidable in general for homogeneous monoids and even for multi-homogeneous monoids.

1. Introduction

The notion of conjugation, which is extremely natural for groups, can be generalized to semigroups in more than one way. Two elements x and y of a group G are conjugate (in G) if there exists $g \in G$ such that $y = g^{-1}xg$. A first attempt, due to Lallement [Lal79], to generalize this definition to a semigroup S is to define the *ℓ-conjugacy* relation $\sim_\ell$ by

$$x \sim_\ell y \iff (\exists g \in S^1)(xg = gy). \tag{1.1}$$

(where S^1 denotes the semigroup S with an identity adjoined). It is easy to prove that $\sim_\ell$ is reflexive and transitive, but not symmetric. A symmetric analogy, introduced by Otto [Ott84], is *o-conjugacy*, defined by

$$x \sim_o y \iff (\exists g, h \in S^1)(xg = gy \wedge hx = yh). \tag{1.2}$$

The relation $\sim_o$ is an equivalence relation. When the semigroup S contains a zero 0_S, both $\sim_\ell$ and $\sim_o$ are the universal relation $S \times S$, because taking $g = h = 0_S$ in (1.1) and (1.2) shows that all elements of S are $\sim_\ell$- and $\sim_o$-related. This motivated Araújo, Konieczny, and the second author [AKM14] to introduce the c-conjugacy relation $\sim_c$, which is not simply the universal

2010 *Mathematics Subject Classification.* 20M10 (Primary) 05C05 03D10 20M05 (Secondary).

Key words and phrases. Conjugacy; decidability; homogeneous monoid; sylvester monoid.

During the research that led to this paper, the first author was initially supported by the European Regional Development Fund through the programme COMPETE and by the Portuguese Government through the FCT (Fundação para a Ciência e a Tecnologia) under the project PEst-C/MAT/UI0144/2011 and through an FCT Ciência 2008 fellowship, and later supported by an FCT Investigador advanced fellowship (IF/01622/2013/CP1161/CT0001).

For the second author, this work was developed within the project PEst-OE/MAT/UI0143/2014 of CAUL, FCUL..

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/00297/2013 (Centro de Matemática e Aplicações). The authors thank the anonymous referee for their very careful reading of the paper and for many helpful comments and suggestions.

relation when S has a zero, but is genuinely useful. In semigroups that do not contain zeroes, including the semigroups and monoids we consider in this paper, $\sim_c$ and $\sim_o$ coincide, and so we will not give the full definition of $\sim_c$.

Another approach is to define the *primary conjugacy* relation $\sim_p$ by

$$x \sim_p y \iff (\exists u, v \in S^1)(x = uv \wedge y = vu).$$

However, $\sim_p$ is reflexive and symmetric, but not transitive; hence it is sensible to follow Kudryavtseva & Mazorchuk [KM09, KM07] in working with its transitive closure $\sim_p^*$. It is easy to show that $\sim_p \subseteq \sim_p^* \subseteq \sim_o \subseteq \sim_\ell$. In some circumstances, equality holds. For instance, in the free inverse monoid, $\sim_p^*$ and $\sim_o$ coincide when restricted to non-idempotents [Cho93]; this result has been extended to inverse monoids presented by a single defining relation that has the form of a Dyck word [Sil96].

This paper is concerned with conjugacy in homogeneous monoids (that is, monoids with presentations where the two sides of each defining relation have the same length) and multihomogeneous monoids (that is, monoids with presentations where the two side of each defining relation contain the same number of each generator; see § 2 for the formal definitions). Homogeneous monoids do not contain zeroes, so $\sim_c$ and $\sim_o$ coincide. The Plactic monoid [Lot02, ch. 7] and Chinese monoid [CEK$^+$01], which are multihomogeneous monoids with important connections to combinatorics and algebra, both have elegant combinatorial characterizations of conjugacy: in both monoids, the relations $\sim_p^*$ and $\sim_o$ coincide, and two elements, expressed as words over the usual set of generators, are conjugate (that is, $\sim_p^*$- and $\sim_o$-related) if and only if each generator appears the same number of times in each word; see [LS81, § 4] and [CEK$^+$01, Theorem 5.1].

Our first main result shows that the same characterization of conjugacy holds for the sylvester monoid (Theorem 3.4). The sylvester monoid was defined by Hivert, Novelli & Thibon [HNT05] as an analogue of the plactic monoid where Schensted's algorithm for insertion into Young tableaux (see [Lot02, ch. 5]) is replaced by insertion into a binary search tree; from the sylvester monoid, one then recovers the Hopf algebra of planar binary trees defined by Loday & Ronco [LR98]. Recently, the authors and Gray proved that the standard presentations for the finite-rank versions of sylvester monoids form (infinite) complete rewriting systems, and that finite-rank sylvester monoids are biautomatic [CGM15, § 5].

We then prove that there exist homogeneous monoids where the problem of deciding o-conjugacy is undecidable (Theorem 4.1). This strengthens a result of Narendran & Otto showing that o-conjugacy is undecidable in general for monoids presented by finite complete rewriting systems [NO86, Lemma 3.6] and by almost-confluent rewriting systems [NO85, Theorem 3.4]. (Homogeneous presentations form almost-confluent rewriting systems [NO85, Proposition 3.2].) We then apply a technique the authors developed with Gray [CGM] to deduce that there are multihomogeneous monoids in which the o-conjugacy problem is undecidable (Theorem 4.8).

2. Preliminaries

2.1. **Homogeneous presentations and monoids.** Let M be a monoid presented by $\langle A \mid \mathcal{R} \rangle$. If $w, w' \in A^*$ represent the same element of M, we write $w =_M w'$.

A monoid presentation $\langle A \mid \mathcal{R} \rangle$ is *homogeneous* if $|u| = |v|$ for all $(u, v) \in \mathcal{R}$. A monoid is *homogeneous* if it can be defined by a homogeneous presentation.

For a word $u \in A^*$ and a symbol $a \in A$, the number of symbols a in u is denoted $|u|_a$. The *content* of $u \in A^*$ is the function $a \mapsto |u|_a$. Two words $u, v \in A^*$ therefore have the same content if $|u|_a = |v|_a$ for all $a \in A$. A presentation $\langle A \mid \mathcal{R} \rangle$ is *multihomogeneous* if u and v have the same content for all $(u, v) \in \mathcal{R}$. A monoid is *multihomogeneous* if it can be defined by a multihomogeneous presentation. A multihomogeneous presentation is necessarily homogeneous, and thus a multihomogeneous monoid is necessarily homogeneous.

