Free Products of Semigroups and Monoids with a Deterministic Context-Free Word Problem

Peter Kostolányi

Department of Computer Science, Comenius University in Bratislava, Mlynská dolina, 842 48 Bratislava, Slovakia

Abstract

The class of all finitely generated semigroups with a deterministic context-free word problem is shown to be closed under free products, answering a question of T. Brough, A. J. Cain, and M. Pfeiffer. On the other hand, it is proved that the class of all finitely generated monoids with a deterministic context-free word problem is not closed under monoid free products.

Keywords: Word problem, Free product, Semigroup, Monoid, Deterministic pushdown automaton

1. Introduction

The *word problem* for a semigroup S generated by a finite set A can be described as follows: given two nonempty words u, v over the alphabet A upon input, decide whether these two words evaluate to the same element in S. The word problem for a finitely generated monoid is defined in a similar way, except that the input words need not be nonempty; the definition is the same for groups, *i.e.*, one views a finitely generated group G as a monoid generated by some finite set A, and asks which pairs of words over A evaluate to the same element of G.

Every decision problem can in principle be represented by a formal language; for word problems, this representation turns out to be most natural in the case of groups. When G is a group finitely generated by A as a monoid, then actually already the language of all words over A evaluating to the identity element of G contains all the essential information about the word problem for G. The study of connections between the properties of this language and the properties of G itself was initiated by A. V. Anisimov [1], who observed that the above-described language is rational (*i.e.*, regular) if and only if G is finite. This line of study was most notably extended by the Muller-Schupp theorem [21, 6], according to which the said language is context-free if and only if G is virtually free. Following these classical results, groups with word problems from many other known classes of languages have been studied [3, 7, 9, 11, 17, 18, 19, 20]. See, e.g., [12] for an introduction to the language-theoretic approach to the word problem for groups.

The language-theoretic representation of the word problem for groups makes no sense for semigroups and usually does not contain enough information for monoids. For this reason, A. Duncan and R. H. Gilman [5] proposed to represent the word problem for a semigroup S, generated by a finite set A, by the language of all words $u\#v^R$, where $u, v \in A^+$ evaluate to the same element in S (see Section 2 for the precise definition) and $\# \notin A$. Similarly, the word problem for a monoid M, finitely generated by A, corresponds to the language of all $u\#v^R$, where $u, v \in A^*$ evaluate to the same element in M and $\# \notin A$. In a sense, these definitions *generalise* the one for groups – the classical results for groups [1, 21] remain valid when they are understood as monoids.

A. Duncan and R. H. Gilman [5] asked for a characterisation of finitely generated semigroups or monoids with a context-free word problem, which would generalise the Muller-Schupp theorem. This

Email address: kostolanyi@fmph.uniba.sk (Peter Kostolányi)

still remains an important open problem [22]. Nevertheless, many basic properties of semigroups and monoids with a context-free word problem were explored, especially by M. Hoffmann et al. [10] and by T. Brough, A. J. Cain, and M. Pfeiffer [4].

A classical question studied in connection to the word problem viewed from a language-theoretic perspective is whether the class of all finitely generated groups, monoids, or semigroups, whose word problem corresponds to a language from some class $\mathcal{C}$, is closed under taking free products. Already A. V. Anisimov [1] proved that the class of finitely generated groups with a context-free word problem is closed under free products; similar results for semigroups and monoids with a context-free word problem were established in [4]. Recently, a more systematic approach to such questions was initiated by C.-F. Nyberg-Brodda [23], who proved that a class of finitely generated semigroups or monoids with a word problem from $\mathcal{C}$ is closed under free products whenever $\mathcal{C}$ is a super-AFL closed under reversal.

Although the question of closure under free products is settled by these results for semigroups and monoids with word problems from many important language classes, some cases still remain open. In particular, T. Brough, A. J. Cain, and M. Pfeiffer [4] asked whether the class of finitely generated semigroups with a *deterministic context-free* word problem is closed under free products; this is also mentioned as an open problem in [23]. While the class of finitely generated groups with a deterministic context-free word problem coincides with the class of finitely generated groups with a context-free word problem [12], the inclusion is strict both for semigroups and for monoids [4].

We observe in this article that the question of [4] can actually be readily answered in affirmative, as a minor change in the construction of [4] for the context-free case is sufficient for this purpose. This means that *the class of all finitely generated semigroups with a deterministic context-free word problem is closed under free products.* On the other hand, we show that the situation is much different for monoids, where it turns out that *the class of all finitely generated monoids with a deterministic context-free word problem is not closed under monoid free products.*

These two results also happen to answer a question of C.-F. Nyberg-Brodda [23], who asked about the existence of a class of languages $\mathcal{C}$ closed under inverse homomorphisms such that the class of all finitely generated semigroups with a word problem in $\mathcal{C}$ is closed under semigroup free products, but the class of all finitely generated monoids with a word problem in $\mathcal{C}$ is not closed under monoid free products. The results anticipated above imply that the class of all deterministic context-free languages is an example of such a class $\mathcal{C}$.

