Word Problem Languages for Free Inverse Monoids

Tara Brough

Word Problem Languages for Free Inverse Monoids

Tara Brough *

Centro de Matemática e Aplicações, Faculdade de Ciências e Tecnologia,
Universidade Nova de Lisboa, 2829–516 Caparica, Portugal
t.brough@fct.unl.pt

Abstract. This paper considers the word problem for free inverse monoids of finite rank from a language theory perspective. It is shown that no free inverse monoid has context-free word problem; that the word problem of the free inverse monoid of rank 1 is both 2-context-free (an intersection of two context-free languages) and ET0L; that the co-word problem of the free inverse monoid of rank 1 is context-free; and that the word problem of a free inverse monoid of rank greater than 1 is not poly-context-free.

Keywords: Word problems; Co-word problems; Inverse monoids; ET0L languages; Stack automata; Poly-context-free languages

1 Introduction

The word problem of a finitely generated semigroup is, informally, the problem of deciding whether two words over a given finite generating set represent the same element of the semigroup. Although it is undecidable [23], even for finitely presented groups [21, 2], there has been much study (especially for groups) of word problems that are in some sense 'easily' decidable, for example by having low space or time complexity, or being in certain low-complexity language classes.

For groups, the obvious formalisation of the word problem is as the set of all words over the set of generators and their inverses representing the identity element, since two words u and v represent the same element iff uv^{-1} represents the identity. This has been generalised to semigroups in two ways: the first, which we call the *word problem* of a semigroup S with respect to finite generating set A is the set $\mathrm{WP}(S, A) = \{u\#v^{\mathrm{rev}} \mid u =_S v, u, v \in A^+\}$ (where $\#$ is a symbol not in A and v^{rev} denotes the reverse of v); the second, the *two-tape word problem* of S with respect to A, is the relation $\iota(S, A) = \{(u, v) \in A^+ \times A^+ \mid u =_S v\}$. Monoid versions of these are obtained by replacing A^+ with A^*. The word problem has been studied in [7, 12, 13] and the two-tape word problem in [22, 4].

* The author was supported by the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through an FCT post-doctoral fellowship (SFRH/BPD/121469/2016) and the projects UID/Multi/04621/2013 (CEMAT-CIÊNCIAS) and UID/MAT/00297/2013 (Centro de Matemática e Aplicações).

A semigroup S is *inverse* if for every $x \in S$ there is a *unique* $y \in S$ such that $xyx = x$ and $yxy = y$. The classes of inverse semigroups and inverse monoids each form varieties of algebras and hence contain free objects. The free inverse monoid on a set X is denoted $\mathrm{FIM}(X)$; if $|X| = k$ then we also use the notation FIM_k, and k is called the *rank* of FIM_k. All results in this paper are stated for free inverse monoids, but are equally true for free inverse semigroups, since in a free inverse monoid the only representative of the identity is the empty word ϵ.

Word problems of free inverse monoids have already been studied from a time and space complexity perspective: they are recognisable in linear time and in logarithmic space [18]. The aim of this paper is to understand these word problems from a language-theoretic perspective. All free inverse monoid word problems are context-sensitive, since this is equivalent to recognisability in linear space. Our main goal is thus to determine in which subclasses of the context-sensitive languages the free inverse monoid word problem lies. Before summarising the results, we introduce several of the language classes considered. All classes mentioned here are closed under inverse generalised sequential machine mappings, and hence the property of having word problem in any of these classes is closed under change of finite generating set.

The non-closure of the class $\mathcal{CF}$ of context-free languages under complementation and intersection [15] leads naturally to the definition of the classes of *co*$\mathcal{CF}$ and *poly-*$\mathcal{CF}$ languages, being respectively the classes of complements and finite intersections of context-free languages. A language is called k-$\mathcal{CF}$ if it is an intersection of k context-free languages. Groups with co$\mathcal{CF}$ word problem were studied in [14], and groups with poly-context-free word problem in [3]. For groups, having co$\mathcal{CF}$ word problem is equivalent to the *co-word problem* (the complement of the word problem, or abstractly the problem of deciding whether two words represent *different* elements) being context-free. For monoids, we generalise this terminology on the abstract rather than technical level: the complement of the word problem is not an algebraically interesting language, so we define $\mathrm{coWP}(M, X) = \{u\#v^{\mathrm{rev}} \mid u =_M v, u, v \in X^*\}$. The two-tape co-word problem is the complement of the two-tape word problem.

Stack automata, introduced in [10], are a generalisation of pushdown automata that allow the contents of the stack to be examined in 'read-only' mode. They are a special case of the *nested stack automata* introduced slightly later by Aho [1] to recognise indexed languages. The *checking stack languages* are recognised by the more restricted *checking stack automata* [11], in which the stack contents can only be altered prior to commencing reading of the input.

ET0L languages are another subclass of indexed languages, standardly defined by *ET0L-systems*, which are essentially finite collections of 'tables' of context-free-grammar-type productions. These operate similarly to context-free grammars except that at each step in a derivation, productions all from the same 'table' must be applied to *every* nonterminal in the current string (each table is required to have productions from every nonterminal, though these of course may be trivial). The more restricted *EDT0L languages* have the further requirement that in each table of productions there be only one production from each

nonterminal. An automaton model for ET0L languages was given in [25]: it consists of a checking stack with attached push-down stack, operating in such a way that the pointers of the two stacks move together. See [24] (especially Chapter V) for further information on ET0L languages and their many relatives.

In the rank 1 case our goal is achieved fairly comprehensively, with both types of word problem for FIM_1 being shown to be 2-$\mathcal{CF}$ (but not context-free), co-$\mathcal{CF}$ and a checking stack language (and hence ET0L). As far as the author is aware, this is the first known example of a semigroup with ET0L but not context-free word problem. This result is particularly interesting because of the long-standing open problem of whether the indexed languages – of which the ET0L languages form a subclass – give any additional power over context-free languages for recognising word problems of groups [19, 9]. In higher ranks we show that $\mathrm{WP}(\mathrm{FIM}_k)$ for $k \geq 2$ is not poly-$\mathcal{CF}$. We conjecture that the same is true for $\iota(\mathrm{FIM}_k)$, and that neither version of the word problem is co$\mathcal{CF}$ or indexed except in rank 1.

