The Post Correspondence Problem and Equalisers for Certain Free Group and Monoid Morphisms

Laura Ciobanu
Heriot-Watt University, Edinburgh, Scotland, UK
l.ciobanu@hw.ac.uk

Alan D. Logan
Heriot-Watt University, Edinburgh, Scotland, UK
a.logan@hw.ac.uk

Abstract

A marked free monoid morphism is a morphism for which the image of each generator starts with a different letter, and immersions are the analogous maps in free groups. We show that the (simultaneous) PCP is decidable for immersions of free groups, and provide an algorithm to compute bases for the sets, called equalisers, on which the immersions take the same values. We also answer a question of Stallings about the rank of the equaliser.

Analogous results are proven for marked morphisms of free monoids.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory; Theory of computation → Complexity classes; Mathematics of computing → Combinatorics on words; Mathematics of computing → Combinatorial algorithms

Keywords and phrases Post Correspondence Problem, marked map, immersion, free group, free monoid

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.120

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version https://arxiv.org/abs/2002.07574

Funding Research supported by EPSRC grant EP/R035814/1.

1 Introduction

In this paper we prove results about the classical Post Correspondence Problem ($\mathrm{PCP}_{\mathrm{FM}}$), which we state in terms of equalisers of free monoid morphisms, and the analogue problem $\mathrm{PCP}_{\mathrm{FG}}$ for free groups ([5], [19]), and we describe the solutions to $\mathrm{PCP}_{\mathrm{FM}}$ and $\mathrm{PCP}_{\mathrm{FG}}$ for certain classes of morphisms. While the classical $\mathrm{PCP}_{\mathrm{FM}}$ is famously undecidable for arbitrary maps of free monoids [21] (see also the survey [12] and the recent result of Neary [20]), $\mathrm{PCP}_{\mathrm{FG}}$ for free groups is an important open question [8, Problem 5.1.4]. Additionally, for both free monoids and free groups there are only few results describing algebraically the solutions to classes of instances known to have decidable $\mathrm{PCP}_{\mathrm{FM}}$ or $\mathrm{PCP}_{\mathrm{FG}}$. Our results apply to *marked* morphisms in the monoid case, and to their counterparts in free groups, called *immersions*. Marked morphisms are the key tool used in resolving the $\mathrm{PCP}_{\mathrm{FM}}$ for the free monoid of rank two [9], and therefore understanding the solutions to the $\mathrm{PCP}_{\mathrm{FG}}$ for immersions is an important step towards resolving the $\mathrm{PCP}_{\mathrm{FG}}$ for the free group of rank two. The density of marked morphisms and immersions among all the free monoid or group maps is strictly positive (Section 10), so our results concern a significant proportion of instances.

An *instance* of the $\mathrm{PCP}_{\mathrm{FM}}$ is a tuple $I = (\Sigma, \Delta, g, h)$, where Σ, Δ are finite alphabets, Σ^*, Δ^* are the respective free monoids, and $g, h : \Sigma^* \to \Delta^*$ are morphisms. The *equaliser* of g, h is $\mathrm{Eq}(g, h) = \{x \in \Sigma^* \mid g(x) = h(x)\}$. The $\mathrm{PCP}_{\mathrm{FM}}$ is the decision problem:

47th International Colloquium on Automata, Languages, and Programming (ICALP 2020).
Editors: Artur Czumaj, Anuj Dawar, and Emanuela Merelli; Article No. 120; pp. 120:1–120:16
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$\mathbf{PCP_{FM}}$: Given $I = (\Sigma, \Delta, g, h)$, is the equaliser $\mathrm{Eq}(g, h)$ trivial?

Analogously, an instance of the $\mathrm{PCP_{FG}}$ is a four-tuple $I = (\Sigma, \Delta, g, h)$ with $g, h : F(\Sigma) \to F(\Delta)$ morphisms between the free groups $F(\Sigma)$ and $F(\Delta)$, and $\mathrm{PCP_{FG}}$ is the decision problem pertaining to the similarly defined $\mathrm{Eq}(g, h)$ in free groups.

Beyond $\mathrm{PCP_{FM}}$, in this paper we also consider the *Algorithmic Equaliser Problem*, denoted $\mathrm{AEP_{FM}}$ (or $\mathrm{AEP_{FG}}$ in the group case), which for an instance $I = (\Sigma, \Delta, g, h)$ with g, h free monoid morphisms (or free group morphisms for $\mathrm{AEP_{FG}}$), says:

$\mathbf{AEP_{FM}}$: Given $I = (\Sigma, \Delta, g, h)$, output

(a) a finite basis for $\mathrm{Eq}(g, h)$, or

(b) a finite automaton recognising the set $\mathrm{Eq}(g, h)$.

If a finite basis or finite automaton for $\mathrm{Eq}(g, h)$ does not exist then Part (a) or (b), respectively, of the problem is insoluble. Note that (a) and (b) are connected: for free groups these two problems are in fact the same when $\mathrm{Eq}(g, h)$ is finitely generated, while for free monoids (a) implies (b). Part (a) of the $\mathrm{AEP_{FM}}$ is known to be soluble when $|\Sigma| = 2$ and one of g or h is non-periodic, and insoluble otherwise [13] [12, Corollary 6].

Sets of morphisms. We are particularly interested in sets S of morphisms (not just two morphisms f, g) and their equalisers $\mathrm{Eq}(S) = \bigcap_{g,h\in S} \mathrm{Eq}(g, h)$, and we prove structural results for arbitrary sets and algorithmic results for finite sets. Our results resolve the *simultaneous* $\mathrm{PCP_{FG}}$ *and* $\mathrm{PCP_{FM}}$ for immersions and marked morphisms; these problems take as input a finite set S of maps and ask the same questions about equalisers as in the classical setting. Analogously, one could further define the "simultaneous $\mathrm{AEP_{FG}}$ and $\mathrm{AEP_{FM}}$". However, the simultaneous $\mathrm{AEP_{FG}}$ is equivalent to the $\mathrm{AEP_{FG}}$, and Part (b) of the simultaneous $\mathrm{AEP_{FM}}$ is equivalent to Part (b) of the $\mathrm{AEP_{FM}}$, as follows. As bases of intersections of finitely generated subgroups of free groups are computable (and as Parts (a) and (b) of the $\mathrm{AEP_{FG}}$ are equivalent), if the $\mathrm{AEP_{FG}}$ is soluble for a class $\mathcal{C}$ of maps then there exists an algorithm with input a finite set S of morphisms from $F(\Sigma)$ to $F(\Delta)$, $S \subseteq \mathcal{C}$, and output a basis for $\mathrm{Eq}(S)$. Similarly, automata accepting intersections of regular languages are computable, and so if Part (b) of the $\mathrm{AEP_{FM}}$ is soluble for a class $\mathcal{C}$ of maps then there exists an algorithm with input a finite set S of morphisms from Σ^* to Δ^*, $S \subseteq \mathcal{C}$, and output a finite automaton whose language is $\mathrm{Eq}(S) = \bigcap_{g,h\in S} \mathrm{Eq}(g, h)$.

Main results. A set of words $\mathbf{s} \subseteq \Delta^*$ is *marked* if any two distinct $u, v \in \mathbf{s}$ start with a different letter of Δ, which implies $|\mathbf{s}| \leqslant |\Delta|$. A free monoid morphism $f : \Sigma^* \to \Delta^*$ is *marked* if the set $f(\Sigma)$ is marked. An *immersion of free groups* is a morphism $f : F(\Sigma) \to F(\Delta)$ where the set $f(\Sigma \cup \Sigma^{-1})$ is marked (see Section 3 for equivalent formulations). Halava, Hirvensalo and de Wolf [11] showed that $\mathrm{PCP_{FM}}$ is decidable for marked morphisms; inspired by their methods we were able to obtain stronger results (Theorem A) for this kind of map, as well as expand to the world of free groups (Theorem C), where we employ "finite state automata"-like objects called *Stallings graphs.*

▶ **Theorem A.** *If S is a set of marked morphisms from Σ^* to Δ^*, then there exists a finite alphabet Σ_S and a marked morphism $\psi_S : \Sigma_S^* \to \Sigma^*$ such that* $\mathrm{Image}(\psi_S) = \mathrm{Eq}(S)$*. Moreover, for S finite, there exists an algorithm with input S and output the marked morphism ψ_S.*

▶ **Corollary B.** *The simultaneous* $\mathrm{PCP_{FM}}$ *is decidable for marked morphisms of free monoids.*

▶ **Theorem C.** *If S is a set of immersions from $F(\Sigma)$ to $F(\Delta)$, then there exists a finite alphabet Σ_S and an immersion $\psi_S : F(\Sigma_S) \to F(\Sigma)$ such that* $\mathrm{Image}(\psi_S) = \mathrm{Eq}(S)$. *Moreover, when S is finite, there exists an algorithm with input S and output the immersion ψ_S.*

▶ **Corollary D.** *The simultaneous* $\mathrm{PCP}_{\mathrm{FG}}$ *is decidable for immersions of free groups.*

The Equaliser Conjecture. Our work was partially motivated by Stallings' Equaliser Conjecture for free groups, which dates from 1984 [22, Problems P1 & 5] (also [7, Problem 6] [24, Conjecture 8.3] [1, Problem F31]). Here $\mathrm{rk}(H)$ stands for the *rank*, or minimum number of generators, of a subgroup H:

▶ **Conjecture 1** (The Equalizer Conjecture, 1984)**.** *If $g, h : F(\Sigma) \to F(\Delta)$ are injective morphisms then* $\mathrm{rk}(\mathrm{Eq}(g, h)) \leqslant |\Sigma|$.