Moreover, as already mentioned, the word problem of a finitely generated group is deterministic context-free if and only if it is context-free [12], and the class of all such groups is closed under free products. As a result, the deterministic context-free languages give an example of a class of languages $\mathcal{C}$ closed under inverse homomorphisms, such that the classes of finitely generated semigroups and groups with a word problem in $\mathcal{C}$ are closed under semigroup free products and group free products, respectively, but the class of all finitely generated monoids with a word problem in $\mathcal{C}$ is not closed under monoid free products.

2. Preliminaries

Some familiarity with the basics of formal language theory [13, 14] and with elementary concepts related to semigroups and monoids [15, 16] is assumed on the part of the reader.

We denote by $\mathbb{N}$ the set of all *nonnegative* integers and write $[n] = \{1, \ldots, n\}$ for all $n \in \mathbb{N}$. As usual, given any finite alphabet A, we denote by A^* the *free monoid* on A and by A^+ the *free semigroup* on A. The *empty word* over any alphabet A is denoted by ε. Moreover, the *reversal* of a word $w \in A^*$ is denoted by w^R, and the *number of occurrences* of $a \in A$ in $w \in A^*$ by $|w|_a$.

Recall that a *pushdown automaton* is a septuple $\mathcal{A} = (Q, \Sigma, \Gamma, T, q_0, \dashv, F)$, where Q is a finite state set, Σ is a finite input alphabet, Γ is a finite pushdown alphabet, $T \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times Q \times \Gamma^*$ is a finite transition set, $q_0 \in Q$ is the initial state, $\dashv \in \Gamma$ is the bottom-of-pushdown symbol, and $F \subseteq Q$ is the set of final states. A *configuration* of $\mathcal{A}$ is a triple (q, w, γ), where $q \in Q$ is a state, $w \in \Sigma^*$ represents the remaining suffix of the input word, and $\gamma \in \Gamma^*$ represents the word stored on the pushdown with bottom on the left. A *step* of $\mathcal{A}$ is a binary relation $\vdash_{\mathcal{A}}$ on the set of all configurations of $\mathcal{A}$, defined for all $p, q \in Q$,

$u, v \in \Sigma^*$, and $\gamma_1, \gamma_2 \in \Gamma^*$ by $(p, u, \gamma_1) \vdash_{\mathcal{A}} (q, v, \gamma_2)$ if and only if there are $z \in \Sigma \cup \{\varepsilon\}$, $\gamma, \beta \in \Gamma^*$, and $Z \in \Gamma$ such that $u = zv$, $\gamma_1 = \gamma Z$, $\gamma_2 = \gamma\beta$, and $(p, z, Z, q, \beta) \in T$. The *language* $\|\mathcal{A}\|$ recognised by $\mathcal{A}$ then consists of precisely all $w \in \Sigma^*$ such that $(q_0, w, \dashv) \vdash_{\mathcal{A}}^* (q, \varepsilon, \gamma)$ for some $q \in F$ and $\gamma \in \Gamma^*$.

A pushdown automaton $\mathcal{A} = (Q, \Sigma, \Gamma, T, q_0, \dashv, F)$ is said to be *deterministic* if the following conditions are satisfied:

(*i*) For all $p \in Q$, $z \in \Sigma \cup \{\varepsilon\}$, and $Z \in \Gamma$, there exists at most one $q \in Q$ and $\beta \in \Gamma^*$ such that $(p, z, Z, q, \beta) \in T$;

(*ii*) If $(p, c, Z, q, \beta) \in T$ for some $p, q \in Q$, $c \in \Sigma$, $Z \in \Gamma$, and $\beta \in \Gamma^*$, then there are no $q' \in Q$ and $\beta' \in \Gamma^*$ such that $(p, \varepsilon, Z, q', \beta') \in T$.

A language is called *deterministic context-free* if it is recognised by some deterministic pushdown automaton; we denote the class of all deterministic context-free languages by $\mathscr{L}(\mathrm{detCF})$. See, e.g., [13, Chapter 10] for the basic theory of deterministic pushdown automata and deterministic context-free languages.

Given a semigroup S generated by a finite set $A \subseteq S$ and $u, v \in A^+$, we write $u =_S v$ if $\nu(u) = \nu(v)$ for the unique semigroup homomorphism $\nu\colon A^+ \to S$ such that $\nu(a) = a$ for all $a \in A$.[1] The *word problem* of S with respect to A is then defined to be the language

$$\mathrm{WP}_A(S) = \{u\#v^R \mid u, v \in A^+;\ u =_S v\},$$

where $\# \notin A$ is a fixed delimiter symbol.