Let $M = \langle A \mid \mathcal{R} \rangle$ be a homogeneous monoid and let $x \in A^*$. Since the two sides of every defining relation in $\mathcal{R}$ have the same length, applying a defining relation does not alter the length of a word. Thus if $y \in A^*$ and $y =_M x$, then $|y| = |x|$.

Furthermore, if A is finite, we can effectively compute the set W_x of all words in A^* equal to x in M, since this set only contains words that are the same length as x. In particular, this means that the word problem is soluble for finitely generated homogeneous monoids: to decide whether x and y are equal in M, simply compute W_x and check whether y is contained in W_x.

2.2. **Rewriting systems.** In this subsection, we recall the basic properties of string rewriting systems needed for this paper. Fort further background reading, see [BO93].

A *string rewriting system*, or simply a *rewriting system*, is a pair $(A, \mathcal{R})$, where A is a finite alphabet and $\mathcal{R}$ is a set of pairs (ℓ, r), often written $\ell \to r$, known as *rewriting rules*, drawn from $A^* \times A^*$. The single reduction relation $\to$ is defined as follows: $u \to_{\mathcal{R}} v$ (where $u, v \in A^*$) if there exists a rewriting rule $(\ell, r) \in \mathcal{R}$ and words $x, y \in A^*$ such that $u = x\ell y$ and $v = xry$. The reduction relation $\to^*$ is the reflexive and transitive closure of $\to$. A word $w \in A^*$ is *reducible* if it contains a subword ℓ that forms the left-hand side of a rewriting rule in $\mathcal{R}$; it is otherwise called *irreducible*.

The string rewriting system $(A, \mathcal{R})$ is *noetherian* if there is no infinite sequence $u_1, u_2, \ldots \in A^*$ such that $u_i \to_{\mathcal{R}} u_{i+1}$ for all $i \in \mathbb{N}$. The rewriting system $(A, \mathcal{R})$ is *confluent* if, for any words $u, u', u'' \in A^*$ with $u \to^* u'$ and $u \to^* u''$, there exists a word $v \in A^*$ such that $u' \to^* v$ and $u'' \to^* v$. A rewriting system is complete if it is both confluent and Noetherian.

Let $(A, \mathcal{R})$ be a complete rewriting system. Then for any word $u \in A^*$, there is a unique irreducible word $v \in A^*$ with $u \to^*_{\mathcal{R}} v$ [BO93, Theorem 1.1.12]. The irreducible words are said to be in *normal form*. These irreducible words form a cross-section of the monoid $\langle A \mid \mathcal{R} \rangle$; thus this monoid may be identified with the set of normal form words under the operation of 'concatenation plus reduction to normal form'.

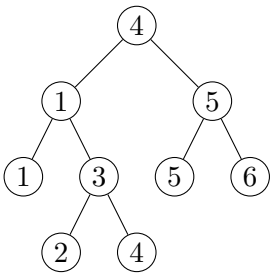


FIGURE 1. Example of a binary search tree T. The root has label 4, so every label in the left subtree of the root is less than or equal to 4 (and indeed the label 4 does occur) and every label in the right subtree of the root is strictly greater than 4. Notice that (for example) $T = \mathcal{BST}(265415314)$, and that $LRP(T) = 124315654$.

3. Sylvester monoids

Let A be the infinite ordered alphabet $\{1 < 2 < \ldots\}$. Let $\mathcal{R}$ be the (infinite) set of defining relations

$$\{(cavb, acvb) : a \leq b < c, v \in A^*\}.$$

Then the *sylvester monoid*, denoted S, is presented by $\langle A \mid \mathcal{R} \rangle$ [HNT05, Definition 8]. Note that the presentation $\langle A \mid \mathcal{R} \rangle$ is multihomogeneous.

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; see the example in Figure 1.

Given a binary search tree $\mathcal{T}$ and a symbol $a \in A$, one inserts a into $\mathcal{T}$ as follows: if $\mathcal{T}$ is empty, create a node and label it a. If $\mathcal{T}$ 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 node. Denote the resulting tree $a \cdot \mathcal{T}$. It is easy to see that $a \cdot \mathcal{T}$ is also a binary search tree.

Given any word $w \in A^*$, define its corresponding binary search tree $\mathcal{BST}(w)$ as follows: start with the empty tree and iteratively insert the symbols in w from right to left; again, see the example in Figure 1.

The left-to-right postfix reading $LRP(\mathcal{T})$ of a binary search tree $\mathcal{T}$ is defined to be the word obtained as follows: recursively perform the left-to-right postfix reading of the left subtree of the root of $\mathcal{T}$, then recursively perform the left-to-right postfix reading of the right subtree of the root of $\mathcal{T}$, then output the label of the root of $\mathcal{T}$; again, see the example in Figure 1. Note that $\mathcal{BST}(LRP(\mathcal{T})) = \mathcal{T}$ [HNT05, Proposition 15].

Proposition 3.1 ([HNT05, Theorem 10])**.** *Let $w, w' \in A^*$. Then $w =_S w'$ if and only if $\mathcal{BST}(w) = \mathcal{BST}(w')$.*

Lemma 3.2. *If two elements of S are o-conjugate, they have the same content.*

Proof. Let $x, y \in A^*$ be such that $x \sim_o y$ in S. Then there exists $g \in A^*$ such that $xg =_S gy$. Let $a \in A$. Since the presentation $\langle A \mid \mathcal{R} \rangle$ is

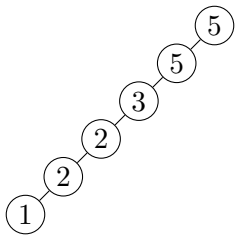


FIGURE 2. Example of a left full binary search tree, in this case $\mathcal{BST}(122355)$.

multihomogeneous, $|xg|_a = |gy|_a$. Thus $|x|_a + |g|_a = |g|_a + |y|_a$, and so $|x|_a = |y|_a$. Since $a \in A$ was arbitrary, x and y have the same content. □

Call a binary search tree *left full* if all of its nodes have empty right subtrees. That is, a binary search tree is left full if every node is a left child of its parent node. (See Figure 2.) An element $x \in S$ is *left full* if $\mathcal{BST}(x)$ is left full.

If $\mathcal{T}$ is a left full binary search tree, then by the definition of the left-to-right postfix reading,

$$LRP(\mathcal{T}) = 1^{k_1} 2^{k_2} \cdots m^{k_m}$$

for some $m \in \mathbb{N} \cup \{0\}$ and $k_i \in \mathbb{N} \cup \{0\}$. It is thus clear that there is exactly one left full element with each content.