2 Background

2.1 Free inverse monoids

Recall that a monoid M is *inverse* if for every $x \in M$ there is a unique $y \in M$ such that $xyx = x$ and $yxy = y$. The element y is called the *inverse of* x and is usually denoted x^{-1}. In this paper we will also often use the notation $\overline{x}$ for the inverse of x. Given a set X, we use the notation X^{-1} for a set $\{\overline{x} \mid x \in X\}$ of formal inverses for X, and $X^{\pm}$ for $X \cup X^{-1}$. For an element $x \in X^{\pm}$, if $x \in X$ then $x^{-1} = \overline{x}$, while if $x = \overline{y}$ for $y \in X$ then $x^{-1} = y$. We can extend this to define the inverse of a word $w = w_1 \dots w_n$ with $w_i \in X^{\pm}$ by $w^{\mathrm{inv}} = w_n^{-1} \dots w_1^{-1}$.

For any set X, the *free inverse monoid* $\mathrm{FIM}(X)$ on X exists and is given by the monoid presentation

$$\mathrm{FIM}(X) = \langle X^{\pm} \mid u = uu^{\mathrm{inv}}u, uu^{\mathrm{inv}}vv^{\mathrm{inv}} = vv^{\mathrm{inv}}uu^{\mathrm{inv}} \ (u, v \in (X^{\pm})^*)\rangle.$$

This presentation is not very useful for working with the word problem of free inverse monoids. A more powerful tool is given by *Munn trees* [20]: certain labelled directed finite trees that stand in one-to-one correspondence with the elements of $\mathrm{FIM}(X)$, such that the product of two elements can easily be computed using their corresponding trees. To obtain the Munn tree for an element $m \in \mathrm{FIM}(X)$, we use the Cayley graph $\mathcal{G}(F(X), X)$ of the free group $F(X)$. This is a labelled directed tree with $|X|$ edges labelled by the elements of X entering and leaving each vertex. (The Cayley graph also has its vertices labelled by the elements of G, but these are not needed for our purposes.) Given any word $w \in (X^{\pm})^*$ representing m, choose any vertex of $\mathcal{G}(F(X), X)$ as the start vertex and label it α. From vertex α, then trace out the path defined by reading w from left to right, where for $x \in X$, follow the edge labelled x leading *from* the current vertex upon reading x, and follow the edge labelled x leading *to* the current vertex upon reading $\overline{x}$. Mark the final vertex of the path thus traced

as ω, and remove all edges not traversed during reading of w. The result is the *Munn tree* of w, and the free inverse monoid relations ensure that two words produce the same Munn tree iff they represent the same element of $\mathrm{FIM}(X)$.

To multiply two Munn trees, simply attach the start vertex of the second tree to the end vertex of the first tree, and identify any edges with the same label and direction issuing from or entering the same vertex. From this it can be seen that the *idempotents* (elements x such that $x^2 = x$) in $\mathrm{FIM}(X)$ are those elements whose Munn trees have $\alpha = \omega$, and that these elements commute.

For a more detailed discussion, with diagrams, see [17, Section 6.4].

2.2 Word problems of inverse monoids

Two notions of word problem for inverse monoids will occur throughout this paper. For an inverse monoid M with finite generating set X, the *word problem* of M with respect to X is the set

$$\mathrm{WP}(M, X) = \{u\#v^{\mathrm{inv}} \mid u, v \in (X^{\pm})^*, u =_M v\},$$

while the *two-tape word problem* of M with respect to X is

$$\iota(M, X) = \{(u, v) \in (X^{\pm})^* \times (X^{\pm})^* \mid u =_M v\}.$$

If the generating set X is irrelevant, we may use the notation $\mathrm{WP}(M)$ or $\iota(M)$.

Each of these notions generalises the definition of the group word problem $W(G, X)$ as the set of all words over $X^{\pm}$ representing the identity. If M is a group, then $W(G, X)$ and $\mathrm{WP}(G, X)$ are obtained from each other by very simple operations (deletion or insertion of a single #), and so membership in any 'reasonable' language class will not depend on whether we consider the group or inverse monoid word problem. For the two-tape word problem the generalisation is of a more algebraic nature: $\iota(M, X)$ and $W(G, X)$ are each the lift to $(X^{\pm})^*$ of the natural homomorphism from the free inverse monoid (respectively free group) on X to M (respectively G). The kernel of a group homomorphism is a set, while the kernel of a semigroup homomorphism is a relation.

The word problem for semigroups in general has been studied in [12], where it is defined as the set of words $u\#v^{\mathrm{rev}}$ with u and v representing the same element. For inverse monoids, this is equivalent to the word problem considered here, since $u\#v^{\mathrm{inv}}$ is obtained from $u\#v^{\mathrm{rev}}$ by simply replacing every symbol after the # by its inverse. This operation can be viewed as an inverse generalised sequential machine mapping, and thus all classes of languages we consider are closed under it (and hence all results in this paper hold for the definition in [12] as well). Note that it is still essential to include the 'dividing symbol' #: as an example, if $F = \mathrm{FIM}(X)$ and $x \in X$, then $x\#\overline{x} \in \mathrm{WP}(F, X)$, but $x\overline{x}\# \notin \mathrm{WP}(F, X)$.