Similarly, when M is a monoid finitely generated by $A \subseteq M$ and $u, v \in A^*$, we write $u =_M v$ if $\eta(u) = \eta(v)$ for the unique monoid homomorphism $\eta\colon A^* \to M$ such that $\eta(a) = a$ for all $a \in A$. The *word problem* of M with respect to A is then given by

$$\mathrm{WP}_A(M) = \{u\#v^R \mid u, v \in A^*;\ u =_M v\}$$

for some fixed delimiter symbol $\# \notin A$. Note that this language is different from the one obtained when M is viewed as a semigroup; this minor ambiguity in notation can nevertheless be always easily resolved from the context.

[1] Note that $=_S$ is actually a congruence on A^+ and that $A^+/{=_S} \cong S$. This means that for a semigroup S presented by $S = \langle A \mid \varrho \rangle$, the relation ϱ generates the congruence $=_S$.

For every finitely generated semigroup S and any two finite generating sets $A, B \subseteq S$ of S, there exists a homomorphism $h\colon (A \cup \{\#\})^* \to (B \cup \{\#\})^*$ such that $\mathrm{WP}_A(S) = h^{-1}(\mathrm{WP}_B(S))$, and the same property also holds for any two generating sets of any finitely generated monoid. This means that when $\mathcal{C}$ is a class of languages closed under inverse homomorphisms and S is a finitely generated semigroup, then $\mathrm{WP}_A(S) \in \mathcal{C}$ either for all finite generating sets A of S, or for no such generating set; and the same also holds for any finitely generated monoid. Under these circumstances, one can thus simply say that "*the* word problem" of a semigroup or a monoid is or is not in $\mathcal{C}$. In this article, we use this convention for the class $\mathcal{C} = \mathscr{L}(\mathrm{detCF})$, which is closed under inverse homomorphisms.

Let S_1, S_2 be (not necessarily finitely generated) semigroups presented by $S_1 = \langle A_1 \mid \varrho_1 \rangle$ and $S_2 = \langle A_2 \mid \varrho_2 \rangle$, while $S_1 \cap S_2 = \emptyset$. The *free product* of S_1 and S_2 then is the semigroup

$$S_1 * S_2 = \langle A_1 \cup A_2 \mid \varrho_1 \cup \varrho_2 \rangle.$$

Similarly, let M_1, M_2 be monoids with identity elements 1_{M_1} and 1_{M_2}, which are presented by $M_1 = \langle A_1 \mid \varrho_1 \rangle$ and $M_2 = \langle A_2 \mid \varrho_2 \rangle$ and satisfy $(M_1 \setminus \{1_{M_1}\}) \cap (M_2 \setminus \{1_{M_2}\}) = \emptyset$. The *monoid free product* of M_1 and M_2 then is the monoid

$$M_1 * M_2 = \langle A_1 \cup A_2 \mid \varrho_1 \cup \varrho_2 \rangle.$$

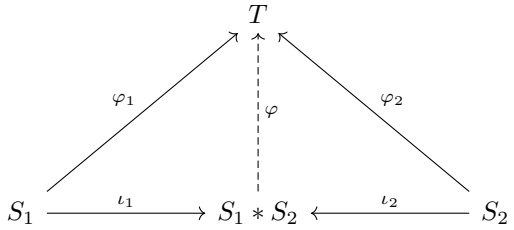
Note that the angle brackets denote *monoid presentations* in the context of monoids and *semigroup presentations* in the context of semigroups. This subtle difference implies that although both definitions given above look precisely the same, the semigroup free product of two monoids is actually a different thing than their monoid free product. In particular, the identity elements of both monoids M_1, M_2 are identified in the monoid free product, resulting in the identity element of $M_1 * M_2$. On the contrary, the semigroup free product of two monoids is not even a monoid. In case $*$ is applied to two monoids in this article, it always denotes the *monoid* free product.

The semigroup free product $S_1 * S_2$ thus consists of all words $a_1 \dots a_m$ with $m \in \mathbb{N} \setminus \{0\}$ over a possibly infinite alphabet $S_1 \cup S_2$ such that for $k = 2, \dots, m$, one has $a_k \in S_i$ for $i \in \{1, 2\}$ if and only if $a_{k-1} \in S_{3-i}$. Given $\mathbf{a} = a_1 \dots a_m$ and $\mathbf{b} = b_1 \dots b_n$ from $S_1 * S_2$ with $m, n \in \mathbb{N} \setminus \{0\}$ and $a_1, \dots, a_m, b_1, \dots, b_n \in S_1 \cup S_2$ satisfying the condition above, the product $\mathbf{ab}$ is defined

by $\mathbf{ab} = a_1 \dots a_m b_1 \dots b_n$, where $a_m b_1$ should be interpreted as multiplication in S_1 if $a_m, b_1 \in S_1$, multiplication in S_2 if $a_m, b_1 \in S_2$, and concatenation otherwise.