Lemma 3.3. *If two elements of S have the same content, they are $\sim_p^*$-conjugate.*

Proof. Since $\sim_p^*$ is an equivalence relation, it will suffice to prove that every element of S is $\sim_p^*$-related to the left full element with the same content.

Consider a binary search tree $\mathcal{T}$. Starting from the root, follow left child nodes until a node is found that has no left child. Call this node p_1; this will be the minimal node of the binary search tree. Let $p_2, \ldots, p_m$ be the successive ancestor nodes of p_1 (that is, p_{i+1} is the parent of p_i for $i = 1, \ldots, m-1$, and p_m is the root node). We define the *left branch length* of $\mathcal{T}$ to be the maximal k such that none of $p_1, \ldots, p_k$ have right subtrees. Note that p_1 may have a right subtree; in this case $\mathcal{T}$ has left branch length 0. Note further that $\mathcal{T}$ is left full if and only if its left branch length is equal to the number of nodes in $\mathcal{T}$, or equivalently equal to $|LRP(\mathcal{T})|$. In general, the left branch length of $\mathcal{T}$ is bounded above by $|LRP(\mathcal{T})|$.

Let $h \in \mathbb{N}$. We will prove the result for elements of S of length h using reverse induction (from h down to 0) on the left branch length of the corresponding binary search trees. Let $x \in S$ be such that $|x| = h$ and the left branch length of $\mathcal{BST}(x)$ is $|x|$. Then, as noted above, $\mathcal{BST}(x)$ is a left full binary search tree, and so x is a left full element and there is nothing to prove. This is the base case of the induction.

For the induction step, let $x \in S$ be such that $|x| = h$ and the left branch length k of $\mathcal{BST}(x)$ is strictly less than $|x|$, and assume that every element of length h whose binary search tree has left branch length strictly greater than k is $\sim_p^*$-related to the left full element with the same content.

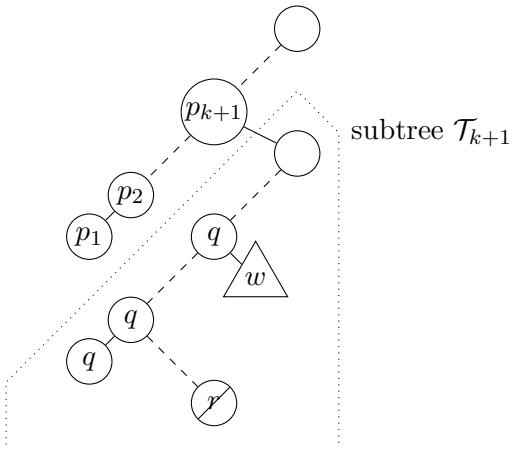


FIGURE 3. The locations of $p_1, \ldots, p_{k+1}$, the nodes q, and the subtree with left-to-right postfix reading u. There is no vertex r in the right subtree of any node q below the uppermost.

Let $\mathcal{T} = \mathcal{BST}(x)$, and let the $p_1, \ldots, p_k, \ldots, p_m$ be as in the discussion above. Since $\mathcal{T}$ is not left full, k (which is the left branch length of $\mathcal{T}$) is strictly less than h, and p_{k+1} has a right subtree, which we denote by $\mathcal{T}_{k+1}$ for future reference. (See Figure 3.) Note that $p_1 \leq p_2 \leq \ldots \leq p_k$.

So $LRP(\mathcal{T})$ is of the form $p_1 p_2 \cdots p_k u p_{k+1} v$, where u is the left-to-right postfix reading of the non-empty right subtree $\mathcal{T}_{k+1}$ of p_{k+1}, and v is the left-to-right postfix reading of the remainder of the tree above and to the right of p_{k+1} (and which will include $p_{k+2}, \ldots, p_m$ in that order, but not necessarily consecutively). Notice that if $m = k+1$, then p_{k+1} is the root note and $v = \varepsilon$. On the other hand, if $m > k+1$, then every symbol of the right subtree of p_{k+1} is greater than p_{k+1} and less than or equal to p_{k+2}. Thus we deduce that $p_{k+1} < p_{k+2}$ and therefore p_{k+1} is less than every symbol in v. Therefore in both cases p_{k+1} is less than every symbol in v.

Starting from the root of $\mathcal{T}_{k+1}$, follow left child nodes until a node is found that has no left child. Let q be this node. Follow the successive parent nodes until the uppermost node with value q is located, and let w be the (possibly empty) left-to-right postfix reading of the right subtree of this uppermost q. Suppose there are ℓ such nodes q; note that $\ell \geq 1$. (See Figure 3 again.)

Suppose for *reductio ad absurdum* that one of the $\ell - 1$ nodes q below the uppermost has a right subtree. Let r be some symbol of this right subtree. Then $r > q$, since r is in this right subtree of this q, but $r \leq q$, since r is in the left subtree of the uppermost q; this is a contradiction. Hence only the uppermost q can have a non-empty right subtree.

Therefore $u = LRP(\mathcal{T}_{k+1}) = q^{\ell-1} w q u'$, where u' is the left-to-right postfix reading of the remainder of the $\mathcal{T}_{k+1}$ above and to the right of the uppermost q. Notice that $p_{k+1} < q$, since q is on the right subtree $\mathcal{T}_{k+1}$ of p_{k+1}. Similarly, q is less than every symbol of w. By the choice of the uppermost node with value q each symbol of u' is strictly greater than q.

As noted above, if $m = k+1$ then $v = \varepsilon$ and so vacuously q is less than or equal to every symbol in v. If, on the other hand, $m > k+1$, then q is

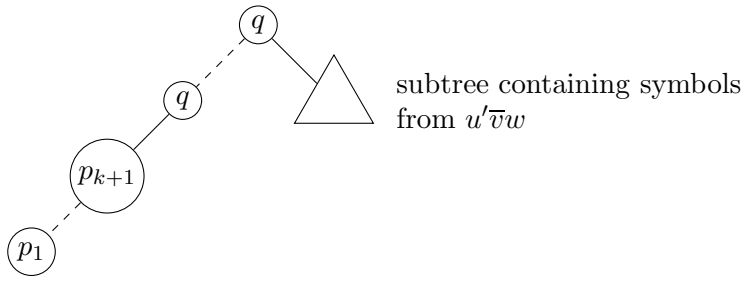


FIGURE 4. Result of computing $\mathcal{BST}(u'\overline{v}wp_1p_2\cdots p_kp_{k+1}q^\ell)$.

in the left subtree of p_{k+2} and we have $p_{k+1} < q \leq p_{k+2}$ and hence all the nodes above and to the right of the node p_{k+1} have value greater or equal than q: that is, q is less than or equal to every symbol in v.