3 The rank 1 case

Since the free group of rank 1 is isomorphic to $(\mathbb{Z}, +)$, Munn trees in the rank 1 case can be viewed as intervals of integers containing zero (the starting point α),

with a marked point (ω). This allows elements of FIM_1 to be represented by a 3-tuple of integers $(-l, n, m)$ with $l, n \in \mathbb{N}_0$ and $-l \leq m \leq n$, where $[-l, n]$ is the interval spanned by the Munn tree and m is the marked point. Multiplication in this representation of FIM_1 is given by

$$(-l, n, m)(-l', n', m') = (\min\{-l, m - l'\}, \max\{n, m + n'\}, m + m').$$

Equipped with this model of FIM_1, we can determine that free inverse monoids never have context-free word problem.

Theorem 1. *For any $k \in \mathbb{N}$, neither $\mathrm{WP}(\mathrm{FIM}_k)$ nor $\iota(\mathrm{FIM}_k)$ is context-free.*

Proof. Suppose that $\mathrm{WP}(\mathrm{FIM}_k, X)$ is context-free (X any finite generating set of FIM_k). Then for any $x \in X$, the language $L := \mathrm{WP}(\mathrm{FIM}_k, X) \cap x^*\overline{x}^*x^*\#\overline{x}^*$ is also context-free. For $n \in \mathbb{N}$, let $w_n = x^n\overline{x}^n x^n \#\overline{x}^n$, which is in L for all $n \in \mathbb{N}$. For n greater than the pumping length p of L, we can express w_n in the form $uvwyz$ such that $|vy| \geq 1$, $|vwy| \leq p$, and the strings v, y can simultaneously be 'pumped'. Thus there must exist $i, j \in \mathbb{N}_0$, not both zero, such that all strings of one of the following three forms must be in L for $m \geq -1$:

$$x^{n+im}\overline{x}^{n+jm}x^n\#\overline{x}^n, \quad x^n\overline{x}^{n+im}x^{n+jm}\#\overline{x}^n, \quad x^n\overline{x}^n x^{n+im}\#\overline{x}^{n+jm}.$$

However, in all cases, some words of the given form are not in L:

Word form	Not in L for
$x^{n+im}\overline{x}^{n+jm}x^n\#\overline{x}^n$	$(i \neq 0 \wedge m \geq 1) \vee (i = 0 \wedge j \neq 0 \wedge m \geq 1)$
$x^n\overline{x}^{n+im}x^{n+jm}\#\overline{x}^n$	as above
$x^n\overline{x}^n x^{n+im}\#\overline{x}^{n+jm}$	$(j \neq 0 \wedge m = -1) \vee (j = 0 \wedge i \neq 0 \wedge m \geq 1)$

Thus L, and hence $\mathrm{WP}(\mathrm{FIM}_k, X)$, is not context-free. The proof for $\iota(\mathrm{FIM}_k, X)$ is similar, by intersecting with $(x^*\overline{x}^*x^*, \{x, \overline{x}\}^*)$ and using the pumping lemma on $(x^n\overline{x}^n x^n, x^n)$ for sufficiently large n. □

For the remainder of this section, let FIM_1 be generated by $X = \{x\}$ and let $Y = X^{\pm} = \{x, \overline{x}\}$. For $w \in Y^*$, denote the image of w in FIM_1 by $\hat{w}$. We define functions λ, ν and μ from Y^* to $\mathbb{Z}$ by setting $(-\lambda(w), \nu(w), \mu(w)) = \hat{w}$. It will often be helpful to regard words in Y^* as paths in the integers starting at 0, with x representing a step in the positive direction and $\overline{x}$ a step in the negative direction. We will refer to and visualise these directions as right (positive) and left (negative). Thus for $w \in Y^*$ the path traced out by w has rightmost point $\nu(w)$, leftmost point $-\lambda(w)$ and endpoint $\mu(w)$.

The idempotents in FIM_1 are the elements $(-l, n, 0)$ for $l, n \in \mathbb{N}_0$. We define the set of *positive idempotents* $E^+ = \{(0, n, 0) \mid n \in \mathbb{N}_0\}$ and similarly the set of *negative idempotents* $E^- = \{(-l, 0, 0) \mid l \in \mathbb{N}_0\}$ in FIM_1. (Note that in these definitions, the identity $(0, 0, 0)$ is counted as both a positive and a negative idempotent.) Grammars for the sets of positive and negative idempotents form an important building block in Theorem 2 (as well as in the ET0L grammar mentioned following Corollary 1).

Lemma 1. *Let* $Y = \{x, \overline{x}\}$ *and* $L_{E^+} = \{w \in Y^* \mid \hat{w} \in E^+\}$. *Then* L_{E^+} *is generated by the context-free grammar* $\Gamma^+ = (\{S\}, Y, P^+, S)$ *with* P^+ *consisting of productions* $P_1 : S \to SS$, $P_2 : S \to xS\overline{x}$ *and* $P_3 : S \to \varepsilon$. *Similarly,* $L_{E^-} := \{w \in Y^* \mid \hat{w} \in E^-\}$ *is generated by the context-free grammar* $\Gamma^- = (\{S\}, Y, P^-, S)$ *with* P^- *the same as* P^+ *except that* P_2 *is replaced by* $P_2' : S \to \overline{x}Sx$.

Proof. We can view L_{E^+} as the language of all paths starting and ending at 0 and never crossing to the left of 0. Concatenating two such paths gives another such path, so we have $(L_{E^+})^* = L_{E^+}$. Let L be the language of all paths in Y^* that start and end at 0 without visiting 0 in between. Then $L_{E^+} = L^*$ and $w \in Y^*$ is in L if and only if either $w = \varepsilon$ or there exists $v \in L_{E^+}$ such that $w = xv\overline{x}$. That is, $L_{E^+} = (xL_{E^+}\overline{x})^*$.