The monoid free product $M_1 * M_2$ consists of all words $a_1 \dots a_m$ with $m \in \mathbb{N}$ over a possibly infinite alphabet $(M_1 \setminus \{1_{M_1}\}) \cup (M_2 \setminus \{1_{M_2}\})$ such that for $k = 2, \dots, m$, one has $a_k \in M_i \setminus \{1_{M_i}\}$ for $i \in \{1, 2\}$ if and only if $a_{k-1} \in M_{3-i} \setminus \{1_{M_{3-i}}\}$; when $m = 0$, we denote the empty word $a_1 \dots a_m$ by $1_{M_1 * M_2}$. Given $\mathbf{a} = a_1 \dots a_m$ and $\mathbf{b} = b_1 \dots b_n$ from $M_1 * M_2$ with $m, n \in \mathbb{N}$ and $a_1, \dots, a_m, b_1, \dots, b_n$ from $(M_1 \setminus \{1_{M_1}\}) \cup (M_2 \setminus \{1_{M_2}\})$ satisfying the condition above, the product $\mathbf{ab}$ can be defined inductively on $\min\{m, n\}$: we set $\mathbf{ab} = \mathbf{b}$ if $m = 0$ and $\mathbf{ab} = \mathbf{a}$ if $n = 0$; if $m, n > 0$, $a_m, b_1 \in M_i \setminus \{1_{M_i}\}$, and $a_m b_1 = 1_{M_i}$ in M_i for some $i \in \{1, 2\}$, then $\mathbf{ab}$ is the product of $\mathbf{a}' = a_1 \dots a_{m-1}$ and $\mathbf{b}' = b_2 \dots b_n$; in all the remaining cases, $\mathbf{ab}$ is the same as for the semigroup free product. Note that $1_{M_1 * M_2}$ is the identity element of $M_1 * M_2$.

As obvious from the defining presentations, the free factors S_1, S_2 are usually thought of as subsemigroups of $S_1 * S_2$. When S_1 is generated by A_1 and S_2 by A_2, this can be expressed by the existence of canonical semigroup monomorphisms $\iota_1 \colon S_1 \to S_1 * S_2$ and $\iota_2 \colon S_2 \to S_1 * S_2$ such that $\iota_1(a) = a$ for all $a \in A_1$ and $\iota_2(a) = a$ for all $a \in A_2$. The importance of semigroup free products mainly stems from the fact that given any semigroup T and homomorphisms $\varphi_1 \colon S_1 \to T$, $\varphi_2 \colon S_2 \to T$, there exists a unique semigroup homomorphism $\varphi \colon S_1 * S_2 \to T$ such that $\varphi_1 = \iota_1 \varphi$ and $\varphi_2 = \iota_2 \varphi$, as illustrated by the following commutative diagram.



The situation is much the same for monoid free products and monoid homomorphisms. Using the language of category theory, these properties can be concisely expressed by saying that semigroup and monoid free products take the role of coproducts in the category of semigroups and monoids, respectively; see, e.g., [2, Example 3.9].

3. Results

We now establish our actual results. Let us first address the following question of T. Brough, A. J. Cain, and M. Pfeiffer [4]: *Is the class of all finitely generated semigroups with a deterministic context-free word problem closed under free products?* This is also mentioned as an open problem by C.-F. Nyberg-Brodda [23].

The affirmative answer to this question is given by the following theorem – and it turns out that it can be obtained quite easily. T. Brough, A. J. Cain, and M. Pfeiffer [4] describe a construction based on pushdown automata showing that the class of all semigroups with a *context-free* word problem is closed under free products. They observe that their construction does not work for deterministic pushdown automata, which leads them to raising the above-reproduced question. However, only a slight modification of their construction appears to be sufficient for the deterministic case as well, as we show in the proof of the following theorem.

Theorem 3.1. *Let S_1, S_2 be finitely generated semigroups with a deterministic context-free word problem such that $S_1 \cap S_2 = \emptyset$. Then the word problem of the semigroup free product $S_1 * S_2$ is deterministic context-free as well.*

Proof. Let $A_1 \subseteq S_1$ be a finite generating set of the semigroup S_1, and $A_2 \subseteq S_2$ a finite generating set of S_2. As the statement of the theorem is trivial when S_1 or S_2 is empty, we may assume nonemptiness of A_1 and A_2. Let $\mathcal{A}_1, \mathcal{A}_2$ be deterministic pushdown automata recognising the languages $\mathrm{WP}_{A_1}(S_1)$ and $\mathrm{WP}_{A_2}(S_2)$, respectively. Assume without loss of generality that the bottom-of-pushdown symbols $\dashv_1$, $\dashv_2$ of these two automata are never pushed or popped during their runs, so that they remain forever on the bottom of the pushdown and appear only there. We construct a deterministic pushdown automaton $\mathcal{A}$ recognising the language $\mathrm{WP}_{A_1 \cup A_2}(S_1 * S_2)$.

The automaton $\mathcal{A}$ has to accept precisely the words

$$u_1 \dots u_n \# v_n^R \dots v_1^R,$$

where $n \in \mathbb{N} \setminus \{0\}$ and there exists a mapping $\sigma \colon [n] \to \{1, 2\}$ satisfying $\sigma(k) = 3 - \sigma(k-1)$ for $k = 2, \dots, n$ such that $u_k, v_k \in A_{\sigma(k)}^+$ and $u_k =_{S_{\sigma(k)}} v_k$ holds for $k = 1, \dots, n$.