Assume that v contains s instances of the symbol q. Let $\overline{v}$ denote the word obtained from v by deleting the s symbols q (and keeping the remaining symbols in the same order).

Thus we have

$$\begin{aligned}
x &= p_1p_2\cdots p_kq^{\ell-1}wqu'p_{k+1}v \\
&=_S q^{\ell-1}wqu'p_1p_2\cdots p_kp_{k+1}v \\
&\qquad [\text{by multiple applications of } \mathcal{R} \text{ with } a \text{ being the } p_i, \\
&\qquad\quad b = p_{k+1}, \text{ and } c \text{ being letters of } q^{\ell-1}wqu'] \\
&\sim_p u'p_1p_2\cdots p_kp_{k+1}vq^{\ell-1}wq \\
&=_S u'\overline{v}p_1p_2\cdots p_kp_{k+1}q^sq^{\ell-1}wq \\
&\qquad [\text{by multiple applications of } \mathcal{R} \text{ with } a \text{ being the } p_i \text{ or the } q, \\
&\qquad\quad b = q, \text{ and } c \text{ being letters of } v] \\
&=_S u'\overline{v}wp_1p_2\cdots p_kp_{k+1}q^sq^\ell \\
&\qquad [\text{by multiple applications of } \mathcal{R} \text{ with } a \text{ being the } p_i \text{ or the } q, \\
&\qquad\quad b = q, \text{ and } c \text{ being letters of } w]
\end{aligned}$$

[Note that we always have $a \leq b < c$ as required by the defintion of $\mathcal{R}$.]

When we apply the insertion algorithm to compute $\mathcal{BST}(u'\overline{v}wp_1p_2\cdots p_kp_{k+1}q^sq^\ell)$, the $s+\ell$ symbols q are inserted first, followed by the symbols $p_{k+1}, p_k, \ldots, p_1$. So the root node is q, and, since $q > p_{k+1} \geq p_k \geq \ldots \geq p_1 \geq p_1$, these symbols are always inserted as left child nodes of at the leftmost node in the tree.

Since all of the symbols in $u'vw$ are strictly greater than q, they are all inserted into the right subtree of the root node q. (See Figure 4.) So the left branch length of $\mathcal{BST}(u'\overline{v}wp_1p_2\cdots p_kp_{k+1}q^sq^\ell)$ is $(k+1)+(s+\ell-1)$, which is greater than k since $\ell \geq 1$. Hence, by the induction hypothesis, $u'\overline{v}wp_1p_2\cdots p_kp_{k+1}q^sq^\ell$ is $\sim_p^*$-related to the left full element with the same content as $u'\overline{v}wp_1p_2\cdots p_kp_{k+1}q^sq^\ell$, which has the same content as x. By transitivity, x is $\sim_p^*$-related to the left full element with the same content as x. Thus completes the induction step and thus the proof. □

Theorem 3.4. *In the sylvester monoid $\langle A \mid \mathcal{R} \rangle$, two elements are o-conjugate if and only if they have the same content (when viewed as words). Moreover, o-conjugacy is decidable for the sylvester monoid and $\sim_o = \sim_p^*$.*

Proof. This is immediate from Lemmata 3.2 and 3.3 and the fact that $\sim_p^* \subseteq \sim_o$. □

4. Homogeneous monoids

4.1. **Decidability of $\sim_p^*$-conjugacy.** It is easy to see that $\sim_p^*$-conjugacy is decidable for finitely generated homogeneous monoids, as follows: Let $M = \langle A \mid \mathcal{R} \rangle$ be a homogeneous monoid with A finite and let $x, y \in A^*$. We can compute the set P_x of all words $\sim_p^*$-conjugate to x by setting $Y_0 = \{x\}$ and then iteratively computing Y_k to be the set of words of the form vu for some $uv \in W_t$ for some $t \in Y_{k-1}$. Since W_t is effectively computable (as discussed in Subsection 2.1), computing each Y_k is effective. Since every set in the sequence $Y_0 \subseteq Y_1 \subseteq \dots$ is contained in the finite set $A^{|x|}$, we must have $Y_k = Y_{k+1} = \dots = P_x$ for some k. Hence computing P_x is effective.

In particular, this means that the $\sim_p^*$-conjugacy problem is soluble for homogeneous monoids: to decide whether x and y are $\sim_p^*$-conjugate, simply compute P_x and check whether y is contained in P_x.

4.2. **Undecidability of o-conjugacy.** We are now going to prove that o-conjugacy is undecidable for homogeneous monoids. We will extend this to multihomogeneous monoids in the next subsection. This section assumes familiarity with Turing machines and undecidability; see [HU79, Chs 7–8] for background.

Theorem 4.1. *There exists a finitely presented homogeneous monoid $M = \langle A \mid \mathcal{R} \rangle$ in which the o-conjugacy problem is undecidable. That is, there is no algorithm that takes as input two words over A and decides whether they are o-conjugate.*

Let us outline the general strategy of the proof before beginning. The aim is to take a Turing machine with undecidable halting problem, and build a finite complete rewriting system that simulates its computation. The rewriting system will present a homogeneous monoid in such a way that the Turing machine halts on a given input w if and only if the element corresponding to the initial configuration of the machine with input w is o-conjugate to the element corresponding to a given halting configuration. However, the definition of the rewriting system is complicated by our need to build a *homogeneous* presentation.

Proof. Let $\mathfrak{M} = (\Sigma, Q, \delta, q_0, q_a)$ be a deterministic Turing machine with undecidable halting problem, where Σ is the tape alphabet (which contains a blank symbol b), Q is the state set, δ is the transition (partial) function, q_0 is the initial state, and q_a is the unique halting state.

For our purposes, a *configuration* of the Turing machine $\mathfrak{M}$ is a word uqv with $u, v \in (\Sigma \cup \{b\})^*$ and $q \in Q$. The symbol q records the current state of the machine, the word uv records the contents of all cells of the tape that were either initially non-blank or which have been visited by the read/write head of the Turing machine, and the read/write head is currently pointing

at the cell corresponding to the leftmost symbol of v. (Thus if the leftmost cell visited by read/write head is now blank, then u will begin with b. This is non-standard: normally, a Turing machine configuration records only the smallest part of the tape that includes all non-blank cells and the current position of the read/write head. Thus one normally assumes that u begins with a symbol from Σ and v ends with a symbol from Σ.)