This conjecture has its roots in "fixed subgroups" $\mathrm{Fix}(\phi)$ of free group endomorphisms $\phi : F(\Sigma) \to F(\Sigma)$ (if $\Sigma = \Delta$ then $\mathrm{Fix}(\phi) = \mathrm{Eq}(\phi, \mathrm{id})$), where Bestvina and Handel proved that $\mathrm{rk}(\mathrm{Fix}(\phi)) \leqslant |\Sigma|$ for ϕ an automorphism [3], and Imrich and Turner extended this bound to all endomorphisms [14]. Bergman further extended this bound to all sets of endomorphisms [2]. Like Bergman's result, our first corollary of Theorem C considers sets of immersions, which are injective, and answers Conjecture 1 for immersions.

▶ **Corollary E.** *If S is a set of immersions from $F(\Sigma)$ to $F(\Delta)$ then* $\mathrm{rk}(\mathrm{Eq}(S)) \leqslant |\Sigma|$.

In free monoids, equalisers of injections are free [12, Corollary 4] but they are not necessarily regular languages (and hence not necessarily finitely generated) [12, Example 6]. In order to understand equalisers $\mathrm{Eq}(S)$ of sets of maps we need to understand intersections in free monoids. Recall that the intersection $A^* \cap B^*$ of two finitely generated free submonoids of a free monoid Σ^* is free [23] and one can find a regular expression that represents a basis of $A^* \cap B^*$ [4]. However, the intersection is not necessarily finitely generated [17]. The following result is surprising because we have finite generation, even for the intersection $\mathrm{Eq}(S) = \bigcap_{g,h \in S} \mathrm{Eq}(g, h)$.

▶ **Corollary F.** *If S is a set of marked morphisms from Σ^* to Δ^* then $\mathrm{Eq}(S)$ is a free monoid with* $\mathrm{rk}(\mathrm{Eq}(S)) \leqslant |\Sigma|$.

The Algorithmic Equaliser Problem. The $\mathrm{AEP}_{\mathrm{FG}}$ is insoluble in general, as equalisers in free groups are not necessarily finitely generated [24, Section 3], and is an open problem of Stallings' if both maps are injective [22, Problems P3 & 5]. Our next corollary of Theorem C resolves this open problem for immersions.

▶ **Corollary G.** *The* $\mathrm{AEP}_{\mathrm{FG}}$ *is soluble for immersions of free groups.*

The $\mathrm{AEP}_{\mathrm{FM}}$ is insoluble in general, primarily as equalisers are not necessarily regular languages [10, Example 4.6]. Even for maps whose equalisers form regular languages, the problem remains insoluble [18]. Another corollary of Theorem A is the following.

▶ **Corollary H.** *The* $\mathrm{AEP}_{\mathrm{FM}}$ *is soluble for marked morphisms of free monoids.*

Outline of the article. In Section 2 we prove Theorem A and its corollaries about free monoids. The remainder of the paper focuses on free groups, where the central result is Theorem 18, which is Theorem C for $|S| = 2$. In Section 3 we reformulate immersions in terms of Stallings' graphs, and in Section 4 define the "reduction" $I' = (\Sigma', \Delta', g', h')$ of an instance $I = (\Sigma, \Delta, g, h)$ of the $\mathrm{AEP}_{\mathrm{FG}}$ for immersions. Repeatedly computing reductions is the key process in our algorithm. In Section 5 we prove the process of reduction decreases the "prefix complexity" of an instance (so the word "reduction" makes sense), and in Section 6 we prove Theorem 18, mentioned above. In Section 7 we prove Theorem C and its corollaries. In Section 9 we give a complexity analysis for both our free monoid and free group algorithms, and in Section 10 we show that the density of marked morphisms and immersions among all the free monoid or group maps is strictly positive.

2 Marked morphisms in free monoids

In this section we prove Theorem A and its corollaries. We use the following immediate fact.

▶ **Lemma 2.** *Marked morphisms of free monoids are injective.*

Proof. Let $f : \Sigma^* \to \Delta^*$ be marked and let $x \neq y$ be nontrivial. One can write $x = zax'$ and $y = zby'$, where $a, b \in \Sigma$ are the first letter where x and y differ. As f is marked, $f(a) \neq f(b)$, hence $f(x) = f(z)f(a)f(x') \neq f(z)f(b)f(y') = f(y)$, so f is injective. ◀

We may assume $\Sigma \subseteq \Delta$, as $|\Sigma| \leqslant |\Delta|$ holds whenever $f : \Sigma^* \to \Delta^*$ is marked.

Consider morphisms $g : \Sigma_1^* \to \Delta^*$ and $h : \Sigma_2^* \to \Delta^*$. The set of non-empty words over an alphabet Σ is denoted Σ^+. For $a \in \Delta$, a pair $(u, v) \in \Sigma_1^+ \times \Sigma_2^+$ is an *a-block* if (i) $g(u) = h(v)$ starts with a, and (ii) u and v are minimal, that is, the length $|g(u)| = |h(v)|$ is minimal among all such pairs. If the pair (g, h) has blocks $a_i = (u_i, v_i)$, $1 \leqslant i \leqslant m$, then let Σ' be the alphabet consisting of these blocks and define $g' : (\Sigma')^* \mapsto \Sigma_1^*$ by $g'(a_i) = u_i$ and $h' : (\Sigma')^* \mapsto \Sigma_2^*$ by $h'(a_i) = v_i$. These maps are computable and, by an identical logic to [11, Section 2], are seen to be marked. Then $gg' = hh'$, and we let $k = gg' = hh'$ (so $k : (\Sigma')^* \to \Delta^*$). Since k is the composition of marked morphisms, it is itself marked. We therefore have the following.

▶ **Lemma 3.** *If $g : \Sigma_1^* \to \Delta^*$ and $h : \Sigma_2^* \to \Delta^*$ are marked morphisms then the corresponding maps $g' : \Sigma'^* \to \Sigma_1^*$, $h' : \Sigma'^* \to \Sigma_2^*$ and $k : \Sigma'^* \to \Delta^*$, $k = gg' = hh'$, are marked and are computable.*

The *reduction* of an instance $I = (\Sigma, \Delta, g, h)$ of the marked $\mathrm{PCP}_{\mathrm{FM}}$, as defined in [11], is the instance $I' := (\Sigma', \Delta, g', h')$ where Σ' is defined as above, and where g' and h' are as above, but with codomain Δ (which we may do as $\Sigma \subseteq \Delta$). We additionally assume that $\Sigma' \subseteq \Sigma$; we can do this as $|\Sigma'| \leqslant |\Sigma|$ by Lemma 3.

The following relies on [11, Lemma 1], which we strengthen by replacing the notion of "equivalence" with that of "strong equivalence": Two instances I_1 and I_2 of the $\mathrm{PCP}_{\mathrm{FM}}$ are *strongly equivalent* if their equalisers are isomorphic, which we write as $\mathrm{Eq}(I_1) \cong \mathrm{Eq}(I_2)$.

▶ **Lemma 4.** *Let $I' = (\Sigma', \Delta', g', h')$ be the reduction of $I = (\Sigma, \Delta, g, h)$ where g and h are marked. Then I and I' are strongly equivalent, and $g'(\mathrm{Eq}(I')) = \mathrm{Eq}(I) = h'(\mathrm{Eq}(I'))$.*

Proof. Firstly, note that $g'(\mathrm{Eq}(I')) \leqslant \mathrm{Eq}(I)$ [11, Lemma 1, paragraph 2]. From [11, Lemma 1, paragraph 1] it follows that $g'(\mathrm{Eq}(I')) \geqslant \mathrm{Eq}(I)$, so $g'(\mathrm{Eq}(I')) = \mathrm{Eq}(I)$. As g' is injective, the map $g'|_{\mathrm{Eq}(I')}$ is an isomorphism. Hence, I and I' are strongly equivalent, and, by symmetry for the h' map, $g'(\mathrm{Eq}(I')) = \mathrm{Eq}(I) = h'(\mathrm{Eq}(I'))$ as required. ◀

We can now improve the existing result on the marked $\mathrm{PCP}_{\mathrm{FM}}$. We store a morphism $f : \Sigma^* \to \Delta^*$ as a list $(f(a))_{a \in \Sigma}$.

▶ **Theorem 5.** *If $I = (\Sigma, \Delta, g, h)$ is an instance of the marked* $\mathrm{PCP_{FM}}$ *then there exists an alphabet $\Sigma_{g,h}$ and a marked morphism $\psi_{g,h} : \Sigma_{g,h}^* \to \Sigma^*$ such that* $\mathrm{Image}(\psi_{g,h}) = \mathrm{Eq}(I)$. *Moreover, there exists an algorithm with input I and output the marked morphism $\psi_{g,h}$.*

Proof. We explain the algorithm, and note at the end that the output is a marked morphism $\psi_{g,h} : F(\Sigma_{g,h}) \to F(\Sigma)$ with the required properties, and so the result follows.

Begin by making reductions $I_0, I_1, I_2, \ldots$, starting with $I_0 = I = (\Sigma, \Delta, g, h)$, the input instance. Then by [11, Section 5, paragraph 1] we will obtain an instance $I_j = (\Sigma_j, \Delta, g_j, h_j)$ such that one of the following will occur:

1. $|\Sigma_j| = 1$.
2. $|g_j(a)| = 1 = |h_j(a)|$ for all $a \in \Sigma_j$.
3. There exists some $i < j$ with $I_i = I_j$ (sequence starts cycling).