As the class $\mathscr{L}(\mathrm{detCF})$ is closed under intersection with a rational language, for which we can take

$$A_1(A_1 \cup A_2)^*\#(A_1 \cup A_2)^*A_1 \cup \\ \cup\ A_2(A_1 \cup A_2)^*\#(A_1 \cup A_2)^*A_2,$$

we may actually assume that the input words of $\mathcal{A}$ all take the form

$$u_1 \dots u_m \# v_n^R \dots v_1^R$$

for $m, n \in \mathbb{N} \setminus \{0\}$, $u_k \in A_{\sigma(k)}^+$ for $k = 1, \dots, m$, and $v_\ell \in A_{\sigma(\ell)}^+$ for $\ell = 1, \dots, n$, where $\sigma\colon [\max\{m,n\}] \to \{1,2\}$ is some mapping satisfying $\sigma(k) = 3 - \sigma(k-1)$ for $k = 2, \dots, \max\{m,n\}$. The automaton $\mathcal{A}$ thus only needs to check whether $m = n$ and $u_k =_{S_{\sigma(k)}} v_k$ for $k = 1, \dots, n$.

This is done as follows: when $\mathcal{A}$ reads the first symbol from the input, this surely belongs to A_i for some $i \in \{1,2\}$. The automaton $\mathcal{A}$ can thus determine $\sigma(1)$, push $\dashv_{\sigma(1)}$ above its own bottom-of-pushdown symbol $\dashv$, and simulate the run of $\mathcal{A}_{\sigma(1)}$ on u_1. Whenever a simulation of $\mathcal{A}_{\sigma(j)}$ on some u_j for $j \in [m-1]$ is finished and a symbol from $A_{\sigma(j+1)}$ is encountered, the current state of the simulated automaton $\mathcal{A}_{\sigma(j)}$ is pushed on the pushdown; in case $j = 1$, the symbol representing this state on the pushdown is also marked as corresponding to the first factor u_1. After the state is pushed, the automaton $\mathcal{A}$ pushes $\dashv_{\sigma(j+1)}$ and continues with a new simulation of $\mathcal{A}_{\sigma(j+1)}$ on the next factor u_{j+1}, starting with the initial configuration of $\mathcal{A}_{\sigma(j+1)}$.

After reading $\#$, the automaton $\mathcal{A}$ continues in the simulation of $\mathcal{A}_{\sigma(m)}$ on u_m by treating $\# v_n^R$ as the rest of its input in case $\sigma(n) = \sigma(m)$; the run of $\mathcal{A}$ rejects when $\sigma(n) \neq \sigma(m)$. Assume that $\sigma(n) = \sigma(m)$ and $\mathcal{A}_{\sigma(m)}$ successfully goes through $\# v_n^R$. Then, whenever a simulation of $\mathcal{A}_{\sigma(k)}$ gets to the end of v_k^R for some $k \in [n]$, the automaton $\mathcal{A}$ rejects if the simulated automaton $\mathcal{A}_{\sigma(k)}$ does not accept its input. If the simulated automaton accepts its input and a symbol from $A_{3-\sigma(k)}$ is left on the input of $\mathcal{A}$, the automaton $\mathcal{A}$ pops all the symbols from the pushdown alphabet of $\mathcal{A}_{\sigma(k)}$ that find themselves on the top of the pushdown. If the bottom-of-pushdown symbol $\dashv$ of $\mathcal{A}$ is encountered after doing so, the automaton $\mathcal{A}$ rejects. Otherwise the automaton $\mathcal{A}$ can recover the state of $\mathcal{A}_{\sigma(k-1)}$ that has been stored on the pushdown below the last popped symbol $\dashv_{\sigma(k)}$. After this is performed, $\mathcal{A}$ continues in the simulation of $\mathcal{A}_{\sigma(k-1)}$, from the configuration given by this state and the contents of its pushdown that now appear on the top of the pushdown store of $\mathcal{A}$, on the next factor v_{k-1}^R.

For m being the number of factors $u_1, \dots, u_m$ to the left of $\#$, the automaton can easily find out when it is reading the m-th factor v_k^R to the right of $\#$. If $m \geq 2$, this can be noticed when a marked state corresponding to u_1 is popped from the pushdown; for $m = 1$, the automaton $\mathcal{A}$ can simply remember that it never switched between the simulated automata. In any case, the information about reading the m-th factor v_k^R to the right of $\#$ can be kept in the state of $\mathcal{A}$. This means that it is enough for the automaton $\mathcal{A}$ to get into an accepting state precisely when it is reading the said m-th factor to the right of $\#$ and the simulated automaton $\mathcal{A}_{\sigma(1)}$ is in an accepting state. □

The construction described in the proof of the preceding theorem is almost identical to the construction for nondeterministic pushdown automata given in [4]. The only modification is that the first state of a simulated automaton pushed to the pushdown store of $\mathcal{A}$ is marked, so that the deterministic pushdown automaton can use it to decide when it should accept (the impossibility to do so for the original construction was noted in [4]).