We will make some assumptions about $\mathfrak{M}$; it is easy to see that any deterministic Turing machine can be modified to give an equivalent one satisfying these assumptions:

- $\mathfrak{M}$ either changes the symbol on the tape where the read/write head is positioned, or moves the head left or right. (That is, it does not change the symbol *and* move the head.) Thus we view the transition function as being a map from $Q \times \Sigma$ to $Q \times (\Sigma \cup \{L, R\})$.
- $\mathfrak{M}$ has no defined computation after entering the halting state q_a. That is, δ is undefined on $\{q_a\} \times \Sigma$.
- $\mathfrak{M}$ stores its initial tape contents in some part of the tape and, immediately before entering the halting state q_a, erases all symbols from the tape except this original content and moves to the left of the non-blank part of the tape. So $w \in L(\mathfrak{M})$ if and only if $q_0 w \vdash^* b^\alpha q_a w b^\beta$ for some α, β.

Let Σ' be a set in bijection with Σ under the map $x \mapsto x'$. Let $A = \Sigma \cup \Sigma' \cup Q \cup \{h, h', d, z\}$; we are going to define a rewriting system $(A, \mathcal{R})$. Let $\mathcal{R}$ consist of the following rewriting rules, for all $q_i \in Q$, $s_i, s_j \in \Sigma$ and $x, y \in \Sigma \cup \{b\}$:

Group I (for $q_i, q_j \in Q$, $s_\ell \in \Sigma$ and $x, y \in \Sigma \cup \{b\}$):

$$q_i x d \to d q_j s_\ell \qquad \text{where } (q_i, x)\delta = (q_j, s_\ell), \tag{4.1}$$

$$q_i x d h \to d^2 q_j h \qquad \text{where } (q_i, x)\delta = (q_j, b), \tag{4.2}$$

$$q_i x d y \to d q_j b y \qquad \text{where } (q_i, x)\delta = (q_j, b), \tag{4.3}$$

$$q_i h d \to q_j s_\ell h \qquad \text{where } (q_i, b)\delta = (q_j, s_\ell), \tag{4.4}$$

$$q_i h d \to d q_j h \qquad \text{where } (q_i, b)\delta = (q_j, b), \tag{4.5}$$

$$q_i x d \to d x' q_j \qquad \text{where } (q_i, x)\delta = (q_j, R), \tag{4.6}$$

$$q_i h d \to b' q_j h \qquad \text{where } (q_i, b)\delta = (q_j, R), \tag{4.7}$$

$$y' q_i x d \to d q_j y x \qquad \text{where } (q_i, x)\delta = (q_j, L), \tag{4.8}$$

$$s'_\ell q_i h d \to d q_j s_\ell h \qquad \text{where } (q_i, b)\delta = (q_j, L), \tag{4.9}$$

$$b' q_i h d \to d^2 q_j h \qquad \text{where } (q_i, b)\delta = (q_j, L), \tag{4.10}$$

$$h' q_i x d \to h' q_j b x \qquad \text{where } (q_i, x)\delta = (q_j, L), \tag{4.11}$$

$$h' q_i h d \to d h' q_j h \qquad \text{where } (q_i, b)\delta = (q_j, L). \tag{4.12}$$

Group II (for $s_i \in \Sigma$):

$$b' q_a \to d q_a, \tag{4.13}$$

$$q_a s_i d \to d q_a s_i. \tag{4.14}$$

Group III (for $s_i \in \Sigma$ and $x, y \in \Sigma \cup \{b\}$):

$$xyd \to xdy, \tag{4.15}$$

$$s_i hd \to s_i dh, \tag{4.16}$$

$$x'd \to dx', \tag{4.17}$$

$$h'd \to dh'. \tag{4.18}$$

Group IV (for $s_i, s_j \in \Sigma$):

$$s_i hz \to s_i zh, \tag{4.19}$$

$$s_i s_j z \to s_i z s_j, \tag{4.20}$$

$$h' q_a s_i z \to z h' q_0 s_i. \tag{4.21}$$

Let $M = \langle A \mid \mathcal{R} \rangle$. Notice that $\langle A \mid \mathcal{R} \rangle$ is a homogeneous presentation, and so M is homogeneous. We will now proceed to show that the rewriting system $(A, \mathcal{R})$ is complete (Lemmata 4.2 and 4.3). We will then show how the halting problem for $\mathfrak{M}$ reduces to the o-conjugacy problem for M.

Lemma 4.2. *The rewriting system $(A, \mathcal{R})$ is noetherian.*

Proof. Let $\leq$ be any partial order on the alphabet A satisfying

$$\begin{aligned} &z < d < b < b' < s_i < h < q_k < s_j' < h' && \text{for } s_i, s_j \in \Sigma \text{ and } q_k \in Q, \\ &q_i = q_j && \text{for } q_i, q_j \in Q. \end{aligned}$$

Let $\leq_{\text{lenlex}}$ be the partial length-plus-lexicographic order induced by $\leq$ on A^*.

By inspection, applying any rewriting rule in $\mathcal{R}$ results in a strict reduction with respect to $\leq_{\text{lenlex}}$. Since there are no infinite descending chains in the partial length-plus-lexicographic order, any process of rewriting using $\mathcal{R}$ must terminate. Hence $(A, \mathcal{R})$ is noetherian. □

Lemma 4.3. *The rewriting system $(A, \mathcal{R})$ is locally confluent.*

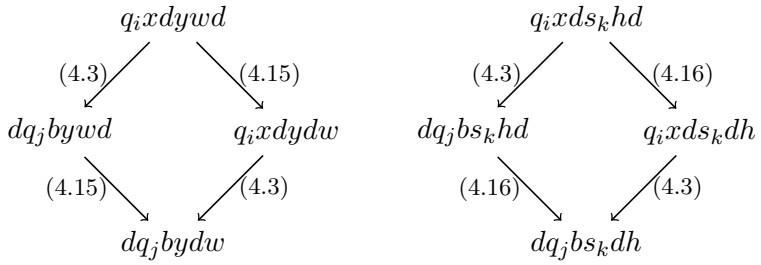
Proof. We will systematically examine possible overlaps of left-hand sides of rules in $\mathcal{R}$ and show that they resolve.

Group I–Group I overlaps. No non-trivial overlaps. This is because symbols from Q occur only once in these left-hand sides, and exact overlaps of left-hand sides do not occur between rules of different types because $\mathfrak{M}$ is deterministic and so every left-hand side determines a unique right-hand side.

Group I–Group II overlaps. No non-trivial overlaps. This is because symbols q_a (the halting state of $\mathfrak{M}$) do not occur on the left-hand side of any Group I rule, and symbols b' and d do not occur at the start of left-hand sides of Group I rules.