Let M be the language generated by Γ^+. We show by induction on the length of words that $L_{E^+} = M$. Note that for any w in L_{E^+} or M we have $|w|_x = |w|_{\overline{x}}$, so both languages consist of words of even length. To begin with, $L_{E^+} \cap Y^0 = M \cap Y^0 = \{\varepsilon\}$. Now suppose that $L_{E^+} \cap Y^{2i} = M \cap Y^{2i}$ for all $i < n$. For $w \in Y^{2n}$, we have $w \in L_{E^+}$ if and only if either $w = w_1w_2$ or $w = xv\overline{x}$ for some $w_1, w_2, v \in L_{E^+}$. By induction, this occurs iff $w_1, w_2 \in M$ respectively $v \in M$, iff $S \to SS \Rightarrow w_1w_2$ respectively $S \to xS\overline{x} \Rightarrow xv\overline{x}$ in Γ^+, iff $w \in M$. Hence $L_{E^+} = M$. The language L_{E^-} is the reverse of L_{E^+}, and the grammars Γ^+ and Γ^- are the reverse of one another, hence L_{E^-} is generated by Γ^-. □

A word $u\#v^{\text{inv}}$ with $u, v \in Y^*$ is in $\text{WP}(\text{FIM}_1, X)$ if and only if it traces out a path starting and ending at 0 which reaches its rightmost and leftmost points each at least once before and at least once after the #. If the minimum or maximum is achieved at the end of u, this counts as being achieved both before and after #. (The path must end at 0 because if $\hat{u} = \hat{v}$ then $\hat{u}(\hat{v})^{-1}$ is an idempotent.) We now show that although the word problem of FIM_1 is not context-free, it can be expressed as an intersection of two context-free languages.

Theorem 2. $\text{WP}(\text{FIM}_1)$ *and* $\iota(\text{FIM}_1)$ *are both* 2-$\mathcal{CF}$.

Proof. Let $X = \{x\}$, $Y = \{x, \overline{x}\}$ and $L = \text{WP}(\text{FIM}_1, X)$. We can express L as the intersection of the following two languages:

$$L_\nu = \{u\#v^{\text{inv}} \mid u, v \in Y^*, \nu(u) = \nu(v) \wedge \mu(u) = \mu(v)\}$$

and

$$L_\lambda = \{u\#v^{\text{inv}} \mid u, v \in Y^*, \lambda(u) = \lambda(v) \wedge \mu(u) = \mu(v)\}.$$

We will show that L_ν and L_λ are each context-free and hence L is 2-$\mathcal{CF}$. Since L_λ is simply the reverse of L_ν, it suffices to prove that L_ν is context-free.

Let $\Gamma_\nu = (V, \Sigma, P, S)$ be the context-free grammar with nonterminals $V = \{S, T, Z, Z'\}$, terminals $\Sigma = \{x, \overline{x}, \#\}$ and productions P as follows:

$$\begin{aligned} S &\to ZSZ \mid xS\overline{x} \mid T \\ T &\to ZTZ \mid \overline{x}Tx \mid \# \\ Z &\to ZZ \mid \overline{x}Zx \mid \varepsilon. \end{aligned}$$

Any derivation in Γ_ν can be expressed as

$$S \Rightarrow \alpha S \beta \to \alpha T \beta \Rightarrow u_1 u_2 \# v_2 v_1, \tag{1}$$

where $\alpha \Rightarrow u_1$, $\beta \Rightarrow v_1$ and $T \Rightarrow u_2 \# v_2$.

For any $\alpha' \in \{Z, x\}^*$ and $\beta' \in \{Z, \overline{x}\}^*$ with $|\alpha'|_x = |\beta'|_{\overline{x}}$, there is a partial derivation in Γ_ν, not involving the production $S \to T$, from S to $\alpha' S \beta'$. Conversely, any partial derivation from S not involving $S \to T$ results in a string $\alpha S \beta$ in which α and β can be derived from some such α' and β' respectively.

Let $\alpha \in \{Z, x\}^*$ and $w \in Y^*$ with $\alpha \Rightarrow^* w$. By Lemma 1, the subwords of w produced from instances of Z in α evaluate to negative idempotents, and so have no effect on $\nu(w)$ or $\mu(w)$, whereas each x in α increases both $\nu(w)$ and $\mu(w)$ by 1. Hence $\nu(w) = \mu(w) = |\alpha|_x$. Thus a pair of words u_1 and v_1 can appear in the derivation (1) if and only if $\nu(u_1) = \mu(u_1) = \mu(v_1^{\mathrm{inv}}) = \mu(v_1^{\mathrm{inv}})$. Similarly, it can be shown that $T \Rightarrow u_2 \# v_2$ if and only if $\nu(u_2) = \nu(v_2^{\mathrm{inv}}) = 0$ and $\mu(u_2) = \mu(v_2^{\mathrm{inv}}) \leq 0$.

Hence $S \Rightarrow u \# v$ if and only if we can write $u = u_1 u_2$ and $v = v_2 v_1$ such that there exist $l_1, l_2, l_1', l_2', m, n \in \mathbb{N}_0$ with

$$\begin{aligned} u_1 &=_{\mathrm{FIM}_1} (-l_1, n, n) & \qquad u_2 &=_{\mathrm{FIM}_1} (-l_2, -m, 0) \\ v_1^{\mathrm{inv}} &=_{\mathrm{FIM}_1} (-l_1', n, n) & \qquad v_2^{\mathrm{inv}} &=_{\mathrm{FIM}_1} (-l_2', -m, 0). \end{aligned}$$

If $u \# v \in L_\nu$, then we can express u and v in this way by setting u_1 to be the shortest prefix of u such that $\nu(u_1) = \nu(u)$ and v_1^{inv} the shortest prefix of v^{inv} such that $\nu(v_1^{\mathrm{inv}}) = \nu(v^{\mathrm{inv}})$. Conversely, supposing we can express u and v in this way, we have

$$\begin{aligned} u &=_{\mathrm{FIM}_1} (-l_1, n, n)(-l_2, n, -m) = (-i, n, n-m) \\ v^{\mathrm{inv}} &=_{\mathrm{FIM}_1} (-l_1', n, n)(-l_2', 0, -m) = (-j, n, n-m) \end{aligned}$$

for some $i, j \in \mathbb{N}_0$. That is, $u \# v \in L_\nu$.