Keeping in mind the fact that reductions preserve equalisers (Lemma 4), we obtain in each case a subset $\Sigma_{g,h}$ (possibly empty) which forms a basis for $\mathrm{Eq}(I_j)$: For Case (1), writing $\Sigma_j = \{a\}$, the result holds as if $g(a^i) = h(a^i)$ then $g(a)^i = h(a)^i$ and so $g(a) = h(a)$ as roots are unique in a free monoid. For Case (2), suppose $g_j(x) = h_j(x)$. Then g_j and h_j agree on the first letter of $x \in \Sigma_j^*$ because the image of each letter has length one, and inductively we see that they agree on every letter of x. Hence, a subset $\Sigma_{g,h}$ of Σ_j forms a basis for $\mathrm{Eq}(I_j)$.

For Case (3), suppose there is a sequence of reductions beginning and ending at I_j:

$$I_j \to I_{j+1} \to \cdots \to I_{j+(i-1)} \to I_{j+i} = I_j$$

and write $r := j + i$. By Lemma 4, $\mathrm{Eq}(I_j) = g_{j+1}g_{j+2}\ldots g_r(\mathrm{Eq}(I_r)) = \mathrm{Eq}(I_r)$; thus $\overline{g_r} := g_{j+1}g_{j+2}\ldots g_r$ restricts to an automorphism of $\mathrm{Eq}(I_j)$, so $\overline{g_r}|_{\mathrm{Eq}(I_j)} \in \mathrm{Aut}(\mathrm{Eq}(I_j))$. The automorphism $\overline{g_r}$ is necessarily length-preserving ($|\overline{g_r}(w)| = |w|$ for all $w \in \mathrm{Eq}(I_j)$). Consider $x \in \mathrm{Eq}(I_j) = \mathrm{Eq}(I_r)$. Then $\overline{g_r}$ maps the letters occurring in x_r to letters and so $g_j (= g_r)$ and $h_j (= h_r)$ map the letters occuring in x to letters, and it follows that every letter occuring in x is a solution to $I_r = I_j$. Hence, a subset $\Sigma_{g,h}$ of Σ_j forms a basis for $\mathrm{Eq}(I_j)$ as required.

Therefore, in all three cases a subset $\Sigma_{g,h}$ of Σ_j forms a basis for $\mathrm{Eq}(I_j)$, and since Σ_j is computable, this basis is as well. In order to prove the theorem, it is sufficient to prove that there is a computable immersion $\psi_{g,h} : \Sigma_{g,h}^* \to \Sigma^*$. Consider the map $\tilde{g} = g_1g_2\cdots g_j : \Sigma_j^* \to \Sigma^*$ (and the analogous $\tilde{h}$). Now, each g_i is marked, by Lemma 3, and so $\tilde{g}$ is the composition of marked morphisms and hence is marked itself. Define $\psi_{g,h} := \tilde{g}|_{\Sigma_{g,h}^*}$. This map is computable from $\tilde{g}$, and as $\Sigma_{g,h} \subseteq \Sigma_j$, the map $\psi_{g,h}$ is marked. As $\mathrm{Image}(\psi_{g,h}) = g_1g_2\ldots g_j(\mathrm{Eq}(I_j)) = \mathrm{Eq}(I)$, by Lemma 4 and the above, the result follows. ◀

Theorem 5 together with Lemma 6 give the non-algorithmic part of Theorem A. A subsemigroup M of a free monoid Σ^* is *marked* if it is the image of a marked morphism.

▶ **Lemma 6.** *If $\{M_j\}_{j\in J}$ is a set of marked subsemigroups of Σ^* then the intersection $\bigcap_{j\in J} M_j$ is marked.*

Proof. Firstly, suppose $x, y \in M_j$ for some $j \in J$. Then there exist two words $x_0 \ldots x_l$ and $y_0 \ldots y_k$, with $x_i, y_i \in \Sigma$, such that $\phi(x_0 \ldots x_l) = x$ and $\phi(y_0 \ldots y_k) = y$, where ϕ is a marked morphism. If x and y have a nontrivial common prefix, then because ϕ is marked we get $x_0 = y_0$, and $\phi(x_0)$ is a prefix of both x and y, and in particular $\phi(x_0) \in M_j$. By continuing this argument, if z is a maximal common prefix of x and y, then $z \in M_j$.

Now, suppose $x, y \in \bigcap_{j\in J} M_j$, and suppose they both begin with some letter $a \in \Sigma \cup \Sigma^{-1}$. By the above, their maximal common prefix z_a is contained in each M_j and so is contained in $\bigcap_{j\in J} M_j$. Therefore, z_a is a prefix of every element of $\bigcap_{j\in J} M_j$ beginning with an a. It follows that $\bigcap_{j\in J} M_j$ is immersed, as required. ◀

We now prove the algorithmic part of Theorem A (this is independent of Lemma 6).

▶ **Lemma 7.** *There exists an algorithm with input a finite set of marked morphisms S from Σ^* to Δ^* and output a marked morphism $\psi_S : \Sigma_S^* \to \Sigma^*$ such that* $\mathrm{Image}(\psi_S) = \mathrm{Eq}(S)$.

Proof. We use induction on $|S|$. By Theorem 5, the result holds if $|S| = 2$. Suppose the result holds for all sets of n marked morphisms, $n \geqslant 2$, and let S be a set of $n+1$ marked morphisms. Take elements $g, h \in S$, and write $S_g = S \setminus \{g\}$. By hypothesis, we can algorithmically obtain marked morphisms $\psi_{S_g} : \Sigma_{S_g}^* \to \Sigma^*$ and $\psi_{g,h} : \Sigma_{g,h}^* \to \Sigma^*$ such that $\mathrm{Image}(\psi_{S_g}) = \mathrm{Eq}(S_g)$ and $\mathrm{Image}(\psi_{g,h}) = \mathrm{Eq}(g, h)$.

By Lemma 3, there exists a (computable) marked morphism $\psi_S : \Sigma_S^* \to \Sigma^*$ such that $\mathrm{Image}(\psi_S) = \mathrm{Image}(\psi_{S_g}) \cap \mathrm{Image}(\psi_{g,h})$ (the map ψ_S corresponds to the map k in Lemma 3, and Σ_S to Σ'). Then, as required: $\mathrm{Image}(\psi_S) = \mathrm{Image}(\psi_{S_g}) \cap \mathrm{Image}(\psi_{g,h}) = \mathrm{Eq}(S_g) \cap \mathrm{Eq}(g, h) = \mathrm{Eq}(S)$. ◀

We now prove Theorem A, which states that the equaliser is the image of a marked map.

Proof of Theorem A. By applying Lemma 6 to Theorem 5, there exists an alphabet Σ_S and a marked morphism $\psi_S : \Sigma_S^* \to \Sigma^*$ such that $\mathrm{Image}(\psi_S) = \mathrm{Eq}(S)$, while by Lemma 7 if S is finite then such a marked morphism can be algorithmically found. ◀

We now prove Corollary F, which says that $\mathrm{Eq}(S)$ is free of rank $\leqslant |\Sigma|$.

Proof of Corollary F. Consider the marked morphism $\psi_S : \Sigma_S^* \to \Sigma^*$ given by Theorem A. By Lemma 2, ψ_S is injective so $\mathrm{Image}(\psi_S)$ is free. As ψ_S is marked the map $\Sigma_S \to \Sigma$ taking each $a \in \Sigma_S$ to the initial letter of $\psi_S(a)$ is an injection, so $|\Sigma_S| \leqslant |\Sigma|$ as required. ◀

We now prove a strong form of the $\mathrm{AEP}_{\mathrm{FM}}$ for marked morphisms.

▶ **Corollary 8.** *There exists an algorithm with input a finite set S of marked morphisms from Σ^* to Δ^* and output a basis for* $\mathrm{Eq}(S)$.

Proof. To algorithmically obtain a basis for $\mathrm{Eq}\,(S)$, first use the algorithm of Theorem A to obtain the marked morphism $\overline{\psi}_S : \Sigma_S^* \to \Sigma^*$ such that $\mathrm{Image}(\psi_S) = \mathrm{Eq}(S)$. Then, recalling that we store ψ_S as a list $(\psi_S(a))_{a\in\Sigma}$, the required basis is the set of elements in this list, so the set $\{\psi_S(a)\}_{a\in\Sigma}$. ◀

Corollary H, the $\mathrm{AEP}_{\mathrm{FM}}$ for marked morphisms, follows from Corollary 8 by taking $|S| = 2$, while Corollary B, the simultaneous $\mathrm{PCP}_{\mathrm{FM}}$, also follows as $\mathrm{Eq}\,(S)$ is trivial if and only if its basis is empty.

3 Immersions of free groups

We denote the free group with finite generating set Σ by $F(\Sigma)$, and view it as the set of all *freely reduced words* over $\Sigma^{\pm 1} = \Sigma \cup \Sigma^{-1}$, that is, words not containing xx^{-1} as subwords, $x \in \Sigma^{\pm 1}$, together with the operations of concatenation and free reduction (that is, the removal of any xx^{-1} that might occur when concatenating two words).

We now begin our study of immersions of free groups, as defined in the introduction. We first state the characterising lemma, then explain the terms involved before giving the proof.