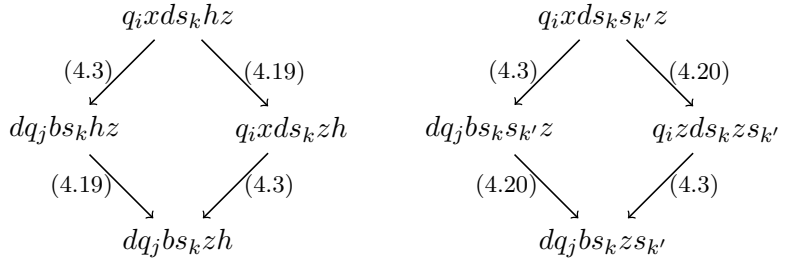
Group I–Group III overlaps. Two possible overlaps, between the left-hand sides of (4.3) and (4.15), and between the left-hand sides of (4.3) and (4.16),

both of which resolve:



(Note that the symbols x and y in rules of type (4.15) and (4.16) have been renamed here to y and w, to avoid conflicting with the x in rules of type (4.3). Similarly, the subscript i in rules of type (4.16) has been renamed to k to avoid conflicting with the subscript i in rules of type (4.3).) These are the only possible overlaps because the symbols d only appear once in each left-hand side, and in rules in Group III are preceded by two symbols in $\Sigma \cup \{b\}$, which never happens in rules in Group I. So overlaps cannot involve symbols d. The only remaining possibilities are overlaps involving symbols to the right of d in Group I rules, which happens precisely in the cases we consider.

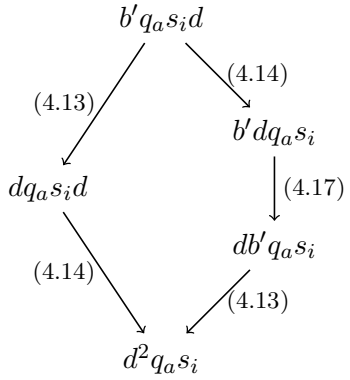
Group I–Group IV overlaps. Two possible overlaps, between the left-hand sides of (4.3) and (4.19), and between the left-hand sides of (4.3) and (4.20), both of which resolve:



(Note that the subscript j in rules of type (4.20) has been renamed here to k', to avoid conflicting with the subscript j in rules of type (4.3).) These are the only possible overlaps since symbols z do not appear in rules in Group I and symbols d do not occur in rules in Group III. So overlaps cannot involve symbols z. The only remaining possibilities are overlaps involving symbols to the right of d in Group I rules, which happens precisely in the cases we consider.

Group II–Group II overlaps. By inspection, the only possible overlaps are between the left-hand sides of rules of type (4.13) and type (4.14), and these

overlaps resolve:



Group II–Group III overlaps. There are no overlaps between the left-hand side of a rule of type (4.13) and a Group III rule, because q_a does not appear in left-hand sides of Group III rules, and no left-hand side of a Group III rule ends in b'. There are no overlaps between left-hand side of a rule of type (4.14) and a Group III rule since if d is involved in the overlap, the preceding letter of the Group III rule must also be involved, which limits us to rules of type (4.15); this is a contradiction since in rules of type (4.15) the symbol d is preceded by two symbols from $\Sigma \cup \{b\}$, not the symbols q_a from the rule of type (4.13).

Group II–Group IV overlaps. There are no overlaps between the left-hand sides of a Group II rule and a Group IV rule since the left hand sides of Group II rules start and end with symbols b', q_a, and d, and only q_a appears in the left-hand side of a Group IV rule (in type (4.21) rules), but between symbols h' and s_iz, and h' and z do not appear on left-hand sides of Group II rules.

Group III–Group III overlaps. There are no overlaps between left-hand sides of Group III rules, since they all end in symbols d that appear nowhere else in the left-hand sides.

Group III–Group IV overlaps. There are no overlaps between left-hand sides of Group III rules and Group IV rules, since all left-hand sides of Group III end in symbols d, which do not appear on left-hand sides of Group IV rules, and left-hand sides of Group IV rules end in symbols z, which do not appear on left-hand sides of Group III rules.

Group IV–Group IV overlaps. There are no overlaps between left-hand sides of Group IV rules, since they all end in symbols z that appear nowhere else in the left-hand sides.

Since all overlaps resolve, the rewriting system $(A, \mathcal{R})$ is locally confluent. □

By Lemmata 4.2 and 4.3, the rewriting system $(A, \mathcal{R})$ is complete.

Lemma 4.4. *Let $u_1q_1v_1$ and $u_2q_2v_2$ be configurations of $\mathfrak{M}$. Let $\hat{v}_1$ and $\hat{v}_2$ be such that $v_1 = \hat{v}_1b^{\alpha_1}$ and $v_2 = \hat{v}_2b^{\alpha_2}$, where α_1 and α_2 are maximal (possibly 0). Then $u_1q_1v_1 \vdash u_2q_2v_2$ if and only if*

$$h'u_1'q_1\hat{v}_1hd \to^* d^{1+|u_1|-|u_2|+|\hat{v}_1|-|\hat{v}_2|}h'u_2'q_2\hat{v}_2h.$$

Furthermore, in this case, $1 + |u_1| - |u_2| + |\hat{v}_1| - |\hat{v}_2|$ is either 0, 1, or 2.

Proof. This is essentially a case-by-case checking using Group I and Group III rules in $\mathcal{R}$. Essentially, the symbol d moves to the left of the h using a rule of type (4.16) (note that by definition $\hat{v}_1$ does not end in a symbol b and thus ends with a symbol s_i if it is non-empty) and then using rules (4.15) until it is one symbol to the right of q_1. Then a rewrite using a Group I rule occurs. (If $\hat{v}_1$ is the empty word, so that q_i and h are adjacent, the symbol d does not move left: rules (4.4), (4.5), (4.7), (4.9), or (4.10) apply immediately.) This produces 0, 1, or 2 symbols d which then move leftwards using rules (4.17) and (4.18). Note that