Hence L_ν is generated by Γ_ν and is context-free, and therefore $L_\nu^{\mathrm{rev}} = L_\lambda$ is also context-free. Thus $\mathrm{WP}(\mathrm{FIM}_1, X) = L_\nu \cap L_\lambda$ is 2-$\mathcal{CF}$.

For variety, we give an automaton proof for $\iota(\mathrm{FIM}_1, X)$. Define sublanguages L_ν^ι and L_λ^ι of $Y^* \times Y^*$ analogously to L_ν and L_λ. Let $x_1 = (x, \epsilon)$, $x_2 = (\epsilon, x)$ and define $\overline{x}_1$, $\overline{x}_2$ similarly. Reading x_i or $\overline{x}_i$ means that we read an x or $\overline{x}$ from the i-th tape and nothing from the other tape. Define a pushdown automaton $\mathcal{A}_\nu$ with states q_0, q_1 by the following transitions for $i = 1, 2$ (Z is the bottom-of-stack symbol):

$$\begin{aligned} (q_0, Z, (x, x)) &\mapsto (q_0, Z) & \qquad (q_1, Z, (\overline{x}, \overline{x})) &\mapsto (q_1, Z) \\ (q_0, Z, \overline{x}_i) &\mapsto (q_0, Y_i Z) & \qquad (q_1, Z, \overline{x}_i) &\mapsto (q_1, Y_i Z) \\ (q_0, Y_i, \overline{x}_i) &\mapsto (q_0, Y_i Y_i) & \qquad (q_1, Y_i, \overline{x}_i) &\mapsto (q_1, Y_i Y_i) \\ (q_0, Y_i, x_i) &\mapsto (q_0, \epsilon) & \qquad (q_1, Y_i, x_i) &\mapsto (q_1, \epsilon). \\ (q_0, Z, \epsilon) &\mapsto (q_1, Z) \end{aligned}$$

The language $\mathcal{A}_\nu$ accepts by empty stack consists of all pairs (u, v) where $u, v \in (E^- x)^n (E^- \overline{x})^k E^-$ for some $n, k \in \mathbb{N}_0$, which is precisely the language L^ι_ν. Switching the roles of x and $\overline{x}$ in $\mathcal{A}_\nu$ gives rise to a pushdown automaton $\mathcal{A}_\lambda$ accepting L^ι_λ. Hence $\iota(\mathrm{FIM}_1, X)$ is also 2-$\mathcal{CF}$. □

Theorem 3. *Both versions of the co-word problem of* FIM_1 *are context-free.*

Proof. Let $K = \mathrm{coWP}(\mathrm{FIM}_1, X) = \{u\#v^{\mathrm{inv}} \mid u, v \in Y^*, u \neq_{\mathrm{FIM}_1} v\}$. A word $w = u\#v$ with $u, v \in Y^*$ is in K if and only if the path traced out by w starting at 0 either does not end at 0, or its minimum or maximum value is not achieved both before and after # (recall that this includes not being achieved at the end of u). Thus a context-free grammar for K with start symbol S is given by the following productions:

$$\begin{aligned}
S &\to M \mid U \mid D & U &\to ZxU\overline{x}Z \mid xE\overline{x}Z\# \mid \#ZxE\overline{x}Z \\
M &\to ExA \mid E\overline{x}B & D &\to Z'\overline{x}DxZ' \mid \overline{x}ExZ'\# \mid \#Z'\overline{x}ExZ' \\
A &\to xA \mid EAE \mid \epsilon & Z &\to ZZ \mid \overline{x}Zx \mid \epsilon \\
B &\to \overline{x}B \mid EBE \mid \epsilon & Z' &\to Z'Z' \mid xZ'\overline{x} \mid \epsilon. \\
E &\to ZE \mid Z'E \mid \epsilon
\end{aligned}$$

M generates all $u\#v^{\mathrm{inv}}$ with $\mu(u) \neq \mu(v)$; U generates all $u\#v^{\mathrm{inv}}$ with uv^{inv} idempotent but $\nu(u) \neq \nu(v)$, and D does the same as U but for λ instead of ν.

The two-tape co-word problem of FIM_1 with respect to X is the language $M = \{(u, v) \in Y^* \times Y^* \mid u \neq_{\mathrm{FIM}_1} v\}$. A pushdown automaton recognising M can be expressed as the union of automata $\mathcal{B}_\mu$, B_ν, B_λ. The automaton $\mathcal{B}_\mu$ checks that $|u|_x - |u|_{\overline{x}} \neq |v|_x - |v|_{\overline{x}}$ for input (u, v), and thus accepts all pairs with $\mu(u) \neq \mu(v)$. The automaton $\mathcal{B}_\nu$ has states q_0, q_1, q_2, f, with f being the unique final state, input symbols x_i, $\overline{x}_i$ (as in the proof of Theorem 2) and transitions:

$$\begin{array}{lll}
(q_0, x_1, Z) \to (q_0, XZ) & (q_0, \epsilon, *) \to (q_1, *) & (q_2, x_2, Z) \to (f, Z) \\
(q_0, x_1, X) \to (q_0, XX) & (q_1, \epsilon, Y) \to (q_1, \epsilon) & (q_2, x_2, X) \to (q_2, \epsilon) \\
(q_0, x_1, Y) \to (q_0, \epsilon) & (q_1, x_2, X) \to (q_2, \epsilon) & (q_2, x_2, Y) \to (q_2, \epsilon) \\
(q_0, \overline{x}_1, *) \to (q_0, Y*) & (q_1, \overline{x}_2, X) \to (q_2, YX) & (q_2, \overline{x}_2, *) \to (q_2, Y*) \\
 & & (q_2, \epsilon, X) \to (f, X),
\end{array}$$

where Z is the bottom-of-stack marker and $*$ denotes any stack symbol (X, Y, Z). In state q_0, $X^{\nu(u)}Y^{\nu(u)-\mu(u)}$ is placed on the stack. State q_1 removes all Y's. State q_2 then checks $\nu(v)$ against $\nu(u)$, moving to the final state f if we either find that $\nu(v) > \nu(u)$ or nondeterministically if $\nu(v') < \nu(u)$ for the prefix v' of v read so far, in which case $\mathcal{B}_\nu$ accepts if there is no further input. Thus $\mathcal{B}_\nu$ accepts the language of all (u, v) with $\nu(u) \neq \nu(v)$. The automaton $\mathcal{B}_\lambda$ is obtained by swapping the roles of x_i and $\overline{x}_i$ in $\mathcal{B}_\nu$, and accepts (u, v) with $\lambda(u) \neq \lambda(v)$. □