Let us now observe that the situation is much different for monoids. Although T. Brough, A. J. Cain, and M. Pfeiffer [4] proved that the class of all finitely generated monoids with a context-free word problem is closed under monoid free products, we now observe that this property no longer holds for finitely generated monoids with a *deterministic* context-free word problem.

Theorem 3.2. *There are finitely generated monoids M_1, M_2 with deterministic context-free word problems such that the word problem of the monoid free product $M_1 * M_2$ is not deterministic context-free.*

Proof. Consider the monoids

$$M_1 = \langle a, b \mid ba = a \rangle$$

and

$$M_2 = \langle c, d \mid cd = dc, cd = 1 \rangle,$$

the latter being isomorphic to $\mathbb{Z}$, and their generating sets $A = \{a, b\}$ and $C = \{c, d\}$. As M_1 is

presented by a finite complete rewriting system $ba \to a$, we obtain

$$\begin{aligned}\mathrm{WP}_A(M_1) = \{&b^n\#b^n \mid n \in \mathbb{N}\} \cup \\ \cup\ \{&ub^n\#b^nv \mid n \in \mathbb{N};\ u \in A^*a;\ v \in aA^*; \\ &|u|_a = |v|_a\}.\end{aligned}$$

Moreover, clearly

$$\begin{aligned}\mathrm{WP}_C(M_2) = \{&u\#v^R \mid u, v \in C^*; \\ &|u|_c - |u|_d - |v|_c + |v|_d = 0\}.\end{aligned}$$

Both these languages are deterministic context-free. This is obvious for the language $\mathrm{WP}_C(M_2)$. A deterministic pushdown automaton recognising $\mathrm{WP}_A(M_1)$ can operate as follows: when b is read from the input before # is encountered, push it on the pushdown. When a is read under these circumstances, first pop (using transitions on ε) all the b's on the top of the pushdown in case there are some, and only then push the letter a. After # is read, the automaton can first read several b's, popping a b from the pushdown for each. If the bottom-of-pushdown symbol finds itself on the top of the pushdown at this point, the automaton can accept. If a is on the top of the pushdown, the automaton can pop it while reading a from the input and switching to another state, in which it reads a's from the input while popping one a from the pushdown for each, and ignores any b on the input. When the bottom-of-pushdown symbol finally finds itself on the top of the pushdown, the automaton can accept.

We show that the language $\mathrm{WP}_{A\cup C}(M_1 * M_2)$ is not deterministic context-free.

Assume for the purpose of contradiction that $\mathrm{WP}_{A\cup C}(M_1 * M_2) \in \mathscr{L}(\mathrm{detCF})$. Then by closure of $\mathscr{L}(\mathrm{detCF})$ under intersection with a rational language,

$$\begin{aligned}L := \mathrm{WP}_{A\cup C}(M_1 * M_2) \cap \\ \cap\ c^+a^+\#(a^+c^+ \cup a^+c^+d^+bc^+)\end{aligned}$$

has to be deterministic context-free as well. As $u, v \in c^+a^+$ satisfy $u =_{M_1*M_2} v$ if and only if $u = v$, while $u \in c^+a^+$ and $v \in c^+bd^+c^+a^+$ satisfy $u =_{M_1*M_2} v$ if and only if $u = c^ma^n$ and $v = c^mbd^kc^ka^n$ for some $m, n, k \in \mathbb{N} \setminus \{0\}$, we actually get

$$\begin{aligned}L = \{&c^ma^n\#a^nc^m \mid m, n \in \mathbb{N} \setminus \{0\}\} \cup \\ \cup\ \{&c^ma^n\#a^nc^kd^kbc^m \mid m, n, k \in \mathbb{N} \setminus \{0\}\}.\end{aligned}$$

Let $\mathcal{A}$ be a deterministic pushdown automaton recognising the language L.

Without loss of generality, we may assume that $\mathcal{A}$ contains no transitions upon ε leading from final states (see, e.g., [13, Corollary on p. 239]). Let $\mathcal{B}$ be a deterministic pushdown automaton obtained by modifying $\mathcal{A}$ such that the *first* occurrence of d can only be read from a final state of $\mathcal{A}$. This implies that the first d can only follow after reading a prefix $c^ma^n\#a^nc^m$ for some $m, n \in \mathbb{N} \setminus \{0\}$, so that $\mathcal{B}$ can only accept words of the form $c^ma^n\#a^nc^m$ or $c^ma^n\#a^nc^md^mbc^m$ for $m, n \in \mathbb{N} \setminus \{0\}$. On the other hand, as $\mathcal{A}$ is deterministic without ε-transitions from final states and as $\mathcal{A}$ recognises L, both $\mathcal{A}$ and $\mathcal{B}$ always find themselves in a final state after reading a prefix $c^ma^n\#a^nc^m$ for $m, n \in \mathbb{N} \setminus \{0\}$ and going through all subsequent ε-transitions. This means that $\mathcal{B}$ actually accepts $c^ma^n\#a^nc^m$ for all $m, n \in \mathbb{N} \setminus \{0\}$ and that it can also continue by a transition upon d after reading $c^ma^n\#a^nc^m$, while in the latter case it eventually accepts whenever $\mathcal{A}$ accepts. As a result, $\mathcal{B}$ accepts *precisely all* words $c^ma^n\#a^nc^m$ and $c^ma^n\#a^nc^md^mbc^m$ for $m, n \in \mathbb{N} \setminus \{0\}$.