▶ **Lemma 9.** *Let $g : F(\Sigma) \to F(\Delta)$ be a free group morphism. The following are equivalent.*

1. *The map g is an immersion of free groups.*
2. *Every word in the language $L(\Gamma_g, v_g)$ is freely reduced.*
3. *For all $x, y \in \Sigma \cup \Sigma^{-1}$ such that $xy \neq 1$, the length identity $|g(xy)| = |g(x)| + |g(y)|$ holds.*

Characterisation (3) is the established definition of Kapovich [15]. Characterisation (2) is the one we shall work with in this article. It uses "Stallings graphs", which are essentially finite state automata that recognise the elements of finitely generated subgroups of free groups. We define these now, and refer the reader to [16] for background on Stallings graphs.

The *(unfolded) Stallings graph* Γ_g of the free group morphism g is the directed graph formed by taking a bouquet with $|\Sigma|$ petals attached at a central vertex we call v_g, where each petal consists of a path labeled by $g(x) \in (\Delta \cup \Delta^{-1})^*$; the elements of Δ^{-1} occur as edges traversed backwards and we denote by e^{-1} the edge e in opposite direction, and by $E\Gamma_g^{\pm 1}$ the sets of edges in both directions. A path $q = (e_1, \dots, e_n)$, $e_i \in E\Gamma_g^{\pm 1}$ edges, is *reduced* if it has no backtracking, that is, $e_i^{-1} \neq e_{i+1}$ for all $1 \leqslant i < n$. We denote by $\iota(p)$ the initial vertex of a path p and $\tau(p)$ for the terminal vertex, and call a reduced path p with $\iota(p) = u = \tau(p)$ a *closed reduced circuit*.

We shall view Γ_g as a finite state automaton (Γ_g, v_g) with start and accept states both equal to v_g. Then the *extended language accepted by* (Γ_g, v_g) is the set of words labelling reduced closed circuits at v_g in Γ_g:

$$L(\Gamma_g, v_g) = \{label(p) \mid p \text{ is a reduced path with } \iota(p) = u = \tau(p)\}.$$

Immersions are precisely those maps g such that every element of $L(\Gamma_g, v_g)$ is freely reduced; this corresponds to the automaton (Γ_g, v_g) and the "reversed" automaton $(\Gamma_g, v_g)^{-1}$, where edge directions are reversed, both being deterministic (map g in Figure 1 is not an immersion; although the automaton (Γ_g, v_g) is deterministic, $(\Gamma_g, v_g)^{-1}$ is not). For such maps, $L(\Gamma_g, v_g)$ is precisely the image of the map g [16, Proposition 3.8].

Proof of Lemma 9. (1) ⇔ (2). Every element of $L(\Gamma_g, v_g)$ is freely reduced if and only if the graph Γ_g, with base vertex v_g, is such that for all $e_1, e_2 \in (E\Gamma_g)^{\pm 1}$ such that both edes start at v_g or both edges end at v_g, then e_1 and e_2 have different labels (so $\gamma_g(e_1) \neq \gamma_g(e_2)$). This condition on labels is equivalent to $g(\Sigma \cup \Sigma^{-1})$ being marked, as required.

(1) ⇔ (3). Condition (3) is equivalent to the condition that for all $x, y \in \Sigma \cup \Sigma^{-1}$ such that $xy \neq 1$, free cancellation does not happen between $g(x)$ and $g(y)$, which in turn is equivalent to the condition that for all such x, y the elements $g(x^{-1})$ and $g(y)$ start with different letters of $\Delta \cup \Delta^{-1}$. This is equivalent to $g(\Sigma \cup \Sigma^{-1})$ being marked, as required. ◀

▶ **Example 10.** Let $g : F(a, b) \to F(x, y)$ be the map defined by $g(a) = x^{-2}y$ and $g(b) = y^2x$. Then the graph Γ_g, where the double arrow represents x and the single arrow y, is depicted in Figure 1. The map g is not an immersion since there are two edges labeled x entering v_g (violating Characterisation (2)). Similarly, $g(a)$ and $g(b^{-1})$ both start with x^{-1} (violating Characterisation (1)) and $|g(ba)| = 4 < 6 = |g(a)| + |g(b)|$ (violating Characterisation (3)).

Using Characterisation (2), we see that immersions are injective [16, Proposition 3.8]:

▶ **Lemma 11.** *If $g : F(\Sigma) \to F(\Delta)$ is an immersion then it is injective.*

4 The reduction of an instance in free groups

By an *immersed* instance of the $\mathrm{PCP}_{\mathrm{FG}}$ we mean an instance $I = (\Sigma, \Delta, g, h)$ where both g and h are immersions. In this section we define the "reduction" of an immersed instance of the $\mathrm{PCP}_{\mathrm{FG}}$, which is similar to the reduction in the free monoid case.

Let Γ be a directed, labeled graph and $u \in V\Gamma$ a vertex of Γ. The *core graph of* Γ *at* u, written $\mathrm{Core}_u(\Gamma)$, is the maximal subgraph of Γ containing u but no vertices of degree 1, except possibly u itself. Note that $L(\mathrm{Core}_u(\Gamma), u) = L(\Gamma, u)$. For Γ_1, Γ_2 directed, labeled

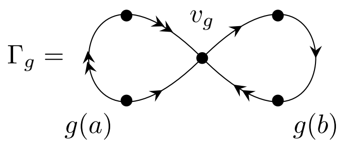

Figure 1 The graph Γ_g for the map $g : F(a, b) \to F(x, y)$ defined by $g(a) = x^{-2}y$, $g(b) = y^2x^{-1}$.

graphs, the *product graph of* Γ_1 *and* Γ_2, denoted $\Gamma_1 \otimes \Gamma_2$, is the subgraph of $\Gamma_1 \times \Gamma_2$ with vertex set $V\Gamma_1 \times V\Gamma_2$ and edge set $\{(e_1, e_2) \mid e_i \in E\Gamma_i^{\pm 1}, label(e_1) = label(e_2)\}$. One may think of the standard construction of an automaton recognising the intersection of two regular languages, each given by a finite state automaton Γ_i with start state s_i, where the core of $\Gamma_1 \otimes \Gamma_2$ at (s_1, s_2) is the automaton recognising this intersection.

Core graph of a pair of morphisms. Let $g : F(\Sigma_1) \to F(\Delta)$, $h : F(\Sigma_2) \to F(\Delta)$ be morphisms. The *core graph of the pair* (g, h), denoted $\mathrm{Core}(g, h)$, is the core graph of $\Gamma_g \otimes \Gamma_h$ at the vertex $v_{g,h} = (v_g, v_h)$, so $\mathrm{Core}(g, h) = \mathrm{Core}_{v_{g,h}}(\Gamma_g \otimes \Gamma_h)$. We shall refer to $v_{g,h}$ as the *central vertex of* $\mathrm{Core}(g, h)$. Note that $\mathrm{Core}(g, h)$ represents the intersection of the two images [16, Lemma 9.3], in the sense that

$$L(\mathrm{Core}(g, h), v_{g,h}) = \mathrm{Image}(g) \cap \mathrm{Image}(h).$$

Write $\delta_g : \mathrm{Core}(g, h) \to \Gamma_g$ and $\delta_h : \mathrm{Core}(g, h) \to \Gamma_h$ for the restriction of $\mathrm{Core}(g, h)$ to the g and h components, respectively, so $\delta_g(e_1, e_2) = e_1$, etc.

Now, let g, h be immersions. The graph $\mathrm{Core}(g, h)$ is a bouquet and every element of $L(\mathrm{Core}(g, h), v_{g,h})$ is freely reduced [16, Lemma 9.2]. We therefore have free group morphisms $g' : L(\mathrm{Core}(g, h), v_{g,h}) \to L(\Gamma_g, v_g)$ and $h' : L(\mathrm{Core}(g, h), v_{g,h}) \to L(\Gamma_h, v_h)$ induced by the maps δ_g, δ_h, where $L(\Gamma_g, v_g) = F(\Sigma_1)$ and $L(\Gamma_h, v_h) = F(\Sigma_2)$. These maps are computable [16, Corollary 9.5]. Let Σ' be the alphabet whose elements consist of the petals of $\mathrm{Core}(g, h)$. Then Σ' generates the free group $L(\mathrm{Core}(g, h), v_{g,h})$, so $F(\Sigma') = L(\mathrm{Core}(g, h), v_{g,h})$, and we see that both g' and h' are immersions with $g' : F(\Sigma') \to F(\Sigma_1)$, $h' : F(\Sigma') \to F(\Sigma_2)$. The map $gg' = hh'$, which we shall call k (so $k : F(\Sigma') \to F(\Delta)$) is the composition of immersions and hence is itself an immersion. We therefore have the following.

▶ **Lemma 12.** *If $g : F(\Sigma_1) \to F(\Delta)$ and $h : F(\Sigma_2) \to F(\Delta)$ are immersions then the corresponding maps $g' : F(\Sigma') \to F(\Sigma_1)$, $h' : F(\Sigma') \to F(\Sigma_2)$ and $k : F(\Sigma') \to F(\Delta)$, where $k = gg' = hh'$, are immersions and are computable.*

Reduction. The *reduction* of an immersed instance $I = (\Sigma, \Delta, g, h)$ of the $\mathrm{PCP}_{\mathrm{FG}}$ is the instance $I' = (\Sigma', \Delta, g', h')$ where g' and h' are as above, but with codomain Δ (which we may do as $\Sigma \subseteq \Delta$). We additionally assume that $\Sigma' \subseteq \Sigma$; we can do this as $|\Sigma'| \leqslant |\Sigma|$ by Lemma 12. As I is immersed, I' is also immersed by Lemma 12. In the next section we show that the name “reduction” makes sense, as it reduces the “prefix complexity” of instances.