Given the model of elements of FIM_1 as marked intervals in $\mathbb{Z}$, stack automata provide possibly the most natural class of automata to consider as acceptors of its word problem. It turns out that it suffices to use a checking stack automaton.

Theorem 4. $\mathrm{WP}(\mathrm{FIM}_1)$ *and* $\iota(\mathrm{FIM}_1)$ *are each recognised by a checking stack automaton.*

Proof. The idea of the checking stack automaton for $\mathrm{WP}(\mathrm{FIM}_1)$ is to use the stack contents as a model for an interval of integers $[-l, n]$ (chosen nondeterministically before beginning to read the input), and check for input $u\#v^{\mathrm{inv}}$ whether $\hat{u} = \hat{v} = (-l, n, m)$ for some $m \in [-l, n]$. Following the set-up phase, the stack contents will always be of the form $L^l O R^n$ for some $l, n \in \mathbb{N}_0$, with the leftmost symbol L or O being marked with a superscript $^-$, and the rightmost symbol O or R being marked with a superscript $^+$. Such a string represents a guess that the input string $u\#v^{\mathrm{inv}}$ will have $\lambda(u) = \lambda(v) = l$ and $\nu(u) = \nu(v) = n$. For $\alpha \in L^*OR^*$, we denote the string 'α with marked endpoints' by $[\alpha]$. For example, $[LLORR] = L^-LORR^+$ and $[O] = O^{\pm}$. Before beginning to consume the input, the stack marker is moved to $O^{(+,-,\pm)}$.

During the checking phase, the automaton $\mathcal{A}$ moves up and down the stack, tracing out the path given by $u\#v^{\mathrm{inv}}$, accepting if and only if three conditions are satisfied: (i) both the left and right endpoints of $[\alpha]$ are reached at least once before and after the #; (ii) the automaton never attempts to move beyond the endpoints of $[\alpha]$; and (iii) the automaton ends with the stack marker pointing at $O^{(+,-,\pm)}$. If during the set-up phase the string $[L^l OR^n]$ was placed on the stack, then the set of words accepted by $\mathcal{A}$ following that particular set-up will be the language $\{u\#v^{\mathrm{inv}} \mid u, v \in Y^*, \lambda(u) = \lambda(v) = l, \nu(u) = \nu(v) = n, \mu(u) = \nu(v)\}$.

More formally, the checking transitions of $\mathcal{A}$ are described as follows, using states $\{q_i, q_i^+, q_i^-, q_i^*, f \mid i = 1, 2\}$, with f the unique final state. Following the setup phase (which can be achieved non-deterministically using two states), $\mathcal{A}$ is in state q_1, with stack contents $[\alpha]$ for some string $\alpha \in L^*OR^*$, and the stack marker pointing at the symbol corresponding to O in $[\alpha]$. The symbol \$ is an end-of-input marker, standardly included in the definition of stack automata. Let $\Delta^- = \{O^-, R^-\}$ and $\Delta^+ = \{O^+, L^+\}$. The left-hand side of a transition represents the current automaton configuration (state,input,stack symbol). The right-hand side has first component the state to be moved to, and second component the direction in which to move the stack marker (with $-$ denoting no change). The full set of stack symbols is $\Gamma = \{L^{(+,-)}, O^{(+,-,\pm)}, R^{(+,-)}\}$. For $i = 1, 2$:

$(q_1, \#, O^{\pm}) \mapsto (q_2^*, -)$,

$(q_i, x, C) \mapsto (q_i, \uparrow)$, $C \in \{L, O, R\}$

$(q_i, \overline{x}, C) \mapsto (q_i, \downarrow)$, $C \in \{L, O, R\}$

$(q_i, x, C) \mapsto (q_i^-, \uparrow)$, $C \in \Delta^-$ $(q_i^*, x, C) \mapsto (q_i^*, \uparrow)$, $C \notin \{R^+, O^+, O^{\pm}\}$

$(q_i, \overline{x}, C) \mapsto (q_i^+, \downarrow)$, $C \in \Delta^+$ $(q_i^*, \overline{x}, C) \mapsto (q_i^*, \downarrow)$, $C \notin \{L^-, O^-, O^{\pm}\}$

$(q_i^+, x, C) \mapsto (q_i^*, \uparrow)$, $C \in \Delta^-$ $(q_1^*, \#, C) \mapsto (q_2, -)$, $C \in \Gamma$

$(q_i^-, \overline{x}, C) \mapsto (q_i^*, \downarrow)$, $C \in \Delta^+$ $(q_2^*, \$, C) \mapsto (f, -)$, $C \in \{O, O^-, O^+, O^{\pm}\}$.

Note that these transitions involve no push or pop operations, so $\mathcal{A}$ is a checking stack automaton. Now assume that $\mathcal{A}$ has reached the reading phase, with stack contents $[L^l OR^n]$ for some $l, n \in \mathbb{N}_0$. Let $L_{(l,n)}$ denote the language of all words accepted by $\mathcal{A}$ from this configuration. The case $(l, n) = (0, 0)$ is degenerate, since $[O] = O^{\pm}$ and the only path from q_0 to f in this case is on input

#\$, which is exactly as desired since the empty word is the only representative of the identity $(0,0,0)$ in Y^*. Henceforth assume at least one of l or n is non-zero.