- Rules (4.1), (4.3), (4.5), and (4.12) correspond to cases where $|u_1| = |u_2|$ and $|\hat{v}_1| = |\hat{v}_2|$. The first three rules correspond to cases where the read/write head does not move and a blank does not replace a non-blank symbol at the extreme right of the non-blank portion of the tape. The last rule is simply a special case to cover the case when the entire tape is blank and the read/write head is moved left, which is not covered by over rules.
- The rule (4.6) corresponds to cases where $|u_1|+1 = |u_2|$ and $|\hat{v}_1|-1 = |\hat{v}_2|$. This is the case where the read/write head moves to the right somewhere in the non-blank part of the tape.
- Rules (4.8) and (4.9) correspond to cases where $|u_1| - 1 = |u_2|$ and $|\hat{v}_1| + 1 = |\hat{v}_2|$. This is the case where the read/write head moves to the left somewhere in the non-blank part of the tape.
- The rule (4.7) corresponds to cases where $|u_1| + 1 = |u_2|$ and $|\hat{v}_1| = |\hat{v}_2|$. This is the case where the read/write head is pointing to a blank symbol somewhere to the right of the non-blank part of the tape, and then moves further right.
- The rule (4.10) corresponds to cases where $|u_1| - 1 = |u_2|$ and $|\hat{v}_1| = |\hat{v}_2|$. This is the case where the read/write head is pointing to a blank symbol at least one symbol to the right of the non-blank part of the tape, and then moves left.
- Rules (4.4) and (4.11) corresponds to cases where $|u_1| = |u_2|$ and $|\hat{v}_1|+1 = |\hat{v}_2|$. The first rule is where the read/write head is pointing either to the first blank symbol to the right of the non-blank part of the tape, or to some blank symbol further right, and replaces it with a non-blank symbol. The second rule is where the read/write head is pointing either to the leftmost non-blank symbol or some blank symbol, then moves further left.
- The rule (4.2) corresponds to cases where $|u_1| = |u_2|$ and $|\hat{v}_1| - 1 = |\hat{v}_2|$. This rule is where the read/write head is pointing to the rightmost non-blank symbol on the tape, and replaces it with a blank.

□

Lemma 4.5. *Using notation from Lemma 4.4, $u_1 q_1 v_1 \vdash^* u_2 q_2 v_2$ if and only if there is some natural γ such that*

$$h' u_1' q_1 \hat{v}_1 h d^\gamma \to^* d^{\gamma+|u_1|-|u_2|+|\hat{v}_1|-|\hat{v}_2|} h' u_2' q_2 \hat{v}_2 h.$$

In particular, $q_0 w \vdash^* (b')^\alpha q_a w b^\beta$ *if and only if there is some natural* γ *such that*

$$h' q_0 w h d^\gamma \to^* d^{\gamma-\alpha} h' (b')^\alpha q_a w h \to^* d^\gamma h' q_a w h. \tag{4.22}$$

Proof. The first part follows by induction using Lemma 4.4. In (4.22), the first reduction is simply a particular case of the first part. The second reduction uses rules (4.13), (4.17), and (4.18). □

We have now established the correspondence between rewriting using $(A, \mathcal{R})$ and computation in $\mathfrak{M}$. We can now prove the connection with o-conjugacy:

Lemma 4.6. *For* $w \in \Sigma^+$*, we have* $w \in L(\mathfrak{M})$ *if and only* $h' q_0 w h \sim_o h' q_a w h$.

Proof. Suppose $w \in L(\mathfrak{M})$. Then $q_0 w \vdash^* q_a w$. So by Lemma 4.5, $h' q_0 w h d^\alpha \to^* d^\alpha h' q_a w h$ for some natural α. Furthermore, using Group IV rules, $h' q_a w h z \to^* z h' q_0 w h$. Hence $h' q_0 w h \sim_o h' q_a w h$.

Suppose now that $h' q_0 w h \sim_o h' q_a w h$. Then there exists a word $u \in \Sigma^*$ such that $h' q_0 w h u \leftrightarrow^* u h' q_a w h$. Assume without loss that u is irreducible. First note that $u h' q_a w h$ is irreducible, since no rule in $\mathcal{R}$ can be applied to $h' q_a w h$ since no symbols d, z, or b' are present, and there is no rule whose left-hand side contains h' except at the start. Hence $h' q_0 w h u \to^* u h' q_a w h$.

Suppose first u contains a symbol z. Then u factors as $u_1 z u_2$ where u_1 does not contain z. The word $h' q_0 w h$ is irreducible since no symbols symbols d, z, or b' are present, so rewriting $h' q_0 w h u$ must begin with a rewriting rule that includes the distinguished symbol h and a non-empty prefix of u. In rules in $\mathcal{R}$, the symbol h is only followed by a symbol d or z. Since u_1 does not include z, the first symbol of u_1 must therefore be d. As in Lemma 4.4, this d will move to the left, where it will either disappear, or one or two symbols d will emerge to the left of the h'. The h remains next to the remainder of u_1, so by the same reasoning the next symbol of u_1 must also be d. Repeating this reasoning, we see $u_1 = d^\alpha$ and $h' q_0 w h u = h' q_0 w h d^\alpha z u_2 \to^* d^\beta h' \cdots q_c \cdots h z u_2$ for some non-negative integers α and β. Notice that $d^\beta h' \cdots q_c \cdots h$ is irreducible, so further rewriting must use a rule of type (4.19) followed by rules of type (4.20) to move the z to the left until it is one symbol to the right of q_c. To apply the rule (4.21), which is necessary if we want to obtain $u h' q_0 w h$, where all the symbols z are to the left of the symbol from Q, is only possible if $q_c = q_a$. By Lemma 4.5, this implies that reading w causes $\mathfrak{M}$ to enter the halting state q_a; and hence w is accepted by $L(\mathfrak{M})$.

If u does not contain a symbol z, then by the same reasoning as in the last paragraph, $u = d^\alpha$ and so $h q_0 w h' d^\alpha \to^* d^\alpha h q_a w h'$. So $q_0 w \vdash^* q_a w$ by Lemma 4.5 and so $w \in L(\mathfrak{M})$. □

By Lemma 4.6, the problem of whether $\mathfrak{M}$ halts on a given input reduces to the o-conjugacy problem for M. Hence the o-conjugacy problem for M is undecidable. □

Since $\sim_p^*$ is decidable for homogeneous monoids and $\sim_o$ is undecidable in general for homogeneous monoids by Theorem 4.1, we have (very indirectly) proved the following corollary:

Corollary 4.7. *In the class of homogeneous monoids, $\sim_p^*$ and $\sim_o$ do not coincide in general.*

4.3. **Multihomogeneous monoids.**

Theorem 4.8. *There exists a finitely presented multihomogeneous monoid $N = \langle X \mid \mathcal{S} \rangle$ in which the o-conjugacy problem is undecidable. That is, there is no algorithm that takes as input two words over X and decides whether they are o-related.*

Proof. Let $M = \langle A \mid \mathcal{R} \rangle$ be a finitely presented homogeneous monoid with undecidable o-conjugacy problem; such a monoid exists by Theorem 4.1. Suppose $A = \{a_1, \ldots, a_n\}$. Let $X = \{x, y\}$. Define a map

$$\phi : A^* \to X^*; \qquad a_i \mapsto x^2 y^i x y^{n-i+1}.$$

Let

$$\mathcal{S} = \mathcal{R}\phi = \big\{(u\phi, v\phi) : (u, v) \in \mathcal{R}\big\}.$$

Let $N = \langle X \mid \mathcal{S} \rangle$. Then $\langle X \mid \mathcal{S} \rangle$ is a multihomogeneous presentation, and thus N is a multihomogeneous monoid: the proof is not difficult; see [CGM, Proposition 5.8] for details. Furthermore, it is easy to prove that ϕ embeds M into N [CGM, Proposition 5.9].