A deterministic pushdown automaton $\mathcal{B}'$ obtained by modifying $\mathcal{B}$ to accept only if at least one d was actually read thus recognises the language

$$L' := \{c^ma^n\#a^nc^md^mbc^m \mid m, n \in \mathbb{N} \setminus \{0\}\}.$$

However, this language clearly is not context-free – a contradiction. □

Remark 3.3. As the class of all finitely generated monoids with a context-free word problem is closed under monoid free products [4], the word problem of the free product $M_1 * M_2$ of monoids from the preceding proof is context-free – however, nondeterminism is crucial in the construction of the pushdown automaton for this word problem described in [4], where it is used to guess factors of the input word evaluating to identity elements in M_1 and M_2. On the other hand, our proof of Theorem 3.2 heavily relies on the assumption of the pushdown automaton $\mathcal{A}$ being deterministic. If this was not the case, the automaton $\mathcal{B}$ constructed from $\mathcal{A}$ would not have to recognise the said language, as there would be no guarantee that an accepting run of $\mathcal{A}$ upon $c^ma^n\#a^nc^md^mbc^m$ has to find itself in a final state after reading the prefix $c^ma^n\#a^nc^m$ – the automaton $\mathcal{A}$ could accept $c^ma^n\#a^nc^m$ via some different run.

The class of all finitely generated *groups* with a deterministic context-free word problem coincides

with the class of all finitely generated groups with a context-free word problem [12], and as such it is closed under group free products. This means that $\mathscr{L}(\text{detCF})$ gives an example of a class of languages $\mathcal{C}$ closed under inverse homomorphisms such that the classes of finitely generated semigroups and groups with a word problem in $\mathcal{C}$ are closed under free products (of semigroups and groups, respectively), while the class of finitely generated monoids with a word problem in $\mathcal{C}$ is not closed under monoid free products. Existence of such a class happens to answer one of the questions of C.-F. Nyberg-Brodda [23].

While a context-free word problem of a finitely generated group is necessarily deterministic context-free, it is an open problem whether the same holds for all finitely generated cancellative monoids with a context-free word problem. As noticed by one of the anonymous reviewers, the counterexample from the proof of Theorem 3.2 shows at least that this property does not hold for all finitely generated left-cancellative monoids with a context-free word problem: the monoids M_1 and M_2 are both left-cancellative with a context-free word problem – their free product $M_1 * M_2$ is thus again left-cancellative [8] and has a context-free word problem [4]; however, the word problem of $M_1 * M_2$ was observed not to be deterministic context-free.

Theorem 3.2 is an example of a result showing that properties of semigroups and monoids with a deterministic context-free word problem might be slightly less satisfying compared to the case of context-free word problems. In fact, the very definition of the word problem of a semigroup or a monoid becomes less robust when deterministic context-free word problems are considered. For instance, one of the anonymous reviewers noted that the word problem

$$\mathrm{WP}_A(M) = \{u\#v^R \mid u, v \in A^*;\ u =_M v\}$$

of a monoid M finitely generated by A is context-free if and only if the language

$$\mathrm{WP}'_A(M) = \{u^R\#v \mid u, v \in A^*;\ u =_M v\}$$

– *i.e.*, an "equally good candidate" for the definition of the word problem – is context-free (and the same is true for semigroups).[2] This reviewer asked whether the same holds for semigroups and monoids with a *deterministic* context-free word problem as well. We now give a negative answer to this question.

Proposition 3.4. *There is a monoid M generated by a finite set A such that $\mathrm{WP}_A(M) \in \mathscr{L}(\text{detCF})$, but $\mathrm{WP}'_A(M) \notin \mathscr{L}(\text{detCF})$.*