With few exceptions, the automaton moves up the stack on input x and down on $\overline{x}$. The exceptions are when this would otherwise result in moving beyond the top or bottom of the stack. In these cases there are no transitions defined and so the automaton fails. Thus for $w \in Y^*$ the stack marker traces out the path given by w, provided this path remains within the interval $[-l, n]$.

When the automaton is in state q_i and has reached the top of the stack (indicated by a symbol in Δ^+), on the next input it either fails (on x) or moves to state q_i^+ (on $\overline{x}$). Similarly, after reaching the bottom of the stack (symbols in Δ^-), the automaton either fails (on $\overline{x}$) or moves to q_i^- (on x). Following either of these events, the automaton will move to state q_i^* after reaching the opposite end of the stack, provided it does not fail. Thus being in state q_0^* indicates that the automaton has read some $u' \in Y^*$ with $\lambda(u') = l$ and $\nu(u') = n$.

The only transition on the symbol # is from state q_0^* to q_1 (regardless of stack symbol), and the only transition on \$ is from q_1^* to the final state f and requires the automaton to be pointing at $O^{(+,-)}$ (both of these transitions leave the stack unchanged). Hence $L_{(l,n)}$ contains exactly those words in $Y^*\#Y^*\$$ which trace out a path in $[-l, n]$ starting and ending at $O^{(+,-)}$ which visit the top and bottom of the stack each at least once before and after the #; that is, $L_{(l,n)}$ consists of all $u\#v^{\mathrm{inv}}\$$ such that $u\#v^{\mathrm{inv}}$ is in $\mathrm{WP}(\mathrm{FIM}_1, X)$ and $\lambda(u) = l$, $\nu(u) = n$, as desired. Since the language accepted by $\mathcal{A}$ is $\bigcup_{l,n\in\mathbb{N}_0} L_{(l,n)}$, we conclude that $\mathcal{A}$ accepts $\mathrm{WP}(\mathrm{FIM}_1, X)$.

To recognise $\iota(\mathrm{FIM}_1, X)$, we make a few small modifications to $\mathcal{A}$: in the setup phase, we additionally mark some symbol of the stack contents $[\alpha]$ to denote a guess as to the location of $\mu(u) = \mu(v)$ (where the input is $(u\$, v\$)$). In states q_i, q_i^+, q_i^-, q_i^*, we read from the i-th tape ($i = 1, 2$). On reaching the end symbol \$ on each tape, we only proceed if the stack marker is pointing at the marked symbol. We introduce an intermediate state between q_1^* and q_2 which returns the stack marker to the symbol corresponding to O in α. In all other respects, the automaton behaves the same as $\mathcal{A}$. Thus the stack contents of the modified automaton represent an element $(-l, n, m)$ of FIM_1, and with these stack contents the automaton accepts all (u, v) such that u and v both evaluate to $(-l, n, m)$. Evaluating over all possible stack contents yields $\iota(\mathrm{FIM}_1, X)$. □

Note that the classes of 2-$\mathcal{CF}$ and checking stack languages are incomparable. The language $\{ww \mid w \in A^*\}$ for $|A| \geq 2$ is not poly-$\mathcal{CF}$ (an easy application of [3, Theorem 3.9]), but is accepted by a checking stack automaton that starts by putting a word w on the stack and then checks whether the input is ww. The language $\{(ab^n)^n \mid n \in \mathbb{N}\}$ is not even indexed [15, Theorem 5.3], but is 3-$\mathcal{CF}$.

Since E(D)T0L languages have been shown to describe various languages arising in group theory [5, 6] (but not word problems), it is worth noting the following.

Corollary 1. $\mathrm{WP}(\mathrm{FIM}_1)$ *and* $\iota(\mathrm{FIM}_1)$ *are both ET0L.*

Proof. This follows from Theorem 4 and the fact that the class of checking stack languages is contained in the ET0L languages [25]. □

The author has constructed nondeterministic ET0L grammars for both versions of the word problem of FIM_1 with 9 tables and 11 nonterminals. The nondeterminism arises from the fact that for any word $w \in Y^*$, we may insert idempotents arbitrarily at any point in w without changing the element represented, provided that these idempotents are not 'large' enough to change the value of $\nu(w)$ or $\lambda(w)$.

Conjecture 1 *Neither* $\mathrm{WP}(\mathrm{FIM}_1)$ *nor* $\iota(\mathrm{FIM}_1)$ *is EDT0L.*

4 Rank greater than 1

The word problem for inverse monoids in higher ranks is more complex from a language theory perspective.

Lemma 2. *For any* $k \geq 3$, $\mathrm{WP}(\mathrm{FIM}_k)$ *is not* $(k-2)$-$\mathcal{CF}$.

Proof. For any $k \geq 2$, let $X_k = \{x_1, \ldots, x_k, \overline{x}_1, \ldots, \overline{x}_k\}$ and let

$$L_k = \{x_1^{m_1}\overline{x}_1^{m_1} \ldots x_k^{m_k}\overline{x}_k^{m_k} \# x_1^{n_1}\overline{x}_1^{n_1} \ldots x_k^{n_k}\overline{x}_k^{n_k} \mid m_i, n_i \in \mathbb{N}_0\}.$$

Let $W_k = \mathrm{WP}(\mathrm{FIM}_k, X_k)$. Then W_k consists of all those words in L_k with $m_i = n_i$ for all i (since idempotents in FIM_k commute). By [3, Theorem 3.12], W_k is not $(k-1)$-context-free [1]. Since W_k is the intersection of $\mathrm{WP}(\mathrm{FIM}_k, X_k)$ with the context-free language L_k, this implies that $\mathrm{WP}(\mathrm{FIM}_k, X_k)$ is not context-free. □

Since for $k \geq 2$ FIM_k contains submonoids isomorphic to FIM_n for all n, the following theorem is immediate from Lemma 2.