▶ **Example 13.** Consider the maps $g, h : F(a, b, c) \to F(x, y, z)$ given by $g(x) = aba^2, g(b) = y^{-1}, g(c) = zxz$ and $h(a) = x, h(b) = yx^2y, h(c) = z$.

Then the graph $\mathrm{Core}(g,h)$ is a bouquet with two petals labelled xyx^2y and zxz, and $\mathrm{Image}(g) \cap \mathrm{Image}(h) = \langle xyx^2y, zxz\rangle$. Moreover, $g(ab^{-1}) = h(ab) = xyx^2y$ and $g(z) = h(zxz) = zxz$. Then we can take $\Sigma' = \{a', b'\}$, and the maps given by $g'(a') = ab^{-1}, g(b') = c$ and $h'(a') = ab, h'(b') = cac$ are the reduction of (g,h).

We now prove that reduction preserves equalisers. Two instances I_1 and I_2 of the $\mathrm{PCP_{FG}}$ are *strongly equivalent* if the equalisers are isomorphic, which we write as $\mathrm{Eq}(I_1) \cong \mathrm{Eq}(I_2)$.

▶ **Lemma 14.** *Let $I' = (\Sigma', \Delta', g', h')$ be the reduction of $I = (\Sigma, \Delta, g, h)$ where g and h are immersions. Then I and I' are strongly equivalent, and $g'(\mathrm{Eq}(I')) = \mathrm{Eq}(I) = h'(\mathrm{Eq}(I'))$.*

Proof. It is sufficient to prove that $g'|_{\mathrm{Eq}(I')}$ is injective and $g'(\mathrm{Eq}(I')) = \mathrm{Eq}(I)$; that $h'(\mathrm{Eq}(I')) = \mathrm{Eq}(I)$ follows as $g'|_{\mathrm{Eq}(I')} = h'|_{\mathrm{Eq}(I')}$.

As g' is an immersion it is injective, by Lemma 11. Therefore, $g'|_{\mathrm{Eq}(I')}$ is injective. To see that $\mathrm{Image}(g'|_{\mathrm{Eq}(I')}) \leqslant \mathrm{Eq}(I)$, suppose $x' \in \mathrm{Eq}(I')$. Writing $x = g'(x') = h'(x')$, we have $g(x) = gg'(x') = hh'(x') = h(x)$ and so $x = g'(x') \in \mathrm{Eq}(I)$, as required.

To see that $\mathrm{Image}(g'|_{\mathrm{Eq}(I')}) \geqslant \mathrm{Eq}(I)$, suppose $x \in \mathrm{Eq}(I)$. Then there exists a path $p_x \in \mathrm{Core}(g,h)$, $\iota(p_x) = v_{g,h} = \tau(p_x)$, such that $\gamma_g\delta_g(p_x) = g(x) = h(x) = \gamma_h\delta_h(p_x)$ [16, Proposition 9.4], where $\gamma_g : \Gamma_g \to \Gamma_\Delta$ is the canonical morphism of directed, labeled graphs from Γ_g to the bouquet Γ_Δ with Δ petals. Hence, writing x' for the element of $F(\Sigma')$ corresponding to $p_x \in L(\mathrm{Core}(g,h), v_{g,h})$, we have that $gg'(x') = g(x) = h(x) = hh'(x)$. As h and g are injective, by Lemma 11, we have that $g'(x') = x = h'(x')$ as required. ◀

5 Prefix complexity of immersions in free groups

In this section we associate to an instance I of the $\mathrm{PCP_{FG}}$ a certain complexity, called the "prefix complexity". We prove that the process of reduction does not increase this complexity, and that for all $n \in \mathbb{N}$ there are only finitely many instances with complexity $\leqslant n$.

Let $I = (\Sigma, \Delta, g, h)$ be an immersive instance of the $\mathrm{PCP_{FG}}$. We define, analogously to [11, Section 4] (see also [9]), the *prefix complexity* $\sigma(I)$ as:

$$\begin{aligned}\sigma(I) = {} & |\cup_{a\in\Sigma^{\pm 1}}\{x \in F(\Delta) \mid x \text{ is a proper prefix of } g(a)\}| \\ & + |\cup_{a\in\Sigma^{\pm 1}}\{x \in F(\Delta) \mid x \text{ is a proper prefix of } h(a)\}|\,.\end{aligned}$$

In the maps in Example 13, $\sigma(I) = 10 + 6 = 16$, and $\sigma(I') = 2 + 4 = 6$.

The process of reduction does not increase the prefix complexity, and we prove this by using the fact that, for any $a \in \Sigma^{\pm 1}$, the proper prefixes of $g(a)$ and $h(a)$ are in bijection with the proper initial subpaths of the petals of Γ_g and Γ_h, respectively.

▶ **Lemma 15.** *Let $I = (\Sigma, \Delta, g, h)$ be an instance of the $\mathrm{PCP_{FG}}$ with g and h immersions, and let I' be the reduction of I. Then $\sigma(I') \leqslant \sigma(I)$.*

Proof. We write $V_g\,\mathrm{Core}(g,h) = \{(v_g, v) \in V\,\mathrm{Core}(g,h) \mid v \in \Gamma_h\} = \delta_g^{-1}(v_g)$ for the set of vertices in the $\mathrm{Core}(g,h)$ whose first component is the central vertex v_g of Γ_g, and similarly for $V_h\,\mathrm{Core}(g,h)$. Note that $V_g\,\mathrm{Core}(g,h) \cap V_h\,\mathrm{Core}(g,h) = \{v_{g,h}\}$.

By construction, each petal of Γ_g and Γ_h corresponds to a letter $a \in \Sigma^{\pm 1}$, and we shall denote the petal also by a. Write $P\Gamma$ for the set of reduced paths in a graph Γ. Similarly to $a \in P\Gamma_g$ and $a \in P\Gamma_h$, we map write $a \in P\,\mathrm{Core}(g,h)$ for the petal in $\mathrm{Core}(g,h)$ corresponding to $a \in (\Sigma')^{\pm 1}$. From now on, all paths are assumed to be reduced. Define

$$G = \cup_{a\in\Sigma^{\pm 1}}\{p \in \Gamma_g \mid p \text{ is a proper initial subpath of petal } a \in \Gamma_g\},$$
$$G' = \cup_{a\in(\Sigma')^{\pm 1}}\{p \in \operatorname{Core}(g,h) \mid p \text{ is a proper initial subpath}$$
$$\text{of } a \in \operatorname{Core}(g,h) \text{ s.t. its end vertex } \tau(p) \in V_g \operatorname{Core}(g,h)\},$$

and define H and H' analogously. Hence, $\sigma(I) = |G|+|H|$ and analogously $\sigma(I') = |G'|+|H'|$.

For $a \in (\Sigma')^{\pm 1}$ let $q \in G'$ be a subpath of $a \in P\operatorname{Core}(g,h)$. Denote by r_q the shortest subpath of a intersecting q at only one point, their common end vertex (that is, $\tau(q) = \tau(r_q) = q \cap r_q$), such that $\iota(r_q) \in V_h \operatorname{Core}(g,h)$; the paths q and r_q can be seen as "facing" one another on a. As $V_g \operatorname{Core}(g,h) \cap V_h \operatorname{Core}(g,h) = \{v_{g,h}\}$, and as q is a proper initial subpath of a, the projection $\delta_h(r_q)$ is a non-trivial path in Γ_h. Note also that $\iota(\delta_h(r_q)) = v_h$, as $\iota(r_q) \in V_h \operatorname{Core}(g,h)$, hence there exists some $b \in \Sigma^{\pm 1}$ such that $\delta_h(r_q)$ is a proper initial subpath of the petal $b \in \Gamma_h$. Therefore, $\delta_h(r_q) \in H$. Let $\xi_H : G' \to H$ be the map given by $\xi_H(q) = \delta_h(r_q)$.

We now prove that ξ_H is injective. Suppose $p, q \in G'$ are such that $\xi_H(p) = \xi_H(q)$, and let r_p and r_q be the paths obtained from p and q, respectively, such that $\xi_H(p) = \delta_h(r_p)$ and $\xi_H(q) = \delta_h(r_q)$. Write e_p for the terminal edge of r_p, and e_q for the terminal edge of r_q, and note that these two edges have the same label and direction as $\delta_h(e_p) = \delta_h(e_q)$. Now, $\delta_g(\tau(e_p)) = v_g = \delta_g(\tau(e_q))$ as $\tau(r_p), \tau(r_q) \in V_g \operatorname{Core}(g,h)$, and as e_p and e_q have the same label and direction we have that $\delta_g(e_p) = \delta_g(e_q)$. Therefore, both δ_g and δ_h agree on e_p and e_q, and so as $\operatorname{Core}(g,h)$ is a subgraph of $\Gamma_g \times \Gamma_h$ we have that $e_p = e_q$. As $\operatorname{Core}(g,h)$ is a bouquet, there exists a unique shortest reduced path s such that $\iota(s) = v_{g,h}$ and $\tau(s) = s \cap e_p = \tau(e_p)$. Hence, $p = s = q$ as required.