Proof. Let us consider $A = \{a, b\}$, and let M be presented by

$$M = \langle a, b \mid aa = a, ba = a\rangle.$$

As this is equivalently presented by a finite complete rewriting system $aa \to a, ba \to a$, we get

$$\begin{aligned}\mathrm{WP}_A(M) = {} & \{b^n\#b^n \mid n \in \mathbb{N}\} \cup \\ & \cup\ \{ub^n\#b^nv \mid n \in \mathbb{N};\ u \in A^*a;\ v \in aA^*\},\end{aligned}$$

which is clearly a deterministic context-free language. However, the language

$$\begin{aligned}\mathrm{WP}'_A(M) = {} & \{b^n\#b^n \mid n \in \mathbb{N}\} \cup \\ & \cup\ \{b^nu\#vb^n \mid n \in \mathbb{N};\ u \in aA^*;\ v \in A^*a\}\end{aligned}$$

is not deterministic context-free, as otherwise the language

$$\begin{aligned}\mathrm{WP}'_A(M) \cap b^+a\#\left(ab^+ \cup ab^+ab^+\right) = {} & \\ = \{b^na\#ab^n \mid n \in \mathbb{N} \setminus \{0\}\} \cup & \\ \cup\, \{b^na\#ab^mab^n \mid m, n \in \mathbb{N} \setminus \{0\}\} & \end{aligned}$$

would be in $\mathscr{L}(\text{detCF})$ as well. Similarly as in the proof of Theorem 3.2, a deterministic pushdown automaton recognising this language could be assumed to contain no transitions upon ε leading from final states, and modified such that the third occurrence of a can only be read from a final state. The resulting deterministic pushdown automaton would then recognise the language

$$L = \{b^na\#(ab^n)^k \mid n \in \mathbb{N} \setminus \{0\};\ k \in \{1, 2\}\},$$

which is obviously not context-free – a contradiction. □

One can argue almost identically when semigroups are considered instead of monoids.

[2] This follows easily, e.g., by the closure of the class of all context-free languages under rational transductions, intersection with a rational language, and conjugacy – the latter being defined for all alphabets A and languages $L \subseteq A^*$ by $\mathrm{CYCLE}(L) = \{uv \mid u, v \in A^*;\ vu \in L\}$ [13].

Acknowledgements

I would like to thank the anonymous reviewers for their helpful comments, suggestions, and questions raised.

References

[1] A. V. Anisimov. Group languages. *Kibernetika*, 4:18–24, 1971.

[2] S. Awodey. *Category Theory*. Oxford University Press, 2nd edition, 2010.

[3] T. Brough. Groups with poly-context-free word problem. *Groups – Complexity – Cryptology*, 6(1):9–29, 2014.

[4] T. Brough, A. J. Cain, and M. Pfeiffer. Context-free word problem semigroups. In *Developments in Language Theory, DLT 2019*, pages 292–305, 2019.

[5] A. Duncan and R. H. Gilman. Word hyperbolic semigroups. *Mathematical Proceedings of the Cambridge Philosophical Society*, 136(3):513–524, 2004.

[6] M. J. Dunwoody. The accessibility of finitely presented groups. *Inventiones mathematicae*, 81:449–457, 1985.

[7] M. Elder, M. Kambites, and G. Ostheimer. On groups and counter automata. *International Journal of Algebra and Computation*, 18(8):1345–1364, 2008.

[8] J. Fountain and M. Kambites. Graph products of right cancellative monoids. *Journal of the Australian Mathematical Society*, 87(2):227–252, 2009.

[9] T. Herbst. On a subclass of context-free groups. *RAIRO – Informatique Théorique et Applications*, 25(3):255–272, 1991.

[10] M. Hoffmann, D. F. Holt, M. D. Owens, and R. M. Thomas. Semigroups with a context-free word problem. In *Developments in Language Theory, DLT 2012*, pages 97–108, 2012.

[11] D. F. Holt, M. D. Owens, and R. M. Thomas. Groups and semigroups with a one-counter word problem. *Journal of the Australian Mathematical Society*, 85(2):197–209, 2008.

[12] D. F. Holt, S. Rees, and C. E. Röver. *Groups, Languages and Automata*. Cambridge University Press, 2017.

[13] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[14] J. M. Howie. *Automata and Languages*. Clarendon Press, 1991.

[15] J. M. Howie. *Fundamentals of Semigroup Theory*. Clarendon Press, 1995.

[16] T. W. Hungerford. *Algebra*. Springer, 1974.

[17] M. Kambites. Word problems recognisable by deterministic blind monoid automata. *Theoretical Computer Science*, 362(1–3):232–237, 2006.

[18] R. P. Kropholler and D. Spriano. Closure properties in the class of multiple context-free groups. *Groups – Complexity – Cryptology*, 11(1):1–15, 2019.

[19] S. R. Lakin and R. M. Thomas. Context-sensitive decision problems in groups. In *Developments in Language Theory, DLT 2004*, pages 296–307. Springer, 2004.

[20] S. R. Lakin and R. M. Thomas. Space complexity and word problems of groups. *Groups – Complexity – Cryptology*, 1(2):261–273, 2009.

[21] D. E. Muller and P. E. Schupp. Groups, the theory of ends, and context-free languages. *Journal of Computer and System Sciences*, 26(3):295–310, 1983.

[22] C.-F. Nyberg-Brodda. On the word problem for special monoids. *Semigroup Forum*, 105:295–327, 2022.

[23] C.-F. Nyberg-Brodda. On the word problem for free products of semigroups and monoids. *Journal of Algebra*, 622:721–741, 2023.