Theorem 5. *For any* $k \geq 2$, $\mathrm{WP}(\mathrm{FIM}_k)$ *is not poly-*$\mathcal{CF}$.

Note that the argument in Lemma 2 does not work for $\iota(\mathrm{FIM}_k)$, since the set $\{(x_1^{m_1}\overline{x}_1^{m_1} \ldots x_k^{m_k}\overline{x}_k^{m_k}, x_1^{m_1}\overline{x}_1^{m_1} \ldots x_k^{m_k}\overline{x}_k^{m_k}) \mid m_i \in \mathbb{N}_0\}$ is context-free. Writing the idempotent on the second tape in reverse (that is, starting with $x_k^{m_k}\overline{x}^{m_k}$) does not help, as the resulting language is still 2-$\mathcal{CF}$. It appears likely that the intersection of $\iota(\mathrm{FIM}_k)$ with the following $\mathcal{CF}$ language would not be $(k-1)$-$\mathcal{CF}$:

$$\{(x_1^{l_1}\overline{x}_1^{l_1} \ldots x_k^{l_k}\overline{x}_k^{l_k} x_1^{m_1}\overline{x}_1^{m_1} \ldots x_k^{m_k}\overline{x}_k^{m_k}, x_1^{n_1}\overline{x}_1^{n_1} \ldots x_k^{n_k}\overline{x}_k^{n_k}) \mid l_i, m_i, n_i \in \mathbb{N}_0\},$$

but proving this would require delicate arguments about intersections of stratified semilinear sets beyond the scope of this paper.

The author conjectures that neither version of the word problem for FIM_k, $k \geq 2$ is indexed. While a nested stack automaton can easily be used to store a Munn tree, there appears to be no way to check while reading a word $w \in X^*$ that the path traced out by w visits every leaf of the stored Munn tree.

[1] The language $L^{(2,k)}$ in the referenced result is not precisely W_k, but the associated set of integer tuples differs from that associated to W_k only by a constant (arising from the symbol #), which does not affect stratification properties and therefore does not affect the property of not being $(k-1)$-$\mathcal{CF}$.

References

1. A. Aho, 'Nested stack automata', *J. ACM* **16**(3), 383–406 (1969).
2. W.W. Boone, 'The word problem', *Ann. of Math.*, **2**(70), 207–265 (1959).
3. T. Brough, 'Groups with poly-context-free word problem', *Groups Complex. Cryptol.* **6**(1), 9–29 (2014).
4. T. Brough, 'Inverse semigroups with rational word problem are finite', Unpublished note: `arxiv:1311.3955` (2013).
5. L. Ciobanu, V. Diekert and M. Elder, 'Solution sets for equations over free groups are EDT0L languages', In: Automata, Languages, and Programming. ICALP 2015. *Lecture Notes in Comput. Sci.* **9135**. Springer, Berlin, Heidelberg (2015).
6. L. Ciobanu, M. Elder and M. Ferov, 'Applications of L systems to group theory', *Internat. J. Algebra Comput.* **28**, 309–329 (2018).
7. A. Duncan and R.H. Gilman, 'Word hyperbolic semigroups', *Math. Proc. Cambridge Philos. Soc.* **136**, 513–524 (2004).
8. N.D. Gilbert and R. Noonan Heale, 'The idempotent problem for an inverse monoid', *Internat. J. Algebra Comput.* **21**, 1170–1194 (2011).
9. R.H. Gilman and M. Shapiro, 'On groups whose word problem is solved by a nested stack automaton', `arxiv:math/9812028`.
10. S. Ginsburg, S.A. Greibach and M.A. Harrison, 'One-way stack automata', *J. ACM* **14**(2), 389–418 (1967).
11. S. Greibach, 'Checking automata and one-way stack languages', *J. Comptr. System Sci.* **3**, 196–217 (1969).
12. M. Hoffman, D.F. Holt, M.D. Owens and R.M. Thomas, 'Semigroups with a context-free word problem', *DLT '12 Proceedings of the 16th international conference on Developments in Language Theory*, 97–108 (2012).
13. D.F. Holt, M.D. Owens and R.M. Thomas, 'Groups and semigroups with a one-counter word problem', *J. Aust. Math. Soc.* **85**, 197–209 (2005).
14. D.F. Holt, C.E. Röver, S.E. Rees and R.M. Thomas, 'Groups with a context-free co-word problem', *J. Lond. Math. Soc.* **71**, 643–657 (2005).
15. J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley (1979).
16. M. Kambites, 'Anisimov's Theorem for inverse semigroups', *Internat. J. Algebra Comput.* **25**, 41–49 (2015).
17. M.V. Lawson, 'Inverse Semigroups: The Theory of Partial Symmetries', World Scientific (1998).
18. M. Lohrey and N. Ondrusch, 'Inverse monoids: Decidability and complexity of algebraic questions, *Inform. and Comput.*, **205**(8), 1212–1234 (2007).
19. L.P. Lisovik and V.N. Red'ko, 'Regular events in semigroups', *Problemy Kibernetiki* **37**, 155–184 (1980).
20. W.D. Munn, 'Free inverse semigroups', *Proc. Lond. Math. Soc.* **s3-29**(3), 385–404 (1974).
21. P.S. Novikov, 'On the algorithmic unsolvability of the word problem in group theory', *Amer. Math. Soc. Transl. Ser. 2* **9**, 1–122 (1958).
22. M.J. Pfeiffer, 'Adventures in applying iteration lemmas', PhD thesis, University of St Andrews (2013).
23. E. Post, 'Recursive unsolvability of a problem of Thue', *J. Symb. Log.* **12**(1), 1–11 (1947).
24. G. Rozenberg and A. Salomaa, *The Book of L*, Springer (1986).
25. J. van Leeuwen, 'Variations of a new machine model', *Conf Record 17th Annual IEEE Symp on Foundations of Computer Science*, 228–235 (1976).