Thus ξ_H is injective, and so $|G'| \leqslant |H|$. The same will hold for an analogously defined function ξ_H from H' to G, so $|H'| \leqslant |G|$. Therefore, $\sigma(I') = |G'|+|H'| \leqslant |G|+|H| = \sigma(I)$. ◀

For a fixed number $n \geqslant 1$ there are obviously only finitely many words which have $\leqslant n$ proper prefixes, and so the following is clear:

▶ **Lemma 16.** *There exist only finitely many distinct instances $I = (\Sigma, \Delta, g, h)$ of the* $\mathrm{PCP}_{\mathrm{FG}}$ *that satisfy $\sigma(I) \leqslant n$.*

As the reduction I' of an instance I gives $\sigma(I') \leqslant \sigma(I)$, and as $|\Sigma'| \subseteq |\Sigma|$, this means that the process of iteratively computing reductions will eventually cycle.

6 Solving the Algorithmic Equaliser Problem in free groups ($\mathrm{AEP}_{\mathrm{FG}}$)

The algorithm for solving the $\mathrm{AEP}_{\mathrm{FG}}$ for immersions is analogous to the algorithm for marked free monoid morphisms in Section 2. Our algorithm starts by making reductions $I_0, I_1, I_2, \ldots$, beginning with $I_0 = I$, the input instance. By Lemma 16, we will obtain an instance $I_j = (\Sigma_j, \Delta, g_j, h_j)$ such that one of the following will occur:

1. $|\Sigma_j| = 1$.
2. $\sigma(I_j) = 0$.
3. there exists some $i < j$ with $I_i = I_j$ (sequence starts cycling).

Keeping in mind the fact that reductions preserve equalisers (Lemma 14), we obtain in each case a subset $\Sigma_{g,h}$ (possibly empty) which forms a basis for $\operatorname{Eq}(I_j)$: For Case (1), writing $\Sigma_j = \{a\}$, the result holds as if $g(a^i) = h(a^i)$ then $g(a)^i = h(a)^i$ and so $g(a) = h(a)$ as roots are unique in a free group. For Case (2), $\sigma(I_j) = 0$ is equivalent to $|g(a)| = |h(a)| = 1$ for all $a \in \Sigma$. Suppose there exists some non-trivial reduced word $x = a_{i_1}^{\epsilon_1} \cdots a_{i_n}^{\epsilon_n}$ such that $g(x) = h(x)$.

Then as g and h are injective, the words $g(a_{i_1})^{\epsilon_1} \cdots g(a_{i_n})^{\epsilon_n}$ and $h(a_{i_1})^{\epsilon_1} \cdots h(a_{i_n})^{\epsilon_n}$ are freely reduced and hence are the same word, and so $g(a_{i_j}) = h(a_{i_j})$. The result then follows for Case (2). Case (3) has a more involved proof.

▶ **Lemma 17.** *Let $I = (\Sigma, \Delta, g, h)$ be an immersive instance of the* $\mathrm{PCP_{FG}}$ *that starts a cycle (i.e. starting the reduction process with I eventually gives I again). If* $\mathrm{Eq}(I)$ *is non-trivial then a subset of Σ forms a basis for* $\mathrm{Eq}(I)$*.*

Proof. There is a sequence of reductions beginning and ending at I:

$$I = I_0 \to I_1 \to \cdots \to I_{r-1} \to I_r = I$$

where $I_i = (\Sigma_i, \Delta, g_i, h_i)$. By Lemma 14, $\mathrm{Eq}(I_0) = g_1 g_2 \ldots g_r(\mathrm{Eq}(I_r)) = \mathrm{Eq}(I_r)$ and so $\overline{g_r} = g_1 g_2 \ldots g_r$ restricts to an automorphism of $\mathrm{Eq}(I_0)$, that is, $\overline{g_r}|_{\mathrm{Eq}(I_0)} \in \mathrm{Aut}(\mathrm{Eq}(I_0))$. For $\overline{h_r}$ defined analogously, $\overline{h_r}|_{\mathrm{Eq}(I_0)} \in \mathrm{Aut}(\mathrm{Eq}(I_0))$. Write $\mathrm{Eq}(I_k)^{(n)}$ for the set of words in $\mathrm{Eq}(I_k)$ of length precisely n, and $\mathrm{Eq}(I_k)^{(\leqslant n)}$ for the set of words in $\mathrm{Eq}(I_k)$ of length at most n. Consider some $x_0 \in \mathrm{Eq}(I_0)$ and write $x_r = \overline{g_r}^{-1}(x_0)$. Then

$$x_0 = g_1 g_2 \ldots g_r(x_r) = \overline{g_r}(x_r),$$
$$x_0 = h_1 h_2 \ldots h_r(x_r) = \overline{h_r}(x_r).$$

By Lemma 12, both g_i and h_i are immersions for each i, and so by Characterisation (3) of Lemma 9 we see that $|g_i(w)| \geqslant |w|$ for all $w \in F(\Sigma_i)$. Hence, $|x_0| \geqslant |x_r|$. Therefore, for all $m \geqslant 1$ the map $\overline{g_r}$ induces a map $\overline{g_r}^{(m)} : \mathrm{Eq}(I_r)^{(m)} \to \mathrm{Eq}(I_r)^{(\leqslant m)}$. Clearly $\overline{g_r}^{(1)}$ is a bijection, and so we inductively see that $\overline{g_r}^{(m)}$ has image $\mathrm{Eq}(I_r)^{(m)}$. Therefore, the automorphism $\overline{g_r}$ of $\mathrm{Eq}(I_0)$ is length-preserving ($|\overline{g_r}(w)| = |w|$ for all $w \in \mathrm{Eq}(I)$), and so maps the letters occurring in x_r to letters. Hence, $g_0 (= g_r)$ and $h_0 (= h_r)$ map the letters occuring in x_r to letters, and it follows that every letter occuring in x_r is a solution to I_0. Hence, a subset $\Sigma_{g,h}$ of Σ_r forms a basis for $\mathrm{Eq}(I_r)$. ◀

We now prove the central theorem of this article, which gives an algorithm to describe $\mathrm{Eq}(I)$ as the image of an immersion. Note that not every subgroup of a free group is the image of an immersion: for example, if $|\Sigma| = n$, then no subgroup of $F(\Sigma)$ of rank $> n$ is the image of an immersion. We store a morphism $f : F(\Sigma) \to F(\Delta)$ as a list $(f(a))_{a \in \Sigma}$.

▶ **Theorem 18.** *There exist an algorithm with input an immersive instance $I = (\Sigma, \Delta, g, h)$ of the* $\mathrm{PCP_{FG}}$ *and output an immersion $\psi_{g,h} : F(\Sigma_{g,h}) \to F(\Sigma)$ such that* $\mathrm{Image}(\psi_{g,h}) = \mathrm{Eq}(I)$*.*

Proof. Start by making reductions $I = I_0 \to I_1 \to \cdots$. By Lemma 16 we will obtain an instance $I_j = (\Sigma_j, \Delta, g_j, h_j)$ satisfying one of the Cases (1)–(3) above, and in each case a subset $\Sigma_{g,h}$ of Σ_j forms a basis for $\mathrm{Eq}(I_j)$. Since Σ_j is computable, this basis is as well.

In order to prove the theorem, it is sufficient to prove that there is a computable immersion $\psi_{g,h} : F(\Sigma_{g,h}) \to F(\Sigma)$. Consider the map $\tilde{g} = g_1 g_2 \cdots g_j : F(\Sigma_j) \to F(\Sigma)$ (and the analogous $\tilde{h}$). Now, each g_i is an immersion, so $\tilde{g}$ is the composition of immersions and hence is an immersion. Define $\psi_{g,h} = \tilde{g}|_{F(\Sigma_{g,h})}$. This map is computable from $\tilde{g}$, and as $\Sigma_{g,h} \subseteq \Sigma_j$, the map $\psi_{g,h}$ is an immersion. As $\mathrm{Image}(\psi_{g,h}) = g_1 g_2 \ldots g_j(\mathrm{Eq}(I_j)) = \mathrm{Eq}(I)$, by Lemma 14 and the above, the result follows. ◀

We now prove Corollary G, which solves the $\mathrm{AEP_{FG}}$ for immersions of free groups.

Proof of Corollary G. To algorithmically obtain a basis for $\mathrm{Eq}(I)$, first obtain the immersion $\psi_{g,h} : F(\Sigma_{g,h}) \to F(\Sigma)$ given by Theorem 18. Then, recalling that we store $\psi_{g,h}$ as a list $(\psi_{g,h}(a))_{a \in \Sigma}$, the required basis is the set of elements in this list, so the set $\{\psi_{g,h}(a)\}_{a \in \Sigma}$. ◀

7 Sets of immersions

We now prove Theorem C and its corollaries. We first give a general result, from which the non-algorithmic part of Theorem C follows quickly. An *immersed subgroup* H of a free group $F(\Sigma)$ is a subgroup which is the image of an immersion. The proof of Lemma 19 is fundamentally identical to the proof of Lemma 6, via Characterisation 1 of Lemma 9.

▶ **Lemma 19.** *If $\{H_j\}_{j\in J}$ is a set of immersed subgroups of $F(\Sigma)$ then the intersection $\bigcap_{j\in J} H_j$ is immersed.*

Proof. Firstly, suppose $x, y \in H_j$ for some $j \in J$, and let z be their maximal common prefix. Then z decomposes uniquely as $z_1z_2\cdots z_nz'_{n+1}$ such that each $z_k \in H_j$. As H_j is immersed, and as z is a maximal common prefix of x and y, we have that $z \in H_j$.