Suppose $p, q \in M$ and $w \in N$ are such that $(p\phi)w =_N w(q\phi)$. Since $(p\phi)w$ contains a prefix of length $|p\phi|$ that is in $\operatorname{im}\phi$, a prefix of $w(q\phi)$ of length $|p\phi|$ is in $\operatorname{im}\phi$ since application of relations in $\mathcal{S}$ preserves subwords that lie in $\operatorname{im}\phi$; see [CGM, Proof of Proposition 5.9] for further details. Hence $(p\phi)w$ contains a prefix of length $2|p\phi|$ in $\operatorname{im}\phi$, and so $w(q\phi)$ contains a prefix of length $2|p\phi|$ in $\operatorname{im}\phi$ by the same lemma. Iterating this process, eventually we see that $w(q\phi)$ contains a prefix of length at least $|w|+2$ that lies in $\operatorname{im}\phi$. So suppose $v = v_1 v_2 \cdots v_m \in A^*$ (where $v_i \in A$) is such that $v\phi$ is a prefix of $w(q\phi)$ of length at least $|w| + 2$. The subword $q\phi$ begins with x^2, and so $w(q\phi)$ beings with wx^2. Since $|v\phi| \geq |w|+2$, this x^2 must be in the prefix $v\phi$ of $w(q\phi)$. Subwords x^2 only occur in $v\phi$ at the start of the subwords $v_i\phi$, and so $w = (v_1 \cdots v_{m'})\phi$ for some $m' < m$. Thus $(pv_1 \cdots v_{m'})\phi =_N (v_1 \cdots v_{m'} q)\phi$. Since ϕ is an embedding, $pv_1 \cdots v_{m'} =_M v_1 \cdots v_{m'} q$.

Therefore, o-conjugacy on $\operatorname{im}\phi$ is simply o-conjugacy in N restricted to $\operatorname{im}\phi$. Since o-conjugacy is undecidable for M and thus for $\operatorname{im}\phi$, it is therefore undecidable for the multihomogeneous monoid N. □

References

[AKM14] J. Araújo, J. Konieczny, & A. Malheiro. 'Conjugation in semigroups'. *J. Algebra*, 403 (2014), pp. 93–134. DOI: 10.1016/j.jalgebra.2013.12.025.

[BO93] R. V. Book & F. Otto. *String-Rewriting Systems*. Texts and Monographs in Computer Science. Springer-Verlag, New York, 1993.

[CEK+01] J. Cassaigne, M. Espie, D. Krob, J. C. Novelli, & F. Hivert. 'The Chinese monoid'. *Internat. J. Algebra Comput.*, 11, no. 3 (2001), pp. 301–334. DOI: 10.1142/S0218196701000425.

[CGM] A. J. Cain, R. D. Gray, & A. Malheiro. 'On finite complete rewriting systems, finite derivation type, and automaticity for homogeneous monoids'. arXiv: 1407.7428.

[CGM15] A. J. Cain, R. D. Gray, & A. Malheiro. 'Rewriting systems and biautomatic structures for Chinese, hypoplactic, and sylvester monoids'. *Internat. J. Algebra Comput.*, 25, no. 1-2 (2015). DOI: 10.1142/S0218196715400044.

[Cho93] C. Choffrut. 'Conjugacy in free inverse monoids'. *Internat. J. Algebra Comput.*, 3, no. 2 (1993), pp. 169–188. DOI: 10.1142/S0218196793000135.

[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.

[HU79] J. E. Hopcroft & J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison–Wesley Publishing Co., Reading, Mass., 1979.

[KM07] G. Kudryavtseva & V. Mazorchuk. 'On conjugation in some transformation and Brauer-type semigroups'. *Publ. Math. Debrecen*, 70, no. 1-2 (2007), pp. 19–43.

[KM09] G. Kudryavtseva & V. Mazorchuk. 'On three approaches to conjugacy in semigroups'. *Semigroup Forum*, 78, no. 1 (2009), pp. 14–20. DOI: 10.1007/s00233-008-9047-7.

[Lal79] G. Lallement. *Semigroups and Combinatorial Applications*. John Wiley & Sons, New York-Chichester-Brisbane, 1979.

[Lot02] M. Lothaire. *Algebraic combinatorics on words*, vol. 90 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2002.

[LR98] J. L. Loday & M. O. Ronco. 'Hopf algebra of the planar binary trees'. *Adv. Math.*, 139, no. 2 (1998), pp. 293–309. DOI: 10.1006/aima.1998.1759.

[LS81] A. Lascoux & M. P. Schützenberger. 'Le monoïde plaxique'. In *Noncommutative structures in algebra and geometric combinatorics (Naples, 1978)*, vol. 109 of *Quad. "Ricerca Sci."*, pp. 129–156. CNR, Rome, 1981.

[NO85] P. Narendran & F. Otto. 'Complexity results on the conjugacy problem for monoids'. *Theoret. Comput. Sci.*, 35, no. 2-3 (1985), pp. 227–243. DOI: 10.1016/0304-3975(85)90016-7.

[NO86] P. Narendran & F. Otto. 'The problems of cyclic equality and conjugacy for finite complete rewriting systems'. *Theoret. Comput. Sci.*, 47, no. 1 (1986), pp. 27–38. DOI: 10.1016/0304-3975(86)90131-3.

[Ott84] F. Otto. 'Conjugacy in monoids with a special Church-Rosser presentation is decidable'. *Semigroup Forum*, 29, no. 1-2 (1984), pp. 223–240. DOI: 10.1007/BF02573327.

[Sil96] P. V. Silva. 'Conjugacy and transposition for inverse monoid presentations'. *Internat. J. Algebra Comput.*, 6, no. 5 (1996), pp. 607–622. DOI: 10.1142/S0218196796000349.

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
E-mail 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

Centro de Álgebra da Universidade de Lisboa, Av. Prof. Gama Pinto 2, 1649–003 Lisboa, Portugal
E-mail address: ajm@fct.unl.pt