Now, suppose $x, y \in \bigcap_{j\in J} H_j$, and suppose they both begin with some letter $a \in \Sigma \cup \Sigma^{-1}$. By the above, their maximal common prefix z_a is contained in each H_j and so is contained in $\bigcap_{j\in J} H_j$. Therefore, z_a is a prefix of every element of $\bigcap_{j\in J} H_j$ beginning with an a. It follows that $\bigcap_{j\in J} H_j$ is immersed, as required. ◀

The following lemma corresponds to the algorithmic part of Theorem C. Similar to the above, the proof of the lemma is fundamentally identical to the proof of Lemma 7.

▶ **Lemma 20.** *There exists an algorithm with input a finite set of immersions S from $F(\Sigma)$ to $F(\Delta)$ and output an immersion $\psi_S : F(\Sigma_S) \to F(\Sigma)$ such that* $\mathrm{Image}(\psi_S) = \mathrm{Eq}(S)$.

Proof. We proceed by inducting on $|S|$. By Theorem 18, the result holds if $|S| = 2$. Suppose the result holds for all sets of n immersions, $n \geqslant 2$, and let S be a set of $n+1$ immersions. Take elements $g, h \in S$, and write $S_g = S \setminus \{g\}$. By hypothesis, we can algorithmically obtain immersions $\psi_{S_g} : F(\Sigma_{S_g}) \to F(\Sigma)$ and $\psi_{g,h} : F(\Sigma_{g,h}) \to F(\Sigma)$ such that $\mathrm{Image}(\psi_{S_g}) = \mathrm{Eq}(S_g)$ and $\mathrm{Image}(\psi_{g,h}) = \mathrm{Eq}(g, h)$.

By Lemma 12, there exists a (computable) immersion $\psi_S : F(\Sigma_S) \to F(\Sigma)$ such that $\mathrm{Image}(\psi_S) = \mathrm{Image}(\psi_{S_g}) \cap \mathrm{Image}(\psi_{g,h})$ (the map ψ_S corresponds to the map k in the lemma, and Σ_S to Σ'). Then we have the required equality:

$$\begin{aligned}\mathrm{Image}(\psi_S) &= \mathrm{Image}(\psi_{S_g}) \cap \mathrm{Image}(\psi_{g,h})\\ &= \mathrm{Eq}(S_g) \cap \mathrm{Eq}(g, h)\\ &= \mathrm{Eq}(S).\end{aligned}$$ ◀

We now prove Theorem C, which states that the equaliser is the image of a computable immersion.

Proof of Theorem C. By Lemma 19, there exists an alphabet Σ_S and an immersion $\psi_S : F(\Sigma_S) \to F(\Sigma)$ such that $\mathrm{Image}(\psi_S) = \mathrm{Eq}(S)$, while by Lemma 20 if S is finite then such an immersion can be algorithmically found. ◀

We now prove Corollary D, which solves the simultaneous $\mathrm{PCP}_{\mathrm{FG}}$ for immersions.

Proof of Corollary D. First find a basis for $\mathrm{Eq}(I)$: obtain the immersion $\psi_{g,h} : F(\Sigma_{g,h}) \to F(\Sigma)$ given by Theorem C. Then, recalling that we store $\psi_{g,h}$ as a list $(\psi_{g,h}(a))_{a\in\Sigma}$, the required basis is the set of elements in this list, so the set $\{\psi_{g,h}(a)\}_{a\in\Sigma}$. Then $\mathrm{Eq}(S)$ is trivial if and only if this basis is empty. ◀

Finally, we prove Corollary E, which says that $\mathrm{Eq}(S)$ is of rank $\leqslant |\Sigma|$.

Proof of Corollary E. Consider the immersion $\psi_{g,h} : F(\Sigma_{g,h}) \to F(\Sigma)$ given by Theorem C. As $\text{Image}(\psi_{g,h}) = \text{Eq}(S)$ we have that $\text{rk}(\text{Eq}(S)) \leqslant |\Sigma_{g,h}|$, while as $\psi_{g,h}$ is an immersion we have that $|\Sigma_{g,h}| \leqslant |\Sigma|$, and the result follows. ◀

8 Algorithm to compute the equaliser

Theorems A and C produce the equaliser of a set S of morphisms as the image of a computable map ψ_S. For $S = \{g, h\}$, the structure of the algorithm that gives ψ_S (as a list of elements representing the images of the generators) is given below. The values for M in step 3 correspond to the number of instances of complexity $\leqslant \sigma(I)$, as explained in Section 9.

Algorithm 1 Structure of the algorithm that gives ψ_S.

```
1. Input $I = (\Sigma, \Delta, g, h)$.
2. Set $c =; 0$, $i := 0$, $I_0 := I$
3. Set $M := (|\Delta| + 1)^{2|\Sigma|(\sigma(I)+1)}$ (monoids) or $M := (2|\Delta|)^{2|\Sigma|(\sigma(I)+1)}$ (groups)
4. $i := i + 1$
5. Reduce instance $I_{i-1}$ to $I_i$ (as in Sections 2 and 4); store $I_i$ in memory
6. If $I_i$ has source alphabet of size 1 or $\sigma = 0$ then:
   a. Compute a basis $B$ for $\text{Eq}(I_i)$
   b. Print composition($B$, $i$) (see below) and terminate.
7. If $I_i$ is simpler than $I_{i-1}$ (smaller source alphabet or $\sigma$) then set $c = 0$ and goto (4)
8. If $c > M$ then there exists a cycle which starts with $I_i$.
   a. Compute a basis $B$ for $\text{Eq}(I_i)$
   b. Print composition($B$, $i$) and terminate.
```

Procedure `composition(`B`, `i`)` computes the composition of a map, stored as a list B, with the maps obtained in the reduction process, indexed from i downwards.

Algorithm 2 `composition(`B`, `i`)`.

```
1. Set $B := g_i(B)$, where $g_i$ is loaded from memory
2. $i := i - 1$
3. If $i \geqslant 0$, goto (1); else, output $B$.
```

9 Complexity analysis

The size of an instance $I = (\Sigma, \Delta, S)$, S a set of morphisms, is $|\Sigma| + |\Delta| + \sum_{g \in S} \sum_{a \in \Sigma^{\pm 1}} |g(a)|$. The algorithm underlying Theorem A can be run with $O(2^n)$ space, where n is the size of the input instance I, which gives a time bound of $O(2^{2^n})$. The space grows exponentially, unlike in [11], because the algorithm computes instances that must each be stored (as the immersion $\psi_{g,h}$ is their composition; this corresponds to the function `composition(`B`, `i`)`, above). To obtain this space complexity, first suppose $|S| = 2$ (so consider the function `pairs(`g`, `h`)`, above). There are at most $(|\Delta| + 1)^{2|\Sigma|(\sigma(I)+1)}$ instances I_j with $\sigma(I_j) \leqslant \sigma(I)$ [11, Proof of Lemma 3], which is $O(2^n)$. Every other procedure requires asymptotically less space, and hence if $|S| = 2$ we require $O(2^n)$ space. For $S = \{g_1, \ldots, g_k\}$, note that we only need to compute the immersions corresponding to $\text{Eq}(g_i, g_{i+1})$ for $1 \leqslant i < k$ (as these intersect to give $\mathrm{Eq}(S)$), and these can all be stored in $(k-1)\times O(2^n)=O(2^n)$ space. Intersection corresponds to reduction, and reduction can be done in PSPACE [11, Section 6]. Hence, the algorithm can be run in $O(2^n)$ space.

Similarly, the algorithm underlying Theorem C runs in $O(2^n)$ space, where n is the input size. The main difference to the above is that there are $O(2^n)$ instances I_j with $\sigma(I_j)\leqslant\sigma(I)$. To see this, write $m:=\sigma(I)$ and $d:=|\Delta|$. If $I_j=(\Sigma_j,\Delta_j,g_j,h_j)$ is such that $\sigma(I_j)\leqslant m$ then $|g(a)|\leqslant m+1$ for all $a\in\Sigma_j^{\pm1}$, as $g(a)$ has at most m proper prefixes, and similarly $|h(a)|\leqslant m+1$. There are $2d(2d-1)^m$ freely reduced words of length $m+1$ in $F(\Sigma_j)$, and so (by using the empty word) we see that there are at most $(2d)^{m+1}$ freely reduced words of length *at most* $m+1$. As each list of $2|\Sigma_j|$ words defines an instance, there are at most $(2d)^{2|\Sigma_j|(m+1)}\leqslant(2d)^{2|\Sigma|(m+1)}$ instances that satisfy $\sigma(I)\leqslant m$. This is $O(2^n)$ as required.

10 The density of marked morphisms and immersions

Here we show that immersions and marked morphisms are not a negligible (i.e. density zero) subset of the entire set of free group and free monoid morphisms, respectively, but represent a strictly positive proportion of those.

Suppose $\Sigma=\{a_1,\ldots,a_k\}$, and $k=|\Sigma|\geqslant|\Delta|=m$. A morphism in a free monoid or free group, $\phi:\Sigma^*\to\Delta^*$ or $\phi:F(\Sigma)\to F(\Delta)$, is uniquely determined by $(\phi(a_1),\ldots,\phi(a_k))$.

We start with the monoid case. There are m^n words of length n in Δ^*, and $\sum_{1\leqslant i\leqslant n}m^i\sim cm^n$ words of length $\leqslant n$, where $c=\frac{m}{m-1}$ and we write $a_n\sim b_n$ for $\lim_{n\to\infty}\frac{a_n}{b_n}=1$. If α_n is the number of morphisms from Σ^* to Δ^* with images of length at most n, then $\alpha_n\sim(cm^n)^k$. Now let β_n be the number of marked morphisms from Σ^* to Δ^* with images of length at most n. For a marked morphism ϕ, each word in the list $(\phi(a_1),\ldots,\phi(a_k))$ must start with a different letter, followed by any word of length $\leqslant n-1$. Since there are $\binom{m}{k}k!$ options for the first letters, $\beta_n\sim\binom{m}{k}k!(cm^{n-1})^k$ and we get:

▶ **Proposition 21.** *If α_n and β_n are the numbers of morphisms and marked morphisms, respectively, from Σ^* to Δ^*, with images of length at most n, then the density of the marked morphisms among all morphisms is a positive constant:*

$$\lim_{n\to\infty}\frac{\beta_n}{\alpha_n}=\lim_{n\to\infty}\frac{\binom{m}{k}k!(cm^{n-1})^k}{(cm^n)^k}=\frac{m!}{m^k(m-k)!}.$$

In the free group case the counting is similar, but there are more restrictions on the images of an immersion: first, all images need to be reduced words, and second, not just their first letters are constrained, but also their last letters. For some ϕ, let the set of first letters of $(\phi(a_1),\ldots,\phi(a_k))$ be $F\subset\Delta^{\pm1}$, and the set of inverses of the last letters be $L\subset\Delta^{\pm1}$. Then $\phi:F(\Sigma)\mapsto F(\Delta)$ is an immersion if all letters in F are distinct, all the letters in L are distinct, which implies $|F|=|L|=k$, and furthermore $F\cap L=\emptyset$. An image $\phi(a_i)$ of length n has the form $\phi(a_i)=\alpha x_1x_2\ldots x_{n-2}\beta$, where $\alpha\in F$, $\beta^{-1}\in L$, $x_i\in\Delta^{\pm1}$, and $\phi(a_i)$ is reduced, so $x_1\neq\alpha^{-1}$ and $x_{n-2}\neq\beta^{-1}$. Counting such words is more delicate than in the monoid case, but the asymptotics are similar, due to the following result ([6, Proposition 1]).

▶ **Proposition 22.** *Let A and B be subsets of $\Delta^{\pm1}$. The number of elements of length n in $F(\Delta)$ that do not start with a letter in A and do not end with a letter in B is equal to*

$$f_{A,B}(n)=\frac{(2m-|A|)(2m-|B|)(2m-1)^{n-1}+xm+(-1)^n(|A||B|-ym)}{2m},$$

where $x=|A\cap B|-|A^{-1}\cap B|, y=|A\cap B|+|A^{-1}\cap B|$, and $m=|\Delta|$.

Let $A = \{\alpha^{-1}\}$ and $B = \{\beta^{-1}\}$; then since the number of possible $\phi(a_i)$ of length $\leqslant n$ is equal to the number of reduced words of length $\leqslant n-2$ starting with a letter different from α^{-1} and ending with a letter different from β^{-1}, this number is $\sum_{1\leqslant j\leqslant n-2} f_{A,B}(j)$. Since $|A| = |B| = 1$, $f_{A,B}(j)$ is asymptotically $(2m-1)^j$, and the number of possible $\phi(a_i)$ is $\sim c_1(2m-1)^{n-2}$, where c_1 is a constant depending on m. Thus for fixed sets F and L the number of immersions ϕ with images in the ball of radius n is $\sim (c_1(2m-1)^{n-2})^k$. Since there are only finitely many choices for sets F and L of first and last letters, respectively, and the number of k-tuples of elements in $F(\Delta)$ of length $\leqslant n$ is $\sim (c_2(2m-1)^n)^k$ for some constant c_2, the number of immersions over the total number of maps $F(\Sigma) \mapsto F(\Delta)$ is $\sim \frac{(c_1(2m-1)^{n-2})^k}{(c_2(2m-1)^n)^k}$; so as $n \mapsto \infty$, this ratio is a positive constant depending on k and m.

References

1 Gilbert Baumslag, Alexei G. Myasnikov, and Vladimir Shpilrain. Open problems in combinatorial group theory. Second edition. In *Combinatorial and geometric group theory (New York, 2000/Hoboken, NJ, 2001)*, volume 296 of *Contemp. Math.*, pages 1–38. Amer. Math. Soc., Providence, RI, 2002. doi:10.1090/conm/296/05067.

2 George M. Bergman. Supports of derivations, free factorizations, and ranks of fixed subgroups in free groups. *Trans. Amer. Math. Soc.*, 351(4):1531–1550, 1999. doi:10.1090/S0002-9947-99-02087-5.

3 Mladen Bestvina and Michael Handel. Train tracks and automorphisms of free groups. *Ann. of Math. (2)*, 135(1):1–51, 1992. doi:10.2307/2946562.

4 Meera Blattner and Tom Head. Automata that recognize intersections of free submonoids. *Information and Control*, 35(3):173–176, 1977.

5 Laura Ciobanu, Armando Martino, and Enric Ventura. The generic Hanna Neumann Conjecture and Post Correspondence Problem, 2008. URL: http://www-eupm.upc.es/~ventura/ventura/files/31t.pdf.

6 Laura Ciobanu and Sasa Radomirovic. Restricted walks in regular trees. *Electronic J. of Combinatorics*, 13(R93), 2006. doi:10.37236/1119.

7 Warren Dicks and Enric Ventura. *The group fixed by a family of injective endomorphisms of a free group*, volume 195 of *Contemporary Mathematics*. American Mathematical Society, Providence, RI, 1996. doi:10.1090/conm/195.

8 Volker Diekert, Olga Kharlampovich, Markus Lohrey, and Alexei Myasnikov. Algorithmic problems in group theory. *Dagstuhl seminar report 19131*, 2019. URL: http://drops.dagstuhl.de/opus/volltexte/2019/11293/pdf/dagrep_v009_i003_p083_19131.pdf.

9 A. Ehrenfeucht, J. Karhumäki, and G. Rozenberg. The (generalized) Post correspondence problem with lists consisting of two words is decidable. *Theoret. Comput. Sci.*, 21(2):119–144, 1982. doi:10.1016/0304-3975(89)90080-7.

10 A. Ehrenfeucht and G. Rozenberg. Elementary homomorphisms and a solution of the D0L sequence equivalence problem. *Theoret. Comput. Sci.*, 7(2):169–183, 1978. doi:10.1016/0304-3975(78)90047-6.

11 Vesa Halava, Mika Hirvensalo, and Ronald de Wolf. Marked PCP is decidable. *Theoret. Comput. Sci.*, 255(1-2):193–204, 2001. doi:10.1016/S0304-3975(99)00163-2.

12 Tero Harju and Juhani Karhumäki. Morphisms. In *Handbook of formal languages, Vol. 1*, pages 439–510. Springer, Berlin, 1997.

13 Štěpán Holub. Binary equality sets are generated by two words. *J. Algebra*, 259(1):1–42, 2003. doi:10.1016/S0021-8693(02)00534-3.

14 W. Imrich and E. C. Turner. Endomorphisms of free groups and their fixed points. *Math. Proc. Cambridge Philos. Soc.*, 105(3):421–422, 1989. doi:10.1017/S0305004100077781.

15 Ilya Kapovich. Mapping tori of endomorphisms of free groups. *Comm. Algebra*, 28(6):2895–2917, 2000. doi:10.1080/00927870008826999.

16 Ilya Kapovich and Alexei Myasnikov. Stallings foldings and subgroups of free groups. *J. Algebra*, 248(2):608–668, 2002. doi:10.1006/jabr.2001.9033.

17 Juhani Karhumäki. A note on intersections of free submonoids of a free monoid. *Semigroup Forum*, 29(1-2):183–205, 1984. doi:10.1007/BF02573324.

18 Juhani Karhumäki and Aleksi Saarela. Noneffective regularity of equality languages and bounded delay morphisms. *Discrete Math. Theor. Comput. Sci.*, 12(4):9–17, 2010.

19 Alexei Myasnikov, Andrey Nikolaev, and Alexander Ushakov. The Post correspondence problem in groups. *J. Group Theory*, 17(6):991–1008, 2014. doi:10.1515/jgth-2014-0022.

20 Turlough Neary. Undecidability in binary tag systems and the post correspondence problem for five pairs of words. In *32nd International Symposium on Theoretical Aspects of Computer Science*, volume 30 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 649–661. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2015.

21 Emil L. Post. A variant of a recursively unsolvable problem. *Bull. Amer. Math. Soc.*, 52:264–268, 1946. doi:10.1090/S0002-9904-1946-08555-9.

22 John R. Stallings. Graphical theory of automorphisms of free groups. In *Combinatorial group theory and topology (Alta, Utah, 1984)*, volume 111 of *Ann. of Math. Stud.*, pages 79–105. Princeton Univ. Press, Princeton, NJ, 1987.

23 Bret Tilson. The intersection of free submonoids of a free monoid is free. *Semigroup Forum*, 4:345–350, 1972. doi:10.1007/BF02570808.

24 E. Ventura. Fixed subgroups in free groups: a survey. In *Combinatorial and geometric group theory (New York, 2000/Hoboken, NJ, 2001)*, volume 296 of *Contemp. Math.*, pages 231–255. Amer. Math. Soc., Providence, RI, 2002. doi:10.1090/conm/296/05077.